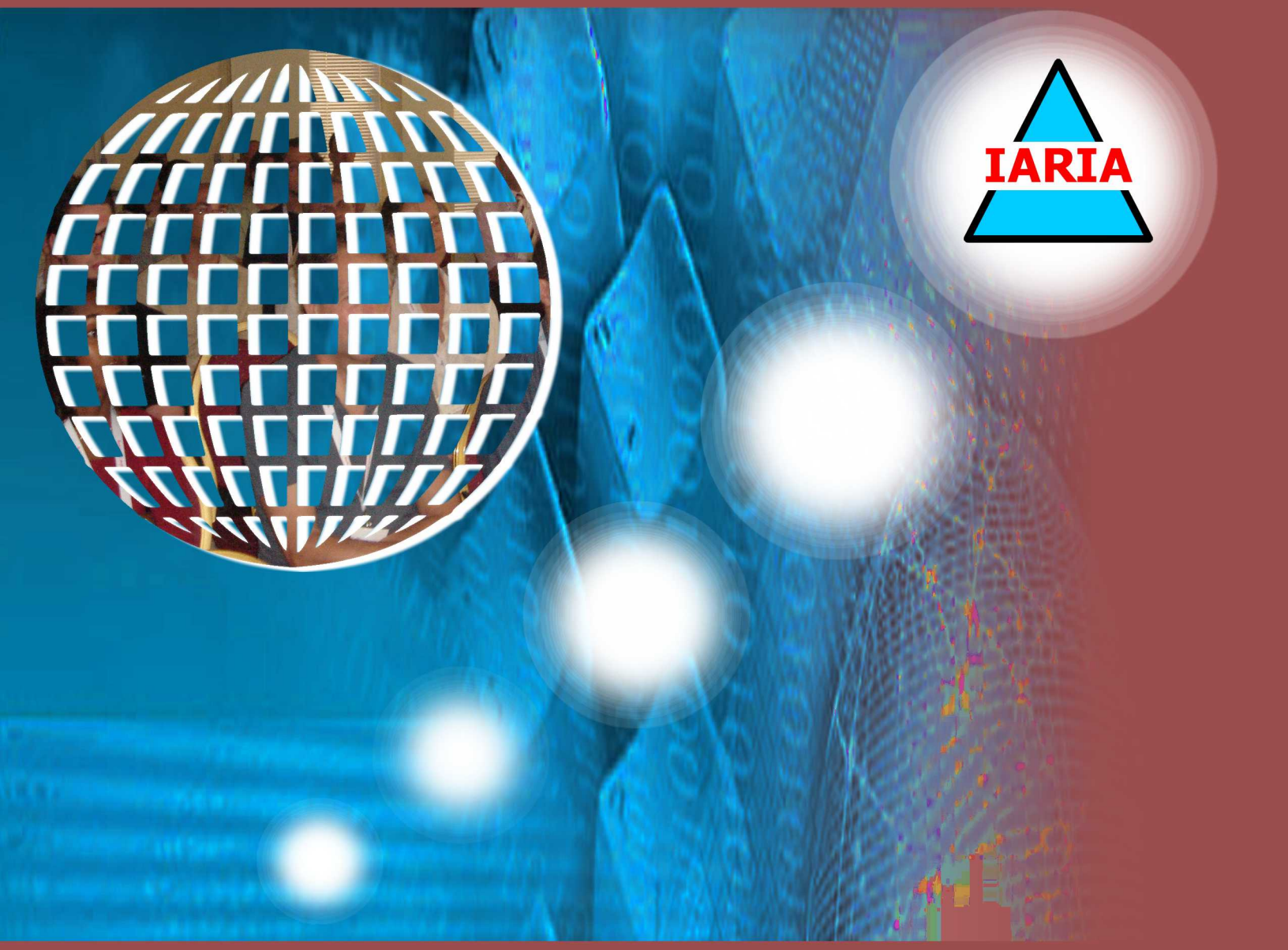
International Journal on Advances in Security



2009 vol. 2 nr. 1

The *International Journal On Advances in Security* is Published by IARIA.
ISSN: 1942-2636
journals site: http://www.iariajournals.org
contact: petre@iaria.org

Sponsored by IARIA
www.iaria.org

International Journal On Advances in Security
Volume 2, Number 1, 2009

- Aljosa Pasic, ATOS Origin, Spain
- Vladimir Stantchev, Berlin Institute of Technology, Germany
- Michiaki Tatsubori, IBM Research - Tokyo Research Laboratory, Japan
- Ian Troxel, SEAKR Engineering, Inc., USA
- Marco Vieira, University of Coimbra, Portugal
- Hans P. Zima, Jet Propulsion Laboratory/California Institute of Technology - Pasadena, USA // University of Vienna, Austria

Security in Internet

- Evangelos Kranakis, Carleton University, Canada
- Clement Leung, Victoria University - Melbourne, Australia
- Sjouke Mauw, University of Luxembourg, Luxembourg
- Yong Man Ro, Information and Communication University - Daejon, South Korea

International Journal On Advances in Security
Volume 2, Number 1, 2009

Foreword

The first 2009 number of the International Journal On Advances in Security compiles a set of papers with major enhancements based on previously awarded publications. It brings together a set of articles that share a common link to security. For this issue, nine contributions have been selected.

In the first article, Y. Zafar and D. Har propose a countermeasure catered towards side channel attacks in FPGA implementations. The major danger in such attacks is the extraction of information from algorithmically secure systems. The proposed countermeasure hardens the FPGAs with respect to this attack.

Miroslav Sveda and Radimir Vrba follow with a presentation of principles of meta-design support for safe and secure embedded system networking. A framework is presented both from the builder's perspective as well as the user's perspective.

As a third article, Wolfgang Leister et al. delve into security and authentication aspects of wireless patient monitoring systems. The proposed framework is based on MPEG-21 and can withstand a variety of threats. A usable test bed for the proposal is also presented.

The fourth article by Peter Merz et al. deals with reputations systems as it applies to a super-peer desktop grid. In peer to peer networks where resources are shared without a tight control, users can be greedy and in the end degrade the overall environment.

In the next article, Jani Suomalainen et al. propose a secure incentive mechanism for peer to peer environments for mobile devices. As these mobile devices begin to host peer to peer applications, new concerns arise given their inherent lack of resources. The proposal in this work includes a centralized incentive mechanism which can analyze and classify threats, while at the same time ensure security.

Jiri Schafer et al. present currently known threats in decentralized peer to peer networks. In the second half of the article, a new methodology of automatic download and detection for threats is presented.

The seventh article by Elizabeth Papadopoulou et al. delve into handling user privacy in the context of pervasive systems. A user may have requests for different services, and in turn, each service will need a different subset of private data from the user in order to complete requests. An approach is described on how a user can have multiple virtual identities with different privacy settings and hence be able to use a different identity depending on the requested service.

The following article by Juhani Eronen et al. provides a case study of antivirus software. The conclusions are very relevant to critical systems which can be compromised due to risks present in the antivirus software itself.

Michael Hartle et al. conclude this issue of the journal considering data format and its relevance to security. A model is presented which can be used to define arbitrary data formats.

We hope that the contents of this journal will add to your understanding of security, and that you will be inspired to contribute to IARIA's conferences that include topics relevant to this journal.

Stefanos Gritzalis, Editor-in-Chief
Petre Dini, IARIA Advisory Committees Board Chair

International Journal On Advances in Security
Volume 2, Number 1, 2009

CONTENTS

A Novel Countermeasure to Resist Side Channel Attacks on FPGA Implementations

Y. Zafar and D. Har
Dept. of Information & Communication, Gwangju Institute of Science & Technology,
1 Oryong–dong, Buk–gu, Gwangju 500–712, Rep. of Korea
e-mail: yzafar@gist.ac.kr , hardon@gist.ac.kr

Abstract—**Side Channel Attacks (SCAs) have proven to be very effective in extracting information from algorithmically secure systems. Since the earliest reports of attacks exploiting side channels such as power consumption, timing behavior and electromagnetic radiation etc., the countermeasures to resist such attacks have also been proposed. A variety of countermeasures resisting SCAs have been presented that continue to fade away as resistant attack techniques are developed. Field Programmable Gate Arrays (FPGAs) were originally thought to be resistant to such attacks because of some inherent characteristics. Later, they were also found to leak information over the side channels. In this article a novel countermeasure is presented that hardens an FPGA based system with cipher embodiment, against SCAs. The proposed methodology of embedding single inverter ring oscillators within the synchronous cores helps improve immunity against electromagnetic, fault and glitch attacks, while the introduction of frequency hopping by randomly varying frequency driving the cipher hardens the system against power and timing attacks. The incorporated countermeasure enhances the immunity of FPGA based implementation against multiple types of SCAs without adversely affecting cost or performance.**

Keywords-Side Channel Attacks; countermeasures; FPGAs; frequency hopping

I. INTRODUCTION

The reconfigurable nature of FPGAs offers major advantages for cryptographic applications because of ever-changing standards and the high Application Specific Integrated Circuit (ASIC) costs. However, like other Complementary metal–oxide–semiconductor (CMOS) based custom ASICs, commercially available FPGAs based on similar technology also leak information over the side channel, requiring incorporation of countermeasures to resist SCAs [1–2]. When incorporating these countermeasures the biggest challenge is to keep design cost from escalating and performance from degrading [3]. A complex countermeasure that increases the design real-estate considerably is therefore detested and the one enhancing immunity of the system against multiple kinds of SCAs is venerated.

SCAs against ASIC implementations and their countermeasures have extensively been reported in literature. Introducing SCA countermeasures to FPGA implementations is a relatively recent trend [3–5]. A globally asynchronous locally synchronous ASIC with Advanced Encryption Standard (AES) cipher reported by Gurkaynak et al. [6], combines operation reordering and unpredictable latencies with asynchronous clock domains and self varying clock cycles to counter SCAs. However, it contains local clock generators with delay control and other asynchronous elements customized at switch level, incompatible with FPGA implementation. In this article an FPGA based multi-clock system with embedded single inverter ring oscillators (SIROs), embodying 128-bit AES cipher is presented that employs frequency hopping to enhance immunity against SCAs. FPGA implementation of SIROs driving micropipelined and synchronous architectures has already been reported [7–8].

The proposed design consists of different synchronous units triggered by their local SIRO based clock sources that reside within them. A special circuit called the hopper, with its local SIRO based clock source randomly selects the frequency that encrypts each new 128-bit data frame. We refer to this random frequency selection, as frequency hopping.

The remaining sections of this paper are organized as follows. Cryptanalysis and the side channel attacks are reviewed in Section II. Vulnerability of FPGAs against SCAs is also discussed in the same section. Possible countermeasures against the SCAs are summarized in section III. FPGA compliance of SIRO–based designs is high-lighted in Section IV. Section V describes the counter mode of Rijndael AES contained within the system. An overview of the experimental setup and system architecture is presented in Section VI. Frequency hopping incorporated through Hopper circuit, thwarting SCAs is discussed in Section VII and the conclusion is drawn in Section VIII by attributing enhanced SCA immunity to the proposed system.

II. CRYPTANALYSIS AND THE SCAS

Cryptanalysis (from the Greek *kryptós*, "hidden", and *analýein*, "to loosen" or "to untie") is the study of methods for obtaining the meaning of encrypted information, without access to the secret information which is normally required

to do so [9]. The first known recorded explanation of cryptanalysis was given by 9th-century Arab polymath, Abu-Yusuf Al-Kindi where he presented the method of frequency analysis to decipher encrypted messages [10].

For a cryptographic system to remain secure, the secret keys used in performing the required security services must not be revealed. In cryptanalysis, typically the protected keys are revealed by analyzing the cryptographic algorithm. The task is time consuming and cumbersome. Another technique used in cryptanalysis is based on the fact that cryptographic algorithms are always implemented in software or hardware on physical devices which interact with and are influenced by their environments. These physical interactions can be monitored by adversaries to extract protected information. This type of information is called side-channel information, and the attacks exploiting side-channel information are called side-channel attacks. The whole idea of SCA attack is to look at the way cryptographic algorithm is implemented, rather than at the algorithm itself [11].

The first official information related to SCA attack dates back to the year 1965, when P. Wright reported in [12] that MI5, the British intelligence agency, was trying to break a cipher used by the Egyptian Embassy in London. However, a major contribution in this field was made by Paul Kocher in 1996, when he successfully launched an SCA called the timing attack against secure implementations. In his report, Kocher had measured the amount of time required for private key operations, to experimentally reveal secret information [13]. Another kind of SCA called the fault attack associated with incorporating faults in the operation of cryptosystems was first reported in 1997 by Boneh et al. [14]. Hardware processing done at abnormally high or low frequencies or under extreme temperature conditions may induce faults that an attacker can observe to evaluate hidden information.

SCAs observing power consumption were soon discovered to be far more effective than the ones observing other side channels. In 1999, Paul Kocher introduced Simple Power Analysis (SPA) and Differential Power Analysis (DPA) [15]. Power analysis attacks exploit the fact that power consumption of a CMOS circuit is data dependent based on the switching activity governed by (1).

$$P_D = C_L (V_{DD})^2 P_{0\rightarrow1} f \quad (1)$$

where P_D is the dynamic power consumed by a single CMOS gate; C_L is the gate load capacitance; V_{DD} is the supply voltage; $P_{0\rightarrow1}$ is the probability of a $0\rightarrow1$ output transition making the measurement data dependent and f is the clock frequency [16]. For DPA, the attacker measures the power consumption of the device while it processes a large set of cryptographic operations. As opposed to SPA, where the information is extracted directly from the power measurements, DPA uses statistical techniques to compare measured values to a set of estimated values.

In 2002 Agrawal et al., reported some of the first successful attacks based on the analysis of electromagnetic emissions [17].

Over the past decade perceptions regarding FPGAs have changed. Instead of being viewed as devices used only for prototyping, they are now seen as end-products containing low-cost, reconfigurable hardware. The status of Hardware Descriptive Languages (HDLs) has also changed from tools used for hardware simulation to synthesizable mediums. This transition has resulted in the migration of ASIC-based designs to the FPGA platforms, especially in the areas like cryptography that require frequent upgrade of end-products.

FPGAs have also been looked upon favorably for cryptographic applications due to certain inherent features, believed to provide practical security against SCAs. These features include the intrinsic parallel computing capability of FPGAs that leads to algorithmic noise complicating measurements related to a specific event and the ability of different bits in the observation area to contribute differently to the overall power consumption. This variation is due to the pre-laid out structure of interconnects between computational elements that results in different effective capacitances [18]. However, this notion of inherent FPGA immunity was negated in 2003 by Örs et al., when they successfully mounted a power analysis attack against an FPGA based elliptic curve cryptographic processor [19]. Attacks exploiting the electromagnetic leakage of FPGAs were first reported by Carlier in 2004 [20].

III. SCA COUNTERMEASURES

Countermeasures typically enhance the SCA immunity of a cryptosystems by either abating or adulterating leakages. Noise addition, data randomization, duplication & Boolean masking and implementation using dynamic & differential logic styles are some of the techniques involved in incorporating countermeasures [5]. The purpose is it to resist leakage measurements or to complicate the comparison of estimated vs. the measured values during statistical analysis based attacks.

Introduction of countermeasures to enhance SCA immunity in hardware often deals with transistor-level changes of the logic that are not easily applicable to FPGAs without manufacturers' support [3]. We therefore, observe a general trend in evolution of SCAs and related countermeasures targeting ASICs in the beginning and later being modified for FPGA compliance.

Different methodologies have been proposed for each category, and new methodologies continue to spring up as countermeasures cede to new kinds of attacks, immune to the outdated versions.

IV. FPGA RESIDENT SIRO

Embedded SIRO based clock sources have already been reported to drive co-existing synchronous systems in FPGAs [8]. The implementation of a ring oscillator in FPGA is simple and consists of a single inverting stage comprising of a 2–input NAND gate, the output of which is fed back to one of its inputs while a logic low on the other gate input is used to pause the oscillator. Fig. 1(a) shows a simple SIRO circuit. The 2-input NAND based SIRO implementation in FPGA is metaphorical, for the FPGA–centric circuit uses a logic element (LE) in the combinatorial mode to implement the functionality of gate, as shown in Fig. 1(b). Fast interconnects (direct / local) establish the feedback ring. Simplicity is necessary in case of FPGA based SIRO, for it to be technology independent as well as power efficient [7–8]. A change in environmental conditions affects it in the same way as it affects the other components of the co-existing system. Therefore, the SIRO based clock source and the circuit driven by it adapt to physical conditions in a similar manner [8].

In this article, an FPGA based multi-clock system with cipher embodiment is proposed. It consists of different synchronous units triggered by their local SIRO based clock sources that reside within them. A special circuit called the hopper, with its local SIRO randomly selects the frequency that drives the cipher. The system is hardened against electromagnetic, fault and glitch attacks, due to presence of multiple SIROs that are adaptive, unsynchronized and physically invisible (on any external pin) outside the FPGA. On the other hand, timing and power attacks are thwarted by the proposed frequency hopping scheme.

V. ADVANCED ENCRYPTION STANDARD

A block cipher developed by two Belgian cryptographers, Joan Daemen and Vincent Rijmen, submitted under the name 'Rijndael' was formally adopted as Advanced Encryption Standard (AES) by National Institute of Standards and Technology (NIST), USA on 6th of December, 2001 [21]. AES is a block cipher where several different transformations, such as 'Sub Bytes' (substitution of bytes by alternate values from S-Box), 'Shift Rows', 'Mix Columns' and 'Add Round Key' are iterated in so called rounds. Additional modes have also been proposed for AES to enhance its ciphering capabilities. In 2001, Dworkin recommended some modes for AES including the counter mode (CTR) guaranteeing change in ciphered value even when plain data and encryption key remain unchanged [22]. In counter mode AES (AES–CTR) the cipher key actually encrypts an initialization vector (IV) instead of plain data, as shown in Fig. 2. The resulting ciphered IV is the mask that is bitwise XORed with plain data in order to encrypt it. The ciphered data is XORed with the mask once again on the decryption side to regenerate plain data. In the presented model, a 128-bit IV initialized by Linear Feedback Shift Register (LFSR), consisting of 64-bit nonce for identification and 64-bit counter field is used. A unique counter value per frame ensures distinct ciphered data even if the two plain data frames are identical.

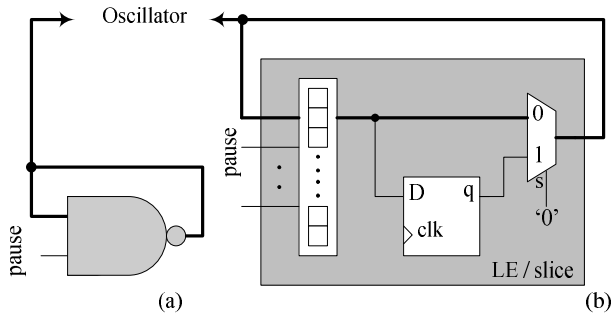


Figure 1. Single inverter ring oscillator (SIRO): (a) 2-input NAND based gate level design, (b) Logic Element based FPGA implementation.

VI. SYSTEM ARCHITECTURE

Xilinx's XC3S1000 low-cost FPGA was used to implement the system. A dedicated FPGA board with three serial ports was designed. Interestingly, despite being used for the implementation of a multi-clock system, the board does not house any oscillator.

The developed FPGA–based cipher system is plugged into an existing audio communication device as shown in Fig. 3. On the transmitter side, the device receives data from a sensor. After pre-amplification and filtering, the data is digitized by analog-to-digital converter (ADC). The digital data passes through a codec where the bit rate is reduced and the 18-byte frames consisting of two header bytes and 16 data bytes are serially transmitted to a modem at 57.6kbps with 1 stop bit and no parity. While receiving, the modem sends serial data with similar specifications to the codec. The codec after decompressing pushes the digital data to a digital-to-analog converter (DAC) that generates an analog signal which after amplification is sent to the transducer. The cipher system communicates with the codec and modem of the audio communication device serially. It does not require any external oscillator to communicate with these devices as its serial ports are designed to have the same

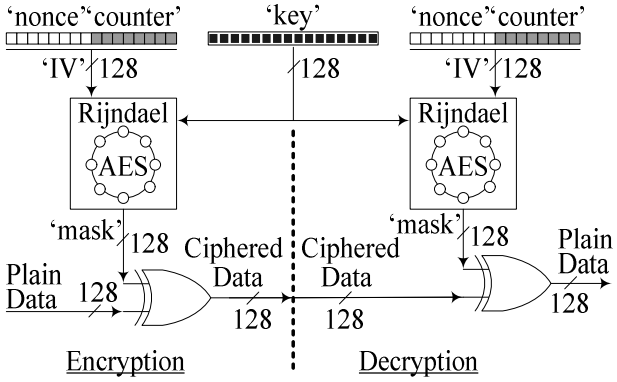


Figure 2. Block diagram of 128-bit AES–CTR

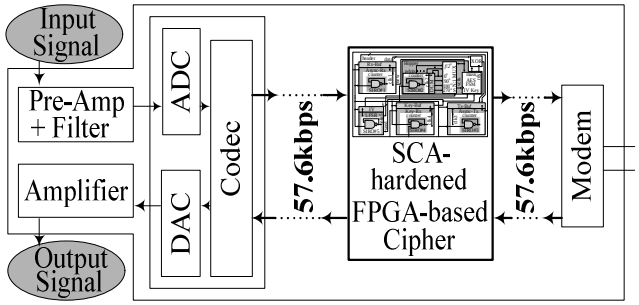


Figure 3. SCA hardened cipher system plugged into pre-existing audio communication device

specifications (57.6kbps, 1 stop bit with no parity) as required by the host communication device.

The transmitter section of the presented multi-clock system as shown in Fig. 4, consists of five units. These units are: the receiver, the cipher, the linear feedback shift register (LFSR), the key receiver and the transmitter. Each unit is independently synchronous, driven by its local SIRO-based clock source that resides within it. The five clock sources local to the five units are independent of each other, none affecting the other or its unit.

Data enters the FPGA-based system serially, in the form of 18-byte frames. Each frame consists of two header bytes used for synchronization and 128-bit (16-byte) data. The data is received by an asynchronous receiver 'Async-Rx' designed to communicate with the host device at its port settings, as explained in the last paragraph. The start bit is a 'Space', while a single stop bit is represented by 'Mark State'. 'Async-Rx' after receiving the data, stores it in a 128-bit buffer 'Rx-buf'. 'Async-Rx' and 'Rx-Buf' constitute the first unit of the system called the receiver unit. The receiver unit has an embedded oscillator 'SIRO#1', the output of which is divided by a counter to generate clock source 'clk1', which is further divided to generate the desired baud rate.

This unit is also responsible for generating the 'new' pulse for two 'clk1' cycles that triggers the cipher unit to prepare 'mask' for XORing required by AES operating in the counter mode. At the rising edge of 'new', the least significant bits of LFSR select the frequency to be used for

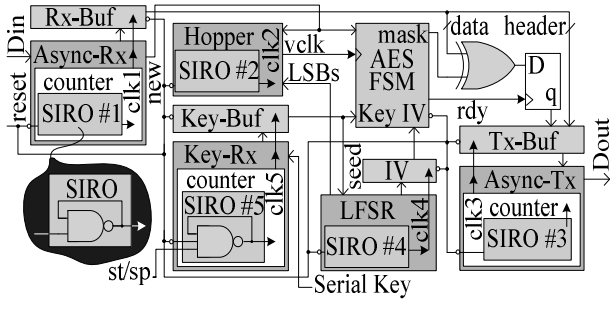


Figure 4. Frequency–hopping based SCA hardened Cipher system

ciphering process. So while the 16-bytes of plain data are buffered into the 'Rx-Buf', cipher unit calculates the 'mask'.

The most important unit of all is the cipher unit. It contains three subsections. The first one is a finite state machine called AES FSM, responsible for the execution of various steps of AES algorithm. This FSM controlling the steps of encryption and key scheduling is driven by an oscillating signal 'vclk' provided by the other subsection of the unit called the hopper circuit. This circuit responsible for the realization of frequency hopping will be discussed at length in Section VII of the article. A 128-bit XOR, performing the operation between 'mask' and the plain data constitutes the third subsection of the cipher unit.

AES FSM performs the Rijndael encryption and key scheduling in parallel and the round keys are generated at runtime. Different steps of the two concurrent processes are distributed over the states of FSM, in such a manner that the round key to be added during a round is generated prior to its addition. NOPs (no operations) are inserted along the two concurrent processes to synchronize the sequences and schedule the use of a single, 4-byte S-box implemented using limited memory resources of FPGA, over the two processes.

Insertion of 'new' initiates the FSM that pulls the 'rdy' signal low and latches the 'key' provided by the key receiver unit and the 'IV' provided by the LFSR unit. This 'rdy' signal does not go high till all the 10 rounds of AES are complete and any change in cipher's inputs 'IV' or 'key', is ignored till 'rdy' is low. Once all the rounds are complete, the 'rdy' signal goes high to inform the unit that the 'mask' is ready to be XORed with plain data. FSM maintains the state keeping 'rdy' signal high, till a positive edge is encountered on 'new' and the FSM is re-initialized.

As AES FSM calculates the 'mask', 'Rx-Buf' is filled with serially received data. Once the 'Rx-Buf is full, it is bitwise XORed with 'mask' and sent to the transmitter unit, driven by 'SIRO#3' generated 'clk3'. The ciphered data is loaded into the 128-bit transmit buffer 'Tx-Buf' of the transmitter section, from where the asynchronous transmitter 'Async-Tx' fetches a byte at a time to transmit it out of the cipher system to the modem, serially, at the system specifications of 57.6kbps, 8 data bits, 1 stop bit and no parity. Baud rate for transmission is generated in this unit by its local clock source 'clk3'.

Another unit is the 128-bit LFSR driven by its locally embedded 'SIRO#4'. Its responsibilities include initialization of IV and providing support to the hopper circuit.

The fifth unit is the key receiver. Its main responsibility is to serially receive encryption 'key' and store it. For this purpose its structure resembles that of the receiver unit. It consists of an asynchronous receiver 'Key-Rx' operating at the same specifications as the other serial ports and a 128-bit buffer called 'Key-Buf' that latches the 'key' once it is fed to the system from outside. This unit is clocked by its local

'SIRO#5' based clock source 'clk5'. However, 'SIRO#5' is different from other ring oscillators of the system as it consists of a three input NAND gate instead of a two input NAND. The additional input is the st/sp signal that stops 'SIRO#5' from oscillating once the 128-bit 'key' is completely buffered into 'Key-Buf'. This 'key' is used by the cipher unit for the encryption. In addition it is also used to initiate the LFSR by acting as its seed.

VII. HOPPER CIRCUIT AND FREQUENCY HOPPING

Fig. 5 shows the hopper circuit of the cipher unit. This circuit is responsible for the realization of frequency hopping presented in this article that hardens the FPGA-based cipher against SCAs. Hopper circuit embodies 'SIRO#2', the oscillating output of which is divided by an n-bit counter to generate n sub-frequencies. Four of these sub-frequency signals are fed to the Digital Clock Managers (DCMs) of the FPGA device. Each of the DCMs creates additional four frequency signals with phase shifts of 0°, 90°, 180° and 270° w.r.t. the input signal. One of these N number of frequencies, at the rising edge of signal 'new' generated by the receiver unit, is selected as 'vclk', by an N:1 multiplexer. The select lines of the multiplexer are connected to the k least significant bits (LSBs) of LFSR, where $N = 2^k$. As discussed in the previous section, 'vclk' is the signal that drives AES FSM of the cipher unit and positive transition on 'new' forces a new frequency to ride this signal. It is to say that while a 16-byte frame of plain data is being buffered in, 'IV' is encrypted by 'key' to generate a 'mask' for XORing, as a randomly selected

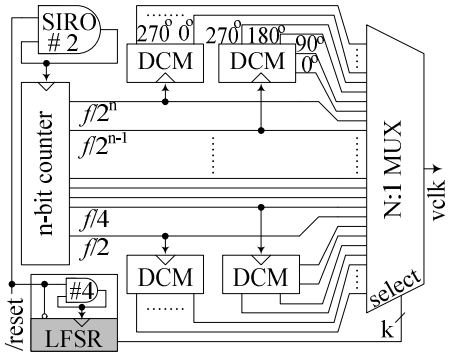


Figure 5. Hopper circuit

frequency drives the AES FSM. The FSM creates a new 'mask' with incremented IV as a new randomly selected frequency drives it for the next data frame. One can observe 'vclk' driving AES FSM, hopping to a different frequency after every 'new' pulse in Fig. 6(a). We refer to this cipher frequency randomization as the process of incorporating frequency hopping in communication security, a characteristic generally associated with transmission security.

Fig. 6 shows the operation of AES FSM under the influence of frequency hopping. The post-layout simulation results for AES FSM driven by 'vclk' are presented in Fig. 6(a). Signal 'new' pulls 'rdy' low to initiate the execution of Rijndael steps. The 'key' encrypts 'IV' to generate the 'mask', upon which 'rdy' goes high and stays high till a positive transition on 'new' is encountered. Each 'new' pulse selects a different frequency for 'vclk'. The serial data

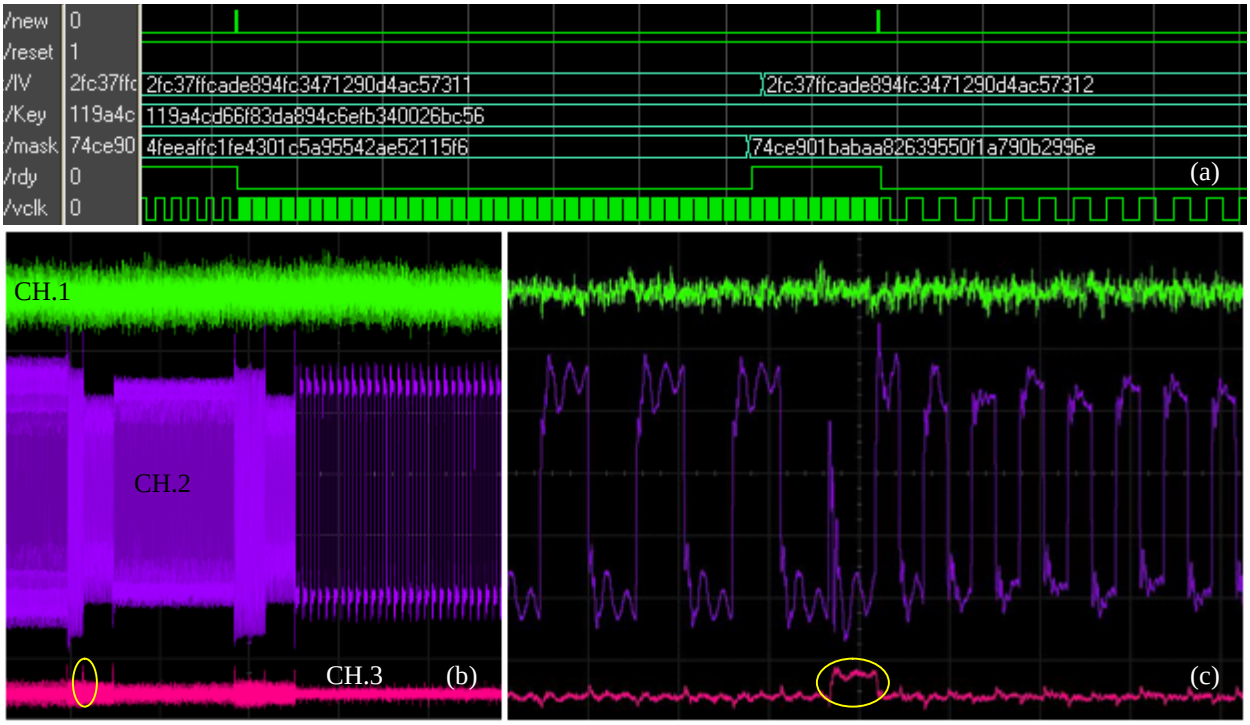


Figure 6. Frequency hopping based AES–CTR: (a) Post–layout simulation high-lighting frequency hopping [23], (b) Current measurements upon frequency hopping captured on oscilloscope, (c) zoomed view of oscilloscope capture

streams in and out of our AES–CTR based system at a constant rate of 57,600bps. If there is an error free transmission of data, then each frame consisting of 16 data bytes, 2 header bytes, one start and one stop bit per byte, streams in and out of the system in approximately 3.125 mSec. Whereas, the experimental results demonstrate that minimum amount of time with highest self-generated frequency selected to perform ciphering, creates the 'mask' in 2.052635 µSec. Therefore, any frequency can be randomly selected by hopper to drive AES FSM that does not interrupt the continuous flow of data. The point to be noted here is that the cipher itself executes a particular step of algorithm (like the Sub Byte) under observation for power measurements, at random intervals w.r.t. the initialization of encryption process because of the randomly selected frequency driving AES FSM for each frame.

Fig. 6 (b & c) present the oscilloscope screen captures during physical measurements used for analysis. Agilent Infiniium DS08104A Oscilloscope was used for these measurements. Channel no. 1 (CH.1) displays the current readings acquired using Agilent 1147A current probe with the main power supply of the FPGA board. Channels no. 2 and 3 (CH.2 & CH.3) display the status of signals 'vclk' and 'new'. In Fig. 6(b) frequency of 'vclk' is observed to change upon transitions on 'new'. However, this frequency change does not affect the current readings. Fig. 6(c) is the zoomed view of a segment of capture presented in Fig. 6(b). A transition on 'new' changes the frequency of signal 'vclk', significantly, but the current patterns used by the attacker, do not high-lighting this change, explicitly. Therefore, the activity of a particular section of the algorithm like the s-box under observation cannot easily be identified / distinguished on the time scale, especially when repeated measurements of the same instance are required to deduce the exact information, as in case of DPA. Monitored activity is distributed randomly on the time axis because of this frequency drift, and the physical current measurements do not convey enough information to identify this activity at specific time. This makes very complicated the synchronization of the DPA and SPA signatures, thwarting power analysis based on statistical techniques. The result is what may be described as "smearing the peaks of differential trace due to de-synchronization effect". The attacker is therefore, forced to collect more data, and the system thus exhibits itself as the one with enhanced immunity against power analysis attacks.

VIII. CONCLUSION

The proposed system, because of its architectural features inherently resists fault and glitch attacks. The clock signals do not present themselves externally. Therefore, the extraction of timing information or insertion of glitches is more cumbersome. Manipulation of frequencies for fault injection is also difficult because of the adaptive nature of SIROs [8]. Multiple and unsynchronized oscillating signals immunize the system against Electromagnetic attacks. Furthermore, random selection of oscillating frequency at run time, to act as a clock source driving AES FSM means unpredictable clock, that complicates the synchronization of the power signatures. When using differential power analysis, the exact time of a particular operation under observation varies randomly due to this characteristic referred to as frequency hopping in the article. It can therefore safely be concluded that the presented methodology of introducing frequency hopping to embedded SIRO driven FPGA implementations enhances their immunity against multiple types of SCAs without performance or cost trade-offs.

ACKNOWLEDGMENT

This work was supported in part by the Center for Distributed Sensor Network at GIST and in part by the Regional Innovation Program funded from Ministry of Knowledge Economy.

REFERENCES

[1] F.-X. Standaert, "Secure and efficient use of reconfigurable hardware devices in symmetric cryptography", Ph.D. Thesis, UCL Crypto Group, Universite' catholique de Louvain, Belgium, June 2004.

[2] E. Peeters, F.-X. Standaert, N. Donckers, and J.-J. Quisquater, "Improved higher-order side-channel attacks with FPGA experiments", CHES 2005, LNCS 3659, 2005, pp. 309-323.

[3] T. Wollinger, J. Guajardo, and C. Paar, "Cryptography on FPGAs: State of the art implementations and attacks", ACM Transactions on Embedded Computing Systems, Aug. 2004, vol. 3, no. 3, pp. 534–574.

[4] F. -X. Standaert, F. Mace, E. Peeters, and J. -J. Quisquater, "Updates on the security of FPGAs against power analysis attacks", ARC 2006, LNCS 3895, 2006, pp. 335-346.

[5] F.-X. Standaert, E. Peeters, G. Rouvroy, and J.-J. Quisquater, "An overview of power analysis attacks against Field Programmable Gate Arrays", Proceedings of the IEEE, vol. 94, no. 2, Feb. 2006, pp. 383-394.

[6] F. Gurkaynak, S. Oetiker, H. Kaeslin, N. Felber, and W. Fichtner, "Improving DPA security by using globally-asynchronous locally-synchronous systems", Proceedings of ESSCIRC 2005**,** Sep. 2005, pp.407- 410.

[7] Y. Zafar and M. M. Ahmad, "A novel FPGA compliant micropipeline", IEEE Transactions on Circuits & Systems II, Sep. 2005, 52, (9), pp. 611–615.

[8] Y. Zafar, "FPGA-compliant micropipeline based asynchronous systems", PhD thesis, M.A. Jinnah Univ., 2005. http://eprints.hec.gov.pk/510/1/383.html.htm

[9] M. J. Aqel, Z. A. Alqadi, and I. M. El Emary, "Analysis of stream cipher security algorithm", Journal of Information and Computing Science, 2007, vol. 2, no. 4, pp. 288-298.

[10] I. A. Al-Kadi , "The origins of cryptology: The Arab contributions", Cryptologia, vol. 16, no. 2, April 1992, pp. 97–126.

[11] Y. Zhou and D. Feng, "Side-channel attacks: Ten years after its publication and the impacts on cryptographic module security testing", NIST Physical Security Testing Workshop, Hawaii, USA, Sep. 2005. Cryptology ePrint Archive, Report 2005/388, 2005, http://eprint.iacr.org/

[12] P. Wright, "Spy Catcher: The candid autobiography of a senior intelligence officer", Viking Press (Penguin Group), 1987.

[13] P. C. Kocher, “Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems”, LNCS 1109, 1996, pp. 104-113.

[14] D. Boneh, R. A. Demillo, and R. J. Lipton, “On the importance of checking protocols for faults”, Advances in Cryptology–Eurocrypt '97, LNCS 1233, Springer, Berlin, 1997, pp. 37-51.

[15] P. Kocher, J. Jaffe, and B. Jun, “Differential Power Analysis”, CRYPTO 1999, LNCS 1666, Aug. 1999, pp. 388-397.

[16] J. M. Rabaey, “Digital integrated circuits”, Prentice Hall International, 1996.

[17] D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi, “The EM side-channel(s)”, CHES 2002, LNCS 2523, August 2002, pp. 29-45

[18] L. Shang, A. Kaviani and K. Bathala, “Dynamic power consumption in Virtex-II FPGA family”, ACM/SIGDA tenth international symposium on Field-programmable gate arrays (FPGA'2002), Feb. 2002, pp. 157-164.

[19] S. B. Örs, E. Oswald, and B. Preneel, “Power-analysis attacks on an FPGA—First experimental results”, CHES 2003. LNCS 2279, 2003, pp. 35–50.

[20] V. Carlier, H. Chabanne, E. Dottax, and H. Pelletier, “Electromagnetic side channels of an FPGA implementation of AES”, IACR E-print Archive 2004/145, 2004. http://eprint.iacr.org

[21] National Institute of Standards and Technology, “Advanced Encryption Standard”, FIPS PUB 197, 2001. http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[22] M. Dworkin, “Recommendation for block cipher modes of operation: Methods and techniques”, NIST Special Publication 800–38A, 2001.

[23] Y. Zafar and D. Har, “A novel countermeasure enhancing side channel immunity in FPGAs”, Proceedings of IARIA International Conference on Advances in Electronics and Micro-electronics (ENICS 2008), Sep.2008, pp. 132-137.

Yousaf Zafar received a B.S. degree in Electrical Engineering from the University of Wyoming, USA in 1993. His MS in Electronics Engineering from GIK Institute of Engineering Sciences & Technology, Pakistan led to his PhD in the same field from M.A. Jinnah University, Pakistan in 2005. He is currently working as a Postdoctoral Research Associate at the Department of Information and Communication, Gwangju Institute of Science and Technology, Rep. of Korea. His research interests include Reconfigurable Asynchronous Processing and hardware realization of complex mathematical algorithms related to communication and information security.

Dongsoo Har received the B.S. and M.S. degrees in Electronics Engineering from Seoul National University, Rep. of Korea in 1986 and 1988, respectively. He finished his PhD in Electrical Engineering at Polytechnic University at Brooklyn, NY, USA in 1997. He is currently an Associate Professor at the Department of Information and Communication, Gwangju Institute of Science and Technology, Rep. of Korea. His research interests include multimedia processing IP design and implementation, as well as design and implementation of Low-power embedded systems.

Meta-Design with Safe and Secure Embedded System Networking

Miroslav Sveda
Faculty of Information Technology
Brno University of Technology
Brno, Czech Republic
sveda@fit.vutbr.cz

Radimir Vrba
Faculty of Electrical Engineering and Communication
Brno University of Technology
Brno, Czech Republic
vrbar@feec.vutbr.cz

Abstract—**The paper presents several validated principles of a meta-design support for end-user development of safe and secure embedded system networking. The devised approach offers a reusable framework for Internet-based embedded system applications. Such resulting framework provides a flexible development environment kernel that can be adapted for various safety/security critical embedded system application domains. It stems from the IEEE 1451.1 smart transducer interface standard that provides an object-based networking model mediating efficient and unified access to distributed components through both wired and wireless networks. The paper discusses this framework not only from the viewpoint of framework builders, but also end-user developers. In this context, it demonstrates how to use that approach for a safety and security-critical application with Internet and ZigBee.**

Keywords-embedded system application; networking; security; safety; meta-design

I. INTRODUCTION

Not surprisingly, meta-design relates to design in the similar way as meta-modeling relates to modeling. But, while modeling and meta-modeling are identical activities with the only difference of interpretation, designing and meta-designing are targeted differently. Model, which is the object of modeling, remains an abstract notion in the similar sense as meta-model with the only exception of abstraction level. On the contrary, objects of design and meta-design differ: in the former case the process produces an artifact, in the letter case it is design of a design process including related development environment.

This paper [1] discusses a deployment of meta-design principles for building up a flexible design framework focused on embedded systems and their components inter-connected by Ethernet-based wired Internet and wireless ZigBee. Necessarily under-designed open source tools and techniques create design spaces for end-user developers. Hence, the paper demonstrates both the use of this framework for implementation of a development environment aimed at Internet-supported smart sensor applications and, concurrently, the utilization of this framework for development of pressure and temperature measurements and safety and security management along gas pipes.

The following section discusses end-user roles and deployment of meta-design principles in an embedded application development process, which is introduced from the end-user viewpoints. After that, the section 3 restates dependability principles characteristic for the considered application domain containing also industrial, safety and security critical applications.

The section 4 deals with related standards and standard structures that create not only solution constraints, but also design support. The subsections discuss subsequently the family of standards IEEE 1451 and, in more detail, standard IEEE 1451.1, which creates a framework foundation for logical design structures, standard ZigBee/IEEE 802.15.4 protocol profile, and TCP/IP-ZigBee interconnection structure.

The next section describes a case study based on a real industrial application that demonstrates utilization and refinement of the introduced design approach aiming at Internet-based, tiered architecture with wireless sensor networks. The applied design framework, which is conceptually based on the IEEE 1451.1 environment, is refined and extended to deal with pressure and temperature measurement and safety and security management along gas pipes. This section presents especially communication, safety and security issues, and resulting structure of the 1451.1 implementation providing efficient and unified access to distributed components through both wire and wireless networks.

The case study discusses the developed framework not only as a meta-design tool from the viewpoint of framework builders, but also from the viewpoint of end-user developers. In this context, it demonstrates how to use that approach for a safety and security-critical application with Internet and ZigBee.

II. META-DESIGN WITH END-USER

Typical system development process involves multiple participant roles [2]. Framework builders create the infrastructure for system components to interact; developers

[1] The current manuscript extends and updates the paper [1].

identify suitable domains and develop new components for them; application assemblers select domain-specific components and assemble them into applications; and end users employ component-based applications to perform daily tasks. Obviously, the fifth role can be included in this pipe-line: end-user developers positioned between application assemblers and end users. These end-user developers are able to tailor applications at runtime because they have both domain expertise and technical know-how. They can interact with applications to adjust individual components, and modify existing assemblies of components to create new functionality. Furthermore, they can play a critical role when component-based systems have to be redesigned for new requirements. End-user development activities can range from customization to component configuration and programming.

Meta-design offers techniques and processes for creating new environments allowing end users to act as designers [3]. In all design processes, two basic stages can be distinguished: design time and use time. At design time, system developers create environments and tools. In conventional design they create complete systems. Because the needs, objectives, and situational contexts of users can only be anticipated at design time, users often find the system unfit for their tasks at use time. Thus, they require adaptation of the existing environment and tools for new applications. Meta-design extends the traditional notion of system development to include users in an ongoing process as co-designers, not only at design time but throughout the entire life-cycle of the development process. Rather than presenting users with closed development systems, meta-design provides them with concepts and tools to extend the system to fit their needs. Hence, meta-design promotes designing the design process.

Evidently, meta-design not only promotes designing the design process, but also can support the end-user development of dependable embedded applications.

III. DEPENDABILITY

Dependability [4] is that property of a system that allows reliance to be justifiably placed on the service it delivers. A failure occurs when the delivered service deviates from the specified service. Dependability measures consist namely of reliability, availability, security, safety and survivability. Availability is the ability to deliver shared service under given conditions for a given time, which means namely elimination of denial-of-service vulnerabilities. Security is the ability to deliver service under given conditions without unauthorized disclosure or alteration of sensitive information. It includes privacy as assurances about disclosure and authenticity of senders and recipients. Security attributes add requirements to detect and avoid intentional faults. Safety is the ability to deliver service under given conditions with no catastrophic affects. Safety attributes add requirements to detect and avoid catastrophic failures.

A failure occurs when the delivered service deviates from the specified service. The failure occurred because the system was erroneous: an error is that part of the system state which is liable to lead to failure. The cause of an error is a fault. Failures can be classified according to consequences upon the environment of the system. While for benign failures the consequences are of the same order of magnitude (e.g. cost) as those of the service delivered in the absence of failure, for malign or catastrophic failures the consequences are not comparable.

A fail-safe system attempts to limit the amount of damage caused by a failure [5]. No attempt is made to satisfy the functional specifications except where necessary to ensure safety. A mishap is an unplanned event (e.g. failure or deliberate violation of maintenance procedures) or series of events that results in damage to or loss of property or equipment. A hazard is a set of conditions within a state from which there is a path to a mishap.

A fail-stop system never performs an erroneous state transformation due to a fault. Instead, the system halts and its state is irretrievably lost. The fail stop model, originally developed for theoretical purposes, appears as a simple and useful conception supporting the implementation of some kinds of fail-safe systems. Since any real solution can only approximate the fail-stop behavior and, moreover, the halted system offers no services for its environment, some fault-avoidance techniques must support all such implementations.

Obviously, design of any safe system requires deploying security to avoid intentional catastrophic failures. And vice versa, system's security can be attacked using a safety flaw. The greater the assurance, the greater the confidence that a security system will protect against threads, with an acceptable level of risk [6].

The above statement deals with trust, which is assured reliance on the character, ability, strength, or truth of someone or something [7]. Trust can be defined as the belief that an entity is capable of acting reliably, dependably, and securely in a particular case. In frame of network systems, trust is a complex subject that should be managed. Trust management entails collecting the information necessary to establish a trust relationship and dynamically monitoring and adjusting the existing trust relationship. Principally, security represents the combination of confidentiality, integrity and availability, and necessarily complements safety in safety-critical industrial applications.

IV. ENVIRONMENT

While preceding sections of this paper review methodology, i.e. meta-design, and required properties, i.e. safety and security, this section discusses architectural means that enable refining the framework into straightforward design and implementation of the networked sensor-based systems.

Embedded system networking concepts stem from hierarchically interconnected networks of various kinds: Internet, local area wire and wireless networks, and wireless

sensor networks. The logical hierarchy is usually based on tiered architectures that bring cost-effectiveness and scalability, and allow adapting straightforwardly to various application requirements.

Actually, networking appears a basic feature of promising embedded systems architectures. Internet access to individual components of distributed embedded systems can be based on both wire and wireless LAN technologies, predominantly on IEEE 802.3 and related Ethernet standards, and on IEEE 802.11b WiFi and associated wireless LAN protocols. Embedded systems and their components can be attached directly to Ethernet with TCP/IP protocol stack, but also indirectly or exclusively through various wired Fieldbuses or wireless technologies such as IEEE 802.11b WiFi, IEEE 802.15.1 Bluetooth, and IEEE 802.15.4 with related ZigBee. Sensor networks bring an important pattern with single base station connected to a wired network on one side and wirelessly to transducers, i.e. sensors and actuators, on the other side. When sensors are clustered, the base station communicates to cluster heads and through them to individual sensors. Moreover, the applications can use also ad-hoc wireless network architectures that enable to extend wireless part of the system network over physical limits and bring new dimension to fulfill application requirements.

The next subsections provide a brief outline of the IEEE 1451 and ZigBee communication architectures aimed at smart sensors and fitting embedded system networks of which smart sensor networks constitute essential subset.

A. IEEE 1451 Architecture

The IEEE 1451 consists of the family of standards for a networked smart transducer interface that include namely (i) a smart transducer software architecture, 1451.1, targeting software-based, network independent, transducer applications, and (ii) a standard digital interface and communication protocol, 1451.2, for accessing the transducer or the group of transducers via a microprocessor modeled by the 1451.1. The next three standards extend the original hard-wired parallel interface 1451.2 to serial multi-drop 1451.3, mixed-mode (i.e. both digital and analog) 1451.4, and wireless 1451.5 interfaces, see Figure 1.

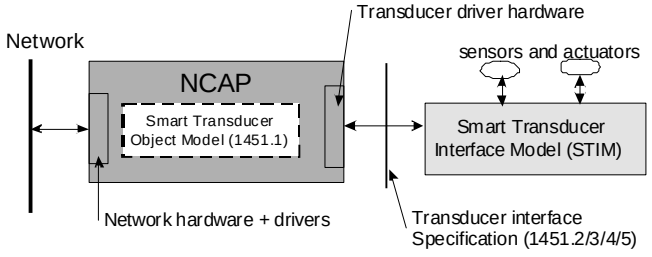


Figure 1. Smart transducer networking

The document 1451.0 complements the above standard set defining the structure of Transducer Electronic Data Sheets (TEDS), and associations between 1451.1 on one side and 1451.2/3/4/5 on the other side with message exchange protocols and command set for transducers.

The IEEE 1451.1 software architecture [8] provides three models of the transducer device environment: (i) the object model of a network capable application processor (NCAP), which is the object-oriented embodiment of a smart networked device; (ii) the data model, which specifies information encoding rules for transmitting information across both local and remote object interfaces; and (iii) the network communication model, which supports client/server and publish/subscribe paradigms for communicating information between NCAPs. The standard defines a network and transducer hardware neutral environment in which a concrete sensor/actuator application can be developed with respect to a concrete network.

The object model definition encompasses the set of object classes, attributes, methods, and behaviors that specify a transducer and a network environment to which it may connect. This model uses block and base classes offering patterns for one Physical Block, one or more Transducer Blocks, Function Blocks, and Network Blocks. Each block class may include specific base classes from the model. The base classes include Parameters, Actions, Events, and Files, and provide component classes.

Block classes form the major blocks of functionality that can be plugged into an abstract card-cage to create various types of devices. One Physical Block is mandatory as it defines the card-cage and abstracts the hardware and software resources that are used by the device. All other block and base classes can be referenced from the Physical Block.

The Transducer Block abstracts all the capabilities of each transducer that is physically connected to the NCAP I/O system. During the device configuration phase, the description is read from the hardware device what kind of sensors and actuators are connected to the system. The Transducer Block includes an I/O device driver style interface for communication with the hardware. The I/O interface includes methods for reading and writing to the transducer from the application-based Function Block using a standardized interface. The I/O device driver provides both plug-and-play capability and hot-swap feature for transducers.

The Function Block provides a skeletal area in which to place application-specific code. The interface does not specify any restrictions on how an application is developed. In addition to a State variable that all block classes maintain, the Function Block contains several lists of parameters that are typically used to access network-visible data or to make internal data available remotely.

The Network Block abstracts all access to a network employing network-neutral programming interface supporting both client-server and publish-subscribe patterns for configuration and data distribution.

B. ZigBee Architecture and Security

The ZigBee/IEEE 802.15.4 protocol profile [9], [10] is intended as a specification for low-powered wireless networks. ZigBee is a published specification set of high level communication protocols designed to use small low power digital radios based on the IEEE 802.15.4 standard for wireless personal area networks. The document 802.15.4 specifies two lower layers: physical layer and medium access control sub-layer. The ZigBee Alliance builds on this foundation by providing the network layer and the framework for application layer, which includes application support sub-layer covering ZigBee device objects and manufacturer-defined application objects.

Responsibilities of the ZigBee network layer include mechanisms used to join and leave a network, to apply security to frames and to route frames to their intended destinations. In addition to discovery and maintenance of routes between devices, including discovery of one-hop neighbors, it stores pertinent neighbor information. The ZigBee network layer supports star, tree and mesh topologies. Star topology network is controlled by one single device called ZigBee coordinator, which is responsible for initiating and maintaining devices on the network. Those devices, known as end devices, directly communicate with the ZigBee coordinator. In mesh and tree topologies, the ZigBee coordinator is responsible for starting the network and for choosing key network parameters.

The ZigBee application layer includes application support sub-layer, ZigBee device objects and manufacturer-defined application objects. The application support sub-layer maintains tables for binding, which is the ability to match two devices together based on their services and their needs, and forwards messages between bound devices. The responsibilities of the ZigBee device objects include defining the role of the device within the network (e.g., ZigBee coordinator or end device), initiating and/or responding to binding requests and establishing a secure relationship between network devices. The ZigBee device object is also responsible for discovering devices on the network and determining which application services they provide.

The ZigBee specification defines also techniques and services for safety and security support [11]. To increase reliability of data transmission, DSSS (Direct Sequence Spread Spectrum) is used. For keeping message integrity, it is possible to use 0, 32, 64, 128 bits data integrity options. Sequential data transmission guarantees the freshness that protects the network against replay attack when the attacker replays an older message to obtain answer. Each ZigBee device manages counters for maintaining data freshness.

Authentication on the network level using a common network cipher key protects the system against attacks from outside of the network with minimal memory requirements. Authentication on the device level is achieved by a unique link key shared by two communicating devices. This is prevention against attacks both from inside and outside of the network, but it leads to higher memory requirements. Confidence provides protection of the information in face of unauthorized users by data encryption. Data encryption prevents to learn only a part of message; we talk about semantic security. One of the features of semantic security is initial value, which is used with encryption. When the same message is encrypted two times, two different crypto texts are produced. ZigBee employs Advanced Encryption Standard (AES) with 128 bits key. Encryption and decryption is performed on the network level or on the device level. The encryption on the network level employs network key, which ensures protection against outside of the network. The second option is a protection on the device level. In this case we used a link key is used between two devices, what ensures protection against attacks both from inside and outside of the network.

C. TCP/IP-ZigBee Interconnection

According to the ISO Open Systems Interconnection vocabulary, two or more sub-networks can be interconnected using equipment called as intermediate system whose primary function is to relay selectively information from one sub-network to another and to perform protocol conversion where necessary. A bridge or a router provides the means for interconnecting two physically distinct networks, which differ occasionally in two or three lower layers respectively. The bridge converts frames with consistent addressing schemes at the data-link layer while the router deals with packets at the network layer. Lower layers of these intermediate systems are implemented according to the proper architectures of interconnected networks. When sub-networks differ in their higher layer protocols, especially in the application layer, or when the communication functions of the bottom three layers are not sufficient for coupling, the intermediate system, called in this case as gateway, contains all layers of the networks involved and converts application messages between appropriate formats.

An intermediate system represents typically a node that belongs simultaneously to two or more interconnected networks. The backbone network interconnects more intermediate systems that enable to access different networks. If two segments of a network are interconnected through another network, the technique called tunneling enables to transfer protocol data units of the end segments nested in the proper protocol data units of the interconnecting network.

Gateways and bridges offer two different ways how to provide connectivity in between TCP/IP and ZigBee networks. In context of ZigBee, gateways provide a full featured connectivity and allow a greater diversity of devices and applications that can be interconnected by ZigBee networks. Bridges are much simpler than gateways but serve a smaller application space. Gateway is a device that allows disparate networks to exchange information. Gateways allow wireless sensor networks to use wireless protocols such as ZigBee that are well suited for the harsh RF environment as well as battery powered applications and allow them to be integrated into existing applications. Gateways convert the

wireless protocols and sensor data into various formats necessary for industrial, commercial, and residential systems.

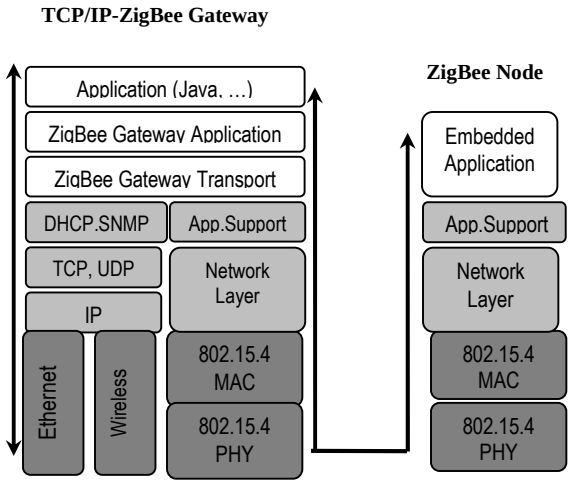


Figure 2. Gateway layered architecture

The ZigBee Gateway on Figure 2 provides an interface between ZigBee and IP devices through an abstracted interface on IP side. The ZigBee Gateway translates both addresses and commands between ZigBee and IP.

V. CASE STUDY

This section describes a case study based on a real industrial application that demonstrates utilization and refinement of the introduced framework aiming at Internet-based, tiered architecture with wireless sensor networks. The framework, which is conceptually based on the IEEE 1451.1 environment, is refined and extended to deal with pressure and temperature measurement and safety and security management along gas pipes. The related implementation stems directly from the IEEE 1451.1 model with Internet and the IEEE 1451.5 wireless communication based on ZigBee running over the IEEE 802.15.4.

The case study presents the framework both as a product of meta-design and as a design means including its refinement providing the final implementation.

A. Safety and Security Issues

The application architecture comprises several groups of wireless pressure and temperature sensors with safety valve controllers as base stations connected to wire intranets that dedicated clients can access effectively through Internet, see Figure 3. The WWW server supports each sensor group by an active web page with Java applets that, after downloading, provide clients with transparent and efficient access to pressure and temperature measurement services through controllers. Controllers provide clients not only with secure access to measurement services over systems of gas pipes, but also communicate to each other and cooperate so that the system can resolve safety and security-critical situations by shutting off some of the valves.

Each wireless sensor group is supported by its controller providing Internet-based clients with secure and efficient access to application-related services over the associated part of gas pipes. In this case, clients communicate to controllers using a messaging protocol based on client-server and publish-subscribe patterns employing 1451.1 Network Block functions. A typical configuration includes a set of sensors generating pressure and temperature values for the related controller that computes profiles and checks limits for users of those or derived values. When a limit is reached, the safety procedure, which is derived from the fail-stop model discussed above, closes valves in charge depending on safety service specifications. Examples include too low pressure for a pipeline segment, which means a gas leakage, or too high temperature, which means a danger of explosion. In both cases the safety procedure provides a pipeline reconfiguration by shutting off/opening some of the valves.

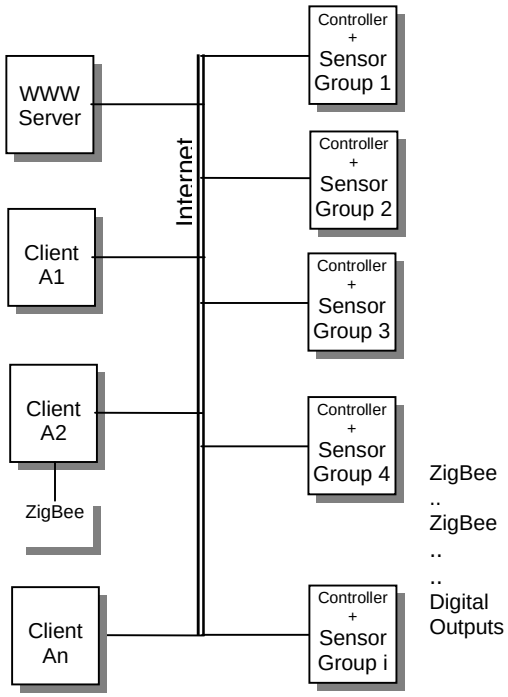


Figure 3. Network configuration

Security configurations in this case can follow the tiered architecture mentioned previously. To keep the system maintenance simple, all wireless communication uses standard ZigBee hop-by-hop encryption based on single network-wide key because separate pressure and/or temperature values, which can be eavesdropped, appear useless without the overall context. Not surprisingly, the application deploys standard ZigBee authentication.

Security support in frame of Intranet subnets stems from current virtual private network concepts. The discussed application utilizes ciphered channels based on tunneling between a client and a group of safety valve controllers. The tunnels are created with the support of associated authentications of each client.

B. Structure of the 1451.1 Implementation

The 1451.1 network model provides an application interaction mechanism supporting both client-server and publish-subscribe paradigms for event and message generation and distribution. Controllers play the role of clients or subscribers for the wireless part of the system network, and the role of servers or publishers for the wired part. Moreover, they compute temperature and pressure profiles, check the limit values and handle the safety valves.

In the transducer's 1451.1 object model, basic Network Block functions initialize communication between a client, which passed an authentication procedure, and the controller identified by a unique unicast IP address. The client-server style communication, which in this application covers both the configuration of controllers and initialization actions, is provided by two basic Network Block functions: *execute* and *perform*. The standard defines a unique ID for every function and data item of each class. If the client wants to call some function on server side, it uses command *execute* with appropriate parameters. On server side, this request is decoded and used by the function *perform*. That function evaluates the requested function with the given arguments and, in addition, it returns the resulting values to the client. Those data are delivered by requested variables in *execute* arguments.

The publish-subscribe style of communication, which in this application covers primarily distribution of measured data, but also distribution of group configuration commands, employs IP multicasting. All regular clients wishing to receive messages from a controller, which is joined with an IP multicast address, register themselves to this group. After that, when this controller generates a message by Block function *publish*, this message is delivered to all members of this group, without unnecessary replications.

Each controller communicates wirelessly with its sensors through 1451.5 interfaces by proper communication protocol. In the discussed case the P1451.5-ZigBee protocol, which means ZigBee over IEEE 802.15.4, was selected because it fits application requirements, namely those dealing with power consumption, response timing, and management.

C. Sensor Node Implementation

A typical node configuration, depicted on Figure 4, consists of STIM (Smart Transducer Interface Module) connected with a PSD sensor for differential pressure measurements, and with auxiliary temperature sensor for signal conditioning.

Of course, NCAP can be either embedded in a complex smart sensor, or shared among more simple smart sensors. On the other hand, from the viewpoint of Internet, only NCAP is directly addressable being equipped by its own IP address. Therefore, we can also denote as smart sensor the device consisting of an NCAP accessing one or more STIMs with connected sensors.

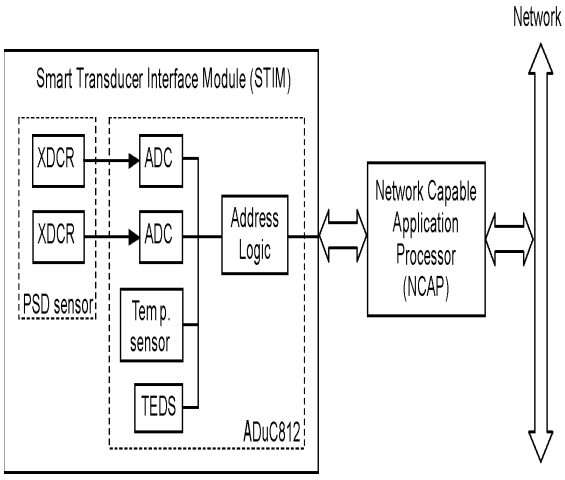


Figure 4. Sensor node example

This example discusses the pressure sensors with reflected laser beam and diffractive lens. The sensitive pressure sensor is based on a nitride membrane and an optoelectronic read-out subsystem. Measured pressure values are transformed into related thick-layer nitride membrane deflections. The nitride membrane serves as a mirror for laser beam, and it can move the related reflected laser mark. The mark's position is sensed using position-sensing device, which is a photo-lateral diode. Diode double current signal is amplified and conditioned digitally by the ADuC812 microcontroller. This microcontroller provides also the IEEE 1451.5 interface.

The sensing subsystem combines two principles that provide both high precision and wide range pressure measurements. Large displacements are measured by the position of reflected focused laser beam. Small position changes are measured by one-side layer diffractive lens principle. Sensor output signal is conditioned in digital by the ADuC812 single-chip microcontroller, which provides, with a simple hardware support, the IEEE1451.5 interface as one of its communication ports. This microcontroller calculates the position of the light spot and converts that position on the measured pressure using an internal table. Figure 4 depicts principles of the implementation of that smart sensor. The STIM contains (1) a PSD sensor with two analog differential transducers (XDCR), (2) a microcontroller ADuC812 with nonvolatile memory containing a TEDS field (Transducer Electronic Data Sheet) that props IEEE 1451.5 storing sensor specifications, (3) a TII (Transducer Independent Interface), (4) a temperature sensor necessary

for signal conditioning, (5) an analogue-to-digital conversion units (ADC), and (6) a logic circuitry to facilitate communication between the STIM and related NCAP.

The ADuC812 microcontroller, the basic building block of the smart pressure sensor electronics, includes on-chip high performance multiplexers, ADCs, FLASH program and data storage memory, an industrial standard 8052 microcontroller core, and supports several serial ports. The microcontroller may also utilize nonvolatile memory containing a TEDS field.

In this case, STIM represents a smart sensor serially interconnected with NCAP that provides controller functions and accesses Internet. Of course, each NCAP may be connected to many smart sensors, can access dedicated WWW server, and can be accessed by clients registered in related multicast group.

D. Design Pattern

In conclusion of this case study it could be helpful to restate the crucial architectural refinements of the proposed framework through building up a design pattern based on its basic abstract components, i.e. IEEE 1451 architecture, communication procedures and IP multicasting on the Internet side, and ZigBee Gateway on ZigBee side, which introduce a more detailed structure reusable for similar applications.

The 1451.1 object model provides skeleton supporting individual components. Its Network Block is refined so that it enables to access data-link layer communication services through unicast on IP with client-server procedure for start-up configuration and run-time maintenance, or through IP multicast with publish-subscribe procedure for run-time process measurements on both application data users and transducers sides. This refinement covers also selection of the most appropriate multicast routing protocol for local Internet traffic in case when the relevant parts of the network are accessible through routers.

The Transducer Block includes methods for reading and writing to transducers from the application-based Function Block using the standardized interfaces. The I/O device driver provides both plug-and-play capability and hot-swap feature for each transducer. It enables run-time reconfiguration of sensors that can support robustness of the system or improve measurement efficiency.

The Function Block contains a measurement application code. In the current case it prescribes sampling times, data filtering, linearization, conversions and transformations improving measurement accuracy and stability. The Function Block contains lists of parameters used to access net-work-visible data and, concurrently, to make internal data available remotely. The Function Block enables in this case to compute pressure profiles along the pipeline, pressure or temperature gradients, and the speed of pressure or temperature changes in time.

The ZigBee Gateway provides a prototypical interface between ZigBee and IP devices through an abstracted interface on IP side. The gadget translates both addresses and commands between ZigBee and IP standard architectures. Instead of that gateway, much more simple bridge can be used; unfortunately, such type of interconnection restricts substantially functionality of the application.

From more general viewpoint, the design pattern provides embodiment of meta-design principles for creating flexible design environments that can support development of various dependable applications respecting not only special functional requirements, but also requirements on system's safety and security. Necessarily under-designed open source tools and techniques create an open design space for end-user developers in various application domains of networked embedded systems.

VI. CONCLUSION

Internet technologies complemented by wireless access networks are rapidly becoming the preferred choice for building next generation distributed measurement and control systems. The framework, which can provide for current trends, stems from the IEEE 1451.1 standard specifying smart transducer interface architecture that enables to unify interconnecting smart sensors and sensor-based embedded systems with various wireless networks, and their direct coupling to recent Ethernet-based Intranets. Supporting techniques included in the frame-work, namely publish-subscribe messaging by IP multicast and security maintenance, offer scalable and traffic-saving solution important from viewpoint of the contemporary Internet. The schemes discussed can properly interplay with each other and can supply suitable support for design of networked, sensor–based embedded system applications.

This paper discusses a deployment of meta-design principles for building up an adaptable design framework focused on embedded systems and their components interconnected by Internet and ZigBee. Necessarily under-designed open source tools and techniques create design spaces for end-user developers. In that way the meta-design approach supports reusability among various application domains. Explicitly, the paper demonstrates both the use of this framework for implementation of development environment aimed at Internet-based smart sensor applications and, concurrently, the utilization of this framework for development of pressure and temperature measurement and safety and security management along gas pipes. The paper brings this design approach in manner suitable not only for framework builders, but also for end-user developers in a variety of application domains.

The current manuscript, which extends and updates the paper [1], stems partly from the previous research tasks published in [12], [13], and [14]. As a main contribution it delivers meta-design approach to a real world example of the design framework's stepwise refinement for a safety and security-critical sensor networking application.

A. Future Work

The next related research will be focused on the deployment of formal specification techniques including related tools for the industrial embedded systems domain, in more detail mentioned in the paper [15]. We will strive to create a design and development environment supporting automatic or semi-automatic generation of executable prototypes with behaviors satisfying not only functional requirements, but also safety and security constraints, from verified formal specifications.

ACKNOWLEDGMENT

The research has been supported by the Czech Ministry of Education in frame of the Research Intentions MSM 0021630528: Security-Oriented Research in Information Technology and MSM 0021630503: MIKROSYN -- New Trends in Microelectronic Systems and Nanotechnologies, and by the Grant Agency of the Czech Republic through the grant GACR 102/08/1429: Safety and Security of Networked Embedded System Applications.

The authors acknowledge also contributions to this research by their colleagues from the Networks and Embedded Systems Research Group and from the Secure and Reliable Network Architectures Research Group at the Department of Information Systems, Faculty of Information Technology of the Brno University of Technology, and from the Department of Microelectronics, Faculty of Electrical Engineering and Communication of the Brno University of Technology.

Initially, this work was supported also by the Grant Agency of the Czech Republic through the grants GACR 102/05/0723: A Framework for Formal Specifications and Prototyping of Information System's Network Applications and GACR 102/05/0467: Architectures of Embedded Systems Networks.

REFERENCES

[1] M. Sveda and R. Vrba, "Meta-Design Support for Safe and Secure Networked Embedded Systems", Proceedings Third International Conference on Systems, ICONS 2008, IARIA, Published by the IEEE Computer Society, 2008, pp. 69-74.

[2] A.I. Morch, et al., "Component-Based Technologies for End-User Development", Communications of the ACM, Vol.47, No.9, 2004, pp.59-62.

[3] G. Fischer, et. al., "Meta-Design: A Manifesto for End-User Development", Communications of the ACM, Vol.47, No.9, 2004, pp.33-37

[4] B. Melhart and S. White, "Issues in Defining, Analyzing, Refining, and Specifying System Dependability Requirements", Proceedings of the IEEE Conference and Workshop ECBS'2000, IEEE Computer Society, Edinburgh, Scotland, 2000, pp.334-340.

[5] N.G. Leveson, "Software Safety in Computer-Controlled Systems", IEEE Computer, February 1984, pp.48-55.

[6] I.-G. Kim, et al., "Formal Verification of Security Model using SPR Tool", Computing and Informatics, Vol.25, No.5, 2006, pp.353-368.

[7] Li, H. and M. Singhal, "Trust Management in Distributed Systems", IEEE Computer, Vol.40, No.2, 2007, pp.45-53.

[8] IEEE 1451.1, Standard for a Smart Transducer Interface for Sensors and Actuators -- Network Capable Application Processor (NCAP) Information Model, IEEE, New York, USA, 2000.

[9] IEEE 802.15.4, Wireless Medium Access Control and Physical Layer Specification for Low-Rate Wireless Personal Area Networks, IEEE, New York, USA, 2003.

[10] ZigBee: ZigBee Specification. ZigBee Alliance Board of Directors, 2006, Website http://www.zigbee.org/.J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[11] P. Baronti, et al., "Wireless Sensor Networks: A Survey on the State of the Art and the 802.15.4 and ZigBee Standards", Computer Communications, Vol. 30, 2007, pp.1655-1695.

[12] M. Sveda, "End-User Development Framework for Embedded System Applications", Proceedings 14th IEEE International Conference on the Engineering of Computer-Based Systems ECBS07, IEEE Computer Society, Tucson, Arizona, USA, 2007, pp.186-192.

[13] M. Sveda and R. Vrba, "Dependability-driven Embedded Systems Networking", Proceedings 6th International Conference on Networking ICN 2007, IARIA, Published by the IEEE Computer Society, 2007, pp.483-488.

[14] M. Sveda and R. Trchalik, "Safety and Security-driven Design of Networked Embedded Systems", Proceedings 10th Euromicro Conference on Digital Systems, Digital System Design Architectures, Methods and Tools, IEEE Computer Society, Lübeck, Germany, 2007, pp.420-423.

[15] M. Sveda, O. Ryšavý, and R. Vrba, "Pattern-driven Reuse of Behavioral Specifications in Embedded Control System Design", Frontiers in Robotics, Automation and Control, Vienna, AT, IN-TECH, 2008, pp.151-164.

Security and Authentication Architecture Using MPEG-21 for Wireless Patient Monitoring Systems

Wolfgang Leister and Truls Fretland
Norsk Regnesentral
Oslo, Norway
email: {wolfgang.leister, truls.fretland}@nr.no

Ilangko Balasingham
Interventional Center
Rikshospitalet University Hospital
Oslo, Norway
email: ilangkob@medisin.uio.no

***Abstract*—Privacy and security are two major concerns in the ubiquitous deployment of wireless patient monitoring systems, where wireless sensor networks become an integral part of the monitoring process. To address and handle threats which arise from the use of wireless sensors, we propose a framework using MPEG-21. MPEG-21 is an architecture that can handle end-to-end management of multimedia content in diverse networks. We propose and evaluate a framework that is designed to protect patient monitoring systems using resource-constrained wireless sensor networks. The analyses show that security architectures based on the MPEG-21 framework can handle a variety of threats. We also present a test bed in which our framework is about to be implemented.**

***Index Terms*—MPEG-21, security, biomedical sensor networks, medical digital item**

I. Introduction

Patient monitoring systems are one of the major data sources in a health care environment. These typically consist of sensors which are connected to the patient, communicate by wire or wirelessly to a bedside monitoring device. Furthermore, the sensor data are stored in databases, which are connected to the health care enterprise's information infrastructure for storage, maintenance and retrieval of data. Health Care information systems can be interconnected in order to exchange data, where patient monitoring systems become an integral part of the networked infrastructure. This facilitate data to be available to the different user terminals and systems both inside and outside the enterprise. Therefore, an end-to-end security mechanism is needed to protect the medical data, as we presented recently [1].

Wireless technology is increasingly used in health care enterprises to eliminate the use of cables in patient monitoring systems, providing mobility advantages for patients and medical personnel. In this case the sensors communicate wirelessly with monitoring systems, which are located close to the patient. However, wireless communication can be intercepted easily. Threats to security goals like confidentiality, integrity, and availability of data still apply, and weaknesses of the system treating health care data could be exploited by attackers.

A biomedical sensor network (BSN) can be considered a special case of a wireless sensor network (WSN). A WSN often comprises tiny, low-cost wireless electronic devices, capable of gathering vital signs and environmental information and forwarding them to a base station. The limited computational and communication capabilities, their reduced cost, and enforced size introduce resource-related challenges in their function, efficiency, and security. Attacks can compromise system security with negative consequences for both the patient, the health care enterprise, and third parties. This exhibits that the security requirements need careful consideration during design, development and deployment for the whole infrastructure including the patient monitoring systems and BSN.

State-of-the-art systems in health care enterprises employ technical approaches [2] such as service-oriented architectures (SOA) [3], [4], in order to address re-usability, interoperability, and portability. For some application areas standards such as DICOM [5], [6] are used. However, none of these technologies addresses content adaptation taking into account of characteristics of end user terminals (screen size, resolution, real time rendering capabilities, etc.), the user's preferences (automatic update, popup menu and functions, etc.), and wireless channel conditions (bandwidth, data rate, packet loss, interference environment, etc). However, MPEG-21 is one potential technology which can support a set of quality of service metrics, security features as well as the above mentioned adaptation features to internal factors. Therefore MPEG-21 appears to be a far better solution for wireless sensor networks than other above mentioned standards and technologies.

MPEG-21 [7], [8] is an international standard on multimedia services and provides a multimedia framework at the application level to enable transparent and augmented use of multimedia resources across a wide range of networks and devices. MPEG-21 addresses sharing and transferring digital media content, including management, adaptation of resources, protection of privacy, integrity, and digital rights.

The contribution of this paper is a novel approach to protect the application layer of medical data in patient monitoring systems using BSN. The paper is organised as follows: Section III gives an overview of security issues for wireless patient monitoring systems and biomedical sensor networks, including a generic system model. Section IV discusses the security assumptions and requirements for such systems, which lead to a short threat assessment in Section V. Section VI reviews the relevant parts of MPEG-21 which are used in our proposed

architecture that is presented Section VII. In Section VIII we discuss the employed security mechanisms and the suitability before concluding the paper.

II. Related Work

Regulations for handling health care data are strict in most parts of the world. This is manifested by the relevant legislation in Norway [9] and the European Union [10]. Especially the issues of privacy and data integrity are stated in these documents.

The security issues of wireless sensor networks (WSN) have become an important research topic. We base our work on an overview of security goals, threats, attacks and countermeasures on all communication layers [11] and perform analyses on BSNs given their constraints using literature on security issues in WSN [11], [12], [13]. There are still many unsolved security issues, such as the integrity of the collected data and the privacy of the patient [14], [15].

One of the ongoing works is the IEEE 802.15.6 standard for body area sensor network [16]. The submitted proposals so far on security provisions discuss issues related to outgoing and incoming frame security procedures, higher layer security functions in the medium access (MAC) layer, common and different information elements across layers, encryption key usage, handling and update, etc. Incorporation of parts of the MPEG-21 in this proposed standard may become beneficial to handling security provisions as well as quality of services in a single framework.

The recently approved standard IEEE 1451.5 [17] envisions encryption and security functionalities in the presentation layer of sensor networks. The framework proposed in this paper uses MPEG-21, which also addresses issues in the presentation layer. Therefore, a combination of MPEG-21 and IEEE 1451.5 can be interesting.

It will be a challenging task to design and deploy appropriate security mechanisms that take availability, user friendliness, high throughput of data, etc. into consideration. Early work on using MPEG-21 as a framework in health care has been conducted by Landén [18], which has been recently extended [19] to include the hospital infrastructure. After a threat analysis of patient monitoring systems [20] we extended this architecture to include BSN [1].

Other approaches to use MPEG-21 in health care use the IPMP part of MPEG-21 for patient records [21]. Recently, a framework for using MPEG-21 IPMP components for a security framework for pervasive health care architectures has been presented [22], including wireless communication between personal digital assistants (PDA). While this work introduces MPEG-21 for medical applications, our work includes the use of MPEG-21 for BSN and uses a generic model to analyse the threats.

III. Patient Monitoring Systems

Patient monitoring systems and BSNs can be applied in a variety of health care scenarios ranging from paramedic, diagnostic, surgical, to post-operative phases. In general, patient monitoring systems comprise of different kinds of sensors, data communication, storage, processing, and presentation of medical data. In order to articulate the security requirements we identify three important scenarios: (1) hospital scenario (using an array of biomedical sensors for diagnostics, surgical, and post operative phases), (2) nursing and citizen homes (patients equipped with wireless biomedical sensors triggering alarms; surveillance of patients after being discharged from hospital), and (3) paramedic.

A. Biomedical Data

While a health care information system must handle all types of medical data, we concentrate on biomedical sensor data. The sensor nodes measure biomedical signals, process them and transmit the results to a sink node. Typical biomedical data measured by biomedical sensors can be electrocardiogram (ECG), electroencephalography (EEG), blood oxygen saturation, blood pressure, temperature, and sound. They are data samples at a given sampling resolution and sampling rate with typical data rates from some few bits per second (bps) up to 12000 bps. The sampling rate and resolution are examples of biomedical metadata, that is, data that contain information about the biomedical measurements. In the near future also images and video, will emerge as data types from BSN.

Biomedical data can be described as streamed multimedia data with corresponding requirements about confidentiality, integrity, availability, as well as data authenticity, service quality and adaptation.

The biomedical sensor data consist of one or several tracks of sampled measured values, supplemented with metadata, e.g., a time-stamp and the identity of the sensor. The biomedical data must be protected against modification and deletion. While it is necessary to implement a detection mechanism for modification and deletion, the reconstruction of data destroyed by an attacker would be desirable, in order to provide the availability of data.

B. A Model for Wireless Patient Monitoring Systems

A generic system model for the overall patient monitoring system and threat analysis have been proposed [20], where the components and communication channels have been identified.

The generic system model, shown in Fig. 1, illustrates that patient data are generated by sensors attached to a patient. In a concrete instance of this model, each abstract component can be realised by means of several physical components, and a physical component again can be realised as several abstract (sub-)components. A biomedical sensor network consists of several sensor nodes that measure biomedical data. These data, accompanied by metadata, are transmitted by a wireless network (Channel A) to the sink of the wireless network and to the patient data collector (PDC). The PDC collects different data streams for a patient and forwards data to the health care information system at the hospital (Channel B), or to the patient data accessing unit giving data access to the medical staff on site (Channel E). The ID data mapper functionality

(Channel G) is necessary to handle patients, where the identity might not be known. To implement Channel G no real communication needs to be involved while the system is in use. Retrieval of patient data from the health care information system involves Channel D.

The **components** of the generic system model are:

- *Sources* of patient data, e.g., sensors, storage units, or user input devices. Each source is attached to only one patient at a time. A source is assumed to have very limited capabilities of protecting the communication.
- *Patient Data Collectors (PDC)* collect patient data from one or more sources. A PDC is trusted to handle unencrypted, personally identifiable patient data.
- The *Health Care Information System (HCIS)* receives patient data for processing or storage.
- The *Patient Data Accessing Unit (PDAU)* receives patient data and presents these to the medical staff.
- The logical element *ID-data mapper* determines the identity of the patient to whom patient data pertains and sends the identity to the PDC or the HCIS. The ID-data mapper is usually implemented as an interface rather than a separate entity.

The generic model model includes the following **channels**:

- Channel A between the source and the PDC, is based on a short-range wired and/or wireless communication links. The Channel A might be implemented by a wireless biomedical sensor network.
- Channel B is a long-range wired or wireless communication link between the PDC and the HCIS. Channel B can be implemented in a trusted environment or possibly over untrusted public networks by external providers, e.g., GSM, GPRS, UMTS, WiMAX or PSTN networks.
- Channel D may be implemented as any type of communication link, possibly over a public network.
- Channel E may be an internal interface or a wired/wireless short-range communication link. Long-range communication between PDC and PDAU should be provided via the HCIS.
- Channel G is implemented as an internal interface in one of the components used to retrieve the patient identity.

C. Applying the Generic Model

The generic model is applied to the relevant scenarios in various ways, where the components and channels of the generic model can be implemented differently. Therefore, the threats might be different depending on the chosen scenario. The security requirements for Channel A are equal in all scenarios, and are therefore treated separately in the course of our work.

In a hospital scenario (Scenario 1) the PDC, PDAU, and HCIS might be part of an approved protected infrastructure and authenticated to each other, e.g., implemented by using a virtual LAN. In such an environment, Channel A needs specific attention regarding security. In different sub-scenarios the PDC might be implemented as a bedside terminal serving just one patient, or as a base station that might serve an entire hospital corridor. The threats affecting the patient monitoring system differ for these cases.

For a nursing and citizen homes scenario (Scenario 2) most channels might be outside the protected infrastructure of a hospital. Channels B, D, and E are implemented various short- and long-range technologies, possibly using third party providers like telecommunication providers. The PDC might be implemented in connection with a set-top box in the patient's home.

For a paramedic scenario (Scenario 3) the PDC might be implemented in an ambulance, serving one or several patients. Channels B and D might be non-existent or implemented by the means of a long distance communication possibly using third party providers. Channel E is implemented in most cases using a short-range scenario. The paramedic scenario has the most stringent requirements, since emergency routines must be available, e.g., in case a sensor must be replaced or moved to another patient, or the patient's identity might be unknown.

Common for all scenarios is that Channel A is implemented by the means of a WSN, the placement of the sensors is done under the responsibility of authorised personnel using approved routines, and that channels which are not part of a protected environment are secured properly. Whether the PDC handles one or several patients might have an impact on threats, but the architecture should be able to address these.

D. Implementation of Selected Communication Channels

The channels in the generic system model can be implemented in a variety of communication technologies. When not within a trusted infrastructure, these channels must be protected. A threat analysis gives answers on what to protect, and possible countermeasures to specific threats, for instance by the use of virtual private networks (VPN). Since securing the single technologies against these threats will not result in a uniform architecture, we advocate for handling most threats in the application layer.

For long-distance communication public networks based on different technologies (GSM, SMS, GPRS, UMTS, WiMAX, etc.) are used, which includes the transfer of data through the networks of external providers. Since it is not allowed to transfer medical and sensitive data unsecured, measures must be taken to protect these data, e.g., by using a VPN, securing the application layer, or other means [6]. Here, the Channels B or D are considered as long-distance communication channels.

The Bluetooth technology is emerging as communication technology for parts of the wireless infrastructure for short-range communication, designed to facilitate communication without the need of wires. In our scenarios (parts of) the Channels A, B, D, and E can be implemented using Bluetooth, using a point-to-point connection.

E. Biomedical Sensor Networks

Biomedical sensor networks [11], [20] can be a part of a patient monitoring system, located as source, (parts of) Channel A, and possibly the PDC. BSNs comprise of one

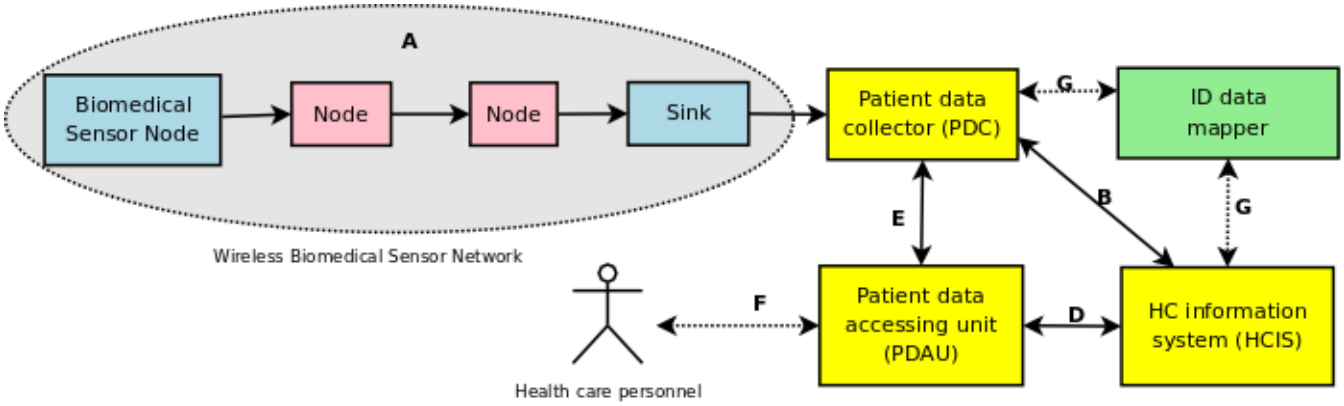


Fig. 1. Generic system model with Channel A shown in detail.

or several sensor nodes, possibly several *transfer nodes*, and one or more sink nodes which are attached to the PDC.

A biomedical sensor node is an electronic device which performs the tasks of sensing, processing, sending, and/or receiving biomedical data. The sensor node can be decomposed into four abstract parts: sensor, receiver, processing unit, and transmitter. Technically, a sensor node is built up of an MCU or CPU, memory, a wireless communication device, biomedical sensors, and a power supply often based on a battery. The functionality of a biomedical sensor node is controlled by software, usually consisting of firmware, operating system, and specific application software for treating the biomedical signals and their transfer.

An extensive list of constraints that apply to sensor networks have been given in a report by DARPA [23], classified as (a) sensor node constraints and (b) networking constraints. Many of these constraints, like small memory resources, imposes limitations to the implementation of traditional security mechanisms on sensors. However, not all of these constraints apply to the special case of a *biomedical* sensor network. In particular the unattended operation constraint does usually not apply, since the sensors will be attached to patients that are conscious or supervised by health care personnel.

As shown in Fig. 1 a BSN also can contain transfer nodes that forward the medical data from the source to the PDC. These transfer nodes are used to enlarge the distance for reaching the PDC even for sensor nodes that cannot reach the PDC directly. This is especially important for devices that have little battery capacity, and thus limited transmission power.

Despite of the scarce resources of the sensor nodes, lightweight implementations for encryption of data have been developed, e.g., Sizzle [24].

IV. Security Assumptions and Requirements

Generally, from the requirements and the nature of the application or system, an analysis results in a list of threats. These threats are then analysed, and countermeasures and recommendations will then be used in the design, implementation, and deployment of systems. For the various parts of a complex system like a patient monitoring system we analyse the characteristics of the components and channels in the generic system model, before considering security aspects of the entire model. Presenting the identification of security aspects to wireless patient monitoring systems, it is customary to take into account the different abstraction levels [20], such as the stakeholder level, the application level, the communication level, and the technical level. While we recognise that it is important to analyse all levels in a health care enterprise, we concentrate on the technical aspects in the application and communication levels.

A. Security Requirements for Patient Monitoring Systems

For a patient monitoring system the security goals are (a) *availability*, the intended receiver are able to read the data; (b) *data confidentiality*, only the intended receiver are able to read the data; (c) *data integrity*, the received data has not been tampered with or destroyed, and violations of this must be detectable; and (d) *data authenticity*, the sensor data are linked to the correct patient. Note that linking data to the wrong patient could lead to wrong diagnosis and eventually mistreating the patient, and we do not consider safety requirements, like radiation, or other physical effects having an impact on the patient.

For those parts of a patient monitoring system that communicate within a trusted environment, we consider all technical security requirements to be fulfilled, since the regulations for the use of ICT systems in health care enterprises require this to be approved. For channels or components outside a trusted environment, we consider the Dolev-Yao threat model [25], where the attacker can overhear, intercept, and synthesise any message, and is only limited by the constraints of the cryptographic methods used. For certain use cases we consider a selection of components to be trusted, i.e., we assume that such components are not compromised. Also certain channels, e.g., those within a protected infrastructure within a health care enterprise are considered to be secure.

For all channels, personally identifiable patient data shall be protected from eavesdropping and unauthorised modifications

when transmitted across open networks, in order to provide confidentiality and integrity. Additionally, Channels D and E shall authenticate the user; Channel D shall additionally authenticate the HCIS, and Channel E shall authenticate the PDC. All components that handle unencrypted data must deny unauthorised access of any kind, such as viewing, insertion, transformation, deletion of patient data.

The HCIS as a part of the hospital infrastructure shall (1) verify the integrity of patient data, (2) authenticate the PDC, (3) know the identity of the patient, (4) know to whom the patient data pertain, and (5) know the type of source used to produce the patient data. The PDAU shall in addition to the requirements for all components verify the integrity of the patient data.

For all components where emergency access functionality is available, the invocation of emergency access shall override the restriction on read access. For all components, except the source, an emergency access shall trigger extended monitoring of relevant events to enable the detection of unnecessary access.

B. Security Requirements for BSN

Related to the generic system model the wireless BSN is a specific implementation of Channel A. The characteristics of Channel A include that the signals are not limited to a controlled area or device, and hence are accessible to anybody in the proximity with the appropriate equipment. Authentication and measures against eavesdropping are therefore a necessity.

While security measures for BSN must be available at all communication layers most of the security requirements can be handled on the presentation layer. However, some issues, e.g., tied to routing and consequently refer to the requirement of availability, are better handled at the network layer. We refer to the layer model shown in Fig. 2 where the security measures are placed in the presentation layer.

Using the Dolev-Yao threat model [25] to analyse a BSN we consider the source, and the sink to be trusted, while Channel A, including possible transport nodes could be in the control of the attacker. The possible intrusion by an Dolev-Yao attacker implies that the existence of transfer nodes must be considered, also in a one-hop environment. A potential attacker could provide an extra node without anybody knowing about it, often denoted as “man in the middle” or proxy attack. The use of time and distance bounding protocols [26] on the network level can counter such threats.

The existence of fake nodes must be considered, why authentication of sensors must be mandatory, so that data sent from a sensor are not disclosed, and fake data are recognised.

Since the data source has limited storage it shall not store data longer than necessary. Due to limited battery capacity unnecessary communication, both sending and receiving, shall be avoided.

Even if the sensor data are encrypted, the sensor encryption is lightweight and can be broken given sufficient time and resources. Hence, the data sent from the sensor to the PDC should not contain person-identifiable data such as a social security number. Instead, the sensor data should be linked to the corresponding person in the PDC, by using the ID-data mapper. Person-identifiable data may be included at the PDC, since it has more resources available, and can implement stronger encryption algorithms and longer keys than an ordinary sensor node.

An adversary shall not be able to inject data packets without these being detected. Re-played data packets must be detected to ensure data freshness. Since characteristics of routing and forwarding make it necessary to detect duplicate arriving packets the the network layer or above must offer detection mechanisms using counters or other identifying data.

On the network layer attacks to WSN, and BSN in particular, include attacks to routing and forwarding. On the upper network layers such attacks are recognisable by missing, defect, manipulated, duplicated, or delayed packets. While these attacks can be detected they cannot be prevented on the presentation layer.

We assume that deploying a sensor, key establishment, assigning an identity to a sensor, and coupling the sensor node to a patient identity is done in a routine ahead of the normal operation of the sensor network; trusted medical personnel performs this operation, e.g., by coupling the sink and the sensor node.

The sink node has rather large computation and communication properties in order to perform data aggregation, validity check, identity assignment, etc. Transfer nodes are not supposed to perform these tasks, and hence do not need keys to decrypt sensor data.

Any form of data aggregation on transfer nodes should be avoided by two reasons: (1) data aggregation on these nodes would need extra resources; and (2) data aggregation would make it necessary that data are decrypted unless privacy transformations [27] are employed. For aggregation in the transfer nodes the key distribution problem would be much more complex. Therefore, we do not foresee data aggregation mechanisms within Channel A.

In contrast to a general wireless sensor, the biomedical sensors are in a rather controlled environment, so that we do not consider destruction or tampering with the sensors. Compromise of a sensors key does not affect the rest of the network, since its key is shared only with the sink node.

Communication on Channel A shall be short-range, while the transmitted data shall be protected from eavesdropping. Integrity protection of the transmitted data is necessary, since interferences from other medical instruments are possible. Automatic roaming to other PDCs shall not be allowed.

The PDC shall in addition to the requirements for all components: (1) verify the correct identity of the source, (2) not modify the patient data, except for aggregation or other defined transformations, and (3) not store data longer than necessary to ensure successful transmission of patient data to the HCIS.

To provide flexibility we must consider security challenges for software upgrades via the wireless network, reconfiguration, self-organisation, and device-mobility (e.g.,

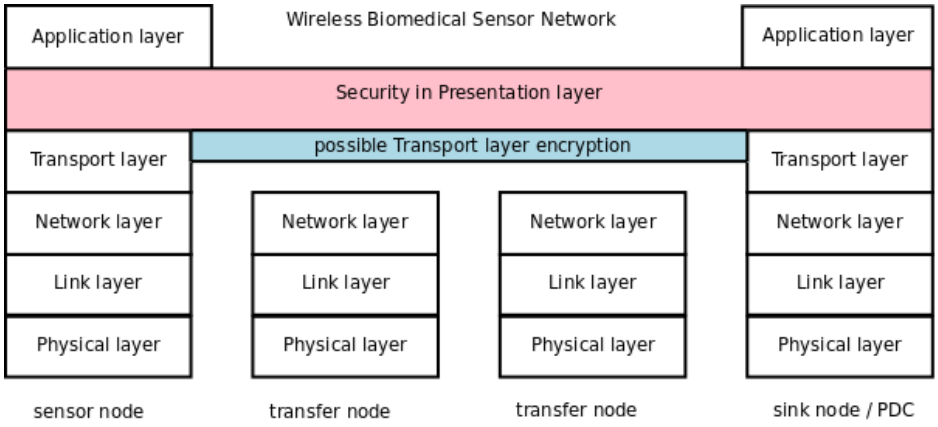


Fig. 2. Layer model for BSN.

handovers on different layers). Configuration control (e.g., check whether versions of hardware and software fit together, verify whether the patient is in the correct situation) shall be employed.

Aspects of availability are often neglected, which include scalability, quality of service (QoS), and power consumption. A power outage of a sensor may compromise the availability of data and thus threaten the patient. QoS includes mechanisms to provide an agreed service level regarding parameters such as network bandwidth, data throughput, signal quality, response time, latency, packet delivery rate, jitter, and power consumption. In complex communication environments, mechanisms should be employed to mitigate interference from co-existing wireless networks and various medical applications, which requires robust error detection and correction algorithms in packet transmission procedures.

V. Threat Assessment of Patient Monitoring Systems

The general threats and attacks to these security objectives are eavesdropping, denial of service, masquerading, and disclosure. For the components the threats and associated factors include (1) compromised or fake components (physical or logical attack), (2) destroyed, malfunctioning, lost, or stolen component, (3) software errors (e.g., failure in security mechanisms, routing, etc.), (4) misuse of emergency access, and (5) denial of service attacks (physical or logical attacks, bad quality, accidents, etc.). For the channels threats include: (6) compromised or fake (components of) communication infrastructure (physical or logical attack), (7) unstable communication infrastructure (physical or logical attack, bad quality, accidents), and (8) eavesdropping of communication. These threats and vulnerabilities may lead to the unwanted consequences that information or equipment might be unavailable, incorrect information is received (medical data, patient identity, sensor type, etc.), sensitive information is leaked, and eventually damage to the patient, operators or equipment.

Many of the security mechanisms of the generic system model employed as countermeasures are distributed over several components. These components include (1) the key generation service for session keys; (2) security protocols that support authentication, confidentiality, integrity, key establishment, and the associated cryptographic algorithms; (3) signature generation and verification services, along with a local storage for credentials and keys; (4) access enforcement services for both users and system components; (5) the log collector to collect metadata about events; and (6) front-ends for various services, such as security administration and authentication.

An architecture for patient monitoring systems must address patient identification and identity management of patients, devices and personnel. Other elements include device administration (which requires key management), user administration, authorisation policy, mandatory encryption for data stored on mobile devices, communication encryption, communication and data source authentication, data manipulation detection, and logging of critical events [28].

A. Vulnerabilities in Communication Channels

For short- and long-range communication the channels not residing in the secure enterprise infrastructure include wireless short-range, wired and wireless long-range communication. When security mechanisms are not offered, e.g., when using a channel over a public network, presentation or application layer mechanisms must be employed, e.g., using authentication and encryption mechanisms offered by the protocols of the appropriate layer.

Bluetooth, which often is used for short-range communication, employs rather weak encryption based on the assumption that the communication is short-range. However, this argument may be questionable since well equipped attackers using signal amplification and directional antennae can enlarge the communication range substantially. Also the fact that the Bluetooth devices often are very mobile and can contact many other Bluetooth devices short-range while passing by, is a substantial threat.

The sole secret credential in a Bluetooth network is the PIN code of the device. While weaknesses in the cryptographic

protocol can be neglected, there are different implementation weaknesses for the equipment of some vendors. When these are exploited, devices and sessions can be taken over by an intruder [29].

Since communication over SMS is not reliable, the use of IP over GPRS, EDGE, 3G, or similar technologies is recommended. An encryption scheme and intruder detection must be employed in order to provide privacy. Note that the risks for attacks are different whether an attack affects one patient or many patients. While there are several PKI solutions available, the use of keys stored in the SIM cards reflects the security needs for mobile patient monitoring systems.

B. Threats Affecting BSNs

The general threats for biomedical sensor networks can be characterised into the following domains: (1) the entity domain, (2) the network domain, (3) the routing and forwarding domain, and (4) the specific protocol domain.

Any adversary can eavesdrop on the traffic, inject new messages, replay or change previous messages. The biomedical sensors do not trust each other; while each sensor node trusts itself. Base stations (gateways) are assumed not to be compromised. However, compromising them could render the entire sensor network useless. Thus the base stations are a necessary part of the trusted computing base, and all sensor nodes trust the base station.

A sensor network should be both preventive and resilient to severe types of attacks regarding both control traffic and data traffic. Typical examples of control traffic are routing, monitoring whether a node is awake, asleep, or dead, topology discovery, and distributed location determination. Control traffic attacks include blackhole attacks, wormhole attacks [30], rushing attacks [31], sybil attacks and compromised sensors [32], [33], sinkhole attacks [34], and HELLO flood attacks [34]. Control attacks are especially dangerous because they can be used to subvert the functionality of the routing protocol and create opportunities for a malicious node to launch data traffic attacks such as dropping all or a selective subset of data packets. A sensor network should be both preventive and resilient to severe type of attacks such as wormhole attack, the Sybil attack, and compromised sensors [35].

While secure routing ensures that data are forwarded to the correct recipient, it does not include confidentiality and protection against replay attacks. This is caused by underlying spoofed, altered or replayed routing information, selective forwarding, acknowledgement spoofing, and the attacks mentioned above.

At the sensor node domain, an attack could change settings in a sensor or transmitter unit, its software or data, resulting in a threat. Consequences might be exposure to increased heating or radiation from the device. The general attacks by a malicious entity can therefore be classified as fake emergency warnings, prevention of legitimate warnings from being reported, battery power depletion, excessive heating in the tissue of the patient, and radiation from the entity.

Countermeasures to threats affecting a BSN are often specific to the employed communication technology, and thus security mechanisms should be implemented in communication layers below the transport layer [20]. Since BSN use technologies that are introduced in the market, the implementation of lower communication layer countermeasures are not always viable in a health care enterprise. However, countermeasures to avoid that attackers exploit weaknesses include link layer encryption and authentication using symmetric keys, the use of packet counters to avoid replay attacks, encryption, verification of identities, and various countermeasures on the link-layer [34].

In order to meet the challenges that the threats impose we propose that countermeasures are employed on the presentation layer, which are not specific to the underlying communication technology. Since not all threats on the lower layers can be prevented on the presentation layer, such as attacks to routing or denial of service, inconsistencies should be at least detectable at the application layers. Measures at the application layer to be taken should include encryption, measures for data integrity and authenticity.

For the sake of scalability, authorisation schemes must be role-based (as opposed to being individual-based). The authorisation database stores information about roles and their assigned privileges, which might be constrained by contextual information. Session keys are preferably generated during the authentication protocol, while confidentiality is established through encryption, and integrity is established through a signature generation service. Since the HCIS cannot authenticate all sources, such as biomedical sensors, these sources must be authenticated by a proxy, i.e., the PDC identifies the source, and guarantees its authenticity towards the HCIS.

Since broadcast channels are used in BSNs and energy consumption by the device is an issue, security mechanisms must be carefully designed. Some security mechanisms, such as encryption, use extra computing cycles and therefore consume extra energy. The broadcast of unnecessary data consumes extra energy at both the sender and the receiver nodes in a BSN. Wherever a sufficient security mechanism is offered at lower layers in the communication stack, these should be employed, since these are better fitted to the employed technology than higher level mechanisms. However, there should be a minimal set of mechanisms in the application level that protect the data.

While the relationship between the patient and the patient data must be given at all times, it is not recommended that data that identify the patient are transmitted over unsecured channels. Instead, the relationship between patient and patient data is achieved by authenticating the device or sensor.

VI. MPEG-21 in Patient Monitoring Systems

MPEG-21 is a general framework for handling multimedia, from the content provider to the end-user. It aspires to be a unifying framework in the media industry facilitating multimedia transactions, and to equip the content providers with a tool to restrict illicit use of copyrighted material. Given the

recent battles between copyright holders and people illegally sharing the copyrighted material, this is still an unresolved issue. Despite this, MPEG-21 is not only suited for enforcing copyright, it can also be used to protect the life cycle of patient data.

The vision of MPEG-21, defined in ISO/IEC 21000, is *to enable transparent and augmented use of multimedia resources across a wide range of networks and devices* [8]. Since medical data qualify as multimedia resources this vision suits well within a patient monitoring system with its wired and wireless networks, portable and stationary devices, and a variety of multimedia resources. MPEG-21 also provides means to protect the content, so that the desirable level of privacy can be achieved. While not all of the eighteen parts of MPEG-21 are suited for our purpose, the most relevant parts will be presented briefly in the following.

A. Relevant Parts of MPEG-21

The most fundamental concept in MPEG-21 is that of a digital item (DI), which is described in Part 2 of MPEG-21. The DI is a structured object, represented as an XML-description, that contains the multimedia resources (or references), and metadata that describe these resources. Attempts have been made to adapt the DI to the health care sector as medical digital item (MDI) [18].

To identify a DI, Part 3, Digital Item Identification (DII), offers a shell where users can choose their own relevant identifier regime, for example patient identity or sensor identity.

A protected form of the DI is provided by the intellectual property management and protection (IPMP) components in Part 4 of MPEG-21. Parts of a DI can be encrypted and digitally signed, and hence confidentiality, authenticity and integrity of patient data can be provided assuming that the system, including keys, protocols and algorithms, is secure. However, the specific tools to encrypt and sign are not provided by the standard, and must therefore be implemented by the application or middleware components.

Expressing the digital rights, i.e., the rights to access specific data under given circumstances, is governed by Parts 5 and 6 of MPEG-21, the rights expression language (REL) and the rights data dictionary (RDD), respectively. Combined with IPMP the usage of patient data can be restricted, e.g., by giving a specific nurse the permission to view a specific patient's blood pressure during a specified time interval.

In order to make the DIs available on networks and devices with different capabilities, and to users with different preferences in various environments, Part 7 of MPEG-21 defines the digital item adaptation (DIA). In this context a user can refer to a person, a group or an organisation. Adaptation can be useful in a health care environment, e.g, to give different views of the same data on terminals with different screen sizes and network bandwidths [19].

Transmitting the DI as a potentially large XML-representation is not convenient on a resource and bandwidth constrained network like BSN. To address this, MPEG-21 Part 16, entitled "MPEG-21 binary format", describes how to binary encode the XML representation in order to significantly reduce the bit rate [7], [36]. The technology for encoding the XML representation is provided by ISO/IEC 23001 Part 1, "Binary MPEG format for XML" or shortly BiM [37]. The BiM encoding is standardised in MPEG-7 [38], [39] and adapted for use in MPEG-21.

In general, MPEG-21 is agnostic to which concrete algorithms are used; the digital items only refer to the algorithms, and contain the respective payload. An evaluation of different cryptographic algorithms for the use in WSN has been carried out elsewhere [40].

B. Representing Medical Data with MPEG-21

The Digital Item defined by MPEG-21 has previously been adapted for use in the health care sector [19] as medical digital item (MDI). The MDI can contain both the biomedical data as defined in Section III-A, and the metadata. The metadata originating from the sensor node comprise of (1) the sensor id; (2) the stream id; (3) time stamps; (4) sequence numbers; (5) descriptive metadata, like sampling rates; and (6) encryption-related data.

To keep the amount of data processed and sent from the sensor nodes at a minimum we introduce the concept of the *lightweight MDI* for the sensor (μMDI) that solely contains the necessary biomedical data and corresponding metadata. For privacy reasons data that can identify the patient are not included in the μMDI. Except the sensor id, all these data are represented encrypted, and packaged efficiently using BiM before being sent to the PDC. An example of a very simple μMDI is given in Fig. 3.

C. Efficient Encoding of XML

As XML-representations of data are known to have a huge overhead, the use of the Binary MPEG format for XML (BiM) is a necessity for the resource constrained sensors, thus the sensors will only have to process and transfer a binary encoded version of the μMDI. The major properties of BiM include that it represents a schema oriented encoding method using a preparsed, typed binary format; it also allows different refresh rates of sub-parts of the XML document. BiM uses a tree representation of XML, where the tree nodes can be addressed, and operations containing the payload can be applied. A BiM-encoded template reduces the size of the XML-code by 90-95% [36].

On the source node the data are encoded by an automaton, which can be generated from the XML schema describing the μMDI that is used on this sensor node. On the PDC, the code to decode the μMDI is generated from the same XML schema. Transfer nodes and other sensor nodes are not aware of the MDI schema, since the content is not supposed to be processed there.

To further reduce the amount of transmitted data, not all metadata need to be included in every packet. This is possible since BiM allows different refresh rates of the XML document. Descriptive metadata, such as bitrate, could be sent out less often than critical metadata like time stamps, and stream id.

```
<DIDL xmlns="urn:mpeg:mpeg21:2002:02
-DIDL-NS" xmlns:dii="urn:mpeg:mpeg21:
2002:01-DII-NS">
 <Container id="test">
  <Item id="myitem">
   <Descriptor>
    <Statement mimeType="text/xml">
     <dii:Identifier>
      urn:grid:a1-abcde-9873216540-f
     </dii:Identifier>
    </Statement>
   </Descriptor>
   <Descriptor>
    <Statement mimeType="text/xml">
     <dii:Type>
      urn:sensor:bloodpressure
     </dii:Type>
    </Statement>
   </Descriptor>
  </Item>
  <Item id="bloodpressure">
   <Component id="systole">
    <Resource mimeType="text/plain">
     160
    </Resource>
   </Component>
   <Component id="diastole">
    <Resource mimeType="text/plain">
     80
    </Resource>
   </Component>
  </Item>
 </Container>
</DIDL>
```

Fig. 3. Example of a simple MDI for blood pressure in XML.

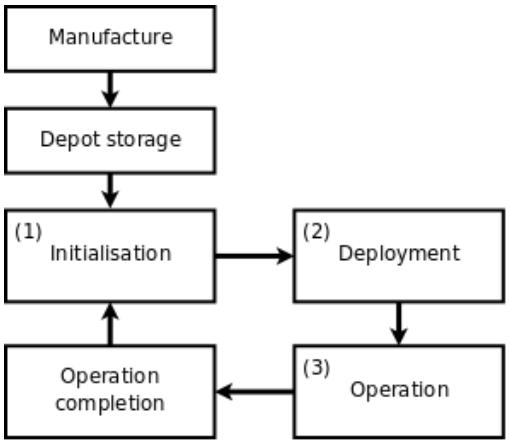


Fig. 4. Lifecycle of a biomedical sensor node (adapted from [23]).

VII. Proposed Architecture

Our architecture builds on the medical digital items (MDI) introduced in a health care environment for patient monitoring systems [18], and suggested for BSN [19]. Our architecture envisages that all medical data use the MDI in the presentation layer of all channels of the generic model. When used between all components this enables security and adaptivity of health

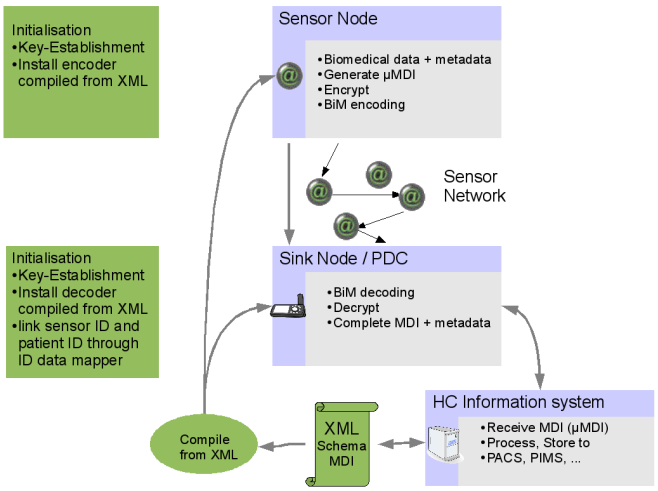


Fig. 5. Proposed architecture showing the initialisation and deployment phases (green) and operation phase (blue).

care data, including medical data and metadata. Generally, all medical data sent over the channels are encoded using the MDI schema, while the components use the tools defined by MPEG-21 to handle the medical data.

As outlined previously the properties of Channel A of the generic model require a refinement of the general architecture due to resource limitations and other properties of this channel. We will elaborate on how to implement our architecture for Channel A by the means of a BSN in the following.

Even though the BSN has limited resources we encode the biomedical data and the relevant metadata into MDI containers to transfer these from the sensor node to the PDC. Due to the resource constraints in BSN we use additionally BiM to reduce the data rate, and select carefully which portions of the medical data to transfer at which rate, and which data to protect.

Fig. 4 shows a diagram of the entire life cycle of a sensor node [23], which we have adapted to the health care area. We focus on the three phases that are numbered in this diagram, namely the initialisation, the deployment, and the operation of a sensor node.

For a more detailed view on these three phases, including the overall architecture and data flow, we refer to Fig. 5. During the initialisation phase the sensor nodes receive the appropriate software and keys in a secure manner shown in the green elements of this diagram. The initialisation is facilitated by installing software including the right credentials on the sensor node. This software is compiled from the XML schema to produce the μMDI efficiently. The sink-node receives the credentials and decoding-software accordingly, while there is no need to install extra software or credentials on the transfer nodes. The relationship between patient and sensor node is established in the deployment phase through the ID data mapper.

During the operation phase, denoted as the blue elements in

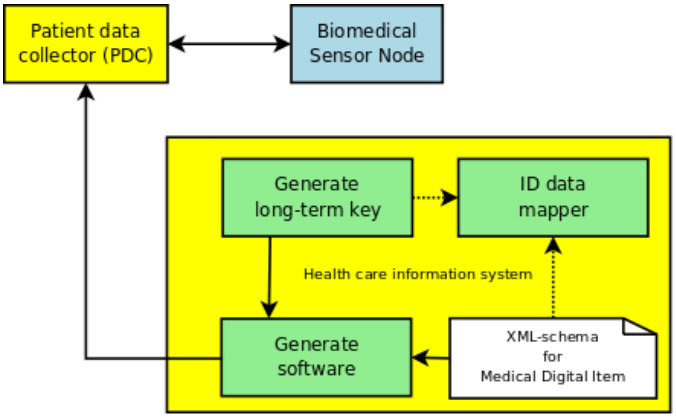


Fig. 6. Initialisation of a sensor node.

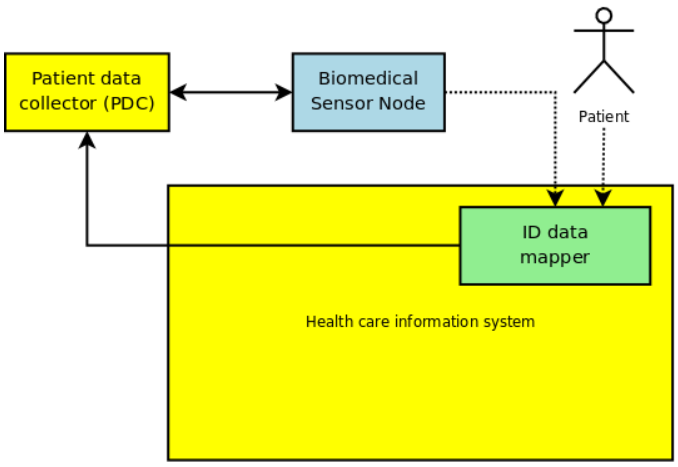


Fig. 7. Deployment of a sensor node.

Fig. 5, the medical data and metadata are sent from the sensor nodes to the PDC, and further to the other channels. The data are produced in the sensor node and forwarded as μMDI to the sink node. There, the μMDI is completed to form a full MDI and forwarded to the HCIS and the PDAU.

A. Initialisation of Sensor Nodes

Prior to deployment of the sensor nodes, the initialisation phase performs the following tasks: (a) key-establishment; (b) installation of XML-generated code on the sensor nodes. Fig. 6 illustrates the initialisation.

Due to the limited computing resources on the sensors, we avoid the usage of public key algorithms, which is reflected also for the key-establishment [41]. We propose that each sensor and the PDC share a long-term, pair-wise, symmetric key that is used to establish session keys.

The long-term key has to be securely pre-distributed from the hospital infrastructure. This provides a challenge, since, on most sensors, the communication interface is wireless, and the pre-distribution will be broadcast for everyone to listen. As a countermeasure to eavesdropping, the key establishment could be performed using a special device, e.g., installed in a Faraday cage, which will shield the electromagnetic signals so that a potential eavesdropper is unable to intercept the long-term key. The long-term key is supposed to last the entire lifetime of a sensor, or until it is compromised or exchanged routinely to renew the keys. Hence the initialisation procedure does not have to be performed every time a sensor is deployed on a patient.

The pair-wise symmetric *session* key will then be established by using a key establishment protocol that utilises a pre-distributed symmetric key, such as Authenticated Key Exchange Protocol 2 (AKEP2) [42], [43]. Thus, a node will only be able to read messages encrypted by the PDC with their shared key, and not messages encrypted by other nodes. The PDC will be able to read messages encrypted by all nodes that have their keys securely stored in the PDC.

Due to the limited capabilities of the sensor nodes these are unsuited to handle the full complexity of encoding and parsing XML-documents. A more viable approach for BSN is to deploy software into the sensor nodes during the initialisation phase that is able to produce a μMDI from a template. More powerful devices without the constraints of sensor nodes, e.g., the PDC, will then receive and process these data using the full capabilities of XML.

B. Deployment of Sensor Nodes

In the deployment phase all identities are linked together within the ID data mapper. A schematic view on this phase is given in Fig. 7. The dotted lines indicate that the ID of the entity is sent to the ID data mapper. The PDC stores the sensor and stream identities and links them to the patient identity during deployment of the sensor. We distinguish between stream ID and sensor ID to support multi-sensors capable of handling several data streams simultaneously. Before attaching a sensor to a patient a manually initiated procedure assigns sensor and stream identities in order to identify each stream uniquely. By also including an examination identity, the patient identity and examination are linked to the sensor and streams. This information is resident in the PDC, and is communicated to the hospital infrastructure via an MDI.

C. Operation of Sensor Nodes

During the operation of a sensor node the following workflow takes place: (1) *Measurement*, the sensor measures biomedical data from a patient, (2) *packaging*, the biomedical data and the associated metadata are encrypted, encoded, packaged and signed in the μMDI template, and finally, (3) *sending*, the μMDI is sent to the PDC.

Steps (2) and (3) are illustrated in Fig. 8. The encryption in Step (2) uses a symmetric session-key that is shared only between the originating sensor and the PDC. This secured μMDI is then binary encoded using BiM. The sensor node sends the encrypted, encoded and signed version to the PDC, possibly via transfer nodes which will not be able to learn any of the contents since these are not in the possession of the decryption key.

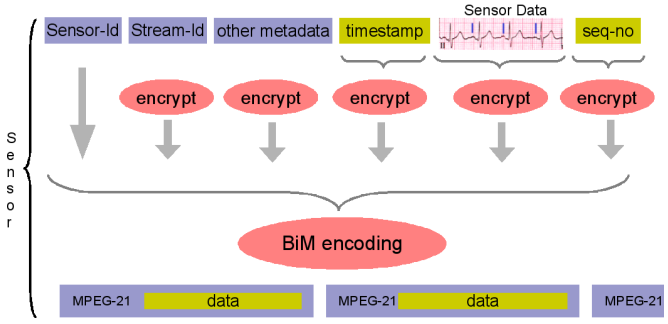


Fig. 8. Operational phase of the sensor node. Biomedical and metadata are encrypted, encoded and packaged into a μMDI.

After the μMDI arrives at the PDC it is decoded, decrypted and the signature is verified. The content of the μMDI, all relevant context information, such as references to the XML-schema in use, and patient data are then aggregated to produce a full MDI. This MDI might also include data from other sources about the same patient, before being sent further along the channels described in the generic system model.

VIII. Discussion

Our framework uses an international and open multimedia standard for end-to-end content management to meet the security goals of data confidentiality and integrity of medical data in patient monitoring systems using a BSN. An important argument for our choices is to use MPEG-21 as a mechanism in the entire work flow of the health care information system. However, due to the resource restrictions in BSN we need to show that the use of MPEG-21 is viable, and to what extent our architecture meets the security requirements stated previously.

A. Security

The security mechanisms proposed in our architecture are applied on the presentation layer, and address integrity and confidentiality requirements. Threats on the network layer or below, such as routing attacks, denial of service attacks or hardware failures are not addressed, nor are attacks on cryptographic primitives, since we assume the Dolev-Yao attacker model. Further, we cannot exclude that an attacker might receive a limited amount of information from the existence of messages and additional knowledge that the attacker might have from the context or from access to the patient. Additionally, message rate, message size and sensor frequency might reveal confidential information. Therefore, measures to achieve 'transactional confidentiality' [44] must also be considered.

We claim that the use of MPEG-21 can protect against confidentiality threats, such as eavesdropping the original medical data, by encryption using a symmetric key. Since the key establishment is performed using physical security measures the medical data and most of the metadata remain as secure as the employed cryptographic method permits, and the keys are not compromised.

Since the strength of the employed cryptographic methods is limited due to the resource limitations of the sensor nodes, the μMDI does not contain data which directly identify the patient. Note that this is in contrast to frameworks that do not consider the use of BSN explicitly [22], where the patient ID is not even protected.

To address the integrity of the medical data and metadata the MDI is protected by an encrypted hash value, as employed in MPEG-21 IPMP. This and the use of sequence numbers for packets enable the detection of injected, re-played, deleted or modified data packets. Altering and injection of messages would be detected since the Dolev-Yao-attacker would not be able to correctly sign messages.

B. Encryption Scheme

The encryption methods as such are not part of MPEG-21. In principle, all schemes and implementations can be used in combination with MPEG-21. However, encryption schemes are a vital part of the protection of medical data and metadata. The use of symmetric keys, rather that public/private key pairs, can inflict potential issues of authentication, key-distribution and key-management [45]. However, we argue that our proposed scheme resolves these issues since, in contrast to general WSN, both initialisation and deployment are performed in a physically controlled environment by humans.

Pair-wise symmetric key pairs allow data source authentication. Since each sensor only needs to store one key, the memory requirements on the sensor are limited to the key-size. On the PDC the memory requirements are proportional to the number of sensors that are connected to it. Assuming a symmetric key-size of 128 bits, 20 patients with 25 nodes each, the key storage will require around 8 kB on the PDC. Even if the pre-distributed keys might be used for a long time, and be subject to cryptanalysis by an adversary, a brute-force attack is considered to be infeasible on keys with 128 bits.

C. Suitability for BSN

Since the sensor nodes are resource restricted the use of XML on the sensor nodes is not viable due to space constraints. Instead BiM-encoding is used. On the sensor nodes parsing of message content is avoided. As a consequence, XML is processed only outside the sensor nodes.

The software to BiM-encode the MDIs in the sensor node is generated outside the sensor node from XML schemas, and uploaded to the sensor node during the initialisation phase. This software is typically implemented as a rather simple automaton which has a small fingerprint. Since the μMDI structure is constant while operating a sensor node, a bitstream binding language [36] which could provide a more flexible framework is not necessary.

Compared to sending the medical data and metadata in a fixed packet format with pre-defined fields we gain much flexibility with MPEG-21 to the cost of an overhead. Additionally to the payload the BiM-encoded messages contain path information which increases the packet length. Benchmarks show that we can expect a considerable overhead for the

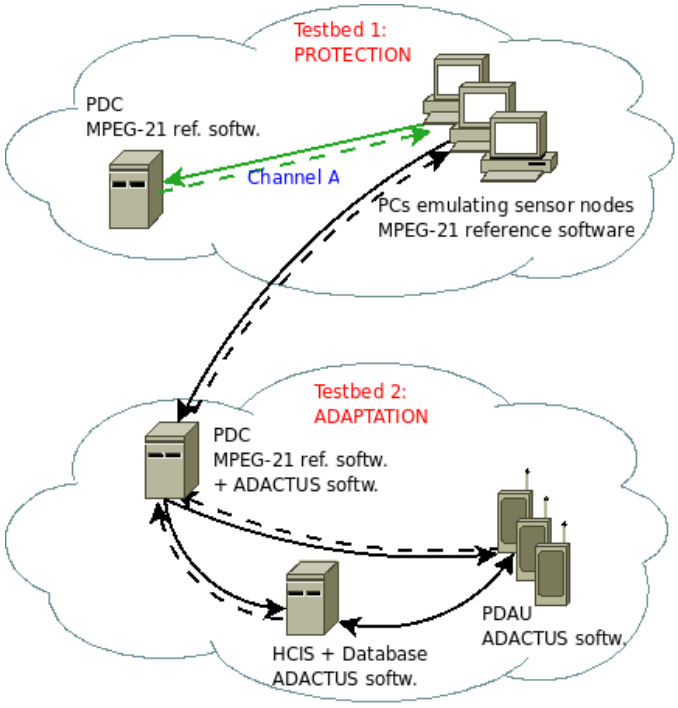


Fig. 9. Test bed for evaluating MPEG-21 for use in Patient Monitoring Systems and BSN.

general case [38]. Therefore the layout of the XML-tree to represent the μMDI has been carefully designed in order to minimise the overhead in the BiM-encoding. Also the ratio between payload, i.e., measured data, and the data necessary to secure the μMDI must be balanced.

D. Trust

Sensor nodes can fail by other means than communication, e.g., deliver bogus data, be misplaced on the body, etc. While our proposed framework does not protect against these threats, we recognise the importance of a sanity checks of data in health care applications. For proofs and adequate evidence about creator, creation, and historical process of data the term evidential value [46] is used. Methods to maintain the evidential value must be employed on devices that have enough capabilities to perform these operations. Our architecture architecture contributes to trust by using the measures of authentication, hash values and encryption as outlined in Section VIII-A. Other elements of the evidential value can be embedded in the μMDI when the sensor is able to provide these.

E. Evaluation Test Bed

To evaluate the different aspects of MPEG-21 we have developed a test bed shown in Fig. 9. This test bed consists of one part for evaluating properties of data protection and security, especially for Channel A; and of another part for evaluating properties of adaptation which is most relevant for presenting data at the PDAU. In Test Bed 1 we use parts of the reference software for MPEG-21 [47], and reference software for MPEG-7 [48] for BiM encoding running on PCs. In the Test Bed 2 we also employ software from ADACTUS [49] to evaluate adaptation issues which are beyond the scope of our paper. Both parts together implement a test bed that represents a patient monitoring system with all components and channels of the generic system model.

During our experiments with the test bed the reference software for MPEG-21 could be installed and used without greater obstacles, while the reference software for MPEG-7 BiM needed several XML files which were no longer available at their original locations. Both software packages use XML code that is partially incompatible, leading to the need of developing XML files that can be used in both reference software packages. We succeeded in encoding and decoding μMDI messages, and could confirm the estimates for the compression ratio using BiM as known from the literature [36].

Since the reference software for MPEG-7 BiM is demonstration software only, the production of a μMDI in our test bed is done via XML, instead of directly encoding the sensor data as outlined in Section VI-C.

IX. Conclusion and Future Work

We propose a framework for BSN that uses MPEG-21 for transmitting the biomedical data from sensor nodes to the hospital infrastructure. The tools within MPEG-21 have the capabilities of encrypting the patient data and assigning detailed rights to them. In addition, it is suitable for handling multimedia data on different devices and networks, which can be used to enhance the perceived quality of service (QoS) for a user. The proposed BiM-encoding technique facilitates a way to incorporate MPEG-21 resources in a resource-constrained BSN.

The scalability in terms of larger networks with many sensor nodes and user terminals, denoted as clients, can easily be handled in the application and presentation layers. Furthermore, implementation of our framework in the test bed and on real sensor nodes, with careful design of the μMDI will provide an efficient way of optimising wireless data transmission, data processing, power consumption, and memory usage in the sensor nodes with adequate security mechanisms. Incorporating Part 7 of MPEG-21 (DIA) into our security framework can be considered in future.

We anticipate a large scale testing of the proposed framework using the test bed described in Section VIII-E. The simulation of a BSN and evaluating packet size, overhead from the BiM encoding, and resource consumption will be useful prior to deploying such a system.

We also envisage the use of formal methods [50] as a proof of the correctness of our framework, its implementation and the employed security protocols. This includes the analysis of attacks on the BiM-encoded packets under given assumptions, authentication, and integrity of the medical data.

To study the impact of small footprint of the software to be deployed in the initialisation phase, an implementation on real sensor nodes (motes) can be considered. For generating the software code for the sensor nodes, a framework for code

generation [51] that allows generation of code from an abstract model to several potential sensor node types can be used. We are confident that simulations in the test bed, and the implementation of the framework on real BSN will open for a secure deployment of wireless technologies in health care applications.

ACKNOWLEDGEMENTS

This work is funded by the SAMPOS project in the VERDIKT programme of the Norwegian Research Council. We appreciate the help of Thomas Skjølberg and Peder Drege at ADACTUS. The authors wish to thank Habtamu Abie, Arne-Kristian Groven, and Lothar Fritsch for contributions and discussions in previous versions of this paper.

REFERENCES

[1] Wolfgang Leister, Truls Fretland, and Ilangko Balasingham. Use of MPEG-21 for security and authentication in biomedical sensor networks. *Proc. ICSNC'08, International Conference on Systems and Networks Communication*, 0:151–156, 2008.

[2] Integrating Healthcare Enterprise. http://www.ihe.net. Last accessed: May. 15, 2009.

[3] S.C. Chu. From component-based to service oriented software architecture for healthcare. In *Proc. 7th Annual International Workshop on Enterprise networking and Computing in Healthcare Industry, HEALTHCOM 2005*, pages 96–100, 2005.

[4] Eric Newcomer and Greg Lomow, editors. *Understanding SOA with Web Services*. Addison Wesley, 2005. ISBN 0-321-18086-0.

[5] Digital Imaging and Communications in Medicine (DICOM). http://medical.nema.org/dicom/, 2008. Last accessed May 15, 2009.

[6] Ilangko Balasingham, Halfdan Ihlen, Wolfgang Leister, Per Røe, and Eigil Samset. Communication of medical images, text, and messages in inter-enterprise systems: A case study in Norway. *IEEE Transactions on Information Technology in Biomedicine*, 11(1):7–13, 2007.

[7] I. Burnett, F. Pereira, R. van de Walle, and R. Koenen, editors. *The MPEG-21 Book*. John Wiley & Sons, 2006. ISBN 0-47001011-8.

[8] ISO/IEC TR 21000-1: 2004. information technology - multimedia framework (MPEG-21) - part 1: Vision, technologies and strategy, November 2004.

[9] The Government of Norway. Act of 18 may 2001 no. 24 on personal health data filing systems and the processing of personal health data (personal health data filing system act). Stortinget, 2002.

[10] Article 29 Data Protecting Working Party of Directive 95/46/EC. Working document on the processing of personal data relating to health in electronic health records (EHR), 2007. Adopted on 15 February 2007, European Commission, Directorate General Justice.

[11] J.P. Walters, Z. Liang, W. Shi, and V. Chaudhary. Wireless sensor network security: A survey. In Yang Xiao, editor, *Security in Distributed, Grid, and Pervasive Computing*, chapter 17. Auerbach Publications, CRC Press, 2006.

[12] H. Yang, H. Luo, S. Ye, S. Lu, and L. Zhang. Security in mobile ad hoc networks: Challenges and solutions. *IEEE Wireless Communications*, 11(1):38–47, 2004.

[13] R. Savola and J. Holappa. Self-measurements of the information security level in a monitoring system based on mobile ad hoc networks. In *Proc of the 3rd International Workshop in Wireless Security Technologies 2005 (IWWSXT'05)*, 2005.

[14] R.S.H. Istepanian, E. Jovanov, and Y.T. Zhang. Guest editorial introduction to the special section on m-health: Beyond seamless mobility and global wireless health-care connectivity. *IEEE Transactions on Information Technology in Biomedicine*, 8(4):405–414, 2004.

[15] L Schwiebert, S.K.S. Gupta, and J. Weinmann. Research challenges in wireless networks of biomedical sensors. In *Seventh Annual ACM/IEEE International Conference on Mobile Computing and Networking*, 2001.

[16] IEEE 802.15.6 Body Area Network. https://mentor.ieee.org/802.15/documents, 2009. Last accessed May 15, 2009.

[17] IEEE 1451.5 draft standard for a smart transducer interface for sensors and actuators - wireless communication protocols and transducer electronic data sheets (TEDS) formats, 2006. March 26, 2007.

[18] M. Landén. An MPEG-21 approach to creating the first multimedia electronic patient journal system. Master's thesis, NTNU, 2003.

[19] Arne Lie, Knut Grythe, and Ilangko Balasingham. On the use of the MPEG-21 framework in medical wireless sensor networks. In *Proc. of the IEEE 1st Int Symp on Applied Sciences in Biomedical and Communication Technologies (ISABEL)*, Aalborg, Denmark, 25.-28. October, 2008.

[20] W. Leister, H. Abie, A.-K. Groven, T. Fretland, and I. Balasingham. Threat assessment of wireless patient monitoring systems. In *Proc. ICTTA'08*, Damascus, Syria, 2008.

[21] Georg A Brox. MPEG-21 as an access control tool for the national health service care records service. *Journal of Telemedicine and Telecare*, 11 Suppl 1:23–5, 2005.

[22] Anastasios Fragopoulos, John Gialelis, and Dimitrios Serpanos. Security framework for pervasive healthcare architectures utilizing MPEG-21 IPMP components. *International Journal of Telemedicine and Applications*, 2009:1–9, 2009.

[23] D.W. Carman, P.S. Kruus, and B.J. Matt. Constraints and approaches for distributed sensor network security (Final). *DARPA Project report, (Cryptographic Technologies Group, Trusted Information System, NAI Labs), September*, 1, 2000.

[24] V. Gupta, M. Millard, S. Fung, Zhu Yu, N. Gura, H. Eberle, and S.C. Shantz. Sizzle: a standards-based end-to-end security architecture for the embedded internet. In *Third IEEE conf on Pervasive Computing and Communications, PerCom 2005*, pages 247–256, 2005.

[25] D. Dolev and A.C. Yao. On the security of public key protocols. In *Proc. of the IEEE 22nd Annual Symposium on Foundations of Computer Science*, pages 350–357, 1981.

[26] Srdjan Čapkun, Jean-Pierre Hubaux, and Levente Buttyán. Mobility helps security in ad hoc networks. In *MobiHoc '03: Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, pages 46–56, New York, NY, USA, 2003. ACM.

[27] Stanley R. M. Oliveira and Osmar R. Zaïane. A privacy-preserving clustering approach toward secure and effective data analysis for business collaboration. *Computers & Security*, 26(1):81–93, 2007.

[28] R. Arnesen, J. Danielsson, J. Vestgården, and J. Ølnes. Wireless health and care security requirements. Research note DART/01/05, Norsk Regnesentral, 2004.

[29] Hans Jakob Rivertz. Bluetooth security. Research note DART/05/05, Norsk Regnesentral, 2005.

[30] Y. Hu, A. Perrig, and D. Johnson. Packet leashes: A defense against wormhole attacks in wireless ad hoc networks. In *Proc. INFOCOM 2003*, pages 1976–1986, 2003.

[31] Y. Hu, A. Perrig, and D. Johnson. Rushing attacks and defense in wireless ad hoc network routing protocols. In *Proc. WISE 2003*, pages 30–40, 2003.

[32] J. Newsome, E. Shi, D. Song, and A. Perrig. The sybil attack in sensor networks: Analysis and defenses. In *Proc. IPSN 2004*, pages 259–268, 2004.

[33] J. Douceur. The sybil attack. In *Proc. IPTPS 2002, LNCS Vol. 2429*, pages 251–260, 2002.

[34] C. Karlof and D. Wagner. Secure routing in sensor networks: Attacks and countermeasures. In *Proc. SNPA 2003*, pages 113–127, 2003.

[35] L. Lazos and R. Poovendran. Serloc: Secure range-independent localization for wireless sensor networks. In *ACM Workshop on Wireless Security*, 2004.

[36] J. Thomas-Kerr, I. Burnett, and P. Ciufo. Bitstream binding language - mapping XML multimedia containers into streams. In *IEEE International conference on Multimedia and Expo, 2005, ICME 2005*, pages 626–629, 2005.

[37] P. de Cuetos, C. Seyrat, and C. Thienot. BiM white paper. http://www.chiariglione.org/mpeg/technologies/mpb-bim/index.htm, January 2006. Last accessed May 15, 2009.

[38] U. Niedermeier, J. Heuer, A. Hutter, W. Stechele, and A. Kaup. An MPEG-7 tool for compression and streaming of XML data. In *Proc. 2002 IEEE Intl. Conf. on Multimedia and Expo*, pages 521–524, 2002.

[39] S. Devillers, C. Timmerer, J. Heuer, and H. Hellwagner. Bitstream syntax description-based adaptation in streaming and constrained environments. *IEEE Transactions on Multimedia*, 7(3):463–470, June 2005.

[40] R. Roman, C. Alcaraz, and J. Lopez. A survey of cryptographic primitives and implementations for hardware-constrained sensor network nodes. *Mobile Networks and Applications*, 12(4):231–244, 2007.

[41] A. Perrig, R. Szewczyk, J.D. Tygar, V. Wen, and D.E. Culler. Spins: Security protocols for sensor networks. *Wireless Networks*, 8:521–534, 2002.

[42] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 2001.

[43] Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In *CRYPTO '93: Proceedings of the 13th annual international cryptology conference on Advances in cryptology*, pages 232–249, New York, NY, USA, 1994. Springer-Verlag New York.

[44] S. Pai, S. Bermudez, S.B. Wicker, M. Meingast, T. Roosta, S. Sastry, and D.K. Mulligan. Transactional confidentiality in sensor networks. *Security & Privacy, IEEE*, 6(4):28–35, 2008.

[45] Chieh-Yih Wan, Mark Yarvis, and Jens Mache. Lightweight key distribution for sensor networks. http://www.freshpatents.com/Lightweight-key-distribution-and-management-method-for-sensor-networks-dt20081127ptan20080292105.php. Last accessed: May. 15, 2009.

[46] Jianquiang Ma, Habtamu Abie, and Torbjørn Skramstad. Preservation of trust and security in long-term record management. In *Proc. 6th Annual Conference on Privacy, PST2008, Security and Trust, Graduate Student Symposium, Fredericton, New Brunswick, Canada*, 2008.

[47] ISO/IEC 21000-8: 2008. information technology - multimedia framework (MPEG-21) - part 8: Reference software, 2008.

[48] ISO/IEC 15938-6: 2003. information technology - multimedia content description interface (MPEG-7) - part 8: Reference software, 2003.

[49] ADACTUS. http://adactus.no, 2009. Last accessed May 15, 2009.

[50] Anders Moen Hagalisletto, Lars Strand, Wolfgang Leister, and Arne-Kristian Groven. Analysing protocol implementations. In Feng Bao, Hui Li, and Guilin Wang, editors, *ISPEC*, volume 5451 of *Lecture Notes in Computer Science*, pages 171–182. Springer, 2009.

[51] M.M.R. Mozumdar, F. Gregoretti, L. Lavagno, L. Vanzago, and S. Olivieri. A framework for modeling, simulation and automatic code generation of sensor network application. In *Proc. 5th Annual IEEE Communication Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, SECON'08*, pages 515–522, 2008.

A Distributed Reputation System for Super-Peer Desktop Grids

Peter Merz, Florian Kolter, Matthias Priebe
Distributed Algorithms Group
Department of Computer Science
University of Kaiserslautern
D-67663 Kaiserslautern, Germany
Email: {pmerz,f_kolter,priebe}@informatik.uni-kl.de

***Abstract*—Desktop Grids leverage otherwise unused resources of idle desktop computers, providing vast amounts of cumulative computational power. However, resource sharing in Peer-to-Peer environments with selfish participants suffers from the free-riding phenomenon unless the environment provides appropriate countermeasures. In Peer-to-Peer-based Desktop Grids, cooperative participants require protection against free-riding job distributors. In this article, we present a decentralized shared-history reputation mechanism designed for use with Desktop Grids built on dynamic super-peer structures. Embedded in a distributed Desktop Grid workflow model, our concept promotes reciprocity and discourages free-riding. In simulations based on real-world network delay and workload information, we show that our concept offers a considerable speedup over non-distributed computation while effectively thwarting free-riding and maintaining the system's commitment to robustness and scalability.**

***Keywords*—Peer-to-Peer; Distributed Computing; Desktop Grids; Free-Riding; Reputation Systems**

I. Introduction

For decades, supercomputers have been the method of choice in cases where the solution to a problem depended on the availability of massive computational power. They provide computational resources to an extent that exceeds that of ordinary desktop computers by orders of magnitude. However, supercomputers are costly to build and operate, and there may be considerable lead time until one becomes available.

On the other hand, desktop PCs are occasionally idle, spending unused processor cycles for no return. This is especially true for multi-core CPUs in which one or more cores may find themselves underloaded for considerable periods of time. Harvesting these idle cycles to form a virtual supercomputer that offers its cumulative computation power to the general public represents a promising alternative to conventional supercomputing. Inside this virtual computation engine, a self-organizing system would interconnect the participating machines, splitting large jobs into small tasks, distributing the tasks among the desktop machines, and reassembling the individual results into a single one as it would have been computed had the job been processed by a supercomputer in the traditional way. As in Grid computing, resources are connected via wide-area networks, in particular the Internet. A major difference lies in the way resources are deployed and administrated. In Grid Computing, resources are supervised by trustworthy administrators and have been acquired with the intention of being remotely accessed to handle distributed work. They are commonly available for long periods of time.

On the contrary, desktop PCs operated by scientific institutions, companies, public authorities, and private households are purchased for the purpose of performing work locally. A desktop computer is usually less powerful than a computational Grid resource, and the quality of its Internet connection may vary considerably [2]. The remote utilization of idle cycles from these devices has begun as volunteer computing. In volunteer computing, computer owners are asked to voluntarily donate idle CPU time. In this model, a user would run problem-specific software that receives a job of a well-known kind from a pre-defined server, computes it in its idle periods, and returns the results to the server. A prominent example of this approach is SETI@home [3] which analyzes radio signals from outer space for hints of extraterrestrial life.

For a number of applications, Peer-to-Peer (P2P) networks have emerged as a viable alternative to client/server schemes such as SETI@home. In their fully distributed form, P2P networks contain no single point of failure, scale to millions of participants – called *peers* due to the absence of a hierarchy –, and include properties of self-organization that improve their robustness. Distributed computing in a P2P setting enables every participant to initiate the computation of a job by asking its fellow peers to join the computing effort. Because of its focus on desktop resources, a system that taps the idle computation resources provided by desktop units, is based on a P2P network and enables everyone to submit arbitrary jobs for distributed computation is referred to as a Desktop Grid [4], [5], [6], [7], [8]. The cumulative potential leveraged by a Desktop Grid grows with the number of resources connected to it. Hence, a Desktop Grid benefits from the capability to integrate a heterogeneous range of computers in a dynamic, volatile, decentralized environment. The requirement to support a substantial number of participants emphasizes the importance of scalability in this context. The construction of Desktop Grid structures on P2P overlays improves scalability in large-scale settings [5], [9]. It is further improved by the concept of *super-peers* which combines advantageous proper-

ties of client/server systems and unstructured P2P topologies [10]. Unlike ordinary Grids, peers participating in Desktop Grids bring all the resources including the infrastructure. Since overlay dynamics are explicitly accounted for, joining a Desktop Grid does not require significant administrative efforts besides running the software which is required for access.

While a Desktop Grid offers attractive features, the peers are operated by humans whose goals may be selfish rather than altruistic. In the purely selfish case, a peer may join the Desktop Grid, submit its job, wait for its completion, and once finished, leave the system. In this case, that particular peer does not contribute any resources to the system but exploits it for its own good: it is a *free-rider*. While a certain amount of free-riders might be tolerated, the system would break down when there are too many because the demand for resources exceeds the supply. Hence, a Desktop Grid needs to deploy effective measures to thwart free-riding.

In this article, we propose a reputation system tailored for distributed Desktop Grids that are built on super-peer structures. This reputation system mitigates the adverse effects of selfish, uncooperative peer behavior while maintaining the commitment to meet scalability and robustness requirements. It permits cooperative peers to detect and eventually avoid free-riding job distributors. It is accompanied by a workflow model which adapts the common distributed computing workflow to a super-peer setting. By means of experiments, we demonstrate the gains over configurations with no protection for cooperative peers. This article is an extended version of [1], adding an investigation of the collusion phenomenon and a series of experiments that quantify the overhead incurred by the proposed reputation system, the numbers of accepted and declined task requests, and the effects of varying free-rider fractions. Moreover, it contains enhanced coverage of reputation-related issues in P2P networks and a summary of aspects relevant to the construction of super-peer topologies.

The remainder of this article is organized as follows. Section II sketches the architecture of super-peer overlay topologies along with a distributed algorithm that creates and maintains a topology of this type. Section III outlines issues caused by rational-selfish peer behavior. Section IV presents the proposed Desktop Grid model which consists of workflow, reputation, and incentive components. Section V describes the scenario, setup and outcome of simulation experiments with the proposed model. Section VI discusses related efforts. The article concludes with directions for future research in Section VII.

II. Overlay

Super-peers are peers that act as relays for a number of attached common peers called *edge peers*, while at the same time, they form a network of equals among themselves. That way, the entire P2P network may remain connected over a substantially smaller number of links than before [10], [11]. Besides enhancing scalability, super-peers may route communication between edge peers that cannot directly communicate with each other due to the presence of firewalls. An example

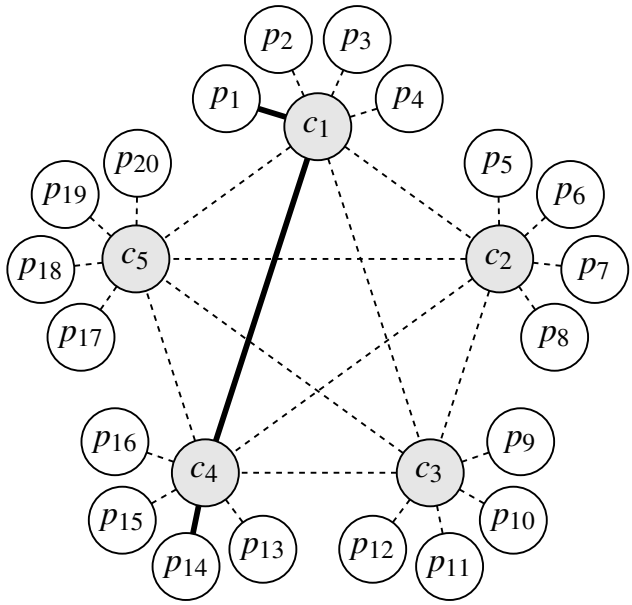


Fig. 1. Example of a P2P network with selected super-peers

for this feature is depicted in Figure 1 where edge peer p_1 wishes to communicate with edge peer p_{14}. For the first hop, p_1 sends the message to its super-peer, c_1, which forwards it to the destination's super-peer, c_4. Finally, c_4 passes the message on to the destination peer, p_{14}. In a fully meshed super-peer overlay as shown in Figure 1, the network diameter is only 3 hops.

This section briefly describes the Super-Peer Selection Algorithm (SPSA) [12], [13], a distributed algorithm that creates and maintains a delay-optimized, self-organizing, fully meshed super-peer overlay. SPSA has been conceived to establish a P2P overlay on top of which a Desktop Grid can be built. The algorithm runs continuously on every peer in the overlay. It promotes suitable peers to super-peer level and makes nearby edge peers connect to them. No central instance is required for SPSA to perform. The first peer to join the overlay instantly becomes a super-peer, while all peers joining subsequently receive edge peer status at join time.

SPSA strives to minimize the number of connections a super-peer needs to keep up. In a fully meshed overlay, a super-peer maintains one connection to each of its edge peers and one to every other super-peer, respectively. Let C be the set of super-peers. With n peers in the system, the number of connections per super-peer is minimized for $|C| = \sqrt{n}$ [13]. Every super-peer can estimate the current number of peers in the overlay by counting the number of connections held to other super-peers; let this number be h. Also, every super-peer knows the number of edge peers attached to it; let this number be z. Now, a super-peer changes its role and downgrades to edge peer level if $z < 0.5 \cdot h$. Conversely, if a super-peer finds itself overloaded by determining that $z > 2 \cdot h$ holds, it picks one of its edge peers and promotes it to super-peer level so that some of the remaining edge peers switch to the new

super-peer. To minimize the end-to-end message delay, edge peers always connect to the closest super-peer, overloaded super-peers always pick the edge peer for promotion that minimizes the sum of delays regarding its connections, and all super-peers regularly check if one of their edge peers can perform their duty at lower cost in terms of network delay. This approach scales well when delays are estimated using network coordinates [14], [15].

III. Peer behavior

A free-rider is an individual who does not provide any resources to a system but exploits it selfishly [16], [17], [18], [19], [20], [21]. While an altruistic setting may support a certain extent of free-riding, it is expected to collapse if exceedingly many peers choose to act as free-riders. Therefore, Desktop Grids benefit from mechanisms which curb free-riding. One important aspect in this context is *reputation*. Reputation conveys information about a peer's past conduct to influence expectations of its future behavior [22]. It quantifies the risk in trusting others. By disseminating aggregated information on peer behavior, a reputation scheme encourages selfish peers to act in a fair manner [17]. That way, peers may avoid cooperation with other peers that exhibit poor reputation, providing an incentive to peers to accumulate positive feedback by acting accordingly. Generally, reputation aggregates feedback information. Without a reputation mechanism, a peer's abusive behavior toward a certain peer will not harm the misbehaving peer when interacting with other peers.

It is assumed that every peer has a private type which determines its strategy. The model distinguishes between three types of peers [16], [23], [24], [25], [26]:

- *Altruists* serve to the best of their capabilities every request issued by other peers without asking for reciprocity. Hence, exhibiting the opposite behavior to complete selfishness, altruists always obey and cooperate with other peers regardless of external rewards. Thus, altruists require no protection against free-riders.
- Selfish participants expect to receive a benefit from the Desktop Grid:
  - *Collaborators* are generally willing to cooperate if they can expect their counterpart to eventually reciprocate. In particular, they are ready to reserve resources for use by others provided that these resources are not occupied by free-riding users.
  - *Free-riders* are selfish participants that do not wish to contribute resources to the Desktop Grid; rather, they strive to exploit the computational resources of unsuspecting or unprotected peers.

Collaborators not protected by a reputation system cannot tell about the past behavior of peers they are about to interact with. Hence, their resources are easily occupied by free-riders that exploit this lack of protection. Technically, while both accept any job, altruists and unprotected collaborators differ in that altruists do not mind being exploited because they gain utility from the mere act of unconditionally donating their resources to others [25]. On the contrary, collaborators seek reciprocity and shun exploitation. A reputation system enables collaborators to assess the risk in dealing with other peers, creating the opportunity to choose whether to engage in or to abstain from an interaction with a given remote peer, and generating incentives for peers to adopt reciprocity in their own behavior.

The interaction between two peers is commonly modelled as an instance of the *Iterated Prisoner's Dilemma* where both players can pick one of the strategies *cooperate* or *defect* (i.e. attempt to exploit the other player) independently from each other [23], [26], [27]. The situation depicts a conflict between group rationality and individual rationality: two rational players will make the game result in the game's only Nash equilibrium (defect, defect) which is individually rational but not Pareto efficient since both would enjoy more benefit if they had played (cooperate, cooperate). Regarding the modelled types, altruists will always choose to cooperate and free-riders will always choose to defect. Hence, two altruists will cooperate, two free-riders will not cooperate, and a free-rider will exploit the resources of an altruist but neither will mind. On the contrary, collaborators seek other collaborators (or alternatively, altruists): they are willing to cooperate if the other player cooperates, too. In the original Prisoner's Dilemma or when the number of dilemma iterations is known beforehand, defection is the rational strategy to play. However, if the number of played rounds is unknown, cooperation becomes the rational strategy. If rational players cannot estimate the negative consequences of defection, they are better off by choosing cooperation [27].

In P2P networks, peers may obtain non-persistent identities at virtually no cost, leading to the whitewashing phenomenon in which a selfish peer is tempted to leave and immediately rejoin the network with a new identity after having performed actions that are detrimental to its reputation in the first place [19], [20]. With no persistent identities, there is no definitive means to distinguish a true newcomer from a whitewasher. Placing general mistrust in newcomers is one way to discourage whitewashing [5], [22]. The cost of this approach has been assessed in [22]. An alternative to general mistrust has been suggested in [19]. It consists of an adaptive stranger policy that judges new peers on the basis of new peers' behavior in the past by maintaining a *stranger account* that accumulates recent experience with new nodes. Plenty of free-riders joining the Desktop Grid will tend to decrease a stranger account's value, while a majority of collaborators will increase it. Hence, a stranger account may serve as a predictor for an unknown peer's reliability.

In this article, we propose to combine the adaptive stranger policy with a super-peer-based approach. Super-peer overlays consist of a comparatively small number of super-peers and a large number of edge peers. At join time, a peer has a neutral reputation with all other peers, influenced only through the stranger account. When super-peers regularly charge their edge peers reputation for the super-peer service and spread this reputation information to other peers, an edge peer's reputation

declines unless the edge peer performs work for other peers or becomes a super-peer itself. However, to become a super-peer, the SPSA protocol requires an edge peer to be appointed by another super-peer. This way, a super-peer is free to choose a trustworthy edge peer for promotion. We expect that with this scheme, a reputation system will effectively repulse free-riders in the long term.

IV. Concept

The absence of a central authority leads to the issue of effective resource discovery. Nodes will need to communicate with their fellow peers to keep track of available resources, handle job submissions and care for their completion. This joint effort may be directed by a Desktop Grid middleware. When run on every participating machine, it enables peers to coordinate their actions, providing a platform for applications to have arbitrary jobs processed by the Desktop Grid. In the concept presented in this article, every peer can submit a job which will be split into a number of tasks that may be processed by other peers. A peer that processes a task will be referred to as *worker*, whereas a peer that submits a job will be referred to as *initiator* of that job. Every peer is supposed to first compute a local power index based on a dummy job which may be distributed along with a Desktop Grid middleware. The computation of that job yields a benchmark number that eases the comparison of the peers' capabilities, similar to the *Horse Power Factor* in [28].

Given a super-peer network in which every edge peer is attached to exactly one super-peer each, our approach operates in a distributed way and completely bypasses the need for a virtual currency by relying on a reputation-based mechanism. There is no single point of failure and no requirement for pre-trusted nodes. Design goals have also included scalability and churn resistance to enable the concept to prevail in a dynamic environment. We ignore irrational malicious peers (i. e., those who cause harm to the system without benefitting from such action). To ensure fairness, cooperative peers grant reputation to other peers for successfully completing a task and for acting as a super-peer. The remainder of this section introduces all components of our concept including the workflow model, the proposed reputation system and incentives for peers to collaborate.

A. Workflow

The workflow model assumes that initiators want to lease computation time on idle workers. An initiator generally wishes to minimize its job's total time to completion (referred to as *makespan* [7]). Following this approach, the time the slowest worker takes will determine the makespan.

Two kinds of jobs are considered. The first kind is *independent-task applications* (ITA), also known as *bag-of-tasks* [5], where the workers process their tasks independently and isolated from each other. Prominent examples for ITA jobs include parameter-sweep applications [7]. Moreover, there are *connected problems* (CP) [28] which benefit from the interaction of workers, requiring the participating workers to perform their computation efforts concurrently. For example, connected problems include evolutionary algorithms that address combinatorial optimization problems. There, workers communicate with others to indicate that they have found a new best known solution. That way, the best known solution is replicated among the workers, removing the strict need to replace a failing worker. Since the global optimum is unknown, a criterion to terminate the computation is required, e. g., a deadline. In the ITA case, however, a job is split into a number of tasks that are distributed among the workers. It is unknown how long a task will take to be completed by a worker, and a worker's failure implies that its assigned task must be computed again by another worker.

A potential initiator first determines its job requirements which encompass the minimum and maximum numbers of desired worker peers, a worker's minimum acceptable local power index, the job type (CP or ITA) and a deadline until which all workers must have finished their computations. If the initiator is an edge peer, it passes these requirements on to its super-peer. The super-peer broadcasts a message containing a request for worker participation and the job requirements to all other super-peers, which, in turn, forward it to their attached edge peers. Conversely, the super-peers collect approvals from their edge peers and forward them to the initiator along with details on their respective power index and availability. This multiplexing reduces network load by preventing peers from individually replying to the initiator. Now, the initiator holds a list of worker candidates which it sorts according to power index, reputation and availability, and picks a sufficient number of workers which shall perform the distributed computation. The workers start computing upon reception of code and data transmitted to them by the initiator.

It is assumed that tasks running on worker peers can contact the initiator anytime. That way, a task may return its result after completion to the initiator, and may even submit intermediate results as proof-of-work to confirm that the worker has indeed provided the task with computational resources. The results are returned directly by the task running on the worker machine. A dishonest worker needs to know the result message's format to fake its contents, requiring it either to tap the line and intercept the message sent by the running task or to retrieve the message format from reengineering the task's code. Both ways entail efforts that exceed the effort of computing the actual task. Hence, a rational peer will choose to honestly compute the task. The workflow ends with the initiator validating the partial results and merging them into the complete outcome that would have resulted had the job been computed on a single machine in a non-distributed way.

B. Reputation system

A reputation system collects information on the past behavior of other peers [29], [30]. It enables cooperative peers to identify other cooperative peers, and it creates incentives for peers to act cooperatively by providing rewards for cooperative behavior [29], [31]. It enables casting the "shadow of the future" [27] which ensures that peers need to consider the

consequences of defection. Cooperative peers learn about the true willingness of others to reciprocate only when requesting remote resources on their own, but they can reduce the risk of erroneously donating resources to uncooperative peers by relying on a reputation system.

The model presented in this article has been built on the notion that the contribution of computational resources for use by others deserves rewards. These rewards are implemented by positive feedback. In a Desktop Grid setting, peers prefer to cooperate with other peers that have high reputation, and they wish to establish and maintain the potential to become successful initiators. This requires the ability to attract a sufficient number of workers to have the job computed in a reasonable period of time. Since the level of attraction is directly linked to their individual reputation, peers benefit from behaving in a way that earns them positive reputation. The aim of the proposed reputation system is to enable cooperative peers to detect free-riding initiators, shielding the detecting peer from being exploited. The system is fully distributed: every collaborating peer runs an instance of it. The proposed model assumes that free-riders will not compute any tasks for others; when collaborators refuse to compute tasks for free-riders, free-riders are left with the choice to switch to cooperative behavior or to leave the Desktop Grid.

Reputation systems differ in the way reputation is recorded. If a peer records reputation information only on those peers with which it has directly interacted in the past, the recorded information is unforgeable, but in large dynamic systems with considerable turnover, peers rarely encounter the same peer again for interaction. In contrast, shared-history reputation systems additionally incorporate experiences made by other peers [19].

Reputation may also serve as a currency replacement. Workers charge initiators a payment for the work they have performed on behalf of the respective initiator; the payment depends on the work's volume and is deducted from the initiator's local reputation account held by the worker. This way, reputation may be built by serving others and spent by straining others.

In the context of Desktop Grids, spoof feedback, the transmission of faulty or purposefully forged opinions on other peers, is the most prominent kind of attack on shared-history reputation systems [32]. In particular, a shared-history reputation system may face *collusion*, the phenomenon of interacting selfish peers (*colluders*) that mutually forge reputation ratings to benefit fellow peers in the colluder group and harm those outside it. In the context of this article, a colluder always attempts to free-ride.

Due to its purposeful, organized and durable nature, collusion is a severe form of spoof feedback. If peers act selfishly but autonomously, there is no incentive to spread counterfeit reputation information on other peers. However, if a group of colluding peers agrees to spread only positive information on the group's members to other peers, ordinary peers need to remain vigilant when receiving shared-history feedback from a remote peer. Systems in which all peers agree on a common (objective) reputation for every peer are especially affected by collusion [19].

In a super-peer scenario, colluders may become super-peers. In this state, they may exert malevolent influence on their edge peers. For instance, a super-peer which participates in a collusion could decide to forward only messages from fellow colluders and drop all others, or exchange forged reputation with its edge peers. To assess threats of this kind, one may consider the worst case that all free-riders know each other and form a single collusion group. Let n be the number of peers in the overlay out of which $\sqrt{n}$ are super-peers as selected by SPSA. Moreover, let p be the probability that an arbitrary peer is a colluding free-rider. The number of super-peers which belong to the colluder group follows the binomial distribution with parameters $\sqrt{n}$ and p. Let X be a random variable with this distribution. Its distribution function is [33]

$$Pr(X \leq x) = F(x) = \sum_{i=0}^{x} \binom{\lceil \sqrt{n} \rceil}{i} \cdot p^i \cdot (1-p)^{\lceil \sqrt{n} \rceil - i} \qquad (1)$$

Hence, the probability that at least one of the super-peers is a colluding free-rider amounts to

$$Pr(X \geq 1) = 1 - Pr(X \leq 0) = 1 - (1-p)^{\lceil \sqrt{n} \rceil} \qquad (2)$$

For a network of $n = 100$ peers and $p = 0.1$, this probability equals 65.13 %; at $p = 0.2$, it reaches 89.26 % and at $p = 0.3$, 97.18 %. To attain a probability of 50 % that a colluder is among the super-peers, the fraction of free-riders in a 100-peer system need only be $p = 0.067$, i. e., 7 colluding peers are already sufficient. In a 10 000-peer system, it is $p = 0.007$, requiring 70 colluders to achieve the same effect. These results show that it is feasible for a comparatively small number of colluding peers to get hold of a part of a super-peer overlay's infrastructure.

The impact of collusive behavior on P2P overlays becomes still more severe with the *Sybil attack* which constitutes a particularly intense special case of collusion [34]. It may occur in systems where identities cannot be verified due to the lack of a trustworthy authority. In the Sybil scenario, a group of colluding peers may be controlled by the same entity. Hence, the assumption of administrative autonomy and independence of peers does not hold for the Sybil attack. There is no general solution to this attack in fully distributed systems that satisfies the requirements of efficiency and scalability, but a number of approaches exist to limit the attack's effects in various application domains [35]. In Desktop Grids, besides weighting remote feedback with the submitting peer's credibility [19], a similarity measure can alleviate the effects of spoof feedback [31], [36]. Using a similarity measure, peers pay more attention to peers reporting similar opinions as one's own.

Peers spread feedback on the behavior of other peers they have interacted with. In particular, this concerns the initiator-worker relationship. For this purpose, every peer maintains a list of recent contacts, i. e., peers with which successful interaction has taken place in the past. Additionally, peers maintain reputation records on arbitrary other peers by keeping

a list of weighted positive (a) and weighted negative (b) experiences per remote peer. From these parameters, a peer may set up a β-distributed random variable B(a,b) for every remote peer and use its mean to reflect the respective remote peer's trustworthiness. While raw feedback is binary (0 equals poor, 1 equals good performance), reputation may assume any value $v \in [0,1]$ due to feedback weighting and aggregation. Both a and b will be initialized with 1 such that the distribution mean $\frac{a}{a+b}$ equals the neutral reputation value, 0.5, in the beginning [37]. This may be overridden by the stranger account.

To keep the reputation list size manageable, there shall be a time window that will remove outdated peers (i. e., those with no recent sign of activity). The list is split in two: one direct interaction history and one holding shared-history information received from other peers. Reputation is periodically spread to the last batch of peers with which a peer has interacted in the past (for every edge peer this includes its respective super-peer). Stored reputation is subject to a periodically applied exponential decay using a weighting factor $z \in (0,1)$ to focus on the rated peer's recent behavior, effectively leading to a short-term history [19], [37]. Essentially, for every remote peer on which reputation information is available, the decay process performs the following update:

$$a := 1 + (a-1) \cdot z \tag{3}$$

$$b := 1 + (b-1) \cdot z \tag{4}$$

That way, past reputation will fade out over time, taking the distribution's parameters gradually back to the initial setting unless new reputation information arrives.

When a request for participation as a worker from an initiator i with reputation r_{ji} arrives at a peer j, j will check with its reputation list if i is acceptable. Peer j instantly agrees to become a worker for i if j's directly observed reputation about i exceeds 0.5, or j's combined weighted local and remote observations about i reach or exceed the neutral value of 0.5. The neutral value's inclusion at this point supports bootstrapping the system in the beginning when no worker-initiator interaction has taken place yet.

If i is rejected according to the aforementioned rule, j switches to random-acceptance mode: i's reputation with j is compared to the moving average m of the last f seen initiators incorporating a tolerance factor $\beta \in [0,1]$. If $r_{ji} > (1+\beta) \cdot m$, i is accepted, while i is rejected if $r_{ji} \leq (1-\beta) \cdot m$. If $(1-\beta) \cdot m \leq r_{ji} < (1+\beta) \cdot m$, j makes its decision on a randomized basis by drawing a random number $d \in [0,1]$ from a uniform distribution. If $d < r_{ji}$, i is accepted, otherwise it is rejected. The random-acceptance mode supports the emergence of cooperation in environments with many free-riders. With a modest probability, it permits unknown collaborating peers to be accepted as initiators. However, the higher the tolerance level, the more free-riders are likely to pass through. It is for this reason that β needs to be chosen carefully to avoid an adverse impact on the reputation system's effectiveness. In all cases where no reputation information is available on an initiator, the worker resorts to the stranger account.

Every peer i maintains an interval t_i which defines the frequency of its reputation exchange. This interval need not equal the reputation decay interval. After a period of length t_i has elapsed, i transmits its stored reputation information (the weighted average of the distribution means for both the local and the remote observations) to the q members of its recent contacts list. Since reputation information basically only consists of one peer ID and one floating point number per rated peer, the reputation-related message volume remains reasonable. Peer i expects its contacted peers to reciprocate by replying with their respective reputation information. If a contacted remote peer fails to fulfill this expectation multiple times in a row, i removes it from its contact list. That way, the reputation exchange cycle encourages mutual updates.

When j receives a reputation list from i, j first checks i's credibility. Peer i's information will be incorporated into j's remote reputation list if i's reputation with j, r_{ji}, or the *personalized similarity measure* [36] between i and j, quantified as s_{ji}, exceeds the neutral value of 0.5. If i passes this test, j proceeds with normalizing the received reputation items (*votes*) to avoid subjective exaggerations, otherwise i's information is discarded. In the positive case, let v_{ik} be the vote cast by i about peer k. For every vote v_{ik} submitted by i, j now computes

$$v_{ik}^* = \frac{v_{ik} - 0.5}{\sum_l |v_{il} - 0.5|} \tag{5}$$

to normalize i's information, yielding $v_{ik}^* \in [-1,1]$. Let

$$r_{ji}^* = \alpha + \max\left\{0,\ 2 \cdot (1-\alpha) \cdot (r_{ji} - 0.5)\right\} \tag{6}$$

with $\alpha \in [0,1]$ being the minimum weight that j applies to remote opinions. $r_{ji}^* \in [\alpha,1]$ expresses the particular weight which j associates with i's opinion. Let g_{ji} be that weight weighted with the similarity between i and j, i. e., $g_{ji} := s_{ji} \cdot r_{ji}^*$, to yield the final weight j applies to each of i's votes. Ultimately, for every v_{ik}^*, j updates its remote observation list: if $v_{ik}^* > 0$, i's opinion on k is positive, hence j sets $a_{jk} := a_{jk} + g_{ji} \cdot v_{ik}^*$. If $v_{ik}^* < 0$, i's opinion on k is negative, so j sets $b_{jk} := b_{jk} + g_{ji} \cdot v_{ik}^*$.

Local reputation is modified as the outcome of a worker-initiator relationship. Workers receive rewards from initiators for completed tasks only. Hence, workers will want to finish an ongoing computation. An initiator i determines a worker's reputation gain primarily by the computation time the worker has expended. Concisely, let w be the worker in question, t the number of i's reputation periods – including fractions – of length t_i the task has occupied computational resources on w, and power_x the local power index of peer x. Now, i quantifies w's work effort h_{iw} as

$$h_{iw} = \frac{\text{power}_w}{\text{power}_i} \cdot t \tag{7}$$

Peer i recomputes the β-distribution parameters it keeps for w, a_{iw} and b_{iw}. If w has completed its task, i sets $a_{iw} := a_{iw} + h_{iw}$, otherwise $b_{iw} := b_{iw} + 2 \cdot h_{iw}$, as a direct observation. The factor of 2 serves as a penalty for the failed completion of a task. Note that neither i's choice of its reputation update

interval length t_i nor the worker-initiator power ratio biases the reputation propagation as reputation information transmitted to other peers is processed in an aggregated form only (by means of the distribution mean), independent of both the interval length and the power ratio.

Workers charge initiators reputation as the price of computation by adding negative feedback to the respective initiator's local reputation account. Since free-riders do not accrue positive reputation on their own, subsequent requests for workers by free-riders will be declined.

C. Incentives

We wish to obtain a self-sustaining system with a mutual reputation exchange. To this end, we propose a reputation-based incentive scheme that operates without the need to exchange virtual money. Rather, it fosters reciprocity in a dynamic environment. This scheme is built on the notion of positive reputation that may be obtained by adequate behavior. Positive reputation can only be earned by accomplishing work, boosting the willingness of peers to become workers. The following actions supplement the model's incentive scheme:

- Workers and initiators have established a trustful relationship after having successfully cooperated. Afterwards, they begin to exchange reputation information periodically. In our model, successfully completing a worker-initiator cycle or becoming part of a super-peer-edge-peer relationship are the only ways to exchange reputation information with another peer. If either peer cheats, the other peer will consider the relationship not to be a success. However, both peers are interested in reputation information about other peers, and successfully interacting over a longer period of time builds mutual trust. This hampers collusion as any peer wanting to spread false feedback will first need to complete a worker-initiator cycle or become a super-peer. Moreover, peers will stop transmitting reputation information to other peers if these remote peers do not provide reputation information themselves. This is similar to the Tit-for-Tat reputation exchange strategy used in [32]. We suggest the reputation exchange between two peers to be alternately initiated. This prevents peers from cheating by replying with the same reputation values it has just received; it also prevents the similarity measure from being tricked, because with equal reputation ratings, the similarity is maximum (1).
- Super-peer activity is considered a computation job with unknown duration. All edge peers are considered initiators and their respective super-peer is the single worker of their job. Consequently, super-peers charge their edge peers reputation while edge peers grant their super-peers reputation. Peers can receive an additional incentive to become super-peers by granting a super-peer the right to require its attached edge peers to compute a task for it. Every edge peer takes its turn in a round-robin or randomized fashion, and the turn length equals the length of the super-peer's reputation exchange interval. That way, every edge peer expends some computation power to the super-peer once in a while. If a super-peer attempts to cheat by requiring one edge peer too often to compute a task, that edge peer may disconnect at any time and seek another super-peer. Conversely, if an edge peer refuses to compute, the super-peer itself disconnects the link. The job may concern the computation of an arbitrary problem, or in the default case, be a dummy job which returns proof-of-work messages to the super-peer.

With these incentives, it becomes attractive for peers to become super-peers and to exchange reputation information with others.

V. Experiments

A reputation system faces the challenge of distinguishing cooperative from free-riding peers. In the beginning, all peers consider all other peers equal due to lack of experience. With time passing, collaborators interact with other collaborators. Hence, the proposed reputation system is expected to adapt to its environment over the course of time, accepting a limited number of free-riders in the beginning but repelling them after having swung in. To assess the performance of our approach, a number of experiments have been conducted in a custom discrete-event simulation environment using Java [12]. We were particularly interested in the speedup attained by our Desktop Grid concept compared to single-machine computation and in its resistance to free-riders. We also wanted to determine the impact of free-riding behavior on unprepared, unprotected Desktop Grids. We have expected the average makespan of jobs to increase with the number of free-riders. In our experiments, the makespan computation has exclusively concerned jobs initiated by cooperative peers. The makespan experienced by free-riders has not been taken into account as the utility of free-riders is considered irrelevant in this context.

A. Setup

Initiators are anticipated to submit computationally intensive jobs only since small jobs may be computed locally without needing to deal with the additional complexity of a Desktop Grid [18]. Hence, a Desktop Grid is likely to be exposed to jobs that would otherwise be processed by supercomputer-class systems. To model the volume of such jobs, we have resorted to real-world workload traces[1] of jobs processed by a Linux cluster at the Ohio Supercomputing Center. After data sanitization, the data of 30414 jobs were available. In a simulation experiment, jobs were either all CP or all ITA; in the ITA case, the initiator did not know about the length of the job, so every ITA job was actually distributed. Since ITA jobs tend to exert more negative influence on the average makespan than CP jobs due to the unknown runtime, ITA jobs have been chosen unless noted otherwise. Cooperative peers initiated jobs independently from their current local load. The interarrival time between two jobs submitted by the same cooperative

[1] from http://www.cs.huji.ac.il/labs/parallel/workload/l_osc/index.html

peer was modelled as a uniformly distributed random variable $U(0;5 \cdot \text{median single-machine CPU time per job})$.

The sample network consisted of 100 nodes with known RTT between all nodes as measured on PlanetLab [38], [39]. The maximum delay was estimated at 3 seconds. The simulation environment included a SPSA super-peer overlay optimization process running on every peer which reshaped the overlay for low end-to-end communication delay.

The reputation spreading scheme pushed reputation information to the elements of its recent contacts list (workers, initiators, and, in case of an edge peer, its super-peer) every 45 seconds of simulated time. The reputation aggregation mechanism placed a weight of 0.75 on local and a weight of 0.25 on remote observations. The history decay factor z was set to $z = 0.999$ to enable a graceful decay and long keeping of positive reputation accrued through previous work for other peers. To limit the load, the recent contacts list has been restricted to $q = 10$ elements and the unique initiator reputation history also to $f = 10$ elements. The minimum granted reputation weight α was set to 0.1, the random-acceptance tolerance level β to 0.02. After having completed a task, worker j modified an initiator i's local reputation by setting $a_{ji} := a_{ji} + \frac{1}{4} \cdot t_c$ and $b_{ji} := b_{ji} + \frac{3}{4} \cdot t_c$, where t_c is the number of reputation intervals – including fractions – that j has spent computing i's task.

Free-riding behavior has been modelled as the refusal to accept foreign tasks for local computation, the desire to submit a job for distributed computation upon entry, and the departure from the system when the job has been completed. Once a free-rider had left the system, a very brief period of simulated time (3 seconds) elapsed until it was replaced by another free-rider which immediately attempted to submit a new job. This aggressive behavior partially suppressed the establishment of reputation-backed cooperative relationships and spread mistrust in the system due to the large amount of jobs initiated by free-riders; it strained the reputation system, testing a case which is substantially more severe than is likely to be encountered in practice due to the exceedingly high volume of jobs submitted by free-riders. Free-riders attempted 10 times to distribute their job before giving up on that job. Additionally, in some experiments, free-riders could collude. Collusion has been modelled on the basis of the wish to spread reputation information which maximally benefits all members of the colluding peer group while at the same time, maximally debases all other peers to reduce their capability to initiate a job of their own. Hence, a colluder's feedback on fellow colluders was quantified as 1, while feedback on other peers was quantified as 0. We have followed a worst-case evaluation approach, assuming that all free-riders in the system are colluders, and there is only one colluder group in the system, i. e., every colluder knows all other colluders. In our system, the only way to transmit forged reputation information to a collaborating peer is through the super-peer-edge-peer relationship. Since super-peers and edge peers consider themselves workers and initiators, respectively, they exchange reputation information until the colluding peer has completed its job and leaves the overlay.

While our model supports heterogeneous worker capabilities, we have assumed identical peer equipment in these experiments to facilitate comparisons. The willingness of peers to process tasks has been limited to a maximum of 24 hours of computation time per task. The simulation lasted for 7 days of simulated time. The admission of jobs has been narrowed to include only those of 10 days or less of total computation time on a single CPU. Moreover, jobs with very short CPU time requirements (less than 5 minutes) have been removed because such jobs are unlikely to be distributed on a Desktop Grid. In total, 18 156 jobs have met the admission requirements and as such, have been made available to the simulation environment. The longest admitted job required 8 days and 5 hours on a single CPU, the shortest one 5 minutes. Median job CPU time consumption equalled 5 hours, 48 minutes. For every experimental scenario, 30 runs have been performed.

B. Results

We have first examined the impact of free-riding behavior on a Desktop Grid with cooperative workers not protected by a reputation scheme. To this end, we have exposed the system to free-rider fractions of various sizes and measured the average makespan experienced by cooperative peers. Figure 2 shows the outcome, confirming that the makespan grows with the fraction of free-riders. This reflects the negative impact of free-riders on system performance, highlighting the necessity to introduce countermeasures that inhibit free-riding.

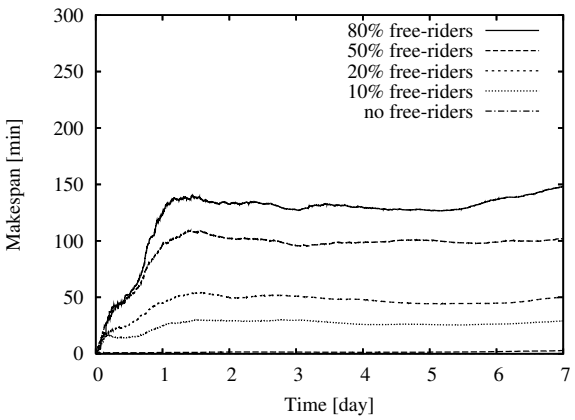


Fig. 2. Effects of free-riding on average makespan

In another experiment, the performance of a free-riding-resilient Desktop Grid which uses the proposed reputation system has been compared to an unprotected but otherwise identical Desktop Grid. Figures 3, 4, 5 and 6 show the outcome for free-rider levels of 10 %, 30 %, 50 % and 70 %, respectively. The makespan is considerably lower when a reputation scheme is available to the collaborating peers than in the original setting where unprotected collaborating peers cannot detect free-riders. With free-riders in the system, the reputation system takes time in the beginning to adapt to the environment but then swings into stabilizing the average

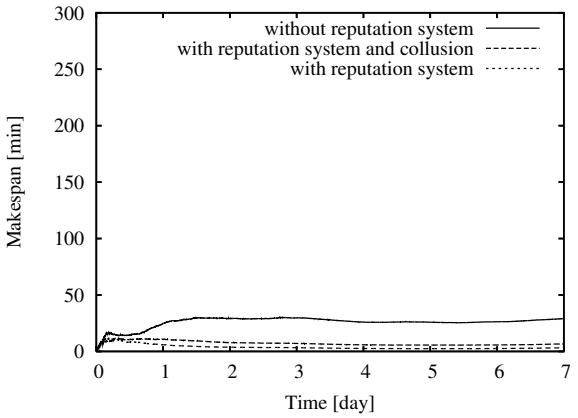


Fig. 3. Average makespan for jobs with 10 % free-riders in the Desktop Grid

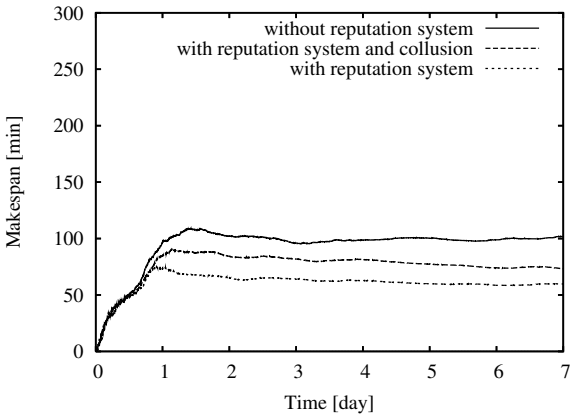


Fig. 5. Average makespan for jobs with 50 % free-riders in the Desktop Grid

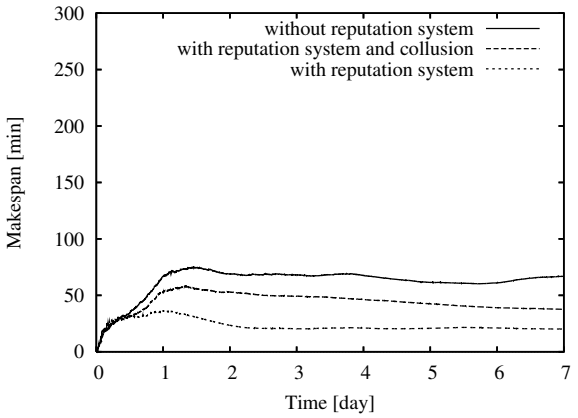


Fig. 4. Average makespan for jobs with 30 % free-riders in the Desktop Grid

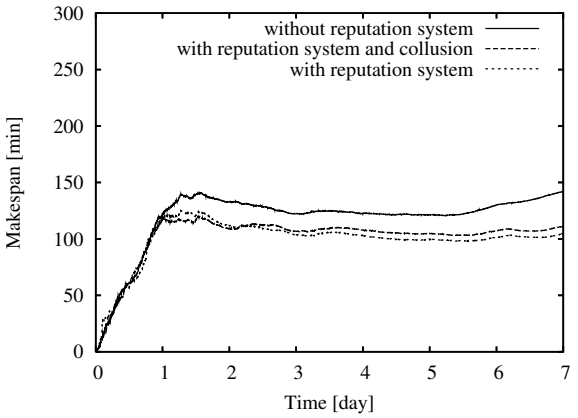


Fig. 6. Average makespan for jobs with 70 % free-riders in the Desktop Grid

makespan at a low level in the long term. This fact implies that our concept does actually exhibit the desired resilience towards free-riders.

From the figures, it can be seen that the reputation system resists a collusion but fares worse than with independent free-riders. Colluders, like regular free-riders, whitewash and return with a new identity 3 seconds after having left the Desktop Grid. If colluders take more time to return, the situation improves. Figure 7 depicts the average makespan for a free-rider fraction of 50 % when colluding peers take time to return to the overlay with a whitewashed identity after having left with their job completed. Their respective return time is randomly drawn from a uniformly distributed random variable bounded by the time limits given in the figure. While collusion still retains an impact on the average makespan, it is far less severe than in the aggressive case represented by a constant 3-second delay.

The handling of requests for participation in a scenario with 10 % free-riders is illustrated by Figure 8. It shows the cumulative numbers of tasks accepted and declined by worker peers, differentiated between free-riders and collaborators, the latter being either unprotected or protected by the proposed reputation system. The figure proves that in the unprotected case, collaborators' resources are quickly exhausted by the requests of free-riders, hence few resources remain for computing tasks initiated by other collaborators. In contrast, collaborators protected by the reputation system decline the vast majority of free-riders' requests. As desired, this behavior enables them to dedicate their resources to other collaborators.

The *speedup* of jobs has been depicted in Figure 9, with speedup defined as the ratio of the single-machine makespan to the makespan attained in the distributed computation case. This experiment has been conducted to quantify the general benefit of Desktop Grids which is to accelerate the computation of arbitrary jobs using otherwise idle resources. As also seen in Figure 3, the speedup of jobs drops to a low level already with as little as 10 % of free-riding peers if protection against free-riders is unavailable. The experimental results confirm that in the optimal case with all peers being altruists and zero overhead from a reputation system, a 100-peer Desktop Grid can generate a peak average speedup of approximately 60. The overhead generated by the reputation system exerts only a small influence on the speedup. This can also be seen in Figure 10 which relates to the average makespan: ensuring scalability, the proposed reputation system adds little overhead to the Desktop Grid's operations. Figure 10 depicts a comparison between two Desktop Grids

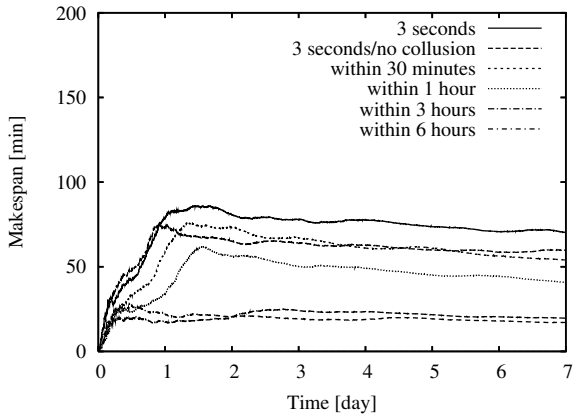


Fig. 7. Average makespan under collusion when colluders take time to return

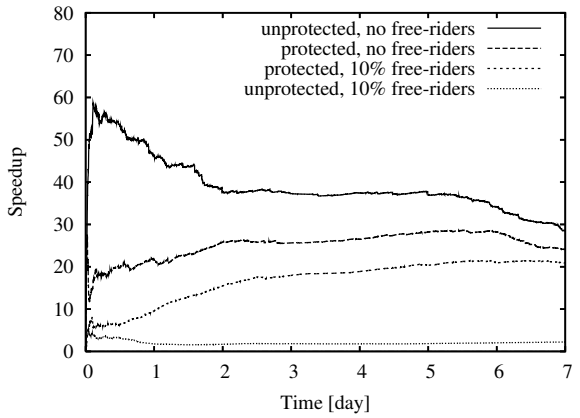


Fig. 9. Average speedup through distributed computation

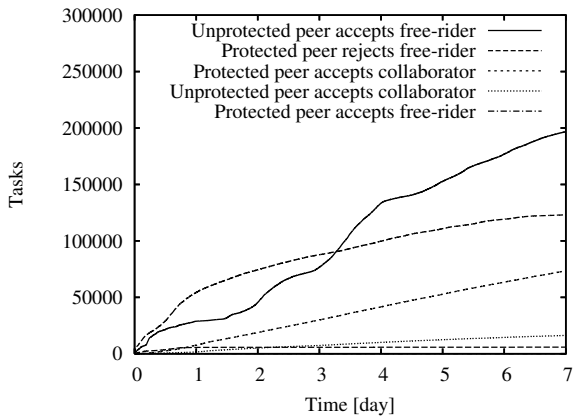


Fig. 8. Cumulative numbers of accepted and declined tasks

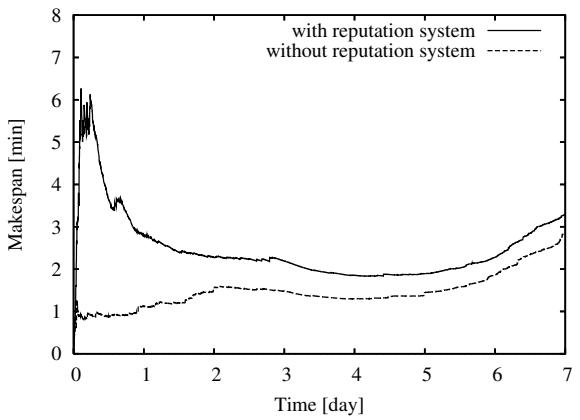


Fig. 10. Influence of reputation system on average makespan

populated with cooperative peers only: one completely unprotected system with all peers as altruists, and one system protected with our approach with all peers as collaborators. Since job volumes differ considerably, large-volume jobs can increase the makespan as seen in the figure. A volume-adjusted assessment of the overhead impact is given in Figure 11 which shows the average number of workers per job. Apart from the initial phase where collaborative peers first need to gain trust in each other under the control of the proposed reputation system, the difference in performance remains minor.

In summary, the simulation experiments confirm the effectiveness of the proposed reputation system. With only a minor effect on efficiency caused by the administrative overhead, the system enables cooperative peers to detect and avoid free-riders, fostering reciprocative behavior in the Desktop Grid.

VI. Related work

The benefits of self-organizing P2P structures to Desktop Grids have been pointed out in [6]. Decentralized control well suits Desktop Grids which can encompass millions of desktop computers. The approach in [6] is based on mobile agents forming a tree-like overlay. It considers altruistic behavior only, and its capability to deal with peer failures is limited.

A two-layered middleware for distributed computing in P2P overlays has been introduced in [28] by the name of *Vishwa*. Vishwa incorporates load migration, fault-tolerance, and a notion of proximity awareness as nearby peers form clusters, but it is still based on altruistic peer behavior and does not tap the benefits of super-peer overlays.

The free-riding phenomenon affects distributed systems whose benefit relies on voluntary contributions from its selfish participants. Free-riding has been observed in P2P file-sharing networks as a cause for performance degradation [16], [19], [21]. Due to a different nature of resources in Desktop Grids, solutions for free-riding prevention in file-sharing networks need to be modified for a Desktop Grid setting. While files may be distributed and stored, CPU cycles are volatile and can neither be stored nor replicated.

An application-neutral reputation scheme is introduced in [17]. It is based on solicited first-hand feedback acquired through time-to-live-bounded random walk sampling. Its utility for Desktop Grids is limited because of the expensive polling scheme and the inability to deal with considerable churn. As [19] confirms, a shared interaction history fits large-scale, churn-prone settings better than a witness-only approach.

The EigenTrust algorithm creates a global trust estimate in a distributed way, based on the concept of transitive trust [21], [40]. However, EigenTrust relies on a set of pre-trusted

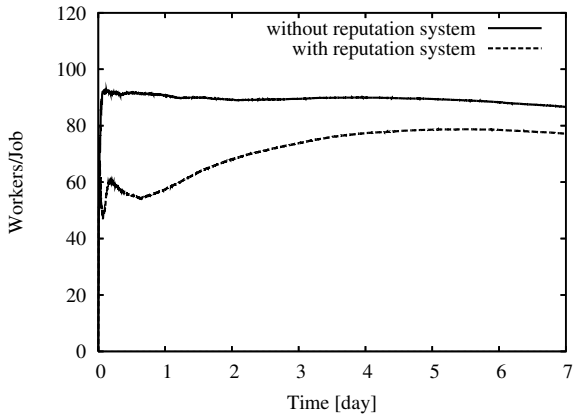


Fig. 11. Influence of reputation system on average number of workers per job

peers, does not incorporate the advantages of super-peers, and is susceptible to collusion.

OurGrid is a Desktop Grid which also incorporates a reputation mechanism, the *Network of Favors* [5], [18]. OurGrid is not super-peer-based, and the accumulated reputation does not decay over time. As with [17], reputation concerns direct witnesses of interactions in OurGrid only which is incompatible with substantial dynamics.

The framework introduced in [41] explicitly benefits from the presence of super-peers. Resorting to threshold cryptography, it focuses on the security aspects of feedback submission among peers. It does not intend to specify a reputation scheme itself, and due to its application-neutral scope, provides no insight into Desktop Grid workflows.

The reputation scheme proposed in [42] incorporates the notion of incentive-compatibility from the field of mechanism design. It solves the problem of forged feedback on other peers' past performances by seeing to it that telling the truth is in the best self interest of the reporting peers. The basic idea is to spread reputation information via specialized intermediary peers called R-agents. Instead of spreading own feedback directly to other peers, a peer buys reputation information from and sells its own experience with other peers to trustworthy R-agents. While it solves the problem of truthfulness, the scheme exhibits several drawbacks. It requires at least one R-agent to be present in the system at all times, it depends on all R-agents being trustworthy at all times, and it is susceptible to collusion. Also, the scheme suffers from the disadvantage of establishing truthfulness using payments according to the mechanism design approach: there needs to be a virtual currency which all peers in the system, including the R-agents, accept.

With respect to reputation systems, [29] distinguishes symmetric from asymmetric approaches. In symmetric systems, the computation of a remote peer's reputation happens under anonymity; peer identities can be exchanged as long as the trust graph's topology remains unmodified. Except for the trivial constant variant, symmetric reputation functions are vulnerable to collusion. In contrast, asymmetric reputation functions apply when every peer computes remote peer reputations by itself, and can be shown to withstand Sybil attacks under certain conditions. The reputation system proposed in this article uses an asymmetric approach.

A different form of free-riding in P2P networks concerns message routing in the overlay. If peers are unwilling to forward or answer queries, the overlay itself may not work as desired. Exploiting properties of the CAN overlay topology, [32] introduces a distributed reputation system designed to thwart uncooperative behavior in the context of message propagation.

In Desktop Grids, workers may cheat by choosing not to compute their assigned task but to return arbitrary data. To counter this, a replication approach might assign the same task to multiple workers and pick the correct result through majority voting. An alternative to this is Quiz [43] which assigns a number of tasks to one worker, including a Quiz task of which the submitting peer already knows the result. If the Quiz task is properly processed, the worker is assumed to have processed the other tasks properly, too. This sampling method is found to be superior to task replication. It is used in a Desktop Grid scheme which also includes reputation, *Cluster Computing on the Fly* [43].

VII. Conclusion and Future Work

Desktop Grids offer attractive opportunities for large-scale distributed computation. However, free-riding behavior may severely degrade a Desktop Grid's performance. In this article, we have presented a distributed reputation system for dynamic super-peer-based Desktop Grids with selfish peers. All components of our contribution – the reputation system, the underlying super-peer topology and the accompanying workflow model – are designed for scalability, and there is neither a single point of failure nor a need for pre-trusted nodes. Moreover, our concept effectively detects free-riding peers, encouraging cooperative peers to contribute resources. In simulations, we have empirically verified our proposed scheme's effectiveness. Future work includes the integration of a super-peer structure which interconnects the super-peers with a Chord ring [44], [45]. Moreover, to tackle the effects of collusion in scenarios with large fractions of colluding peers, we consider a multi-level similarity measure that incorporates the confidence into one's own ratings to better deal with collusion and less severe forms of spoof feedback. We also plan to use network coordinates for delay-optimized worker selection in CP scenarios where jobs benefit from inter-worker communication. Ultimately, we intend to deploy our proposed scheme on PlanetLab.

References

[1] Peter Merz, Florian Kolter, and Matthias Priebe. Free-Riding Prevention in Super-Peer Desktop Grids. In *Proceedings of the 3rd International Multi-Conference on Computing in the Global Information Technology, ICCGI 2008, IARIA*, pages 297–302, 2008.

[2] Ian T. Foster and Adriana Iamnitchi. On Death, Taxes, and the Convergence of Peer-to-Peer and Grid Computing. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems*, pages 118–128, 2003.

[3] David P. Anderson, Jeff Cobb, Eric Korpela, Matt Lebofsky, and Dan Werthimer. SETI@home: an experiment in public-resource computing. *Communications of the ACM*, 45(11):56–61, 2002.

[4] David P. Anderson and Gilles Fedak. The Computational and Storage Potential of Volunteer Computing. In *Proceedings of the 6th IEEE International Symposium on Cluster Computing and the Grid*, pages 73–80, 2006.

[5] Nazareno Andrade, Francisco Vilar Brasileiro, Walfredo Cirne, and Miranda Mowbray. Automatic grid assembly by promoting collaboration in Peer-to-Peer grids. *Journal of Parallel and Distributed Computing*, 67(8):957–966, 2007.

[6] Arjav J. Chakravarti, Gerald Baumgartner, and Mario Lauria. The organic grid: self-organizing computation on a Peer-to-Peer network. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 35(3):373–384, 2005.

[7] Noriyuki Fujimoto and Kenichi Hagihara. A Comparison among Grid Scheduling Algorithms for Independent Coarse-Grained Tasks. In *Proceedings of the 2004 Symposium on Applications and the Internet Workshops*, pages 674–680, 2004.

[8] Derrick Kondo, Michela Taufer, Charles L. Brooks, Henri Casanova, and Andrew A. Chien. Characterizing and Evaluating Desktop Grids: An Empirical Study. In *Proceedings of the 18th International Parallel and Distributed Processing Symposium*, 2004.

[9] Adriana Iamnitchi and Ian T. Foster. A Peer-to-Peer Approach to Resource Location in Grid Environments. In Jarek Nabrzynski, Jennifer M. Schopf, and Jan Weglarz, editors, *Grid resource management: state of the art and future trends*, pages 413–429. Kluwer, 2004.

[10] Beverly Yang and Hector Garcia-Molina. Designing a Super-Peer Network. In *Proceedings of the 19th International Conference on Data Engineering*, pages 49–62, 2003.

[11] Gian Paolo Jesi, Alberto Montresor, and Özalp Babaoglu. Proximity-Aware Superpeer Overlay Topologies. In Alexander Keller and Jean-Philippe Martin-Flatin, editors, *Proceedings of SelfMan'06*, volume 3996 of *Lecture Notes in Computer Science*, pages 43–57. Springer, 2006.

[12] Peter Merz, Matthias Priebe, and Steffen Wolf. A Simulation Framework for Distributed Super-Peer Topology Construction Using Network Coordinates. In *Proceedings of the 16th Euromicro Conference on Parallel, Distributed and Network-based Processing*, pages 491–498, 2008.

[13] Peter Merz, Matthias Priebe, and Steffen Wolf. Super-Peer Selection in Peer-to-Peer Networks using Network Coordinates. In *Proceedings of the 3rd International Conference on Internet and Web Applications and Services*, pages 385–390, 2008.

[14] Russ Cox, Frank Dabek, M. Frans Kaashoek, Jinyang Li, and Robert Morris. Practical, distributed network coordinates. *Computer Communication Review*, 34(1):113–118, 2004.

[15] Eugene Ng and Hui Zhang. Predicting Internet Network Distance with Coordinates-Based Approaches. In *Proceedings of IEEE INFOCOM*, 2002.

[16] Eytan Adar and Bernardo A. Huberman. Free Riding on Gnutella. *First Monday*, 5(10), 2000.

[17] Emmanuelle Anceaume and Aina Ravoaja. Incentive-Based Robust Reputation Mechanism for P2P Services. In *Proceedings of the 10th International Conference on Principles of Distributed Systems*, pages 305–319, 2006.

[18] Nazareno Andrade, Francisco Vilar Brasileiro, Walfredo Cirne, and Miranda Mowbray. Discouraging Free Riding in a Peer-to-Peer CPU-Sharing Grid. In *Proceedings of the 13th International Symposium on High-Performance Distributed Computing*, pages 129–137, 2004.

[19] Michal Feldman, Kevin Lai, Ion Stoica, and John Chuang. Robust incentive techniques for Peer-to-Peer networks. In *Proceedings of the 5th ACM Conference on Electronic Commerce*, pages 102–111, 2004.

[20] Michal Feldman, Christos H. Papadimitriou, John Chuang, and Ion Stoica. Free-riding and whitewashing in Peer-to-Peer systems. *IEEE Journal on Selected Areas in Communications*, 24(5):1010–1019, 2006.

[21] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. Incentives for Combatting Freeriding on P2P Networks. In *Proceedings of the 9th International Euro-Par Conference*, pages 1273–1279, 2003.

[22] Eric J. Friedman and Paul Resnick. The Social Cost of Cheap Pseudonyms. *Journal of Economics & Management Strategy*, 10(2):173–199, 06 2001.

[23] John Chuang. Designing incentive mechanisms for peer-to-peer systems. In *Proceedings of the 1st IEEE International Workshop on Grid Economics and Business Models*, pages 67–81, 2004.

[24] Michal Feldman and John Chuang. Overcoming free-riding behavior in Peer-to-Peer systems. *ACM SIGecom Exchanges*, 5(4):41–50, 2005.

[25] Philippe Golle, Kevin Leyton-Brown, and Ilya Mironov. Incentives for sharing in peer-to-peer networks. In *Proceedings of the 3rd ACM Conference on Electronic Commerce*, pages 264–267, 2001.

[26] Seth James Nielson, Scott Crosby, and Dan S. Wallach. A Taxonomy of Rational Attacks. In *Proceedings of the 4th International Workshop on Peer-to-Peer Systems*, pages 36–46, 2005.

[27] Robert Axelrod. *The evolution of cooperation*. Basic Books, New York, 1984.

[28] M. Venkateswara Reddy, A. Vijay Srinivas, Tarun Gopinath, and D. Janakiram. Vishwa: A reconfigurable P2P middleware for Grid Computations. In *Proceedings of the International Conference on Parallel Processing*, pages 381–390, 2006.

[29] Alice Cheng and Eric Friedman. Sybilproof reputation mechanisms. In *Proceedings of the 2005 ACM SIGCOMM workshop on Economics of Peer-to-Peer systems*, pages 128–132, 2005.

[30] Paul Resnick, Ko Kuwabara, Richard Zeckhauser, and Eric Friedman. Reputation systems. *Communications of the ACM*, 43(12):45–48, 2000.

[31] Sebastian Kaune, Konstantin Pussep, Gareth Tyson, Andreas Mauthe, and Ralf Steinmetz. Cooperation in P2P Systems through Sociological Incentive Patterns. In *Proceedings of the 3rd International Workshop on Self-Organizing Systems*, pages 10–22, 2008.

[32] Klemens Böhm and Erik Buchmann. Free riding-aware forwarding in Content-Addressable Networks. *The VLDB Journal*, 16(4):463–482, 2007.

[33] Jerry Banks, John S. Carson, and Barry L. Nelson. *Discrete-event system simulation*. Prentice Hall, Upper Saddle River, NJ, 1996.

[34] John R. Douceur. The Sybil Attack. In *Proceedings of the 1st International Workshop on Peer-to-Peer systems*, pages 251–260, 2002.

[35] Brian Neil Levine, Clay Shields, and N. Boris Margolin. A Survey of Solutions to the Sybil Attack. Technical report 2006-052, University of Massachusetts Amherst, Amherst, MA, 2006.

[36] Mudhakar Srivatsa, Li Xiong, and Ling Liu. TrustGuard: countering vulnerabilities in reputation management for decentralized overlay networks. In *Proceedings of the 14th international conference on the World Wide Web*, pages 422–431, 2005.

[37] Audun Josang and Roslan Ismail. The Beta Reputation System. In *Proceedings of the 15th Bled Electronic Commerce Conference*, 2002.

[38] Suman Banerjee, Timothy G. Griffin, and Marcelo Pias. The Interdomain Connectivity of PlanetLab Nodes. In *Proceedings of the 5th International Workshop on Passive and Active Network Measurement*, pages 73–82, 2004.

[39] Brent Chun, David Culler, Timothy Roscoe, Andy Bavier, Larry Peterson, Mike Wawrzoniak, and Mic Bowman. PlanetLab: an overlay testbed for broad-coverage services. *Computer Communication Review*, 33(3):3–12, 2003.

[40] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The EigenTrust algorithm for reputation management in P2P networks. In *Proceedings of the 12th International World Wide Web Conference*, pages 640–651, 2003.

[41] Tassos Dimitriou, Ghassan Karame, and Ioannis T. Christou. SuperTrust - A Secure and Efficient Framework for Handling Trust in Super Peer Networks. In *Proceedings of the 9th International Conference on Distributed Computing and Networking*, pages 350–362, 2008.

[42] Radu Jurca and Boi Faltings. An Incentive Compatible Reputation Mechanism. In *Proceedings of the IEEE Conference on E-Commerce*, pages 285–292, 2003.

[43] Shanyu Zhao, Virginia Lo, and Chris GauthierDickey. Result Verification and Trust-Based Scheduling in Peer-to-Peer Grids. In *Proceedings of the 5th IEEE International Conference on Peer-to-Peer Computing*, pages 31–38, 2005.

[44] Ion Stoica, Robert Morris, David Karger, Frans M. Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 SIGCOMM conference on Applications, technologies, architectures, and protocols for computer communications*, pages 149–160, 2001.

[45] Peter Merz, Steffen Wolf, Dennis Schwerdel, and Matthias Priebe. A Self-Organizing Super-Peer Overlay with a Chord Core for Desktop Grids. In K.A. Hummel and J.P.G. Sterbenz, editors, *Proceedings of the 3rd International Workshop on Self-Organizing Systems*, volume 5343 of *Lecture Notes in Computer Science*, pages 23–34. Springer, 2008.

A Secure P2P Incentive Mechanism for Mobile Devices

Jani Suomalainen[1], Anssi Pehrsson[2] and Jukka K. Nurminen[3,4]
[1]*VTT Technical Research Centre of Finland*
P.O. Box 1000, FI-02044 VTT, Espoo, Finland
Jani.Suomalainen@vtt.fi
[2]*Small Planet Ltd*
Ruoholahdenkatu 8, FIN-00180 Helsinki, Finland
Anssi.Pehrsson@smallplanet.fi
[3]*Nokia Research Center*
P.O. Box 407, FI-00045 Nokia Group, Helsinki, Finland
Jukka.K.Nurminen@nokia.com
[4]*Helsinki University of Technology*
P.O. Box 5400, FI-02015 TKK, Espoo, Finland

Abstract

Peer-to-peer applications are emerging into mobile devices. However, resource limitations of these devices introduce new challenges for P2P technologies. For instance, there is a need for incentive mechanisms, which address the free riding problem but do not waste devices' battery or communication resources. A centralized and user-identity based incentive mechanism enables mobile users to contribute with any device and receive P2P services with mobile devices. We explore security issues related to a centralized incentive mechanism by analyzing and classifying threats and potential security mechanisms. We propose a privacy preserving security architecture. The architecture is based on authentication, software tamper protection, and misbehavior detection mechanisms. Further, we describe a prototype implementation for mobile BitTorrent file sharing peers. We provide a discussion on potential security compromises, not jeopardizing sufficient security level, and compare our work to related research.

Keywords: *P2P; security analysis; incentive; mobile devices; BitTorrent*

1. Introduction

Peer-to-peer (P2P) based applications, particularly content sharing, are currently popular in personal computers and are expected to gain popularity also in mobile devices. Mobile devices have already enough computing and communication capabilities enabling them to participate to P2P networks. However, there are still challenges, which hinder mobile phones participation and contribution. For instance, incentives for users with mobile devices to contribute are not clear. Energy efficiency and communication costs are critical issues, which differentiate mobile devices from personal computers and discourage mobile users' contribution. Consequently, there is a need for incentive mechanisms, which motivate mobile users to contribute but also save mobile device's limited resources.

To address the problem of free riding, users can be rewarded for their contribution to the network. Different incentive mechanisms have been proposed and adopted to P2P networks. Approaches include distributed schemes, where either the contributor itself (P2P client software) or peers monitor contribution, and centralized schemes, where servers maintain records on clients' contribution. In Section 6, we present a survey on existing work related to P2P incentive mechanisms. However, existing schemes have not been designed from the point of view of mobile devices. Hence, we have made an own proposal for an incentive mechanism: the credit system. This mechanism ties rewards to user-identities instead of device-identities and is thus more suitable for users with different kinds of terminals including mobile devices.

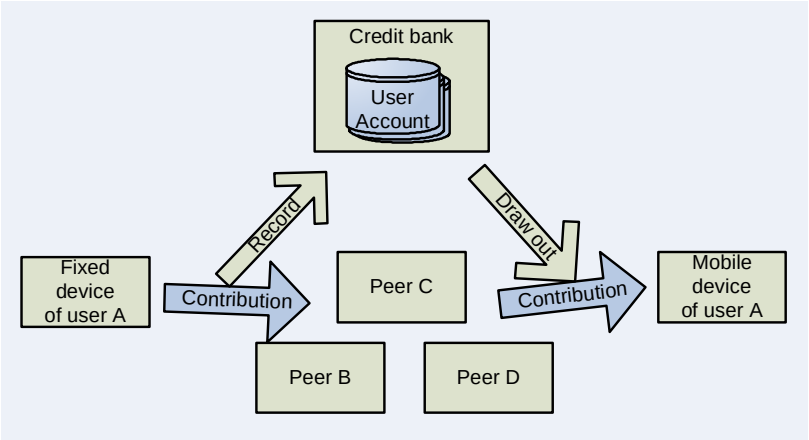


Figure 1. An overview of the P2P credit system. The user A contributes to P2P network with a fixed device, without resource limitations and consumes credits using mobile device, with limited communication and energy resources. The credit bank keeps track of users' contributions and controls how much contribution the user may receive from other peers.

Unfortunately, as the credits used in the incentive mechanism can be monetarily valuable, this system will face different threats from attackers trying to misuse the system. Therefore, security mechanisms, which are feasible also for mobile devices, are needed.

In this paper, we study the credit system from the point of view of security. The previous version of the paper [1] was presented in the ICIW 2008 conference. This version has been extended with more extensive security analysis and literature survey. First, in Section 2, we review our proposal, the credit system. The system was initially proposed and its feasibility evaluated through a mathematical analysis in [2]. In Section 3, we describe threats and potential attacks against incentive mechanisms and the system. In Section 4, we describe security architecture for the credit system. We contribute by surveying and analyzing which security mechanisms are available and how they could be applied for the credit system. In Section 5, we describe a prototype implementation of the credit system for BitTorrent clients, which are running on mobile devices. In Section 6, we compare of existing incentive mechanisms against the proposed solution.

2. The credit system

The proposed P2P credit system is a centralized incentive mechanism, which enables mobile devices to participate P2P network without requiring mobile terminal itself to contribute. Architecture of the credit system is illustrated in Figure 1.

The central entity of the system is the credit bank, which rewards P2P nodes for their contributions with credits and controls that only those nodes with credits can receive services from the network.

Credits are user-specific as the credit bank maintains accounts for each user. This enables the same user to collect credits with different devices. Also, credits can be used with any device belonging to the user or given for other users. This enables mobile users to receive services from P2P networks even if they do not want to contribute with their mobile terminals. For instance, a user can contribute with a PC at home and then use credits from this contribution with a mobile device.

Contribution, providing credits, may mean e.g. sharing content, supplying information on content location, or performing computations. Different contributions may be valuated differently. For instance, sharing of DRM protected content may provide more credits than sharing of unidentified data. Credits can be utilized to get content, services or high quality of service (QoS) level from other peers or, alternatively, from external service providers.

Credits, which can be utilized in other devices, provide incentive for high-capacity servers i.e. 'super-nodes' to contribute. This would motivate commercial service providers to participate P2P network. However, commercial parties require strong security measurements.

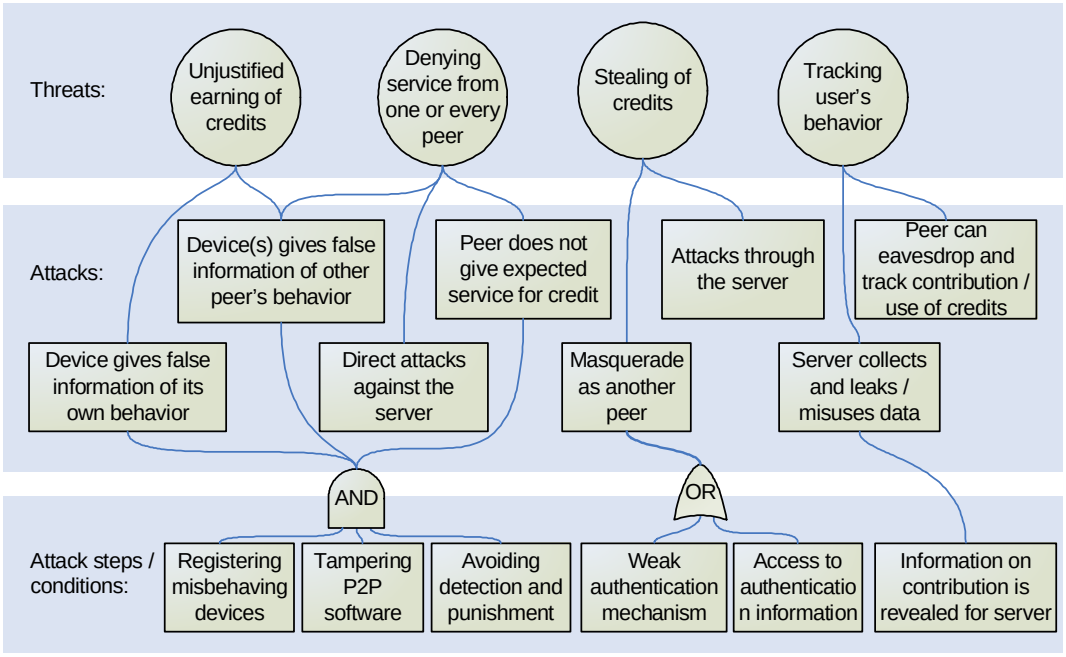


Figure 2. Threats against P2P credit systems. Threats have been classified to four main categories. The image also illustrates some attacks and related attack phases, which may make these threats true.

3. Security threats

Participants of the credit system, the credit bank and peers, may be attacked in different times. Attacks may occur during the contribution phase, after contribution, or in consumption phase. Some examples of security threats are listed in Figure 2. The figure provides also an attack tree illustrating some attacks, which might realize these threats.

The contribution phase is vulnerable for various attacks, where an attacker tries to earn unjustified credits:

1. A device may claim that it contributed, even if it did not, in order to receive credits
2. Nodes may give false (positive) information about their peers. For instance, a user may have two devices giving false information of each other. Also, in the 'Sybil attack' [3], an attacker may have a large amount of virtual peers providing false information. Further, different users may also collaborate and e.g. exaggerate each others' contribution.
3. A device might contribute but the contribution may be bogus. For instance, a device may claim that it made particular analysis of given data without doing it or uploaded content files may be corrupted.

In order to execute attacks, which require peer to give false (positive or negative) information, an attacker must have suitable attack software. This can be achieved by tampering authentic software. Tampering attacks require some skill but after tampering the attacker may distribute attack software to other users through Internet.

The credit system may face different availability related threats:

1. As the credit system is dependent on a centralized credit bank server, it is vulnerable for denial-of-service attacks. These attacks may utilize protocol vulnerabilities in the credit bank server or be brute-force attacks.
2. Devices may give false information about their peers and claim e.g. that peer's contribution was not acceptable. As a consequence, the credit bank may limit victim peer's access to its credits.
3. A peer may claim that a user received contribution in order to decrease amount of user's credentials. This attack may occur when user is expecting service or, potentially, at any time when there are credits in users account.

A credit bank or communication between peers may be attacked in order to steal credits or to get services with credits belonging to others.

1. An attacker may tamper identity information of contribution made by others. This may be possible if peers are not authenticated or if authentication mechanisms have security vulnerabilities.

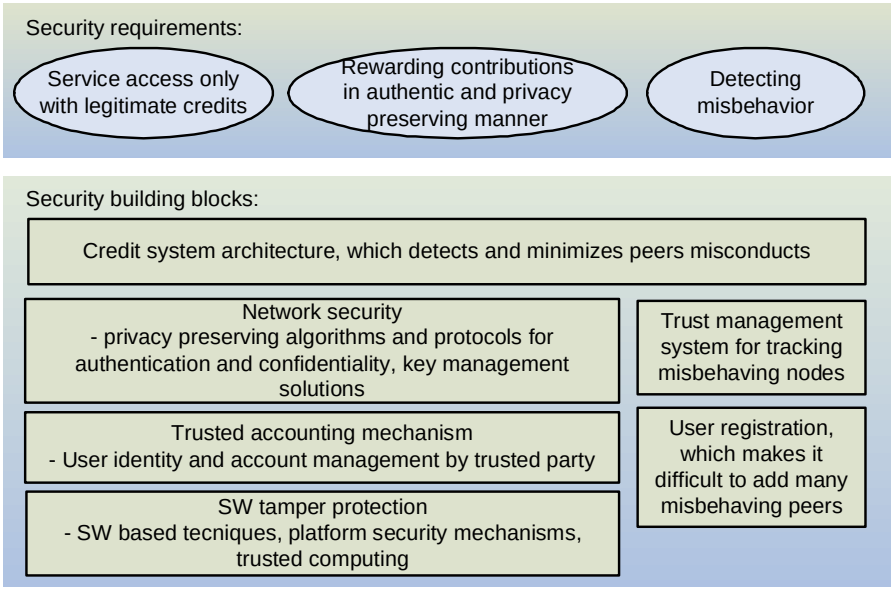


Figure 3. Security requirements and building blocks. The credit system's security architecture is designed to fulfill the three main requirements: Services can be accessed only with legitimate credits. All contributions are rewarded in authentic and privacy preserving manner. Misbehaviors may be detected even if attacks may occur. These requirements are addressed with the underlying security building blocks.

Alternatively, attacker may gain access to authentication information e.g. through malicious software, which is installed to user's device.

2. An attacker may utilize some vulnerability in the credit bank server and gain an access to the server. Then attacker may e.g. modify credit databases.

Peers' activities in P2P system may be tracked and using this information the user may be profiled. For instance, a particular fear is that users may be punished due to their contribution. A centralized credit system provides some new security worries, which should be considered. Firstly, the credit bank provides a single point, which must be trusted and which can be monitored or attacked to resolve information. Secondly, participation with different devices can be mapped to a single user.

4. Security building blocks

A design of a secure P2P credit system must consider the threats identified in the previous section. Essentially, the system must control that service are available only if credits are used in legitimate manner and that rewards for contributions are given for authentic contributors. Further, it is preferable that the system does not cause any new privacy problems. This control can be proactive. However, in practice, proactive solutions cannot provide full protection, and hence there must be a way to detect misbehaving peers. Secondarily, as scalability is a potential bottleneck of the server based credit system, one goal for security architecture should be minimization of required communication and computing resources. Figure 3 illustrates the main requirements for the system and discussed security building blocks. In the following subsection, the secure credit system architecture is first described. Then, design decisions and available building blocks are further discussed.

4.1. Architecture

In the P2P credit system architecture, both contributor and consumer inform the credit bank when a contribution is made. Two sources are required, as individual nodes cannot be trusted to deliver correct information. A message diagram in Figure 4 illustrates communication in the secure P2P credit system. The figure illustrates a basic case where User B contributes with one device and User A consumes credits with another device.

The consumer initiates scenario by requesting contribution from a contributor and by authenticating itself. The contributor will check from the credit bank if the consumer has enough credits for the requested amount of contribution. If there are enough credits, the contribution begins. The credit bank increments User B's account after the credit query has been made.

When the consumer has received the contribution, it will inform the credit bank, which will remove credits from User A's account. Alternatively, the credit bank could remove credits already, when the contributor makes the query. The latter approach would save some signaling costs but is infeasible since it might cause users to loose credits when a contributor goes offline or crashes due to technical failure. The consumer might try to get contribution for free by not sending a verification message. However, the credit bank is able to detect peers who make large amount of content queries but do not make any contribution verifications.

To make the system more scalable a few mechanisms can be applied to minimize amount of communication.

1. A buffering mechanism can be utilized to avoid messaging between peers and the server during every transaction. Contributors and consumers can buffer information of transactions and send larger reports only occasionally e.g. once per a day. After noticing that account balance has gone to negative, the credit bank will block user's participation by not renewing user's authentication information (e.g. time limited certificates).
2. A contributor does not need to inform the server on every transaction. For instance, to save battery resources, a mobile node may choose to not to make confirmations. The consumer should not be able to determine whether a confirmation is made or not and, hence, should not be able to send verification messages at the same time.
3. Some resource optimization can be achieved by selecting which contributions are rewarded and which require credits. For example, credits can be demanded only from information of locations content files instead of demanding them for every small part of content file.

4.2. Authentication mechanisms

Network security mechanisms – security algorithms and protocols – are needed to authenticate communication. The strength of an authentication mechanism should be selected so that efforts of attacking are larger than efforts of contributing. If contribution means uploading of files, cryptographic authentication of contributor may not be needed. This is because capturing, tampering and then uploading a tampered file of may be more difficult than uploading own files. However, if contribution means running some program for some period of time before transmitting, stronger authentication is required.

When there is a large amount of messages between peers and the credit bank related to small contributions, it may not be justifiable to make too heavy and resource consuming authentication. Authentication protocols based on shared secrets may be more feasible, instead of protocols utilizing asymmetric cryptography or heavy handshakes such as TLS.

Single sing-on architectures, for instance solutions from Liberty alliance and Microsoft passport, provide potential authentication infrastructures, which could be adopted also for the credit system. These systems phase similar challenges i.e. enable nodes to authenticate themselves to different servers (in our case other peers and the credit bank).

4.3. Software tamper protection

Peers and P2P software in them cannot be assumed to be trustworthy. A single attacker may modify one copy of the client software and then distribute this tampered version to other users. However, with software and device security mechanisms some additional trustworthiness may be gained.

Some security level can be achieved with obfuscation techniques, which make changing program code more laborious and time consuming. However, determined attackers can circumvent obfuscation based security.

Another approach is to use trusted hardware modules. For instance, trusted computing technologies enable small trusted hardware components to verify identity and integrity of software running in a device. Consequently, the credit bank or contributors could remotely attest and verify that a client device is running authentic software. These remote attestation mechanisms have been proposed also for P2P environments [4]. However, efficiency and scalability issues may limit the usability of remote attestation. Also, current platform security mechanisms in mainstream mobile devices do not support these mechanisms.

4.4. Detecting misbehavior

When every device cannot be assumed to be trustworthy, mechanisms for detecting misbehaving peers are needed. Particularly, there must be a way to monitor and analyze suspicious actions and there must be a way to punish misbehaving clients.

Clearly, suspicious activities for the credit system include cases where a peer makes credit query but a consumer does not confirm to receive content. In individual cases, one suspicious activity is not an evidence of misbehavior or does not indicate who the faulty counterpart is. However, a large amount of suspicious activities might indicate illegitimate behavior.

Detecting misbehavior becomes more challenging if attackers are able to easily introduce large amount of (virtual) misbehaving nodes or to change identities when the credit system tries to punish the user. In order to be able to defend against these attacks, the registration process should not be too easy or cheap. At least, an attacker should not be able to automatically add new virtual nodes.

One characteristic of attacks trough virtual nodes is that these nodes will get most of their credits from the same peers. Therefore, these attacks might be detectable by looking for isolated groups where some peers get exceptionally many contribution verifications. Unfortunately, this kind of mechanisms would detect also users whose contribution is interesting only for some very specialized users. Hence, this kind of mechanism would be an incentive for users to contribute content that is popular for masses. Also, analyzing this kind of behavior would probably be unfeasible for large amount of peers.

Attacks where registered peers collaborate are difficult to prevent. Active manual work may be used against some attacks. For instance, tampered software, which multiplies the amount of notified contribution, may be detectable when it communicates with other peers (these peers must agree on the amount of informed contribution). If these clients emerge, detectors must implement new mechanisms for tracking misbehaving clients.

Punishment mechanisms depend on the nature of P2P network and value of content. In minor cases, available service level could be cut down for potentially suspicious devices. For instance, an account can be decremented or frozen for some period of time. When the monetary value of credits is significant, judicial actions could be possible.

4.5. Privacy enablers

The P2P credit system should not introduce any new unnecessary mechanisms, which would further compromise privacy.

Peers, receiving and verifying contribution, do not themselves need to identify peers who are contributing. However, the credit bank needs to map verifications into contributors. To enable peers to verify contributions without revealing identity to peers, temporary random identifiers can be utilized. Consequently, an attacker cannot utilize these identifiers to determine if different contributions are made by one user and not by several users. It is enough that the credit bank is able to map users' accounts into random identifier. This mapping can be enabled with a message exchange where peers request temporary identifiers from the credit bank. However, message exchange for every contribution means additional communication. A better solution might be that contributors and the credit bank agree shared secrets, which they use to generate identifiers. For example, pseudo random sequences [5] could potentially be applied in a P2P credit system.

As a consequence, use of random identifiers enables a credit system to work with anonymous P2P networks, such as Tarzan [6] and Freenet [7]. Use of temporary identifiers prevents also attackers, who are eavesdropping communication between peers and the credit bank, from resolving peers' communication parties. Alternatively, cryptographic solutions could be utilized to achieve the same effect. To prevent eavesdropper from resolving how much peer is contributing, encrypted bogus traffic could be introduced. However, for mobile devices use of cryptographic techniques and bogus traffic is expensive.

The credit bank needs information on contributors' identity as well as the amount of contribution. Also, the credit bank may require information of real identities in order to implement strong misbehavior detection system. However, information on exact contribution does not have to be revealed.

5. An implementation of the credit system for mobile BitTorrent clients

To evaluate the feasibility of the credit system idea, a prototype was implemented and requirements for security enhancements studied. This prototype contained a mobile application for the BitTorrent-file sharing protocol [8] and a centralized credit bank implementation. The prototype also contained a BitTorrent tracker that stores information of shared files and their locations.

The mobile peer application was implemented with Java Micro Edition (Java ME). The credit server, which communicated with peers over HTTP protocol, was implemented with J2EE Servlet Technology. For

persistency each credit transaction was written to a RDBMS, which also contained user credentials. Passwords were stored to the database in MD5-hashed form to ensure password security inside the server. Apache Tomcat 5.5 was used as a Servlet runner and MySQL 4.1 as a database server. Mobile application was implemented using MIDP 2.0-standard with JSR 75 extension, providing capabilities to read and store files. This peer application was developed and tested with Nokia E65 having Symbian 9.1 operating system.

The credit system introduces additional messaging for BitTorrent clients. Each time a certain file piece was uploaded or downloaded by peer, the credit server is informed of the transaction. This credit server communication was implemented by sending messages in a BitTorrent-specific bEncoded form over HTTP. This way existing logic for data structure handling in peer applications could be reused. These actions received by the credit server were then written to database, and appropriate accounts were compensated respectively.

The implementation works with existing peer applications without changes to the torrent protocol. Figure 4 shows these existing communication sockets with dashed lines. In addition to these sockets, each peer application communicates with credit server with separate connections. These connections are drawn with solid lines in Figure 4. Using separate sockets for additional credit communication allows peers which have not been integrated to credit system to use existing torrent network without problems. The model also enables torrent tracker to communicate with the credit server. This makes it possible for a torrent tracker to prioritize peers while informing others of content availability. This decision could be based on contributing peers' credit balance and contribution actions.

The credit system requires that communicating peers are able to identify and authenticate each others. The BitTorrent protocol introduces a peer identifier to identify peers from each other, but there is currently no logic to ensure that this identifier is globally unique. Currently peer identifier allocation depends on the BitTorrent client implementation, and several of implementations even use all random numbers while generating this 20-bytes long identifier. In current versions of the protocol there is no structured way for constructing such identifier, although some conventions have been applied in Azureus and Shadow's-styles.

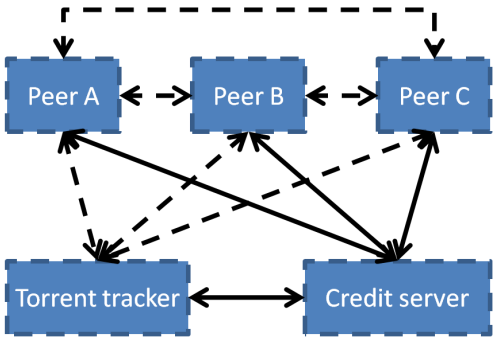


Figure 4. Communication sockets in the prototype. Credit specific communication happens through separate sockets (solid lines). The Prototype does not change existing BitTorrent protocols (dashed lines). The approach enables cooperation with legacy BitTorrent peers.

For the credit system, the peer identifier should not only be unique, but the credit bank should also be able to persistently identify peers between sessions as well. This way the credit server is able to remember contributions performed by certain user between different sessions and terminals. The prototype implementation used user credentials to identify each peer; the credit server user name and password could be changed from BitTorrent peer user interface. Each earned and consumed credit is then handled from a credit account mapped to credit server user name. The communication is protected with a password to prevent non-authorized peers to consume credits earned by someone else. When credit allocation is based on credit accounts, users are able to use various different peer applications even at the same time to earn and use credits.

In BitTorrent, there is not a way to deliver peerid to other peers in trustworthy manner. As discussed in Subsection 4.2, authentication of contributors does not have to be very strong. The costs of attacks are likely to be larger than the costs of contributions. Consumers, however, should be authenticated, so that peers are not able to use credits belonging to other users. Identifiers could be protected with cryptographic methods based e.g. on certificates or shared secrets. This would require either a change in the BitTorrent protocol or an implementation of a new transfer mechanism, in order to authenticate BitTorrent peers. Implementing another socket between the peers is not that reasonable architecture – mobile terminal would need additional server socket and twice as much transfer sockets when compared to existing protocol. Implementing such feature would

also increase resource consumption in the terminal. At the same time, modifying the existing protocol would strict the usage of the peers for specific BitTorrent implementations only.

The prototype does not provide strong peer authentication. Instead of modifying the existing BitTorrent-protocol, the prototype server kept track of peerids, mapped with peers' public IP-addresses. This way the torrent tracker and the credit server always knew the peerid in each tracker request, even when BitTorrent client applications did not explicitly define them. The use of IP addresses brings some protection against attacks where consumer fakes its peerid in order to use other users' credits. These attacks are complex as they must include the network infrastructure so that the files are routed to the attacker instead of the correct owner of the IP address. This approach is relatively simple and requires public and unique IP-address for each peer, which is not currently the case in the real world.

The forthcoming implementations may utilize identifiers, which are allocated by the credit bank. Central allocation of temporary and random user-specific identifiers would also enable privacy as discussed in Subsection 4.5. Also, it is possible to utilize security protocols such as TLS for authenticating consumers and for authenticating peers' communication with the credit bank.

6. Related work

Existing efforts for P2P incentive mechanisms can be classified into three broad approaches: some devices monitor how peers behave, some devices have trusted client software to monitor user behavior, and some devices rely on other peers to monitor how peers behave. Additionally, there are incentive efforts with wireless devices utilizing either trusted hardware or cooperative trust management schemes.

6.1. Monitoring BitTorrent peers

BitTorrent clients already now implement choking or tit-for-tat algorithms [9], which provide an incentive for peers to contribute. The purpose of this mechanism is to enable individual peers to maximize own download rates by selecting best peers. BitTorrent does this by monitoring how much peers contribute and then choking, i.e. temporarily refusing to upload for, those peers providing the worst service.

Each peer selects a fixed amount for peers to be choked once every ten seconds. Peers selected for choking are in principle those, which provide the worst download rate. There is also an optimistic unchoke algorithm. In optimistic unchoke, peers, which have previously provided bad download rate, are given change to provide better performance during a thirty seconds period. If the performance improves over that period, the unchoking of peer is continued. There is also a so called anti-snubbing mechanism. When a client does not receive any data from a peer for one minute, it assumes that that peer has choked it and stops uploading to that peer except during optimistic unchokes.

This tit-for-tat incentive mechanism is essentially file-specific. It provides incentive for peers to share a file at the time they are downloading the file. The algorithm is trustworthy as download rates are measured in the peer, which is also rewarding peers for contribution by uploading content.

When comparing BitTorrent's tit-for-tat mechanism to our credit system, we can see some differences.

Firstly, tit-for-tat does not enable users to upload at different time and download at another time or with another (mobile) device. This means that non-contributing mobile peers will get bad service when tit-for-that model is used. Whereas, the credit system we proposed and implemented for BitTorrent enables collecting rewards at the later time and with different devices. Hence, our model is more suitable for encouraging long term good behavior.

Secondly, BitTorrent's tit-for-tat mechanism is symmetric where a peer rewards only those peers it is communicating directly. In many cases P2P connections are asymmetric. For instance, a peer may be a single contributor of a rare file but may utilize several sources to download a popular file. The proposed credit system is fairer as contributions are evaluated from a point of view of P2P community instead of an individual peer.

Thirdly, tit-for-tat mechanism is vulnerable for selfish peers. For instance, a modified BitTorrent client, BitTyrant [10], showed that peers can receive more that they contribute by carefully selecting contributed peers and upload rates. Also, as noted in [11] the punishment comes within delay and the peers may easily change identities since BitTorrent does not provide strong authentication mechanisms. Our proposal addresses these threats by keeping track of contribution for longer time period and by being compatible with stronger authentication means.

On the other hand, BitTorrent's tit-for-tat mechanism is more efficient mechanism for selecting

optimal peers to communicate with when uploading a particular file.

6.2. Local contribution tracking

Some existing P2P technologies, such as KaZaA, have incorporated own credit mechanisms to client software. These mechanisms track how much the end-user uploads and, according to that information, locally adjust download rates.

These solutions are efficient, easy to implement, and scalable, as they do not require contribution from peers or servers. Also, they keep track of contributions for longer time scale than e.g. BitTorrent's tit-for-tat and thus will provide incentive to contribute also when the user is not downloading.

However, these mechanisms are tied to particular devices and do not consider resource limitations of mobile devices. Also, these local solutions are vulnerable for tampering attacks. For instance, KaZaA Windows clients' participation level information has been stored on Windows registry in obscure format. Users have been able to modify information according to easy guidelines, which are available on the web sites such as [12]. Architectures where trust is not tied to consumer side software, including our proposal, are less vulnerable for tampering.

6.3. Distributed incentive systems

Some research proposals have adopted remote incentive schemes where either a centralized server, as our credit bank, or other peers are used to track contribution and to control which nodes can be provided rewards. Two basic types of approaches for storing and protecting accounted contribution have emerged:

Remote accounts – In these proposals information on contributions is tracked into an account, which is stored in a centralized or distributed repository. The account management is done by a trusted party.

Cryptographically protected electronic currency - In these schemes, peers get tokens from contribution and use tokens to receive service. The advantage of these schemes is that they do not require active participation of a server, which might become a bottleneck. Server's participation may be required only for some operations such as for initial registering to the network and for preventing double spending.

BitStore [13] is one approach proposing remote currency based incentive scheme for BitTorrent. It has been originally designed to address BitTorrent's problem that there may not be complete sources available, especially for rare files. BitStore is a P2P network, which keeps complete copies of content and which is parallel to the BitTorrent network. Participating peers are rewarded with tokens, which are cryptographically protected. The value of tokens depends on an auction based market mechanism.

BitStore is similar to our approach in a sense that it uses centralized nodes to control peers' transactions. BitStore uses trackers as trusted third parties for controlling money and token transactions. However, BitStore does not address challenges of mobility nor bind tokens to particular users, which have been done in our proposal.

PPay [14] has adopted a token based scheme. In PPay, reliance on server is kept on minimum as server is not needed in normal transactions. Frauds are made detectable by leaving to tokens an audit trail, which identifies who has used them. Our proposal is different than PPay in a sense that we require more server interactions. However, by doing this we make double spending attacks proactively impossible. Also, we address one problem of intensive mechanisms (a peer who has collected large amount of credits stops contributing altogether) by enabling adjusting of account balances in flexible manner. For instance, credits could be periodically withdrawn from accounts using some algorithm so that any user cannot that completely stop contributing.

PeerMint [15] introduced remote accounting based solution for an incentive mechanism, which is both reliable and scalable. They used an overlay P2P network to keep store users accounts. In PeerMint, both the contributor and the consumer inform accounts of both peers for a transaction that has taken place. The accounts may not locate in the same peer. To make the system more manageable, PeerMint uses session specific mediator peers, which are informed on contributions during sessions and which at the end of sessions then update accounts of participants.

E-cash [16] proposes a token based approach where users can withdraw tokens from an account in a server, called the central bank. After earning tokens, the user must deposit them to the bank before they can be used. This enables the server to track use of coins and to detect frauds. Sending tokens to server after every transaction and then withdrawing them will also cause additional overhead. The paper [16] does not directly address limitations due to mobility. However, the approach is similar with our proposal when the account holders deposits tokens with a fixed terminal and withdraws them with mobile devices.

6.4. Incentives for wireless devices

Some research work has been given for studying incentives in wireless ad hoc networks. These networks rely on peer nodes to voluntarily route others traffic. Since these nodes typically have very limited battery capacities, they typically have no incentive to do this. Incentive mechanism proposed for wireless ad hoc networks are different from P2P environments in a sense that they are smaller and cannot be assumed to have any long-lived central authority.

Incentive mechanisms based on cryptographically protected electronic currency have been proposed for ad hoc networks. For instance, Nugglets, presented in [17], is a token based approach. This approach does not assume that there is any centralized authority, which would be responsible of issuing or tracking electronic currency. Instead, security in Nugglets is based on tamper protected hardware modules, which are assumed to be available and used in devices.

Trust management systems have also been proposed for incentive in ad hoc networks. In these systems, when a node detects an uncooperative node, it reports this observation to other peers. Peers may then decide not to cooperate with this uncooperative node. Challenges in trust management systems include, as noted e.g. in [18], vulnerability for false reports, complex decision algorithms, and additional signaling. In the context of P2P networks, evaluating nodes cooperative level, i.e. detecting bad behavior, is more challenging as provided contribution depends on factors which are not present in ad hoc networks such as distance between nodes or low capacity communication links. Hence, a peer may be determined to be an uncooperative one despite its willingness to contribute.

7. Conclusions and future work

The P2P credit system provides an incentive for peers to contribute. As the system is centralized and user-identity based it is suitable for users with both fixed and mobile devices. In this journal paper, we explored security challenges and mechanisms for the credit system. The paper extends our ICIW 2008 conference paper [1] with more extensive security analysis and literature survey. A survey and classification of threats against incentive mechanisms was provided. Then, we surveyed requirements and mechanisms for securing the credit system and presented an implementation of the credit system for mobile devices with BitTorrent P2P clients. However, the implementation is not tied to the BitTorrent protocol. In the future, the credit system could support other P2P clients with different protocols.

Every identified attack against the system cannot be prevented. A fundamental security problem in P2P networks is that information coming from individual peers cannot be trusted. This means that with some efforts, an attacker may gain illegitimately credits. However, a reasonable security level can be achieved with a combination of various security mechanisms. At the minimum, architecture must enforce that credits are used and collected in legitimate manner. In practice this requires that contribution is given only for peers with enough credits and that contribution is verified in authentic, preferably in privacy preserving, manner. Also, additional security level may be achieved with misbehavior detection mechanisms as well as with software tamper protection mechanisms.

The effect of the incentive mechanism is that it will make more contributions available. However, it is unclear will this additional contribution justify the overhead, which the related security processing and signaling causes. In the future, this question should be studied with user studies and field trials.

8. References

[1] Jani Suomalainen, Anssi Pehrsson, and Jukka K. Nurminen. A Security Analysis of a P2P Incentive Mechanism for Mobile Devices. The Third International Conference on Internet and Web Applications and Services (ICIW 2008), 2008.

[2] Olli Karonen and Jukka K. Nurminen. Cooperation Incentives and Enablers for Wireless Peers in Heterogeneous Networks. IEEE CoCoNet Workshop Cognitive and Cooperative Wireless Networks, 2008.

[3] John Douceur. The Sybil Attack. International Workshop on Peer-to-Peer Systems, 2002.

[4] Ravi Sandhu and Xinwen Zhnag. Peer-to-Peer Access Control Architecture Using Trusted Computing Technology. Symposium on Access Control Models and Technologies, 2005.

[5] Jari Arkko, Pekka Nikander, and Mats Näslund. Enhancing Privacy with Shared Pseudo Random Sequences. International Workshop on Security Protocols, 2005.

[6] Micheal Freedman and Rober Morris. Tarzan: A Peer-to-Peer Anonymizing Network Layer. 9th ACM Conference on Computer and Communications Security, 2002.

[7] Ian Clarke and Oskar Sandberg. Freenet: A Distributed Anonymous Information Storage and Retrieval System. Workshop on Design Issues in Anonymity and Unobservability, 2000.

[8] BitTorrent Protocol Specification. Version 1.0. September 2006. http://wiki.theory.org/BitTorrentSpecification. [Referenced April 4th 2009].

[9] Bram Cohen. Incentives Build Robustness in BitTorrent. Proceedings of the first Workshop on the Economics of Peer-to-Peer systems, 2003.

[10] M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy, and A. Venkataramani. Do Incentives Build Robustness in BitTorrent? 4th USENIX Symposium on Networked Systems Design & Implementation, 2007.

[11] David Hales and Simon Patarin. How to cheat BitTorrent and why nobody does. University of Bologna Technical Report UBLCS-2005-12, May 2005.

[12] Hack KaZaA participation level – the easy answer. http://www.davesplanet.net/kazaa/. [Referenced April 4th 2009].

[13] Anirudh Ramachandran, Atish Das Sarma, and Nick Feamster. BitStore: An Incentive-Compatible Solution for Blocked Downloads in BitTorrent. The Economics of Networked Systems and Incentive-Based Computing in conjunction with ACM Conference on Electronic Commerce, 2007.

[14] Beverly Yang and Hector Garcia-Molina. PPay: Micropayments for Peer-to-Peer Systems. 10th ACM conference on Computer and communications security, 2003.

[15] David Hausheer and Burkahard Stiller. PeerMint: Decentralized and Secure Accounting for Peer-to-Peer Applications. IFIP Networking Conference, 2005.

[16] Mira Belenkiy, Melisissa Chase, C. Chris Erway, John Jannotti, Alptekin Küpcü, Anna Lysyanskaya, and Eric Rachlin. Making P2P Accountable without Losing Privacy. ACM Workshop on Privacy In The Electronic Society, 2007.

[17] Levente Buttyan and Jean-Pierre Hubaux. Nuglets: a Virtual Currency to Stimulate Cooperation in Self-Organized Mobile Ad Hoc Networks. Swiss Federal Institute of Technology Technical report DCS/2001/001, 2001.

[18] Elgan Huang, Jon Crowcroft, and Ian Wassell. Rethinking Incentives for Mobile Ad Hoc Networks. The ACM SIGCOMM Workshop on Practice and Theory of Incentives in Networked Systems, 2004.

Peer-to-peer Networks: Security Analysis

J.Schäfer
Faculty of Information Technology, Brno University of Technology, Brno, Czech Republic
schafer@fit.vutbr.cz

K. Malinka
Faculty of Information Technology, Brno University of Technology, Brno, Czech Republic
malinka@fit.vutbr.cz

P. Hanáček
Faculty of Information Technology, Brno University of Technology, Brno, Czech Republic
hanacek@fit.vutbr.cz

Abstract

In this work we're dealing with security in highly distributed systems, specifically peer-to-peer networks. We are describing some known theoretical attacks and defenses in these kinds of networks and comparing them with real world data. Classification of attacks realizable in peer-to-peer networks is given. We also discuss influences of their combinations. This could be useful for creating models of peer-to-peer networks' defense and malware spreading. Also we are proposing our new system for automatic downloading and detection of new viruses in peer-to-peer networks, together with all possible extensions.

Key words: P2P, malware, behavior analysis, botnets, DoS.

1. Introduction

This work deals with security problems in decentralized peer-to-peer (P2P) networks, which are part of highly distributed systems. All pieces of knowledge which arises from this research are, sometimes in special manner, applicable to the other types of distributed systems. Discovered information can help us realize some security principles in larger scale, not only in terms of P2P networks.

P2P networks became very popular due to their contents. They contain wide variety of all possible data, including illegal stuff such as movies, MP3 songs etc. Altogether, you get highly dangerous network, which is used by millions of users, who are usually unaware of security and risks of using client software in peer-to-peer networks.

Clients are forced to use special protocols for communication and file downloading, because there is no central server in P2P networks. Smaller subnetworks emerge, in which clients are connected to each other.

Here we get very specific environment suitable for investigating security properties, e.g. specific spreading of malware, monitoring effects on common users or other misuse caused by different types of attacks.

Creating new viruses and worms in P2P networks is often simplified into finding error in specific communication protocol, but in general it is very similar to common environments. Direct misuse of communication protocols presents us with more interesting point of view (in terms of security). DDoS attack can serve us as an example – we will discuss this kind of attack based on impersonation later.

In this article, we define peer-to-peer networks; specify their usage and present basic security problems. The basic enumeration and analysis of attacks is given followed by attack examples from different groups.

The main intent is to verify properties of already known attacks on peer-to-peer networks. Most of these attacks are only theoretical, usually based on number of preconditions. We want to check their power on real world data, as well as implementation requirements.

As far as we know, there are no implementations of attacks in networks we are interested in (DC++). Thus, we choose few appropriate and interesting candidates for our own implementation and further analysis.

We decided to enlarge scope of this article to cover not only P2P networks security overview, but also problems of malware occurrence and spreading, because it is closely connected to other attacks, as we show later.

While primary objective of this article is categorization of malware and attack simulations, it is also shown that phase of gaining information about

infection spreading is very important. This information can be used to improve attacks implementation as well as design of defense against many types of attacks. Due to this fact, we present our system for automatic download and detection of new viruses in peer-to-peer networks, which helps us understand spreading of different types of malware, diffusion of different files and impact on users when infection appears.

In the first section, basic overview of P2P network types and their properties is presented. Individual threats are described in detail. In the second section, effectiveness and feasibility of some theoretical DDoS attacks are verified.

The third section is oriented to problems of viruses in P2P networks. Our system for automatic download and detection of new viruses in peer-to-peer networks is presented. The purpose of this system is getting precise information about state and behavior of P2P networks. Acquired data should show us structure of shared data from malware point of view, which will help us create empiric models of worm spreading.

Better understanding of new strategies of attackers, their methods and tools can be obtained, based on the results of security analysis of P2P networks. Next step of this process is developing an effective defense against these strategies.

2. State of the Art

Definition of P2P network is presented here. Also we describe differences and similarities of their types. Description of attacks on peer-to-peer networks is given.

2.1 Peer-to-peer networks

Peer-to-peer (P2P) [2], can be defined as sharing of computer resources and services among participants using direct exchange. P2P client can ensure direct information exchange, computing time and data sharing. Participant in P2P network acts as client and server simultaneously. For imagination how can be P2P network established see Figure 1.

We've divided some well-known P2P applications into these few groups (separated by usage):

- Cooperation: Geographically distributed teams use communication-based P2P services, e.g. Skype [3].
- Services: Can be moved to places where they are needed more. Distributed service architecture disburdens remote servers.
- Distributed computations: Idle computer resources can be used for greater benefit of whole P2P network, e.g. seti@home project [4].
- Agents: P2P networks enable dynamic merging of power of individual intelligent agents operating on nodes [5].

The most famous P2P networks due such, providing music and movie sharing. Main breakthrough was caused by centrally controlled Napster [6], later replaced by Gnutella and KaZaA, considered as fully decentralized and highly dynamic. There is no central authority in these "new" P2P networks. They are self-organized, with dynamically adjusted structure. Due to the lack of trusted managing authority, P2P represents a great security risk, especially during expansion [7].

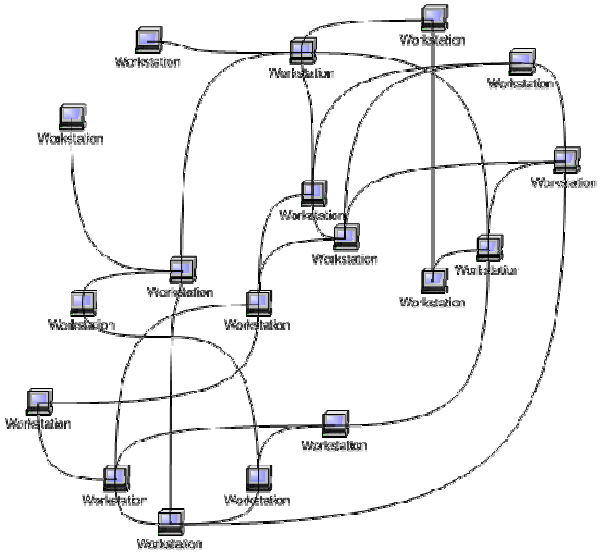


Figure 1 – Peer-to-peer network

Basics of the peer-to-peer file-sharing systems can be described like this: every connected user has its own folder, which contains files user wants to share. Anyone who wants to download file sends a request to all users in the network and then waits. Result contains list of results that matches search query. This list is generated in specific way for every type of network. After selecting one result, client sends request to download. The request is also specific for every peer-to-peer network.

During the file download phase, different approaches are used. One approach is to download from one specific source. Another approach could be downloading from more sources simultaneously. Here, P2P network must ensure content verification to prevent mixture of two different files with the same name. Files are typically downloaded to shared folder

for other users' disposal.

2.2 Attacks on P2P networks

Services are run by specific server or group of servers in standard client – server architecture. The attacker can take down, modify or counterfeit given service by successful attack on only one device. Ebay can serve us as an example: during successful DDoS attack on the main server, no visitor is able to use services of this internet auction centre. All services are closely connected to the server. Attacking the service is equal to attacking the server [7]. But not in P2P networks. Individual participants can be affected by the attack, but services are provided by more of them. So, there is no general effect on whole network. Successful attack on one supernode in Gnutella network does not affect accessibility of files. The only success can be obtained by shutting down the only client proposing specific file. Decentralized P2P networks spread services among all participants. This must be taken into account during security analysis of P2P networks.

Our classification of attacks connected to peer-to-peer networks can be found in Table 1. Selected attacks from table 1 will be discussed later.

Classification of these attacks seems to be a little bit inaccurate, because of their ambiguity. Some of them may belong to more groups than we mention. Nevertheless, the classification is based on the measurement of impact on the destination group (like peer-to-peer users or peer-to-peer network itself) – this means that attack is classified into the group where it can do most damage.

Type of attack	Attack Example
Attacks on Peer-to-peer network	• Listening queries • Filtering queries • P2P network disintegration
Attacks realized through peer-to-peer networks	• Malware spreading • DDoS attack • Setting up Botnets
Attacks on users of Peer-to-peer network	• Content Verification • Anonymity weakening • Stealing Identity

Table 1 - Classification of P2P network attacks

We can see attack summarization and attacks overview in Table 2. Some of these attacks are described later.

Attack name	Target
Leechers	Attack on networks reputation
Social attacks	Attack on users
Searching for sys. files	Attack on users-attack / on peer computer
Listening queries	Observation attack
DDoS attack	Attack on users/attack on peer/attack on other computer
Content verification	Attack on networks reputation
Attack using malware	Combined attack

Table 2 – Attacks overview

2.2.1 Content verification Genuineness verifying of downloaded file is mentioned here, despite it is not an attack at all. Every time we download a file, we must ensure that content of file corresponds to proposed file and doesn't include some unwanted part such as malware. In real world P2P networks, there is no mechanism to ensure this, with one exception – good will of users, which is usually missing [7].

In [8], Jian Liang determines number of fakes in KaZaA. He implemented mechanism for downloading of all music titles, which correspond to latest trends. During analysis of these musical files, he found out that 70% of ones that containing most widespread title "Naughty Song" was depreciated or were fakes.

2.2.2 Listening queries This attack utilizes open architecture of Gnutella network. Dependence on third parties makes it vulnerable to malicious behavior. But if we look at these problems from the perspective of attacking services (not attacking servers), we move to quite different area. In Gnutella network, interconnecting nodes are able to see significant part of queries from all servers in their local sub graph. How much damage can these nodes inflict if they behave badly? Each super node (node with broadband permanent connection to the internet dedicated to routing messages and keeping list of shared files of his sub nodes) can see crucial amount of communication taking place in its sub graph. Gnutella uses 7-jump searching protocol, so every query goes through the whole network via 6 super nodes. If each super node knows about four other super nodes, then it's possible for 1300 super nodes to see this query. In fact, Gnutella architecture is similar to Ethernet broadcast with more than 1300 nodes able to respond to any query. We don't know exactly how serious trouble can be caused by just one node, which is able to eavesdrop this amount of queries and respond to them accordingly to

its will, but it is obvious that in case of compromising such node, anonymity of Gnutella users would go down considerably [7]. Nevertheless, this kind of attack can be carried out in most types of peer-to-peer networks.

2.2.3 Leechers Leechers are P2P network users, who don't use service for file sharing and just downloads data. This type of users does not participate in network's data redundancy and therefore they're usually banned and kicked out of the network.
There are some techniques that could be used to produce enough data to fulfill sharing limits, fake sharing (sharing of non-existing files, modification of communication protocol and another techniques to use P2P network for free [1, 7].

2.2.4 Social attacks P2P networks are mostly used by users with limited knowledge of computer security. Their computers and accounts could be attacked by advanced users mostly by chat service of P2P clients. Attackers can misuse demands of these beginner users for a help to obtaining sensitive data. Typical attacks lead to sharing of whole system drive or to leakage of password and other sensitive data.

2.2.5 Searching for system files P2P users sometimes unintentional share all of his hard drive including operating, system files, application files, registers, private documents and other sensitive data. Some users share this data intentionally to extend amount of shared files to fulfill the rules of some P2P networks. This attack can be also joined with the social attacks – advances users could suggest victims to share their sensitive data.

2.2.6 Attacks using malware Many of described attacks can be combined with malware, for example Content verification with attack using malware.

2.2.7 DDoS attack This kind of attack is very well known, because no defense against this attack exists so far. P2P networks are not an exception. In flooded DDoS (Distributed Denial of Service) attacker abuses a lot of P2P network users, called zombies (see Figure 2-- description of Distributed Denial of Service attack).

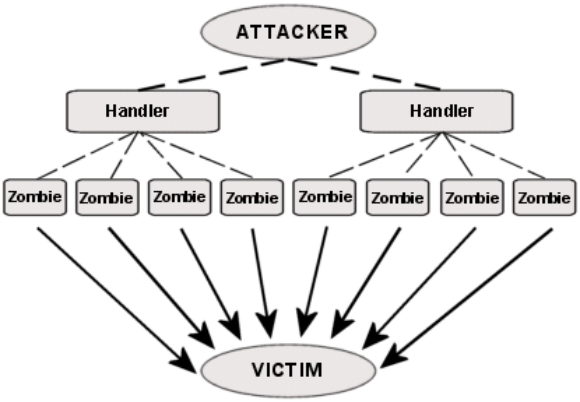


Figure 2 – DDoS Attack

These zombies can send bogus packets to specified target. The main goal of this attack is to exhaust victim's resources, so he is not able to provide or use some services anymore. The key resources are network bandwidth, network latency and TCP resources. From the attacker's point of view, successful attack is not just about exhausting victim's resources, but to use large amount of zombies too. Due to its properties is this attack difficult to detect as well as prevent. In this section we will discuss two main classes of flooding DDoS attack [9].

First one is TCP connection DDoS attack. Main goal of this attack is to exhaust victim's resources by many fully open TCP connections. When normal user tries to use the service, there are no TCP resources left [9].

Second type of attack is called bandwidth attack. In this kind of attack, attacker tries to generate huge amount of packets to overload target's bandwidth. In this kind of attack, UDP, TCP SYN or ICMP packets are used. This attack can be carried out by index poisoning or routing table poisoning [9].

In index poisoning attack attacker inserts fake records into P2P indexing system. These fake records say that target shares very popular and desirable files. The main idea is that the victim has not to be a participant of any P2P network, because each owner of shared files is addressable by his IP address, so the victim can be any mail server, web server or just user desktop. When normal P2P users start searching for these highly wanted files, faked indexes point them to the victim's computer. Then these deluded users try to negotiate download of these popular files, therefore they establish fully open TCP connection and exhaust victim's TCP resources or reach maximum of simultaneously opened TCP connections.

In routing table poisoning attack, attacker tries to insert fake records into routing tables of P2P network. Attacker tries to convince all users that the victim is their neighbor. When poisoned user tries to send a message (for maintaining connection or request query), he chooses a neighbor from his routing table. This is the point when he can choose the victim instead of real neighbor. If we imagine these networks can have about millions of users, even if only a part of it was infected, the communication routed through the victim could lead to the bandwidth DDoS attack [9].

3. Attack simulation

We have decided to implement and simulate only two attacks mentioned above. DDoS attack realized using native DC++ client and Listening queries attack [9], realized using modified client of DC++ networks. All simulations were run in laboratory conditions.

We are at the very beginning of a research on peer-to-peer network attacks; therefore the dimension of simulations is relatively small. First, we need to monitor targeted area, gain more experience with these kinds of attacks and finally, based on acquired data, choose more appropriate candidates for future research.

There is no confrontation of our results with different research groups, because we were unable to find any similar attack implementation.

3.1 DDoS Attack

The goal of this simulation was to implement DDoS attack in DC++ network and check the results of this attack on real data.

3.1.1 Attack resources Network of 20 virtual computers connected to P2P network DirectConnect++ was used for this attack simulation, together with DC++ hub. Hub acts as a supernode used by other nodes to forward their communication. Client programs, operating on nodes, were randomly selected from freely available clients for DirectConnect++, namely CZDC++, StrongDC, DC++. Opendchub version 0.7.15 served as the hub.

Ratio of active nodes to passive nodes was 20:80. By passive node we mean a node without public IP address, all of its communication is forwarded by the supernode. On the other hand, by active node we mean a node with public IP address. Its communication with other nodes is partly direct and partly forwarded by supernode.

3.1.2 Attack description We simulated hijack queries attack by automatic download of non-existing file provided by target client (the one we attack). This state was simply reached by deleting the shared file. Resulting scenario is similar to real attack. Attacker responds to queries. In reply, address of the desired file is substituted by address of target machine, which is obviously not possessing desired file (simulated by file deletion).

In next phase, the group of clients automatically sending requests for non-existing file download was created. The target machine responded by non-existing file error message. This kind of attack was unsuccessful due to the small number of attacking clients. The limited number of clients was caused by laboratory environment. We are working on creating a bigger network with complying parameters, which would allow us to simulate a successful attack.

Different approach to this attack brings us more interesting results. *Force attempt* technique (provided by overwhelming majority of DC++ clients) leads to exhaustion of client resources important for sharing with relatively small number (12) of attacking clients.

Result of successful attack is simple: no one can obtain any other files from target client. In extreme case, communication is affected mutually and target client cannot obtain any files from the rest of participants. This happens when target client is in passive mode – public IP address of this node is not allocated, all communication is forwarded by dedicated node. Taking into account all consequences of the successful attack, reaction of other participants must be expected. There is a high probability of target machine exclusion from P2P network due to constant failures resulting from other nodes attempting to acquire some data.

3.1.3 Results evaluation After detailed problem analysis, we have shown that only a few passive clients (12) were able to break client's functionality. If we want to fully exhaust victim's TCP resources, we should use active clients. It is because of active clients' connection type; active clients are connected via victim's socket server, so when lots of clients try to download non-existing files, victim must fully open this connection and answer that this file is not available. And even after victim sends this error message, previous established connection remains open. This is the easiest way to carry out DDoS attack. This attack was realized with real data and with minimal costs. We

have proved that this kind of attack is very easy to carry out.

3.2 Listening queries

Most users of P2P networks use pseudonyms and only a few of them are recognizable by their IP address. That is why we tried to determine users name or ID by listening to queries, search requests and other communication between P2P users. For this reason we have created a special tool, which is able to connect into DC++ P2P network [10], and act like a normal P2P user. Then we logged and analyzed the communication going through our program.

After analyzing these data, we were able to find out what specific users are searching, what they are downloading and who they are talking to. After detailed analysis, we were able to determine what kind of person it is, what are his hobbies and who is he talking to and which group of people he belongs to.

P2P network DirectConnect++ has slightly different structure than other P2P networks, so we were able to gather communication from only about 9500 users from one hub (hub is something like supernode in other P2P networks). So it is not the true attack on P2P network, but it is an attack which can lower the anonymity of users in this kind of P2P network. Though P2P networks do not guarantee anonymity, most of the users use some kind of pseudonyms and try to conceal their true identity. That is why the possibilities of user privacy compromise must be taken into account.

We have proved that even if user uses pseudonyms there are techniques that can help us to reveal his true identity.

4. Malware behavior analysis in P2P networks

This section deals with malware occurrence in P2P networks, especially with the possibility of compromising particular nodes and further exploitation. Attackers are trying to compromise more computers, which can be later use for further infection. They acquire control of these machines by using specific malware.

If we know malware true behavior in peer-to-peer network (propagation model, speed and impact on users), we can predict every possible consequence and lower the impact in case of real infection.

4.1 Botnets

Recently we are experiencing change in a way how attackers compromise some systems: wide spread worms, which infect hundreds or thousands of machines (similar to CodeRed [11] or Slammer [12]), are rather rare. This behavior can be caused by two main reasons. First, worms do not offer the attacker any means of remotely controlling attacked system – once the worm spreads, attacker cannot redirect the attack or even add some additional commands to worm. Second, attacker gains no financial benefit from releasing the worm. Ten years ago, most attackers were motivated by technical challenges or by effort of proving vulnerabilities, today most of attacks are motivated by money. Botnets are currently one of the serious internet problems [12].

Botnets can be defined as networks of compromised computers, which can be remotely controlled by the attacker. Every compromised machine (called bot) has a special program installed, which is remotely controlled by the attacker. Typical examples of these „remotely controlled networks“ are IRC networks and http servers. Few years ago, botnets based on P2P networks appeared. These botnets can be used to perform malicious activities, e.g. DDoS attack, sending spam, phishing, stealing important data or further spreading some malware [14][15].

4.2 Worm spreading in P2P networks

In order to develop the countermeasures, we are interested in model of malware and worms spreading. Studying worms in the phase of propagation is important for various reasons. First, warning systems capable of detecting worms can be created and (ideally) preliminary analysis of propagation can be given. No such system exists presently and it will take a while to deploy one. Second interesting aspect is threat analysis according to spread rate and number of hosts which can be infected by the worm. Last, but not least, we can stop quick establishing of large botnets by appropriate filtering out the worms.

Because P2P networks already have an established structure, these worms do not have to search for new victims by scanning (e.g. random scanning). Also these worms do not make large number of unsuccessful connections and their communication can be integrated into other ongoing communication. In [16], Hiestand showed that detection systems based on worm analysis

are not able to distinguish worm communication from communication of other subjects [17].

According to the way of propagation, we can divide worms into two categories:

- P2P worm using topological scanning: In this case worm takes advantage of information obtained from victim about his neighbors. This strategy can significantly speed up worms spreading, because he does not have to be bothered by scanning his possible victims. To improve this way of spreading, worms might use so called "hit-list" (list of victims). In case of collecting addresses before releasing, the worm gains more time in the early stage of attack. Once user is infected, he becomes involved in spreading the infection among users from hit-list, until the list is eventually empty. But no such worm appeared so far.
- P2P worm using passive scanning: These correspond to the current type of P2P worms. They do not actively search for victims, the just stay in shared folder on the infected machine. Being downloaded by another user, they begin to replicate and infect more shared files. As an example, we can mention worm Benjamin. Gunman worm works in slightly different way – he positively responds to all search queries by renaming infected file according to the query. When client downloads and runs this file, worm infects files of unsuspecting client [17] [18].

4.3 System for automatic file downloading from P2P networks

It is crucial to find out empiric model of malware behavior in P2P networks, as we proposed in previous section. It is important to find these models because precautionary measures and detections systems can be built on these empiric models. These precautionary measures and detection systems can help us with detecting of new malware spreading. These systems, which are able to analyze future propagation of spreading malware, do not exist, so far. The first part of creating on this system is to create empiric model of malware spreading. For creating such model we need a system, which allows us to access a large amount of data in P2P networks.

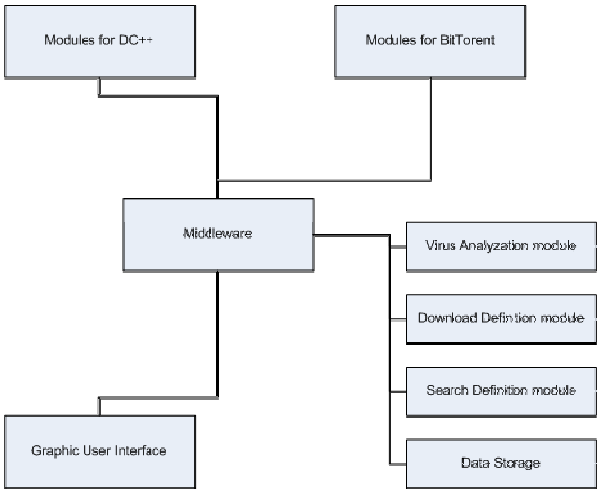


Figure 3 – System architecture

We need a system, which can tell us what each participant of P2P network is sharing, and system, which can help us analyze these data. That is why we designed a system, which can access a BitTorrent and DC++ P2P network, analyze traffic in these networks and can access the data shared in these networks. Most of this system is already implemented. System's design is described on Figure 3.

In this system, BitTorrent module cooperates with internet BitTorrent search engines, downloads the newest torrents, analyzes them and finally decides if it is desirable to download and analyze these files or not. This system also works in DC++, where it analyzes the search results and compares file hashes to decide if it is desirable to download this file or not.

The whole system is based on special architecture, which allows us to join two different P2P networks and gather search result from both of them. Then it can compare them (in case of positive infection in one network, we are able to search for similar files in other network). This system is based on modular approach, so virus analysis is similar for both types of networks and we are able to gather results based on the same metrics. Finally, these results can be afterwards discussed.

This system can be described as a group of cooperating independent modules, which are strictly specialized. System consists of a few groups of modules: module for communication with P2P networks, module for download and search definition, module for virus analysis, module for communication analysis and module for maintaining communication between other modules.

Individual modules are implemented independently.

They work on different platforms in different network due to proposed architecture. Modules communicate via own adaptive communication protocol transported by Middleware. Protocol enables special routing which allows duplicate work of modules belonging to the same functional group and mutual redundancy in case that one module disconnects from the network.

We have implemented only modules for two P2P networks so far, but this system is not limited only to these. System is highly scalable and these two networks were implemented first, because there are open source clients for them, but other networks will be implemented soon.

Thanks to this system we are able to download and analyze the newest files available in P2P networks. This kind of data is very valuable for building empiric models of malware spreading. For example, when we run into some infected file while randomly analyzing some new files, system notices it and starts a full analysis of this file, spreads this file metadata through all connected P2P networks and tries to find out the level of diffusion of the file across these networks. Then system does this diffuse analysis regularly. Thanks to this approach we can acquire specific information about the speed of file spreading, number of users involved and how long this file remains on infected users' computers. With this information we can adequately design and build an empiric model of file diffusion in peer-to-peer network. Obtaining empiric models for different kinds of P2P networks and different kinds of files is very difficult, because we need lots of data. That is why we can spread some fake files (with very popular names) by ourselves and then gather results by observing spreading of these files.

We present only basics of this system in this article, because we do not possess enough result data yet. But we have already gathered some data that can lead us to more promising results.

5. Conclusion

In this article, we dealt with relatively well-known questions of network security in relatively less common environment of distributed networks, particularly DC++. We described problems of P2P networks security with focus on particular attacks.

We have successfully proved that some attacks on peer-to-peer networks, more precisely on peer-to-peer users, can be carried out with minimal efforts and price. We were able to simulate some well-known attacks with very good results in real networks. It proved great vulnerability and low resistance of these networks. For example, it is possible to deny the access to service by DDoS attack even with small number of attacking machines. There is no need to use large botnet to carry out DDoS attack in peer-to-peer network DC++, all you need is just a few users participating in this network. We have proved that some of theoretically designed attacks can be realized very easily and that they are very effective against peer-to-peer network users.

New questions, related to the privacy of P2P network users, were opened during our research. We have pointed out that this issue has not been sufficiently discussed, for example previously proposed easily realizable DDoS attacks, or listening queries attacks, which leads to lowering the anonymity mainly there, where users from social networks, share data, but try to stay anonymous (hiding behind nicknames). We showed that it is quite easy to collect data about communicating entities and get enough data to be able to make judgments about user's identity and behavior.

We have proposed basic categorization of attack on peer-to-peer networks in this article and we have shown some basic attacks and their analysis and evaluation. We have separated attacks into a few groups, attack realizable thru peer-to-peer network, attacks on peer-to-peer network and attack on peer-to-peer users and we have detail described each kind of attacks.

In last section we presented our tool for automated file download. We showed its basic structure, its possibilities and its worth for creating and verifying empirical models of worm propagation in P2P networks. This area of research gives many (still open) questions, which we want to devote to in our future work.

This research was supported by the Research Plan No. MSM, 0021630528 -- Security-Oriented Research in Information Technology.

6. References

[1] Schafer Jiri, Malinka Kamil, Hanáček Petr: Peer-to-peer networks security, In: The Third International Conference on Internet Monitoring and Protection, Bucharest, RO, IEEE CS, 2008, s. 13-13, ISBN 978-0-7695-3189-2

[2] N. Minar, M. Hedlund, C. Shirky, and others. Peer-to-Peer: Harnessing the Power of Disruptive Technologies, O'Reilly, March 2001.

[3] Skype Limited. Skype. http://www.skype.com/, May 2008.

[4] Seti@home, http://setiathome.berkeley.edu/, May 2008.

[5] D. DeFigueiredo, A. Garcia, and B. Kramer. Analysis of peer-to-peer network security using gnutella. Technical report, University of California at Davis, University of California at Berkeley, National Energy Research Scientific Computing Center, Lawrence Berkeley National Laboratory, April 2002.

[6] LLC Napster. Napster. http://free.napster.com/, May 2008.

[7] V. lachos, S. Androutsellis-Theotokis, and D. Spinellis. Security applications of peer-to-peer networks. Technical Report 2, New York, NY, USA, 2004.

[8] Y. Xi K. Ross J. Liang, R. Kumar. Pollution in p2p file sharing systems, 2005.

[9] Kalafut, A. Acharya, and M. Gupta. A study of malware in peer-to-peer networks. In IMC ’06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement, pages 327–332, New York, NY, USA, 2006. ACM.

[10] DCFORGE. DirectConnect++, http://www.dcforge.com/, May 2008.

[11] Steve Friedl. Analysis of the new "Code Red II" Variant. http://www.unixwiz.net/techtips/CodeRedII.html, May 2008.

[12] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, N. Weaver. Inside the Slammer Worm. IEEE Educational Activities Department, 2003.

[13] T. Cymru. The underground economy: Priceless. Technical report, ;Login: vol.31, no.6, December 2006.

[14] J. Goebel, T. Holz, and C. Willems. Measurement and analysis of autonomous spreading malware in a university environment. In Bernhard M. H´nmmerli and R. Sommer, editors, DIMVA, volume 4579 of Lecture Notes in Computer Science, pages 109–128. Springer, 2007.

[15] J. B. Grizzard, V. Sharma, C. Nunnery, B. B. Kang, and D. Dagon. Peer-to-peer botnets: overview and case study. In HotBots’07: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets, pages 1–1, Berkeley, CA, USA, 2007. USENIX Association.

[16] C. Göldi and R. Hiestand. Scan detection based identification of worm-infected hosts. Technical report, Institut fur Technische Informatik und Kommunikationsnetze, April 2005.

[17] Wagner, T. Dübendorfer, B. Plattner, and R. Hiestand. Experiences with worm propagation simulations. In WORM ’03: Proceedings of the 2003 ACM workshop on Rapid malcode, pages 34–41, New York, NY, USA, 2003. ACM.

[18] N. Khiat, Y. Carlinet, and N. Agoulmine. The emerging threat of peer-to-peer worms. Technical report, University of Evry, France, September 2006.

User Preferences to Support Privacy Policy Handling in Pervasive/Ubiquitous Systems

Elizabeth Papadopoulou, Sarah McBurney, Nick Taylor, M. Howard Williams, Yussuf Abu Shaaban

School of Math. and Comp. Sciences, Heriot-Watt University, Riccarton, Edinburgh, UK

{ ceeep1, ceesmm1, nkt, mhw, ya37}@macs.hw.ac.uk

Abstract

An important approach for handling user privacy in ubiquitous or pervasive systems is identity management, in which the user has a number of different virtual identities that conceal his/her real identity. One extension of the basic approach identifies the private data needed by services and uses the notion of a privacy policy to determine what access should be granted to private data by a service. This can then be used to determine an appropriate virtual identity. However, this is fairly complex and difficult for a naïve user to set up and control. Thus a major challenge lies in determining to what extent the decisions relating to the selection of virtual identities can be done automatically, and to what extent the user needs to be involved. This paper describes an approach in which user preferences are used to assist in taking these decisions - both to generate privacy policies and to select an appropriate virtual identity. The user preferences are simpler for the user to create and modify and are also easier for automatic learning techniques to update. This approach will help to create more user-friendly and acceptable identity management systems. These ideas have been explored within the Daidalos pervasive system while further work is being carried out for the Persist system.

Keywords: *User preferences; privacy; pervasive systems; policies*

1. Introduction

The importance of security and privacy in ubiquitous and pervasive systems is universally agreed. This paper is an extension of initial work in this area that was presented in [1].

The original vision of ubiquitous computing portrayed an environment surrounding the user that is filled with computing entities, supporting the user in a variety of ways without continual direction [2]. Since then there have been significant developments in areas such as sensor technologies and communications that are bringing us closer to enabling these predictions to be realised. The problem for the user lies in the increasing complexity that needs to be managed. Pervasive computing seeks to address this and to enable the user to control and manage this situation [3]. Although a successful system has not yet emerged, there is a growing view that during the next decade or so acceptable solutions will be found for many of the outstanding problems facing ubiquitous and pervasive computing and by 2020 this technology will be a reality. In the meanwhile some major problems still lie ahead and some of the global challenges of the next decade lie in this area [4].

In order to develop a pervasive computing system that is acceptable to the end user, it is important that it should satisfy two end user requirements:

(1) It should take account of the user's needs and preferences in any relevant decision making. With any system as complex as this, it is essential that it should adapt its behaviour according to the individual needs and preferences of the end user. The importance of incorporating user preferences is generally accepted and has been recognized in a number of projects, where preferences are either entered manually by the user or where learning is used to support the acquisition of preferences.

(2) It should adequately protect the privacy of the user. Both users and services need to know which services they can trust and what information they can share with them. This starts with the identity of the user – and whether or not the user is prepared to reveal his/her real identity to a service. To handle this, a

system of virtual identities may be used. The other aspect of this is authorisation – deciding who should be given access to what. To handle this an extension of the basic idea of virtual identities requires the pervasive system to identify the private data belonging to a user (e.g., location, credit card details) that a service wishes to access and then determine what access should be granted to the service and under what conditions. This may be achieved through the use of *privacy policies*.

However, privacy policies are fairly complex and difficult for a naïve user to set up and control. This paper describes how these two requirements can be brought together to use user preferences to support this aspect of privacy. It describes how user preferences can be used to generate privacy policies and to select virtual identities. These preferences may have the same format as other user preferences in the system and are therefore simpler for the user to understand and hence to create and modify. They are also easier for automatic learning techniques to generate automatically and for the user to understand what has been produced by the learning system. Thus by building up a flexible context-aware set of user preferences that can be used to assist in taking these decisions, one can increase the degree to which this can be handled automatically, and improve acceptability by the user.

These ideas have been explored within Daidalos, a European research project, a major aim of which was to develop a pervasive system, focussing especially on mobile users, in which security and privacy are key components. This work is being continued in Persist, another European research project developing a pervasive system based on Personal Smart Spaces.

This paper outlines briefly the approach and how it is being implemented. The next section provides a brief background to personalisation and user preferences and to privacy and pseudonymity followed by a brief introduction to Daidalos and Persist. Section 3 concentrates on virtual identities and the process of selecting an appropriate one. It also outlines how user preferences and personalization can assist in the automatic selection of virtual identities. Section 4 describes user privacy preferences while Section 5 presents more detail on the formats of the user preferences for virtual identity selection. Section 6 discusses some research challenges for the future, while Section 7 provides a brief conclusion.

2. Background

This section describes some of the background relating to this research. It gives a brief overview of issues relating to personalisation and to privacy, and then describes briefly the two projects, Daidalos and Persist.

2.1. Personalisation and User Preferences

The importance of individual user preferences and their application in decision making in a ubiquitous environment is generally accepted. In such an environment *personalization* refers to the process of creating, maintaining and applying user preferences in decision making since it has the effect of tailoring the system's behaviour to the individual needs and wishes of the user so that it appears or acts differently for different users or for the same user under different circumstances.

Thus far the work done on different ubiquitous/pervasive systems has incorporated personalization techniques with varying degrees of success. Early developments concentrated on the use of context information rather than on user preferences – producing a context aware rather than a personalized approach. However, the importance of incorporating user preferences into the decision making was soon identified and projects such as the Intelligent Home [5] and Blue Space [6] implemented both context awareness and personalization - although they relied on user input of preference information, resulting in minimal sets of user preferences.

The problem of capturing and maintaining user preferences was soon recognized and the need to assist the user in this process was established as an important requirement for future systems. As a result, projects such as the Adaptive House [7], GAIA [8] and MavHome [9] use monitoring and learning algorithms to gather preferences and environment information such as user movement and actions which are used to predict future movements and actions. Based on predictions, environments are automatically adapted by applying user preferences. However, removing the possibility for user input reduces user control which may lead to confusing or frustrating situations.

The Synapse project tries to find a balance between automation and user control by operating in active or passive mode [10]. If a preference has a probability above some threshold, it is applied automatically in active mode, whereas passive mode consults the user with suggestions before preference application. Although this can produce more accurate personalization, there is a risk that the user could be inundated by pop-up messages.

More recently, projects such as Ubisec [11], Spice [12] and Mobilife [13] aim to provide the user with

personalized services in a global environment. Once again preferences are applied automatically but improved personalization is provided by implementing more responsive implicit personalization mechanisms, which respond rapidly to changes in the user's behaviour patterns and update the set of user preferences in real time.

2.2. Protecting Privacy

Privacy can be regarded as "the right of individuals to protect their ability to selectively reveal information about themselves" [14]. Much work has been done on privacy in the context of the Web and four specific requirements for designing privacy protection have been identified. These are: anonymity, pseudonymity, unlinkability and unobservability [15][16]. Pervasive systems have a lot in common with the Web and the same requirements apply.

A number of papers (e.g., [17][18][19]) have been written on the design of privacy aware ubiquitous systems, reporting on their analysis of end-user requirements and the approaches they follow in order to satisfy them.

One of the important requirements is that there should be simple and appropriate mechanisms for the user to control the release of information. To this end, the notions of pseudonymity and anonymity have been adopted.

Pseudonymity is used as a tool to hide the user's identity from services and in so doing conceal the user's digital trail in a pervasive world. At the same time, a pervasive system that allows such mechanisms, should also cater for accountability and should provide mechanisms to protect the user's privacy without encouraging the user to avoid being held accountable for his/her actions [14].

Pseudonymity is useful in online transactions since not every service that is being used needs to identify the user. Authentication does not imply identification. The notion of separate personas, private and public, have been proposed which place different restrictions on the information they release to services [17]. This concept is similar to that of virtual identities, in which the user has a number of virtual identities to protect their real identity. One difference between them is that personas are created based on user preferences and service trust levels while virtual identities are created to match service trust levels and service privacy policies.

Anonymisation goes one step further. Kobsa and Schreck state that anonymisation hides the relationship or linkage between an individual user and his/her stored personal data [20]. With anonymisation, users are never identified and while this works for privacy, it does not allow dynamic personalization or learning of user preferences. Anonymity also creates more problems than it solves due to the fact that it cannot provide accountability [14]. On the other hand, pseudonymity provides a balance between protecting the user's privacy while at the same time offering advanced personalization practices. By using different pseudonyms for different service transactions, pseudonymity provides additional protection to the user's privacy as it partitions the user's interactions and thus hides any direct link between those interactions [21].

Pseudonymity is not sufficient unless unlinkability and unobservability are also satisfied as requirements. If pseudonyms of a user can be linked to each other then the transactions made with one pseudonym belong to the same user that made the transactions with the rest of the linked pseudonyms. This results in gathering of a vast amount of information about the activities of the user, allowing access to the identity of a user from unauthorized services and revealing personal data to unauthorized parties. Unobservability requires that any attacker monitoring the users' interactions cannot identify which interactions belong to the same user [21]. Unobservability becomes more crucial as a requirement when thinking in terms of the user of a pervasive system. The user can be monitored more easily than a user of a traditional system because of the amount of context information maintained about the user in the system.

2.3. Daidalos and Persist

Daidalos is a large European research project, whose overall aim was to create a pervasive environment for mobile users [22]. This was achieved by integrating a range of heterogeneous networks and devices and creating a pervasive system on top of this which protects the user from the complexity of the underlying infrastructure while providing personalized and context aware services with minimal user intervention.

Within Daidalos a layered approach was adopted, separating the lower level network functionality from the higher level pervasive system. Security and privacy were a priority area in Daidalos and affected all components.

At the higher level the pervasive system depended on personalization and user preferences. Initially a simple approach was used to capture user preferences but later this was extended to use both stereotypes and learning to capture the preferences effectively [23].

These preferences are used for Service Discovery and Selection, Service Composition, Session Management, Security and Privacy, Context Management and Personalization [24]. In addition, user privacy is essential, and this functionality affects both knowing who is running what services and controlling access to user data.

On the other hand, the Persist project is a much smaller project, funded under the Seventh Framework. It builds on some of the work initiated in Daidalos. Once again it aims to produce a prototype pervasive system, but this one will be based on the notion of a self-improving Personal Smart Space (PSS). The vision of Persist is that a Personal Smart Space will replace the fixed smart spaces associated with buildings and the mobile ad hoc networks associated with users, creating a single uniform approach. This will provide an interface to link users to the various services and devices that surround them, as well as to other neighbouring Personal Smart Spaces.

The notion of a smart space is usually associated with a real physical space. From this point of view a smart space can be defined as "*a multi-user, multi-device, dynamic interaction environment that enhances a physical space by virtual services*" [25][26]. The services are the means of interaction between participants, objects and the smart spaces. A Personal Smart Space extends this notion but is not necessarily associated with a fixed location. It is a collection of devices connected in an ad hoc network that may be fixed in a particular location (e.g., a room, office, house, etc.) or may move around with the user. It may even be composed of devices located in different locations but associated with the same user. This type of architecture has some significant advantages over the more conventional approaches.

3. Using User Preferences to Select Virtual Identities

Pseudonymity is achieved in Daidalos through the use of multiple Virtual Identities (or VIDs) [27]. These VIDs form subsets of the user's profile and are used to authenticate the user with services. For any user the set of VIDs may be viewed as a set of different user names, which the user may use for different purposes, and which may conceal all or part of his/her real identity. Each user may have any number of VIDs. None of the user's VIDs can be linked to any of the others so that if a user uses two VIDs with the same service, that service will treat these as two different users. By not providing a direct link between all the services used by a user, user monitoring services will not be able to trace all of the user's transactions, and as a result the user's privacy is protected. This also allows for good personalization practices since users can use services for different activities and have different preferences for each activity.

Although the services that the user may use can only see the user's virtual identity and whatever subset of personal information the user allows, deep within the system in the Security and Privacy component the virtual identities can be mapped to real identities for the purposes of accounting.

When the user switches on the system and authenticates him/herself a default VID is used. Once the user is authenticated, he/she can request a service. In setting up to use the service an appropriate VID needs to be selected for the purpose.

A VID may be created in one of two ways. It may be created explicitly by the user (using a Graphical User Interface) or implicitly by the user setting up specific preferences that allow the system to create a VID based on these preferences and to be used in specific contexts. Selecting a VID to be used presents more challenges than creating it. However, creating a VID can be a part of the process of selecting a VID as will be presented later.

Initially one can make the simple assumption that the user will always select the appropriate VID before requesting any service. However, this assumption is too simplistic and puts an unnecessary burden on the user. In particular, as the user accumulates VIDs, this will become increasingly arduous, just as remembering different user names and passwords for different Internet sites has become a problem. The situation is further complicated by the fact that the use of different VIDs may depend on the context of the user. For example, the user may have two different VIDs for the same service, one associated with work use and the other for use at home. It is essential, therefore, that the system itself should manage the automatic selection of VIDs wherever possible and only require action from the user when it cannot take a decision or when the user disagrees with the decision taken. This is important in order to realise a user-friendly pervasive environment that is acceptable to the user.

A VID is more than just a user name but also defines the set of user data that can be made available to a service. Consequently before selecting a VID one needs to establish what data the service will want to access. Then, if the system is not willing to provide all the access requested by the service, or there are any

constraints that the user has, it needs to negotiate with the service and reach an agreement about the access rights that will be granted to the service and the conditions under which these are granted.

The user's wishes with regard to access to personal data items need to be formalised in an appropriate way. Thus for each item of personal data any constraints that the user may want to place on access to it are captured and expressed in the form of a Privacy Policy. The latter is used as the basis for negotiating with the service and this process of negotiation is referred to as Privacy Policy Negotiation (PPN). If an agreement can be reached the result is referred to as a Privacy Policy Agreement. Once this has been agreed an appropriate VID can be selected. Thus the whole process can be broken down into four steps as follows.

(1) *Establish requirements for data*. The most important factor in determining the VID to be used in any particular situation is the user data that a service needs to access and the access rights (read, write, update) it requires. Thus when a service requests a VID, the first step is to determine what user data the service will want to access. This can then be compared with the user's instructions on access to his/her data as expressed in the privacy policies. This specifies what access should be given to any item of data and under what conditions this access may be granted – for example, the length of time this data may be held or whether or not it may be shared with other services.

(2) *Negotiate use of data*. Once the data requested has been matched against the privacy policies, the system may need to negotiate with the service to ensure that whatever user data is provided to the service will only be used in accordance with the user's wishes. To do this the set of privacy policies is passed to the Negotiating Agent to negotiate with the service on behalf of the user the terms of use based on these outcomes. This negotiation should result in an agreement that meets all the requirements in the privacy policies. This process is similar to that of trust negotiation [28].

(3) *Match PPN outcomes with potential VIDs.* Whether or not the Negotiation Agent needs to conduct a negotiation with the service, the system establishes a Privacy Policy Agreement which consists of a list of private data items (such as personal information, context attributes or preferences) that the service can access. This list is then used to identify the set of possible VIDs that will allow access to all of the items in the list without providing access to much else of significance.

This should result in the identification of one or more VIDs that can be selected for use with this service. If no VIDs are found which would provide the required access to the data items in this list, the user needs to be consulted. A GUI is invoked and the user is given the option of changing the data access constraints associated with an existing VID or creating a new VID with the required data access properties. Alternatively the user could select a different service as there is no way in which the current service can be run with the VIDs that are available.

(4*) Select final VID.* Once the set of possible VIDs has been established, the final VID can be selected from this list. This process happens even when only one matching VID is found.

4. User Privacy Preferences

In order to make use of user preferences in this process, the approach described in the previous section can be extended with an additional step at the beginning of the process in which user preferences are used to generate the privacy policy. One way of viewing this is to divide step (1) as follows:

(1a) *Establish data item requirements*. Here the pervasive system needs to determine from the service what items of private data it wishes to access.

(1b) *Generate privacy policies*. From the user preferences relating to these particular data items, generate a set of privacy policies.

For this purpose one may have a set of preferences, referred to as *User PPN preferences* that define what the user wishes in each situation. These may depend on external factors such as context conditions (e.g., the user's location, activity, people in his/her proximity, etc) or service-specific conditions (e.g., reputation of service). To aid the user privacy preferences, service trust levels are maintained in the system for each user. Some users may believe that service X is sufficiently reliable that it can be trusted with confidential personal information while others on the other hand might not trust it and only be prepared to provide access to limited subsets of information (or possibly even none at all). To aid the user, the system can maintain service trust levels for each user. These can be used as part of a condition in a user PPN preference. In each case, the result of evaluating a PPN preference tells the system whether or not a piece of personal data can be allowed to be disclosed, and under what conditions. The evaluation of these PPN preferences for all the requested user data results in a set of privacy policies.

This set is used by the Negotiating Agent to negotiate with the service on behalf of the user the terms of use based on these outcomes.

One can also use user preferences in the final step to select the actual VID. User VID selection preferences are used to determine which VID to use. User VID Selection Preferences define the circumstances under which a VID should be used and with what kind of service. The outcome of the evaluation of these preferences will state that a specific VID should be used in a specific situation.

This means that these preferences contain references to actual VID identifiers in contrast with other user preferences in which there are only references to specific context data. In the case of a new situation where a VID cannot be determined from preferences, the system should explicitly query the user at this stage and offer the list of VIDs for the user to choose from or allow him/her to create a new VID for this situation.

Thus two different types of preference rules may be used to support the user in the process of selecting a VID, namely *User PPN preferences* and *User VID Selection Preferences*. In the Daidalos system, there is also a third type of privacy related preference which is used for "context obfuscation". These are used in the case of certain context attributes where the level of detail that is returned may be controlled. For example, in the case of a request for location information, if one is in one's office at university, one might return the office number or one might return the building or possibly just the university or even the part of the city or the city itself. Thus these preferences determine the accuracy to which context items are delivered to services. However, they are beyond the scope of this paper.

5. Formats of User Privacy Preferences

The format of the privacy policies is based on the industry standards P3P [29] and XACML [30]. It also has the facility for users to create their own custom privacy preferences.

On the other hand the formats for the user PPN preference rules are the same as those adopted for all preference rules in the Daidalos system so that they can be easily understood and manipulated by the user, a basic requirement in designing privacy aware systems [31]. This consists of a simple "if-then(-else)" rule in which the condition part is a Boolean expression comprising one or more simple conditions such as checks on context attributes, the status and attributes of other services being run by the user, attributes of the service requesting access to the data, previous usage of the requesting service, trust levels of the service which have been formed by the user and/or groups of users sharing such trust levels. Each then-part or else-part may be either an action or a nested if-then(-else) statement. This has been fully specified but constraints on space do not permit a fuller discussion on this here.

The action part of a user PPN preference rule contains a list of the conditions that govern the disclosure of a piece of user data to a service. Thus the outcome after evaluating such a preference would be either positive (i.e. disclose the piece of data), negative (i.e. do not disclose it) or a conditional expression of the form "positive if a list of requirements is met". These latter requirements are conditions such as the data retention policy of the requesting service, the data usage policy of the requesting service and other such conditions subject to negotiation with the service.

A PPN preference rule is associated with a single data item. Thus for each data item or context attribute associated with a particular user, one may have a separate rule. To illustrate this consider the following example:

IF symbolic_location = 'work' AND
dayOfWeek = 'weekday' AND
LocalTrustLevel(requestor) >0.5 AND
GlobalTrustedReputationLevel(service) > 0.7
THEN PrivacyPolicyRule:
Effect: "Permit"
Obligations:
1) Data_Retention_Policy<12 hours
2) Share information with 3rd parties: NO

The above example of a PPN preference in our format is a rule with four conditions. The first two conditions are context conditions which specify that the rule should be enforced only when the user's symbolic location (in a high level form inferred from raw sensor data such as GPS coordinates, as opposed to the original raw data) is "work" and the current day of the week is a weekday. We are aware of the complexity associated with inferring symbolic location from the raw location data but this is dealt with by others within the Daidalos system and is beyond the scope of this paper.

The following two conditions require the use of other components in the system such as the local trust handling component and the global reputation system. In the first case, the local trust component refers to a

system that gathers information about previous transactions with services by monitoring the use of the service by the user and prompting the user to provide such information. Each service is assigned a trust value within some range of values to indicate the level of trust the user has in the privacy practices of that service. In the case that a service has not been used before, the local trust component can deduce a range of trustworthiness for a service by examining the trustworthiness of services with similar privacy practices and guarantors. In the example given, the preference states that the local reputation value should be no less than 0.5. This value, the service reference and a number of parameters pertaining to the service are given to the local trust system to evaluate whether it is true or not.

The fourth condition refers to a global reputation system that a user can query to acquire the trustworthiness level of a service as viewed by a collection of users. This value has been calculated by combining the trust values submitted to that system by a number of different users. Thus when this condition in the preference rule is evaluated, a query is forwarded to one or more reputation systems to acquire the trustworthiness of this service as judged by other users. The values returned are averaged based on an algorithm that takes into account the user's indicated trust in these reputation systems. The final value is checked against the value stated in the preference (in our example the user has indicated the value of 0.7).

The outcome of a PPN preference specifies what should happen if the overall condition is met. In the example, the preference states that the service should be permitted to access the specific piece of data if and only if a number of requirements are met by the service itself. The outcome of a PPN preference is a collection of requirements that the system will negotiate with the service. If the service does not agree with these requirements during the negotiation process, the data item will not be disclosed to the service. In the example, two requirements are specified for the service to agree to. The first declares that the data item value should not be logged by the service for more than 12 hours since the first initiation of the service. This is required so that services do not accumulate a large history of the user's context information that can result in compromising his/her privacy. The second requirement states that any disclosed information will not be forwarded or sold to third parties by the service.

It should be noted that it is possible that more than one preference can exist that determines whether or not a specific piece of data can be disclosed. For example, the user can set a preference whose effect is to deny disclosing his/her location if certain conditions are met. Depending on the conditions specified in the preferences, it is possible for the system to result in two outcomes, one that allows the disclosure of the information and one that does not. In this case, the system will not allow the disclosure of the information because a rule with effect "Deny" has precedence over all preferences with effect "Permit".

If the conditions are met, the outcome will be translated on the fly to the following XACML policy snippet:

```
    <Obligation ObligationID=”data_retention”
FulfillOn=”Permit”>
    <AttributeAssignment AttributeId=”data_retention”
DataType=
    ”http://www.w3.org/2001/XMLSchema#Duration”
>
    P0Y0M0DT12H0M0S
    </AttributeAssignment>
    <Obligation ObligationID=”3rd_pty_disclosure”
FulfillOn=”Permit”>
    <AttributeAssignment
AttributeId=”3rd_pty_disclosure” DataType=
    ”http://www.w3.org/2001/XMLSchema#String”>
    NO
    </AttributeAssignment>
```

In practice the condition part will in general be more complex than this, growing as the user adds to it or as automatic learning modifies it. To make things easier for the user, the notion of a "situation" has been introduced, which the user can associate with a specific set of context values.

When the PPN preferences for the whole set of data requested by the service are evaluated and the outcomes are combined, the result is a privacy policy set that specifies under which circumstances access to user data should be granted and forms the basis for negotiation with the service. This negotiation should result in an agreement that meets all the requirements in the privacy policy set. Thus the outcome of the negotiation is a privacy policy that specifies under which circumstances access to user data should be granted.

The user VID selection preferences have the same basic format except that the action part specifies a VID to be used. Thus conditions can include context conditions such as the location of the user, the current time, his/her activity and any other context attribute

that exists in the context management system. The outcome specifies a specific VID to be used. For example, the user can have different VIDs for a VoIP application depending on his/her activity, location and current time:

```
IF (location='work' OR time.between(0900,1700))
AND dayOfWeek='weekday'
THEN VID='workVID'
ELSE VID='defaultVID'
```

There will be cases where no VID will match the user's VID selection preferences and in these cases, the user should be queried using a Graphical User Interface to select a VID from his/her pool of VIDs or be offered the option to create a new VID that will match in this case. If the latter is what the user wishes to do, the new VID will reference a list of user data and a VID selection preference will be set up for this VID to be used in the specific context in which it was created.

6. Some Research Issues

Some research challenges associated with this approach include the following. The first challenge concerns the way VIDs are handled to ensure VID isolation. A fundamental assumption here is that no service should have access to more information on a user's VIDs than is absolutely necessary for its functioning. In particular, no service should be able to associate independent VIDs belonging to the same user, and hence infer or gain access to personal information on the user to which it is not entitled. This applies to all services outside the Security Manager module and to some extent even within it. This has consequences for the design of the user preference subsystem.

Another major issue is how to engage the user in the decision making. If it is completely automatic, it will be difficult for the user to change when the need arises; if it is completely manual, it will be too arduous for the user. A compromise is to take the decision for the user and inform him/her of the VID selected, giving him/her the opportunity to intervene and change the VID selected. If the user does so, he/she may change to an existing VID or create a new one. The design of suitable GUIs with flexible representations of VIDs without releasing more information than necessary and enabling VID linkage is another serious challenge.

By monitoring the user's actions in accepting or changing VIDs and applying machine learning techniques to this information, the set of user preferences can be built up and maintained automatically. However, the problem of VID isolation affects the machine learning and could result in a laborious process - although this has been overcome in our system. Nevertheless one is still faced with the problem of distinguishing between short-term and long-term changes in preferences.

At a different level one has issues relating to the storage and protection of the preferences. While user preferences in general represent sensitive data which needs to be protected from unauthorised access, user preferences for privacy are even more crucial because of the way in which they are accessed and used and because they control the degree of access a service is permitted to the user's personal data (including other preferences). Although their format is essentially the same, the action performed is highly confidential since they affect the selection of VIDs. Thus this set of user preferences needs to be treated differently from the rest of the user profile.

One simple way of handling this would be to create a special-purpose preference management subsystem together with a learning component, which is a subset of the normal preference management subsystem, and which is contained completely within the Security and Privacy subsystem. This would ensure privacy although at the expense of a considerable amount of duplicated code.

An alternative solution would be for the Security and Privacy subsystem to utilize the normal preference management and learning facilities of the pervasive environment even though these are not trusted. It can do so by using cryptographic techniques. By encrypting actions relating to the selection of VIDs before passing information to the preference management subsystem, and decrypting the information returned, the privacy of the user can be protected. The preference management and learning subsystem can handle the preferences as it does for any other service without understanding the actions. This solution avoids the expense of the additional code.

7. Conclusion

In developing pervasive computing technologies that are acceptable to the end user, it is essential to take account of user needs and preferences to personalize decision making within such a system. One important area where they may be used to improve the user-friendliness of pervasive systems is in the identity

management approach to dealing with user privacy. User preferences can be used both in determining what information can be released about the user and in the process of selecting a virtual identity to hide the real identity of the user. These are easier for the user to understand and manipulate, especially if the formats of such user preferences are consistent with those of other user preferences in the system.

One of the aims of the Daidalos project was to develop a pervasive system which uses a system of virtual identities (VIDs) to hide the real identity of the user and thereby provide privacy protection through pseudonymity. At the same time a lower-level objective was the provision of different forms of personalisation through the use of user preferences to make the system acceptable to an end-user. To this end the approach described in this paper was developed.

The Persist project is another European research project aimed at developing a pervasive system based on a radically different approach – the notion of Personal Smart Spaces. The approach described here will be used within the Persist prototype.

This paper is concerned with the use of virtual identities in providing adequate protection of privacy in the context of pervasive systems. It extends and elaborates on the ideas presented in [1].

For services to be context aware, personalized or simply "pervasive", such a system must maintain large amounts of personal data and disclose these when required. This practice poses enormous threats to the privacy of individuals if not handled with the utmost care and protection.

The paper goes on to describe a solution that has been investigated to address these challenges in the context of the Daidalos pervasive system and which will be used in the implementation of the pervasive system prototype in the Persist project.

Acknowledgment

This work was supported in part by the European Union as part of the Daidalos project under the Sixth Framework Programme and the PERSIST project under Framework 7. The authors wish to thank all colleagues in the Daidalos project developing the pervasive system, and especially those who have developed the concepts and components for handling privacy. However, it should be noted that this paper expresses the authors' personal views, which are not necessarily those of the Daidalos consortium. They also gratefully acknowledge the support of the European Union for the Daidalos project but note that apart from funding the Daidalos project, the European Commission has no responsibility for the content of this paper.

References

[1] E. Papadopoulou, S. McBurney, N. Taylor, M.H. Williams, K. Dolinar and M. Neubauer, "Using User Preferences to Enhance Privacy in Pervasive Systems", Third Int. Conf. on Systems (ICONS 2008), Mexico, 2008, pp. 271-276.

[2] M. Weiser, "The computer for the 21st century", *Scientific American*, vol. 265(3), pp. 94-104, 1991.

[3] M. Satyanarayanan, "Pervasive computing: vision and challenges", *IEEE PCM*, vol. 8(4), pp. 10 - 17, 2001.

[4] The UK Grand Challenges Exercise. Available: http://www.ukcrc.org.uk/grand_challenges/ 28.05.2009

[5] V. Lesser, M. Atighetchi, B. Benyo, B. Horling, A. Raja, R. Vincent, T. Wagner, P. Xuan and S. Zhang, "XQ.: The Intelligent Home Testbed", in *Proc. Anatomy Control Software Workshop (Autonomous Agent Workshop)*, 1999, pp. 291-298.

[6] S. Yoshihama, P. Chou and D. Wong, "Managing Behaviour of Intelligent Environments", in *Proc. First IEEE Int. Conf. on Pervasive Computing and Communications (PerCom '03)*, 2003, pp. 330-337.

[7] M. C. Mozer, "Lessons from an Adaptive House", in D. Cook & R. Das (Eds.), *Smart Environments: Technologies, protocols and applications*, 2004, pp. 273-294.

[8] B. D. Ziebart, D. Roth, R. H. Campbell and A. K. Dey, "Learning Automation Policies for Pervasive Computing Environments", in *Proc. 2nd Int. Conf. on Autonomic Computing (ICAC '05)*, 2005, pp. 193-203.

[9] M. G. Youngblood, L. B. Holder and D. J. Cook, "Managing Adaptive Versatile Environments", in *Proc. 3rd IEEE Int. Conf. on Pervasive Computing and Communications (PerCom '05)*, 2005, pp. 351-360.

[10] H. K. Y. Si, "A Stochastic Approach for Creating Context-Aware Services on Context Histories in Smart Home", in *Proc. ECHISE2005, Pervasive '05*, 2005, pp. 37-41.

[11] J. Groppe and W. Mueller, "Profile Management Technology for Smart Customizations in Private Home Applications", in Proc. 16th Int. Workshop on Database and Expert Systems Applications (DEXA '05), 2005, pp. 226-230.

[12] C. Cordier, F. Carrez, H. Van Kranenburg, C. Licciardi, J. Van der Meer, A. Spedalieri, J. P. Le Rouzic, and J. Zoric, "Addressing the Challenges of Beyond 3G Service Delivery: the SPICE Service Platform", in *Proc. Workshop on Applications and Services in Wireless Networks (ASWN '06)*, (2006).

[13] M. Strutterer, O. Coutand, O. Droegehorn, and K. David, "Managing and Delivering Context-Dependent User Preferences in Ubiquitous Computing Environments", in *Proc. Int. Symp. on Applications and the Internet Workshops (SAINTW '07)*, 2007.

[14] T. Z. Zarsky, "Thinking Outside the Box: Considering Transparency, Anonymity and Pseudonymity as Overall Solutions to the Troubles of Information Privacy", *Miami Law Review*, 58(4) 2004, p. 1301

[15] C. Kalloniats, E. Kavakli, and S. Gritzalis., "Dealing with Privacy Issues during the System Design Process", in *Proc. 5th IEEE Int. Symposium on Signal Processing and Information Technology*, December 2005, Athens, Greece.

[16] A. Pfitzmann and M. Köhntopp, "Anonymity, unobservability, and pseudonymity", in H. Federrath, Editor, *Designing Privacy Enhancing Technologies* , 2001, pp. 1–9.

[17] A. Brar and J. Kay, "Privacy and Security in Ubiquitous Personalized Applications", in *Proc. User Modelling Workshop on Privacy-Enhanced Personalization*, Edinburgh, UK, July 2005.

[18] M. Langheinrich, "A privacy awareness system for ubiquitous computing environments", in Proc. 4th Int. Conf. on Ubiquitous Computing, London, UK, 2002, pp. 237--245.

[19] J.I. Hong and J.A. Landay, "An Architecture for Privacy-Sensitive Ubiquitous Computing", in *Proc. 2nd Int. Conference on Mobile Systems, Applications, and Services (MobiSYS)*, Boston, Massachusetts, USA, 2004.

[20] A. Kobsa and J. Schreck, "Privacy through pseudonymity in user-adaptive systems", *ACM Trans. Internet Techn*., Vol. 3(2), 2003, pp. 149-183.

[21] J.R. Rao and P. Rohatgi, "Can Pseudonyms Really Guarantee Privacy?", in Proc. 9th USENIX Security Symposium, Denver, Colorado, Aug. 2000.

[22] M. H. Williams, N. K. Taylor, I. Roussaki, P. Robertson, B. Farshchian and K. Doolin, "Developing a Pervasive System for a Mobile Environment", in *eChallenges 2006 – Exploiting the Knowledge Economy*, IOS Press, 2006, pp. 1695 – 1702.

[23] E. Papadopoulou, S. McBurney, N. Taylor, M. H. Williams and G. Lo Bello, "Adapting Stereotypes to Handle Dynamic User Profiles in a Pervasive System", in Proc. 4th Int. Conf. on Advances in Comp. Sc. and Tech. (ACST '08), 2008, pp. 7-12.

[24] M.H. Williams, I. Roussaki, M. Strimpakou, Y. Yang, L. MacKinnon, R. Dewar, N. Milyaev, C. Pils and M. Anagnostou, "Context Awareness and Personalisation in the Daidalos Pervasive Environment", in *Proc. Int. Conference on Pervasive Systems (ICPS 05)*, Santorini, July 2005, pp. 98 – 107.

[25] T. Kindberg, et al, "People, Places, Things: Web Presence for the Real World", in *Proc. Third IEEE Workshop on Mobile Computing Systems and Applications*, 2000, 19-28.

[26] C. Prehofer, J. van Gurp and C. di Flora, "Towards the Web as a Platform for Ubiquitous Applications in Smart Spaces".

[27] J. Girao, A. Sarma and R. Aguiar, "Virtual identities - a cross layer approach to identity and identity management", in *Proc. 17th Wireless World Research Forum*, Heidelberg, Germany, November 2006.

[28] T. Yu, M. Winslett and K.E. Seamons, "Supporting structured credentials and sensitive policies through interoperable strategies for automated trust negotiation", *ACM Trans. Inf. Syst. Secur*. Vol. 6(1), pp. 1-42, 2003.

[29] The Platform for Privacy Preferences 1.1 (P3P 1.1) specification. Available at: http://www.w3.org/TR/P3P11/ 28.05.2009.

[30] OASIS XACML homepage. Available at: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml 28.05.2009

[31] J.I. Hong and J.A. Landay, "An Architecture for Privacy-Sensitive Ubiquitous Computing", in *Proc. 2nd Int. Conference on Mobile Systems, Applications, and Services (MobiSYS)*, Boston, Massachusetts, USA, 2004.

Software Vulnerability vs. Critical Infrastructure - a Case Study of Antivirus Software

Juhani Eronen*, Kati Karjalainen, Rauli Puuperä
Erno Kuusela, Kimmo Halunen, Marko Laakso, Juha Röning
Oulu University Secure Programming Group
Department of Electrical and Information Engineering
P.O. Box 4500
90014 University of Oulu
Email: ouspg@ee.oulu.fi
*Finnish Communications Regulatory Authority FICORA
P.O. Box 313
00181 Helsinki
Email: juhani.eronen@ficora.fi

Abstract

During the last decade, the realisation of how vulnerable critical infrastructures are due to their interdependencies has hit home with more gravity than ever. The abundance of vulnerabilities in the software that is widely used in critical systems could have escalating consequences. In this paper, we used the PROTOS MATINE model to systematically examine the scope of software systems used in critical infrastructure. Dependency analysis methods indicated antivirus software as a critical subject to study, as its use is mandated and as it processes data from malicious sources. We determined that antivirus software is by nature susceptible to various risks and has exhibited significant vulnerability, but the issue is neither widely recognized nor reported. Awareness on the drawbacks of AV software should be spread among the planners of the critical infrastructures. Due to inherent risks, the suitability of antivirus software in critical systems should be reconsidered on a system-by-system basis.

Keywords: *Vulnerabilities, critical infrastructure, dependency analysis, antivirus software*

1 Introduction

According to NATO, critical infrastructure is defined as "those facilities, services and information systems which are so vital that their incapacity or destruction would have a debilitating impact on public and governmental security, economy, public health and safety and the effective functioning of the government" [32]. The need for critical infrastructure protection (CIP) has become paramount in recent years with the advent of new asymmetric threats, both physical and cyber.

While the physical risks have been manifested by the threats of natural disasters and terrorism, awareness of cyber risks has also been increased by cases of cascading failures in electrical networks and cases of premeditated damage by disgruntled employees. Warning signs have been raised by authorities on attacks against the supervisory control and data acquisition (SCADA) systems controlling critical systems [41]. However, there is much more to the cyber risks than SCADA.

Computerisation and ubiquitous network connectivity have been leading trends in the services of society during the last few decades. Systems comprising the critical infrastructure are no exception. Previously, control systems of critical services have been custom-designed software and hardware systems situated in dedicated networks. For reasons of synergy, efficiency and increased functionality, commercial off the shelf (COTS) hardware, networks and operating systems have frequently superseded the conventional wisdom in building critical systems. For similar reasons, control systems are increasingly interconnected with production and office networks and even the Internet.

Besides from offering a number of self-evident benefits, the transition of control systems into the realm of traditional computing predisposes critical systems to a number of risks, e.g. the growth of system complexity increases its failure

modes, and interconnections constitute vectors for attacking the system. The major risks that have been identified in CIP research are generally the loss of a major asset, or a cascading failure of multiple assets due to their interdependencies. These interdependencies are generally classified as physical, geographical, logical or cyber [35]. The cyber interdependencies are among the most complicated and varied of these interdependencies [4].

In previous research, we have created the PROTOS MATINE model [19] for deciphering technical dependencies of the critical infrastructure. The model includes reviews of specifications and other technical facts, as well as expert interviews to capture the tacit knowledge regarding the deployment and use of systems. Media and market share analyses enable prioritising of further study. Visualisation is used to present the results of the study in a quickly understandable and concise manner.

In this paper, we employ the PROTOS MATINE model to extend upon a study on a type of cyber dependencies reported in our earlier paper [2]. We analyse the cyber dependencies on multiple levels, including software vulnerability and interdependency due to factors such as data propagation and shared protocols, code, and libraries. In recent years, the use of antivirus (AV) software, whose goal is to keep malicious programs (malware) at bay, has become a widely adopted procedure among critical infrastructure systems. We selected AV software as our target of study, and explore some of their dependencies, as well as their vulnerability history. AV software vulnerabilities are not in general reported by the media, even though the number of AV vulnerabilities has expanded rapidly in recent years [40]. Although the overall vulnerability numbers seem to have decreased, the future progression of AV vulnerabilities is unpredictable.

As a part of our previous research, we scrutinised the vulnerability of AV software with a robustness test set [34]. We used a responsible vulnerability co-ordination process to ensure that any vulnerabilities found would be fixed by the vendors whose products they affect. The results of our robustness testing indicate that there is still much work to be done to counter the mounting complexity of current software. While bugs were found and eliminated, new ones continue to emerge at a constant rate. In this paper, we follow up the results of the vulnerability co-ordination process, and investigate whether affected AV vendors had actually fixed the vulnerabilities reported to them.

The current status of the AV use of software is a complex phenomenon. AV software does not automatically increase security, but may be a source of unnecessary risk, especially for information infrastructure. In addition to added complexity, dependency and vulnerability, there are issues related to vulnerability disclosure and reliability of AV software.

Recent studies on the efficiency of AV software raise concern about their effectiveness in the current threat landscape. In a test by the security company Team Cymru, only 37% of 1,066 pieces of current malware were detected by a sample of 32 antivirus software [17]. Another recent study by the AV vendor Panda Security observed the infection rate of unprotected systems at 33%, and that of protected systems at 23% [30]. The observed low detection rates combined with a mere ten percent point reduction in infection risk might not warrant for the usage of software with inherent risks.

Despite the stated problems, AV software is at present commonly considered as a basic element of safe computer use. For example, FICORA (Finnish Communications Regulatory Authority) recommends that AV software should be installed on computer systems in order to protect them from malware. HIPAA [25] and Sarbanes-Oxley Act, (SOX) [39] have extended these security requirements to laws. The same conception of security produced by AV software is popularized by security policies, user education and media. There is a considerable lack of controversial opinions in all of these areas.

The current security paradigm is the main reason for problems in the context of AV software use. Although AV software may be a necessity to fight off specific malware threats, its de facto and de jure use should be reconsidered in critical infrastructure systems. In many cases, the use of AV software may expose the system to unnecessary vulnerabilities and cause needless dependencies.

The next section presents background on the context of antivirus vulnerabilities in critical infrastructures, as well as an explanation of the PROTOS MATINE model and supplementary models for dependency analysis. The third section presents the results gathered by the dependency analysis and the ensued robustness tests. The paper concludes with observations on the results and on areas of future improvement.

2 Background

In this section, we present a definition for vulnerability, and list the unique aspects of AV software with respect to vulnerabilities. Next, we define our concept on different levels of technical dependency, and how it can be used to augment dependency analysis in traditional CIP perspective. This is followed by an explanation of the PROTOS MATINE model, as well as a short review of complementary dependency analysis methods.

2.1 Vulnerabilities and Antivirus Software

All software contains bugs due to various factors, such as inherent difficulty in translating the requirements to code, complexity of the requirements or the underlying system, immature programming practices and methods [21, 6]. An old maxim of the quality control industry states that the number of flaws in a system is generally proportionate to the complexity of the system. This can be restated as "the number of flaws in a system is roughly proportionate to the extent of its functionality". All modern COTS software systems are very complex, which raises the number of their bugs to towering amounts. Bugs with security implications are called vulnerabilities.

Although AV software is commonly thought to increase security, it is produced by the same programming processes, which can result in insecure programs in general. As a rule, all software is breakable [6]. By definition, AV software must process potentially malicious input in a wide variety of data formats. As parsing different protocols and formats have historically been proven error-prone, AV software is particularly susceptible to programming errors.

Current malware employs a variety of methods to thwart detection, such as packing, polymorphism, obfuscation, anti-analysis and anti-unpacking. Some malicious code has even been reported to exploit vulnerabilities in analysis software. The defence methods force AV systems to employ emulation, deobfuscation, unpacking, and other functions in order to successfully detect malware. As these operations are very sophisticated and handle potentially malicious input, the error-sensitivity of AV software is further highlighted.

Many AV software share the same integral scanning engine or engines [8, 46]. The scanning engine, responsible for identifying malicious files using signature databases, is the main component of an AV software. Homogeneity facilitates the design process of malware, as it is relatively quick to test the malware in development with all of the most common AV software [44]. This may be reflected in the recently observed low detection rates of current malware [17, 30].

AV software population is quite homogeneous, which in itself is a warning sign: it enables the spread of malware [5] that targets against dominant AV products. The market is dominated by a few leading vendors and using more than one AV program at a time is usually impossible [23], which may be fortunate since each vulnerable AV program would add to the attack surface by exposing more code to exploitation attempts. AV software requires high access rights in order to monitor the system, which makes them attractive attack vectors for systems compromise.

2.2 Dependencies and Critical Infrastructure in the Antivirus Vulnerability Context

During the last decade, the importance of critical infrastructure has been realised more acutely than ever. Dependencies between different infrastructures have been recognised as a major cause for escalating consequences for errors in point components. In critical infrastructure, dependencies can be identified on multiple levels, including technology, functions, people, processes and location. Failures of infrastructure components have been identified to induce immediate or delayed problems or failures in dependent components, which may in turn lead to cascading failures. Electricity and energy in general are prime examples of this behaviour, as practically all other infrastructure elements depend on these. Thus, different dependency tracking models have increasingly been taken into use in the context of critical infrastructure. A good rundown of these models is the CRN International CIIP Handbook, which presents national policy approaches to critical information infrastructure protection and the methods and models used to assess the vulnerability and security of these structures [26].

In this paper, a dependency is defined as a linkage between entities or common metadata among them. Dependencies are discovered by forming descriptive metadata and links from given information and then analysing common features and differences of this semantic data. As an example in the critical infrastructure context, the dependency of a communications network on electricity could be portrayed as a link between a power plant and a cell phone tower, whereas their location in the same building could be described with location metadata containing GPS coordinates. The concepts of links and metadata can be considered equivalent to RDF [48] triplets with nodes and literals, as defined by the W3C semantic networks initiative. Similarly, the dependency graphs essentially form a semantic network. However, many of the concepts of semantic networks, such as data types and strict ontologies, have not been identified as beneficial for the rapid analysis of dependencies in previous research [21]. Thus, the approach to semantic data used in the scope of this paper is that of lightweight tagging and folksonomies rather than the stricter and more formal semantic approach.

The case presented in this paper is that the dependency of critical infrastructure components on the robustness of AV software may induce risks to those components, which may lead to wider infrastructure-level risks through cascading failures. Measuring the threat that software constitutes is quite impossible without understanding it in a detailed level. Identifying the technical dependency of software may serve as a decent first aid for this purpose, while bestowing multiple benefits such as increased understanding of the ac-

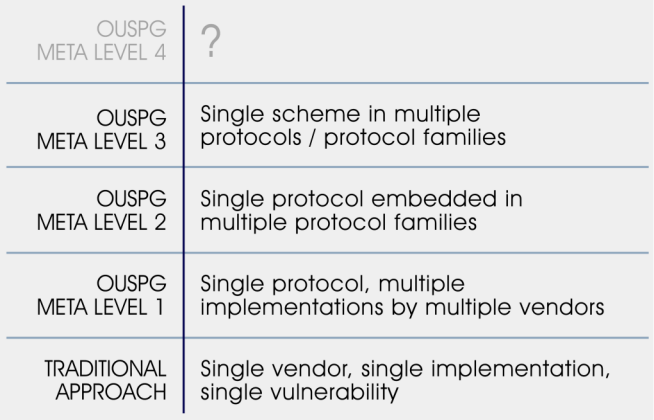


Figure 1. OUSPG metalevels

tual reason for and the scope of different kinds of failures.

Technical dependencies span multiple levels of abstraction, and thus, need to be examined in an iterative fashion. First, the boundaries of different software systems and their interfaces are enumerated, contributing to the basic understanding of the composition of the system. Communication via interfaces is performed by protocols, and thus, the used communication protocols need to be identified. Analysis on the data flows of these protocols sheds light on the propagation of data among systems, and possible attack vectors. Once the critical avenues of attack are attained, the analysis can be prioritised on the code that handles them. As most current systems are modular, this analysis can be broken down further in examinations of libraries and software subsystems that handle distinct inputs, use cases, and so forth. The data gathered by this method can be used for discerning the impact of vulnerabilities in system components of different granularity.

The concept of meta levels (see Figure 1 on page 4 is applicable to any context with inherent dependencies. Meta level is an attribute of a vulnerability, which describes its level of abstraction as well as its scope. Information on the structure of different systems and their relations highlights elements, which are highly connected or common between multiple systems. Vulnerabilities in these elements are typically of a higher meta level, as they can result in epidemic failures due to their wide implementation base, or cascading effects due to the failure of a high number of dependent elements [19].

Meta level zero describes the case where a vulnerability only affects a single implementation (a software version). Meta level one vulnerabilities affect a whole class of systems (all software that implements a certain interface). Meta level two vulnerabilities affect a super-system consisting of multiple classes of systems (all software having any interface that includes a certain subsystem). Meta level three affects an element that is used for widely disparate purposes, perhaps by a great number of systems (all systems that use a certain notation, encoding, or other function) [19].

In our previous paper [2], we have given some preliminary results of our research and a brief explanation of the methods used in this research. Our research was focused on the file formats that different AV software handles, as they form a common public interface. Side-by-side comparison of the exposure of AV software to file format vulnerabilities is not straightforward, as the support for different formats varies considerably among AV software.

Uncovering dependencies in the handling of archive file formats may be difficult due to a number of implementation details. A file format implemented in two software products can cause similar but unrelated problems in them, which could constitute a dependency false positive. Files of some archive formats may embody files in other archive formats, which may lead to the use of different algorithms in different parsing implementations of these files. Analysis of cases such as these is difficult, as specifications or source codes for commercial AV software are not available.

2.3 The PROTOS MATINE Model

The research method is based on an earlier OUSPG (Oulu University Secure Programming Group) project, PROTOS MATINE. The project focused on the interdependencies of network protocols and produced the PROTOS MATINE model [19] (see Figure 2) and the semantic tool Graphingwiki [20], which are now put into use in the context of AV vulnerabilities. The model presents an iterative method for rapidly gaining an insight into a field of study.

The PROTOS MATINE model was originally developed to illustrate protocol dependencies in critical infrastructure from multiple angles. Understanding protocol dependencies has been seen beneficial for the assessment of the wider technical dependencies of infrastructure, and the impact vulnerabilities would have on it. The method was developed to collect protocol specific data, which is spread out in multiple sources, e.g. newspapers, mailing lists, technical documents, protocol specifications and experts, who have tacit knowledge of protocol usage. During the development of the model, we noticed that tracking only one or two sources gives a biased picture of protocol's history, usage and prevalence, and by combining several data gathering methods, the accumulated data coincides better with the real situation.

The different data sources, such as specifications, literature, media and experts, work towards a common goal - understanding a technological subject on multiple levels. These levels include contents and structure of the subject, its history as well as projected future, its use cases and areas

of usage and its environment and relations to other subjects. With this kind of knowledge, the weight of a subject can be determined in the desired context, such as a system, a network, a corporation or a sector of the critical infrastructure. As an example, various data gathering methods were used to perform analyses of the effects of vulnerabilities found in parsers of the prevalent ASN.1 notation [21, 19]. The analyses were conducted with heavy emphasis on systems used in critical infrastructures, and resulted in a number of test suites to test the robustness of different protocol implementations [19].

The results of the PROTOS MATINE model are presented by visualisations that aim to portray different aspects of the protocol. Visualisations were also used as a communication method between researchers, managers and other operatives. The first views that we created, depicted the protocol's specification history (protocol view), its technical linkage (technological view), and its usage scenarios or general usage in different sectors of society (organisational view). These views were constructed using various data sources and methods: experts (interviews), public attention (media follow-up), protocol definitions (standards, technical specifications) and the prevalence of protocol implementations (historical data, usage environments). The main views were adapted according to a specific target group or usage scenario. However, we quickly discovered that more versatile and automatically generated views were needed.

We started to develop Graphingwiki, a semantic wiki tool that enables the deepened analysis of the Wiki data, by augmenting it with semantic data in a simple, practical and easily usable manner. Graphingwiki can be used to automatically present the semantic data as tables and to visualise it as graphs. These visualisations are used to clarify the resulting body of knowledge so that only the essential information for a usage scenario is displayed [21]. The key aspects of the workflow are automated gathering of baseline data, augmenting the data by experts and manual data gathering, and generating automated visualisations.

AV software was selected as our target because such software has an extensive attack surface due to a wide variety of file formats it must handle, they are run with high privileges, and their usage is mandated in many cases. Vulnerable AV software would be tempting attack vectors for systems compromise. We wanted to visualise AV vulnerabilities and, with the help of the dependency graphs, find out if there are any linkages between file formats, AV vendors and software vulnerabilities. Preliminary results helped us to focus PROTOS Genome robustness test set on archive formats and offered a context for the vulnerability co-ordination process.

In the context of AV software, vulnerability databases represent the main data sources of the PROTOS MATINE model. Media tracking and review of the market situation were performed in the year 2006 and the following results were also represented in the previous paper [2]. Media tracking and review of the market situation lay out the priorities of later data gathering and the relative importance of different AV software. Expert interviews and publicly available specifications were only used to discern the usage of archive formats.

The semantic information on AV vulnerabilities, for example impact type and file format, was gathered from the U.S. National Vulnerability Database (NVD) [31]. NVD's descriptors of vulnerabilities are categorised and presented a in specified standard format. Additional information was gathered as a media follow-up, which was focused to national level. The media follow-up consisted of regular observation of Digitoday Finland [16], commercial news database focusing on IT sector, throughout the year 2006. News considering AV issues was classified and analysed with content analysis. The focus of media follow-up was on how the AV software and vendors are presented in the media.

2.4 Previous Work

Dependency analysis methods span disparate fields, such as graph theory, social network analysis, computing and natural language processing. Some methods of relevant fields are examined in the light of the PROTOS MATINE model, and their suitability for use in the context of antivirus software is evaluated.

In graph theory, dependencies are naturally defined by the links between nodes in the graph. The links usually do not have other attributes than their direction and possibly a numeric value signifying the strength of the link. Graph theory analyses graphs with measures such as cliques, connectedness and centrality [15]. Social network analysis is a closely related field that studies specifically graphs representing relations among people. The basic realisation behind social networks is that weak ties in social networks are more significant that stronger ones. Many sophisticated analysis methods have been developed in this field [37]. However, efficient use of these approaches requires research on which analysis methods and aspects of graphs are the most relevant in the desired context.

Conceptual graphs extend the basic graph model by introducing attributes to the dependencies, which are defined as links between dependent and antecedent [13]. The conceptual graphs model attempts to form a generalised ontology of dependencies, i.e. the set of attributes that apply to every dependency regardless of context: sensitivity, stability, need, importance, strength and impact. The model is very versatile, but its rigorous definition of dependency may represent a hindrance rather than an aid in the context of rapid knowledge discovery. It is also noteworthy that the conceptual graphs model considers only the attributes

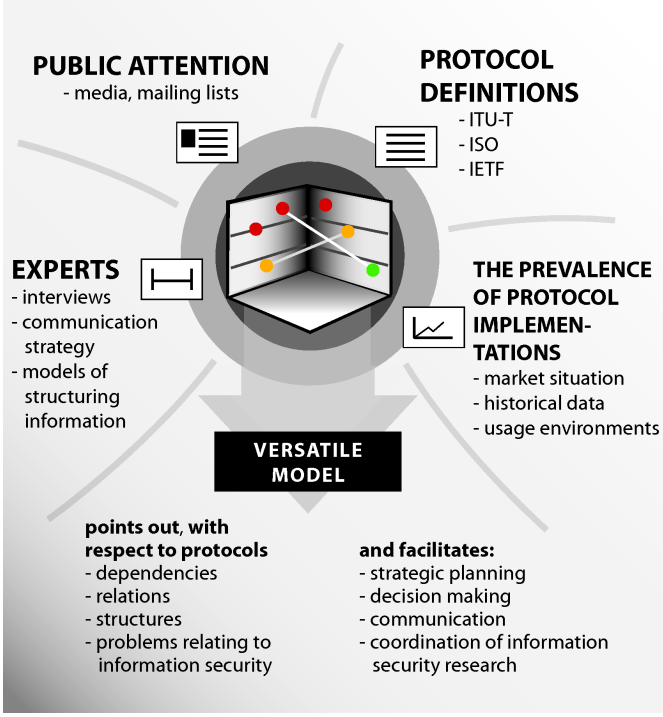


Figure 2. PROTOS MATINE model

of nodes as a source of dependencies, which may limit its usefulness.

There has been a growing interest towards dependency analysis in recent years in critical infrastructure management. The papers on the subject range from studies on the effect of a single event on one part of critical infrastructure [27] to critiques of policies adopted by a whole nation state [24]. We will describe some of the analysis methods that we consider relevant for the purposes of this paper. A more thorough review on the state of the art in critical infrastructure protection can be found in [4].

The Critical Infrastructure Modelling System (CIMS) is a system and a method for modelling dependencies in critical infrastructures, and simulating related events regarding its components. The dependency types used as the systems include physical, informational, geospatial, procedural and societal dependencies. Critical infrastructure systems are modelled by graphs, where dependencies are manifested through linkage or proximity of the nodes [18]. The CIMS software visualises graphs as 3D visualisations that can be layered on, e.g. satellite images or maps. Software-aided support for what-if scenario building is mentioned as a subject for further research.

The use of intelligent software agents to integrate, model and simulate infrastructure components has been suggested in a paper by Tolone *et al.* [45]. The system proposed in this paper is in many ways quite similar to CIMS, and it also includes 3D visualisation and simulations on critical infrastructure failures. The simulations include what-if, goal-driven, probabilistic, and discovery based analyses based on events, i.e. agent state changes. The paper does not define the dependency types used in the simulation, however, simply stating that dependencies vary based on the context of the system.

Both of the systems described in the previous paragraphs, as many other systems used to model dependencies of the critical infrastructure, for that matter, are largely constrained into the physical setting and thus unusable in the AV context. For example, the use of 3D models of may work very well in the physical context, but are unnecessary or even inapplicable to many other contexts. However, many of the ideas used in the models, such as automated scenario building and node proximity as dependency, could provide benefits to use cases such as AV. Similarly, studies that include a temporal dimension to dependencies and fault propagation could be useful, e.g. in modelling attacks to vulnerable systems [36]. We have not yet observed the need for context-varied dependencies in our research, and thus, only find the agent-based approach interesting in an academic perspective.

Graph theoretical methods used in the context of critical infrastructures mostly focus on the availability, reachability or quality of service aspects of graph portraying the infrastructure. Analysis of graph properties such as topol-

ogy and node proximity can be used to aid fault simulation [38]. Fuzzy numbers can be employed to represent incomplete information about the resiliency and other properties of the nodes [14]. Complexity analysis of the graph, in its turns, can provide insight into how efficiently the graph can be traversed in the event of failures [49]. Finally, social network analysis has proven to find critical nodes in the infrastructure graph with simple centrality, degree and variance calculations [7].

All of the above methods have the same goal, namely, to present the dependencies in a highly visual and thus more human readable form than mere documents and spreadsheets. With the PROTOS MATINE method, we aim at this very same goal, and to this end, we have used the Graphingwiki visualisation tool. The nature of the PROTOS MATINE method means that in order to be able to make visualisations with diverse types of information and multiple levels of abstraction, the visualisations need to be quite simple. Many of the methods do not have support for multiple levels of abstraction, whereas with Graphingwiki, we can present visualisations from minute details (such as single vulnerability and its impact) to greater schemes (dependencies between protocols or even dependencies in critical infrastructure). Graphingwiki works well in this context, because there are essentially no restrictions on the type of data that can be represented.

It should be noted that none of the other methods mentioned here have been used in the context of AV software. To our knowledge, there are no other studies on the dependencies in AV software.

3 Results

In this section, we present our review of the historic vulnerability data regarding AV software, and the results of related vulnerability co-ordination work.

3.1 Analysis of AV Vulnerability Data

This section contains the data in numbers and shares and presents the picture gathered from the media during our research. We use the SCAP [42] set of standards (including CVE, CPE, and CVSS) to measure gathered vulnerability data. The gathered data provided insight into the problematic areas of AV software, and guided the development of a test suite to exercise their robustness. The methods used in the coordinating the fixing process are described, as well as the results of the coordination.

In our previous analysis [2], AV vulnerability data was gathered manually from the U.S. NVD database [31]. In this paper, we parsed the data from NVD in XML format and uploaded all entries containing the words 'virus' or 'malware' to Graphingwiki with the help of automated scripts, which were used to ease laborious data gathering process and minimise errors and loss of data in data collection process.

As explained later in greater detail, we combined the NVD data with data from the SecurityFocus database [43]. In this case, the main function of the scripts was to convert the freeform vulnerability descriptions to structural data. The data from different sources can be seen to represent different expert opinions on the vulnerability. Our approach was not to combine these opinions in any way, though different opinions can be formed into a single dependency by considering the combination of various edges between two nodes as a dependency. Currently, the views to the data are generated automatically, but it is the analyst's task to decide on the most appropriate data points for his purposes. Algorithmic or other formal methods to form dependency views could be implemented as custom plugins. Our initial experiences indicate that gathering and comparing data from different vulnerability databases in this manner is a promising, yet largely neglected research area.

As the data gathered for this paper is more systematic, uniform and wider in scope, direct comparison to our previous analysis is not meaningful. The gathered data is represented in the following formats: CVE [10] enumerates unique vulnerabilities, CVSS [11] measures vulnerability severity, CPE [9] enumerates products affected, and finally, CWE Common Weakness Enumeration [12] provides a listing of weakness types.

The total number of vulnerabilities was 346, and included vulnerabilities in the products of practically all known antivirus vendors. The data spanned from the year 1998, although the bulk of the vulnerabilities were from the years 2004-2008, with a noteworthy peak in the year 2005. As can be seen in Figure 3), the number of AV vulnerabilities has expanded rapidly through these years. As measured by their CVSS scores, the average severity of all the gathered AV vulnerabilities is 6.56. This means that antivirus vulnerabilities are generally quite severe, as the NVD database considers vulnerabilities with a CVSS score equal or greater than 7.0 to have a high severity. Further, the severities of vulnerabilities have been in a slight rise during these years, as can be seen in Figure 4. By combining these two statistics, it is clear that after the year 2004 there has been a significant increase of vulnerabilities with high or medium severities.

Most of the vulnerabilities (276 out of 346) were exploitable remotely according to NVD. This shows that, as we speculated, the AV software has difficulties in robust handling of the data it inspects. With antivirus software, the type remotely exploitable means that data can be sent to the system e.g. via email, after which the AV system must inspect it. NVD classified most of the vulnerabilities as having a low access complexity, which means that ex-

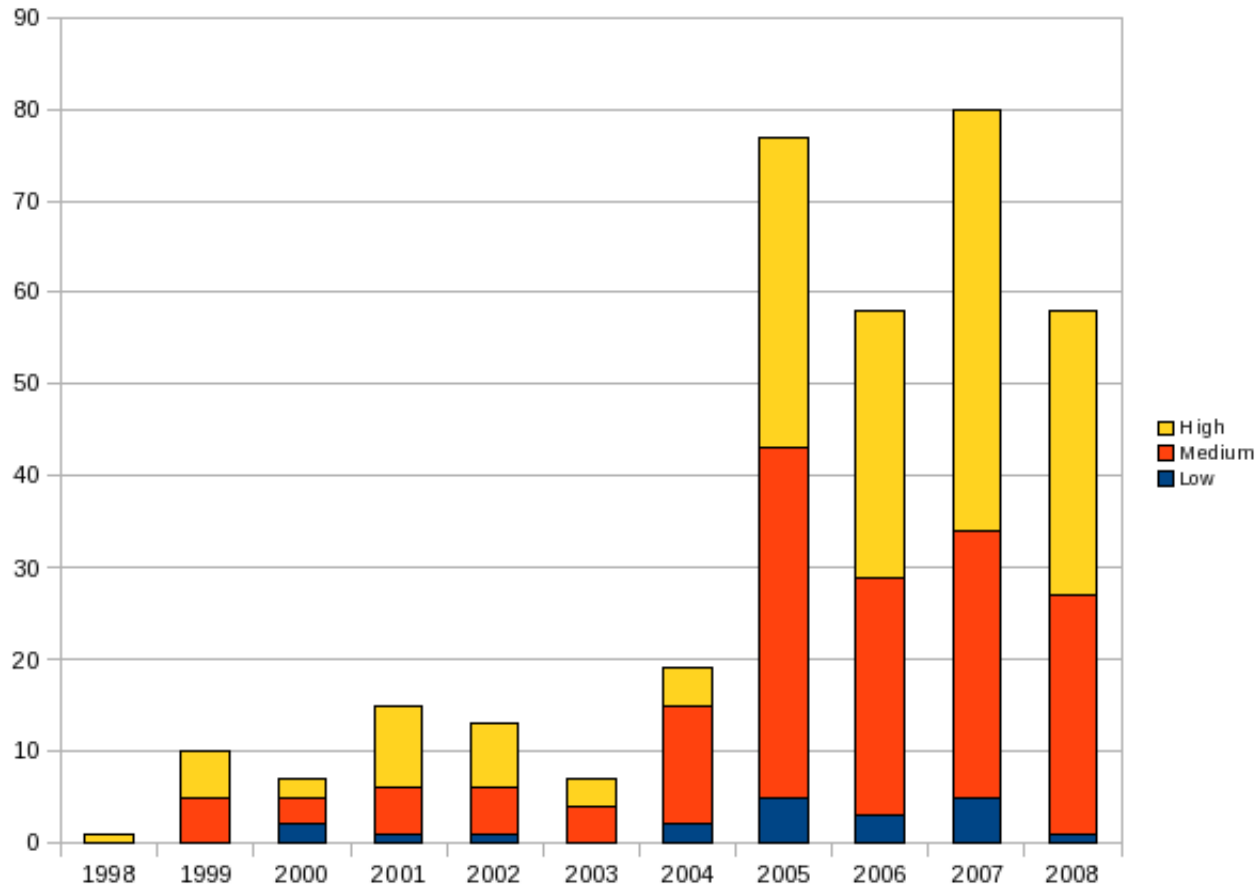


Figure 3. AV vulnerabilities of different severity per year

ploiting the vulnerabilities is not considered to be a difficult task, which further emphasises their severity.

From our data, we noted that archive formats are associated with a large portion of the vulnerabilities (see Figure 5). The most frequent archive formats were RAR, CAB and ZIP. Vulnerabilities in parsing file formats are often trivial to exploit, as well as relatively easy to discover by the means of black-box testing, i.e. fuzzing. This hypothesis can be ascertained by examining the type information of the vulnerabilities.

The NVD vulnerability database presents vulnerability types with CWE identifiers. Only about a fifth of the vulnerabilities we gathered had error type information. Therefore, we used vulnerability type information from the Securityfocus vulnerability database, which has type information about 262, or 76% of the vulnerabilities. The Securityfocus databases use an undocumented vulnerability taxonomy, which according to observations closely resembles the widely used Aslam taxonomy [3] [28].

Our previous analysis showed that the most common error type in AV software is design error. An analysis with more data indicates that boundary condition errors (60 vulnerabilities) and failure to handle exceptional conditions (46 vulnerabilities) are as prevalent as design error (59 vulnerabilities). The yearly observation depicted in Figure 6 indicates that the observed peak in the year 2005 correlates with a similar peak with the vulnerability type failure to handle exceptional conditions. Many of the vulnerabilities of this type were due to problems in parsing.

The observations prompted research in PROTOS GENOME -project, where malformed archive files were generated to test the robustness of AV software. The results of this research are reported in [34]. Our analysis suggests that the biggest factors for the peak in AV vulnerabilities in the year 2005 were related to different archive file formats, mainly RAR and ZIP. The results of PROTOS GENOME archive test set affected in turn the number of vulnerabilities in the year 2008.

The media follow-up was performed in 2006 and gained results of the analysis were also presented in a previous paper [2]. The media analysis resource consisted of 92 news items. The results can be seen in Table 7.

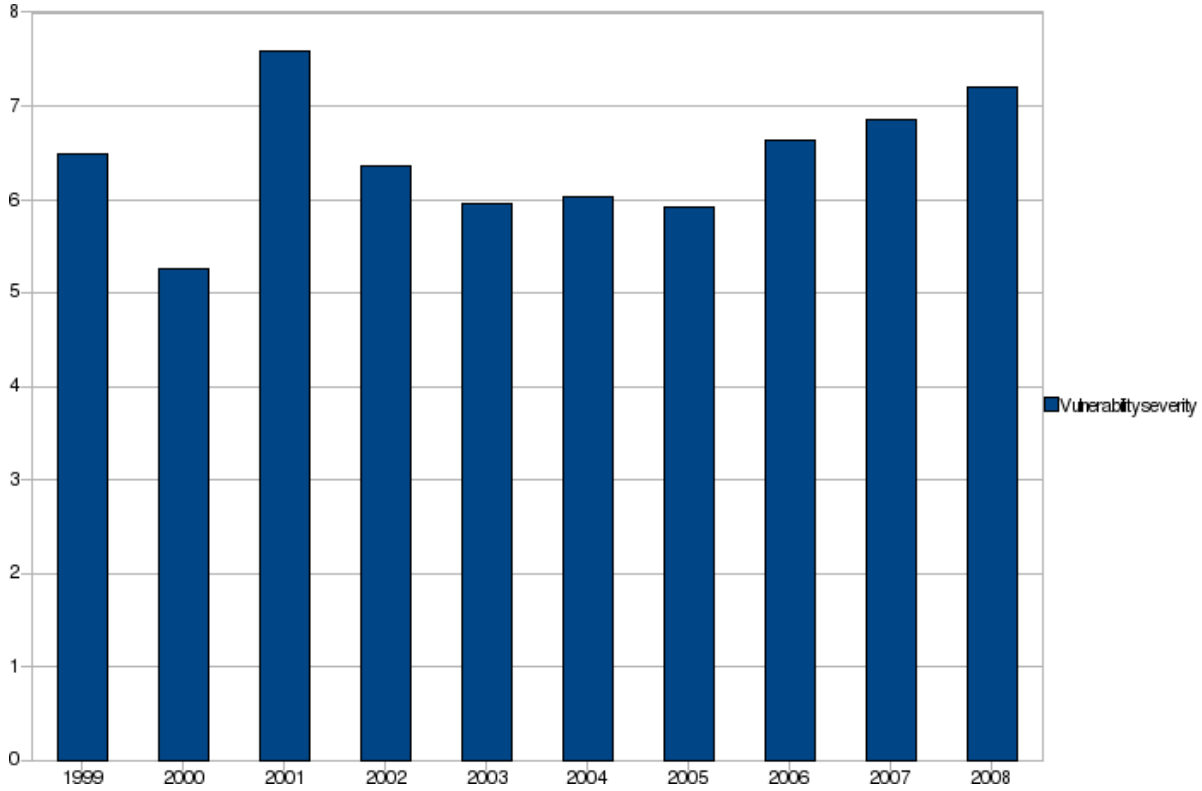


Figure 4. Average yearly severity of AV vulnerabilities

In general, AV software is presented in the news in a very positive light as continuously developing industry, which provides better solutions and increased security. The discussion of more negative issues is neglected. As our analysis of vulnerability data indicated, AV software have remarkable amount of vulnerabilities that can have wide-ranging effects. However, from the results of the media analysis, it can be noted that only 11.9% of all the collected AV-related newspaper articles dealt with AV software vulnerabilities and malfunction. We think that awareness of AV software vulnerabilities and their impact is not at the appropriate level and the usage of antivirus software should be considered more carefully in critical systems.

3.2 Using the PROTOS MATINE Model in AV Vulnerability Disclosure

Effective responsible vulnerability disclosure requires that data about vulnerabilities is presented to all affected vendors to enable them to repair the found vulnerabilities in their software. In the case of the archive format tests, the scope of potentially affected software is colossal. We used the PROTOS MATINE model to form a technical view on the usage of archive formats. The best sources of information for constructing the view in a rapid fashion were expert interviews and sources of formalised data on software. Archive formats have an extensive history, both in specification and implementation, which makes them a tedious subject of study from the literature standpoint.

The first data source we employed was the APT package management system [1], which we used to identify software that used popular archive handling libraries. We visualised this data in a technical view (Figure 8), which was augmented with the help of expert interviews. The view shed light on the scope of the potential problems - as we quickly saw, the usage of archive formats ranged from basic operating system and network functionality to applications.

The technical view was further enhanced using the NVD vulnerability database as a data source. We searched the CVE entries with the help of automated scripts for mentions of the archive formats comprising the archive test set. We filtered the gathered CVE entries by hand to remove the vulnerabilities which were not actually related to archive formats. We divided the vulnerable products gathered in this manner in groups based on their type. We gave the resulting list of products and categories to experts, who supplemented it with products of similar function.

The resulting view presented a clear direction to the vulnerability coordination process, which was performed in two phases. In the first phase, we contacted a small num-

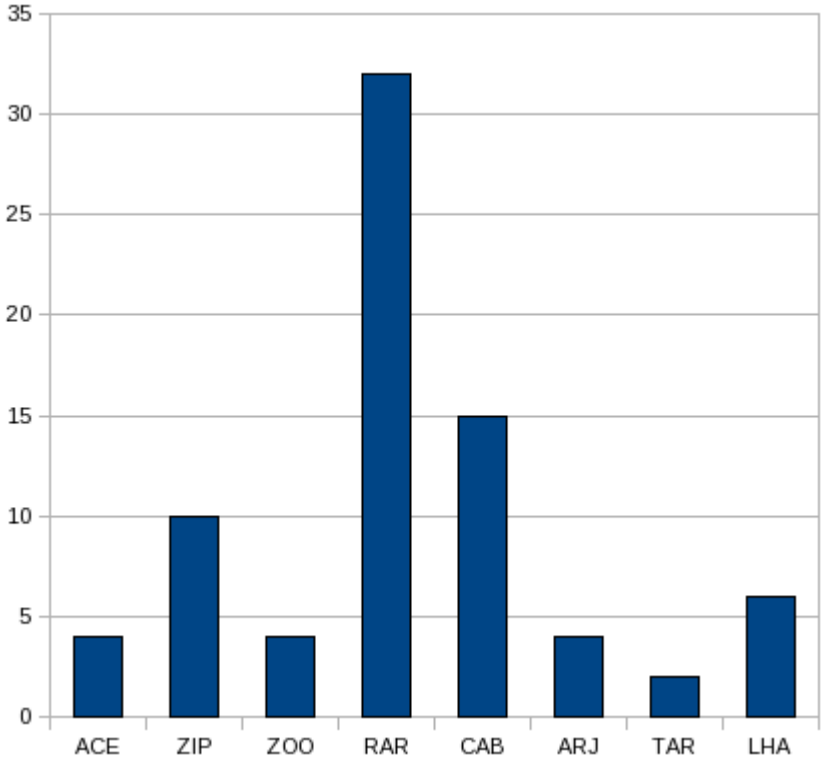


Figure 5. Archive file formats associated with AV vulnerabilities

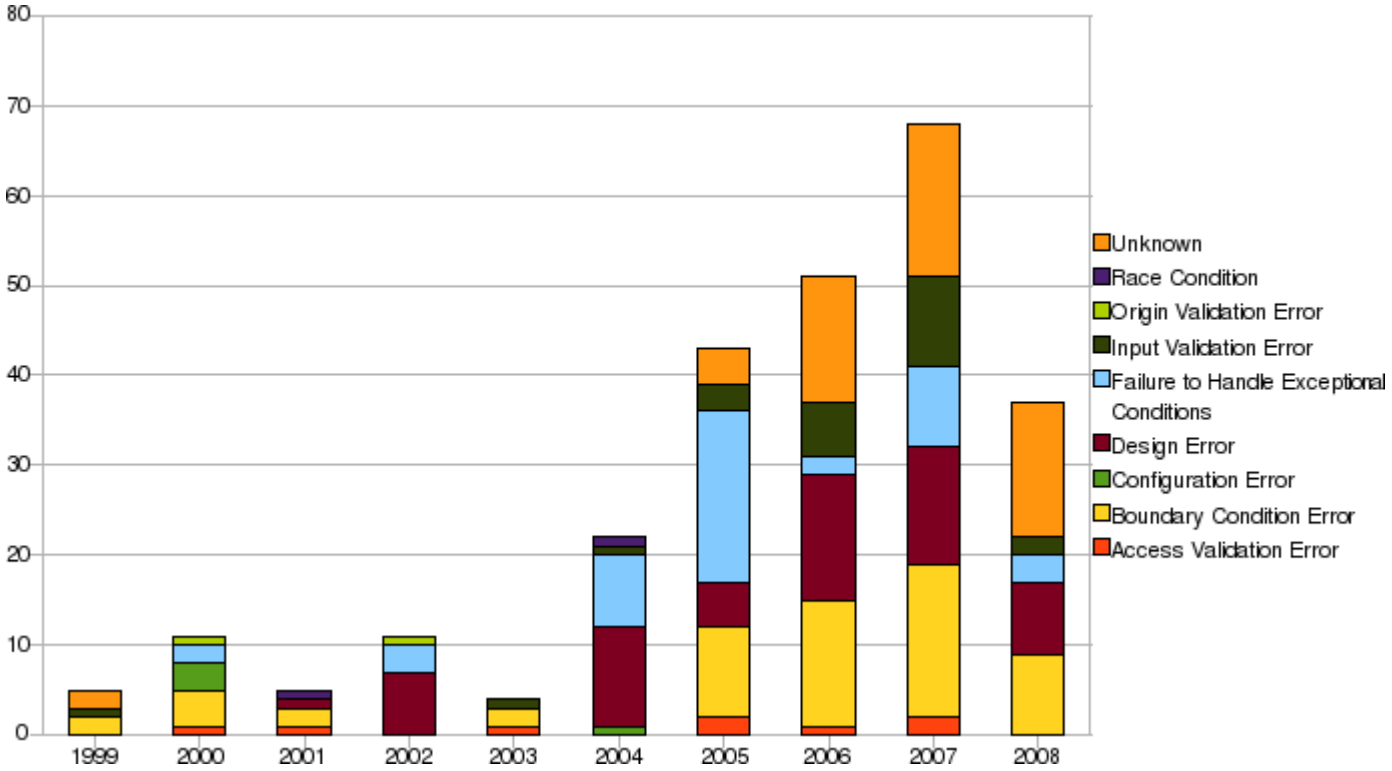


Figure 6. SecurityFocus vulnerability classification of AV vulnerabilities

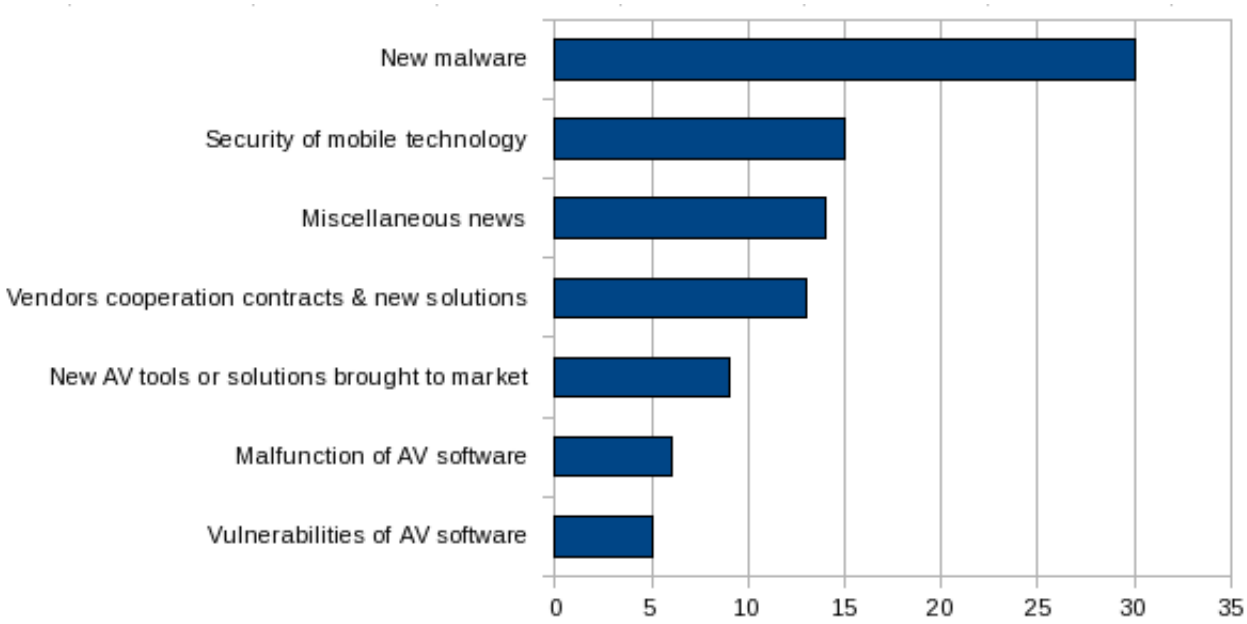


Figure 7. News topics concerning AV

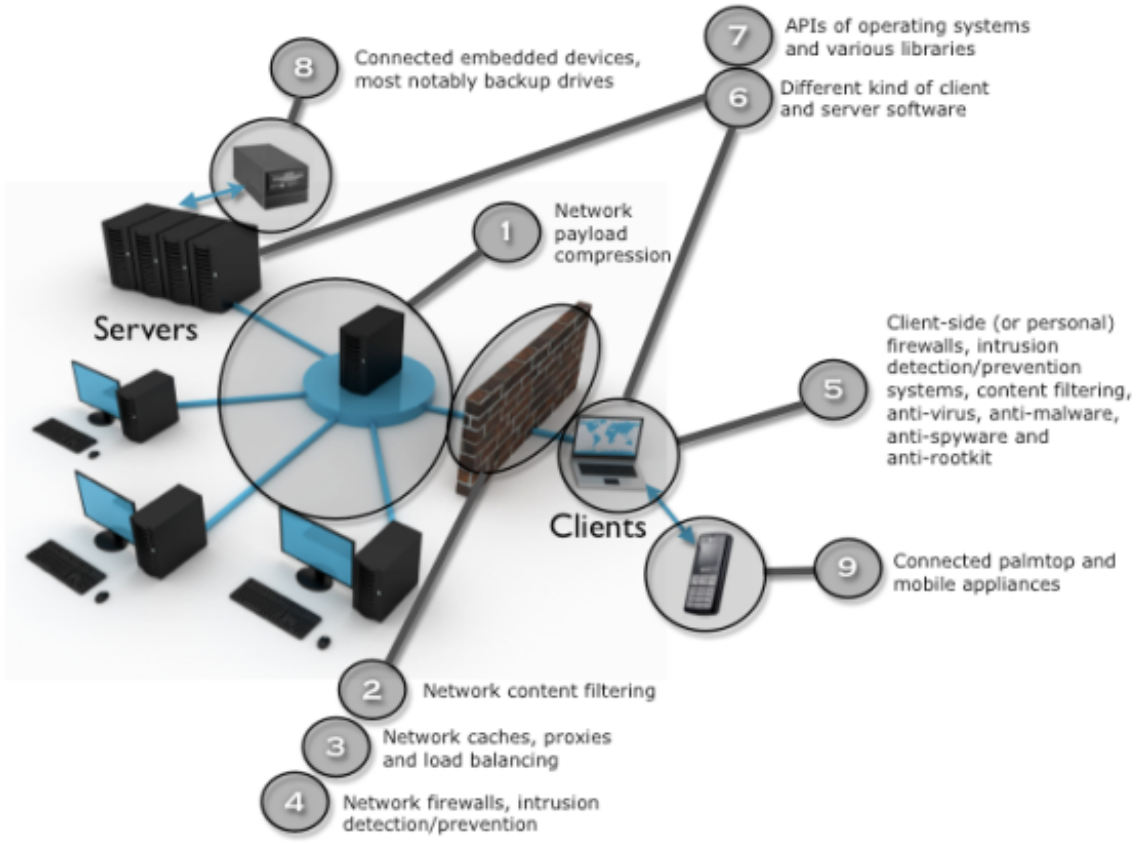


Figure 8. A simple technical view depicting the use of archive formats

ber of key vendors. The rationale for this decision was that some of these vendors have quite extensive product lines and they would require more time for testing. While the first phase was in progress, we continued to enumerate vendors and their contact addresses for the second phase. The coordination process was performed in co-operation with CERT-FI and CPNI (The UK Centre for the protection of National Infrastructure) followed the constructive disclosure process as outlined in [29]. Altogether, over a hundred vendors were contacted by CERT-FI about the test suite. Approximately 25% of the contacted vendors wanted to test their products with the test suite. The advisory stated 12 vendors vulnerable, 8 vendors not vulnerable, and the status of 30 vendors as unknown.

Published vulnerabilities represent only a small subset of the actual vulnerability of software. For various reasons, measuring the numbers of vulnerabilities is not simple [33]. Often, a bug is reported as a vulnerability only if a security conscious developer has a look at it. Bugs and vulnerabilities found internally or as a result of an audit often do not get reported. Many software vendors do not feel comfortable about sharing details about vulnerabilities in their software. It is fairly common to omit mentioning fixed vulnerabilities in change logs, or to refer to them as "reliability fixes". These factors make it difficult to measure the improvement bestowed by the archive test suite. In a more general sense, they contribute to the difficulty of making informed decisions about vulnerabilities.

Fixes were made by various vendors after the publication of the test set. This was in part due to fixes in open source products that were incorporated into commercial products and open source distributions. There is some anecdotal evidence on the fact that some vendors did not perform any testing, and that some vendors had a silent disclosure of patches to the archive set test cases.

In the test set documentation [34], we tested a sample of five antivirus software for vulnerabilities and four of them were vulnerable. We re-tested five of them for the purposes of this paper, and the same four of them were still vulnerable against the same test material.

F-Secure was the only antivirus vendor to publish updates and a security bulletin based on the archive test set at the time of publication [22], though ClamAV did publish a bulletin and an update at a later date. When the grace period before any public announcement of the danger spans months or even years, there will be vendors who issue silent fixes and move on without joining the public advisory.

4 Discussion

We set out to understand the dependency of critical infrastructure on the AV software, nature of AV software with respect to information security, security of the AV software itself both historically and presently and perception of the media and thus general public on the role of the AV software. In short, we aimed to disclose and understand any risks that such security software may pose on the critical infrastructure.

We experienced some difficulties in our data gathering efforts regarding antivirus software. Highly competitive fields do not encourage research, open standards, and open publication of data in general, and the antivirus industry is no exception. Information on the common scanning engines and possible undocumented standards used by the AV industry would provide a significant advantage to deciphering their dependencies in terms of vulnerability. As AV vendors are naturally reluctant to reveal their trade secrets, we are missing the data that would in many other cases be available via public metrics and expert interviews from the developers of products and standards. However, interviews of the actors of the critical infrastructures could provide a further insight into the criticality of applications, and hence the effects of vulnerabilities.

As the amount of public information on AV vulnerabilities leaves much to be desired, the significance of media and other public sources is emphasised. Our observation on the labour-intensiveness of the media follow-up as a data gathering method prompts attention to it as a further field of study. Still, additional sources such as social media could significantly augment the scope of public information. In this paper, we successfully used automatic data gathering methods for vulnerability databases, but employing similar methods for free-form news articles presents further challenges. The field of natural language processing has shown some promising results, methods such as support vector machines have proven useful for many tasks. It has been suggested that some dependencies could be discerned from textual structures alone. All in all, automated methods for gathering public information and refining it to more useful forms could require extensive research.

Selective aggregation of different data feeds also constitutes a promising information gathering method. As an example, package management software and the SCAP project use different nomenclature for software packages, which makes it harder to track vulnerabilities regarding Linux software packages. However, most Linux distributions provide security advisories that include SCAP compliant CVE vulnerability identifiers, which can be used to find SCAP compliant CPE software identifiers. Combining these facts from, e.g. APT popularity contest, project that attempts to map the relative popularity of Debian Linux software packages could provide insight into the vulnerability of some of the most popular Linux software. This is how using the three sources in conjunction could provide otherwise unattainable information.

The large amount of archive file format related bugs in

AV software suggests that software components used in critical infrastructure should be exposed to thorough testing – for it seems that vendors' quality assurance processes do not guarantee a sufficient level of robustness. Finding bugs in any software is not enormously difficult [47], however. As the data gathered in AV case illustrates, most of the bugs are of the same type. Whenever a bug is found, the focus is on fixing the bug, not finding its causes. There is a need for methods for understanding the bugs so that we could write programs with fewer bugs. Also, the prevalence of common vulnerability types, such as boundary condition error, in antivirus software indicates that they are largely created with programming languages that are particularly susceptible to those types of error, such as C and C++. The usage of programming languages that are inherently more secure should be examined in security-critical portions of the code, notwithstanding the possible performance penalties.

Even though the proportion of AV vulnerabilities to all vulnerabilities is diminutive, we had great difficulties in digesting the data. Although we used only one source of information on the vulnerabilities, manually trawling through the relevant vulnerabilities and analysing them was challenging. Methods for analysing the gathered information were sorely needed. Using graphs to visualise data helped in understanding the big picture, but still left plenty of room for development. For some example graphs, see Figures 9 and 10 Currently, we only analyse the graphs by visual observation. Graph theoretical and social networking analysis methods for analysing different properties of the graphs remains a future field of study.

As the summaries in the different fields of dependency analysis showed, there are still a number of dependency types we could study. Dependency through proximity and other context-related dependencies could provide benefits for analysis. The temporal dimension of dependencies is another field of further study. In the current implementation, all historical data are stored, but have not been used in analysis. Dynamic simulation of events and changes in the dependency graph would constitute an interesting line of future research.

Future research also includes related studies in other types of security software. Dependency studies could direct robustness testing efforts on the modules deemed to have the most impact on critical systems. Further research into the generation of test sets could improve their effectiveness in finding vulnerabilities. The testing efforts could serve to raise the bar for the security of critical systems.

5 Conclusion

The main goal for this paper was to examine AV software vulnerabilities and the risks they bring to critical information infrastructure systems. The PROTOS MATINE model was used as a method for disentangling the untrodden field of AV vulnerabilities in a rapid, iteratively expanding fashion. This paper presents the results of our research, which focused on AV software vulnerabilities and the picture depicted by the dependencies between these vulnerabilities.

By applying the PROTOS MATINE model through the study of media follow-ups, expert interviews, specifications, market situation, historical data, public vulnerability data and usage scenarios we found out that there are implementation level security issues in AV software that not only make it ineffective against malware, but also actually open new ways to attack the system. There is a substantial amount of information about such vulnerabilities in the past. AV products are an attractive target due to mono culture and high access privileges involved.

AV software itself was discovered to share a meta level one type of dependency risk exposure through the same archive formats being implemented in all AV products. This dependency risk spans beyond AV products, and when vulnerabilities related to archive forms are disclosed, then a very large vendor base of more generic products varying from consumer devices to network infrastructure have to be considered.

We found that issues with handling archive files have been the main reason for the fast rise of AV vulnerabilities in recent years. Our observations prompted robustness testing research of archive file formats in the PROTOS GENOME project. The results and the followup results presented here demonstrate that archive file formats are still a big issue in AV software. Ten months after public availability, preceded by a year-long period of limited distributions to vendors only, 4 out of 5 tested products were still vulnerable.

AV vendors do not necessarily fix vulnerabilities uncovered by published test sets. At least some AV vendors react to disclosed security issues and improve their products, but overall there is no significant trend for better or worse. More vulnerabilities akin to the ones we observed can be found by testing in a relatively straightforward fashion.

The results from media follow-up draw a quite desolate picture from the viewpoint of equal communication. Media (and the public) mostly do not recognise or discuss the risks related to dependency on the AV products. The AV vulnerabilities are seldom reported, and there truly is a lack of open discussion and controversial opinions, e.g. on the reliability of different AV software. Media concentrates on reporting new malware and fusions of AV enterprises.

Traditionally, AV software security has been measured by its ability to detect malware. However, some recent studies, even by the AV industry itself, have shown that the efficiency of malware may be of suspect. Nevertheless, AV is widely used despite this criticism on the effectiveness of the very approach. The use of AV software in critical in-

Figure 9. Comparison of CWE and Securityfocus vulnerability types in some serious vulnerabilitie

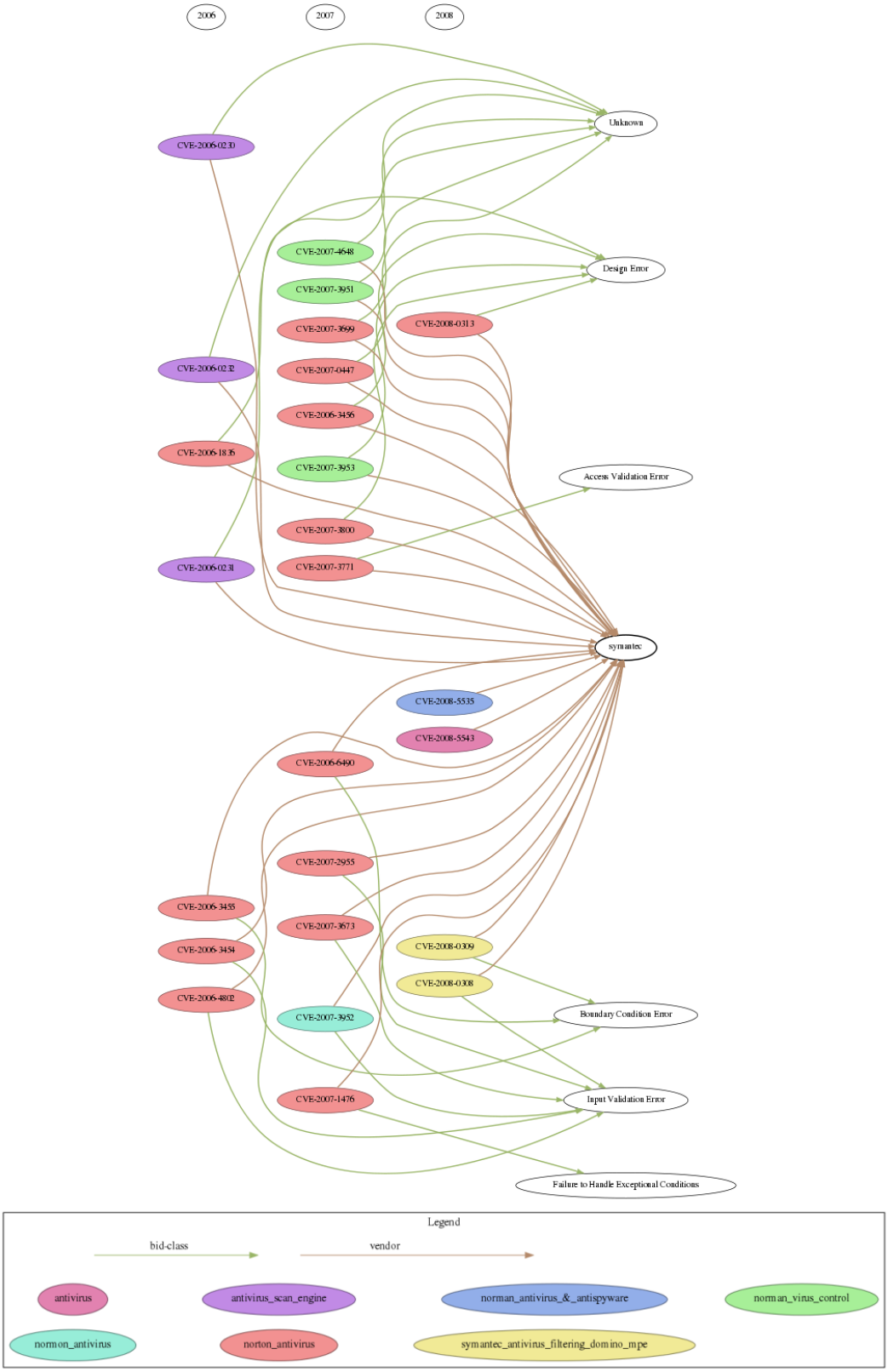


Figure 10. Some recent serious AV vulnerabilities in Symantec products

frastructure is wide spread and sometimes even mandated by laws and regulations

Our research indicates that AV software and AV vulnerabilities should be considered in the context of critical infrastructures. Firstly, awareness on the drawbacks of AV software should be spread among the planners of the critical infrastructures. Following this, the suitability of the software should be reconsidered on a system-by-system basis, along with the planning of divergent defences and defence strategies. Information on the interrelationships of different formats and products, and the vulnerability histories (track records) of the products can prove to be valuable decision-making tools in this process.

The context of AV software vulnerabilities and critical information infrastructure is still to be conceptualised by grounded theory. The graphs created in Graphingwiki could be analysed by means of graph theory to gain more insight into the dependencies. In order to do this, the semantic analysis should be further investigated. This would then provide the meaning for the mathematical results. Consistent and planned media follow-up as well as expert interviews would provide enough material for qualitative analysis. At the same time, a more thorough statistical analysis and graph theoretical approach could add more quantitative information. This future work should provide more in depth understanding of dependencies in AV software. More research should also be done in the field of automated data gathering methods since the media follow-up is laborious to perform manually. Automated data gathering would generate fewer errors, and on a large scale, it would give more reliable results.

This study on the vulnerability dependencies in AV software showed us that the PROTOS MATINE model is well suited for gathering information on a previously unknown subject, organising and analysing that information, finding points of interest for further, more focused research, and finally, extrapolating on the impact of the discovered vulnerabilities. The data gathering part of the method benefits from the use of multiple sources of information. The organising and analysing did benefit from the Graphingwiki visualisations, which pointed to the direction of archive file formats as the source of many vulnerabilities. Finally, expert interviews gave a wider perspective to the impact of these vulnerabilities and enabled the responsible vulnerability disclosure coordination effort, as we realised that these vulnerabilities could be present in various software beyond our initial target.

By applying the model, we collected antivirus prevalence, mandate and vulnerability track record data, we identified antivirus related risk factors and disclosed new information about media perception of antivirus, new vulnerabilities and handling of these vulnerabilities. In short, we disclosed and now understand better the risks that the antivirus software may pose on the critical infrastructure.

6 Acknowledgements

The authors would like to thank MATINE (Scientific Advisory Board for Defencce in Finland) and infotec Oulu for Financial support of the research. The authors express their gratitude also to Jani Kenttälä of Clarified Networks for valuable help on creating some of the pictures in this paper.

References

[1] *Advanced Packaging Tool* `http://en.wikipedia.org/wiki/Advanced\_Packaging\_Tool`, May 8, 2009

[2] Askola, K., Puupera, R., Pietikainen, P., Eronen, J., Laakso, M., Halunen, K., and Röning, J., *Vulnerability Dependencies in Antivirus Software*, SECURWARE 2008, The Second International Conference on Emerging Security Information, Systems and Technologies, pages 273-278, 2008

[3] Aslam T., Krsul I., and Spafford E. H., *Use of a taxonomy of security faults*, 19th NIST-NCSC National Information Systems Security Conference, pages 551-560, 1996.

[4] Bagheri, E. and Ghorbani, A. A., *The State of the Art in Critical Infrastructure Protection: A Framework for Convergence*, International Journal of Critical Infrastructures, Vol.4, no. 3, pages 215-244, 2008.

[5] Bassham, L. E. and Polk. W. T., *Threat Assessment of Malicious Code and Human Threats* (NISTIR 4939) `http://csrc.nist.gov/publications/nistir/ir4939.txt`, May 8, 2009

[6] Beizer, B. *Software Testing Techniques*, Second edition. (1990). International Thomson Computer Press. ISBN: 1-850-32880-3

[7] Chai, C-l., Liu, X., Zhang, W. J., Deters, R., Liu, D., Dyachuk, D., Tu, Y. L., and Baber, Z., *Social Network Analysis of the Vulnerabilities of Interdependent Critical Infrastructures*, International Journal of Critical Infrastructures, Vol.4, no. 3, pages 256-273, 2008.

[8] Christoderescu, M., Jha, S., Seshia, S. A., Song, D., and Bryant, R., *Semantics-Aware Malware Detection*, IEEE Symposium on Security and Privacy (S&P'05), pages 32-46.

[9] *Common Platform Enumeration* `http://cpe.mitre.org/`, May 8, 2009

[10] *Common Vulnerabilities and Exposures* http://cve.mitre.org/about/index.html, May 8, 2009

[11] *Common Vulnerability Scoring System* http://nvd.nist.gov/cvss.cfm?version=2, May 8, 2009

[12] *Common Weakness Enumeration* http://cwe.mitre.org/, May 8, 2009

[13] Cox L. and Delugah H.S., *Dependency Analysis Using Conceptual Graphs*, Proceedings of the 9th International Conference on Conceptual Structures, ICCS 2001.

[14] De Porcellinis, S., Setola, R., Panzieri, S., and Ulivi, G., *Simulation of Heterogeneous and Interdependent Critical Infrastructures*, International Journal of Critical Infrastructures, Vol.4, no. 1/2, pages 110-128, 2008.

[15] Diestel, R., *Graph Theory*, 3rd edition, Graduate Texts in Mathematics, Vol. 173, Springer-Verlag, Heidelberg, 2005.

[16] *Digitoday Finland*, http://www.digitoday.fi/, May 8, 2009

[17] Dixon, J., *How Good Is Your Network Neighborhood Watch*, http://media.techtarget.com/searchFinancialSecurity/downloads/How_Good_Is_Your_Network_Neighborhood_Watch.pdf, May 8, 2009

[18] Dudenhoeffer, D. D., Permann, M. R., and Manic, M., *CIMS: A Framework for Infrastructure Interdependency Modeling and Analysis*, Proceedings of the 2006 Winter Simulation Conference, pages 478-485.

[19] Eronen J. and Laakso M., *A Case for Protocol Dependency*, In proceedings of the First IEEE International Workshop on Critical Infrastructure Protection. Darmstadt, Germany. November 3-4, 2005.

[20] Eronen J. and Röning J., *Graphingwiki - a Semantic Wiki extension for visualising and inferring protocol dependency*, Proceedings of the First Workshop on Semantic Wikis (SemWiki2006 - From Wiki to Semantics), co-located with the 3rd Annual European Semantic Web Conference (ESWC). Budva, Montenegro, 11th - 14th June, 2006.

[21] Eronen, J., *A collaborative method for assessing the dependencies of critical information infrastructures* M.Sc. (Tech) Thesis for the Department of Electrical and Information Engineering at University of Oulu. URL: http://www.ee.oulu.fi/research/ouspg/protos/sota/matine/method-thesis/di.pdf, May 8, 2009

[22] *F-Secure Security Advisory FSC-2008-2*, http://www.f-secure.com/en_EMEA/support/security-advisory/fsc-2008-2.html, May 15, 2009.

[23] Hicks, B., *Network Anti-Virus Market Trends*, Faulkner Information Services, 2005.

[24] Hills, A. *Insidious Environments: Creeping Dependencies and Urban Vulnerabilities*, Journal of Contingencies and Crisis Management, Vol. 13, No. 1, pages 12-20, 2005.

[25] *HIPAA* http://www.cms.hhs.gov/HIPAAGenInfo/Downloads/HIPAALaw.pdf, May 8, 2009

[26] *International CIIP Handbook 2006 (Vol. II)*, eds. Myriam Dunn, Victor Mauer; Center for Security Studies, ETH Zurich.

[27] Itzwerth R. L., MacIntyre C. R., Shah S., and Plant A. J.,*Pandemic influenza and critical infrastructure dependencies: possible impact on hospitals*, The Medical Journal of Australia, 185, pages S70-S72, 2006.

[28] Ko K., Jang I., Kang Y., Lee J., and Eom Y., *Characteristic Classification and Correlation Analysis of Source-Level Vulnerabilities in the Linux Kernel*, Lecture Notes in Computer Science, Volume 3802, pages 1149-1156, 2005.

[29] Laakso, M., Takanen, A., and Röning, J., *Introducing Constructive Vulnerability Disclosures*, The 13th FIRST Conference on Computer Security Incident Handling, 2001.

[30] *Malware infections in protected systems*, http://www.pandasoftware.jp/scan/pdf/panda_lab_research_paper.pdf, May 8, 2009

[31] *National Vulnerabilty Database*, http://nvd.nist.gov/, May 8, 2009

[32] NATO, *The Protection of Critical Infrastructure*, Committee Report, 162 CDS 07 E rev 1, 2007, http://www.nato-pa.int/Default.asp?SHORTCUT=1165, May 15, 2009

[33] Ollman G., *Counting Vulnerabilities*, http://blogs.iss.net/archive/CountingVulns.html, May 8, 2009

[34] *OUSPG PROTOS-Genome Test Suite c10-archive*, http://www.ee.oulu.fi/research/ouspg/protos/testing/c10/archive/index.html, May 8, 2009

[35] Rinaldi, S. M., Peerenboom, J. P., and Kelly, T. K., *Identifying, understanding, and analyzing critical infrastructure interdependencies*, IEEE Control Systems Magazine, Vol. 21, no. 6, pages 11-25, 2001.

[36] Robert, B., de Calan, R., and Morabito, L., *Modeling Interdependencies Among Critical Infrastructures*, International Journal of Critical Infrastructures, Vol.4, no. 4, pages 392-408, 2008.

[37] Roivainen, H-L., *Discovery of hidden social networks in software companies*, M.Sc. (Tech) Thesis for the Department of Electrical and Information Engineering at University of Oulu, 2008.

[38] Rosato, V., Issacharoff, L., Tiriticco, F., Meloni, S., De Porcellinis, S., and Setola, R. *Modelling Interdependet Infrastructures using Interactins Dynamical Models*, International Journal of Critical Infrastructures, Vol.4, no. 1/2, pages 63-79, 2008.

[39] *Sarbanes-Oxley Act* http://www.sarbanes-oxley.com/section.php?level=1\&pub_id=Sarbanes-Oxley, May 8, 2009

[40] *Secunia Vulnerability Statistics* http://www.secunia.com, May 8, 2009

[41] *Cyber Assessment Methods for SCADA Security*, http://www.oe.energy.gov/DocumentsandMedia/Cyber_Assessment_Methods_for_SCADA_Security_Mays_ISA_Paper.pdf, May 15, 2009

[42] *Security Content Automation Protocol* http://nvd.nist.gov/scap.cfm, May 8, 2009

[43] *Securityfocus vulnerability database* http://www.securityfocus.com/vulnerabilities, May 8, 2009

[44] St. Neitzel, M., *Welcome to 2007: the year of professional organized malware ...* (HISPASEC) http://blog.hispasec.com/virustotal/recursos/welcome_2007.pdf, May 8, 2009

[45] Tolone, W. J., Wilson, D., Raja A., Xiang W., Hao H., Phelps S., and Johnson E. W.,*Critical Infrastructure Integration Modeling and Simulation*, Lecture Notes in Computer Science, vol. 3073, pages 214-225, Springer Berlin Heidelberg, 2004.

[46] Turner, D., Entwisle S., Fossi M., Blackbird J., McKinney D., Conneff T., Whitehouse O., *Symantec Internet Security Threat Report vol. X*, http://www.symantec.com/business/theme.jsp?themeid=threatreport, September 2006, May 8, 2009

[47] Viide, J., Helin, A., Laakso, M., Pietikäinen, P., Seppänen, M., Halunen, K., Puuperä, R., and Röning, J., *Experiences with Model Inference Assisted Fuzzing*, Second USENIX Workshop on Offensive Technologies (WOOT'08'), 2008.

[48] World Wide Web Consortium, *Resurce Descrption Framework* http://www.w3.org/RDF/, May 8, 2009

[49] Zio, E., *From Complexity Science to Reliability Efficiency: A New Way of Looking at Complex Network Systems and Critical Infrastructures*, International Journal of Critical Infrastructures, Vol.3, no. 3/4, pages 488-508, 2007.

DATA FORMAT DESCRIPTION AND ITS APPLICATIONS IN IT SECURITY

Michael Hartle[1]*, Andreas Fuchs*[2]*, Marcus Ständer*[1]*, Daniel Schumann*[1]*, Max Mühlhäuser*[1]

Telekooperation[1],
Dept. of Computer Science, TU Darmstadt / Germany,
{mhartle, staender, max}@tk.informatik.tu-darmstadt.de

Fraunhofer-Institut SIT[2], Darmstadt / Germany,
andreas.fuchs@sit.fraunhofer.de

ABSTRACT

Data formats play a central role in information processing, exchange and storage. Security-related tasks such as the documentation of exploits or format-aware fuzzing of files depend on formalized data format knowledge. In this article, we present a model for describing arbitrary data format instances as well as arbitrary data formats as a whole. Using the *Bitstream Segment Graph (BSG)* model and the *BSG Reasoning* approach, we describe a PNG image serving as exploit for Adobe Photoshop CS2 (CVE-2007-2365). We furthermore show directions how our work can be applied to secure data format design as well as formal security analysis.

Index Terms— Data format description, finite bit sequences, documentation of exploits, formal security validation

1. INTRODUCTION

1.1. Motivation

The concept of data formats is central to information processing, exchange and storage. A *data format* defines how information is represented as bits, bytes or characters and thus shapes every format-compliant process that *determines syntax and semantics of data* in order to access information. When a bit sequence is transmitted, both sender and receivers need to agree on its data format for interoperability.

1.2. Use Cases

Knowledge regarding syntax and semantics of data is relevant to IT Security as can be shown through several use cases related to data format knowledge:

- *Documentation of exploits* describe the composition of malicious crafted data. Such a documentation helps developers to explore syntax and semantics of malicious data, understand the delivery mechanism of the exploit, and so to fix processing bugs in affected applications.
- The definition of a data format can include security-relevant flaws hidden in the complexity of a specification, which may lead to exploitable implementations. A visual refined representation can aid security experts with the *identification of tripwires* such as redundant information. If applied during during the modelling phase, such flaws may be prevented or at least be limited by adding security-related programming advice to the specification.
- Similarly, good knowledge about utilized data formats is necessary in the area of *formal security validation*, which is among others required for certain Assurance Levels in Common Criteria. Formally based data format descriptions can therefor fullfill the requirements for reasoning about assumptions and abstractions made in this process.

1.3. Problem

For such use cases, a vast, growing number of existing and evolving data formats turns designing, implementing and maintaining format-specific solutions into a repetitive, neverending task. We can improve the situation by separating data format knowledge into reusable *declarative*, *machine-processible* descriptions for *format-agnostic* solutions. Realizing this idea requires formalized models on *data format instances* as well as on *data formats as a whole*, where we define a data format instance as a finite bit sequence and a data format as a (possibly infinite) set of data format instances [1].

Data formats have been subject of research in various domains like *Multimedia* or *Telecommunication*, including related work on describing data formats. In *Digital Preservation*, substantial related work for managing data formats exists like the *Open Archival Information System* (OAIS) [2], data format registries such as *Global Data Format Registry* (GDFR) [3] and *PRONOM* [4], or the *Typed Object Model* (TOM) [5] for managing format-related operations on data.

Yet, existing approaches for formally describing data formats is domain-specific and not applicable for arbitrary data formats in general.

1.4. Contribution

We present two contributions for describing both data format instances as well as data formats in the context of IT Security. The first contribution is the *Bitstream Segment Graph (BSG)* model for describing arbitrary data format instances. The second contribution is the *BSG Reasoning* approach for describing arbitrary data formats. We extend previous publications [1, 6, 7] with an analysis on modelling arbitrary data formats in general, the Apeiron BSG editor as tool support for creating, exploring and manipulating BSG instance descriptions, as well as a RDF/N3-based notation for BSG instances which makes them interchangeable and machine-processible.

1.5. Outline

We start with surveying related work in literature on data format description from the domains of Multimedia, Telecommunication and Grid Computing in Section 2, focusing on their descriptive capabilities. We then abstract and analyse elemental properties of both data formats and their instances. Here, we investigate how to guarantee these properties in formal models in Section 3, showing theoretical limits given by formal language and computational theory. Building on our analysis, we present the BSG model for describing data format instances, and our rule-based BSG Reasoning approach for describing data formats in Section 4. Subsequently, the presented contributions are put in the context of IT Security by giving an example of their application on an PNG image exploit for Adobe Photoshop CS2 based on CVE-2007-2365 in Section 5. In addition, we describe further applications such as flaw detection during design and formal security validation in Section 6 and close with a summary in Section 7.

2. RELATED WORK

2.1. Data Format Description

We present related work on describing data formats from the research domains of *Multimedia*, *Telecommunication* and *Grid Computing*.

Here, our focus is on the descriptive capabilities of each approach regarding *structures*, *transcodes*, *fragments* and *primitives*, as well as handling *functional dependencies* between them. Quite self-explanatory, a structure represents a concatenation of data. A primitive is encoded data which represents some information with defined semantics. We define a transcode as data which is compressed, encrypted or the result of another reversible, information-preserving block transformation. Moreover, we define a fragment as data that is part of a larger composite. Functional dependencies include cases where the length of a data segment is determined by another, or where the value of a data segment depends on a computation of another, as for CRCs.

For describing the composition of data format instances in general, all these descriptive capabilities are required. For example, the PNG image shown in Figure 5 carries transformed and compressed image data which is fragmented in two segments.

2.1.1. Multimedia

Primary motivation for research in Multimedia on data format description is the *standardization of data formats*, and the *adaptation of digital items* for Universal Media Access (UMA). Both the *MPEG 1/2 methodology* and the *MPEG SDL and Flavor/XFlavor* approach belong to the former line of research, whereas the latter contributed the *Bitstream Syntax Description Language* (BSDL). There are further approaches such as *BFlavor and gBFlavor* [8, 9] as recombinations of BSDL and Flavor, but which do not provide significant extentions regarding their descriptive capabilities.

2.1.1.1. MPEG 1/2 methodology

The "MPEG-1/2 methodology" [10] was used for describing data formats in various parts of the MPEG-1 (ISO/IEC 11172) and MPEG-2 (ISO/IEC 13818) standards.

It is a notation similar to C `struct` definitions, used in a tabular form to describe elements of a data structure with the respective name, data type and size in bits. As the methodology does not cover functional dependencies between elements, these are typically described textually.

The methodology is a "visual language" and targets humans as audience. It is intended for static data structures featuring bit granularity description, but does not handle functional dependencies such as variable-length data ocurring for the Exponential Golomb integer encoding, in PNG chunks or in MPEG-4 boxes.

2.1.1.2. MPEG SDL and Flavor/XFlavor

The *Syntactic Description Language* (SDL) was proposed for describing variable-length data in MPEG-4 data formats [11], was included as part of the MPEG-4 Systems and Description Languages (MSDL) [12] and used in the specification of MPEG-4. It later became the *Formal Language for Audio-Visual Object Representation (Flavor)* with its XML-based extension XFlavor [13, 14].

In Flavor code, data structures are described as *classes*, mixing data declarations with procedural flow-control statements. Flavor enables the translation to Java and C++ code by generating appropriate methods for parsing and serialization. The language focuses on H.263 or MPEG-2 Video which do not need data to be decoded during parsing, and therefore lim-

its itself to parsing only without any decoding, circumventing this "high-level context" problem [15].

Flavor code is procedural and targets machine-processing. It allows for bit granularity of description, covers both structures and primitives, and handles simple functional dependencies such as variable-length data structures. It does neither handle complex functional dependencies such as CRCs, nor does it explicitly support transcoded data or fragmentation.

2.1.1.3. Bitstream Syntax Description Language (BSDL)

The *Bitstream Syntax Description Language* (BSDL) addresses the *Universal Media Access* (UMA) vision as part of the MPEG-21 Digital Item Adaptation standard in Multimedia [16]. It allows for the adaptation of digital items to enable timely delivery of and access to multimedia resources in highly heterogenous, resource-contrained environments.

BSDL extends XML Schema for describing binary data. It focuses on so-called *scalable data formats*, where an specific adaptation of a bitstream is generated through computationally simple filtering rather than re-encoding. Where BSDL supports the definition of format-specific schemata, the *generic BSDL* (gBSDL) variant defines a format-agnostic, general-purpose alternative for use on low-resource environments such as mobile phones.

BSDL allows for bit granularity and is declarative. It supports structures and primitives, but does not explicitly cover transcoded data or fragmentations. Functional dependencies are described using XPath and are evaluated during runtime on an in-memory XML Document Object Model (DOM) representation of parsed data which may still be incomplete. As a consequence, adapting digital items using the BSDL reference implementation suffers from performance and scalability issues [8].

2.1.2. Telecommunication

Similar to Multimedia, the primary motivation on data format description in Telecommunication is the *standardization of data formats* for enabling interoperability in communications between different parties. Yet here, the focus is on representing data in a defined way rather than describing arbitrary representations in their own composition.

Related work includes the *External Data Representation* (XDR), the well-known *Abstract Syntax Notation One* (ASN.1) including the *Encoding Control Notation* (ECN) and other approaches such as the *Concrete Syntax Notation 1* (CSN.1) or *Transfer Syntax Notation One* (TSN.1).

2.1.2.1. External Data Representation (XDR)

The External Data Representation (XDR) is currently defined in RFC 4506 [17] and standardizes a language for describing data and its encoding. It has been used in the definition of network protocols Sun Remote Procedure Call (RPC) [18], the Network File System (NFS) [19] and others.

The XDR language somewhat resembles the C programming language, but does not include flow-control statements and is declarative. It intends to provide support for the exchange of messages using common high-level data types rather than describing arbitrary data.

XDR provides support for structures and primitives from a limited set of data types. It does neither provide bit-level granularity nor explicitly support transcoded data or fragmentation. Regarding simple functional dependencies, it supports variable-length data through specific data types.

2.1.2.2. Abstract Syntax Notation One (ASN.1) and Encoding Control Notation (ECN)

The *Abstract Syntax Notation One* (ASN.1) has been defined by the International Telecommunication Union, Telecommunication Standardization Sector (ITU-T) [20] and allows the definition of data models as ASN.1 modules. In combination with encoding rules such the Packed Encoding Rules (PER) [21], an ASN.1 module defines a specific data format. Alternativ encoding rules can be specified through the *Encoding Control Notation* (ECN) [22]. ASN.1 is used in the definition of numerous file formats and network protocols, such as for X.509 security certificates [23] or for H.225 and H.245 messages used in the H.323 video conferencing protocol [24].

ASN.1 allows the definition of an abstract data model in a declarative manner. The design of ASN.1 and its encoding rules follow the separation of Application layer and Presentation layer in the ISO OSI stack, which allows for the renegotation of representations [25]. While the separation of ASN.1 module definition and its encoding makes sense in an interactive scenario where encodings can be renegotiated, it provides no benefit for data formats in general, as the definition of data models and its representation is often fixed in advance and without means for renegotiation.

ASN.1 allows the description of structures and primitives, yet it does not natively cover transcoded data or fragmentation. Quite specific functional dependencies are supported such as for variable-length data.

2.1.2.3. Concrete Syntax Notation 1 (CSN.1)

The Concrete Syntax Notation 1 (CSN.1) is a language used in the definiton of messages for GSM and UMTS communication protocols by the European Telecommunication Standards Institute (ETSI). CSN.1 has a published specification [26] and is informally described in Annex B of ETSI TS 100 939 [27].

CSN.1 is declarative and describes the composition of data on the bit level quite similar to a formal language grammar. Although similar in naming to ASN.1, CSN.1 directly describes the composition of data on the bit level, whereas ASN.1 describes an abstract data model.

CSN.1 provides support for structures, primitives and offers quite extensive support of functional dependencies com-

pared to all other approaches. However, it does not provide explicit support for transcoded data or fragmentation of data.

2.1.2.4. Transfer Syntax Notation One (TSN.1)

The *Transfer Syntax Notation One* (TSN.1) has been specified by the company Protomatics, Inc [28] and is used in their commercial product offerings. It defines a language for describing the composition of data, yet to our knowledge, it has not been used for existing, published specifications of well-known data formats.

Naming of TSN.1 follows the ASN.1 and CSN.1 acronyms, but the language describes data in a procedural way not unlike the Flavor approach from Multimedia, mixing data declarations with flow-control statements.

TSN.1 is intended for machine-processing and is capable of describing structures and primitives including the simple functional dependency of variable-length data. As with others, it does not explicitly handle transcoded data or fragmentation.

2.1.3. Grid Computing

Last but not least, describing existing data sets for exchange of research results is of interest in Grid Computing. It includes approaches such as *Binary Format Description* (BFD) and the *Data Format Description Language* (DFDL).

2.1.3.1. Binary Format Description (BFD)

Binary Format Description (BFD) is an XML-based language intended for describing "arbitrary" data formats of scientific datasets [29]. It was developed under funding of Pacific Northwest National Laboratory (PNNL) in 2000 and was part of the Scientific Annotation Middleware (SAM) project by the U.S. Department of Energy [30].

BFD extends the *Extensible Scientific Interchange Language* (XSIL) with flow-control statements and enables functional dependencies through XPath-based references to be resolved during runtime. It is intended for machine consumption, is procedural and covers both structures and primitives. It does not explicitly handle transcoded data or fragmentation.

2.1.3.2. Data Format Description Language (DFDL)

The *Data Format Description Language* (DFDL) is a format description language with the explicit goal of describing any data format [31]. It is an extension to XML Schema and is currently defined in working draft 032 of its initial version 1.0.

Similar to the BSDL approach, DFDL extends the XML Schema with application-specific annotations. In its current form, DFDL focuses on "hierarchical nested data", providing support for both structures and primitives. It also handles functional dependencies through XPath-based references. Yet, it does not explicitly handle transcoded data or fragmentation.

2.1.4. Summary

The descriptive capabilities of related work presented from Multimedia, Telecommunication and Grid Computing are summarized in Table 1. There are several aspects we observed that are interesting to note:

- Related work focuses on specific aspects of data formats, such as simple adaptation of scalable data formats in Multimedia, or ensuring interoperability through defining data models in Telecommunication, often limiting their descriptive capabilities to these aspects. A thorough analysis on modelling data formats in general is notably absent from all of them.
- The distinction between representing and describing data is sometimes blurred in literature, although both tasks are conceptually different. Representing data can be limited to specific representations of information, whereas describing data requires complete support of arbitrary information representations and thus is more complex to achieve.
- Next to no cross-pollination apparently occurs between domains in this regard, although several approaches across domain boundaries have adopted the use of XML, XML Schema and XPath.

There are sufficient machine-processible, declarative approaches that both handle structures and primitives on bit granularity as well as functional dependencies on a simple level. Yet, explicit support for transcoded data and fragmentations is missing from all these approaches. The former is required for handling compressed, encrypted, or otherwise transformed data, whereas the latter is required for interleaved data typical for multimedia files. These cases can be observed in PNG images or MPEG-4 movies.

3. ANALYSIS

3.1. Abstraction

Let $\mathbb{D}$ denote the set of all data formats, $d \in \mathbb{D}$ denote a data format, α denote a sender and β denote a receiver.

HYPOTHESIS 3.1: The current state-of-the-art on data format description can be improved by considering a data format d as a normative set of lossless information representations for purposes of storage and transmission between a sender α and a receiver β.

Following the hypothesis, the basic usage scenario of a data format shown in Figure 1 can be stated as follows:

- A sender α has an internal representation m_α of information. The sender ensures the validity of m_α with respect to a data format d and maps m_α to an external representation m_d, which is then sent over a channel c.

	MPEG 1/2	(X)Flavor	BSDL	XDR	ASN.1 & ECN	CSN.1	TSN.1	BFD	DFDL
Support of structures	⊠	⊠	⊠	⊠	⊠	⊠	⊠	⊠	⊠
Support of primitives	⊠	⊠	⊠	⊠	⊠	⊠	⊠	⊠	⊠
Support of transcodes	□	□	□	□	□	□	□	□	□
Support of fragments	□	□	□	□	□	□	□	□	□
Machine-processible	□	⊠	⊠	⊠	⊠	⊠	⊠	⊠	⊠
Declarative, not procedural	⊠	□	⊠	⊠	⊠	⊠	□	⊠	⊠

Table 1. Comparison of descriptive capabilities

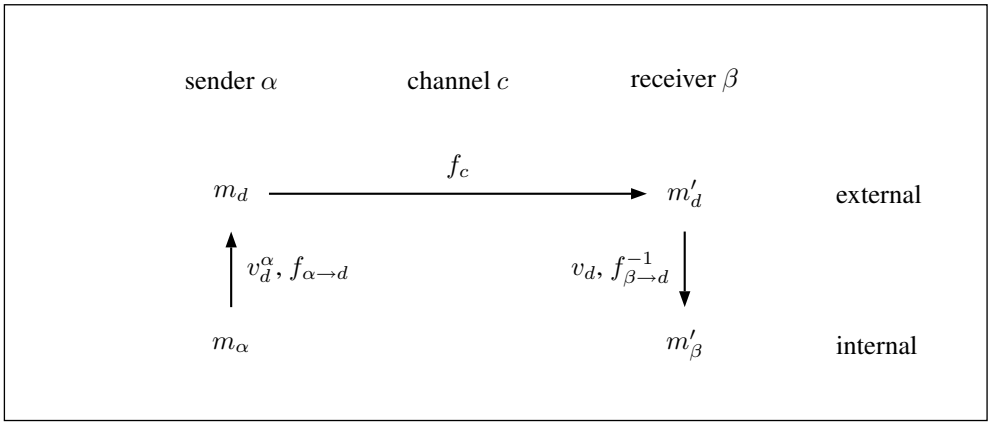


Fig. 1. Abstract information transport using a data format

- A receiver β eventually obtains an external representation m'_d from c, which may be invalid due to an erronous sender, or be different from the originally sent m_d in case of a noisy channel. The receiver therefore ensures the validity of m'_d and maps m'_d to an internal representation m_β.

Both sender and receiver necessarily share information concerning the data format d. Moreover, depending on d, both α and β may share additional context information required for deciding the validity, or for mapping from and to an external representation m_d. Example use cases are the identification of embedded data formats through a separate channel, or the use of encryption in a data format.

3.1.1. Lossless versus lossy data formats

In this regard, it is necessary to clarify the expression "lossless information representation" from our hypothesis with regards to the distinction between lossless and lossy data formats.

A *lossless data format* represents an original information, whereas a *lossy data format* represents an approximation of original information according to a defined metric. In either case, the actually represented information is to be recovered by the receiver, be it original or an approximation, so mappings from and to m_d are necessarily information-preserving. For the course of this article, we therefore define a data format d to specify the representation of information *in a lossless manner*. Aspects regarding the preprocessing of information to be represented through a data format, such as approximations and related metrics for lossy data formats, are not within the scope of this article.

3.2. Formalization

Informally, we define a *data format instance* as a mapping between an internal representation m_γ and an external representation m_d, and define a *data format* through a set of such instances. Formalizing both terms in that order, the following definitions are given step-wise as follows:

3.2.1. Encoding data

As a first step, a way to represent data in an encoded form.

DEFINITION 3.1 (BIT SEQUENCE): A bit sequence b is defined as finite and non-empty. The set of all finite, non-empty bit sequences is defined as $\mathbb{B}$ (Eq. 1).

$$b = \{0,1\}^n, n \geq 1, b \in \mathbb{B} \tag{1}$$

DEFINITION 3.2 (ENCODING): An *encoding* e is a bijective function which maps between an element $x \in \mathbb{X}_e$ and its corresponding bit sequence b (Eq. 2).

$$e : \mathbb{X}_e \leftrightarrow \mathbb{B}_e, \mathbb{B}_e \subseteq \mathbb{B} \tag{2}$$

3.2.2. Representing information

A bit sequence represents encoded *data*, but does not describe its meaning by itself, as it depends on the actual context. In order to represent data including its semantics as *information*, some sort of "labeling" is needed.

DEFINITION 3.3 (LABEL): A *label* l is a symbol that denotes some given semantics. The set of all labels is defined as $\mathbb{L}$.

DEFINITION 3.4 (LABELED BIT SEQUENCE): A *labeled bit sequence* i is defined as a pair $i = (b, \mathbb{L}_i)$, where $b \in \mathbb{B}$ is a bit sequence and $\mathbb{L}_i \subseteq \mathbb{L}$ is a subset of labels that denote the meaning of b (Eq. 3). The set of all labeled bit sequences is defined as $\mathbb{I}$.

$$i = (b, \mathbb{L}_i), b \in \mathbb{B}, \mathbb{L}_i \subseteq \mathbb{L}, i \in \mathbb{I} \tag{3}$$

A labeled bit sequence represents *information* by making the meaning of encoded data explicit regarding the specific context of d. A labeled bit sequence can be categorized depending on whether its information is part of the message to be transported, or whether it is used as "packaging" to enable transportation.

DEFINITION 3.5 (PAYLOAD): A labeled bit sequence $i = (b, \mathbb{L}_i)$ is *payload* if the value of its bit sequence b is functionally independent from other labeled bit sequences.

DEFINITION 3.6 (PACKAGING): A labeled bit sequence $i = (b, \mathbb{L}_i)$ is *packaging* if the value of its bit sequence b is functionally dependent on one or more labeled bit sequences, such as depending on their (relative) location, length, labels or bit sequences.

3.2.3. Representing complex information

Labeled bit sequences serve as building blocks for more complex representations, which are used either as *internal representation* at a sender or receiver γ, or used as *external representation* for exchanging information between a sender and a receiver.

DEFINITION 3.7 (INTERNAL REPRESENTATION): An *internal representation* m_γ represents information in a way that is specific to some sender / receiver γ and is defined as a tuple of one or more labeled bit sequences (Eq. 4) which are semantically distinct. The set of all internal representations is defined as $\mathbb{IR}$.

$$m_\gamma = \{i_1, \ldots, i_n\}, n \geq 1, i_x \in \mathbb{I}, m_\gamma \in \mathbb{IR} \tag{4}$$

Different internal representations may represent the same information, yet in varying *granularity*.

DEFINITION 3.8 (GRANULARITY): The *granularity* of an internal representation m_γ is a relative measure on how fine-grained information is represented. Higher granularity is achieved by a more fine-grained description. The actual granularity of m_γ may vary depending on the sender or receiver γ.

EXAMPLE 3.1: Let $m_{\gamma,1} = \{i\}, i = (b, \mathbb{L}_i)$ be an internal representation, where i represents the color of a pixel as a 24 bit RGB value composed of 8 bits for each color component of red, green and blue. Let $m_{\gamma,2} = \{i_1, i_2, i_3\}, i_x = \{b_x, \mathbb{L}_{i,x}\}$ be an internal representation, where i_1, i_2 and i_3 represent the color of a pixel as a 24 bit RGB value as separate 8 bit red, green and blue color components. In this case, $m_{\gamma,2}$ has a higher granularity than $m_{\gamma,1}$.

Packaging is typically present in a data format in order to describe variable aspects of payload required during the parsing process, such as the length of a variable-length payload. In order to compute packaging information during generation and process packaging information during parsing, a certain minimum granularity of internal representation is required which separates packaging from payload.

DEFINITION 3.9 (EXTERNAL REPRESENTATION): An *external representation* m_d represents information as normatively defined by a data format d. It is defined as a tuple containing exactly one labeled bit sequence (Eq. 5). The set of all external representations is defined as $\mathbb{ER}$.

$$m_d = \{i\}, i \in \mathbb{I}, m_d \in \mathbb{ER} \tag{5}$$

An external representation m_d typically carries some aggregation of information rather than a single primitive value. An external representation m_d is exchanged between sender and receiver over some *channel*.

DEFINITION 3.10 (CHANNEL): An *channel* c passes an external representation $m_d = \{i\}, i = \{b, \mathbb{L}_i\}$ from a sender α to a receiver β, including the bit sequence b and its labels $\mathbb{L}_i$. It is modelled as a channel function f_c (Eq. 6).

$$f_c : \mathbb{ER} \rightarrow \mathbb{ER} \tag{6}$$

A channel c handles the transmission of an external representation $m_d = \{i\}, i = \{b, \mathbb{L}_i\}$ by transferring both the bit sequence b and its labels $\mathbb{L}_i$. A specialized channel c may only pass external representations for a specific set of data formats. Furthermore, a channel c may be noisy and introduce errors into the bit sequence or the set of labels.

3.2.4. Mapping between representations

In order to map between internal and external representations, some means for a bijective *transformation* is needed.

DEFINITION 3.11 (TRANSFORMATION): A *transformation* t is a bijective function which maps between *input* and *output* as two ordered tuples of bit sequences (Eq. 7). As shown in Figure 2, transformations can be categorized through the cardinality of the input and output tuples as

- a *segmentation* of structured data $(1 : m)$,
- a *block transformation* of transcoded data $(1 : 1)$, and
- a *concatenation* of fragmented data into a composite $(n : 1)$.

Arbitrary $n : m$ transformations can be composed from segmentations, block transformations and concatenations. A transformation may optionally have additional parameters that control the bijective mapping.

$$t : \mathbb{B}^n \leftrightarrow \mathbb{B}^m, n \geq 1, m \geq 1 \tag{7}$$

The bijective mapping between an internal representation m_γ and an external representation m_d as defined through transformations gives rise to a *data format instance*.

DEFINITION 3.12 (DATA FORMAT INSTANCE): Given a pair of representations (m_d, m_γ) with $m_d = \{i_0\}, m_\gamma = \{i_1, \ldots, i_n\}, n \geq 1$, a *data format instance* is a rooted, directed, acyclic graph as a *causality graph* on labeled bit sequences. The graph has root in i_0 and has $i_1, \ldots, i_n$ as its leaves. It is composed from a finite set of transformations, where each transformation t defines directed arcs from its input to its output bit sequences. Regarding intermittent nodes in the causality network, their bit sequences is the result of transformations, while their labels functionally depend on neighbouring labeled bit sequences as well as on optional context information depending on d.

As we now have defined the concept of a data format instance, we can proceed towards data formats as potentially infinite sets of data format instances.

3.2.5. Mapping between sets of representations

DEFINITION 3.13 (INTERNAL VALIDATION FUNCTION): For a given sender α and data format d, an *internal validation function* is defined as v_d^α (Eq. 8). An internal representation m_α is *valid* iff $v_d^\alpha(m_\alpha) = 1$. The subset of all valid internal representations of α for d is defined as $\mathbb{IR}_d^\alpha \subseteq \mathbb{IR}$.

$$v_d^\alpha : \mathbb{IR} \rightarrow \{0, 1\} \tag{8}$$

As a data format typically does not provide means for transporting arbitrarily labeled bit sequences, a sender α tests whether an internal representation is valid or not prior to creating a corresponding external representation.

DEFINITION 3.14 (EXTERNAL VALIDATION FUNCTION): For a given $d \in \mathbb{D}$, we define an *external validation function* v_d (Eq. 9). An external representation m_d is *valid* iff $v_d(m_d) = 1$. The subset of all valid external representations for d is defined as $\mathbb{ER}_d \subseteq \mathbb{ER}$.

$$v_d : \mathbb{ER} \rightarrow \{0, 1\} \tag{9}$$

A received external representation m'_d may be invalid, for example due to a degrading storage medium or due to interference on a network link. A receiver β thus must test whether the received m'_d is valid.

In order to transport information from the internal representation $m_\alpha \in \mathbb{IR}_d^\alpha$ to the external representation $m_d \in \mathbb{ER}_d$, and vice versa from a valid external representation $m'_d \in \mathbb{ER}_d$ to the internal representation $m'_\beta \in \mathbb{IR}_d^\beta$, a suited mapping between both sets becomes necessary.

DEFINITION 3.15 (MAPPING FUNCTION): For a given sender α and data format $d \in \mathbb{D}$, a bijective *mapping function* $f_{\alpha \rightarrow d}$ (Eq. 10) maps from $\mathbb{IR}_d^\alpha$ to $\mathbb{ER}_d$ through *encoding* and *serialization*. For a given receiver β and data format $d \in \mathbb{D}$, its inverse $f_{\beta \rightarrow d}^{-1}$ (Eq. 11) maps from $\mathbb{ER}_d$ to $\mathbb{IR}_d^\beta$ through *parsing* and *decoding*.

$$f_{\alpha \rightarrow d} : \mathbb{IR}_d^\alpha \rightarrow \mathbb{ER}_d \tag{10}$$

$$f_{\beta \rightarrow d}^{-1} : \mathbb{ER}_d \rightarrow \mathbb{IR}_d^\beta \tag{11}$$

Both sets $\mathbb{ER}_d$ and $\mathbb{IR}_d^\alpha$ have the same size due to the bijectivity of export and import functions. Both sets $\mathbb{ER}$ and $\mathbb{IR}$ are infinite. Depending on the data format d, the sets $\mathbb{ER}_d$ and $\mathbb{IR}_d^\alpha$ may be finite. Building on the previous definitions, the notion of a *data format* can now be defined.

DEFINITION 3.16 (DATA FORMAT): A data format d is a possibly infinite set of data format instances, which map between a normative $\mathbb{ER}_d \subseteq \mathbb{ER}$ and a canonical $\mathbb{IR}_d^\gamma \subseteq \mathbb{IR}$ and which makes assumptions on the underlying channel c.

3.3. Elemental properties

We can observe elemental properties of a data format for sender and receiver along the flow of information in Figure 1:

- *The sender α can decide whether the internal representation m_α is valid using function v_d^α.* For example, given GIF89a as data format and m_α as an image, the

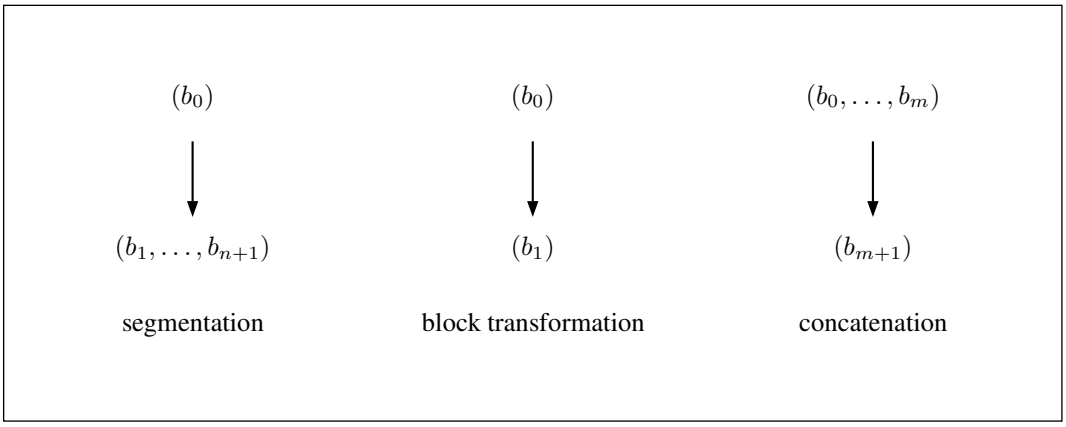


Fig. 2. Transformations ordered by input and output cardinality.

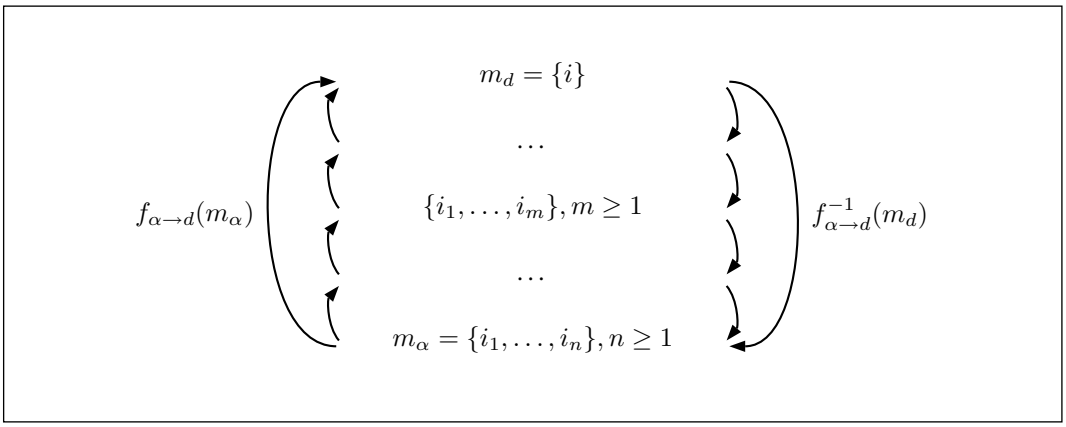


Fig. 3. Bijective mapping between m_α and m_d

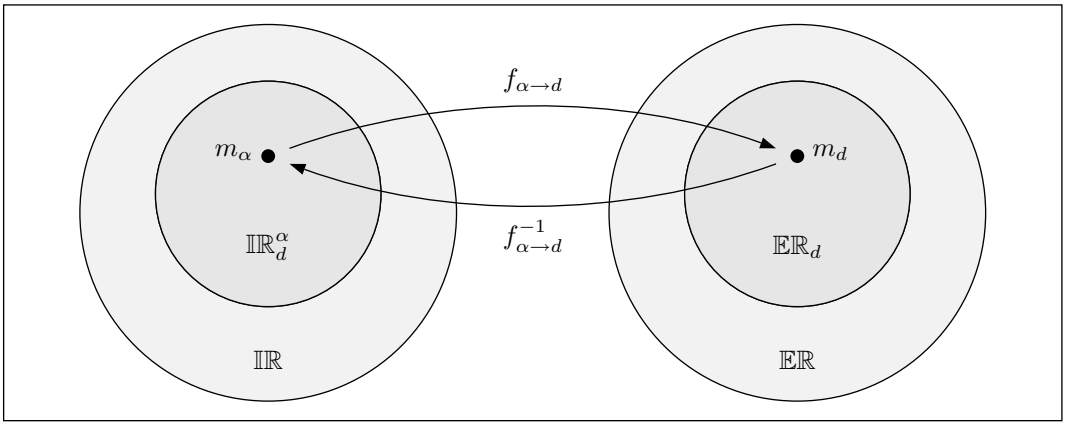


Fig. 4. Bijective mapping between internal representations $\mathbb{IR}^\alpha_d$ and external representations $\mathbb{ER}_d$ through mapping function $f_{\alpha \to d}$ and its inverse $f^{-1}_{\alpha \to d}$

validation function v_d^α would fail if the width of the image were 65536 pixels, as the data format constraints it to $2^{16} - 1 = 65535$ pixels or less.

- *The sender α can compute a valid external representation m_d from a valid internal representation m_α using the bijective function $f_{\alpha \to d}$.* This computation includes steps such as encoding values and serializing the external representation.
- *The receiver β can decide whether the external representation m'_d is valid using function v_d.* The channel c may have introduced errors that invalidate the external representation m'_d, such as bit flips during storage on a deteriorating medium, or interference on a network link during transmission.
- *The receiver β can compute a valid internal representation m'_β from a valid external representation m'_d using the bijective function $f^{-1}_{\beta \to d}$.* This mapping includes steps such as parsing the external representation and decoding its values.

For a formalization of data formats, it thus is desirable for a model on data formats to guarantee *bijectivity*, *decidability* as well as its overall *consistency*.

3.4. Limits for modelling arbitrary data formats

An ideal model would guarantee both the decidability of functions v_d^α, v_d, $f_{\alpha \to d}$ and $f^{-1}_{\beta \to d}$ as well as the bijectivity of functions $f_{\alpha \to d}$ and $f^{-1}_{\beta \to d}$. Moreover, it would guarantee the consistency of validation and mapping functions. Based on established results from formal languages and computational theory, we will now show to which degree this is possible.

3.4.0.1. Decidability

To guarantee decidability, we ideally would like to model the set of *deciders*, that is, machines that always halt. Yet, the problem of deciding whether an Turing Machine (TM) accepts a given input is the well-known *Halting Problem* A_{TM} (Eq. 12), which is undecidable [32, 33].

$$A_{TM} = \{\langle M, w \rangle \mid M \text{ is a TM and accepts } w\} \tag{12}$$

Reducing the computational power of a TM to a linear-bounded automaton (LBA), decidability can be guaranteed again, as the corresponding problem A_{LBA} is decidable [33]. Yet, a reduction from TMs to LBAs excludes valid deciders from being modelled as well.

3.4.0.2. Bijectivity

To guarantee bijectivity, we can represent a mapping function as a *reversible automaton*, where reversibility is ensured through its bijective transition function σ. Due to a finite set of states and an alphabet of finite size, the transition function consists of a finite number of transition formulas, which can be represented as tuples and for which bijectivity can be decided. Bennett proved the Reversible Turing Machine (RTM) as equivalent to the TM in computational power by showing that a three-tape RTM can compute a single-tape TM [34].

3.4.0.3. Consistency

Consistency between bijective mapping functions $f_{\alpha \to d}$ and its inverse $f^{-1}_{\alpha \to d}$ can be guaranteed through a reversible automaton. As well, consistency between $f_{\alpha \to d}$ and v_d^α for sending as well as $f^{-1}_{\alpha \to d}$ and v_d for receiving can be guaranteed for decidable bijective mapping functions by defining v_d^α and v_d to return 1 iff the respective $f_{\alpha \to d}$ and $f^{-1}_{\alpha \to d}$ accepts and thus decides the input, and 0 otherwise.

3.4.0.4. Summary

Computational theory places significant limits on modelling arbitrary data formats in general. A model can guarantee bijectivity, yet no model can both cover arbitrary data formats and still guarantee decidability. We thus opt for a second-best approach by dropping guaranteed decidability as property in favor of a generic approach.

4. MODELS

4.1. Data Format Instances

The following definition of the *Bitstream Segment Graph* from the original publications in [1, 6] has been adapted to the context of this article and extended with a list of operations and an RDF/N3-based storage representation.

4.1.1. Definition

For defining a model to describe the composition of data in a canonic form, we now define the *Bitstream Segment Graph* building on the causality network idea from the analysis.

DEFINITION 4.1 (BITSTREAM SEGMENT): Given a bitstream segment $v \in V$, the set of bitstream segments V, the set of finite consecutive bit sequences $B = \{0,1\}^n, n \in \mathbb{N} \setminus \{0\}$ and $\varphi : V \mapsto B$, then the bitstream segment v represents a finite consecutive bit sequence $\varphi(v) \in B$.

DEFINITION 4.2 (BITSTREAM SOURCE): A bitstream source is a root bitstream segment $v_{Root} \in V$ with a defined $\varphi(v_{Root})$.

A bitstream source represents a digital item which is composed according to a data format. Files, network packets or file systems on some storage medium are examples for octet-aligned bitstream sources.

DEFINITION 4.3 (BITSTREAM ENCODING): Given a bitstream encoding $e = (rel, v, l) \in R_E, v \in V, l \in L$, where R_E is the set of bitstream encodings and L is the set of literals,

Used in encoding $e \in R_E$?	Used in transformation $t \in R_T$?		Type	RDF Type
no	no	(as input)	Generic	`bsg:generic`
yes	no	(as input)	Primitive	`bsg:primitive`
no	segmentation	(as input)	Structure	`bsg:structure`
no	transformation	(as input)	Transcode	`bsg:transcode`
no	concatenation	(as input)	Fragment	`bsg:fragment`
no	concatenation	(as output)	Composite	`bsg:composite`

Table 2. Types of bitstream segments.

then for a given v, e specifies a mapping relation $rel(\varphi(v), l)$, required to be bijective. It is abbreviated with $\phi(v) = l$, where $\phi : V \mapsto L$.

A bitstream segment can represent an encoded literal that is part of the data contained in a bitstream source (a primitive). For example, there are two bitstream segments within a PNG file which contain encoded integers that represent the width and height of the image.

DEFINITION 4.4 (BITSTREAM TRANSFORMATION): Given a bitstream transformation $t = (rel, V_{in}, V_{out}, P) \in R_T$, where V_{in}, V_{out} are totally ordered sets with $V_{in} \subset V, V_{out} \subset V, V_{in} \neq \emptyset, V_{out} \neq \emptyset, V_{in} \cap V_{out} = \emptyset$, R_T is the set of bitstream transformations and P is the set of parameters, then t specifies a mapping relation $rel(V_{in}, V_{out}, P)$, required to be bijective, between V_{in} and V_{out} under application of P.

In general, a bitstream transformation t bijectively maps a set of input bitstream segments V_{in} as *predecessors* to a a set of new bitstream segments V_{out} as *successors* as result of the transformation. *Normalized bitstream transformations* categorized by $|V_{in}| : |V_{out}|$ cardinality are

- the concatenating transformation of multiple fragment segments into one composite segment $(m : 1)$ (for fragments),
- a class of block transformations such as decompression or decryption $(1 : 1)$ (for transcodes) and
- segmenting transformation of a structured segment into multiple separate bitstream segments $(1 : n)$ (for structures).

Arbitrary transformations of $m : n$ cardinality can be composed by concatenating two or more normalized transformations.

DEFINITION 4.5 (BITSTREAM SEGMENT GRAPH): Given a set of bitstream transformations R_T and a set of bitstream encodings R_E, then R_T and R_E induce a bitstream segment graph (BSG). It is a weakly connected, directed acyclic rooted graph $G = (V, E)$ with a set of bitstream segments V as vertices and a set of directed edges $E \subset V \times V$, connecting transformation input/output pairs of bitstream segments. A BSG describes the composition of a bitstream source and is complete iff

$$\forall v \in V\colon (\exists! t = (rel_t, V_{in}, V_{out}, P) \in R_T, v \in V_{in}) \oplus (\exists! e = (rel_e, v_e, l) \in R_E, v = v_e)$$

A BSG is composed from bitstream transformations and encodings, which are required to have bijective mapping relations. It therefore provides a bijective mapping between its bitstream source and its contained literals.

4.1.1.1. Types of Bitstream Segments

In a BSG instance, bitstream segments are categorized into one of 6 *types* as *generic*, *primitive*, *structure*, *transcode*, *fragment* and *composite*, depending on their participation in normalized bitstream transformations and bitstream encodings as shown in Table 2. While the concept of primitives, structures, transcodes and fragments have been previously defined in Section 2, a generic serves for data of yet unknown type in an incomplete description, and a composite describes the aggregation of two or more fragments.

To prevent a conflicting type assignment for bitstream segments that have both the "upward" type and another "downward" type such as a composite that contains a structure, an identity transformation is inserted after the composite and the "downward" type such as the structure is assigned to the newly inserted bitstream segment.

4.1.1.2. Coverage of Bitstream Segments

The *coverage* of a bitstream segment is a measure in the range between 0 and 1 and expresses how completely a bitstream segment is mapped to encoded literals through its successor(s). It is defined as 1 for primitives, 0 for generics, and computed as length-weighted sum over the coverage of all successors otherwise. For example, for a structure bitstream segment `a` with two primitive segments as successors, the coverage of `a` would be 1. In case of one primitive segment and a generic segment of equal length as successors, the coverage of `a` would be 0.5. The coverage of a BSG instance refers to that of its bitstream source.

4.1.2. Composition Algorithm

Using definitions 4.1 to 4.5, we are able to describe the bijective mapping between a bitstream source and its set of contained literals. The following simple algorithm constructs a BSG step-by-step. For a construction at step x, the tuple

$$(v_{Root}, V_x, V_{leaf_x}, V_{literal_x}, R_{T_x}, R_{E_x})$$

describes a designated root bitstream segment v_{Root}, a set of bitstream segments V_x, a set of leaf bitstream segments V_{leaf_x}, a set of literal bitstream segments $V_{literal_x}$, a set of bitstream transformations R_{T_x} and a set of bitstream encodings R_{E_x}, whereas initial values are

$$\begin{aligned} V_0 &= \{v_{Root}\} \\ V_{leaf_0} &= \{v_{Root}\} \\ V_{literal_0} &= \emptyset \\ R_{T_0} &= \emptyset \\ R_{E_0} &= \emptyset \end{aligned}$$

Starting at step $x = 1$, each step either adds a transformation or an encoding through an operation. For a transformation, the addition of $t = (rel, V_{in}, V_{out}, P) \notin R_{T_{x-1}}$, $V_{in} \subseteq V_{leaf_{x-1}}$ results in

$$\begin{aligned} V_x &= V_{x-1} \cup V_{out} \\ V_{leaf_x} &= V_{leaf_{x-1}} \cup V_{out} \setminus V_{in} \\ V_{literal_x} &= V_{literal_{x-1}} \\ R_{T_x} &= R_{T_{x-1}} \cup \{t\} \\ R_{E_x} &= R_{E_{x-1}} \end{aligned}$$

whereas the addition of an encoding $e = (rel, v, l) \notin R_{E_{x-1}}, v \in V_{leaf_{x-1}}$ results in

$$\begin{aligned} V_x &= V_{x-1} \\ V_{leaf_x} &= V_{leaf_{x-1}} \setminus v \\ V_{literal_x} &= V_{literal_{x-1}} \cup \{l\} \\ R_{T_x} &= R_{T_{x-1}} \\ R_{E_x} &= R_{E_{x-1}} \cup \{e\} \end{aligned}$$

For step y, the tuple induces a BSG $G_y = (V_y, E_y)$ where E_y is defined as follows:

$$\begin{aligned} &\forall t = (rel, V_{in}, V_{out}, P) \in R_{T_y}, \\ &\forall v_s \in V_{in}, \forall v_t \in V_{out} : e = (v_s, v_t) \in E_y \end{aligned}$$

For a complete BSG, these steps are repeated until $V_{leaf_z} = \emptyset$, where the algorithm terminates as no further addition of either transformation or encoding to leaf bitstream segments is possible.

4.1.3. Operations

For manual annotation of a bitstream source, it is helpful to break up transformations into a number of smaller, incremental operations. For that purpose, we define a set of parameterized operations under which a BSG instance is closed. Listing these with their respective inverse in pairs, these are as follows:

- **initial_split, final_join**: Replaces a generic segment a with a structure segment b, splits a into two consecutive generic segments a_1, a_2 and adds both in that order as successors to the structure b.
- **split, join**: Under a structure a, replaces a generic segment b with two consecutive generic segments b_1, b_2 in that order which result from splitting b in two.
- **tie, untie**: Replaces a consecutive set of segments A with a structure segment b that has A as its successors.
- **declare_primitive, undeclare_primitive**: Transforms a generic segment a into a primitive segment a.
- **expand, compress**: Transforms a generic segment a into a transcode segment a and adds a generic segment b as its successor.
- **declare_fragment, undeclare_fragment**: Transforms a generic segment a into a fragment segment a.
- **compose, decompose**: Aggregates an ordered set of fragments A into a composite segment b, assigns b as sole successor to each fragment $a \in A$, and adds a generic segment c as sole successor of b.

4.1.4. Storage Representation

For exchanging a BSG instance which describes the composition of binary data, we can now define an RDF vocabulary for the Bitstream Segment Graph model. For expressing a BSG instance using RDF, bitstream segments are represented as RDF resources, belonging to certain classes and having certain properties. For storing BSG instances in RDF, Notation 3 is used [35]. In the following definitions and examples, namespaces and prefixes are used according to Table 3.

4.1.4.1. RDF Classes

Every bitstream segment has a `rdf:type` value of both `bsg:segment` and the specific RDF class corresponding to its type as listed from Table 4, such as `bsg:primitive`. A root bitstream segment additionally has a `rdf:type` of `bsg:source`. It is worth noting that the normalized bitstream transformations of segmentation, block transformation and concatenation from Definition 4.4 correspond to the classes `bsg:structure`, `bsg:transcode` and `bsg:composite`, respectively.

Prefix	Namespace	Comment
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#	Standard RDF namespace
bsg	http://www.dataformats.net/2009/01/25-bsg-syntax-ns#	BSG namespace
bsge	http://www.dataformats.net/2009/01/25-bsg-ext-ns#	BSG Extension namespace

Table 3. RDF namespace declarations

RDF Class	Description
bsg:source	Class for bitstream sources
bsg:segment	Abstract base class for bitstream segments
bsg:generic	Class for bitstream segments where the purpose is undefined
bsg:primitive	Class for bitstream segments representing an encoded literal
bsg:structure	Class for bitstream segments composed from two or more bitstream segments with separate, distinct meaning
bsg:transcode	Class for bitstream segments representing a transcoded bit sequence
bsg:fragment	Class for bitstream segments representing a fragment of a larger bit sequence with a uniform meaning
bsg:composite	Class for bitstream segments representing a bit sequence with a uniform meaning aggregated from two or more fragments

Table 4. RDF classes for bitstream segments.

RDF Class	RDF Property	Cardinality	Description
bsg:source	bsg:href	1..1	Reference to a bitstream source
bsg:segment	bsg:start	1..1	Start position in bits (inclusive)
	bsg:length	1..1	Length in bits
	bsg:end	1..1	End position in bits (exclusive)
	bsg:semantics	0..n (size of list)	Identifier for format-specific semantics
	bsg:predecessor	0..n (size of list)	Ordered list of predecessors (input)
	bsg:successor	0..n (size of list)	Ordered list of successors (output)
bsg:generic	bsg:predecessor	*0..1* (size of list)	*Restriction: Generics have at most one predecessor*
	bsg:successor	*0..0*	*Restriction: Generics do not have successors*
bsg:primitive	bsg:encoding	1..1	Identifier for the encoding used
	bsg:predecessor	*0..1* (size of list)	*Restriction: Primitives have at most one predecessor*
	bsg:successor	*0..0*	*Restriction: Primitives do not have successors*
bsg:structure	bsg:predecessor	*0..1* (size of list)	*Restriction: Structures have at most one predecessor*
	bsg:successor	*2..n* (size of list)	*Restriction: Structures have at least two successors*
bsg:transcode	bsg:codec	1..1	Identifier for the codec used
	bsg:predecessor	*0..1* (size of list)	*Restriction: Transcodes have at most one predecessor*
	bsg:successor	*1..1* (size of list)	*Restriction: Transcodes have exactly one successor*
bsg:fragment	bsg:predecessor	*1..1* (size of list)	*Restriction: Fragments have exactly one predecessor*
	bsg:successor	*1..1*	*Restriction: Fragments have exactly one successor*
bsg:composite	bsg:predecessor	*2..n* (size of list)	*Restriction: Composites have at least two predecessors*
	bsg:successor	*1..1* (size of list)	*Restriction: Composites have exactly one successor*

Table 5. RDF properties for bitstream segments.

4.1.4.2. RDF Properties

Depending on the RDF class, bitstream segments have specific properties according to Table 5. For placement, every bitstream segment has a `bsg:start`, `bsg:length` and `bsg:end` property with integer values. These refer to its exact placement within the bit sequence composed from its predecessor(s), or within its defined bit sequence in case of a bitstream source. A root bitstream segment always starts at 0. All three properties are measured in bits, whereas the start position is included and the end position excluded, which simplifies testing two bitstream segments for neighbourship.

Regarding their composition, every bitstream segment besides the bitstream source has a `bsg:predecessor` property refering to a nonempty RDF list of bitstream segment URIs. Likewise, every bitstream segment besides `bsg:generic` or `bsg:primitive` segments have a `bsg:successor` property referring to a nonempty RDF list of bitstream segment URIs. Class-specific restrictions listed in Table 5 apply which correspond to the underlying BSG model. Only the bitstream source has the `bsg:source` property.

The meaning of a bitstream segment can be assigned a `bsg:semantics` property referring to a nonempty RDF list of string literals. For example, this could refer to *PNG Signature* semantics using `png:signature` as value. For `bsg:primitive` and `bsg:transcode` bitstream segments, the identification of the actual encoding or codec used is given through the `bsg:encoding` and `bsg:codec` properties, respectively. For example, this could include an unsigned integer encoding (most significant bit first), referred to by `bsge:encoder-msbfuint` or a *gzip* transformation as `bsge:transcoder-gzip`. The normative definition of concrete identifiers for semantics, encodings and codecs depends on the data format to be described and is a standardization effort which is not within the scope of this publication.

4.1.5. Visual Representation

Depending on the type, segments in a BSG are depicted as shown in Figure 6, where *start* and *end* denote inclusive start and exclusive end bit positions relative to the parent bitstream segment(s), *type* denotes the bitstream segment type, *parameter* denotes a parameter for some types and *id* denotes some plaintext identification.

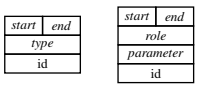


Fig. 6. Visual representations; generic, structure and composite bitstream segments (left); fragment, primitive and transcode bitstream segments (right)

4.1.6. Tool Support

We have developed the *Apeiron BSG Editor* as a tool for the annotation of bitstreams using the BSG model through a combination of graph visualization and table representation of binary data, where operations can be applied. It is written in Java, is based on the *Open Services Gateway initiative (OSGi) R4* specification and uses the *Prefuse Visualization Toolkit* [36]. Apeiron is available online and can be launched from `http://www.dataformats.net` via Java WebStart.

4.2. Data Formats

The following definition of *BSG Reasoning* originally published in [7] has been adapted to the context of this article.

4.2.1. Definition

For defining a data format, we need to define a (potentially infinite) set of data format instances which we can represent using the BSG approach. We define such a set through the *BSG Reasoning* approach as the set of stable models resulting from first-order logic rules on the BSG model, expressed as implications or biconditionals. For rules, predicates are used that refer to either deduced or computed facts. In terms of existing logic languages, the BSG Reasoning approach resembles Datalog [37] extended with functions.

4.2.1.1. Facts

A fact is represented as a predicate where all parameters are ground. For example, the start position of a bitstream segment $a1$ at bit 0 is represented as `bsg:start(a1,0)`.

4.2.1.2. Predicates

Deducible predicates refer to facts that were either given initially or subsequently deduced through rules. They are not limited to BSG-related properties and relations only, but may also include predicates for intermittent facts which may be needed for deducing a BSG instance. For deduced predicates, the open world assumption applies, as a currently unknown fact may become known later. *Computable predicates* refer to facts that can be computed directly, listed in Table 6. They handle aspects such as decoding the literal `?l` of a primitive bitstream segment `?x` from the so-far deduced, partial BSG instance through `bsg:value(?x,?l)`, or for solving the equation ?v=?u+1 through `math:sum(?u,1,?v)` if either `?u` or `?v` are known. These predicates can choose between the open world assumption and the closed world assumption, as they can decide to refute facts that will always fail, such as `math:sum(1,2,4)`.

Predicates have parameters that can either be *ground* and thus have a specific value, or be a *variable*. A *mode* of a predicate states for each of its parameters whether it is ground or variable. Computable predicate may support arbitrary modes, eg. allowing `math:sum` the computation of

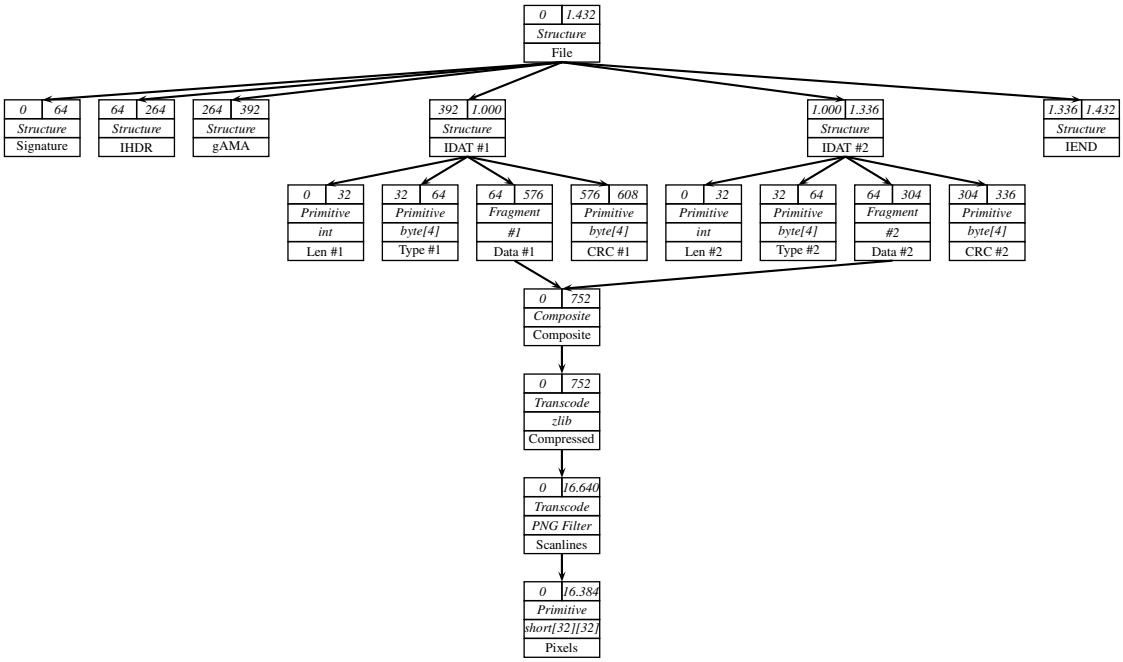


Fig. 5. Partial bitstream segment graph for file "oi2n0g16.png", showing the bijective mapping of two PNG IDAT chunks to a 16 bit grayscale image with a resolution of 32×32 pixel.

`math:sum(?u,4,5)` as well as `math:sum(1,?v,5)` and `math:sum(1,4,?w)`.

4.2.1.3. Rules

Using these types of predicates, we can build rules as implications or biconditionals. These rules can be partitioned into model-specific rules that capture properties and relations inherited from the BSG model itself, and format-specific rules that represent data format knowledge. For example, a BSG-specific rule is that two neighbouring bitstream segments `b` and `c` share a boundary, so from the facts `bsg:follows(b,c)` and `bsg:end(b,512)`, the fact `bsg:start(c,512)` follows. In Figure 5, a format-specific rule of PNG is that if a segment a represents a PNG file as stated by the fact `bsg:semantics(a,png:file)`, then its first successor b is a PNG signature, which is expressed by the facts `bsg:firstSuccessor(a,b)` and `bsg:semantics(b,png:signature)`.

4.2.1.4. Reasoning process

For deducing a BSG instance, initial knowledge on a bitstream source is given, such as the fact `bsg:source(a,'oi2n0g16.png')`. Through a series of iterative steps, the set of rules is applied in a monotone deduction process. In each step for every rule, it is tried to match the antecedents of a rule previously deduced knowledge. If the antecedents of a rule matches, then for its conclusion, the computable predicates are tested and the deducible predicates are deduced. Should a computable predicate fail in this test, the reasoning process aborts, as a conclusion does not hold. This allows the use of validation rules that assert certain properties, eg. that for all bitstream segments, its respective `bsg:start` and `bsg:length` have to sum up to its `bsg:end`, which can be violated in case of contradictory information contained in a damaged or maliciously crafted bitstream source. When no new facts are deduced in a step, then a fixed point consisting of the deducible facts of a BSG instance is reached.

If a fixed point is reached, the resulting BSG facts can then be translated into a BSG instance for that bitstream source. This requires post-processing steps such as assigning the generic bitstream segment type whenever no type was deduced for a bitstream segment. The deduction of a BSG instance therefore can either

- abort with a computable predicate refuting a fact in a rule conclusion, indicating that a conclusion does not hold and thus the bitstream source does not conform to the specified data format,
- reach a fixed point with a coverage $x < 1$, indicating that there are bitstream segments in this data format instance not specified in the set of rules, or
- reach a fixed point with a coverage $x = 1$, indicating that this data format instance is completely covered by the set of rules.

Building a set of rules as data format knowledge is typically an incremental process. It starts with the collection of bitstream sources for a corpus that represents a specific format, and the definition of an initial set of rules. This set of rules can be improved step-by-step by computing the BSG instance for every bitstream source in the corpus and computing its coverage. One then can select BSG instances with a coverage $x < 1$ and focus on generic bitstream segments which need to be described further through additional rules. Actual knowledge on how these generic bitstream segments are actually composed may come from consulting textual specifications, existing implementations or through try-and-error reverse engineering efforts. Repeating this process increases the overall coverage of BSG instances in the corpus. For a corpus, an fitting set of rules is found if the coverage reaches 1 for all of its BSG instances.

4.2.2. Implementations

In order to test the BSG Reasoning approach in practice, we have developed a suited fixed-point reasoning system in Java for BSG reasoning. We thereby defined suited interfaces for processing bitstream transformations and bitstream encodings, and implemented components for handling transformations and encodings as required for the PNG image file format.

5. EXAMPLES

5.1. Describing the composition of data automatically

In order to demonstrate the BSG Reasoning approach, we describe a small subset of the Portable Network Graphics (PNG) image format. We required that of this subset, some data format instances should at least be sufficiently complex as to require all four types of descriptive capabilities (structures, primitives, transcodes and fragments) including functional dependencies as provided by the BSG model.

5.1.1. Setup

We identified a suited subset of PNG images, namely those where compressed image data is stored as separate fragments in so-called *IDAT chunks*. For building a suited corpus, we examined the PNG Test Suite [38] with 156 PNG images for compliance testing, including corrupted files and extreme variants, and selected 8 images with filename pattern `oi??????.png`.

Regarding the granularity of description, we allowed primitive bitstream segments to represent arrays of encoded literals. Without this consideration, the decomposition of arrays such as pixel data into individual pixels would have bloated the resulting description of a data format instance without substantial benefit.

5.1.2. Data format rules

We built a fitting set of rules for our corpus, consisting of 17 model-specific rules (see Table 7) and 36 format-specific rules (see Table 8 for an excerpt).

Regarding model-specific rules, we start with rules on placement regarding a bitstream segment. This begins with a rule for deducing `bsg:start` and `bsg:length` from an initially given `bsg:source` (M1). If any two of `bsg:start`, `bsg:length` and `bsg:end` are given for a bitstream segment, the remaining fact can be deduced (M2-M4). Moreover, if all facts are given for a bitstream segment, it can be validated for ensuring consistency (M5). Further rules include aspects of neighbourship of bitstream segments in a structure (M6 & M7), successorship of bitstream segments (M8-M12), placement in a structure (M13-M15) and resolvability (M16 & M17), which is necessary for decoding the contained literal of primitive bitstream segments.

Finally, we come to format-specific rules on our PNG subset. We start with a rule that deduces the PNG-specific type of 'png:root' for a bitstream source (F1). For such a bitstream segment, we can deduce that there exists a first successor `?s` with `bsg:semantics(?s,'png:signature')` (F2). For a 'png:signature', there exists a following 'png:chunk' structure (F3) as shown in Figure 7, which again always begins with a 'png:chunk-length' bitstream segment (F4), followed by a 'png:chunk-type' bitstream segment (F5).

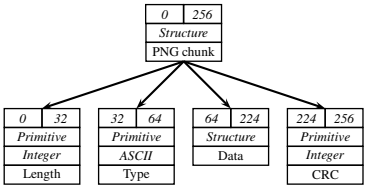


Fig. 7. BSG instance for a PNG chunk.

If the value of a 'png:chunk-length' is 0, then the 'png:chunk-type' is followed directly by the 'png:chunk-crc' bitstream segment as last successor of the chunk (F6). Otherwise, the 'png:chunk-type' bitstream segment is followed by a variable-length 'png:chunk-data' bitstream segment and again the 'png:chunk-crc' bitstream segment (F7). Details on bitstream segments such as their type, encoding and length are provided for 'png:signature' (F8), 'png:chunk-length' (F9), 'png:chunk-type' (F10) and 'png:chunk-crc' (F11) bitstream segments. The PNG-specific type of the chunk is deduced from the 'png:chunk-type' value and assigned as `bsg:semantics` to the chunk (F12). The remaining rules listed in Table 8 state that if there is space left after a chunk, there exists another one following (F13), otherwise the chunk is the last successor of the bitstream source (F14). Further rules handle chunk-specific aspects, eg. for the IHDR chunk

Predicate	Behaviour
`math:lt(?a,?b)`	Tests the formula `?a` $<$ `?b`.
`math:lte(?a,?b)`	Tests the formula `?a` $\leq$ `?b`.
`math:product(?a,?b,?c)`	Computes the formula `?a` $\cdot$ `?b` $=$ `?c` if two parameters are ground and no division by zero occurs, and assigns the result to the third variable parameter. Tests the formula if all parameters are ground.
`math:sum(?a,?b,?c)`	Computes the formula `?a` $+$ `?b` $=$ `?c` if two parameters are ground and assigns the result to the third variable parameter. Tests the formula if all parameters are ground.
`util:concat(?a,?b,?c)`	Concatenates ground strings `?a` and `?b` and binds the result to variable `?c`. Tests whether the concatenation of `?a` and `?b` corresponds to `?c` if all parameters are ground.
`util:sourceLength(?a,?b)`	Gets the length in bits of the ground file `?a` and binds it to variable `?b`. Tests whether file `?a` has length `?b` in bits if both are ground.
`util:skolem(?a,...,?c)`	*Skolem function providing for existential quantification*. Maps the set of ground parameters (`?a`, ...) to a value and binds it to variable `?c`. Maps a ground `?c` to a set of values and binds them to variables (`?a`, ...). Tests whether (`?a`, ...) and `?c` map to each other if all parameters are ground.
`util:value(?a,?b)`	Decodes the contained literal of a ground primitive bitstream segment `?a` if it is `bsg:resolved`, and assigns the result to variable `?b`. Tests whether the bitstream segment `?a` contains the literal `?b` if both parameters are ground.

Table 6. List of computable predicates.

#	Rule
M1	`bsg:source(?a,?f)` $\wedge$ `util:sourceLength(?f,?l)` $\rightarrow$ `bsg:start(?a,0)` $\wedge$ `bsg:length(?a,?l)`
M2	`bsg:length(?a,?l)` $\wedge$ `bsg:end(?a,?e)` $\wedge$ `math:sum(?s,?l,?e)` $\rightarrow$ `bsg:start(?a,?s)`
M3	`bsg:start(?a,?s)` $\wedge$ `bsg:end(?a,?e)` $\wedge$ `math:sum(?s,?l,?e)` $\rightarrow$ `bsg:length(?a,?l)`
M4	`bsg:start(?a,?s)` $\wedge$ `bsg:length(?a,?l)` $\wedge$ `math:sum(?s,?l,?e)` $\rightarrow$ `bsg:end(?a,?e)`
M5	`bsg:start(?a,?s)` $\wedge$ `bsg:length(?a,?l)` $\wedge$ `bsg:end(?a,?e)` $\rightarrow$ `math:sum(?s,?l,?e)`
M6	`bsg:leads(?a,?b)` $\leftrightarrow$ `bsg:follows(?b,?a)`
M7	`bsg:leads(?a,?b)` $\wedge$ `bsg:end(?a,?p)` $\leftrightarrow$ `bsg:follows(?b,?a)` $\wedge$ `bsg:start(?b,?p)`
M8	`bsg:firstSuccessor(?a,?b)` $\rightarrow$ `bsg:successor(?a,?b)`
M9	`bsg:lastSuccessor(?a,?b)` $\rightarrow$ `bsg:successor(?a,?b)`
M10	`bsg:successor(?a,?b)` $\rightarrow$ `bsg:predecessor(?b,?a)`
M11	`bsg:successor(?a,?b)` $\wedge$ `bsg:leads(?b,?c)` $\rightarrow$ `bsg:successor(?a,?c)`
M12	`bsg:successor(?a,?b)` $\wedge$ `bsg:follows(?b,?c)` $\rightarrow$ `bsg:successor(?a,?c)`
M13	`bsg:firstSuccessor(?a,?b)` $\rightarrow$ `bsg:start(?b,0)`
M14	`bsg:lastSuccessor(?a,?b)` $\wedge$ `bsg:length(?a,?c)` $\rightarrow$ `bsg:end(?b,?c)`
M15	`bsg:lastSuccessor(?a,?b)` $\wedge$ `bsg:end(?b,?c)` $\rightarrow$ `bsg:length(?a,?c)`
M16	`bsg:start(?a,?s)` $\wedge$ `bsg:length(?a,?l)` $\wedge$ `bsg:end(?a,?e)` $\wedge$ `bsg:type(?a,?t)` $\wedge$ `bsg:source(?a,?f)` $\rightarrow$ `bsg:resolved(?a)`
M17	`bsg:successor(?a,?b)` $\wedge$ `bsg:start(?b,?s)` $\wedge$ `bsg:type(?b,?t)` $\wedge$ `bsg:resolved(?a)` $\rightarrow$ `bsg:resolved(?b)`

Table 7. List of model-specific rules.

#	Rule
F1	bsg:source(?a,?f) → bsg:semantics(?a,'png:root')
F2	bsg:semantics(?r,'png:root') → util:skolem('F2',?r,?s) ∧ bsg:type(?r,'bsg:structure') ∧ bsg:firstSuccessor(?r,?s) ∧ bsg:semantics(?s,'png:signature')
F3	bsg:semantics(?s,'png:signature') → util:skolem('F3',?s,?f) ∧ bsg:leads(?s,?f) ∧ bsg:semantics(?f,'png:chunk')
F4	bsg:semantics(?c,'png:chunk') → util:skolem('F4',?c,?l) ∧ bsg:firstSuccessor(?c,?l) ∧ bsg:semantics(?l,'png:chunk-length')
F5	bsg:semantics(?l,'png:chunk-length') → util:skolem('F5',?l,?t) ∧ bsg:leads(?l,?t) ∧ bsg:semantics(?t,'png:chunk-type')
F6	bsg:semantics(?l,'png:chunk-length') ∧ bsg:value(?l,0) ∧ bsg:leads(?l,?t) ∧ bsg:successor(?ch,?l) → util:skolem('F6',?l,?t,?ch,?cr) ∧ bsg:lastSuccessor(?ch,?cr) ∧ bsg:leads(?t,?cr) ∧ bsg:semantics(?cr,'png:chunk-crc')
F7	bsg:semantics(?l,'png:chunk-length') ∧ bsg:value(?l,?v) ∧ math:lt(0,?v) ∧ bsg:leads(?l,?t) ∧ bsg:successor(?ch,?l) ∧ math:product(?v,8,?lv) → bsg:leads(?t,?d) ∧ bsg:leads(?d,?cr) ∧ bsg:lastSuccessor(?ch,?cr) ∧ bsg:length(?d,?lv) ∧ bsg:semantics(?d,'png:chunk-data') ∧ bsg:semantics(?cr,'png:chunk-crc')
F8	bsg:semantics(?t,'png:signature') → bsg:type(?t,'bsg:primitive') ∧ bsg:encoding(?t,'http://www.dataformats.net/2008/04/bsg-encodings# ascii-string') ∧ bsg:length(?t,64)
F9	bsg:semantics(?l,'png:chunk-length') → bsg:type(?l,'bsg:primitive') ∧ bsg:encoding(?t,'http://www.dataformats.net/2008/04/bsg-encodings# msbf-uint') ∧ bsg:length(?l,32)
F10	bsg:semantics(?t,'png:chunk-type') → bsg:type(?t,'bsg:primitive') ∧ bsg:encoding(?t,'http://www.dataformats.net/2008/04/bsg-encodings# ascii-string') ∧ bsg:length(?t,32)
F11	bsg:semantics(?cr,'png:chunk-crc') → bsg:type(?t,'bsg:primitive') ∧ bsg:encoding(?t,'http://www.dataformats.net/2008/04/bsg-encodings# msbf-uint') ∧ bsg:length(?cr,32)
F12	bsg:semantics(?ch,'png:chunk') ∧ bsg:semantics(?t,'png:chunk-type') ∧ bsg:successor(?ch,?t) ∧ bsg:value(?t,?v) → util:concat('png:chunk:',?v,?ct) ∧ bsg:semantics(?ch,?ct)
F13	bsg:semantics(?c, 'png:chunk') ∧ bsg:end(?c,?ce) ∧ bsg:successor(?r,?c) ∧ bsg:length(?r,?rl) ∧ math:lt(?ce,?rl) → util:skolem('F13',?c,?ce,?r,?rl,?nc) ∧ bsg:leads(?c,?nc) ∧ bsg:semantics(?nc,'png:chunk')
F14	bsg:semantics(?c, 'png:chunk') ∧ bsg:end(?c,?ce) ∧ bsg:successor(?r,?c) ∧ bsg:length(?r,?rl) ∧ math:eq(?ce,?rl) → bsg:lastSuccessor(?r,?c)

Table 8. Excerpt of format-specific rules for a limited PNG subset. Due to length considerations, the excerpt is limited to a set of rules capable of describing a PNG image to the level of chunk structures.

which contains information on image width and height.

5.1.3. Example deduction steps

For a given initial fact

```
bsg:source('root','oi2n0g16.png')    (13)
```

the deduction process tries to apply all rules to deduce new facts. In the first step, only the rules F1 and M1 are applicable, which yield the following new facts:

```
bsg:semantics('root','png:root')∧
bsg:start('root',0)∧
bsg:length('root',1432)              (14)
```

Again, the deduction process tries to apply all rules, this time on an increased set of facts. In step 2, the rules F2 and M4 yield the following:

```
bsg:type('root','bsg:structure')∧
bsg:firstSuccessor('root','_sc1')∧
bsg:semantics('_sc1','png:signature')∧
bsg:end('root',1432)                 (15)
```

The process of deduction is repeated until either no new facts can be deduced, or a computable predicate refutes a fact in a conclusion. The resulting facts from the reached fixed point describe a BSG instance for the PNG image oi2n0g16.png, which is part of the corpus and has a coverage of 1.0.

5.1.4. Result

After building a fitting set of rules with coverage of 1.0 for our corpus, we tested the set on all remaining PNG images from the PNG Test Suite. We obtained a coverage of 1.0 for 64 images, with the remaining 89 valid images having an average coverage of 0.79. Three corrupt images belonging to the test suite were excluded from the evaluation, as the fitting set of rules did not contain verifying rules for PNG-specific properties.

For a fitting set of rules over the entire PNG Test Suite, additional rules need to be included for palette handling (PLTE and sPLT chunks), transparency (tRNS chunk), background colour (bKGD chunk), textual data (tEXt and zTXt chunks) and other aspects. To estimate the effect of adding further rules, we added two preliminary rules for handling PLTE chunks and re-evaluated our rules on the corpus. We obtained a coverage of 1.0 for 78 images, with the remaining 75 valid images having an average coverage of 0.91.

During evaluation, the deduction process computed a fixed point and halted on all instances. Since errors may be present in a set of rules preventing a fixed point to be reached, a primitive approach on handling the Halting Problem is to place a limit on the iteration steps and abort the deduction beyond that limit. We discovered that the typical number of iterative steps required for our set of rules to reach a fixed point on valid PNG images ranges from 72 up to 170 steps. In case of the image file oi9n2c16.png, the number of iterative steps required was 3000+, as compressed image data is fragmented into bitstream segments with a length of 8 bit, each encapsuled into a separate IDAT chunk. This can be considered an extreme example, but demonstrates what is still considered legal in terms of the original specification. Since data format instances of other data formats such as Apple QuickTime movies have a more complex structure which requires an even higher number of iterations, the use of a semi-naive evaluation method for the deduction process as known from Datalog [37] is absolutely essential.

5.2. Documentation of an exploit

In this section, we give a practical example for documenting an exploit using Bitstream Segment Graphs. We have chosen the vulnerability CVE-2007-2365 [39] which utilizes a crafted PNG image to run malicious code. We use a specific version of this exploit that targets products from Adobe (Photoshop CS2, Photoshop CS3, Photoshop Elements 5) and Corel (Paint Shop Pro 11.20), but focus on the exploit section targetted for Adobe Photoshop CS2. The exploit itself was generated from an exploit generator which is available in C source code [40]. The exploit generator allows to select from two payloads, one starting the Windows Calculator, and another one binding a shell to port 4444, where we chose the former variant for annotation. For the documentation process, we consulted the PNG W3C specification [41] in addition to the crafted PNG image and the C source code of the exploit generator.

We use Apeiron introduced in Section 4.1.6 for creating a BSG instance as annotation on the crafted PNG image file. Splitting the file into its chunks according to the file specification, we arrive at the partial BSG instance shown in Figure 8. It consists of a PNG image signature and several *chunks*, which consist of a 32 bit length descriptor, a 32 bit type indicator, data of a length as indicated by the length descriptor, and a 32 bit CRC on the type and data field, if its length is non-zero.

At the current stage, the exploit contains three recognizable chunks, namely the "IHDR" chunk for describing basic information about the image, the "tIME" chunk for describing the last modification date, and the "pHYs" chunk describing the physical dimensions of pixels. The first two chunks are seemingly valid, describing an image of 509 pixel $\times$ 438 pixel with 256 indexed colors from a palette with standard values for the PNG compression, filtering and interlacing methods, which was last modified on 2007-04-15, 16:16:21 o'clock. The third "pHYs" chunk violates the PNG specification, as its length descriptor is expected to be 9 bytes, yet the ac-

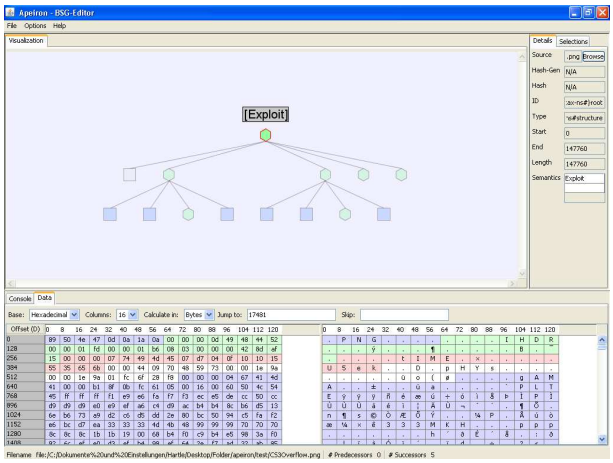


Fig. 8. Apeiron showing the crafted PNG image exploit structure with a signature, a "IHDR" chunk, a "tIME" chunk, an invalid "pYHs" chunk and junk data as its successors.

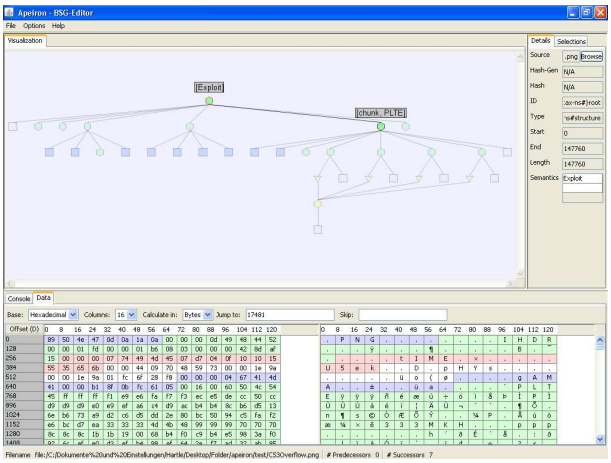


Fig. 9. Apeiron showing the crafted PNG image exploit structure as in Figure 8, adding a "gAMA" chunk and an invalid "PLTE" chunk to its successors when adjusting the "pYHs" length descriptor interpretation.

tual value is 0x4409 bytes. Taking this value for granted, the next location for a chunk contains plain invalid values, with a length descriptor value which is far beyond the actual file length (0xb67d641e), but which does not explain the actual execution of the Photoshop CS2 shellcode.

By either masking the length descriptor to the least significant byte or by assuming its specified length, we observe that the "pHYs" chunk is followed by a valid "gAMA" chunk, describing the gamma value of the purported image, a "PLTE" chunk, which is intended to describe the colors of the indexed palette for the image, shown in Figure 9. As each color is described from a red, green and blue color component each requiring 1 byte, an image with 256 colors should have a "PLTE" chunk no larger than 768 bytes, yet the actual value is 0x160060, which is well beyond the remaining file.

Interestingly, the exploit generator fragments its shellcode one byte at a time every three bytes in the "PLTE" chunk, corresponding to the red color component of the (invalid) indexed colors 1496+, which requires active reassembly through the targetted Photoshop CS2. In Figure 10, the annotation is shown for the first four bytes of the shellcode. We can therefore assume that the exploit at hand effectively overflows the buffer allocated for the red color component of indexed colors.

Another interesting aspect is that if we were to repeat the masking of the length descriptor to the least significant byte for the "PLTE" chunk as well, then the "PLTE" chunk is followed by an "IDAT" chunk, which normally would contain the filtered and compressed image data. We therefore assume that the "IDAT" chunk is actually carrying another exploit for a target application that performs the described masking of the chunk length descriptor for the invalid "pHYs" and "PLTE" chunks.

This example demonstrates the use of the Bitstream Segment Graph approach, as it allows a security expert to systemantically analyse the composition of data and document it appropriately.

6. APPLICATIONS IN IT SECURITY

6.1. Flaw Detection during Design

Just recently, the SANS and MITRE institutes performed a study in which "experts from more than 30 US and international cyber security organizations jointly released the consensus list of the 25 most dangerous programming errors" [42]. According to the consulted experts, "Improper Input Validation" was considered the most harmful.

Even though the topic is well known since the end of the 1980s [43], buffer overruns are still among the most harmful vulnerabilities. While many countermeasures have been researched and implemented, such as Java's ArrayOutOfBoundsException [44] or static and dynamic C code analysis [45], the data format itself has almost never been considered as a source of problems. Buffer overruns are usually considered to be programming mistakes, but they often result from the inherent complexity of data formats to be implemented.

At this point, BSGs can step in. For one, the systematic design process with tool-aided visualizations helps to make the structure and complexity of the data format apprehensive more easily. As well, it reduces the possibility of contradictory informations, which is often an issue in textual specifications. A self-consistent format specification eliminates one source of improper validation causing security flaws.

Second, it is hard to determine if an implementation conforms to a specification, but automatically generated parsers

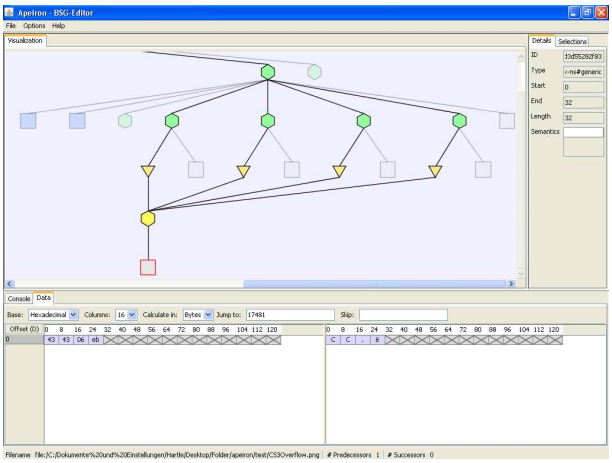


Fig. 10. Apeiron showing the start of fragmented exploit shellcode stored in the red color component of (invalid) indexed colors 1496+.

may often be infeasible or an implementation may already exist. This is where BSGs can be of help to identify extraordinary dangerous parts within a specification. To do so, we analyze a format specification regarding their level of redundancies. Redundant informations within format specifications may naturally lead to security vulnerabilities in a later implementation. Whenever there is e.g. several ways to conclude the length of a datum, a programmer might use one way to reserve memory and the second for the copy length, and a buffer overrun flaw is born. Too often, the programmer is not even aware of this change in informational context. For the programmer, the fast processing of data and an elegant code has the highest priority. By the formal description of parsing rules for BSGs with a underlying grammar, a specification designer can see and reason about the level of danger, i.e. the number of vulnerability-prone redundancies better. Hence, the specification can point out the most dangerous flaws to programmers from the start.

Third, data formats tend to be re-used in a way, other than their original objective. BSGs provide the means to analyze the behavior of nested data-formats and reason about the new situation. When a streaming-format is written to disk, for example, it might provide it's payload-size in the beginning, and the programmer would reserve memory accordingly. But due to the fact, that the filesystem provides the file's length as well, the payload can be read to EndOfFile, which may overrun reserved memory. The original format might not even have had a security relevant redundancy, but due to the nesting into a file, additional information has become available and a new vulnerability is introduced. The framework around BSGs provide a basis on which this kind of possible flaws can be analyzed in advance, but also for existing implementations.

6.2. Data Formats in Formal Security Validation

In the area of formal security validation, researchers as well as practitioners see themselves confronted with very complex computational and communication systems. Reasoning about security properties of a formal model for a full system can usually be considered infeasible. Therefore, the system model use in formal validation usually is based on an abstraction of the concrete system. Nevertheless, the completeness and soundness of the formal validation process demands a completely formally proven validation cycle requiring formalizations of these abstractions. However, most approaches for security verification neglect these abstractions and leave them out completely from their formal methodology. It is being assumed that the security expert, in order to be one, does not fail on assumptions done during the abstraction. A few however make these abstractions explicit. For example, [46], [47] and [48] present approaches on security preserving abstractions of system behavior based on alphabetic language homomorphisms. Still, the application of these formal concepts to the formalization of the abstraction of data formats is not straightforward. Ongoing research is trying to incorporate BSGs with the other to formalize the complete cycle from real to abstract systems and back in order to get evidence on the satisfaction of properties in the real system.

7. SUMMARY

In this article, we have presented the Bitstream Segment Graph and BSG Reasoning approaches for describing both data format instances and data formats. In contrast to related work, our approach is capable of handling arbitrary data format instances, providing the descriptive capabilities for structures, primitives, transcodes and fragments as well as functional dependencies. Moreover, we have shown significant limits from computational theory in our analysis, and chosen a feasible approach for modelling arbitrary data formats in general. We have presented examples of our approaches in use, describing a subset of the PNG data format as well as documenting a malicious crafted PNG image file targetting a vulnerability in Adobe Photoshop CS2. Last but not least, we have shown directions towards further applications and ongoing research regarding data format description in the area of formal security validation.

8. ACKNOWLEDGEMENTS

The authors would like to acknowledge the contribution of several students to our research and thank Arsene Botchak, Benno Kröger, Friedrich-Daniel Möller and Slaven Travar in strict alphabetic order for their master and diploma theses.

9. REFERENCES

[1] Michael Hartle, Daniel Schumann, Arsene Botchak, Erik Tews, and Max Mühlhäuser, "Describing Data Format Exploits using Bitstream Segment Graphs," in *Proceedings of The Third International Multi-Conference on Computing in the Global Information Technology (ICCGI)*, Athens, Greece, March 2008, IARIA, pp. 119–124, IEEE Press, New York, NY.

[2] *Reference Model for an Open Archival Information System (OAIS)*, vol. Blue Book, Consultative Committee for Space Data Systems, January 2002.

[3] Stephen L. Abrams and David Seaman, "Towards a Global Digital Format Registry," in *World Library and Information Congress: 69th IFLA GeneralConference and Council*, Berlin, August 2003.

[4] Adrian Brown, "File Format Registries and the PRONOM Service," in *ERPANET Seminar*, Vienna, 2003.

[5] John Marc Ockerbloom, *Mediating Among Diverse Data Formats*, Ph.D. thesis, Carnegie Mellon Computer Science, 1998.

[6] Michael Hartle, Friedrich-Daniel Möller, Slaven Travar, Benno Kröger, and Max Mühlhäuser, "Using Bitstream Segment Graphs for Complete Data Format Instance Description," in *Proceedings of The Third International Conference on Software and Data Technologies (ICSOFT)*, José Cordeiro, Boris Shishkov, Alphes Kumar Ranchordas, and Markus Helfert, Eds., Porto, Portugal, August 2008, Institute for Systems and Technologies of Information, Control and Communication, pp. 198–205.

[7] Michael Hartle, Arsene Botchak, Daniel Schumann, and Max Mühlhäuser, "A Logic-based Approach to the Formal Description of Data Formats," in *Proceedings of The Fifth International Conference on Preservation of Digital Objects (iPRES)*, London, United Kingdom, September 2008, The British Library, pp. 292–299.

[8] Wesley De Neve, Davy Van Deursen, Davy De Schrijver, Sam Lerouge, Koen De Wolf, and Rik Van de Walle, "BFlavor: A harmonized approach to media resource adaptation inspired by MPEG-21 BSDL and XFlavor," *EURASIP Signal Processing: Image Communication*, vol. 21, no. 10, pp. 862 –889, 11 2006.

[9] Davy De Schrijver, Wesley De Neve, Koen De Wolf, Robbie De Sutter, and Rik Van de Walle, "An optimized MPEG-21 BSDL framework for the adaptation of scalable bitstreams," *Journal of Visual Communication and Image Representation*, vol. 18, no. 3, pp. 217–239, 2007.

[10] Alexandros Eleftheriadis, "Flavor: a language for media representation," in *MULTIMEDIA '97: Proceedings of the fifth ACM international conference on Multimedia*, New York, NY, USA, 1997, pp. 1–9, ACM Press.

[11] Alexandros Eleftheriadis, "A Syntactic Description Language for MPEG-4," Contribution ISO/IEC JTC1/SC29/WG11 MPEG95/M0546, November 1995.

[12] O. Avaro, P. A. Chou, Alexandros Eleftheriadis, C. Herpel, C. Reader, and J. Signes, "The MPEG-4 Systems and Description Languages: A Way Ahead in Audio Visual Information Representation," *SP:IC*, vol. 9, no. 4, pp. 385–431, May 1997.

[13] Alexandros Eleftheriadis and Danny Hong, "Flavor: a formal language for audio-visual object representation," in *MULTIMEDIA '04: Proceedings of the 12th annual ACM international conference on Multimedia*, New York, NY, USA, 2004, pp. 816–819, ACM Press.

[14] Alexandros Eleftheriadis and Danny Hong, "XFlavor: Bridging bits and objects in media representation," in *Proceedings of the IEEE International Conference on Multimedia and Expo, 2002*, 2002, vol. 1, pp. 773–776.

[15] Alexandros Eleftheriadis, "The Benefits of Using MSDL-S for Syntax Description," Contribution ISO/IEC JTC1/SC29/WG11 MPEG96/M1555, November 1996.

[16] Anthony Vetro, Christan Timmerer, and Sylvain Devillers, *The MPEG-21 Book*, chapter Digital Item Adaptation - Tools for Universal Multimedia Access, pp. 243–281, John Wiley and Sons Ltd, 2006.

[17] M. Eisler, "RFC 4506: XDR: External Data Representation Standard," http://tools.ietf.org/html/rfc4506, January 2006.

[18] R. Srinivasan, "RFC 1831: RPC: Remote Procedure Call Protocol Specification Version 2," http://tools.ietf.org/html/rfc1831, August 1995.

[19] B. Callaghan, B. Pawlowski, and P. Staubach, "RFC 1813: NFS Version 3 Protocol Specification," http://tools.ietf.org/html/rfc1813, June 1995.

[20] ITU-T, "Recommendation X.680 (12/97) — Abstract Syntax Notation One (ASN.1): Specification of Basic Notation," ITU-T, Geneva, December 1997.

[21] ITU-T, "Recommendation X.691 (07/02) — ASN.1 Encoding Rules: Specification of Packed Encoding Rules (PER)," ITU-T, Geneva, July 2002.

[22] ITU-T, "Recommendation X.692 (03/02) — ASN.1 Encoding Rules: Specification of Encoding Control Notation (ECN)," ITU-T, Geneva, March 2002.

[23] ITU-T, "Recommendation X.509 (03/00) — The Directory: Public-key and attribute certificate frameworks," ITU-T, Geneva, March 2000.

[24] ITU-T, "Recommendation H.323 (06/06) — Packed-based multimedia communications systems," ITU-T, Geneva, June 2006.

[25] John Larmouth, *ASN.1 Complete*, Morgan Kaufmann, 1999.

[26] Michel Mouly, *CSN.1 Specification Version 2*, Cell & Sys, January 2002.

[27] ETSI, "Digital cellular telecommunications system (Phase 2+); Mobile radio interface signalling layer 3; General aspects (GSM 04.07 version 7.3.0 Release 1998)," December 1999.

[28] Protomatics, *The Transfer Syntax Notation One Specification*, Protomatics, Inc., 2006.

[29] James D. Myers and Alan Chappell, "Binary Format Description (BFD) Language," 2003.

[30] James D. Myers, Alan Chappell, Matthew Elder, Al Geist, and Jens Schwidder, "Re-integrating the research record," *Computing in Science and Engg.*, vol. 5, no. 3, pp. 44–50, 2003.

[31] Michael Beckerle and Alan Powell, "Data Format Description Language (DFDL) v1.0 Core Specification, Working Draft 032," http://forge.gridforum.org/sf/go/doc15262?nav=1, June 2008.

[32] John E. Hopcroft and Jeffrey D. Ullman, *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley, 1979.

[33] Michael Sipser, *Introduction to the Theory of Computation*, PWS Publishing, 1997.

[34] Charles H. Bennett, "Logical reversibility of computation," *IBM Journal of Research and Development*, vol. 17, no. 2, pp. 525–532, 1973.

[35] Tim Berners-Lee, "Notation 3," March 2006.

[36] Jeffrey Heer, Stuart K. Card, and James A. Landay, "Prefuse: A Toolkit for Interactive Information Visualization," in *Proceedings of ACM Human Factors in Computing Systems*, 2005, pp. 421–430.

[37] Jeffrey D. Ullman, *Principles of Database and Knowledge-Base Systems, Volume II*, Computer Science Press, 1989.

[38] Willem van Schaik, "PngSuite - the official set of PNG test images," December 1998, http://www.schaik.com/pngsuite/pngsuite.html, last accessed 2008-01-02.

[39] MITRE, "CVE-2007-2365," http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2007-2365, 2007.

[40] Marsu, "Photoshop CS2/CS3, Paint Shop Pro 11.20 .PNG File Buffer Overflow," http://milw0rm.com/exploits/3812, 2007.

[41] "Portable Network Graphics (PNG) Specification (Second Edition): Information technology – Computer graphics and image processing – Portable Network Graphics (PNG): Functional specification. ISO/IEC 15948:2003 (E)," November 2003.

[42] Bob Martin, "CWE/SANS TOP 25 Most Dangerous Programming Errors," http://www.sans.org/top25errors/, January 2009.

[43] Mark W. Eichin and Jon Rochlis, "With Microscope and Tweezers: An Analysis of the Internet Virus of November 1988," in *Proceedings of the 1989 IEEE Computer Society Symposium on Security and Privacy*, Oakland, Ohio, 1989, pp. 326–343, IEEE Computer Society.

[44] James Gosling, Bill Joy, Guy Steele, and Gilad Bracha, *The Java Language Specification (3rd Edition)*, Addison Wesley, 2005.

[45] David Wagner, Jeffrey S Foster, A. Eric Brewer, and Alexander Aiken, "A First Step towards Automated Detection of Buffer Overrun Vulnerabilities," in *Network and Distributed System Security Symposium*, San Diego, California, USA, February 2000, pp. 3–17.

[46] Sigrid Gürgens, Peter Ochsenschläger, and Carsten Rudolph, "Abstractions Preserving Parameter Confidentiality," in *Proceedings of the 10th European Symposium on Research in Computer Security (ESORICS) 2005*, Milan, Italy, September 2005, vol. 3679 of *Lecture Notes in Computer Science*, pp. 418–437, Springer Verlag.

[47] Peter Ochsenschläger, Jürgen Repp, and Roland Rieke, "Abstraction and composition: a verification method for co-operating systems," *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 12, pp. 447–459, 2000.

[48] Andreas Fuchs, Sigrid Gürgens, and Carsten Rudolph, "On the Security Validation of Integrated Security Solutions," in *Proceedings of IFIP Sec 2009*, 2009.

www.iariajournals.org

International Journal On Advances in Intelligent Systems
ICAS, ACHI, ICCGI, UBICOMM, ADVCOMP, CENTRIC, GEOProcessing, SEMAPRO, BIOSYSCOM, BIOINFO, BIOTECHNO, FUTURE COMPUTING, SERVICE COMPUTATION, COGNITIVE, ADAPTIVE, CONTENT, PATTERNS
issn: 1942-2679

International Journal On Advances in Internet Technology
ICDS, ICIW, CTRQ, UBICOMM, ICSNC, AFIN, INTERNET, AP2PS, EMERGING
issn: 1942-2652

International Journal On Advances in Life Sciences
eTELEMED, eKNOW, eL&mL, BIODIV, BIOENVIRONMENT, BIOGREEN, BIOSYSCOM, BIOINFO, BIOTECHNO
issn: 1942-2660

International Journal On Advances in Networks and Services
ICN, ICNS, ICIW, ICWMC, SENSORCOMM, MESH, CENTRIC, MMEDIA, SERVICE COMPUTATION
issn: 1942-2644

International Journal On Advances in Security
ICQNM, SECURWARE, MESH, DEPEND, INTERNET, CYBERLAWS
issn: 1942-2636

International Journal On Advances in Software
ICSEA, ICCGI, ADVCOMP, GEOProcessing, DBKDA, INTENSIVE, VALID, SIMUL, FUTURE COMPUTING, SERVICE COMPUTATION, COGNITIVE, ADAPTIVE, CONTENT, PATTERNS
issn: 1942-2628

International Journal On Advances in Systems and Measurements
ICQNM, ICONS, ICIMP, SENSORCOMM, CENICS, VALID, SIMUL
issn: 1942-261x

International Journal On Advances in Telecommunications
AICT, ICDT, ICWMC, ICSNC, CTRQ, SPACOMM, MMEDIA
issn: 1942-2601