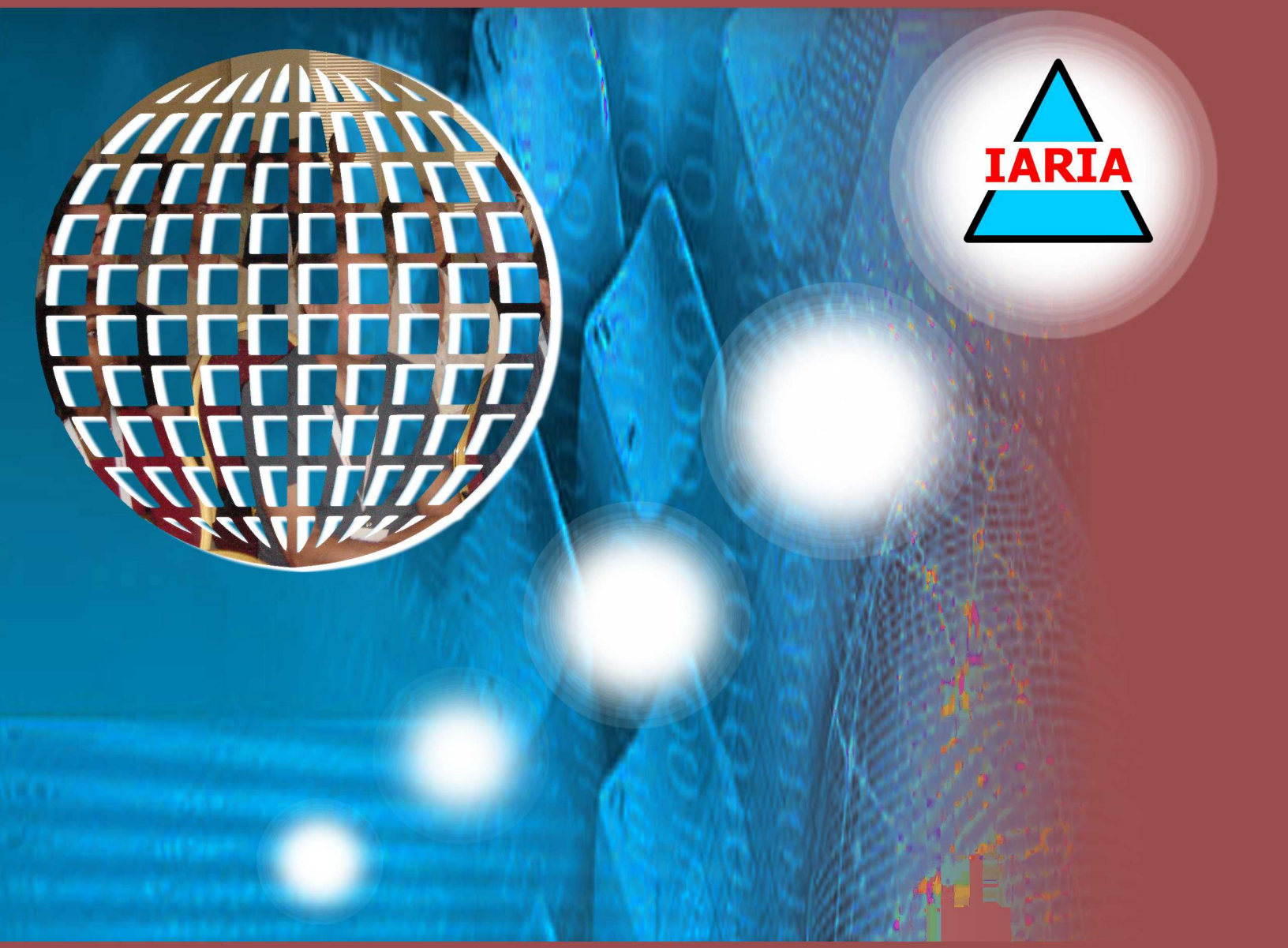
International Journal on Advances in Security



2012 vol. 5 nr. 3&4

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 5, Number 3 & 4, 2012

Shao-Ming Fei, Capital Normal University, Beijing, China
Eduardo B. Fernandez, Florida Atlantic University, USA
Anders Fongen, Norwegian Defense Research Establishment, Norway
Somchart Fugkeaw, Thai Digital ID Co., Ltd., Thailand
Steven Furnell, University of Plymouth, UK
Clemente Galdi, Universita' di Napoli "Federico II", Italy
Emiliano Garcia-Palacios, ECIT Institute at Queens University Belfast - Belfast, UK
Marco Genovese, Italian Metrological Institute (INRIM) -Torino, Italy
Birgit F. S. Gersbeck-Schierholz, Leibniz Universität Hannover, Certification Authority University of Hannover (UH-CA), Germany
Manuel Gil Pérez, University of Murcia, Spain
Karl M. Goeschka, Vienna University of Technology, Austria
Stefanos Gritzalis, University of the Aegean, Greece
Michael Grottke, University of Erlangen-Nuremberg, Germany
Ehud Gudes, Ben-Gurion University - Beer-Sheva, Israel
Indira R. Guzman, Trident University International, USA
Huong Ha, University of Newcastle, Singapore
Petr Hanáček, Brno University of Technology, Czech Republic
Gerhard Hancke, Royal Holloway / University of London, UK
Sami Harari, Institut des Sciences de l'Ingénieur de Toulon et du Var / Université du Sud Toulon Var, France
Dan Harkins, Aruba Networks, Inc., USA
Ragib Hasan, University of Alabama at Birmingham, USA
Masahito Hayashi, Nagoya University, Japan
Michael Hobbs, Deakin University, Australia
Neminath Hubballi, Infosys Labs Bangalore, India
Mariusz Jakubowski, Microsoft Research, USA
Ángel Jesús Varela Vaca, University of Seville, Spain
Ravi Jhawar, Università degli Studi di Milano, Italy
Dan Jiang, Philips Research Asia Shanghai, China
Georgios Kambourakis, University of the Aegean, Greece
Florian Kammueller, Middlesex University - London, UK
Sokratis K. Katsikas, University of Piraeus, Greece
Seah Boon Keong, MIMOS Berhad, Malaysia
Sylvia Kierkegaard, IAITL-International Association of IT Lawyers, Denmark
Marc-Olivier Killijian, LAAS-CNRS, France
Hyunsung Kim, Kyungil University, Korea
Ah-Lian Kor, Leeds Metropolitan University, UK
Evangelos Kranakis, Carleton University - Ottawa, Canada
Lam-for Kwok, City University of Hong Kong, Hong Kong
Jean-Francois Lalande, ENSI de Bourges, France
Gyungho Lee, Korea University, South Korea
Clement Leung, Hong Kong Baptist University, Kowloon, Hong Kong
Diego Liberati, Italian National Research Council, Italy
Giovanni Livraga, Università degli Studi di Milano, Italy
Gui Lu Long, Tsinghua University, China
Jia-Ning Luo, Ming Chuan University, Taiwan

Thomas Margoni, University of Western Ontario, Canada
Rivalino Matias Jr ., Federal University of Uberlandia, Brazil
Manuel Mazzara, UNU-IIST, Macau / Newcastle University, UK
Carla Merkle Westphall, Federal University of Santa Catarina (UFSC), Brazil
Ajaz H. Mir, National Institute of Technology, Srinagar, India
Jose Manuel Moya, Technical University of Madrid, Spain
Leonardo Mostarda, Middlesex University, UK
Jogesh K. Muppala, The Hong Kong University of Science and Technology, Hong Kong
Syed Naqvi, CETIC (Centre d'Excellence en Technologies de l'Information et de la Communication),Belgium
Sarmistha Neogy, Jadavpur University, India
Mats Neovius, Åbo Akademi University, Finland
Jason R.C. Nurse, University of Oxford, UK
Peter Parycek, Donau-Universität Krems, Austria
Konstantinos Patsakis, Rovira i Virgili University, Spain
João Paulo Barraca, University of Aveiro, Portugal
Sergio Pozo Hidalgo, University of Seville, Spain
Vladimir Privman, Clarkson University, USA
Yong Man Ro, KAIST (Korea advanced Institute of Science and Technology), Korea
Rodrigo Roman Castro, Institute for Infocomm Research (Member of A*STAR), Singapore
Heiko Roßnagel, Fraunhofer Institute for Industrial Engineering IAO, Germany
Claus-Peter Rückemann, Leibniz Universität Hannover / Westfälische Wilhelms-Universität Münster / North-German Supercomputing Alliance, Germany
Antonio Ruiz Martinez, University of Murcia, Spain
Paul Sant, University of Bedfordshire, UK
Reijo Savola, VTT Technical Research Centre of Finland, Finland
Peter Schartner, University of Klagenfurt, Austria
Alireza Shameli Sendi, Ecole Polytechnique de Montreal, Canada
Dimitrios Serpanos, Univ. of Patras and ISI/RC ATHENA, Greece
Pedro Sousa, University of Minho, Portugal
George Spanoudakis, City University London, UK
Lars Strand, Nofas, Norway
Young-Joo Suh, Pohang University of Science and Technology (POSTECH), Korea
Jani Suomalainen, VTT Technical Research Centre of Finland, Finland
Enrico Thomae, Ruhr-University Bochum, Germany
Tony Thomas, Indian Institute of Information Technology and Management - Kerala, India
Panagiotis Trimintzios, ENISA, EU
Peter Tröger, Hasso Plattner Institute, University of Potsdam, Germany
Simon Tsang, Applied Communication Sciences, USA
Marco Vallini, Politecnico di Torino, Italy
Bruno Vavala, Carnegie Mellon University, USA
Mthulisi Velempini, North-West University, South Africa
Miroslav Velev, Aries Design Automation, USA
Salvador E. Venegas-Andraca, Tecnológico de Monterrey / Texia, SA de CV, Mexico
Szu-Chi Wang, National Cheng Kung University, Tainan City, Taiwan R.O.C.
Piyi Yang, University of Shanghai for Science and Technology, P. R. China
Rong Yang, Western Kentucky University , USA

Hee Yong Youn, Sungkyunkwan University, Korea
Bruno Bogaz Zarpelao, State University of Londrina (UEL), Brazil
Wenbing Zhao, Cleveland State University, USA

International Journal on Advances in Security
Volume 5, Numbers 3 & 4, 2012

CONTENTS

Managing Timing Implications of Security Aspects in Model-Driven Development of Real-Time Embedded Systems

Mehrdad Saadatmand, Antonio Cicchetti, Mikael Sjödin
Mälardalen Real-Time Research Centre (MRTC)
Mälardalen University, Västerås, Sweden
{mehrdad.saadatmand, antonio.cicchetti, mikael.sjodin}@mdh.se

Thomas Leveque
Orange Labs
Orange, Meylan, France
thomas.leveque@orange.com

***Abstract*—Considering security as an afterthought and adding security aspects to a system late in the development process has now been realized to be an inefficient and bad approach to security. The trend is to bring security considerations as early as possible in the design of systems. This is especially critical in certain domains such as real-time embedded systems. Due to different constraints and resource limitations that these systems have, the costs and implications of security features should be carefully evaluated in order to find appropriate ones which respect the constraints of the system. Model-Driven Development (MDD) and Component-Based Development (CBD) are two software engineering disciplines which help to cope with the increasing complexity of real-time embedded systems. While CBD enables the reuse of functionality and analysis results by building systems out of already existing components, MDD helps to increase the abstraction level, perform analysis at earlier phases of development, and also promotes automatic code generation. By using these approaches and including security aspects in the design models, it becomes possible to consider security from early phases of development and also identify the implications of security features. Timing issues are one of the most important factors for successful design of real-time embedded systems. In this paper, we provide an approach using MDD and CBD methods to make it easier for system designers to include security aspects in the design of systems and identify and manage their timing implications and costs. Among different security mechanisms to satisfy security requirements, our focus in this paper is mainly on using encryption and decryption algorithms and consideration of their timing costs to design secure systems.**

***Index Terms*—Real-Time Embedded Systems; Security; Model-Driven Development; Component-Based Development; Runtime Adaptation; Encryption.**

I. Introduction

To cope with the specific challenges of designing security for real-time embedded systems, appropriate design methods are required. Due to resource constraints in these systems, the implications of introducing security and its impacts on other aspects and properties of the system should be carefully identified as early as possible and the methods used for designing these systems should provide such a feature [1]. Timing properties are of utmost importance in real-time embedded systems. In this paper, we introduce an approach using Model-Driven and Component-Based Development (MDD & CBD) methods for designing secure embedded systems to bring security aspects into early phases of the development and take into account their timing costs and implications.

This work provides an implementation and a methodology for the generic idea that we discussed in [1] and extends it with the result of our works in [2], [3]. In this work we provide a more complete approach and methodology, compared to the two aforementioned works, based on their combination and synergy and discuss how this approach can cover more issue regarding the timing implications of security mechanisms in real-time embedded systems.

The approach basically works by identifying and annotating sensitive data in the component model of the system, and then deriving automatically a new component model which includes necessary security components for the protection of the data. Our main focus in this paper will be on using encryption and decryption algorithms as security mechanisms. The derivation of the new component model is based on a set of pre-defined strategies. Each strategy defines a different set of possible encryption and decryption algorithms to be used as the implementation of the security components. In this approach, since the derived component model conforms to the original meta model, the same timing analysis and synthesis as for the original component model can be used and applied for the derived one.

With the increasing role of computer systems in our daily lives, there is hardly any software product developed these days that does not have to deal with security aspects and protect itself from malicious adversaries [4]. Also with the exponentially growing number of connected and networked devices and more integration between different tools and services that store and exchange different types of data, not only new types of attacks are constantly emerging but also the risks and consequences of security breaches have become more drastic. Even some simple software products and applications which do not store any sensitive information and therefore may not seem to need to care about security aspects can, for example, be the target of buffer overflow attacks [5] and thus help attackers in gaining access to a system. All these

points emphasize that security aspects cannot be taken into account just as an afterthought and added feature to an already developed system [6], but instead should be considered at different phases of development from early phases such as requirements engineering to deployment [4]. What is needed is that instead of adding security features in an "eggshell approach", security should be designed intrinsically and inseparable from the application to be able to address the threats that target the application itself [6].

Considering security from early phases of development is especially critical in the design of real-time embedded systems. These systems typically have limited amount of resources (e.g., in terms of available memory, CPU and computational power, energy and battery) and therefore, implications of security features should also be taken into account. This is basically because of the fact that Non-Functional Requirements (NFRs), such as security, are not independent and cannot be considered in isolation and satisfying one can affect the satisfaction of others [7]. Therefore, costs and implications of security features should be identified to analyze the trade-offs and establish balance among different non-functional requirements of the system. Such costs can be in the form of impacts on timing, schedulability and responsiveness of the system, as well as memory usage, energy consumption, etc. In real-time embedded systems, satisfaction of timing requirements is critical for the successful behavior of the system, therefore, choice of security mechanisms should be done considering their timing characteristics and impacts.

Model-driven development is a promising approach to cope with the design complexity of real-time embedded systems. It helps to raise the abstraction level, enables analysis at earlier phases of development and automatic generation of code [8], [9]. Component-based development, on the other hand, is another discipline in software engineering and a software development method in which systems are built out of already existing components as opposed to building them from scratch [10], [11]. In other words, it promotes developing a system as an assembly of components by reusing already existing software units (components). Model-driven development and component-based development approaches can be used orthogonally to complement and reinforce each other to alleviate the design complexity of real-time embedded systems [10].

In this context, including security aspects in design models helps with achieving the two goals mentioned so far: bringing security aspects into earlier phases of development and enabling analysis of security implications. Moreover, model-based security analysis (not the focus of this paper) in order to identify possible violations of security requirements [12] becomes possible and also system designers with lower levels of expertise and knowledge in security domain can also include and express security concerns [2]. The latter is due to the fact that code level implementation of security features requires detailed security knowledge and expertise, while at the model level, system designers can use modeling concepts and annotations for expressing security concerns (which in turn may also be used for automatic generation of security implementations).

By constructing the model of the system including security features, timing analysis can then be done on the model to evaluate whether the model meets the timing requirements or not. If so, the implementation of the system can then be generated from the model(s). This leads to a fixed set of security mechanisms that are already analyzed as part of the whole system in terms of their timing behaviors and are thus known to operate within the timing constraints of the system. However, there are situations where such a guarantee in terms of timing behaviors cannot be achieved. For example, in performing analysis some assumptions are taken into account, such as worst-case execution times of tasks. If these assumptions are violated at runtime, the analysis results will not hold anymore. Moreover, in complex real-time systems where timing analysis is not practical/economical or not much information about the timing characteristics of each individual task is available, other approaches are needed in order to tackle the timing issues [13]. One solution is to have runtime adaptation to mitigate timing violations and keep the execution of tasks within their allowed time budgets.

The remainder of the paper is structured as follows. In Section II, we discuss the issue of security and its challenges in embedded systems in general. In Section III, the automatic payment system is described as the motivating example of this paper and also as an example of distributed real-time embedded systems with security requirements. The suggested approach and its implementation are described in Sections IV and V. In Section VI, we introduce a runtime adaptation mechanism to mitigate the violations of timing constraints at runtime. Practical aspects of the introduced approach and other related issued are discussed in Section VII. Section VIII discusses the related work and finally in Section IX conclusions are made.

II. Security in Embedded Systems

In the design of embedded systems, security aspects have often been neglected [14]. However, the use of embedded systems in critical applications such as aviation systems, controlling power plants, vehicular systems control, and medical devices makes security considerations even more important. This is due to the fact that there is now a tighter relationship between safety and security in these systems (refer to [15] for the definitions of security and safety and their differences).

Also because of the operational environment of embedded systems, they are prone to specific types of security attacks that might be less relevant for other systems such as a database server inside a bank which is physically isolated and protected, in contrast to smart cards and wireless sensor networks which are physically exposed. Physical and side channel attacks [16] are examples of these types of security issues in embedded systems that bring along with themselves requirements on hardware design and for making systems tamper-resistant. Examples of side channels attack could be the use of time

and power measurements and analysis to determine security keys and types of used security algorithms.

Increase in the use and development of networked and connected embedded devices also opens them up to new types of security issues. Features and devices in a car that communicate with other cars (e.g., the car in front) or traffic data centers to gather traffic information of roads and streets, use of mobile phones beyond just making phone calls and for purposes such as buying credits, paying bills, and transferring files (e.g. pictures, music,etc.) are tangible examples of such usages in a networked environment.

Besides physical and side channel attacks, often mobility and ease of access of these devices also incur additional security issues. For example, sensitive information other than user data, such as proprietary algorithms of companies, operating systems and firmwares, are also carried around with these devices and need protection.

Because of the constraints and resource limitations in embedded systems, satisfying a non-functional requirement such as security requires careful balance and trade-off with other requirements and properties of the systems such as performance and memory usage. Therefore, introducing security brings along its own impacts on other aspects of the systems. This further emphasizes the fact that security cannot be considered as a feature that is added later to the design of a system and needs to be considered from early stages of development and along with other requirements. From this perspective, there are many studies that discuss the implications of security features in embedded systems such as [16], in which considering the characteristics of embedded systems, major impacts of security features are identified to be on the following aspects:

- Performance: Security protocols and mechanisms incur heavy computational demands on a system that the processing capacity of an embedded system might not be able to satisfy easily. For example, using encryption and decryption algorithms not only have high computational complexity but also require good amount of memory. In systems that need to handle heavy input loads, such as routers and many systems that are used in telecommunication domain to handle calls and data traffics, these security features can consume lots of processing capacity of the system and result in missed deadlines of other tasks, dropped throughput level, and overall transaction and data rate of the system.
- Power Consumption: In embedded systems with limited power sources, any resource-consuming feature impacts the operational life of the system. In this regard, security features with their heavy computational and memory demands, as discussed above, require careful considerations. There are studies that investigate this issue and compare power consumption of different encryption/decryption algorithms such as [17] that looks at this issue in wireless sensor networks. The issue of power consumption is especially interesting knowing that the growth of battery capacities are a lot slower and far behind the ever-increasing power requirements of security features. This has also led to investigating optimized security protocols for embedded systems and hardware security solutions [16].
- Flexibility and Maintainability: Flexibility of security features and possibility to adapt them according to new requirements is also a challenge in embedded systems. For example, embedded devices such as mobile phones that are used in different operational modes and environments need to support a variety of security protocols. Moreover, security solutions need to be updated in order to be protected against emerging hacking methods. Therefore, flexibility of security design decisions is important for maintaining the security of the system to apply updates and patches.
- Cost: Cost is also a limiting factor in the design of embedded systems. Considering the issues mentioned above, using a faster and more expensive CPU or adding more memory modules to cope with the demands of security requirements can add to the total cost of an embedded system. Taking into account that these devices are often produced in large amounts (e.g. mobile phones and vehicular systems), a small increase in cost can affect overall revenues and competitive potentials of a product in the market. Therefore, the security features that are implemented in embedded systems should be balanced with hardware requirements and consequently cost limits.

III. Motivation Example: Automatic Payment System

Figure 1 shows the Automatic Payment system which is an example of distributed embedded systems with real-time and security requirements. The main goal in the design of this system is to allow a smoother traffic flow and reduce waiting times at tolling stations (as well as parkings).

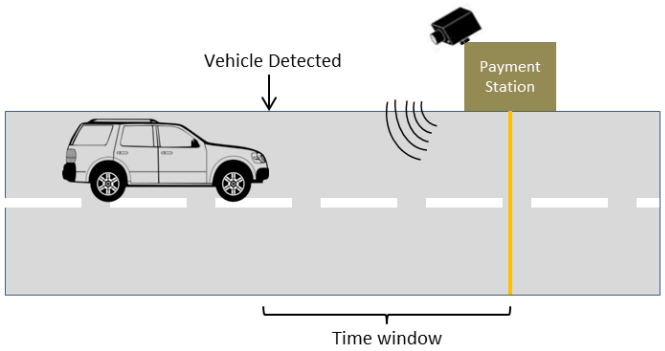


Fig. 1. Automatic Payment System for Toll Roads.

For each toll station, a camera is used to detect a vehicle that approaches the station (e.g. at 100/200 meter distance), and scans and reads its license plate information. This information is passed to the payment station subsystem which then sends the toll fee to the vehicle through a standardly defined wireless communication channel. This amount is shown to the driver in the vehicle through its User Interface (UI) and the driver

inserts a credit card and accepts the payment to be done. The credit card number is then sent securely to the payment station which then performs the transaction on it through a (third party) merchant (e.g., via a wired Internet connection at the station). The driver is then notified about the success of the transaction and receives an *OK* message to go accordingly. The interactions between different objects in this system are shown in Figure 2.

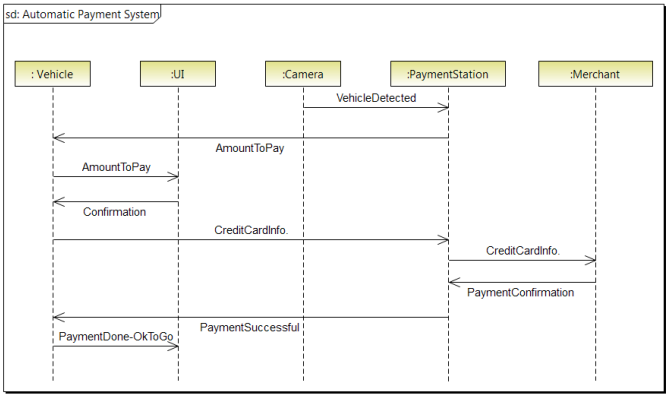


Fig. 2. Automatic Payment System.

To allow a smooth traffic flow, all these operations should be done in a certain time limit. Such time constraints can be calculated considering the specifications of camera and its required time for the detection of an approaching vehicle, traffic and safety regulations (e.g, allowed speed), and other similar factors. For example, if the vehicle is detected at 100 meter distance from the station, and the allowed speed at that point is 20 km/h, then the system has a strict time window during which it should be able to store the vehicle information, establish communication, and send the payment information to it. Different scenarios can happen in this system. For example, it could happen that the driver/vehicle fails to provide credit card information, or the credit card is expired. In this case, the system can log the vehicle information in a certain database and send the bill later to the owner, or even it can be set to not open the gate for the vehicle to pass and also show a red light for other cars approaching that toll station to stop. Besides the mentioned timing constraints that exist in this system, the communication between different nodes and transfer of data need to be secured and protected. In this system, we have the following security requirements:

1) Sensitive data such as credit card information should not be available to unauthorized parties.
2) The vehicle only accepts transactions initiated by the payment station.

To achieve these requirements, the station needs to authenticate itself to the vehicle so that the vehicle can trust and send the credit card information. Moreover, sensitive information that is transferred between different parts should also be encrypted.

Another scenario that can happen in this system is that several vehicles may approach one station with a short time distance between each which can result in bursty processing loads on the system (analogous to bursty arrivals of aperiodic tasks in real-time terms). In such situations, timing requirements may be violated as even by using static analysis of the system, only certain levels of such bursty loads may be covered and not all the possible cases. One solution to mitigate timing violations in this scenario is to introduce runtime adaptation and adapt the security level of the system at runtime; meaning that security mechanisms that are less time-consuming (and presumably less strong) can be used when such situations are detected. As the last resort, when the system realizes that many number of timing violations are occurring, to maintain a smooth traffic flow and prevent any possible accidents and safety issues due to the increasing queuing of the cars at the tolling station, instead of the on-site payment and charging of the vehicles, the system can just store their information to send a bill later to the owner of the vehicle, or even add the amount to the payment done at the next tolling stations on the road (if there are any and they are connected).

To model and build the system (software parts), particularly considering the timing constraints of the system, the following challenges are identified:

1) Modeling security mechanisms with enough details to enable both timing analysis on the model and synthesis of the security implementations,
2) Obtaining timing costs of security mechanisms,
3) Managing possible timing violations of the system at run-time.

The first challenge is discussed in the following two sections. To get the timing costs of security mechanisms, we rely on studies such as [18] that have done such measurements. To solve the third challenge, a runtime adaptation mechanism is introduced and we show how it helps to mitigate the runtime violations of timing constraints.

IV. Approach

Based on the identified challenges in the previous example, we introduce an approach that aims to bring the security concerns in the design of embedded systems. Our suggested approach helps systems designers in expressing the security concerns in a system without the need to have much security expertise on the actual implementation of security mechanisms. It does so by just requiring the system designers to identify sensitive data entities that need to be protected. In the scope of our work, it can be for example the data that need to be confidential and/or whose sender must be authenticated. Moreover, to mitigate potential timing violations of security mechanisms at runtime, the approach provides the option to include an adaptation feature for the security mechanisms.

To implement the approach, ProCom [19] component model has been used; although the approach is not dependent on this specific component model and can be implemented using other component models as well. Security needs are specified as annotations on the component model. A benefit of the ProCom component model is its power in defining new attribute types

using its attribute framework to annotate and specify new types of data. The term component model hereafter is used to basically refer to the component architecture model than the meta-model of ProCom. From the specification of the security needs at the data level and physical platform level, a model transformation is applied on the component model to derive a new component model including security implementations. The derivation of the new component model (which now has appropriate security components implementing the security needs) is done based on a selected strategy. The strategy basically specifies the preferences in terms of security implementations and which of them to choose among a set of different possible ones. Having the necessary information in the model, the steps that have been described so far can be summarized as follows:

1) The component model which specifies the functional and non-functional (extra-functional) part of the system is transformed into a functionally equivalent model with added security implementations;
2) Analysis can be performed on the derived component model that includes security components to identify any possible violations of timing constrains; and
3) Finally, the system is synthesized.

The considered process is iterative and allows to refine security specification after evaluating the resulted system properties such as timing properties. In other words, timing analysis, for example, can be performed on the derived component model and if timing properties of the derived model do not satisfy the timing requirements, the derivation process can be repeated with different preferences to finally gain a model which is satisfactory in terms of timing requirements. The process is depicted in Figure 3 showing different models and annotations that are used as input in each step (i.e., the analysis of the system model as well as synthesis of the implementation).

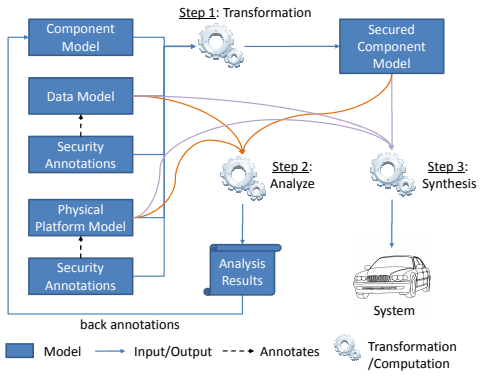


Fig. 3. General description of the approach process.

V. Implementation

A. ProCom Component Model

While the approach principles are component model generic, we implemented it using ProCom. The ProCom component model targets distributed embedded real-time system domain. In particular, it enables to deal with resource limitations and requirements on safety and timeliness concerns. ProCom is organized in two distinct layers that differ in terms of architectural style and communication paradigm. For this paper, however, we consider only the upper layer which aims to provide a high-level view of loosely coupled subsystems. This layer defines a system as a set of active, concurrent subsystems that communicate by asynchronous message passing, and are typically distributed. Figure 4 shows ProCom design of the Automatic Payment System example.

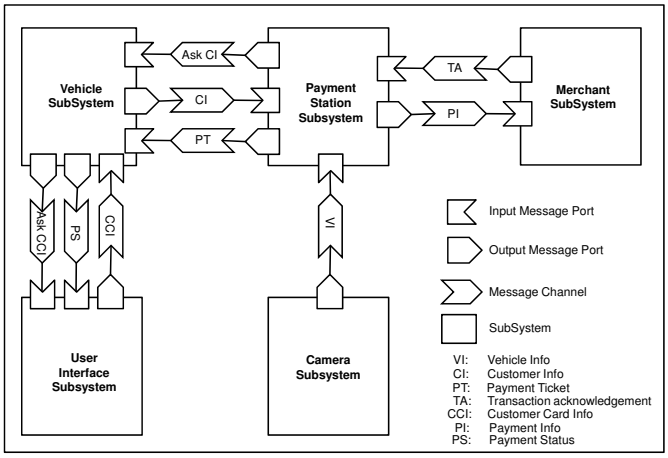


Fig. 4. Component Model of the System using ProCom.

A subsystem can internally be realized as a hierarchical composition of other subsystems or built out of entities from the lower layer of ProCom. Figure 5 shows the implementation of the subsystem E as an assembly of two component C1 and C2. Data input and output ports are denoted by small rectangles, and triangles denote trigger ports. Connections between data and trigger ports define transfer of data and control, respectively. Fork and Or connectors, depicted as small circles, specify control over the synchronization between the subcomponents.

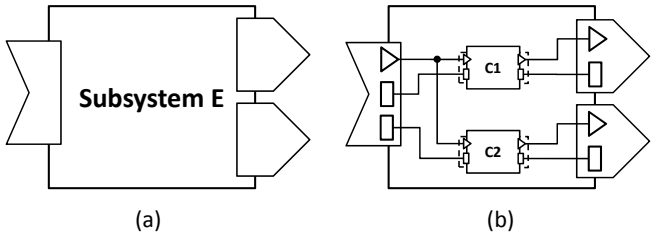


Fig. 5. ProCom SubSystem Implementation.

B. Data Model

As components are usually intended to be reused, their related data may also be reused. To this end, we propose to extend the data-entity approach described in [20] for design-time management of data in component-based real-time embedded systems. In this approach every data entity

is stored in a shared repository and designers are provided with an additional architectural view for data management, namely the data architectural view. The description of a data entity contains its type (string, int...), its maximum size and its unit. A data entity can also be a composite entity defined as a list of data entities. We use the concept of data entity to identify data that are transfered through different message channels in the system (shown in Figure 4) and map them to their respective security concerns (e.g., if they need to be encrypted and protected or not). Table I and Table II show the data entities in our example. As described in the

TABLE I
PRIMITIVE DATA ENTITIES.

Data Entity	Type	Max Size	Unit
CCNumber	String	16	byte
ExpirationDate	String	4	byte
AskCI	Empty	0	byte
AskCCI	Empty	0	byte
PaymentStatus	boolean	1	byte
VehicleNumber	String	20	byte
VehicleType	Enum	8	byte
AmountToPay	float	4	euro

TABLE II
COMPOSITE DATA ENTITIES.

Data Entity	Contains
CreditCard	CCNumber, ExpirationDate
CustomerInfo	VehicleNumber, CreditCard
PaymentTicket	AmountToPay, PaymentStatus
PaymentRequest	AmountToPay, CreditCard

last section, subsystems communicate through asynchronous message passing represented by message channels. A message channel is associated with a list of data entities which defines the message content. Table III presents the mapping between message channels and data entities for our example. We can

TABLE III
MAPPING BETWEEN DATA ENTITIES AND MESSAGE CHANNELS.

Message Channel	Data Entities
AskCI	AskCI
CI	CustomerInfo
PT	PaymentTicket
AskCCI	AskCCI
PS	PaymentStatus
CCI	CreditCard
VI	VehicleNumber, VehicleType
TA	CCNumber, AmountToPay, PaymentStatus
PI	PaymentRequest

observe that the same data entity can be used several times in different message channels. The mapping between data ports of message ports and data entities is based on naming convention which enables to distinguish between the data ports that require to encrypt/decrypt their data and those that do not. We call data model the set of data entities which are used in the related design.

C. Physical Platform And Deployment Modeling

The physical entities and their connections are described in a separate model called Physical Platform Model (see Figure 6). This model defines the different Electronic Computation Units (ECUs), called Physical Nodes, including their configurations such as processor type and frequency, the connections between the physical nodes, and the physical platforms which represent a set of ECUs fixed together.

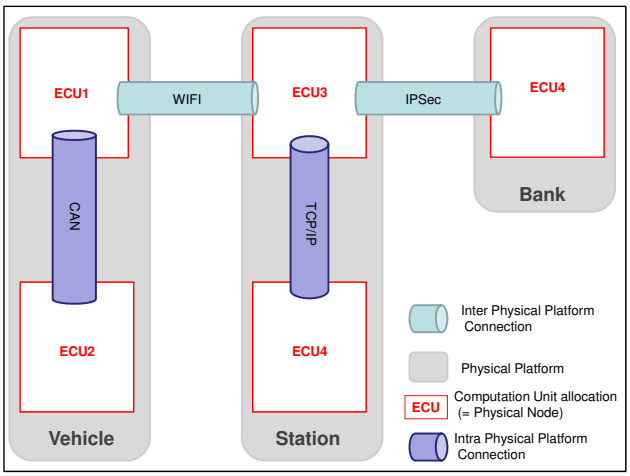


Fig. 6. Physical Platform Model of the System.

ProCom system deployment is modeled in two steps, introducing an intermediate level where subsystems are allocated to virtual nodes that, in turn, are allocated to physical nodes. In a similar way, message connections are allocated to virtual message connections which, in turn, are allocated to physical connections. Figure 7 defines the physical platform and related mapping of Automatic Payment System model. To simplify the example, we assume a one to one mapping between virtual node and physical node.

D. Security Properties

Instead of defining the security properties on the architecture, i.e. the component model, we propose to annotate the data model and compute the required security properties on the architecture, based on these security requirements. It is an original part of our approach where a designer can think about sensitive data without considering the architecture models. The designer applies security properties to identify and annotate sensitive data in the system, which require to be protected using some security mechanisms (e.g., confidentiality and encryption, authentication, integrity, etc.). We consider two types of security properties:

- **Confidentiality** ensures that the considered information cannot be read by any external person of the system; and

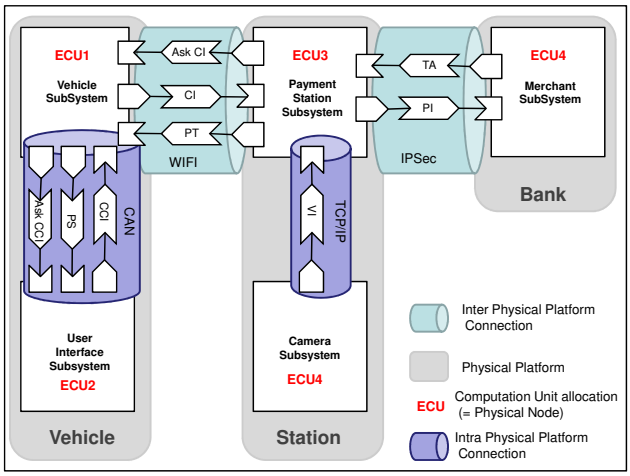


Fig. 7. Deployment Model of the System depicting allocation to Physical Platforms.

- **Authentication** which ensures that the considered information comes from the expected sender.

Table IV shows security annotations associated to data entities for our example. In addition to security properties on

TABLE IV
DATA ENTITY SECURITY PROPERTIES.

Data Entity	Security properties
CCNumber	Confidentiality
VehicleNumber	Authentication
AskCI	Authentication
AskCCI	Authentication
PaymentRequest	Authentication
PaymentStatus	Authentication

the data model, we define the security properties related to the physical platform which are independent of any application:

- **Exposed** defines that the physical platform is potentially accessible to external persons and that they may be able to open it and modify physical parts.
- **NotAccessible** defines that the physical platform is not considered as accessible to unauthorized persons.

In a similar way, physical connections are annotated:

- **Secured** defines that the physical connection is considered as secured due to its intrinsic security implementation.
- **NotSecured** defines that the physical connection protocol does not implement a reliable security (opposite of the above).

Using these properties, the person responsible for the physical platform annotates physical entities and the physical connections between them in the platform model. Thanks to these annotations, we can deduce which parts do not need additional security implementations if it is already provided (by construction). For example, if a link is established using mere TCP/IP, it is annotated as NotSecured, while in case that IPSec protocol suite is used for a link, that link is annotated as Secured. This means that the link is considered trusted and already secured, and no security component is necessary to be added for the link. Table V shows the security properties of Automatic Payment System physical platforms.

TABLE V
SECURITY PROPERTIES OF PHYSICAL ENTITIES.

Physical Platform or Connection	Security properties
Vehicle	Exposed
Station	NotAccessible
Bank	NotAccessible
WIFI	NotSecured
IPSec	Secured
TCP/IP	NotSecured
CAN	NotSecured

E. Cost of Security Implementations

Different encryption/decryption algorithms as security mechanisms can be selected to satisfy the identified security properties in the system. Considering the fact that each security mechanism in the system has its own costs in terms of timing and performance, power consumption and so on, choosing an appropriate security mechanism is critical in order to ensure the satisfaction of timing requirements of the system. For this purpose, and to take into account the timing costs of different security mechanisms, we rely on the results of studies such as [18] that have performed these cost measurements. Based on such methods, we assume the existence of such timing measurements for the platforms used in our system in the form of the Table VI. We assume that execution times can be computed knowing the target platform, algorithm, key size and data size. A timing estimation toolkit may also be provided which provides execution time estimates based on these measurements. As can be observed from the table, we also take into account and add this flexibility that some algorithms may not be supported on some platforms (marked as NS).

TABLE VI
EXECUTION TIMES AND STRENGTH RANKING OF DIFFERENT SECURITY ALGORITHMS FOR A SPECIFIC PLATFORM

Strength Rank	Algorithm	Key Size	ET-P1	ET-P2	ET-Pn
1	AES	128	NS	480	...
2	3DES	56	292	198	...
3	DES	56	835	820	...
...					

(ET-Px: Executime Time on Platform x in bytes per second, NS: Not Supported on corresponding platform)

F. Security Implementation Strategy

As mentioned previously, based on the selected strategy, a security mechanism is chosen from the table and the components implementing it are added to the component model. The user can then perform timing analysis on the derived component model to ensure that the overall timing constraints

hold and are not violated. We propose several strategies to help choosing among all possible security implementations:

- The **StrongestSecurity** strategy selects the strongest security implementation available on the platforms (taking into account that some security mechanisms, namely encryption algorithms here, may not be available and possible on a certain platform, hence selecting the strongest available one);
- The **StrongestSecurityAndLimitImplemNb** strategy selects the strongest security implementation available on the platforms while ensuring that we use as few as possible different security implementations, since each message channel can use a different encryption algorithm (finding the most common security implementation which achieves the strongest level in terms of the strength rankings);
- The **LowestExecTime** strategy selects the security implementation available on the platforms which has the lowest execution time;
- The **LowestExecTimeAndLimitImplemNb** strategy selects the lowest execution time implementation available on the platforms while ensuring that we use as few as possible different security implementations; and
- The **StrongestSecuritySchedulable** strategy selects the strongest security implementation available on the platforms where the system remains schedulable.

The selection is driven by the fact that the same algorithm must be used for the sender and receiver components which may be deployed on different platforms which in turn may not support the same algorithms.

G. Transformation

The transformation is performed in four steps:

1) First, we identify the part of a message which needs to be confidential or authenticated while considering on which communication channels they are transferred;
2) Next, we add components in charge of the encryption and decryption of the identified communication channels;
3) Then, the strategies are used to choose which encryption algorithm to use and generate the code of the added components; and
4) Finally, the Worst Case Execution Time (WCET) of added components is estimated.

The transformation aims to ensure that data decryption is performed once and only once before that data will be consumed and that data encryption is performed once and only once when a message should be sent. To illustrate the algorithm, let's consider the example in Figure 8. We assume that only data D1 needs to be confidential. The pseudo algorithm of the transformation is described in Listing 1.

Listing 1. Transformation Pseudo Algorithm

```
msgToSecure = {}
for all channels M in component model {
 P = M.allocatedPhysicalChannel;
 if ((M.getConfidentialData() <> {}) or
     (M.getAutheticatedData() <> {}) and
   (P.isNotSecured()) and
   ((P.isIntraPlatform() and
     P.sourcePort.platform.isExposed()) or
    (P.isInterPlatform()))
       add M in msgToSecure;
}

for all M in msgToSecure {
 P = M.allocatedPhysicalChannel;

 Source = M.sourcePort;
 EnD = create component
       with same ports as Source;
 if (M.getAutheticatedData() <> {})
    add one output port Digest to EnD
    add one input port Digest to Source
 EnD.inConnections = Source.inConnections;
 create connections where EnD.outPorts
    are connected to corresponding
    Source.inPorts;
 generate EnD implementation code

 Dest = M.destPort;
 DeD = create component
       with same ports as Dest;
 if (M.getAutheticatedData() <> {})
    add one output port Digest to Dest
    add one input port Digest to DeD
 DeD.outConnections = Dest.outConnections;
 create connections where Dest.outPorts
    are connected to corresponding
    DeD.inPorts;
 generate DeD implementation code
}
```

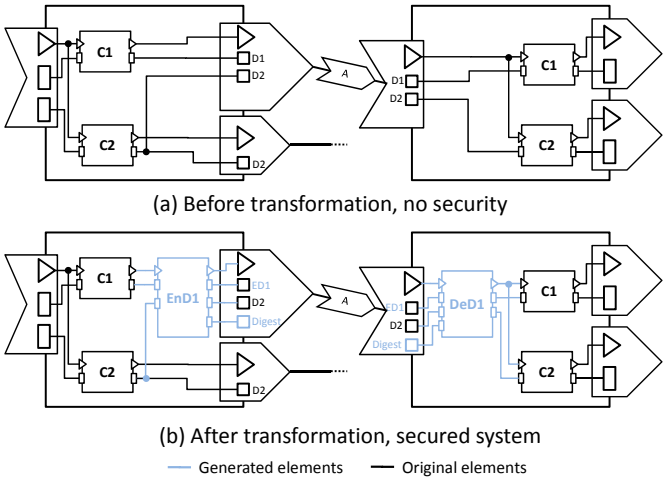


Fig. 8. Transformation.

Encryption/Decryption (in EnD1 and DeD1) is done only for confidential data while other data are just copied. An additional port is used to send the digest used for authentication. The decryption component (DeD1) ensures that all message data will be available at the same time through the output data ports. This implementation ensures the original operational semantic of the component model. Then, the security strategy is used to choose which encryption/decryption algorithm must be used and what its configuration will be.

VI. Runtime Adaptation

The suggested approach results in a *static* and fixed set of security mechanisms to be implemented and used in each invocation and use of the system. The system model including the added security components can then be analyzed in terms of timing properties before reaching the implementation phase and therefore it can be evaluated whether the timing requirements are met or not.

There are, however, cases where such static analysis may not be possible or even economical. For example, when there is not much timing information available about each task in the system to perform timing analysis, particularly in complex real-time systems with a big number of different tasks. In such systems even if enough timing information is available for each task, due to the complexity and big number of tasks, performing timing analysis may actually be not economical. Moreover, in performing static analysis some assumptions are taken into account and if those assumptions are violated at runtime then the static analysis results will not hold anymore. In such situations, a runtime adaptation mechanism can help to cope with the above challenges and mitigate timing violations by establishing balance between timing and security in a *dynamic* fashion.

To bring such adaptation mechanism into our approach, we introduce another strategy called **StrongestSecurityAdaptive**. By selecting this strategy, the implementation of added security components will be synthesized as depicted in Figure 9.

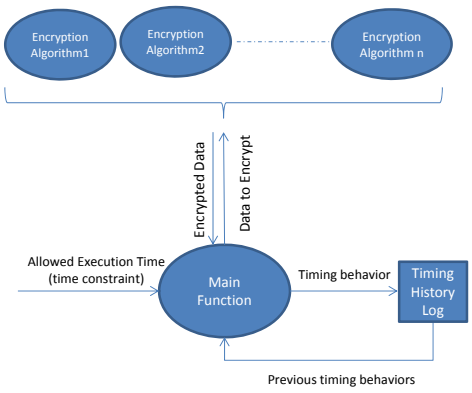


Fig. 9. Adaptation mechanism.

As shown in Figure 9, by using this strategy, in the body of the added security components (here encryption ones), the implementation of all different possible encryption algorithms will also be included. When a request for an encryption arrives, the component firstly tries to use the strongest possible encryption algorithm (based on the rank of algorithms in Table VI) to encrypt the data. The time it takes to perform the encryption is stored in the `Timing History Log`. If this time is more than the specified timing constraint for performing the job, then for the next encryption, another encryption algorithm with a lower rank but with less execution time will be selected (based on the information in Table VI).

In case the encryption job completes sooner than the specified time limit, the unused portion of its time budget is then used to determine whether it is feasible to adopt a higher ranked algorithm for the next encryption job or not. With this approach, the feedback that is produced regarding the timing behavior of encryption algorithm is used by the system to try to adapt itself. Therefore, when the system receives a burst of processing loads which it cannot fulfill under specified time constraints, it adapts itself to this higher load and similarly when the processing load decreases, it can gradually go back to using more time-consuming (and presumably more secure) encryption algorithms. This design is based on the implicit assumption that when it is detected that an executing encryption algorithm is exceeding its allowed time budget, it is basically more costly to terminate it in the middle of the encryption procedure, and restart the encryption of the data with another encryption algorithm, than just letting it finish its job, and instead use one with a lower execution time in the next invocation of encryption components.

The information that is logged in the `Timing History Log` has the following format: *Timestamp, Encryption algorithm, Time constraint, Actual execution time* (timestamp, time constraint and actual execution time are in system ticks unit in the following experiment). An example of the generated log information is shown in Table VII.

TABLE VII
Sample log information.

```
10360, AES, 50, 90
11800, 3DES, 80, 70
14500, 3DES, 60, 70
21353, DES, 60, 10
22464, 3DES, 90, 40
23112, AES, 50, 50
28374, AES, 60, 58
```

Considering the last row from the log as:

ts, alg, t, e

(ts: timestamp, alg: encryption algorithm, t: time constraint, e: actual execution time)

the decision that the system should adopt a lower ranked algorithm is made using the following formula:

(i) $e > t \Rightarrow$ *move down in the encryption algorithms table and select the next algorithm with a lower rank.*

Also, considering the two log records described as follows:

ts(l), alg(l), t(l), e(l) : representing the last log record
ts(h), alg(h), t(h), e(h) : representing the log record for the first encryption algorithm with a higher rank that was used before the last log record;

the decision to adopt a higher ranked algorithm is made using the following formula:

(ii) $e(l) < t(l) \wedge t(l) - e(l) > abs(e(h) - t(h)) \Rightarrow$ *move up in the encryption algorithms table and select the previous*

higher ranked algorithm.

A. Evaluation of the Adaptation Mechanism

In [3], we have tested the introduced adaptation mechanism; here we include the evaluation results produced during that work to demonstrate the benefits of using the adaptation approach. A simulation environment was setup as described in [3] with the use of a tool called CPU Killer [21] to enforce arbitrary CPU loads at desired times.

Figure 10 shows the evaluation results comparing performing encryption with and without using the adaptation mechanism. In each case, CPU loads of 10%, 50%, 70%, and then back to 50%, and 10% were applied.

(I) No Adaptation		(II) With Adaptation	
10%:	580,1,300,258	10:%	584,1,300,276
	859,1,300,269		868,1,300,273
	1145,1,300,276		1155,1,300,277
	1429,1,300,274		1441,1,300,276
	1711,1,300,272		1719,1,300,268
50%:	2027,1,300,305*		2111,1,300,382A
	2474,1,300,429*		2331,2,300,203
	2918,1,300,430*	50%:	2770,1,300,428*
	3364,1,300,432*		2996,2,300,211
	3813,1,300,435*		3229,2,300,214
70%:	4482,1,300,658*		3452,2,300,203
	5399,1,300,900*		3706,2,300,243
	6301,1,300,881*	70%:	4105,2,300,381*
	7199,1,300,880*		4500,3,300,380*
	8115,1,300,898*		4880,3,300,361*
50%:	8640,1,300,505*		5257,3,300,360*
	9086,1,300,429*		5639,3,300,362*
	9529,1,300,428*		5950,3,300,294A
	9974,1,300,430*	50%:	6162,3,300,194
10%:	10285,1,300,296		6380,2,300,197
	10556,1,300,261		6594,2,300,199
	10819,1,300,251		6810,2,300,200
	11083,1,300,255		7023,2,300,200
	11349,1,300,256		7236,2,300,199
			7450,2,300,199
			7641,2,300,177
		10%:	7754,2,300,103
			8026,1,300,262
			8307,1,300,270
			8572,1,300,255
			8846,1,300,264

Fig. 10. Performing encryption with and without adaptation.

The columns for each log record in Figure 10 identify: system time (ticks), encryption algorithm (AES=1, 3DES=2, DES=3) , time constraint (for each invocation; in ticks), and actual execution time (ticks). The records in which the violation of the time constraint has occurred are marked with a '*'. Comparing the two cases (without adaptation and with it) shows that the number of time constraint violations are reduced in the second case compared to the first case where only one encryption algorithm (with the highest execution time) is used. Moreover, in the second case more number of encryption jobs have been performed under a shorter period of time.

Since the goal with this adaptive strategy is to use the strongest security algorithm possible, the adaptation mechanism assumes that the encryption algorithms in Table VI are sorted according to their execution times resulting in the strongest but most time consuming one to be at the top and the weakest but less timing consuming algorithm at the bottom. Also, as a note for the decryption side, there are different ways to match and synchronize the decryption algorithm with the selected encryption algorithm. Our suggested way to do this is to add some additional bits identifying the used encryption algorithm (e.g., through the use of 2-bit or 3-bit ID numbers, according to the number of different algorithms) to the encrypted message for the decryption side to correctly pick and use the appropriate algorithm. Moreover, in the introduced adaptation mechanism and its evaluation, an encryption algorithm and the respective decryption algorithm for it have been assumed to take the same amount of time which is generally valid as mentioned in [18]. However, to extend the adaptation approach for distributed systems where encryption and decryption can be performed on different nodes, more parameters for making adaptation decisions can be added. Such an extension can be to consider the sum of encryption time and decryption time for each algorithm to make adaptation decisions instead of just considering the encryption time only.

VII. Discussion

This approach has been experimented partially in PRIDE, the ProCom development environment. The feasability at model level of the approach has been validated while the code generation part remains as future works. The security annotations have been added using the Attribute framework [22] which allows to introduce additional attribute to any model element in ProCom. The model transformation has been implemented using a QVTo [23] transformation plugged at the end of the process described in [24]. These experiments aim to show the benefits at the design level of the approach where timing properties of the overall system can be analysed. The current implementation only supports the LowestExecTime and StrongestSecurity strategies. The StrongestSecuritySchedulable strategy is hard to implement, however, it is the most interesting one. One of the reasons that we do not claim that we also provide this strategy, in spite of having the execution times of security components, is that the actual execution times in the synthesized system will not necessarily be the sum and individual addition of the execution times of the added security components to the rest of the system. More complex security implementation strategies can be considered but are not covered in this paper.

As for the synthesis of the code of the security components, in order to keep the approach generic, we intend to let certificate specification and other specific parameters of encryption algorithm to be filled in the generated code. One generator is associated for each algorithm. The suitability for timing analysis of the generated component code needs to be planned but at least will allow for measurement based timing analysis as any other ProCom component. While the system functionality remains the same, the system needs also to react to authentication errors. This problem could be partially solved by allowing developers to add code to manage authentication errors in the generated code to define what must be the output data in each specific case.

Regarding the runtime adaptation mechanism, while on one hand, it may make the job of the attackers harder as not a fixed algorithm is used in each invocation and thus it will not

be known and predictable to the attackers (hence some sort of "security through obscurity"), on the other hand, if attackers know the internal mechanism of the runtime adaptation, they can force some processing load on the system to make the system adopt the weakest algorithm possible, and that way, make it easier for themselves to break into the system. Moreover, the adaptation mechanism which was used as part of our general approach in this paper, can also be designed to act as an option; in the sense that it can be turned on and used when a processing load beyond a certain level is detected and turned off otherwise. This can help to mitigate the overhead of the adaptation mechanism itself (although another mechanism to monitor the processing load would need to be added in that case) and only use it when there are many requests for encryption.

VIII. Related Work

Designing security features for real-time embedded systems is a challenging task and requires appropriate methods and considerations. [16] and [25] particularly discuss the specific challenges of security in embedded systems and define it as a new dimension to be considered throughout the development process. Considering the unique challenges of security in embedded systems, [25] also emphasizes that new approaches to security are required to cover all aspects of embedded systems design from architecture to implementation. The methods that we introduced in this paper contribute towards this goal by applying different disciplines in the field of software engineering, such as model-driven development methods, to cope with the specific challenges of designing security for embedded systems.

Also as a non-functional requirement [7], [26], satisfying security requirements in a system has costs and implications in terms of impact on other requirements such as performance, power consumption and so on. In [17], measurement and comparison of memory usage and energy consumption of several encryption algorithms on two specific wireless sensor network platforms have been done. Performance and timing comparisons of several encryption algorithms are offered in [18] where Pentium machines are used as the platform. The approaches we proposed in this paper, work by relying on the timing and performance comparison results of encryption algorithms in such studies.

While model-driven and component-based approaches serve as promising approaches to cope with the design complexity of real-time embedded systems, management of runtime data in these systems has also become an important issues than ever before due to the growing complexity of them. This fact becomes more clear when we realize that keeping track of all data that are passing through different parts of the system is an extremely hard task for a person. In addition, most design methods based on component models focus mainly on functional structuring of the system without considering semantics and meanings for data flows [20]. A data-centric approach for modeling data as well as using real-time databases for runtime data management in real-time embedded systems is proposed in [20]. In this work, however, non-functional (extra-functional) properties such as security are not addressed, and our approach presented in this paper basically follows a similar method for modeling data entities as a basis to define security specification.

As for modeling security aspects, there are several solutions such as UMLsec [12]. UMLsec is a UML profile [27] for the specification of security relevant information in UML diagrams. It is one of the major works in this area and comes with a tool suite which provides the possibility to evaluate security requirements and their violations. SecureUML [28] is also another UML profile for modeling of role-based access controls. UML profile for Modeling and Analysis of Real-time Embedded Systems (MARTE) [29] provides semantics for modeling non-functional properties and their analysis (e.g., schedulability). In [30], we have discussed MARTE and the benefits of extending MARTE with security annotations to better cover the modeling needs of embedded systems. Besides UMLsec and its tool suite which enables analysis of security requirements, in [31], a method for specifying security requirements on UML models and verifying their satisfaction by relating model-level requirements to code-level implementation is offered. In [32], we have provided a small example how it is possible to model security requirements along with some other requirements of telecommunication systems and then perform model-based analysis using the analysis tool suite of UMLsec to identify possible violations of security requirements.

The need to identify sensitive data is also discussed in [33] where an extension to include security concerns as a separate model view for web-services based on Web-Services Business Process Execution Language (WS-BPEL) is offered. However, it does not take into account the consequences of security design decisions on timing aspects, while by identifying sensitive parts of messages which need to be secured, our objective is to ease the computation of the timing impacts of security implementations protecting those sensitive data. Considering the challenges of securing distributed systems [34] has done a survey on the application of security patterns, as a form of software design patterns, to secure distributed systems. Moreover, it discusses different methodologies that make use of these ad-hoc security patterns in a structured way. It also reports that the majority of the studied methodologies lack explicit support for distributed systems and special concerns that these systems have and mentions the development of tailored methodologies for different types of distributed systems as an important future work in this area. The approach that we suggested here could serve as an example for developing such methodologies in particular for distributed real-time and embedded systems in which timing requirements play a key role in the correctness of the whole system.

Regarding the adaptation method that we used as part of our suggested approach, there are also several related studies and approaches that we discuss them here. The study done in [35] is one of the interesting works in the area of security for real-time embedded systems which uses an adaptive method.

In this work, the main focus is on a set of periodic tasks with known real-time parameters, whereas, our main target is complex systems that can consist of any type of real-time tasks. Also, while in our work, the security level of the system is considered implicitly through the selection of algorithms from the encryption algorithms table, in [35], a QoS value has been considered which explicitly represents the security level of the system. Moreover, in our work, it is the encryption algorithms which are adaptively replaced, while the main adaptation component in that work is the key length. Our approach can easily be extended to cover not only different encryption algorithms but also variations of each, including different combinations of key length, number of rounds and so on, as items (rows) in the encryption algorithms table (e.g., AES256, AES128, etc.). Another interesting study with is close to our work is [36], which basically introduces a similar type of adaptation mechanism as ours. The main focus in that work is, however, on client-server scenarios using a database, and to manage the performance of transactions. The security manager component used int his work periodically adjusts the the security level of the system. In our approach, however, the adaptation mechanism is executed per request and is not active when there is no request for encryption. Moreover, it is possible in the approach introduced in this work that an inappropriate encryption method is used by a client, while security level change is occurring. To solve this situation, several acknowledgment messages are sent and the process is repeated to correct this issue. Therefore, it is possible that the security manager faces problems regarding synchronization and message loss due to out of order arrival of messages. As another approach for managing security in real-time systems, in [37], a secure-aware scheduler is introduced which basically incorporates and takes into account timing management of security mechanisms as part of its scheduling policy.

IX. Conclusion and Future Work

In this paper, we introduced an approach to define security specifications in real-time embedded systems at a high level of abstraction based on the benefits of model-driven and component-based methods. Using the suggested approach we bring semantics to the data that are transferred in embedded systems to identify sensitive data. The approach enables also to derive automatically the security implementations and facilitates performing timing analysis including security features at early phases of development. It was also demonstrated how incorporating a runtime adaptation mechanism as part of the approach helps to mitigate the violations of timing constraints at runtime. As mentioned, such runtime adaptation mechanisms are especially useful for complex systems where performing static analysis may not be practical, as well as in cases where the assumptions that have been used for performing static analysis are prone to deviation and violation at runtime which can then lead to the invalidation of analysis results. Moreover, the introduced approach helps system designers to mainly focus on the system architecture and addressing timing properties, and at the same, including security concerns in the design models without needing much expertise on how to implement security mechanisms. This again contributes to bringing security considerations in higher levels of abstraction.

One of the extensions of this work is to define and add more strategies for the designers to choose. Among the currently defined strategies, the StrongestSecuritySchedulable is the most interesting one but is hard to implement and will be part of our future works. One of the reasons that we do not claim that we also provide this strategy, in spite of having the execution times of security components, is that the actual execution times in the synthesized system will not necessarily be the sum and individual addition of the execution times of the added security components to the rest of the system. Also as another idea for the extension of this work, it would be interesting to define and assign required security strength to data and message channels as another factor that also affects the selection of security components. It should also be noted that in this work we mainly addressed encryption as a security mechanism. Considering other mechanisms such as authorization methods and their impacts on timing characteristics of systems is another interesting direction of this work to investigate. Also including other aspects than timing, such as power consumption of security mechanisms, performing trade-off, and establishing balance among them, similar to what we did here for timing properties, can be another extension of this paper and future work.

X. Acknowledgements

This work has been supported by Xdin Stockholm AB [38] and Swedish Knowledge Foundation (KKS) [39] through the ITS-EASY industrial research school program [40] .

References

[1] M. Saadatmand, A. Cicchetti, and M. Sjödin, "On generating security implementations from models of embedded systems," in *The Sixth International Conference on Software Engineering Advances (ICSEA 2011)*, Barcelona, Spain, October 2011.

[2] M. Saadatmand and T. Leveque, "Modeling security aspects in distributed real-time component-based embedded systems," in *Information Technology: New Generations (ITNG), 2012 Ninth International Conference on*, Las Vegas, USA, april 2012, pp. 437 –444.

[3] M. Saadatmand, A. Cicchetti, and M. Sjödin, "Design of adaptive security mechanisms for real-time embedded systems," in *Proceedings of the 4th international conference on Engineering Secure Software and Systems*, ser. ESSoS'12. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 121–134.

[4] P. T. Devanbu and S. Stubblebine, "Software engineering for security: a roadmap," in *Proceedings of the Conference on The Future of Software Engineering*, ser. ICSE '00. New York, NY, USA: ACM, 2000, pp. 227–239.

[5] F.-H. Hsu, F. Guo, and T.-c. Chiueh, "Scalable network-based buffer overflow attack detection," in *Proceedings of the 2006 ACM/IEEE symposium on Architecture for networking and communications systems*, ser. ANCS '06. New York, NY, USA: ACM, 2006, pp. 163–172.

[6] A. Main, "Application security: Building in security during the development stage," *Journal of Information Systems Security*, vol. 13, no. 2, pp. 31–37, 2004.

[7] L. M. Cysneiros and J. C. S. do Prado Leite, "Non-functional requirements: From elicitation to conceptual models," in *IEEE Transactions on Software Engineering*, vol. 30, no. 5, 2004, pp. 328–350.

[8] M. Voelter, C. Salzmann, and M. Kircher, "Model driven software development in the context of embedded component infrastructures," in *Component-Based Software Development for Embedded Systems*, ser. Lecture Notes in Computer Science, C. Atkinson, C. Bunse, H.-G. Gross, and C. Peper, Eds., vol. 3778. Springer Berlin / Heidelberg, 2005, pp. 143–163.

[9] B. Selic, "The pragmatics of model-driven development," *IEEE Software Journal*, vol. 20, pp. 19–25, September 2003.

[10] M. Törngren, D. Chen, and I. Crnkovic, "Component-based vs. model-based development: a comparison in the context of vehicular embedded systems," in *Software Engineering and Advanced Applications, 2005. 31st EUROMICRO Conference on*, aug.-3 sept. 2005, pp. 432 – 440.

[11] I. Crnkovic, "Component-based Software Engineering - New Challenges in Software Development," in *Software Focus*, vol. 2, 2001, pp. 27–33.

[12] B. Best, J. Jurjens, and B. Nuseibeh, "Model-based security engineering of distributed information systems using umlsec," in *Proceedings of the 29th international conference on Software Engineering*, ser. ICSE '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 581–590.

[13] A. Wall, J. Andersson, J. Neander, C. Norström, and M. Lembke, "Introducing temporal analyzability late in the lifecycle of complex real-time systems," in *Real-Time and Embedded Computing Systems and Applications*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2004, vol. 2968, pp. 513–528.

[14] S. Gürgens, C. Rudolph, A. Maña, and S. Nadjm-Tehrani, "Security engineering for embedded systems: the secfutur vision," in *Proceedings of the International Workshop on Security and Dependability for Resource Constrained Embedded Systems*, ser. S&D4RCES '10. New York, NY, USA: ACM, 2010.

[15] E. Albrechtsen, "Security vs safety," NTNU - Norwegian University of Science and Technology http://www.iot.ntnu.no/users/albrecht/rapporter/notat%20safety%20v%20security.pdf, Accessed: December 2012.

[16] P. Kocher, R. Lee, G. McGraw, and A. Raghunathan, "Security as a new dimension in embedded system design," in *Proceedings of the 41st annual Design Automation Conference*, ser. DAC '04, 2004, pp. 753–760, moderator-Ravi, Srivaths.

[17] J. Lee, K. Kapitanova, and S. H. Son, "The price of security in wireless sensor networks," *Journal of Computer Networks*, vol. 54, pp. 2967–2978, December 2010.

[18] A. Nadeem and M. Javed, "A performance comparison of data encryption algorithms," in *First International Conference on Information and Communication Technologies, ICICT 2005.*, 2005, pp. 84 – 89.

[19] S. Sentilles, A. Vulgarakis, T. Bures, J. Carlson, and I. Crnkovic, "A Component Model for Control-Intensive Distributed Embedded Systems," in *Proceedings of the 11th International Symposium on Component Based Software Engineering (CBSE2008)*, M. R. Chaudron and C. Szyperski, Eds. Springer Berlin, October 2008, pp. 310–317.

[20] A. Hjertström, D. Nyström, and M. Sjödin, "A data-entity approach for component-based real-time embedded systems development," in *Proceedings of the 14th IEEE international conference on Emerging technologies & factory automation*, ser. ETFA'09. Piscataway, NJ, USA: IEEE Press, 2009, pp. 170–177.

[21] CPU Killer, http://www.cpukiller.com/, Accessed: December 2012.

[22] S. Sentilles, P. Štěpán, J. Carlson, and I. Crnković, "Integration of Extra-Functional Properties in Component Models," in *12th International Symposium on Component Based Software Engineering*. Springer, 2009.

[23] I. Kurtev, "State of the art of QVT: A model transformation language standard," in *Applications of Graph Transformations with Industrial Relevance*, ser. Lecture Notes in Computer Science. Springer Berlin, 2008, vol. 5088, pp. 377–393.

[24] T. Leveque, J. Carlson, S. Sentilles, and E. Borde, "Flexible semantic-preserving flattening of hierarchical component models," in *37th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE Computer Society, August 2011.

[25] S. Ravi, A. Raghunathan, P. Kocher, and S. Hattangady, "Security in embedded systems: Design challenges," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 3, pp. 461–491, August 2004. [Online]. Available: http://doi.acm.org/10.1145/1015047.1015049

[26] M. Saadatmand, A. Cicchetti, and M. Sjödin, "Toward model-based trade-off analysis of non-functional requirements," in *38th Euromicro Conference on Software Engineering and Advanced Applications(SEAA)*, September 2012.

[27] B. Selic, "A systematic approach to domain-specific language design using uml," in *Object and Component-Oriented Real-Time Distributed Computing, 2007. ISORC '07. 10th IEEE International Symposium on*, may 2007, pp. 2 –9.

[28] T. Lodderstedt, D. A. Basin, and J. Doser, "SecureUML: A UML-Based Modeling Language for Model-Driven Security," in *Proceedings of the 5th International Conference on The Unified Modeling Language*, ser. UML '02. London, UK: Springer-Verlag, 2002, pp. 426–441.

[29] MARTE specification version 1.1, http://www.omgmarte.org, Accessed: December 2012.

[30] M. Saadatmand, A. Cicchetti, and M. Sjödin, "On the need for extending marte with security concepts," in *International Workshop on Model Based Engineering for Embedded Systems Design (M-BED 2011)*, March 2011.

[31] J. Lloyd and J. Jürjens, "Security analysis of a biometric authentication system using umlsec and jml," in *Proceedings of the 12th International Conference on Model Driven Engineering Languages and Systems*, ser. MODELS '09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 77–91.

[32] M. Saadatmand, A. Cicchetti, and M. Sjödin, "Uml-based modeling of non-functional requirements in telecommunication systems," in *The Sixth International Conference on Software Engineering Advances (ICSEA 2011)*, October 2011.

[33] M. Jensen and S. Feja, "A security modeling approach for web-service-based business processes," in *Proceedings of the 2009 16th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems*, ser. ECBS '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 340–347.

[34] "Securing distributed systems using patterns: A survey," *Computers & Security Journal*, vol. 31, no. 5, pp. 681 – 703, 2012.

[35] K.-D. Kang and S. H. Son, "Towards security and qos optimization in real-time embedded systems," in *SIGBED Rev.*, vol. 3. New York, NY, USA: ACM, January 2006, pp. 29–34.

[36] S. H. Son, R. Zimmerman, and J. Hansson, "An adaptable security manager for real-time transactions," in *Proceedings of the 12th Euromicro conference on Real-time systems*, ser. Euromicro-RTS'00. Washington, DC, USA: IEEE Computer Society, 2000, pp. 63–70.

[37] T. Xie and X. Qin, "Scheduling security-critical real-time applications on clusters," in *IEEE Transactions on Computers*, vol. 55, no. 7, july 2006, pp. 864 – 879.

[38] Xdin AB, http://xdin.com/, Accessed: December 2012.

[39] KK-stiftelsen: Swedish Knowledge Foundation, http://www.kk-stiftelsen.org/SitePages/Startsida.aspx, Accessed: December 2012.

[40] ITS-EASY post graduate industrial research school for embedded software and systems, http://www.mrtc.mdh.se/projects/itseasy/, Accessed: December 2012.

Preventing Protocol Switching Covert Channels

Steffen Wendzel[1,2] and Jörg Keller[1]
[1]*Faculty of Mathematics and Computer Science, University of Hagen, Germany*
[2]*Department of Computer Science, Augsburg University of Applied Sciences, Germany*
steffen.wendzel@hs-augsburg.de, joerg.keller@fernuni-hagen.de

***Abstract*—Network covert channels enable a policy-breaking network communication (e.g., within botnets). Within the last years, new covert channel techniques arose which are based on the capability of protocol switching. Such protocol switching covert channels operate within overlay networks and can (as a special case) contain their own internal control protocols. We present the first approach to effectively limit the bitrate of such covert channels by introducing a new active warden. We present a calculation method for the maximum usable bitrate of these channels in case the active warden is used. We discuss implementation details of the active warden and discuss results from experiments that indicate the usability in practice. Additionally, we present means to enhance the practical application of our active warden by applying a formal grammar-based whitelisting and by proposing the combination of a previously developed detection technique in combination with our presented approach.**

Keywords-Protocol Switching Covert Channel; Protocol Channel; Active Warden; Covert Channel Detection; Network Security

I. INTRODUCTION

This work is an extended version of [1] in which we introduced the design and implementation of an active warden countering protocol switching covert channels in a basic version with a constant delay.

The term *covert channel* refers to a type of channel defined as *not intended for information transfer* by Lampson in 1973 [2]. The goal of using covert channels is to transfer information without raising attention while breaking a security policy [3]. Covert channels have been a focus of research for decades. In addition to Lampson's work, the topic was described in [4] and [5]. While covert channels can also occur on local systems, we focus on covert channels within computer networks.

Covert channels are basically divided into two classes: covert storage channels and covert timing channels [4]. To transfer hidden information, a covert storage channel alters storage attributes (e.g., values of a network packet's header) and a timing channel alters timing behavior (e.g., the timing of network packets) [6].

A well-known technology to counter covert channels is the active warden, i.e., a system counteracting a covert channel communication. While passive wardens monitor and report events (e.g., for intrusion detection), active wardens (e.g., traffic normalizers [7]) are capable of modifying network traffic [8] to prevent steganographic information transfer.

Recently, the capability to keep a low profile resulted in a raising importance of network covert channels because of their use cases. For instance, covert channels can be used to control botnets in a hidden way [9]. Covert channel techniques can also be used by journalists to transfer illicit information, i.e., they can generally contribute to the free expression of opinions [6].

A first covert channel able to switch a network protocol based on a user's command called *LOKI2* was presented in 1997 [10]. Within the last decade, different new covert channel techniques occurred and not all of them were already addressed by protection means. These new techniques enable covert channels to switch their communication protocol automatically and transparently. They are enabled to cooperate in overlay networks by using internal control protocols as presented in [11]. Since covert channels are a dual-use good, these novel techniques do also enrich the security of botnets.

We focus on two new covert channel techniques, *protocol hopping covert channels* (PHCC) as well as called *protocol channels* (PCs). Both channels build the family of protocol switching covert channels since they rely on the capability to switch their utilized network protocols.

PHCC were presented in [12] and were improved in [11]. These channels transfer hidden information using several network protocols to raise as little attention as possible due to peculiar protocol behavior, i.e., they combine several single-protocol network storage covert channels. For instance, a simple PHCC could use the "User-Agent" field in HTTP as well as the message number in POP3 "RETR" (retrieve) requests to transfer hidden information. Protocol switches in a PHCC are transparent for the covert channel's user and the user utilizes the channel as a black-box that handles the splitting of the payload in data chunks sent using the different protocols. To work properly, the PHCC must be able to preserve the order of network packets across the different protocols.

PCs were introduced in [13] and signal information solely by transferring network protocols of a pre-defined set in an order that represents hidden information. Protocols are therefore linked to secret values, e.g., a HTTP packet could represent the value "1" and a POP3 packet could represent

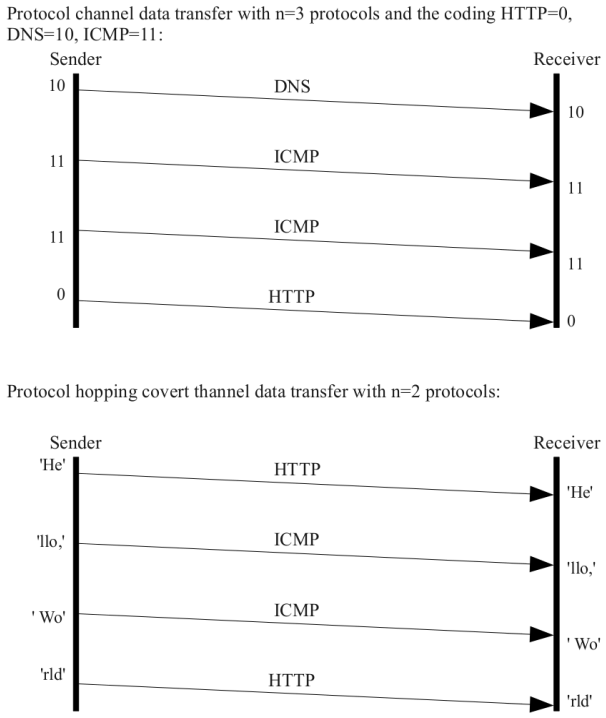


Figure 1. Comparison of a PC and a PHCC.

the value "0". To transfer the message "110" via this PC, the sender is required to send two HTTP packets followed by a POP3 packet. The bitrate of a PC is usually limited to a few hundred bit/s. However, for an attacker this is fast enough to transfer passwords, selected records or tweets.

Figure 1 compares both, the PC and the PHCC concept. The difference between both channels is, that PCs signal their hidden information solely by the use of a selected network protocol in a network packet, while PHCCs combine multiple covert storage channels simultaneously and place hidden information in arbitrary storage locations of these channels.

We present the first concept as well as an implementation of an active warden able to significantly reduce the bitrate of both, PHCC and PC. While we do not present a universal solution countering all covert channels, this is the first work discussing means against PHCC and PC. The limitation of these advanced covert channels is more challenging in comparison to single-protocol covert channels. We evaluate our results via realistic experiments and we demonstrate that our approach is useful for the practical operation in organizations. Due to these achievements, our active warden decreases the attractiveness of both channel types for attackers.

The remainder of this paper is structured as follows. Section II provides an overview of related work in the area of covert channels. Section III introduces the idea and theory of our active warden, while Section IV discusses our implementation and our experimental results, and Section V focuses on the practical aspects of the presented approach and discusses improvements for covert channels to bypass the active warden. We propose improvements for the practical use of our approach by applying a formal grammar in Section VI and by combining our active warden with a previously developed detection capability for PCs in Section VII. We conclude in Section VIII.

II. Related Work

Various techniques for embedding covert channels in network packet data and its timing were developed within the last decades. For instance, Girling and Wolf were the first authors to create network covert channels by modifying LAN frames [14], [15] and Rowland initially presented covert channels for IP and TCP [16]. Rutkowska discovered the idea of a passive covert channel that does not actively generate own traffic but piggibacks regular traffic of a system's users by modifying the TCP Initial Sequence Number (ISN) [17]. Therefore, Rutkowska introduced a translation layer for ISN values into the Linux kernel. Cabuk et al. developed a technique to embedd hidden information in the timing of network packets [18], while Ahsan modified the order of network packets to achieve the same goal [19]. Besides, covert channel presence was discussed in the DHCP protocol [20], in IPv6 [21], in additional areas of TCP (e.g., in TCP timestamps [22]) and IPv4 (e.g., by alternations of fragment sizes [23]), and in business processes [24].

Means were developed to deal with the problem of covert channels, like the pump [25], which is a device that limits the number of acknowledgement messages from a higher to a lower security level and thus affects covert timing channels based on ACKs; a concept extended in [26]. Other well-known techniques are, for instance, covert flow trees [27], which can be used to detect direct and indirect covert channels in source code, as well as the shared resource matrix (SRM) methodology [28] and the extended SRM [29], which can also be applied to source code but can additionally be used in other steps of the software development lifecycle. A newer apprach is program transformation to remove timing leaks and covert timing channels [30], and steganalysis of covert channels in VoIP traffic [31]. Hu introduced fuzzy time to limit the capacity of timing channels between virtual machines on the VAX security kernel [32], and Zander et al. as well as Berk et al. applied different means based on machine learning and statistics to detect covert timing channels in network transmissions [6], [33].

A concept similar to PCs is the idea of a port knocking covert channel as presented by deGraaf et al. [34]. Since port knocking alters information specifying the encapsulated protocol (using the UDP destination port), a port knocking covert channel can be considered as variant of a PC. A first detection algorithm for PC (but not for PHCC) was presented

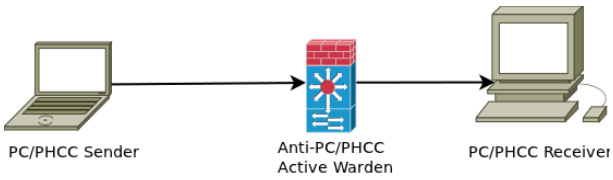


Figure 2. Location of the Anti-PC/PHCC active warden

in [35], but we are not aware of any research to limit the bitrate of PC and PHCC or to prevent them.

PHCCs were also used by Yarochkin et al. to create adaptable network covert channels with the goal to change the utilized network protocols, if required [36]. For instance, a covert channel could use HTTP and ICMP. Due to a change in the firewall configuration between covert channel sender and covert channel receiver, the ICMP protocol get might blocked. The idea of Yarochkin et al. is to filter out such blocked protocols and afterwards switch to another protocol to overcome the firewall. Li and He presented a similar approach based on the concept of natural selection in which survival values are calculated for each network protocol to evaluate its usefulness [37]. The authors only utilize protocols with the currently highest survival values to overcome firewalls.

III. An Active Warden Countering Protocol Switching Covert Channels

The idea of an active warden addressing PHCC and PC focuses on one aspect both covert channel types share: the protocol switches. For both channel types, it is a necessity to ensure that network packets using different network protocols reach their destination in the same order as they were sent. Our approach of an active warden for countering PHCC and PC monitors the protocol switching behavior of network hosts and introduces delays in network packets if a protocol switch occurs.

Like the network pump (a device that only allows acknowledgement messages from a high to a low system in a regulated manner [25]), we have no explicit detection capabilities in our active warden but aim on limiting the channel's bitrate nevertheless. In comparison to the pump, we do not limit acknowledgement messages but alter protocol occurrences. Using the presented technique, we can prevent performance decreases for downloads and uploads and minimize practical implications (cf. Section V-A) by installing the active warden on a company's uplink or between (autonomous) systems. The warden only affects those network packets that change the last occurring network protocol.

The active warden must be located between a covert channel sender and a covert channel receiver (Figure 2). To prevent PC/PHCC-based data leakage for enterprises, a suitable location would be the company's uplink to the Internet. The active warden's delay is configurable and thus makes our approach adjustable, i.e., an administrator can choose the individual optimum between maximized protection and maximized throughput. Formally, this creates a multi-criterion optimization problem. For a given delay d, data leakage can occur at a maximum rate $R(d)$, that is decreasing with increasing d. On the other hand, the side-effects for legitimate users will increase with increasing d, i.e., can be modeled by a function $S(d)$. Ideally, one would like to minimize both, R and S, which is however not possible. One can combine the two in a target function

$$\text{penalty}(d) = \epsilon \cdot R(d) + (1 - \epsilon) \cdot S(d) \ ,$$

which is then to be minimized, i.e., after fixing a suitable $\epsilon \in [0; 1]$ the minimization results in an optimal d that represents the administrator's priorities. As both R and S are assumed to be monotonous, a Pareto optimum can be found in the sense that a further reduction of R by increasing d cannot be achieved without increasing S. Typically, instead of using R and S directly, they are normalized to a certain range such as interval $[0; 1]$, and they might be adapted by linear or non-linear functions that reflect e.g., the severeness of an increased leakage.

Imagine a PC using ICMP (1 bit) and UDP (0 bit) and the goal to transfer the message "0110001". In this case, the sender would need to send UDP, ICMP, ICMP, UDP, UDP, UDP, ICMP. If our active warden is located on a gateway between both hosts and can delay packets, which probably belong to a PC or PHCC, the successful information transfer will be corrupted. At the beginning, the sender sends an UDP packet, which is forwarded by the active warden. Afterwards, the sender sends the ICMP packet, which is delayed for a time d because a protocol switch happened. The next packet is an ICMP packet again and therefore not delayed but forwarded. Afterwards an UDP packet occurs, which is delayed for a time d, too. The next two UDP packets do not change the last protocol and are therefore forwarded. The last ICMP packet results in an additional packet switch and is therefore delayed for time d again. If d is 1 second, then all delayed packets will arrive after the non-delayed packets if the sender did not introduce synthetic delays itself. The resulting packet order at the receiver's side will be UDP, ICMP, UDP, UDP, ICMP, UDP, ICMP, or "01**00**101" (containing two incorrect bit values).

The situation is similar for PHCC where the hidden information is not represented through the protocol itself but through alternations of a protocol's attributes (such as the IPv4 TTL or the HTTP "User-Agent"). If the active warden modifies PHCC transmissions sent via different protocols, the reassembled payload will be jumbled.

In order to prevent the covert channel users to take advantage of learning about the value of d, it might also be randomized, cf. Sections III-B and IV-B.

A. Bitrate Calculation

Tsai and Gligor introduced the formula $B = b \cdot (T_R + T_S + 2 \cdot T_{CS})^{-1}$ for calculating the bandwidth of covert channels using the values T_R (time to receive a covert message), T_S (time to send a covert message), T_{CS} (time required for the context switch between processes), and b (amount of transferred information per message) [38]. While the formula addresses local covert storage channels, all parameters are adaptable to a network covert channel as well.

We use a modification of this formula (cf. formula 1) to calculate the max. possible error-free bitrate of a PC in case our active warden is located between sender and receiver. We introduce the active warden's delay d multiplied with the probability p of a protocol switch per packet. Instead of T_R, T_S and T_{CS}, we use T to represent the *transmission* or *gap* time:

We call T the gap time to represent the minimal time difference between two packets of the covert channel. If T is too small, packets can overrun each other or the receiver cannot process them fast enough, thus, T is limited due to the technical environment of the channel: For high performance computers connected via gigabit links, T is small, while it will be bigger for systems connected via DSL over the Internet.

On the other hand, T can be understood as transmission time if the channel is designed to only transfer packets in a sequential manner, i.e., one packet has to be received before another packet can be sent. In that case, the gap time is equal to the transmission time. The transmission time is the time required for receiving and sending a packet including the processing time required by the active warden.

In the remainder, we call T the gap time, but as explained, T can also be the transmission time between packets since T depends on the implementation of the covert channel (if the channel is not capable of sending a successing packet before an earlier packet was received).

The amount of transferred data per packet (b) is 1 bit/packet in case two protocols are used for a PC since $b = \log_2 n$, where n is the number of protocols used. Thus, p as well as n will increase if more than two protocols are used (more information can be transferred per packet but the switching rate will also increase). In case of a PHCC, b depends on the amount of storage data per packet and not on the number of protocols used. Therefore, p will increase if more protocols are used but since the number of protocols is not linked to the amount of information transferred per packet, b will *not* increase if more protocols are used.

$$B = b \cdot (p \cdot d + T)^{-1} \quad (1)$$

Theoretically, p is 0.5 if randomized input, a uniform coding and a set of two protocols is used since the next packet is either using the same protocol as the last (no protocol switch) or the other protocol (a protocol switch is taking place). In our experiments, the average protocol switching value for a typical protocol switching covert channel using only two protocols was $p = 0.4738806$. Thus, to transfer information without risking a corruption through a delay, a PC/PHCC user is forced to send packets with protocol switches in a way that the delay d cannot corrupt the packet order.

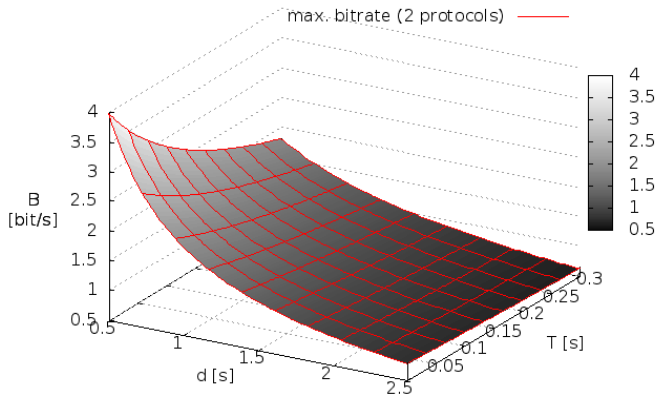


Figure 3. A PC's bitrate (B) using a set of two protocols depending on the delay d and the transmission time T.

As mentioned earlier, the value p depends on the amount of protocols used as well as on the channel's coding. If a uniform coding was used (as with optimized channels) and if two protocols are used p is approx. 0.5. In case four protocols are used, p is approx. 0.75. In general, for n protocols used p is $(1 - 1/n)$. Thus, formula 1 can be modified to the following version:

$$B = b \cdot ((1 - 1/n) \cdot d + T)^{-1} \quad (2)$$

Protocol Channel: As also already discussed, $b = \log_2 n$ in case of a PC. Thus, $B = \log_2 n \cdot ((1 - 1/n) \cdot d + T)^{-1}$. Taking d as well as T into account using formula 1, we calculated the maximum useful bitrates for an uncorrupted PC transfer using a set of two protocols (Figure 3). According to our calculations, a PC's bitrate can be reduced to less than 1 bit/s if the active warden introduces a delay of 2.0 sec for protocol switches. For T, we used a time range obtained from measurements of the original "pct" program.

Protocol Hopping Covert Channel: For a PHCC, the parameter b varies more than parameter T (is, as in the case of a PC, usually very low). Therefore, we created a different plot where we set T to the static value 0.005, which we measured in experiments. Figure 4 shows the bitrate of a PHCC dependent on the delay d as well as the number b of transferred bits per packet. Obviously, the result of the

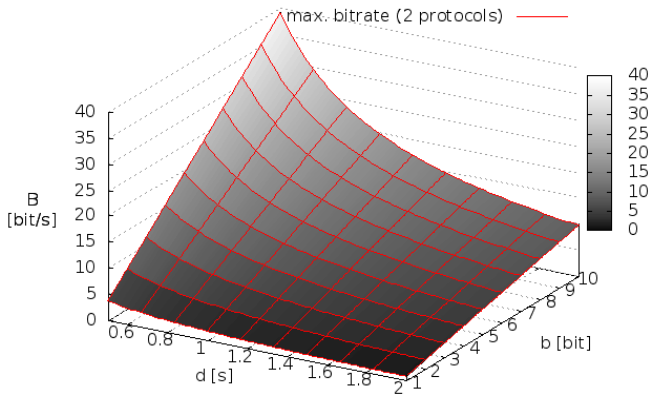


Figure 4. A PHCC's bitrate using *two* protocols, $T = 0.005$ and delays between 0.5 and 2s as well as the capability to transfer between 1 and 10 bits per packet.

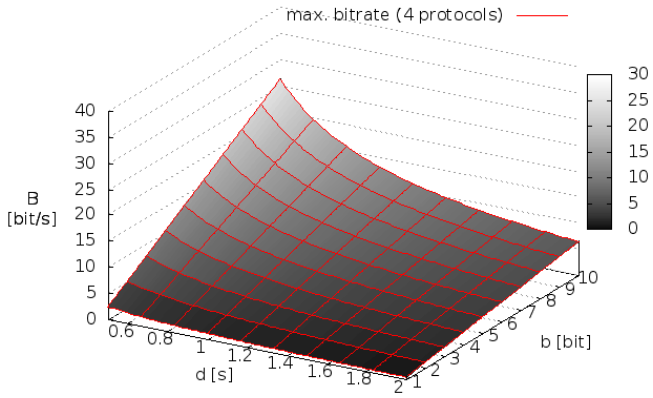


Figure 5. A PHCC's bitrate using *four* protocols, $T = 0.005$ and delays between 0.5 and 2 as well as the capability to transfer between 1 and 10 bits per packet.

PHCC is equal to the result of a PC if $b = 1$. However, PHCCs can carry more information and are therefore harder to limit, i.e., they require higher delays.

As shown in Figure 5, the bitrate of a PHCC decreases if the number of protocols used increases, since more protocol switches ($p = 1 - 1/4$) occur, i.e., the active warden's efficiency for PHCCs is positively correlated with p.

Before we experimentally validate the estimated bitrate of the protocol switching covert channel, we will discuss details of our implementation.

B. Improved Coding

The presented approach addresses PC and PHCC without special coding but with a coding as available in their respective proof-of-concept codes. We highlight the advances of a better coding for both, PC and PHCC, as a means to counter the active warden.

PC with improved coding: If a PC uses a coding that requires to send new packets only if a value unequal to the current value is required to be transferred, it can overcome the active warden, if sender and receiver are synchronized. This is possible if the sender only transfers a network packet if a protocol switch occurs, i.e., two packets of the same protocol are never transferred after another. The timing intervals between the protocol switches represent the amount of bits to transfer. Thus, such a covert channel would be a hybrid version of a timing channel and a PC.

Example: The sender sends a packet of protocol P_1. The active warden delays it for time d and forwards it. Three more bits as represented by P_1 shall be transferred. Therefore, the sender waits for three time slots. Afterwards, a bit represented by P_2 shall be transferred and the sender sends one such a packet. The active warden delays the packet for d and forwards it. If the sender sends P_1 again, it will also be delayed for d. The receiver will receive P_1, three waiting slots, P_2, P_1, i.e., the same input as was sent by the sender. The only disadvantage introduced by the active warden is the delay of d for all packets but this is a minor consequence for the covert channel since even if the message is delayed, it still reaches its receiver without comprising errors. Thus, such a coding would primarily result in side-effets on legitimate traffic due to d but would have no direct affect on the covert channel traffic.

To overcome this problem, an improved version of the active warden was developed. In our previous experiments, we focused on an active warden with a constant delay d. If d varies from packet to packet, or in other words, d is randomized (e.g., $d \in [0.1; 2]$ sec), previous packets are likely to overrun newer ones if the timing interval of the sender is too small. Thus, the sender is forced to use big waiting times and thus, will be forced to decrease its bitrate.

Besides the previously mentioned coding, a PC could also use other codings to improve the amount of bits transferred per protocol switch (b/p). For the default PC coding and two protocols, $p = 0.5$, but when a run length limited (RLL) coding (as used for hard disks [39]) is implemented, p can be decreased.

In case of geometrically distributed symbols, an optimized coding (Huffman coding) can help the covert channel's user to minimize the amount of packets to transfer, but – as usual for covert channel research – we focus on an optimized coding using a uniform distribution (e.g., the covert channel is used to transfer encrypted input).

Another variant of a PC might use unary encoding of

symbols with $n = 2$ protocols. In order to send a value $i \in \{0, \ldots, k-1\}$, $i+1$ packets of the first protocol and $k-i$ packets of the second protocol are sent. Assuming that k is a power of two, then $b = \log_2(k)/(k+1)$ bits are transferred per packet, as k different symbols could be encoded in $\log_2(k)$ bits, and $k+1$ packets are used to transmit one symbol. Note that $b < 1$, for example with $k = 16$, we have $b < 0.25$. The sender then waits for some time before sending the packets for another symbol. The receiver can decode a symbol if the packets for one symbol arrive before any packets for the next symbol arrive. The sender might also use the two protocols alternately to code symbols. In both cases, the gap between two symbols must be rather large to overcome an active warden with a randomized delay d. Because the delay is unknown to the sender, it can neither rely on a maximum delay nor a minimum delay of any packet. Hence, to clearly separate the packets for successive symbols on the receiver side, the sender's waiting time between symbols must be high, at least larger than the maximum value of d. Thus, the bitrate of such a channel is restricted to b/d, which is less than 1 bit/s if $d \geq 1$ s as $b < 1$ bit/packet.

Receiver-side re-calculation attack for PCs: In case a constant delay is used, it is possible for the attacker to recalculate the original sequence of received packets of a PC. However, due to the jitter, it is possible that the attacker is forced to use error-correcting codes. The active warden can implement the previously mentioned randomized d to overcome this problem. Also, the overhead for error correction additionally reduces the available PC bitrate.

PHCC with internal control protocols A drawback of our approach is linked to a feature only available for PHCCs but not for PCs. PHCCs provide usually enough covert space to contain a covert channel-internal control protocol (called a *micro protocol*) [11]. Such micro protocols can be used to embed sequence numbers in covert channel packets as done in [12], [40], [41].

Using these sequence numbers, the receiver can reassemble network packets even if their ordering was disturbed [12]. While the active warden is not able to completely solve this problem, it *forces* a PHCC user to *use* a sequence number. Such a storage channel-internal sequence number usually consists of only 2-3 bits and thus, the active warden can break the receiver-side sorting nevertheless, if d is large enough.

Additionally, since these channels do only provide a few bits per packet, the active warden decreases the available space per packet by forcing the presence of a sequence number or of a larger presence value. Thus, a user is forced to send more packets to transfer the same amount of data than in the case where no active warden would be located on the path between PHCC sender and PHCC receiver.

However, it is important that the size of the micro protocol header is as tiny as possible since the few available bits of space provided within a PHCC should be used to transfer payload. If less space is available per packet due to a larger micro protocol header, more packets are required to be transferred to send a given payload to the receiver.

IV. Experimental Validation

In the following, we discuss our implementation and the results of our validation. We set up an experimental environment in form of a virtual network to represent a realistic scenario for a data leakage in an enterprise network. The content used in the experiment was generated by two available proof of concept tools that could be used by any attacker and thus, also represent realistic user generated traffic.

A. Implementation

For our test implementation, we set up a virtual network between two virtualized Linux 3.0 systems (a covert channel sender as well as a receiver with a local active warden instance) using *VirtualBox* (*www.virtualbox.org*). Both hosts were connected using a virtual Ethernet interface and IPv4. Our proof of concept code focused on layer 4 protocols identifiable by the IPv4 "protocol" field only. Therefore, we modified the *protocol channel tool* (pct) [42] that used ARP and ICMP to use UDP and ICMP instead. Additionally, we implemented the functionality to generate randomized input and to adjust the channel's bitrate.

To implement the network delays on the receiver system that is acting as both the active warden gateway and the protocol switching covert channel receiver at the same time, we made use of the firewall system *netfilter/iptables*. Netfilter/iptables provides a "queue" feature, which can be used to redirect data packets to a userspace program. Berrange implemented the Perl-based program *delay-net* [43] that enforces configurable network delays using the *IPQueue* module [44], which is based on the iptables queue feature. We modified delay-net in a way that it only delays packets after a protocol switch happened. We also implemented a third program to evaluate the correct transmission at the receiver's side, to test our prediction from formula 1 (cf. Section III-A).

Since this test focuses on the protocol switching capability of both, the PC and the PHCC, at the same time, the testing method is valid for both covert channel types. However, we additionally used the *protocol hopping covert channel tool* (phcct) [45] to verify the results for PHCC with and without micro protocols and high data rates since PCs cannot comprise such internal control protocols.

B. Results

In our test configuration, the value T is quite small (we measured 0.005 in average) in comparison to the delay time d. As mentioned in the previous Section, we were able to determine $p = 0.4738806$ through observing the behavior

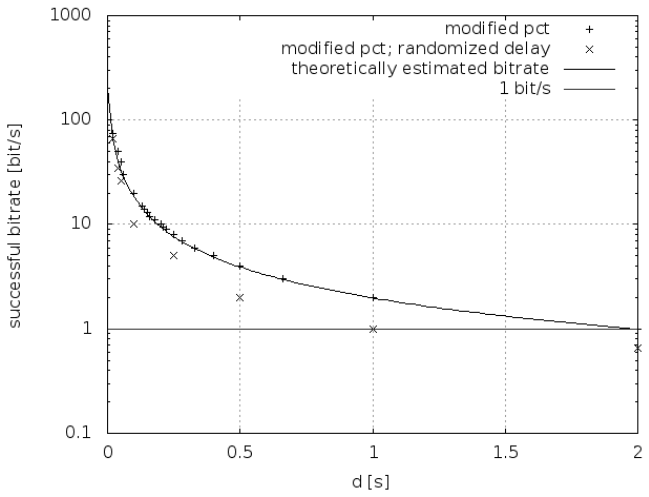


Figure 6. Measured maximal bitrate of the modified *pct* dependent on the active warden's delay in case a constant or a randomized delay is applied. In comparison, we show formula 1 using the estimated protocol switching value.

of the modified *pct* in our virtual test network. However, p turned out to be slightly higher (0.53) in a real network environment, where protocols occur, which do not belong to the PC, and therefore result in few additional protocol switches.

In our test setting we sent PC data using different bitrates and monitored the correctness of the received packet order at the receiver's side. Using this method, we were able to find out the maximum bitrate able to work error-free dependent on the delay introduced by the active warden. Figure 6 shows the results in comparison to our calculation of B (formula 1). The differences between B and our recorded values are small. We evaluated both, an active warden with a constant delay d as well as an active warden with a randomized delay in the range $[0, d[$.

An active warden with a *constant* delay value $d = 2.1\ s$ ($T = 0.005$) reduces the bitrate limit required for a successful transmission of data to $1\ bit/s$. If $d = 1.0\ s$, the bitrate limit is reduced to a maximum of $2.088\ bit/s$.

If we apply a *randomized* delay, the results are better than in case of a constant delay. To reduce the bitrate to $1\ bit/s$ we only needed to apply a max. delay of $d = 1\ s$. A max. delay of $d = 2\ s$ reduces the PC's bitrate to $0.65\ bit/s$. The higher efficiency of a randomized delay comes due to the circumstance, that when a packet is delayed and the following packet is delayed as well, the second packet can be forwarded earlier than the first packet and thus, jumbles the packet sequence. If both packets are delayed with a constant time, a later delayed packet cannot overrun a previously delayed packet. Figure 7 visualizes both delay differences.

Comparing both results as shown in Figure 6, we identified the randomized delay to provide a better efficiency than

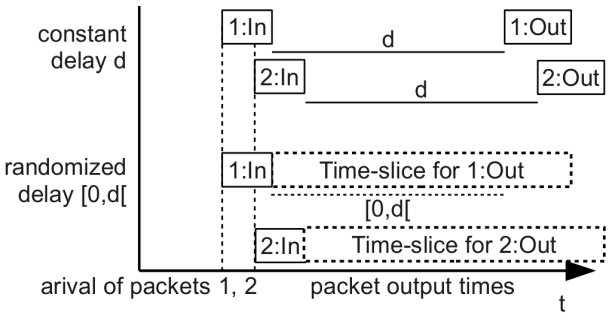


Figure 7. Output of two delayed packets for a constant and a randomized delay.

the constant delay.

Since PC and PHCC can both be seen as covert storage channels, the interesting aspect was to test PHCC situations, which are not exactly the same as for PCs. We therefore ran two experiments:

In the *first experiment*, phcct was used to transfer data with its capability to re-sort jumbled packet sequences using its internal micro protocol. The results showed, that phcct was indeed capable of re-sorting test traffic for 10KByte and 100kByte payload transfers (each sent 10 times through the active warden).

For the *second experiment*, we modified phcct in a way that the re-sorting capability of the internal control protocol was turned off. Since phcct is capable of transferring $b = 792$ bits per packet, the applied delays are less efficient than with a smaller b (as it is given for PCs). However, no transfer without errors was possible (*with and without* the active warden), if phcct was ran and transferred more than 1KByte (11 packets) of payload. Thus, high bitrate transmissions – as done by phcct – without a re-sorting capability are practically not feasible for PHCCs.

V. Discussion

After the concept, implementation and the measured results of the active warden were presented, this Section aims on discussing practical aspects of the active warden and improvements for covert channels to counter the active warden.

A. Practical Aspects

A goal of the presented active warden approach is to design the system for a practical use. The requirement for only small delays is – even if a user's initial request to a website will be delayed – an acceptable limitation for legitimate traffic in high-security environments since delays of only around 2s can reduce the useful bitrate of PCs to a maximum of 1 bit/s. For PHCC, the value can differ if the channel provides high values for b. However, to achieve the goal of practical usefulness, it is necessary to implement additional functionality because of the following reasons:

DNS requests: Typically, a user sends DNS requests to a DNS server and, after receiving a response, connects to a system using another protocol. It is required to take care of this typical effect (and similar effects such as using HTTPS right after a user clicked on a link of a HTTP-based website). Protocol switches occur in both cases (DNS→HTTP and HTTP→HTTPS, respectively). The DNS server and the system a user connects to (usually a web server) will in almost all cases have different addresses, thus it is easy to address this problem if the active warden distinguishes destination hosts. In Section VI, we present an approach based on formal grammar to address this problem.

Different protocols on a single host: However, situations are thinkable in which a user is connected to one host using two different protocols, e.g., to an SMTP server and an IMAP server hosted on the same system. In such cases, whitelisting (e.g., defining trusted hosts) can reduce this problem. This problem is also addressed in Section VI.

Multiple senders: In an enterprise network, there are usually a number of different computers with Internet access. If the active warden is located on the uplink, it will notice many protocol switches since different systems use different protocols to achieve different tasks. The active warden should distinguish source addresses to solve this problem. Therefore, it is also necessary to distinguish source addresses in the active warden to overcome this problem.

Some companies run a *network address translation* (NAT) service *within* their network. These systems would appear as a single system to the active warden although the systems as well as the active warden are located inside the company network. Thus, the NAT'ed systems would face delays. A whitelisting is no sufficient solution since these address translated systems are required to be protected from data leakage too. A possible remedy for that problem would be to use *remote physical device fingerprinting* [46] to count the number of NAT'ed systems and apply smaller delays per packet switch if the number of hosts behind NAT increases (and vice versa).

Redundancy: As all normalizer and firewall-like systems, our prototype of an active warden can result in a single point of failure if not operated on a redundant installation. Modifications of existing redundancy protocols (e.g., the common address redundancy protocol, CARP) might be used to solve this problem. However, as any firewall-like system, the active warden introduces side-effects, i.e., delays, into network traffic.

End-User Limitation: As explained, the effect for end-users is low if the active warden is used.

While an extensive end-user study was not part of this work, we measured different HTTP request-response times for 10MByte downloads over the active warden. The standard download time in our network was 0.41-0.57s. After we installed the active warden, we ran a HTTP download as well as a 0.25 bit/s PC to simulate a number of protocol switches as they occur for modern websites (multiple DNS requests for a whole site are normal since they can include script sources from other domains). This increased the download time to 0.43-3s. We observed that the basic limitation for connections happens in the establishment phase where a new protocol (HTTP over TCP) occurred. In the context of the 4s-rule for website rendering [47], we can assume that our active warden is valuable for practical use-cases.

To summarize, all mentioned problems (except the network address translation) are solvable by adding the mentioned simple features. The configurable delay parameter d provides administrators a way to adjust the efficiency of the active warden to their requirements. Since only protocol switching packets are affected by the active warden, most of a network's traffic is not affected, i.e., download rates and upload rates will not decrease notably.

B. Covert Channel Improvements

While we already discussed improved encoding techniques for both, PC and PHCC, in Section III-B, this Section will highlight architectural means, which can be used by both channel types to bypass the active warden. We will also discuss the subject of covert channel-internal control protocols for PHCC.

Multiple Covert Channel Senders: While the previously mentioned approach of distinguishing source addresses is a requirement for the practical application of the active warden, a distributed covert channel sender can take advantage of it: If the covert channel sender consists of multiple inhouse systems, each associated with a single network protocol, these systems can send PC/PHCC data to a destination host outside of the enterprise network through the active warden without causing protocol switches. Therefore, a coordination of the distributed sender hosts is necessary but can be achieved by introducing a single coordinator host connected to the actual PC/PHCC senders. However, due to the presence of the active warden, the covert channel sender is forced to coordinate its distributed transfer and the command and control traffic between the covert channel sender hosts and the covert channel coordinator system are required to be hidden as well, i.e., will raise the chance of a detection.

A similar approach is a **Covert Channels with Multiple Receivers:** If one covert channel host sends packets to different covert channel receivers and each covert channel receiver is associated with only a single protocol, no protocol switches between a single sender and a single destination occur (since each path is only used for one protocol) and no bitrate is limited in a direct way but in an indirect way: If the covert channel receiver is forced to be a distributed system (i.e., a covert channel-based botnet/zombie network), it has to implement a distributed coordination mechanism (sorting packets and extracting all hidden information on a single system that finally computes the whole hidden message). If

the receiver-side network is monitored as well, the coordination itself must be covered too, and thus can probably be detected or at least raise attention. Also, the bitrate is limited since multiple receivers can receive messages via network access points with different performance and the network packets for the coordination can differ in their timings, too. Therefore, the sender must introduce pause intervals (which limit the bitrate) between new packets to prevent a jumbled result at the receiver.

Improved Control Protocols: PHCCs with internal control protocols can – if a sequence number is present – resort jumbled packet sequences at the receiver side and thus, can make the active warden inefficient even if delays are introduced. However, applied delays can cause the active warden to report abnormal protocol switching traffic and it would be useful, if the size of PHCC-internal control protocols could can be shrinked to prevent such attention raising behavior.

Therefore, we developed a technique called *status updates* [48]. The concept of status updates is to only transfer a header component to the receiver, if the last *status* of the header component changes. For instance, imagine a PHCC between a system A and C via a proxy B, i.e., $A \rightarrow B \rightarrow C$. Therefore, A configures the covert channel proxy B to forward data to C via sending a status update that defines the forwarding destination C. All following packets received by B from A will be forwarded to C and A is not required to define the covert payload's destination C again. However, A can send a new status update for the destination to B to, for instance, send all following traffic to D (instead of sending it to the previously configured destination C). Status updates work for all status-based header components (e.g., also for source addresses or fields to indicate the start/end of a transaction) and can therefore decrease header sizes.

We combined our status update approach in [48] with dynamic routing for covert channel overlays. If a PHCC sender can measure a delay applied on protocol switches between itself and the PHCC receiver by ping'ing the receiving peer with and without causing a protocol switch, it can determine the possible presence of an active warden. In such a case, the PHCC sender could try to find an alternative routing path that does not face delays on protocol switches. Afterwards, the PHCC can even be transformed into a PC since no delay is applied and no packet jumbling will occur. To achieve this goal, we developed the *Smart Covert Channel Tool* – a dynamic routing middleware capable of utilizing various covert channel technologies, which can include PCs, but other covert channel techniques, such as network timing channels and network storage channels, as well.

VI. Improvement Proposal 1: Applying Formal Grammar to Increase Practical Usage

As mentioned earlier, our active warden must comprise an additional means to handle selected practical issues. We therefore propose a means based on formal grammar that addresses the following previously discussed problems:

1) The problem of delaying legitime protocol switches (e.g., DNS $\rightarrow$ HTTP),
2) the problem of host-related protocol switches (e.g., a client downloads from a web-server while sending a large e-mail to the SMTP server),
3) the problem of running multiple services on a single host (e.g., SMTP and IMAP service on the same machine).

Formal grammar has previously been applied in the context of computer security. For instance, Gorodetski et al. used formal grammar for attack modelling in [49] and Trinius and Freiling realized SPAM filters with context-free grammars [50].

A formal grammar $G = (V, \Sigma, P, S)$ comprises the set V of non-terminals, the set Σ of terminal symbols, the set P of productions and the starting symbol $S \in V$ [49].

Our formal grammar-based approach is based on the idea of *whitelisting*, i.e., we define the allowed protocol switching behavior within in the context of a company's network within our formal grammar and afterwards test, whether the network's actual behavior is conform to the rules of the grammar. If the protocol switching behavior *is* conform, the active warden does *not* apply a delay, otherwise it will delay packets.

Therefore, we first define the allowed network protocols as terminal symbols p_x where x is the protocol, e.g., $\Sigma_1 = \{p_{dns}, p_{http}, p_{https}, p_{smtp}, p_{imap}\}$. Afterwards, we define additional terminal symbols for the servers s_y in our network where y is the address of the server (it can be an IP or a hostname, e.g., $\Sigma_2 = \{s_{mail}, s_{name}, s_{web}\}$). Both sets form the set of terminal symbols $\Sigma = \Sigma_1 \cup \Sigma_2$.

In the next step, we define the production rules in P. We define productions, which are built in the form <*server*> <*protocol*>, e.g., $s_{mail}p_{smtp}$, which can comprise allowed rules of the utilized grammar type (e.g., context-free or regular).

The following example grammar allows

1) the use of SMTP and HTTP (both also after DNS), and
2) to switch from HTTP to HTTPS and vice versa, i.e., we support HTTP-based websites with HTTPS links/content (and vice versa),
3) as well as to switch between SMTP and IMAP to allow sending and receiving e-mails at the same time;
4) to run different services on the same machine (here, the SMTP and IMAP services are located on the same machine s_{mail});
5) the simultanous use of e-mail, DNS and web access, since W_2 allows the production M_2 and since M_2 allows the production W_2.

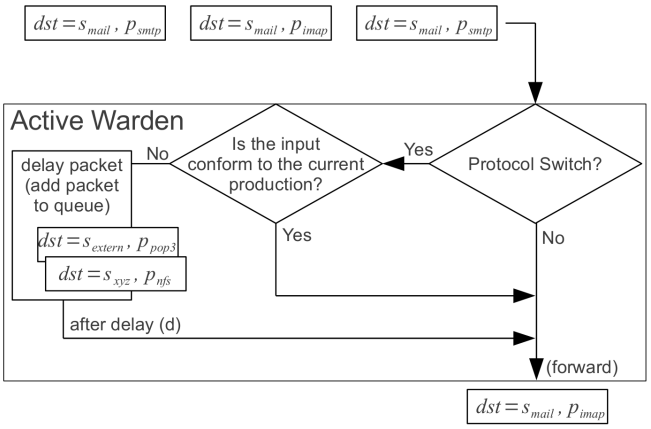


Figure 8. The integration of grammar productions and terminal symbols in the active warden.

$$V = \{S, D, W_1, W_2, M_1, M_2\} \quad (3)$$

$$P = \{S \rightarrow D(W_1|M_1)|W_1|M_1 \quad (4)$$

$$D \rightarrow s_{name}p_{dns} \quad (5)$$

$$W_1 \rightarrow s_{web}(p_{http}|p_{https})W_2 \quad (6)$$

$$W_2 \rightarrow DW_1|W_1|M_2|\epsilon \quad (7)$$

$$M_1 \rightarrow s_{mail}(p_{smtp}|p_{imap})M_2 \quad (8)$$

$$M_2 \rightarrow DM_1|M_1|W_2|\epsilon\} \quad (9)$$

If a new packet arrives at the active warden, the active warden tries to find a production rule that fits the protocol sequence – either by starting a production from scratch or by continuing an existing one – in other words, the active warden tries to find a suitable production to allow a direct forwarding. Figure 8 visualizes this concept.

For instance, if a DNS packet was received (e.g., identified by the DNS port or by advanced protocol identifying tests like [51]), it is not delayed because a production rule ($S \rightarrow D(W_1|M_1)$) allows the DNS packet. Afterwards, an SMTP packet arrives, which is also allowed by the production rules ($S \rightarrow D \rightarrow M_1$).

If multiple packets of the same protocol (e.g., many SMTP packets for sending a larger e-mail or multiple HTTP packets from a file download) occur, they are not delayed because no protocol switch is taking place. Thus, non-protocol switching rules are not required, which ensures small grammar productions.

As an additional means to keep grammars as small as possible, we propose to implement layer-based grammars for the application of rules specific to the network layer of the TCP/IP model. Low-level protocol switching covert channels are not considered a threat in that case because they are not available for routing and low-level frames will not directly pass the active warden. For instance, an ARP request need not be modelled in the grammar nor must it be handled by the active warden since the active warden will not forward ARP requests (as long as it does not explicitly act as an ARP proxy).

VII. Improvement Proposal 2: Detection-capable Active Warden to Counter PCs

In recent work [35], we evaluated the detectability of protocol channels. We presented two algorithms for a PC detection. The first algorithm uses a static formula for a traffic detection, the other algorithm uses machine learning based on the C4.5 algorithm to build a decision tree and provides better results than the first algorithm and shall be discussed in this Section.

The C4.5 algorithm requires a traffic recording of at least a few thousand packets (in our tests, we used 5000 packets) and thus, only existing protocol channels that already transferred information can be detected and the continuing covert channel transmission can be interrupted (e.g., blocked by a firewall or delayed by the active warden).

For the detection of protocol channels, we use the change rate R that reflects how frequently protocol switches for a given sender occur (C is the total number of protocol switches and P the total number of packets of a traffic recording).

$$R = \frac{C}{P} \quad (10)$$

A second parameter introduced is the average time between protocol switches D (T_i is the time of a protocol switch i).

$$D = \frac{\sum_i (T_{i+1} - T_i)}{C} \quad (11)$$

We observed, that protocol channels are linked to higher R and/or smaller D values than normal traffic [35].

Since our technique provides an accuracy of 98-99% for PCs with a bitrate of $4\ bits/s$ or higher, the decision tree-approach can be considered useful in practice if used in conjunction with the active warden. Therefore, the active warden would act as described but would stop applying a delay (or would decrease the applied delay) on traffic that comprises no PC.

As mentioned, the machine learning-based detection approach requires a traffic recording. The active warden must therefore record the traffic on the fly and would only apply delays if enough packets were recorded and if the traffic got classified as being covert channel traffic (optimal results require $n = 5000$ packets). Thus, the active-warden's detection capability would not be usable for the first n packets.

Our problem of requiring enough traffic data before being able to act as required is similar to another kind of active warden: the traffic normalizer. Traffic normalizers face a so-called *cold start problem* [7]: If a traffic normalizer bootstraps, it receives packets of already existing connections and

does not know any details of previously sent packets (e.g., whether a received packet belongs to an actually established connection or not) and cannot apply all normalization rules without causing negative effects on the traffic.

However, after a first traffic recording of n packets got classified, the active warden must continue recording traffic in a queue. The queue should contain $n = 5000$ packets at all times to provide optimal detection results. Therefore, the oldest packet is removed when a new packet is added to the queue. A queue also prevents continuously growing buffers that decrease the active warden's performance over the time.

Since a traffic classification is time-consuming and the performance of the active warden must be ensured, it is not recommendable to classify the traffic recording for each received packet. Instead, the number of packets $< n$ before a new C4.5 classification is applied should be adjusted to the throughput of the links (e.g., every 1000 or 2500 packets).

The active warden can either stop applying delays on traffic if a traffic was classified as *not* being a covert channel, or it can decrease the applied delay. On the other hand, it can apply the delay (or increase the delay) if a traffic was classified as being covert channel traffic.

VIII. Conclusion and Future Work

In this paper, we present the first active warden designed to counter both types of protocol switching covert channels: protocol channels (PCs) as well as protocol hopping covert channels (PHCCs). We limit the useful bitrate of these covert channels by disturbing the protocol switches through synthetically introduced delays. We implemented an active warden with both constant and randomized delay, based on *netfilter/iptables*. The active warden is suitable for a practical use but can still result in side-effects on regular traffic. Therefore, we proposed to combine the active warden with a detection functionality for protocol channels and additionally, to apply whitelisting based on a formal grammar to prevent that delays are applied to legitimate network traffic.

Future work will include to find solutions for the problem of network address translation (NAT) inside a protected network as well as to find solutions for effects of large network environments where load balancing and redundancy protocols are required; the presented prototype was not designed for such environments. Additionally, research must be done to provide an exact bitrate controlling for PHCCs using internal sequence numbers since these channels can resort jumbled packet sequences caused by the active warden. We do not expect our approach to be easily modifiable to counter such advanced protocol hopping covert channels.

Acknowledgement

We would like to thank Sebastian Zander for his comments on the calculation of the max. error-free bitrate.

References

[1] S. Wendzel and J. Keller, "Design and implementation of an active warden addressing protocol switching covert channels," in *Proc. 7th International Conference on Internet Monitoring and Protection (ICIMP 2012)*, A. Wagner and P. Dini, Eds. IARIA, 2012, pp. 1–6.

[2] B. W. Lampson, "A note on the confinement problem," *Commun. ACM*, vol. 16, no. 10, pp. 613–615, 1973.

[3] S. J. Murdoch, "Covert channel vulnerabilities in anonymity systems," Ph.D. dissertation, University of Cambridge, 2007.

[4] *Trusted Computer System Evaluation Criteria (TCSEC, Orange Book)*, Department of Defense (DoD) Std., Aug 1985.

[5] J. P. R. Gallagher, Ed., *A Guide to Understanding Covert Channel Analysis of Trusted Systems (NCSC-TG-030)*. National Computer Security Center, Nov 1993.

[6] S. Zander, G. Armitage, and P. Branch, "Covert channels and countermeasures in computer network protocols," *IEEE Comm. Magazine*, vol. 45, no. 12, pp. 136–142, Dec 2007.

[7] M. Handley, V. Paxson, and C. Kreibich, "Network intrusion detection: Evasion, traffic normalization, and end-to-end protocol semantics," in *Proc. 10th USENIX Security Symposium*, 2001, pp. 115–131.

[8] A. Singh, O. Nordström, A. L. M. dos Santos, and C. Lu, "Stateless model for the prevention of malicious communication channels," *Int. Journal of Comp. and Applications*, vol. 28, no. 3, pp. 285–297, 2006.

[9] G. Gu, R. Perdisci, J. Zhang, and W. Lee, "Botminer: Clustering analysis of network traffic for protocol- and structure-independent botnet detection," in *Proc. USENIX Security Symp.*, 2008, pp. 139–154.

[10] Daemon9, "Loki2 (the implementation)," *Phrack Magazine*, vol. 7, no. 5, September 1997, retrieved: Dec, 2012. [Online]. Available: http://www.phrack.org/issues.html?issue=51&id=6

[11] S. Wendzel and J. Keller, "Low-attention forwarding for mobile network covert channels," in *Proc. 12th IFIP Conf. on Communications and Multimedia Security*. Springer LNCS vol. 7025, 2011, pp. 122–133.

[12] S. Wendzel, "Protocol hopping covert channels," *Hakin9*, vol. 08, no. 03, pp. 20–21, 2008, (in German).

[13] ——, "Protocol channels as a new design alternative of covert channels," *CoRR*, vol. abs/0809.1949, pp. 1–2, 2008.

[14] C. G. Girling, "Covert channels in LAN's," *IEEE Transactions on Software Engineering*, vol. 13, pp. 292–296, February 1987.

[15] M. Wolf, "Covert channels in LAN protocols," in *Proc. Local Area Network Security*. Springer LNCS vol. 396, 1989, pp. 89–101.

[16] C. H. Rowland, "Covert channels in the TCP/IP protocol suite," *First Monday*, vol. 2, no. 5, May 1997, retrieved: Dec, 2012. [Online]. Available: firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/528/449

[17] J. Rutkowska, "The implementation of passive covert channels in the linux kernel," 2004, ftp://ftp.pastoutafait.org/pdf/passive-covert-channels-linux.pdf, retrieved: Dec, 2012.

[18] S. Cabuk, C. E. Brodley, and C. Shields, "IP covert timing channels: design and detection," in *Proc. ACM Conference on Computer and Communications Security*, 2004, pp. 178–187.

[19] K. Ahsan and D. Kundur, "Practical data hiding in TCP/IP," in *Proc. Workshop on Multimedia Security at ACM Multimedia '02*, French Riviera, December 2002.

[20] R. Rios, J. Onieva, and J. Lopez, "HIDE_DHCP: Covert communications through network configuration messages," in *Proc. IFIP TC 11 27th International Information Security Conference, Heraklion, Crete, Greece*. Springer, 2012.

[21] N. Lucena, G. Lewandowski, and S. Chapin, "Covert channels in IPv6," in *Proc. Privacy Enhancing Technologies*. Springer LNCS vol. 3856, 2006, pp. 147–166.

[22] J. Giffin, R. Greenstadt, P. Litwack, and R. Tibbetts, "Covert messaging through TCP timestamps," in *Proc. 2nd international conference on privacy enhancing technologies*. Springer, 2003, pp. 194–208.

[23] S. J. Murdoch and S. Lewis, "Embedding covert channels into TCP/IP," in *Proc. Information Hiding Conference 2005*. Springer LNCS vol. 3727, 2005, pp. 247–261.

[24] R. Accorsi and C. Wonnemann, "Detective information flow analysis for business processes," in *Proc. Business Process, Services Computing and Intelligent Service Management*. GI LNI vol. 147, 2009, pp. 223–224.

[25] M. H. Kang, I. S. Moskowitz, and S. Chincheck, "The pump: A decade of covert fun," in *Proc. 21st Annual Computer Security Applications Conference (ACSAC 2005)*. IEEE Computer Society, 2005, pp. 352–360.

[26] N. Ogurtsov, H. Orman, R. Schroeppel, S. O'Malley, and O. Spatscheck, "Covert channel elimination protocols," University of Arizona, Tech. Rep., 1996.

[27] P. A. Porras and R. A. Kemmerer, "Covert flow trees: A technique for identifying and analyzing covert storage channels," in *Proc. IEEE Symp. on Security and Privacy*, 1991, pp. 36–51.

[28] R. A. Kemmerer, "Shared resource matrix methodology: an approach to identifying storage and timing channels," *ACM Trans. Comput. Syst.*, vol. 1, no. 3, pp. 256–277, 1983.

[29] J. McHugh, "An information flow tool for gypsy - an extended abstract revisited," in *Proc. 17th Annual Computer Security Applications Conference*, 2001, pp. 191–201.

[30] J. Agat, "Transforming out timing leaks," in *Proc. 27th ACM Symp. on Principles of Programming Languages (POPL)*. ACM Press, 2000, pp. 40–53.

[31] C. Krätzer and J. Dittmann, "Früherkennung von verdeckten Kanälen in VoIP-Kommunikation," in *IT-Frühwarnsysteme (BSI-Workshop)*. BSI, 2006, pp. 209–214, (In German).

[32] W.-M. Hu, "Reducing timing channels with fuzzy time," in *Proc. 1991 Symposium on Security and Privacy, IEEE Computer Society*, 1991, pp. 8–20.

[33] V. Berk, A. Giani, and G. Cybenko, "Detection of covert channel encoding in network packet delays," Department of Computer Science - Dartmouth College, Tech. Rep., 2005.

[34] R. deGraaf, J. Aycock, and M. J. Jacobson, "Improved port knocking with strong authentication," in *Proc. 21st Annual Computer Security Applications Conference (ACSAC '05)*. IEEE Computer Society, 2005, pp. 451–462.

[35] S. Wendzel and S. Zander, "Detecting protocol switching covert channels," in *Proc. 37th IEEE Conf. on Local Computer Networks (LCN)*. IEEE, 2012, pp. 280–283.

[36] F. V. Yarochkin, S.-Y. Dai, C.-H. Lin, Y. Huang, and S.-Y. Kuo, "Towards adaptive covert communication system," in *Proc. 14th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC)*. IEEE, 2008, pp. 153–159.

[37] W. Li and G. He, "Towards a protocol for autonomic covert communication," in *Proc. 8th Conf. on Autonomic and Trusted Computing*, 2011, pp. 106–117.

[38] C.-R. Tsai and V. D. Gligor, "A bandwidth computation model for covert storage channels and its applications," in *Proc. IEEE Conf. on Security and Privacy*, 1988, pp. 108–121.

[39] C. D. Mee and E. D. Daniel, *Magnetic Storage Handbook*, 2nd ed. McGraw Hill, 1996.

[40] B. Ray and S. Mishra, "A protocol for building secure and reliable covert channel," in *Proc. 6th Annual Conference on Privacy, Security and Trust (PST 2008)*. IEEE, 2008, pp. 246–253.

[41] D. Stødle, "Ping tunnel – for those times when everything else is blocked," 2009. [Online]. Available: http://www.cs.uit.no/~daniels/PingTunnel/

[42] S. Wendzel, "pct," 2009, retrieved: Dec, 2012. [Online]. Available: http://www.wendzel.de/dr.org/files/Projects/pct/

[43] D. Berrange, "Simulating WAN network delay," 2005, retrieved: Dec, 2012. [Online]. Available: http://people.redhat.com/berrange/notes/network-delay.html

[44] J. Morris, "IPTables::IPv4::IPQueue module for Perl," 2002, retrieved: Dec, 2012. [Online]. Available: http://search.cpan.org/~jmorris/perlipq-1.25/IPQueue.pm

[45] S. Wendzel, "phcct," 2007, retrieved: Dec, 2012. [Online]. Available: http://www.wendzel.de/dr.org/files/Projects/phcct/

[46] T. Kohno, A. Broido, and K. Claffy, "Remote physical device fingerprinting," *IEEE Transactions on Dependable and Secure Computing*, no. 2, pp. 93–108, 2005.

[47] Akamai, "Retail web site performance," 2006, retrieved: Dec, 2012. [Online]. Available: http://www.-akamai.com/dl/reports/Site_Abandonment_Final_Report.pdf

[48] P. Backs, S. Wendzel, and J. Keller, "Dynamic routing in covert channel overlays based on control protocols," in *Proc. International Workshop on Information Security, Theory and Practice (ISTP-2012)*. IEEE, 2012, pp. 32–39.

[49] V. Gorodetski and I. Kotenko, "Attacks against computer network: Formal grammar-based framework and simulation tool," in *Proc. Recent Advances in Intrusion Detection*. Springer LNCS vol. 2516, 2002, pp. 219–238.

[50] P. Trinius and F. Freiling, "Filtern von Spam-Nachrichten mit kontextfreien Grammatiken," in *Proc. Sicherheit 2012*. GI LNI vol. P-195, 2012, pp. 163–174, (in German).

[51] L. Bernaille, R. Teixeira, I. Akodkenou, A. Soule, and K. Salamatian, "Traffic classification on the fly," *SIGCOMM Computer Communication Review*, vol. 36, no. 2, pp. 23–26, Apr. 2006.

Securing Access to Data in Business Intelligence Domains

Ahmad Altamimi
Department of Computer Science
Concordia University
Montreal, Canada
Email: a_alta@cs.concordia.ca

Todd Eavis
Department of Computer Science
Concordia University
Montreal, Canada
Email: eavis@cs.concordia.ca

***Abstract*—Online Analytical Processing (OLAP) has become an increasingly important and prevalent component of Decision Support Systems. OLAP is associated with a data model known as a cube, a multi-dimensional representation of the core measures and relationships within the associated organization. While numerous cube generation and processing algorithms have been presented in the literature, little effort has been made to address the unique security and authorization requirements of the model. In particular, the hierarchical nature of the cube allows users to bypass - either intentionally or unintentionally - partial constraints defined at alternate aggregation levels. In this paper, we present an authorization framework that builds upon an algebra designed specifically for OLAP domains. It is Object-Oriented in nature and uses query re-writing rules to ensure consistent data access across all levels of the conceptual model. For the most part, the process is largely transparent to the user. We demonstrate the scope of our framework with a series of common OLAP query case studies, as well as an experimental performance analysis using a common OLAP benchmark. The end result is an intuitive but powerful approach to database authorization that is uniquely tailored to the OLAP domain.**

Keywords-Data warehouses; Data security; Query processing

I. Introduction

Data warehousing (DW) and On-Line Analytical Processing (OLAP) play a pivotal role in modern organizations. Designed to facilitate the reporting and analysis required in decision making environments, OLAP builds upon a multi-dimensional data model that intuitively integrates the vast quantities of transactional level data collected by contemporary organizations. Ultimately, this data is used by managers and decision makers in order to extract and visualize broad patterns and trends that would otherwise not be obvious to the user.

One must note that while the data warehouse serves as a repository for all collected data, not all of its records should be universally accessible. Specifically, DW/OLAP systems almost always house confidential and sensitive data — identification information, medical data or even religious beliefs or ideologies — that must, by definition, be restricted to authorized users. As a result, various pieces of legislation designed to protect individual privacy have been proposed. One can consider, for example, the United States HIPAA-Health Insurance Portability and Accountability Act, which regulates the privacy of personal health care information, the GLBA (Gramm-Leach-Bliley Act, also known as the Financial Modernization Act), the Sarbanes-Oxley Act, and the EU's Safe Harbour Law. These laws usually require strict technical security measures for guaranteeing privacy, with a failure to comply possibly leading to significant penalties. In this context, organizations must be able to guarantee the correct administration, security and confidentiality of the information they collect and store.

The administrator of the warehouse is ultimately responsible for defining roles and privileges for each of the possible end users. In fact, a number of general warehouse security models have been proposed in the literature. Several authors define frameworks that are likely too restrictive for production warehouses. For example, security models have been based upon the notion of user-specific *authorization views* that allow access only to selected data. However, the administration of these views becomes quite complex when a security policy is added, changed, or removed. Moreover, complex roles can be difficult to implement in practice, and models of this type tend not to scale well with a large number of users. Conversely, other researchers have focused on the design process itself, including the use of Unified Modeling Language (UML) profiles for the definition of security constraints. Here, however, the physical implementation of the underlying authorization system remains undefined.

In a recent paper, we presented an authorization model for OLAP environments that is based on a query rewriting technique [1]. The model enforces distinct data security policies that, in turn, may be associated with user populations of arbitrary size. In short, our framework rewrites queries containing unauthorized data access to ensure that the user only receives the data that he/she is authorized to see. Rewriting is accomplished by adding or changing specific conditions within the query according to a set of concise but robust transformation rules. Because our methods specifically target the OLAP domain, the query rules are directly associated with the conceptual properties and elements of the OLAP data model itself. A primary advantage of this approach is that by manipulating the conceptual data model, we are able to apply query restrictions not only on direct access to OLAP

elements, but also on certain forms of indirect access.

In the current paper, we expand upon the original work in two ways. First, we discuss the data structures and algorithms utilized by the functions that manipulate the hierarchical elements of the conceptual data model. The performance of the transformation process is closely associated with these mechanisms. To underscore the practical viability of the proposed methods, we have also added an experimental section that highlights the processing overhead relative to the execution costs of the underlying query. In addition to these core enhancements, we update the paper with a deeper treatment of the internal representation of the intermediate query, as well as a broader discussion of the work related to this research domain. Finally, an appendix has been included in order to provide the reader with a clear description of the query test cases.

The paper is organized as follows. In Section II, we present an overview of related work. Section III describes the core OLAP data model and associated algebra, and includes a discussion of the object-oriented query structure for which the proposed security model has been designed. The OLAP query rewriting model and its associated transformation rules, including the extended section on query representation and hierarchy processing, are then presented in detail in Section IV. Experimental results are discussed in Section V, with final conclusions offered in Section VI.

II. Related work

The need for strong security mechanisms has long been recognized in the context of relational database management systems. A variety of *Access Control* techniques have in fact been proposed to restrict access to the appropriate authorized users. Each such technique aims to limit users and/or processes to performing only those table or column operations (i.e., read, write, or execute) for which they are actually authorized. The relevant control then either allows or disallows the execution of the specific operation to be performed.

During the early stages of database security research, the primary focus was on *Discretionary Access Controls* (DACs) [2]. The basic form of DAC authorization consists of a triple (*s, o, a*), such that a set of security *subjects s* can execute *actions a* on a set of security *objects o*. The earliest DAC model was the Access Matrix, whereby authorization is represented in an $|s| * |o|$ matrix in which rows are subjects, columns are objects and the mapping of subject and object pairs results in the set of rights the subject *s* has over the object *o*. A primary benefit associated with the use of a DAC is that it can be implemented relatively easily. However, in practice, large organizations give rise to extremely large access matrices. Maintaining matrix contents can be difficult as the matrix needs to be updated with each update to the subjects (e.g., addition of users) or objects (e.g., addition of columns).

In the 1980's the focus moved to *Mandatory Access Controls* (MACs) [3]. The most common form of MAC is the multilevel security policy, which secures data by assigning security labels to subjects and objects, and subsequently compares these labels to the level of sensitivity at which a user is operating. The access controls in MACs restrict subjects from accessing information labeled with a higher level. In other words, a user can access the data in his/her security level or in a lower security level(s) but not in a higher level(s). MAC is relatively straightforward from a design perspective and is considered a good model either for systems in which confidentiality is a primary access control concern, or in which the objects being protected are valuable. That being said, MAC systems can also be expensive to implement due to the necessity for applications to be rewritten to adhere to MAC labels and properties. Also, MACs do not provide each user with a distinct authorization context (i.e., access to only their own data address), nor fine-grained least privilege mechanisms.

An alternative approach was introduced in the 1990's [4]. This new model is known as *Role Based Access Control* (RBAC). RBAC consists of roles, permissions, and users. Roles are created for various job functions, with permissions for specific operations then assigned to these roles. Users are assigned particular roles, and through those role assignments acquire permissions to perform particular operations. The consolidation of access control for many users into a single role entry allows for much easier management of the overall system and much more effective verification of security policies. However, in large systems, role inheritance — and the need for finer-grained customized privileges— makes administration potentially unwieldy. Additionally, it is inappropriate for multi-dimensional data modeling due to the fact that it is based on relational concepts (i.e., tables, columns, rows, and cells), and thus, cannot be implemented directly on top of the multi-dimensional model.

In contrast to the Access Control paradigm, a number of security models that restrict data warehouse access have also been proposed in the literature, including those that focus strictly on the design process. Extensions to the Unified Modeling Language to allow for the specification of multi-dimensional security constraints has been one approach that has been suggested [5]. In fact, a number of researchers have looked at similar techniques for setting access constraints at an early stage in the OLAP design process [6], [7]. Others have developed security requirements for the entire Data Warehouse life cycle [8]–[10]. In this case, they first propose a model (agent-goal-decision-information) to support the early and late requirements for the development of DWs, then extend that model to capture security aspects in order to prevent illegitimate attempts to access the warehouse. Such models have great value of course, particularly if one has the option to create the warehouse from scratch. That being said, their focus is not on authorization algorithms

per se, but rather on design methodologies that would most effectively use existing technologies, including the Model Driven Architecture (MDA) and the standard Software Process Engineering Metamodel Specification (SPEM) from the Object Management Group (OMG).

In terms of true authorization models, several researchers have attempted to augment the core Database Management System (DBMS) with authorizations views [11]–[13]. Typically, alternate views of data are defined for each distinct user or user group. A query Q is inferred to be authorized if there is an equivalent query Q' which uses only authorized views. The end result is often the generation of a large number of such views, all of which must be maintained manually by the system administrator. Clearly, this approach does not scale terribly well, and would be impractical in a huge, complex DW environment.

Query rewriting has also been explored in DBMS environments in a variety of ways, with search and optimization being common targets [14]. Beyond that, however, rewriting has also been utilized to provide fine grained access control in Relational databases [15]. Oracle's Virtual Private Database (VPD) [16], for example, limits access to row level data by appending a predicate clause to the user's SQL statement. Here, the security policy is encoded as policy functions defined for each table. These functions are used to return the predicate, which is then appended to the query. This process is done in a manner that is entirely transparent to the user. That is, whenever a user accesses a table that has a security policy, the policy function returns a predicate, which is appended to the user's query before it is executed.

In the Truman model [15], on the other hand, the database administrator defines a *parameterized* authorization view for each relation in the database. Note that parameterized views are normal views augmented with session-specific information, such as the user-id, location, or time. The query is modified transparently by substituting each relation in the query by the corresponding parameterized view to make sure that the user does not get to see anything more than his/her own view of the database. In this model, the user can also write queries on base relations by plugging in the values of session parameters such as user-id or time before the modified query is executed.

We note, however, that the mechanisms discussed above (e.g., Oracle's VPD) are not tailored specifically to the OLAP domain and, as such, either have limited ability to provide fine grained control of the elements in the conceptual OLAP data model or, at the very least, would make such constraints exceedingly tedious to define. Some commercial tools, such as Microsoft's Analysis Services [17], do in fact provide some support for OLAP-level security specification. Here, however, there is virtually no formal basis for the application of authorization logic and little can be said about the actual scope or limitations of the relevant subsystems. This is in contrast to the work discussed in this paper, where the primary contribution is a query rewriting technique that not only transparently supports *indirect* authorization, but does so on the basis of an explicit policy/rule model. Moreover, the mechanisms are not tightly connected to a specific DBMS product but can be applied to virtually any standard data management system.

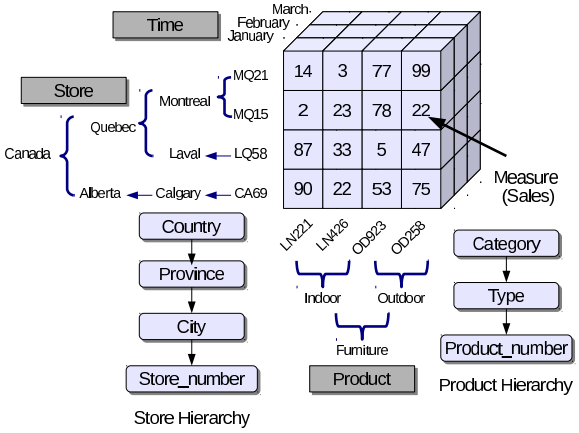


Figure 1. A simple three dimensional data cube

III. The Conceptual Data Model

We consider analytical environments to consist of one or more *data cubes*. Each cube is composed of a series of d dimensions — sometimes called *feature* attributes — and one or more *measures* [18]. The dimensions can be visualized as delimiting a d-dimensional hyper-cube, with each axis identifying the members of the parent dimension (e.g., the days of the year). Cell values, in turn, represent the aggregated measure (e.g., sum) of the associated members. Figure 1 provides an illustration of a very simple three dimensional cube on `Store`, `Time` and `Product`. Here, each unique combination of dimension members represents a unique aggregation on the measure. For example, we can see that Product OD923 was purchased 78 times at Store MQ15 in January (assuming a `Count` measure).

Note, as well, that each dimension is associated with a distinct aggregation hierarchy. Stores, for instance, are organized in `Country` $\rightarrow$ `Province` $\rightarrow$ `City` groupings. Referring again to Figure 1, we see that `Product Number` is the lowest or *base* level in the Product dimension. In practice, data is physically stored at the base level so as to support run-time aggregation to coarser hierarchy levels. Moreover, the attributes of each dimension are partially ordered by the dependency relation $\preceq$ into a dependency lattice [19]. For example, `Product Number` $\preceq$ `Type` $\preceq$ `Category` within the Product dimension. More formally, the dependency lattice is expressed in Definition 1.

Definition 1: A dimension hierarchy H_i of a dimension D_i, can be defined as $H_i = (L_0, L_1, \ldots, L_j)$ where L_0 is the lowest level and L_j is the highest. There is a functional

dependency between L_{h-1} and L_h such that $L_{h-1} \preceq L_h$ where $(0 \leq h \leq j)$.

Finally, we note that there are in fact many variations on the form of OLAP hierarchies [20] (e.g., symmetric, ragged, non-strict). Regardless of the form, however, traversal of these aggregation paths — typically referred to as *rollup and drill down* — is perhaps the single most common query form. It is also central to the techniques discussed in this paper.

A. Native Language Object Oriented OLAP Queries

The cube representation, as described above, is common to most OLAP query environments and represents the user's conceptual view of the data repository. That being said, it can be difficult to implement the data cube using standard relational tables alone and, even when this is possible, performance is usually sub-par as relational DBMSs have been optimized for transactional processing. As a result, most OLAP server products either extend conventional relational DBMSs or build on novel, domain specific indexes and algorithms.

In our own case, the authorization methods described in this paper are part of a larger project whose focus is to design, implement and optimize an OLAP-specific DBMS server. A key design target of this project is the integration of the conceptual cube model into the DBMS itself. This objective is accomplished, in part, by the introduction of an OLAP-specific algebra that identifies the core operations associated with the cube (SELECT, PROJECT, DRILL DOWN, ROLL UP, etc). In turn, these operations are accessible to the client side programmer by virtue of an Object Oriented API in which the elements of the cube (e.g., cells, dimensions, hierarchies) are represented in the *native* client language as simple OOP constructs. (We note that our prototype API uses Java but any contemporary OO language could be used). To the programmer, the cube and all of its data — which is physically stored on a remote server and may be Gigabytes or Terabytes in size — appears to be nothing more than a local in-memory object. At compile time, a fully compliant Java pre-parser examines the source code, creates a parse tree, identifies the relevant OLAP objects, and re-writes the original source code to include a native DBMS representation of the query. At run-time, the pre-compiled queries are transparently delivered to the back end analytics server for processing. Results are returned and encapsulated within a proxy object that is exposed to the client programmer.

As a concrete example, Listing 1 illustrates a simple SQL query that summarizes the total sales of Quebec's stores in 2011 for the data cube depicted in Figure 1. Typically, this query would be embedded within the application source code (e.g., wrapped in a JDBC call). Conversely, Listing 2 shows how this same query could be written in an Object-Oriented manner by a client-side Java programmer. Note

```
Select Store.province, SUM(sales)
From   Store, Time, Sales
Where Store.store_ID = Sales.store_ID AND
 Time.time_ID = Sales.time_ID AND
 Time.year = 2011 AND
 Store.province = 'Quebec'
Group by  Store.province
```

Listing 1. Simple SQL OLAP Query

```
Class SimpleQuery extends OLAPQuery{
 Public boolean select(){
  Store store = new Store();
  DateDimension time = new TimeDimension();
  return (time.getYear() == 2011 &&
      store.getProvince() == 'Quebec');
 }
 Public Object[] project(){
  Store store = new Store();
  Measure measure = new Measure();
  Object[] projections = {
   store.getProvince(),
   measure.getSales()};
  return projections;
 }}
```

Listing 2. An Object Oriented OLAP Query

that each algebraic operation is encapsulated within its own method (in this case, SELECT and PROJECT), while the logic of the operation is consolidated within the `return` statement. It is the job of the pre-parser to identify the relevant query methods and then extract and re-write the logic of the `return` statement(s). Again, it is important to understand that the original source code will never be executed directly. Instead, it is translated into the native operations of the OLAP algebra and sent to the server at run-time.

While it is outside the scope of this paper to discuss the motivation for native language OLAP programming (a detailed presentation can be found in a recent submission [21]), we note that such an approach not only simplifies the programming model, but adds compile time type checking, robust re-factoring, and OOP functionality such as query inheritance and polymorphism. Moreover, query optimization is considerably easier on the backed as the DBMS natively understands the OLAP operations sent from the client side. In the context of the current paper, however, the significance of the query transformation process is that the authorization elements (e.g. roles and permissions) will be directly associated with the operations of the algebra. In fact, it is this algebraic representation that forms the input to the authorization module presented in the remainder of the paper.

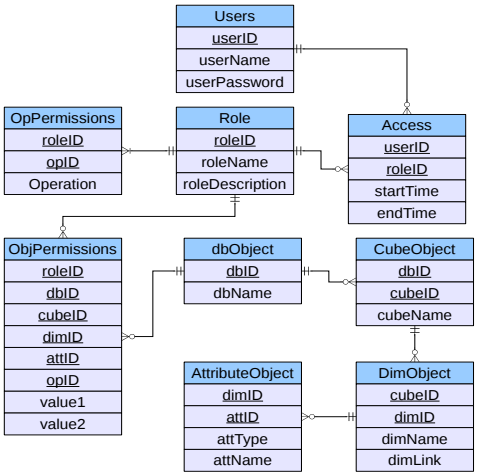


Figure 2. The Authorization DB.

IV. Authentication and Authorization

Without sufficient security countermeasures, open access to the OLAP repository becomes a powerful tool in the hands of malicious or unethical users. *Access Control* is the process that restricts unauthorized users from compromising protected data. This process can be thought of as occurring in two basic phases: **Authentication** and **Authorization**. Authentication is a form of *identity verification* that attempts to determine whether or not a user has valid credentials to access the system. In contrast, Authorization refers to the process of determining if the user has permission to access a specific data resource. In this section, we will describe our general framework, giving a detailed description of its two primary components and the relationship between them.

A. The Authentication Module

The authentication component is responsible for verifying user credentials against a list of valid accounts. These accounts are provided by the system administrator and are kept — along with their constituent permissions — in a backend database (i.e., the Authorization DB). The Authorization DB consists of a set of tables (`users`, `permissions`, and `objects`) that collectively represent the meta data required to authenticate and authorize the current user. For example, the `users` table stores basic user credentials (e.g., name, password), while the `permissions` table records the fact that a given user(s) may or may not access certain controlled objects. Figure 2 illustrates a slightly simplified version of the Authorization DB schema. In the current prototype, storage and access to the Authorization DB is provided by the SQLite toolkit [22]. SQLite is a small, open source C language library that is ideally suited to tasks that require basic relational query facilities to be embedded within a larger software stack.

```
<?xml version='1.0' encoding='UTF-8'?>
<DOCTYPE QUERY SYSTEM "ClientQuery.dtd" []>
<QUERY><DATA_QUERY>
  <CUBE_NAME>Furniture Sales</CUBE_NAME>

 <OPERATION_LIST>
  <OPERATION><PROJECTION>
    <MEASURE_LIST><MEASURE>Sales</MEASURE></MEASURE_LIST>
    <ATTRIBUTE_LIST>
     <PROJECTION_DIMENSION><DIMENSION_NAME>Store</DIMENSION_NAME>
       <ATTRIBUTE>Province</ATTRIBUTE>
       .....
  <OPERATION> <SELECTION>
    <DIMENSION_LIST><COMPOUND_DIMENSION> <DIMENSION_LIST>
      <DIMENSION><DIMENSION_NAME>Store</DIMENSION_NAME>
       <EXPRESSION><RELATIONAL_EXP><BASIC_EXP><SIMPLE_EXP><EXP_VALUE>
         <ATTRIBUTE>Province</ATTRIBUTE></EXP_VALUE></SIMPLE_EXP>
          <COND_OP><RELATIONAL_OP>EQUALS</RELATIONAL_OP></COND_OP>
          <SIMPLE_EXP><EXP_VALUE><CONSTANT>Quebec</CONSTANT> .....
     <LOGICAL_OP>AND</LOGICAL_OP>
     <DIMENSION><DIMENSION_NAME>Time</DIMENSION_NAME>
      <EXPRESSION><RELATIONAL_EXP><BASIC_EXP><SIMPLE_EXP><EXP_VALUE>
        <ATTRIBUTE>Year</ATTRIBUTE></EXP_VALUE></SIMPLE_EXP>
         <COND_OP><RELATIONAL_OP>EQUALS</RELATIONAL_OP></COND_OP>
           <SIMPLE_EXP><EXP_VALUE><CONSTANT>2011</CONSTANT>
            .....
   </OPERATION_LIST>
 <USER_CREDENTIALS>
  <USER_NAME>John</USER_NAME>
  <PASSWORD>J86mn</PASSWORD>
 </USER_CREDENTIALS>
</QUERY>
```

Figure 3. An XML query segment.

Internally, the user's transformed OLAP query is represented in XML format (embedded within the re-written source code). To validate the received XML query, the system relies on a Document Type Declaration (DTD) grammar [23] that is used to describe the structure of the expected XML query (We note that the somewhat more expressive XMLSchema can also be used for this purpose). The grammar itself is quite large but, ultimately, its purpose is to represent the functionality of the analytics queries one would expect to see in a Business Intelligence context. Figure 3 shows an XML-encoded segment of the query depicted in Listing 2. With a little effort one can see how the "total sales in 2011 for Quebec stores" is captured by the sequence of nested XML

The user query itself can be divided into three main parts: CUBE NAME, OPERATION LIST, and USER CREDENTIALS. As one would expect, the CUBE NAME element simply indicates the cube from which data is to be retrieved (the DBMS would likely store multiple cubes). The OPERATION LIST element contains one or more OPERATION elements, with PROJECTION and SELECTION being by far the most common (other analytics operations include CHANGE LEVEL, CHANGE BASE, PIVOT, DRILL ACROSS, UNION, DIFFERENCE, and INTERSECTION). In short, the PROJECTION element lists all attributes and measures the user wants to retrieve (e.g., Store.Province, and SUM(Sales)). The SELECTION element, in turn, limits or filters the data fetched from the data cube. Each SELECTION element consists of one or more criteria combined by

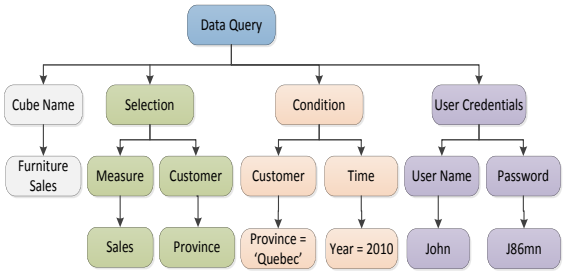


Figure 4. A small Parse Tree fragment.

a LOGICAL OP element (e.g., Store.Province = 'Quebec' AND Time.Year = 2011). Finally, the USER CREDENTIALS element, as the name indicates, contains the user's authentication identifiers (i.e., the user name and password).

Of course, in order to properly authenticate the query, it must first be parsed and decomposed into its algebraic components. In fact, the parsing is done in two phases. First, the DOM parser utility is used to produce a DOM tree that represents the raw contents of the XML document. In this phase, the parser not only builds the tree but also verifies that the received query has valid syntax corresponding to the DTD query grammar. An XML document is considered as valid if it contains only those elements defined in the DTD. If the query is syntactically valid, the query proceeds to the second phase. Otherwise, a parsing error message is returned to the user.

Figure 4 shows the node tree corresponding to the query depicted in Figure 3. We can easily see that the content of this parse tree is equivalent to the OLAP query represented in the XML format. Specifically, it is executed against the cube Furniture Sales and consists of two OLAP operations (Projection and Selection). The projection operation returns the dimension attribute Customer.Province, as well as one measure attribute — Sales. The Selection operation filters the returned information via two conditions on the dimensions Customer (i.e., Province = Quebec) and Time (Year = 2010). The user name "John" and the password "J86mn" represent the user credentials.

In the second phase of the process, the DOM tree is converted into a simplified data structure. This "Query Object" is cached in memory and contains all the query elements (i.e., returned attributes, query conditions along with its dimensions and attributes, and user credentials). The purpose of this final conversion process is to transform the user query into a simple, minimal data structure that represents the query in a compact but expressive form.

Once the parsing is completed, the Authentication module extracts the user credentials to verify them against a valid account stored in the Authorization DB. If the verification is successful, the DBMS proceeds with the authorization process. Otherwise, the query is rejected and the user/programmer is notified. The upper part of Figure 5 depicts the processing logic of the Authentication module. As a final note, we add that the prototype for the authentication and authorization modules has been designed as a third party component that can interact with existing DBMS products. As such, it does not maintain connection-oriented session data and thus requires authentication information to be provided for each query. That being said, this has a very limited impact on performance as the bulk of the processing logic is associated with the authorization module, which must assess user privileges on a query by query basis.

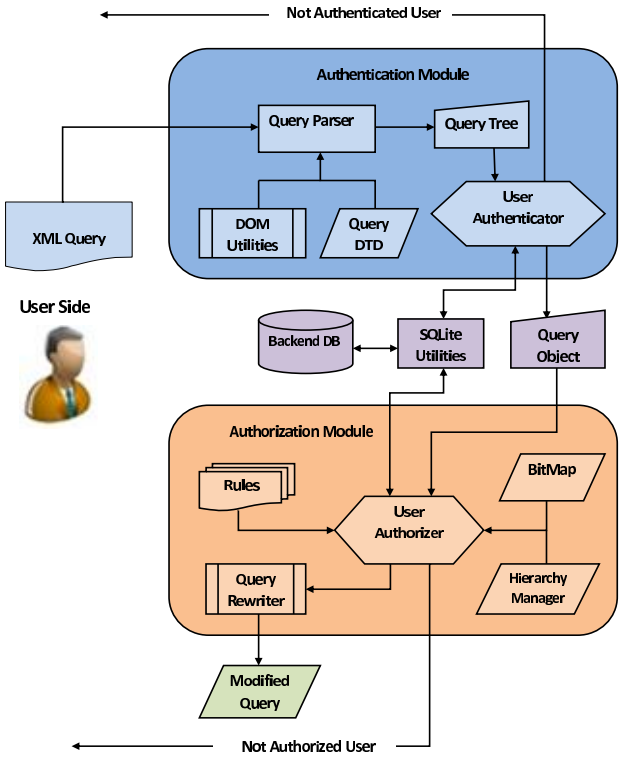


Figure 5. Authentication and Authorization.

B. The Authorization Module

The second — and more significant — phase is authorization, the process of determining if the user has permission to access specific data elements. Specifically, when a user requests access to a particular resource, the request is validated against the *permitted resource list* assigned to that user in the backend database. If the requested resource produces a valid match, the user request is allowed to execute as originally written. Otherwise, the query will either be rejected outright or modified according to a set of flexible transformation rules. To decide if the query will be modified or not, we rely on a set of *authorization objects* against which the rules will be applied. The rules themselves will be discussed in Section IV-E. The lower portion of

Figure 5 graphically illustrates the Authorization module and indicates its interaction with the Authentication component.

C. Specifying Authorization Objects

Authorization is the granting of a right or privilege that enables a subject (e.g., users or user groups) to execute an action on an object. In order to make authorization decisions, we must first define the authorization objects. Note that the *objects* in the OLAP domain are different from those in the relational context. In a relational model, objects include logical elements such as tables, records within those tables, and fields within each record. In contrast, OLAP objects are elements of the more abstract conceptual model and include the dimensions of the multi-dimensional cube, the hierarchies within each dimension, and the aggregated cells (or facts). In practice, this changes the logic or focus of the authorization algorithm. For instance, a user in a relational environment may be allowed direct access to a specific record (or field in that record), while an OLAP user may be given permission to dynamically aggregate measure values at/to a certain level of detail in one or dimension hierarchies. Anything below this level of granularity would be considered too *sensitive*, and hence should be protected. In fact, the existence of aggregation hierarchies is perhaps the most important single distinction between the authorization logic of the OLAP domain versus that of the relational world.

We note that in the discussion that follows, we assume an *open world* policy, where only prohibitions are specified. In other words, permissions are implied by the absence of any explicit prohibition. We use the open world approach for the simple reason that, in contrast to the users in operational database settings, OLAP users are typically drawn from a relatively small pool of enterprise decision makers. As such, these more senior employees generally require broad access to data. It therefore makes sense to use an open world policy that defines a relatively small set of constraints, rather than a closed world approach that would require extensive "positive" privileges to be defined. That being said, there is no theoretical barrier to the use of a closed world strategy.

Before discussing the authorization rules themselves, we first look at a pair of examples that illustrate the importance of proper authorization services in the OLAP domain. We begin with the definition of a policy for accessing a specific aggregation level in a data cube dimension hierarchy.

Example 1: An employee, Alice, is working in the Montreal store associated with the cube of Figure 1. The policy is simple: Alice should not know the sales totals of the individual provinces.

Clearly, Alice is prohibited from reading or aggregating data at the provincial level in the Store dimension hierarchy. However, in the absence of any further restrictions, it would still be possible for her to compute the restricted values from the lower hierarchies levels (e.g., City or Store_Number).

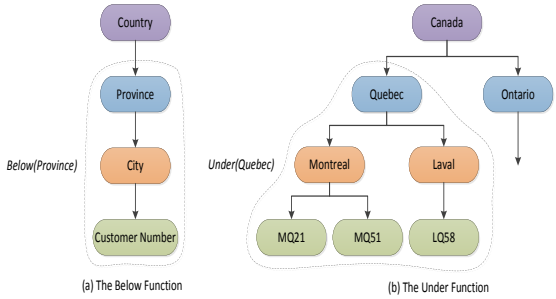


Figure 6. The Below and Under functions.

Ideally, the warehouse administrator should not be responsible for identifying and manually ensuring that all *implied* levels be included in the policy. Instead, our model assumes this responsibility and can, if necessary, restrict access to all *child* levels through the use of the *Below* function. As the name implies, this function returns a list consisting of the specified level L_i and all the lower levels of the associated dimension hierarchy. Figure 6(a) illustrates an example using a *Below(Province)* instantiation. Here, all levels surrounded by the dashed line are considered to be Authorization Objects, and thus should be protected. The formalization of the *Below* function is given by Definition 2.

Definition 2: In any dimension D_i with hierarchy H_i, the function Below(L_i) is defined as Below(L_i) = $\{L_j$: such that $L_j \preceq L_i$ holds$\}$, where L_i is the prohibited dimension level.

As shown in Example 1, a policy may restrict the user from accessing *any* of the values of a given level or levels. However, there are times when this approach is too coarse. Instead, we would like to also have a less restrictive mechanism that would only prevent the user from accessing a specific value *within* a level(s). For instance, suppose we want to alter the policy in Example 1 to make it more specific. The new policy might look like the following:

Example 2: Alice should not know the sales total for the province of Quebec.

In Example 2, we see that Alice may view sales totals for all provinces other than Quebec. However, Alice can still compute the Quebec sales by summing the sales of individual Quebec cities, or by summing the sales of Quebec's many stores. In other words, she can use the values of the lower levels to compute the prohibited value. Hence, all these values should also be protected. To determine the list of restricted member values, our model adds the *Under* function, which is formalized in Definition 3. Figure 6 (b) provides an example using *Under(Quebec)*. Here, all the values surrounded by the dashed line should be protected.

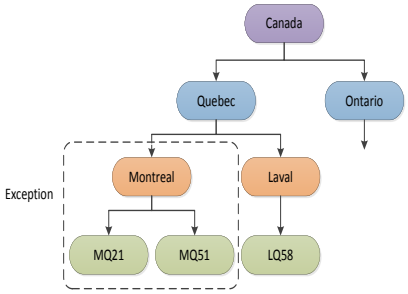


Figure 7. An Authorization Exception.

Definition 3: For any dimension hierarchy level L_i, and any attribute value V_i, the function Under(L_i, V_i) is defined as Under(L_i, V_i) = $\{V_j$: such that $V_j \preceq V_i$ holds$\}$, where L_i is the prohibited dimension level and V is the root value of the restriction.

Finally, it is also possible that *exceptions* to the general authorization rule are required. For instance, Alice should not know the sales of stores in the province of Quebec except for the stores in the city/region she manages (e.g., Montreal). Figure 7 graphically illustrates this policy. In this case, the circled members represent the values associated with the exception that would, in turn, be contained within a larger encapsulating restriction. Note that a user may have one or more exceptions on a given hierarchy. The formalization of the exception object is given in in Definition 4.

Definition 4: For any prohibited level L_i, there may be an Exception E such that E contains a set Ev of values belonging to Under(L_i). That is, Ev $\in$ values of Under(L_i).

To summarize, authorization objects consist of the values of the prohibited level and all the levels below it, excluding zero or more exception value(s). We formalize the concept of the *Authorization Object* in Definition 5.

Definition 5: An Authorization Object O = $\{$v : v $\in$ Under(L_i) - Ev$\}$, where L_i is the prohibited level, and Ev is the exception value.

D. Implementing Below and Under Functions

To efficiently implement Below and Under functions, a number of additional algorithms and data structures are needed in order to manipulate dimension hierarchies and to retrieve attribute values. These structures are initialized once the server receives a query and are subsequently exploited by the DBMS engine during query resolution. Below, we describe the core structures, along with the methods required to implement the associated functions efficiently.

1) Implementation of the Below Function: We begin by giving a brief description of the primary data structures utilized during function execution. The mapGraph is a suite

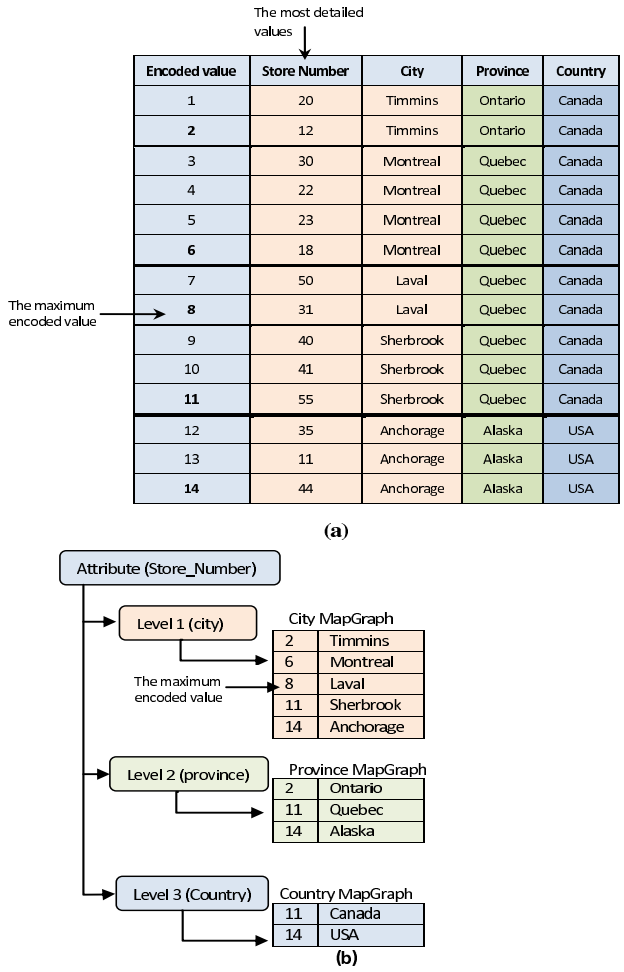


Figure 8. (a) The sorted data of the Store Dimension Table, (b) The corresponding mapGraph.

of algorithms and data structures for the manipulation of attribute hierarchies in "real time" [24]. mapGraph builds upon the notion of *hierarchy linearity* [25]. Briefly, a hierarchy is considered linear if there is a contiguous range of values R_j on dimension attribute A_j that may be aggregated into a contiguous range R_i. Informally, this implies that the totals for a range of values within a child aggregation level are equivalent to those of some range of values at the parent level. As a concrete example, the combined sales totals for the individual *months* of January, February, and March would be exactly equivalent to those of the first *quarter* of the calendar year. To establish the linearity of each dimension hierarchy a sorting technique is employed, with data subsequently stored at the finest level of granularity. If a Time hierarchy is present, for instance, transactional data would be stored at the Day level rather than at the Year level. A compact, in-memory lookup structure is then used to support efficient real time transformations between arbitrary levels of the dimension hierarchy. For example, Figure 8(a)

```
Select Product.Name, Store.province,
    Sum(sales)
From Product, Store, Sales
Where Product.product_ID = Sales.product_ID
  AND Store.store_ID = Sales.store_ID AND
   Store.Contry = 'Canada' AND
   (Product.Name = 'LN*' AND
    Product.price >= 24000)
Group by Product.Name, Store.province
Order by Product.Name, Store.province
```

Listing 3. Simple SQL OLAP Query

depicts the sorted data of the Store dimension table for the data cube depicted in Figure 1, while Figure 8(b) illustrates the corresponding mapGraph for the Store dimension hierarchy.

Each record in the mapGraph consists of two values — a native attribute representation (e.g., values of attribute Type in the Store dimension) and an integer value that represents the corresponding maximum encoded value in the primary attribute. We will look at a concrete example. While the city of Timmins has two stores, Store 1 and Store 2, the city of Montreal has four stores, Store 3 through Store 6. Using this structure, one can easily, and efficiently, perform a mapping from the most detailed encoded level value (i.e., Store_Number) to the corresponding sub-attribute value (i.e., attribute level values), and vice versa. For instance, Store 13 is located in the city of Anchorage and, as a consequence, in the State of Alaska in the USA (Alaska and the USA have a maximum Store_Number = 14).

While a number of commercial products and several research papers do support hierarchical processing for simple hierarchies, specifically those that can be represented as a balanced tree, mapGraph is unique in that it can enforce linearity on unbalanced hierarchies (i.e., optional nodes), as well as hierarchies defined by many-to-many parent/child relationships. The end result is that users may intuitively manipulate complex cubes at arbitrary granularity levels and can navigate easily through dimension levels.

Now recall the policy in Example 1. Suppose that Alice sent the query in Listing 3, which summarizes the total sales of stores in Canada for products of price 24K or more, and whose names start with "LN". To define the authorization objects, the Below function is invoked, taking the prohibited level (i.e., Store.Province) as an argument and using the mapGraph to retrieve a list consisting of the specified level and all the lower levels of the associated dimension hierarchy (i.e., Province, City, and Store Number). Clearly, the prohibited level is in the returned list, and as a result the query should be rejected.

2) Implementation of the Under Function: The Under function is invoked when the policy is less restrictive, as is the case in Example 2. Suppose that Alice now resends the query in Listing 3, assuming this less restrictive policy.

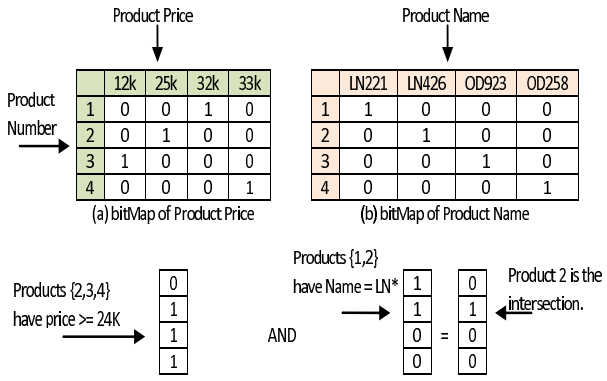


Figure 9. The bitMap of Product Price and Product Name.

To answer or reject the query, we have to determine if the user has requested access to the authorization objects. We note that the user query has two dimension conditions, the first on Product (Product.Name = 'LN*' AND Product.price $\geq$ 24000) and the second on Store (i.e., Store.Country = 'Canada'). The first condition will be ignored, since there is no restriction on the Product dimension in the current policy. For the second condition, we need to determine if the province of Quebec is in Canada (i.e., if Quebec is *Under* Canada). If so, we can say that the user has attempted to access a restricted data element and, as a consequence, the query should be rejected. By using the Under function, we retrieve the encoded values of Canada and Quebec from the mapGraph structure. If there is an intersection between the two, we know that Quebec is Under Canada. In our example, Canada has stores encoded with identifiers 1 through 11, and Quebec has stores encoded as 3 through 11. Clearly, there is an intersection between them, which means that the user has requested access to restricted data.

Definition 6: If there is an intersection between Under(L_i) values — where L_i is the prohibited level — and Under(E_j) values — where E_j is the requested level — then the query should not be executed directly.

As noted, the mapGraph is very useful when hierarchical attribute levels are involved in the OLAP query. However, in some cases, it is a non-hierarchical attribute that is restricted (e.g., the Name or Price attributes of Product). In this case, the FastBit [26] bitmap index structure allows us to easily find those records that contain specific values on a given attribute in the dimension. For example, suppose the Product dimension has four records (i.e., four products), numbered 1 through 4, and a non-hierarchy attribute (Product Price) is added to the Product dimension attributes. The bitmap index for the Product Price attribute is illustrated in Figure 9(a), while Figure 9(b) illustrates the bitmap index for the Product Name. Each index consists of four bit strings (number of products), each of length four. In each string, the 1's indicate the encoded values for the primary key.

```
Selection:
     Product.Name, Store.province, Sum(sales)
Condition:
     Store.Province = 'Quebec' AND
     (Product.Name = 'LN*' AND
      Product.price >= 24000)
From:
     Sales
```

Listing 4. A Query in Simple Form

Now, suppose that Alice is restricted from accessing all products whose names start with "LN". Further, we will assume that she resends the query in Listing 3. Since the Product Price and the Product Name are non-hierarchical attributes, we use their bitmap indexes to retrieve the base level numbers for those products, and then determine if there is an intersection between the two. Figure 9 illustrates how to identify those products whose Name starts with "LN" AND whose price $\geq$ 24K. The array at the lower left represents the products of price $\geq$ 24K, in this case Products 2, 3, and 4. The array in the center represents the products with names starting with "LN". Products 1 and 2 are identified in this case. The AND operator determines the intersection between them, with the final result shown in the last array. As we can see, there is in fact a non-empty intersection (i.e., Product_Number 2 has a price $\geq$ 24K and a name starts with "LN"); thus, the query should be rejected.

Algorithm 1 summarizes the logic of the checking process. In short, we determine if the attributes within the selection predicate(s) are hierarchical in nature. For example, a restriction on a time value (e.g., Day-Month-Year) would be hierarchical in nature, while a restriction on an attribute such as colour or weight would not be associated with any form of hierarchy. Based upon this understanding, query values would be analyzed relative to the information contained in the mapGraph data structure (which stores hierarchical relationship information) or bitmap indexes.

E. Authorization Rules

We now turn to the query authorization process itself. As noted above, pre-compiled queries are encoded internally in XML format. For the sake of simplicity (and space constraints), we will depict the received queries in a more compact form in this section. For example, Listing 4 represents the same query shown in Listing 3. Note that the query is divided into three elements: the SELECTION element, the CONDITION element, and the FROM element. The SELECTION element lists all attributes and measures the user wants to retrieve. The CONDITION element, in turn, limits or filters the data we fetch from the cube. Finally, the FROM element indicates the cube from which data is to be retrieved.

In the discussion that follows, we will assume the existence of a cube corresponding to Figure 1. That is, the

input : The policy condition S, and the Query Q
output: Returns True if Q is valid, False otherwise

Initialize the mapGraph (hM) and the bitMap (fB) if they have not already been initialized;
Let QA be the query attributes;
if *S has a hierarchy attribute* **then**
  Let SR be the range of S using hM;
end
else
  Let SR be the range of S using fB;
end
foreach *attribute* a_i *in QA* **do**
  if a_i *is hierarchy attribute* **then**
    Get the range of a_i QR using hM;
    if $QR \cap SR \neq \emptyset$ **then**
      Return False;
    end
  end
  else if a_i *is non-hierarchy attribute* **then**
    Get the range of a_i QR using fB;
    if $QR \cap SR \neq \emptyset$ **then**
      Return False;
    end
  end
end
Return True;

Algorithm 1: The procedure of Policy Class 2

cube has three dimensions (Product, Store, and Time). Dimension hierarchies include Product_Number $\preceq$ Type $\preceq$ Category for Product, Store_Number $\preceq$ City $\preceq$ Province $\preceq$ Country for Store, and Month $\preceq$ Year for Time. Selection operations correspond to the identification of one or more cells associated with some combination of hierarchy levels.

One of the advantages of building directly upon the OLAP conceptual model and its associated algebra is that it becomes much easier to represent, and subsequently assess, authorization policies. Specifically, we may think of policy analysis in terms of Restrictions, Exceptions, and Level Values that form a bridge between the algebra and the Authorization DB. There are in fact four primary *policy classes*, as indicated in the following list:

1) L_i Restriction + No Exception
2) L_i Restriction + Exception
3) Restriction on a specific value P of level L_i + no Exception
4) Restriction on a specific value P of level L_i + Exception

As mentioned, the query must be validated before execution. If validation is successful, then it can be executed as originally specified. Otherwise, the query is either rejected or rewritten according to a set of *transformation rules*. In the remainder of this section, we describe the four policy

```
Selection:
    Store.City, Product.Type, SUM(sales)
Condition:
    Time.year = 2011 AND
    Store.Country = 'Canada' AND
    Product.Category = 'Furniture'
From:
    Sales
```

Listing 5. Authorization Strategy as per Rule 1

```
Selection:
    Store.province, Product.Type, SUM(sales)
Condition:
    Time.year = 2011 AND
    Store.City = 'Montreal' AND
    Product.Category = 'Furniture'
From:
    Sales
```

Listing 6. Authorization Strategy as per Rule 4

classes and the processing logic relevant to each.

1) ***Policy Class 1****: L_i Restriction + No Exception:* If a user is prohibited from accessing level L_i and the user has no exception(s), then the authorization objects consist of the values of level L_i and all the levels below it. In short, this means that if the user query specifies level L_i or any of its children in the SELECTION element, then the query should simply be rejected. Moreover, if any *value* belonging to the L_i level or any of its children is specified in the CONDITION element of the query, the query should also be rejected. The formalization of the rule and an illustrative example is given below.

Rule 1. *If a user is prohibited from accessing the values of level L_i, and there is no exception, then the Authorization Objects (O) = $\{v : v \in Below(L_i)\}$.*

Example 3: If Alice sends the query depicted in Listing 5, which summarizes the total sales of Canada's stores in 2011 for furnitures products, and she is restricted from accessing/reading provincial sales, the query should be rejected.

Why is this query rejected? Recall that Alice is restricted from accessing provincial sales. Consequently, we see that an implicitly prohibited child level (i.e., City) is a component of the SELECTION element. So, if we allow this query, Alice can in fact compute the provincial sales by summing the associated city sales.

2) ***Policy Class 2****: L_i Restriction + Exception:* In this case, the authorization objects that should be protected consist of the prohibited level value and all values below it, *except* of course for the value of the exception or any value under it. Let us first formalize this case, before proceeding with a detailed description.

Rule 2. *If a user is restricted from accessing the values of level L_i, and the user has an exception E, then the Authorization Objects (O) = $\{v : v \in Below(L_i)$ - $Under(Ev)\}$.*

As such, when a user is prohibited from accessing the L_i level — excluding the *exception* values — then the query can be (i) allowed to execute, or (ii) modified before its execution. Let's look at these two cases now.

Rule 3. *The query will be allowed to execute without modification* if *the prohibited level value Lv or any of its more granular level values in (Below(L_i)) exists in the* CONDITION *element AND is equal to the exception value (Ev) or any of its implied values in (Under(Ev)).*

Example 4: Suppose that we have the following policy: Alice is restricted from accessing provincial sales *except* the sales for Canadian provinces. If Alice resubmits the query in Listing 4, it will now be executed without modification because the prohibited value (e.g., Quebec) is *under* the exception value (e.g., Under(Canada)).

But what if Alice has an exception value only for a more detailed child level of L_i (e.g., the city of Montreal)? In this case, if Alice submits the previous query, it should now be modified by replacing the restricted value (e.g., Quebec) in the CONDITION element with the exception value (e.g., Montreal). In this example, Alice gets only the values that she is allowed to see. The modified query is depicted in Listing 6. Rule 4 gives the formalization of this case.

Rule 4. *If the prohibited level value Lv or any of its more granular level values (Under(Lv)) exists in the* CONDITION *element, and the exception value belongs to this set of values, then the query should be modified by replacing the prohibited value with the exception value.*

In addition to the scenario just described, the query can also be modified by adding a new predicate to the CONDITION element when the prohibited level or any of its child levels exists in the SELECTION element only.

Rule 5. *If the prohibited level Lv or any of its more granular levels (Below(L_i)) exists in the* SELECTION *element only, then the query should be modified by adding the exception E as a new predicate to the query.*

Example 5: Suppose that Alice sends the query depicted in Listing 7. In this case, the query will be modified by adding a new predicate (i.e., Store.Province = 'Quebec'), because the prohibited level (i.e., City) exists in the SELECTION element. After the modification, Alice will see only the cities of Quebec. The modified query is depicted in Listing 8.

The complete processing logic for Policy Class 2 (i.e., Rule 3, Rule 4, and Rule 5) is encapsulated in Algorithm 2. Essentially, the algorithm takes the prohibited level L_i and the exception E as input and produces as output an authorization decision to execute or modify the query. The

```
Selection:
 Store.City, Product.Type, SUM(sales)
Condition:
 Time.Year = 2011 AND
 Product.Type = 'Indoor'
From:
 Sales
```

Listing 7. Simple OLAP Query 2

```
Selection:
 Store.City, Product.Type, SUM(sales)
Condition:
 Time.Year = 2011 AND
 Product.Type = 'Indoor' AND
 Store.Province = 'Quebec'
From:
 Sales
```

Listing 8. Authorization Strategy as per Rule 5

process is divided into two main parts or conditions. In the first case, we are looking at situations whereby the prohibited level L_j exists in the query CONDITION element. Here, the query can either be allowed to execute directly or further modified. It executes directly if the prohibited value *Lv* is equal to the exception value *Ev* or any value under *Ev*. However, if the exception value *Ev* is equivalent to any value under *Lv*, then the query is modified by replacing the prohibited level with the exception level AND the prohibited level value with the exception value.

In the second case, we target the scenario whereby the prohibited level L_j exists in the SELECTION element only. Here, we modify the original query by adding the exception *E* as a new condition.

3) ***Policy Class*** *3: Restriction on a specific value P of level L_i + no Exception:* We now turn to the classes in which specific values at a given level are restricted, as opposed to all members at a given level. We begin with the simplest scenario.

Rule 6. *If a user is prohibited from accessing a specific value P of level L_i, and the user has no exceptions, then the Authorization Objects(O)= $\{v : v \in P \cup Under(P)$ where P is the prohibited value$\}$.*

Here, the prohibited value *P*, or some value under *P*, exists in the query CONDITION element. As per Rule 6, the query should simply be rejected. But what if L_i exists in the SELECTION element only? In this case, the query should be modified by adding the prohibited value as a new predicate to the query CONDITION element. Let's look at the following example.

Example 6: Suppose that Alice is restricted from accessing Quebec's sales. If Alice sends the query depicted in Listing 9, the query should be modified as shown in Listing 10.

input : The prohibited level L_i and the exception *E*
output: Decision to directly execute or modify

Let Ev = *E* value;
foreach *level $L_j \in Below(L_i)$* **do**
  if *L_j exists in the query* CONDITION *element* **then**
    Let Lv = L_j value;
    if *Lv == Ev OR Lv ∈ Under(Ev)* **then**
      Allow the query to execute without modification;
    end
    else if *Ev* ∈ Under(Lv) **then**
      Replace *E* by L_j, and *Ev* by *Lv*, then inform the user, and allow the query to execute;
    end
  end
  else if *L_j exists only in the query* SELECTION *element* **then**
    Add *E* as new condition to the user query, inform the user, and allow the query to execute;
  end
end

Algorithm 2: The procedure of Policy Class 2

```
Selection:
   Store.Province, SUM(sales)
Condition:
   Time.year = 2011 AND
   Product.Type = 'Outdoor'
From:
   Sales
```

Listing 9. Simple OLAP Query 3

The associated query summarizes the sales of provinces in 2011 for outdoor products. As noted, the SELECTION element contains the prohibited level (Province), so instead of rejecting the query we modify it by adding a new predicate to the condition. The modified query returns only the sales that Alice is allowed to see. The logic is formalized in Rule 7 below.

Rule 7. *If the prohibited level L_i exists in the* SELECTION *element only, then the query should be modified by adding a new predicate to the query* CONDITION *element.*

4) ***Policy Class 4****: Restriction on a specific value P of level L_i + Exception:* Finally, we add an exception to the queries described by Class 3. Here, the relevant authorization objects consist of the prohibited value *(P)*, *minus* the exception values.

Rule 8. *If a user is restricted from accessing a value P of level L_i, and the user has an exception E, then the*

```
Selection:
    Store.Province, SUM(sales)
Condition:
    Time.year = 2011 AND
    Product.Type = 'Outdoor' AND
    Store.Province != 'Quebec'
From:
    Sales
```

Listing 10. Authorization Strategy as per Rule 7

```
Selection:
    Store.City, Product.Type, SUM(sales)
Condition:
    Store.City = 'Montreal' AND
    Product.Type = 'Indoor' AND
    Time.Year = 2011
From:
    Sales
```

Listing 11. Authorization Strategy as per Rule 9

Authorization Objects(O)= $\{v : v \in (P \cup Under(P))$ - $(Ev \cup Under(Ev))\}$ where P is the prohibited value and E is the exception.

In this scenario, the query can either be allowed to execute or modified according to the following associated rules.

Rule 9. *The query will be allowed to execute, if the prohibited value Lv exists in the* CONDITION *element AND is equal to the exception value Ev or any value Under(Ev).*

Example 7: Suppose that Alice is restricted from accessing the sales of Canadian provinces, *except* for the sales of Quebec. If Alice sends the Query depicted in Listing 11, the query will be allowed to execute since the prohibited value (i.e., Montreal) is under the exception value (i.e., Quebec).

Rule 10. *If the prohibited level L_i exists in the query* SELECTION *element only, the query will be modified by adding the exception E as a new predicate. In principle, this rule is similar to Rule 4.*

Rule 11. *When Lv exists in the query* CONDITION *element AND Lv is under Ev, the query is modified by replacing the prohibited level L_i by the exception level E AND the prohibited level value Lv by the exception value Ev.*

Algorithm 3 illustrates the full processing logic for Policy Class 4 (Rule 8, Rule 9, Rule 10, and Rule 11). In short, the authorization module takes the prohibited level value *P* and the exception *E* as input and gives as output an authorization decision to execute or modify the query. The algorithm is again divided into two main parts. The first component targets the case whereby the prohibited value *P* exists in the query CONDITION element. Here, the query can be modified or executed directly. If the prohibited value belongs to the set of values under *E* , the query is modified by replacing the condition that contains the prohibited value by a new one containing the exception. Conversely, the query is allowed to execute directly if the prohibited level value *Lv* belongs to the values *Under(P)* AND *Lv* is equal to the exception value *Ev* OR *Ev* belongs to the values *Under(Lv)*.

In the second case, a new condition (exception *E*) is added to the query CONDITION element when the prohibited level *Lv* or any level below it *Below(Lv)* exists in the SELECTION element only.

input : The prohibited value *P* of level L_i and the exception *E*
output: Decision to directly execute or modify

Let Ev = *E* value;
foreach *level* $L_j \in Below(L_i)$ **do**
  if L_j *exists in the query* CONDITION *element* **then**
    Let Lv = L_j value;
    if *(Lv == P) AND (P ∈ Under(Ev))* **then**
      Add E as a new condition instead of the condition that contains L_j, inform the user, and allow the query to execute;
    end
    else if *(Lv ∈ Under(P)) AND (Lv == Ev OR Ev ∈ Under(Lv))* **then**
      Allow the query to execute without modification;
    end
  end
  else if L_j *exists only in the query* SELECTION *element* **then**
    Add E as new condition to the user query, inform the user, and allow the query to execute;
  end
end

Algorithm 3: The procedure of Policy Class 4

F. Authorization Rule Summary

The preceding sections have formalized the authorization framework in terms of four policy classes and their associated transformation rules. Below, we summarize the authorization decision in terms of its three possible outcomes — **Execute**, **Modify**, **Reject**:

1) The query is allowed to execute without modification in two situations:
   - Level L_i is restricted and there is an exception *E*:
     a) If any upper level exists in the SELECTION or PROJECTION query element, OR
     b) If the L_i value or any value from the levels below it exists in the CONDITION element AND this value is equal to the exception value *Ev* or any value under it.

- A specific value of L_i is restricted and there is an exception *E*:
  a) If the prohibited value *Lv* or any value under it exists in the CONDITION element AND it is equal to the exception value *Ev* OR any value under it.

2) The query is modified in one situation:
   - A level L_i is restricted and there is an exception *E*:
     a) If level L_i or any value from the levels below it exists in the query SELECTION element only, then we add the exception *E* as a new condition, OR
     b) If the exception value *Ev* belongs to the values under *Lv*, then we replace the prohibited level in the CONDITION element by the exception *E*.

3) The query is rejected in two situations:
   - A level L_i is restricted, and there is no exception:
     a) If level L_i or any value from a lower level exists in the SELECTION element only, OR
     b) If level L_i or any value from the levels below it exists in the CONDITION element.
   - A specific value *P* is restricted, and there is no exception:
     a) If *P* or any value under it exists in the CONDITION element.

V. EXPERIMENTAL RESULTS

Because of the potential to impact overall query resolution time, considerable effort has been made to ensure the efficiency of the authorization logic, including the exploitation of compact data structures such as mapGraph and the FastBit bitmap indexes. Moreover, the analysis of policy classes is based primarily upon a restricted set of IF/ELSE cases that, in turn, manipulate a small in-memory Authentication Database. Given the motivation to include OLAP-aware authorization mechanisms within fully functional database management systems, however, it is important to actually verify that our checking approach does not in fact seriously degrade query performance. As noted earlier, the authorization framework has been incorporated into a DBMS prototype specifically designed for OLAP storage and analysis. For testing purposes, however, this integrated environment is not necessarily ideal as it is difficult for the reader to determine if the balance between checking and execution is reflective of current systems. Furthermore, it may not be obvious that our authorization model has the potential for integration with standard database servers.

For this reason, we have coupled our framework with MonetDB, a popular open source database management system [27]. MonetDB is a column store DBMS, as opposed to the more familiar row-based systems. Column stores are particularly well suited to OLAP workloads as the ability to efficiently extract only the columns of interest can significantly improve IO performance. Note that in this case, the job of MonetDB is simply to provide execution services — all authorization services are provided by the subsystems defined in this paper. In the current case, we utilize the Star Schema Benchmark (SSB) [28], a variation of the original TPC-H benchmark augmented for OLAP settings. In short, SSB consist of a central Fact Table and four dimension tables, with a set of 13 analytics queries executed against the data. Queries are divided into four query categories, with each category providing increasingly sophisticated restrictions on the associated dimensions. A full listing of the queries can be found in the Appendix. The SSB is particularly valuable in the current context as it provides a common mechanism by which to assess the kinds of queries — in terms of both form and complexity — that one would actually expect to encounter in OLAP settings. As a final note, we stress that Monet does not provide an internal OLAP-aware conceptual model. To ensure compatibility with the mechanisms described throughput this paper, it was necessary to develop SQL conversion middleware, a significant research effort of its own. The details of the middleware architecture are the subject of an upcoming submission.

For the following tests, we have used the SSB generator (with default settings) to produce a Fact table of 180 million records, with each dimension housing between 60,000 and one million records. The experiments themselves were run on a 12-core AMD Opteron server with a CPU core frequency of 2100 MHz, L1/L2 cache size of 128K and 512K respectively, and a shared 12MB L3 cache. The server was equipped with 24 GB of RAM, and eight 1TB Serial ATA hard drives in a RAID 5 configuration. The supporting OS was CentOS Linux Release 6.0. All OS and DBMS caches were cleaned between runs.

A set of four simple but typical authorization policies was created, as follows. We generated one constraint across a full dimension (i.e., the Product.Part is restricted), a second constraint on an attribute, along with an exception (i.e., the attribute s_region is restricted with an s_province exception), a third constraint on an attribute value with an exception value (i.e., d_year $<$ 2009 is restricted except d_year = 2005 or 2006), and the last constraint prohibits access to a cuboid as a whole. Essentially, policies were designed in keeping with the logic of Section IV, but adapted to the specific attributes of the SSB schema.

In terms of the results, we have isolated each of the four query classes and show authorization processing versus the subsequent query execution time in Figure 10, Figure 11, Figure 12, and Figure 13. We note that all queries violated one or more security policies and that these violations were identified and appropriately processes by the authorization module (each authorization decision was manually verified

for correctness). For those that were not candidates for re-writing (i.e., they were simply rejected), the query execution time is still listed so as to give the reader a better sense of the relative balance between checking and execution. A few additional points are also worth noting. First, the ratio of checking time to execution time varies considerably, depending on the specification of the underlying query. In particular, many OLAP queries are *very* expensive to execute, given the amount of sorting and aggregation involved. In this case, Query Classes 1 and 3 have restrictive selection constraints (with the exception of the Query 3.1), thereby reducing the size of intermediate results and, in turn, dramatically limiting aggregation costs. Overall, execution times range from about half a second for Query Class 3 to more than 30 seconds for Query Class 2, where large intermediate results produce massive aggregation costs. As the database gets larger, of course, these execution times will continue to grow.

Second, the checking costs are quite modest, in the range of 100-400 milliseconds. More importantly, the size of the underlying database has no effect upon the checking costs, as only the cube meta data is inspected. In other words, it does not matter that 180 million records exist in the database as authorization decisions are not based upon this data. Rather, only schema information (e.g., cubes, dimensions, hierarchies) and policy specifications (i.e., restrictions and exceptions) are required during this process. In practice, it is extremely unlikely that, from an end user's perspective, authorization costs would have a tangible impact on database access and analysis.

As a final point, we re-iterate that column stores are well suited to this environment, given their ability to minimize I/O costs. The execution times for traditional row store database servers can be one to two orders of magnitude larger [29]. The authors have, in fact, evaluated the current test cases on the open source row-based PostgreSQL DBMS and validated these ratios. In such environments, the ratio of checking to execution costs would be far more extreme, with execution costs being dozens or even hundreds of times larger than checking costs.

VI. Conclusions and Future Work

In this paper, we have discussed a query re-writing model to provide access control in multi-dimensional OLAP environments. We began by defining a conceptual model that focused on the data cube and its constituent dimension hierarchies. From there we introduced the notion of authorization objects designed to identify and constrain the relationships between parent/child aggregation levels. We then presented a series of rules that exploited the authorization objects to decide whether user queries should be rejected, executed directly, or dynamically and transparently transformed. In the latter case, we identified a set of minimal changes that would allow queries to proceed against a subset of the requested data.

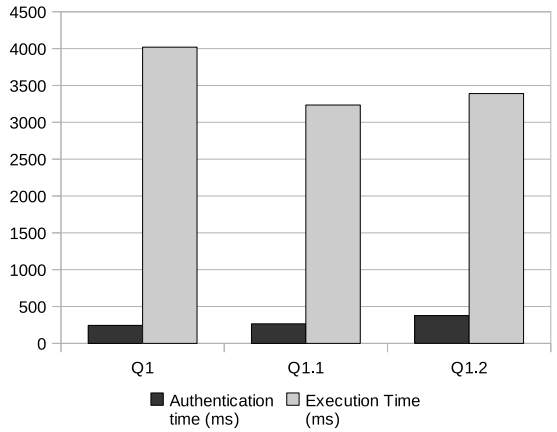


Figure 10. Performance for SSB schema, Query 1 category

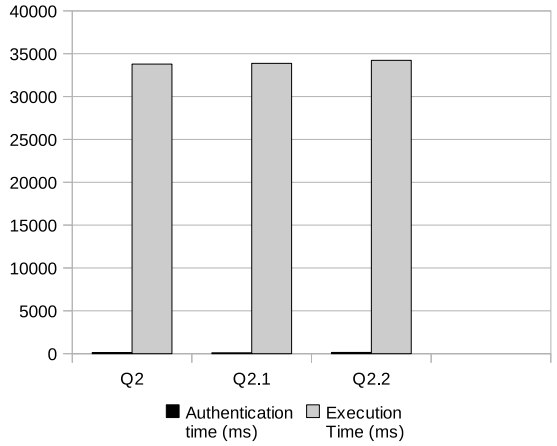


Figure 11. Performance for SSB schema, Query 2 category

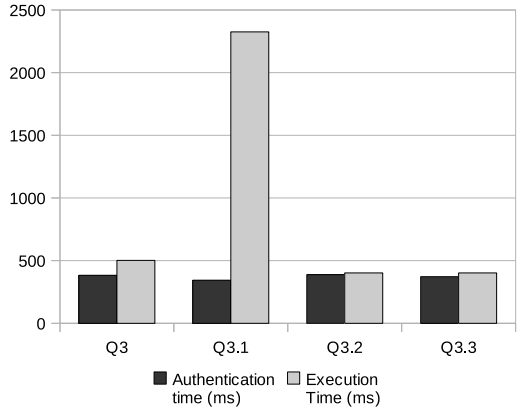


Figure 12. Performance for SSB schema, Query 3 category

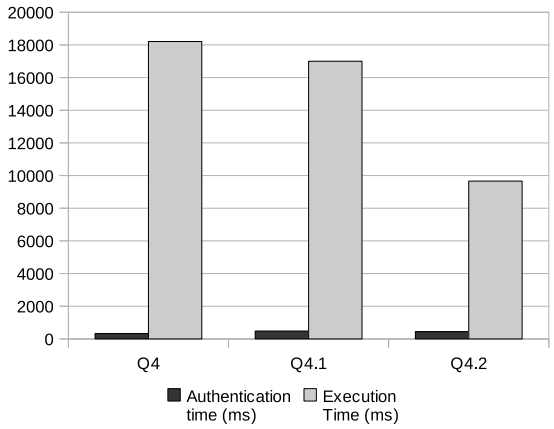


Figure 13. Performance for SSB schema, Query 4 category

While the authentication and authorization framework has been integrated into a prototype DBMS that provides OLAP-specific indexing and storage, we believe that the general principles are broadly applicable to any contemporary DBMS product. To this end, we combined the framework with MonetDB, an open source DBMS that provides efficient column oriented services. Using the Star Schema Benchmark, we showed that for common OLAP queries, authentication and authorization services represent a negligible impact on overall query execution and, in fact, that there is no relationship between authorization and execution costs. For this reason, we believe that our methods are viable for not only OLAP-specific database management systems, but more conventional platforms as well.

Finally, it is important to point out that the framework presented in this paper cannot block all attempts to access restricted data. In particular, it is possible for a user possessing some degree of external knowledge to combine the results of multiple *valid* queries to obtain data that is itself meant to be protected. We refer to such exploits as *inference* attacks. We are currently working on inference detection mechanisms that will piggy back on top of the core authentication and authorization framework to provide an even greater level of security for OLAP data.

REFERENCES

[1] T. Eavis and A. Altamimi, "OLAP authentication and authorization via query re-writing," in *The Fourth International Conference on Advances in Databases, Knowledge, and Data Applications (DBKDA)*, 2012, pp. 130–139.

[2] P. P. Griffiths and B. W. Wade, "An authorization mechanism for a relational database system," *ACM Transactions on Database Systems*, vol. 1, no. 3, pp. 242–255, Sep. 1976.

[3] Biba, "Integrity considerations for secure computer systems," *MITRE Co., technical report ESD-TR 76-372*, 1977.

[4] R. Sandhu, D. Ferraiolo, and R. Kuhn, "The NIST model for role-based access control: towards a unified standard," in *Proceedings of the fifth ACM workshop on Role-based access control*, ser. RBAC '00, 2000, pp. 47–63.

[5] E. Fernández-Medina, J. Trujillo, R. Villarroel, and M. Piattini, "Developing secure data warehouses with a UML extension," *Information Systems*, vol. 32, pp. 826–856, 2007.

[6] C. Blanco, I. G.-R. de Guzman, D. Rosado, E. Fernandez-Medina, and J. Trujillo, "Applying QVT in order to implement secure data warehouses in SQL Server Analysis Services," *Journal of Research and Practice in Information Technology*, vol. 41, pp. 135–154, 2009.

[7] J. Trujillo, E. Soler, E. Fernández-Medina, and M. Piattini, "An engineering process for developing secure data warehouses," *Information and Software Technology*, vol. 51, pp. 1033–1051, 2009.

[8] K. Khajaria and M. Kumar, "Modeling of security requirements for decision information systems," *SIGSOFT Software Engineering Notes*, vol. 36, no. 5, pp. 1–4, Sep. 2011.

[9] M. Kumar, A. Gosain, and Y. Singh, "Stakeholders driven requirements engineering approach for data warehouse development," *JIPS*, vol. 6, no. 3, pp. 385–402, 2010.

[10] Y. Singh, A. Gosain, and M. Kumar, "From early requirements to late requirements modeling for a data warehouse," *Networked Computing and Advanced Information Management, International Conference on*, vol. 0, pp. 798–804, 2009.

[11] N. Katic, G. Quirchmay, J. Schiefer, M. Stolba, and A. Tjoa, "A prototype model for data warehouse security based on metadata," in *DEXA*, 1998, pp. 300–308.

[12] A. Rosenthal and E. Sciore, "View security as the basic for data warehouse security," in *International Workshop on Design and Management of Data Warehouse*, 2000, pp. 8.1–8.8.

[13] ——, "Administering permissions for distributed data: factoring and automated inference," in *Proceedings of the fifteenth annual working conference on Database and application security*, ser. Das'01, 2002, pp. 91–104.

[14] A. Deshpande, Z. Ives, and V. Raman, "Adaptive query processing," *Foundations and Trends in Databases*, vol. 1, pp. 1–140, 2007.

[15] S. Rizvi, A. O. Mendelzon, S. Sudarshan, and P. Roy, "Extending query rewriting techniques for fine-grained access control," in *ACM Special Interest Group on the Management of Data*, ser. SIGMOD '04, 2004, pp. 551–562.

[16] "The Virtual Private Database," June 2012, http://www.oracle.com/technetwork/database/security/index-088277.html.

[17] "Microsoft Analysis Services," June 2012, http://www.microsoft.com/sqlserver/2008/en/us/analysis-services.aspx.

[18] J. Gray, A. Bosworth, A. Layman, D. Reichart, and H. Pirahesh, "Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals," *Data Mining and Knowledge Discovery*, vol. 1, pp. 29–53, 1997.

[19] V. Harinarayan, A. Rajaraman, and J. Ullman, "Implementing data cubes efficiently," in *ACM Special Interest Group on the Management of Data*, ser. SIGMOD '96, 1996, pp. 205–227.

[20] E. Malinowski and E. Zimányi, "Hierarchies in a multidimensional model: from conceptual modeling to logical representation," *Data and Knowledge Engineering*, vol. 59, pp. 348–377, 2006.

[21] T. Eavis, H. Tabbara, and A. Taleb, "The NOX framework: native language queries for business intelligence applications," in *Data Warehousing and Knowledge Discovery (DaWak)*, 2010, pp. 172–189.

[22] "SQL database engine," June 2012, http://www.sqlite.org.

[23] "Definition of the XML document type declaration from Extensible Markup Language (XML) 1.0 (Fifth Edition)," June 2012, http://www.w3.org/TR/xml/.

[24] T. Eavis and A. Taleb, "Mapgraph: efficient methods for complex olap hierarchies," in *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, ser. CIKM '07, 2007, pp. 465–474.

[25] V. Markl, R. Bayer, B. Forschungszentrum, and R. Bayer, "Processing relational OLAP queries with UB-Trees and multidimensional hierarchical clustering," in *In Proceedings of DMDW 2000*, 2000, pp. 5–6.

[26] M. Zaker, S. Phon-amnuaisuk, and S. cheng Haw, "An adequate design for large data warehouse systems: Bitmap index versus B-tree index," 2008.

[27] "MonetDB column store database engine," June 2012, http://www.monetdb.org.

[28] P. O'Neil, E. O'Neil, X. Chen, and S. Revilak, "Performance evaluation and benchmarking," R. Nambiar and M. Poess, Eds., 2009, ch. The Star Schema Benchmark and Augmented Fact Table Indexing, pp. 237–252.

[29] D. J. Abadi, S. R. Madden, and N. Hachem, "Column-stores vs. row-stores: how different are they really?" in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, ser. SIGMOD '08, 2008, pp. 967–980.

APPENDIX

Below, we provide a listing of the 13 queries found in the Star Schema Benchmark.

1. **select** sum(lo_extendedprice*lo_discount) as revenue
   from lineorder, date
   where lo_orderdate = d_datekey and
   d_year = 1993 and
   lo_discount between 1 and 3
   and lo_quantity < 25

   1. 1) **select** sum(lo_extendedprice*lo_discount) as revenue
      from lineorder, date
      where lo_orderdate = d_datekey and
      d_yearmonthnum = 199401 and
      lo_discount between 4 and 6 and
      lo_quantity between 26 and 35

   1. 2) **select** sum(lo_extendedprice*lo_discount) as revenue
      from lineorder, date
      where lo_orderdate = d_datekey and
      d_weeknuminyear = 6 and
      d_year = 1994 and
      lo_discount between 5 and 7 and
      lo_quantity between 26 and 35

2. **select** sum(lo_revenue), d_year, p_brand1
   from lineorder, date, part, supplier
   where lo_orderdate = d_datekey
   and lo_partkey = p_partkey and
   lo_suppkey = s_suppkey and
   p_category = 'MFGR#12' and
   s_region = 'AMERICA'
   group by d_year, p_brand1
   order by d_year, p_brand1

   2. 1) **select** sum(lo_revenue), d_year, p_brand1
      from lineorder, date, part, supplier
      where lo_orderdate = d_datekey and
      lo_partkey = p_partkey and
      lo_suppkey = s_suppkey and
      p_brand1 between 'MFGR#2221' and
      'MFGR#2228' and
      s_region = 'ASIA'
      group by d_year, p_brand1
      order by d_year, p_brand1

   2. 2) **select** sum(lo_revenue), d_year, p_brand1
      from lineorder, date, part, supplier
      where lo_orderdate = d_datekey and
      lo_partkey = p_partkey and
      lo_suppkey = s_suppkey and
      p_brand1 = 'MFGR#2221' and
      s_region = 'EUROPE'
      group by d_year, p_brand1
      order by d_year, p_brand1

3. **select** c_city, s_city, d_year, sum(lo_revenue) as revenue
   from customer, lineorder, supplier, date
   where lo_custkey = c_custkey and
   lo_suppkey = s_suppkey and
   lo_orderdate = d_datekey and
   c_nation = 'UNITED STATES' and
   s_nation = 'UNITED STATES' and

d_year >= 1992 and
d_year <= 1997
where c_city, s_city, d_year
where d_year asc, revenue desc

3. 1) **select** c_nation, s_nation, d_year, sum(lo_revenue) as revenue
from customer, lineorder, supplier, date
where lo_custkey = c_custkey and
lo_suppkey = s_suppkey and
lo_orderdate = d_datekey and
c_region = 'ASIA' and
s_region = 'ASIA' and
d_year >= 1992 and
d_year <= 1997
group by c_nation, s_nation, d_year
order by d_year asc, revenue desc

3. 2) **select** c_city, s_city, d_year, sum(lo_revenue) as revenue
from customer, lineorder, supplier, date
where lo_custkey = c_custkey and
lo_suppkey = s_suppkey and
lo_orderdate = d_datekey and
(c_city='UNITED KI1' or c_city='UNITED KI5') and
(s_city='UNITED KI1' or s_city='UNITED KI5') and
d_year >= 1992 and
d_year <= 1997
group by c_city, s_city, d_year
group by d_year asc, revenue desc

3. 3) **select** c_city, s_city, d_year, sum(lo_revenue) as revenue
from customer, lineorder, supplier, date
where lo_custkey = c_custkey and
lo_suppkey = s_suppkey and
lo_orderdate = d_datekey and
(c_city='UNITED KI1' or c_city='UNITED KI5') and
(s_city='UNITED KI1' or s_city='UNITED KI5') and
d_yearmonth = 'Dec1997'
group by c_city, s_city, d_year
order by d_year asc, revenue desc

4. **select** d_year, s_nation, p_category, sum(lo_revenue - lo_supplycost) as profit
from date, customer, supplier, part, lineorder
where lo_custkey = c_custkey and
lo_suppkey = s_suppkey and
lo_partkey = p_partkey and
lo_orderdate = d_datekey and
c_region = 'AMERICA' and
s_region = 'AMERICA' and
(d_year = 1997 or d_year = 1998) and
(p_mfgr = 'MFGR#1' or p_mfgr = 'MFGR#2')
group by d_year, s_nation, p_category
order by d_year, s_nation, p_category

4. 1) **select** d_year, c_nation, sum(lo_revenue - lo_supplycost) as profit
from date, customer, supplier, part, lineorder
where lo_custkey = c_custkey and
lo_suppkey = s_suppkey and
lo_partkey = p_partkey and
lo_orderdate = d_datekey and
c_region = 'AMERICA' and
s_region = 'AMERICA' and
(p_mfgr = 'MFGR#1' or p_mfgr = 'MFGR#2')
group by d_year, c_nation
order by d_year, c_nation

4. 2) **select** d_year, s_city, p_brand1, sum(lo_revenue - lo_supplycost) as profit
from date, customer, supplier, part, lineorder
where lo_custkey = c_custkey and
lo_suppkey = s_suppkey and
lo_partkey = p_partkey and
lo_orderdate = d_datekey and
c_region = 'AMERICA' and
s_nation = 'UNITED STATES' and
(d_year = 1997 or d_year = 1998) and
p_category = 'MFGR#14'
group by d_year, s_city, p_brand1
order by d_year, s_city, p_brand1

Verification with AVISPA to Engineer Network Security Protocols

Florian Kammüller, Glenford Mapp, Sandip Patel, and Abubaker Sadiq Sani
Middlesex University
Computer Communications Group
f.kammueller@mdx.ac.uk, mapp@mdx.a.cuk, sp1264@live.mdx.ac.uk, ss1234@live.mdx.ac.uk

***Abstract*—This paper summarizes work on formal mechanized verification of security protocols using Avispa, a model checker dedicated to security protocols. Avispa has been successfully used in various Master's projects. In this paper, we present two outstanding projects of quite different nature that highlight the spectrum of formal security protocol verification and lead us to a proposition of engineering practice for the development of secure protocols based on two main ideas (a) refactoring existing formalisations to prove adaptations of security protocols (b) compositional proof of new protocols allowing the combination and reuse of (parts of) existing formalisations of other protocols. This paper presents first Radius-SHA256, an adaptation of the Radius protocol for remote authentication for network access to the secure hash function SHA-256. Second, we present the Secure Simple Protocol which is an extension for security of a protocol developed at our university for next generation networks. Both protocols have been formalized in the Avispa model checker and security has been proved.**

***Keywords*-Security protocols, Model Checking, Cryptographic Hashes, Simple Protocol**

I. Introduction

Radius [19], [20], a remote authentication protocol used for building up secure communications of clients with networks via network access servers, uses the message digest function MD5, a hash function which has meanwhile been proven to have security weaknesses. By contrast, the hash function SHA-256 still remains unchallenged. Although seemingly straightforward and thus tempting, simply replacing MD5 by SHA-256 in the Radius protocol must be considered potentially harmful since authentication protocols are extremely sensitive to minor changes as the history of attacks shows. In December 2008, an attack on the SSL protocol has been demonstrated based on the previously discovered collisions of the MD5 hash function [16]. The engineers of that attack recommend the discontinuation of use of SSL based on MD5. Fortunately, for SSL the use of the hash function is already by design a choice point. For Radius, this flexibility is not yet established; this is the subject and one of the results of this paper. Triggered by the alarming history of attacks of security protocols, formal verification techniques have long been deemed to be a way out. We investigate whether Radius-SHA256 – our proposed adaptation of the Radius protocol – can provide better security guarantees than its original. To provide evidence based on mathematical rigor we use the Avispa model checker. Fortunately, we can rely on the rich data base of this tool providing a model of the original protocol. By adapting this model to our Radius-SHA256 and checking that the original security guarantees still hold we prove two things (a) that Radius-SHA256 is secure and (b) that the security guarantees have general validity, i.e. they can be carried over to protocols Radius-X for hashes X. The latter result corresponds to a reduction of Radius security to the security of the underlying hash function.

Model checking, a push-button technology for mathematical verification of finite state systems has been discovered to be a suitable tool for security analysis of authentication protocols, e.g. [7]. Ever since, this technology has proved to be useful for the engineering of secure protocols, e.g. for adaptation of the Kerberos protocols to mobile scenarios [6]. However, little attention has been given to investigate to what extent we can use known engineering techniques, like refactoring, reuse, and composition to help us engineer formal security verifications of protocols. This paper is to be seen as a first step towards such an engineering process. We mainly present two distinct and unrelated case studies on Avispa formalisations. The first one being the aforementioned Radius and the second one a new specially LAN-centric transport protocol called simple protocol (SP) developed in our research group [21] and extended here by security, i.e. authentication. As a second engineering exercise, we report on this secured version of the SP protocol. This exercise shows how a new development of a special purpose protocol can profit from a simultaneous modelling and analysis with a dedicated modelchecker like Avispa. The two case studies need not be related since they just serve as case studies for engineering security protocols with general engineering principles like refactoring, reuse and composition. Even though, there is a bridge between those seemingly unrelated projects: securing local servers and services. So the first project (Radius) looks at a server protocol and the other project (SP) looks at a secure, optimized and tunable protocol for local servers.

This paper is an extension of the conference paper [4] and is based on the Masters Theses of two of the authors [11], [14]. The extensions are a more verbose introduction to the SP protocol and the technical presentation of the Avispa encoding of SP. In this paper we first provide the prerequisites of this project: brief introductions to the

Radius protocol, the Simple Protocol, Avispa model checking, and hashes (Section II). From there, we develop our new version Radius-SHA256 by introducing its model in Avispa in detail (Section III) and illustrate how this model can be efficiently used to verify security goals (Section III-D). To illustrate that Modelchecking is also useful in the engineering of new protocols we show its application to the Simple Protocol (Section IV). We first give a motivation and deeper introduction to this protocol and its context for future networks thereby extending the original paper [4]. Next we show how this Simple Protocol can be extended step by step introducing cryptographic keys to add authentication and secure it. We finally offer conclusions and an outlook (Section V).

II. Background

A. Radius

One of the major issues with networks is their security and one response to this challenge are authentication protocols. Radius is a popular protocol providing security to communication channels. Radius stands for *Remote Authentication Dial in User Service* and serves to secure communication between *Network Access Servers* (NAS) and so-called Radius servers. Radius satisfies the AAA (Authentication, Authorization and Accounting) protocol standards in both local and roaming situations. In January 1997, Radius standards were first introduced in RFC 2058 and Radius accounting in RFC 2059. After that RFC 2138 and RFC 2139 were published and they made the previous RFC obsolete. They both were made obsolete in turn by RFC 2865 and RFC 2866 respectively. Following were updates by RFCs 2868, 3575, and 5080 [20].

Assume that there is an Internet service provider (ISP) and he has two NAS. A NAS allows a user to connect directly to the ISP's network and be accepted by a core router which directly connect with ISP's network backbone. When a user wants to access his services, he sends a request to the NAS which forwards the user request to the main server to check the supplied credentials. This process is called authentication.

After authentication, the NAS has to check the access list of the user and then decide which services are permitted to this user. The RADIUS server then replies to the NAS with Access Reject, Access Challenge, or Access Accept as illustrated in Figure 1. This information is forwarded by the Radius server to the NAS. This is called authorization. Once a user is authenticated and authorized successfully, the NAS creates a connection between the user and the main server through which both can exchange their information. This secure connection is called a session. All the information regarding the session will be saved by the NAS for its accounting purposes. It includes start time of session, termination time of session, size of total received and sent data, amongst other information for accounting.

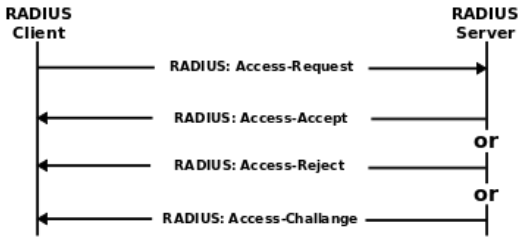


Figure 1. Radius access request and possible replies [18]

B. Simple Protocol

A new trend in next generation networks is the divergence between local area networks (LAN) and wide area networks (WAN) because there is still an increase of efficiency to be expected in LANs. Additionally, the ubiquity of computing devices and common usage of mobile devices asks for a flexibility that is better supported with fixed core networks and flexible wireless networks at the periphery. A reconsideration of the TCP/IP seems appropriate since adaptation of TCP to the often heterogeneous requirements of local wireless networks is not easy. The Simple Protocol (SP) [10] is intended to be used in combination with TCP but TCP for the WAN and SP for the LAN communication. SP is part of a wider development of the Y-Comm framework [21] – a new architecture for mobile heterogeneous networking.

A specially LAN-centric transport protocol has different requirements from a WAN transport protocol, e.g. TCP, since performance issues differ. These requirements mark the design decision that define SP [10]. Since most LAN communications consist of messages or transactions, SP supports a message-based communication in contrast to TCP streams. The higher speed available in LAN is exploited by using a larger window size for SP than WAN protocols: SP supports 4MB message sizes by default and can even be increased. In order to keep packet processing simple, SP uses a small number of connection states as well as packet types. Flexibility is achieved by allowing Quality of Service (QoS) to be set using the packet types.

In this paper (Section IV), we summarize briefly how the Avispa support helped in designing a secure extension of SP by hybrid cryptography. Extending the initial connection part of the protocol, we add public-key based authentication while simultaneously exchanging symmetric session keys for the following secured data exchange of SP. The protocol achieves authenticity by public keys while preserving its efficiency to an extent through the use of faster symmetric key encryption.

C. Avispa

Avispa stands for Automated Validation of Internet Security-sensitive Protocols and Applications [2]. To model and analyze a protocol, Avispa provides its own High-Level Protocol Specification Language (HLSPL). In order to check

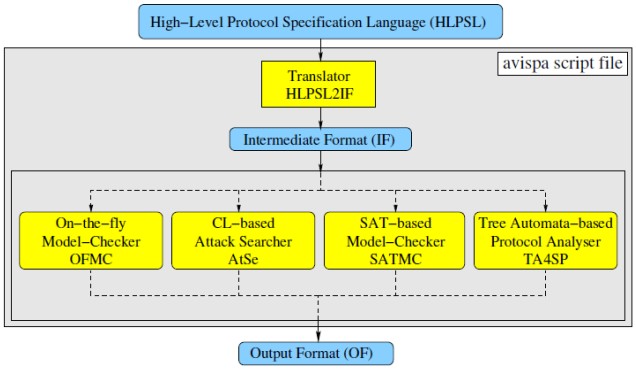


Figure 2. Avispa: language formats and tool architecture [2]

security, Avispa translates the given HLSPL specification in the intermediate format IF, which is then the basis for four different verification machines that can be applied to model check security properties on a protocol expressed as depicted in Figure 2. Avispa uses Dolev-Yao channels annotating them as a type as `channel(dy)`. This means that the attacker is assumed to be able to do eavesdropping, intercepting and faking on these channels. Protocols can be very naturally specified in Avispa using the *role* concept. Every principal is modeled as such a role which enables encapsulating its communication parameters, local variables and constants. Based on that, a role describes state changes by defining transitions between states that may depend on pre- and postconditions of the current state. Roles can furthermore be instantiated in other roles. This enables the composition of the single roles representing the single principals into a protocol session while synchronizing them on their communication. It also enables specifying an attacker. Once the protocol is thus specified predefined HLSPL propositions, most prominently secrecy and authentication can be automatically verified. We introduce more detail on HLSPL constructs, their IF translation, and the verification features when applying them to formalize Radius-SHA256 in the following section.

D. Hash Functions

Hash functions – also known as message digests or compression functions – map arbitrary length inputs to fixed size outputs. They are considered as cryptographic hash functions if they provide the following three properties: (a) they cannot be inverted, i.e. given $y = H(x)$ the input x cannot be found, (b) it is impossible to find collisions, i.e. we cannot find x, y with $H(x) = H(y)$, and (c) given an input hash pair it is impossible to find another input with the same hash value, i.e. for $H(x) = y$ we cannot find x' such that $H(x') = y$. The latter two properties resemble each other expressing the idea of *collision resistance* but the second one is stronger.

These basic properties of good hashes give rise to use them for cryptography. However, since a hash is a deterministic function it has as such not the same quality as an encryption algorithm: anyone can apply the hash. However, a hash can be easily combined with a shared secret to provide authentication which is often used for so-called message authentication codes (MAC). For example, let Kcs be a shared secret. Then, $H(Kcs)$ can be used as an authentication token since only principals who have access to Kcs can produce this token.

III. RADIUS-SHA256

In this section, we present the protocol Radius-SHA256 as derived from the classical Radius of RFC2865/66 by replacing MD5 by SHA-256. At the abstract protocol level this replacement seems simple but in order to ensure that this change of the original protocol preserves the security properties, we start from the formal presentation of the original Radius protocol and develop the new Radius-SHA256 on that formal basis. This enforces a detailed investigation of the necessary adjustment to the old – no longer secure – version of Radius and in addition enables comparison to the previously established security guarantees showing whether they still hold. From an engineering perspective, this procedure corresponds to a kind of refactoring of a protocol specification: re-engineering the previous security specification enables re-invocation of the previous verification by rerunning security check routines.

We introduce the protocol Radius-SHA256 by its formal model in HLSPL, the specification language of Avispa. Its level of abstraction is sufficient to comprehend just the major gist of the protocol. This model contains four roles: client, server, session, and environment. The idea is that the client role represents the NAS and the server role represents the Radius server. In applications, client and server might as well be represented by proxies depending on the type of network. For the formal presentation of the protocol we simplify by summarizing the scenario as a client-server session. As a session we consider the time period of a client-server communication. The attacker is modeled by the role of the environment that specifies the basis for the attacks on protocol executions.

Each of these components client, server, session and environment is modeled by a so-called "role" in HLSPL. Client (Section III-A) and server (Section III-B) define the two matching sides of the protocol; their composition as defined in the role session only gives the full protocol (see Section III-C and Figure 5) which can again be instantiated to model legal session and attacker.

A. Client-side Protocol

The client role is specified in Figure 3. This role definition defines the protocol by specifying the necessary entities, like identifiers, messages and used cryptographic primitives, e.g. the symmetric key `Kcs` in its header. Note, here how we

```
role client(C,S: agent,
   Kcs: symmetric_key,
   SHA256: hash_func,
   Success, Failure: text,
   Access_accept,Access_reject: text,
   SND, RCV: channel(dy))
played_by C def=
local  State: nat,
   NAS_ID, NAS_Port: text,
   Chall_Message: text
const  kcs: protocol_id,
   sec_c_Kcs : protocol_id
init State := 0
transition
   t1. State  = 0 ∧ RCV(start) ⇒
       State':= 1 ∧ NAS_ID':=new()
       ∧ NAS_Port':=new()
       ∧ SND(NAS_ID'.NAS_Port'.SHA256(Kcs))
       ∧ secret(Kcs,sec_c_Kcs,C,S)
   t2. State  = 1 ∧ RCV(NAS_ID.Access_accept) ⇒
       State':= 2 ∧ SND(NAS_ID.Success)
   t3. State  = 1 ∧ RCV(NAS_ID.Access_reject) ⇒
       State':= 3 ∧ SND(NAS_ID.Failure)
   t4. State  = 1 ∧ RCV(NAS_ID.Chall_Message') ⇒
       State':= 4 ∧ SND(NAS_ID.Chall_Message'_Kcs)
                  ∧ witness(C,S,kcs,Kcs)
   t5. State  = 4 ∧ RCV(NAS_ID.Access_accept) ⇒
       State':= 5 ∧ SND(NAS_ID.Success)
end role
```

Figure 3. Client role of Radius-SHA256 in HLSPL

define SHA256 to be a *hash function* in this header by using the Avispa keyword hash_func. This function is applied in the first transition of the following client-side of the protocol specification. In detail the steps of the protocol are defined as *state transitions* that are conditional on logical conditions of a current state State $\in \{1, \ldots, 5\}$: each of the five rules in the transition section in Figure 3 defines a *precondition* for this current state (to the left of the implication arrow $\Rightarrow$) and a *postcondition* on the post state State' of a transition after the $\Rightarrow$. The conditions are conjoined by logical conjunction with $\wedge$. The initial state is State zero. For example, the first transition t1 in Figure 3 can be read as follows. If the precondition holds, i.e. the current state is "state 0" and the role receives on its input channel RCV the message start, then the transition t1 is enabled. If this transitions fires, the post-state is "state 1" and the message NAS_ID.Success is sent on the output channel SND. The following transitions can be read in the same manner. Since the client represents only one principal in this protocol, we need to need to define the server side of the protocol to complement it.

B. Server-side Protocol

Figure 4 now shows the definition of the second principal in the model of Radius-SHA256: the Radius-server. The transitions defined in the role server correspond to the transitions of the client. Each SND on one side corresponds to a RCV on the other side. However, in order to put

```
role server(C,S: agent,
            Kcs: symmetric_key,
            SHA256: hash_func,
            Success, Failure: text,
            Access_accept,Access_reject: text,
            SND, RCV: channel(dy))
played_by S def=
local State: nat,
      NAS_ID, NAS_Port : text,
      Chall_Message     : text
const kcs: protocol_id,
      sec_s_Kcs : protocol_id
init State := 11
transition
t1. State  = 11
    ∧ RCV(NAS_ID'.NAS_Port'.SHA256(Kcs)) ⇒
    State':= 12 ∧ SND(NAS_ID'.Access_accept)
                ∧ secret(Kcs,sec_s_Kcs,C,S)
t2. State  = 12 ∧ RCV(NAS_ID.Success) ⇒
    State':= 13
t3. State  = 11
    ∧ RCV(NAS_ID'.NAS_Port'.SHA256(Kcs)) ⇒
    State':= 14 ∧ SND(NAS_ID'.Access_reject)
t4. State = 14 ∧ RCV(NAS_ID.Failure) ⇒
    State':= 15
t5. State  = 11
    ∧ RCV(NAS_ID'.NAS_Port'.SHA256(Kcs)) ⇒
    State':= 16 ∧ Chall_Message':=new()
                ∧ SND(NAS_ID'.Chall_Message')
t6. State  = 16 ∧ RCV(NAS_ID.Chall_Message_Kcs) ⇒
    State':= 17 ∧ SND(NAS_ID.Access_accept)
                     ∧ request(S,C,kcs,Kcs)
t7. State  = 17 ∧ RCV(NAS_ID.Success) ⇒
    State':= 18
end role
```

Figure 4. Server role of Radius-SHA256 in HLSPL

these building blocks together, we first have to define the composition. This is done in a further role for the session, presented in the following section.

C. Session and Attacker

The two roles of client and server are combined by defining a role for the session. Session uses the composition keyword to couple the two instances of client and server synchronized by common parameters.

```
role session(C,S: agent,
             Kcs: symmetric_key,
             SHA256: hash_func,
             Success, Failure: text,
             Access_accept,Access_reject: text) def=
local
  S1, S2 : channel (dy),
  R1, R2 : channel (dy)
composition
  client(C,S,Kcs,SHA256,Success,Failure,
         Access_accept,Access_reject,S1,R1) ∧
  server(C,S,Kcs,SHA256,Success,Failure,
         Access_accept,Access_reject,S2,R2)
end role
```

The synchronization couples the transitions of the client with the server over their connecting channels. For example, the

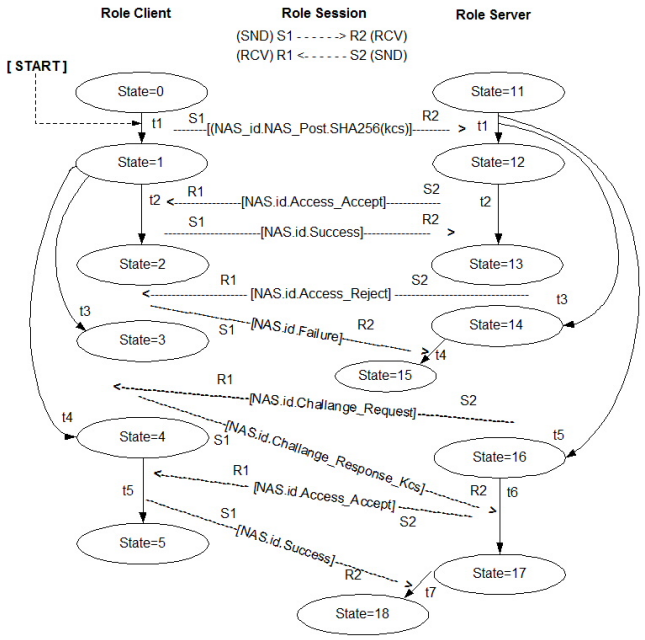


Figure 5. Composition of client and server yields protocol.

message `SND(NAS_ID.Success)` of `t2` in client is now being sent over `S1` and coupled via `R2` to the message `RCV(NAS_ID.Success)` of server. The composition that is defined in the role session actually defines the protocol between the roles client (Section III-A) and server (Section III-B) by instantiating their channels such that they mutually connect; the overall protocol is best illustrated graphically (see Figure 5).

The environment represents the attacker and uses a composition, now in turn of two session instances, where the first is one between two agents `c1` and `s1` and the second generalizes the first agent to be `i` – an unspecified agent that triggers the search for intruder possibilities incorporating agents. Note, also that the `SHA256` is given openly to the environment signifying that the attacker knows it and can use it which formalizes the idea of a hash that it is applicable by everyone (see Section II-D).

```
role environment() def=
const c1,s1: agent,
      sha256: hash_func,
      succs, fails: text,
      acc_acp, acc_rej: text,
      kcsk, kisk, kcik: symmetric_key,
      kcs: protocol_id
intruder_knowledge = {c1,s1,sha256,kisk,kcik,
                      succs, fails, acc_acp, acc_rej}
composition
session(c1,s1,kcsk,sha256,succs,fails,acc_acp,acc_rej)
∧
session(i, s1,kisk,sha256,succs,fails,acc_acp,acc_rej)
end role
```

The representation is abstract enough to be comprehensible while being in places a bit superficial. We dig deeper down into the lower levels of the Avispa model in the next section to investigate the influence of the hash function on the Radius-SHA256.

D. Security Verification

This section now illustrates how the actual model checking process of the Avispa tool automatically translates the high level protocol model in HLSPL defined in the previous section and performs a complete state analysis over the resulting internal Kripke structure representing this model. The verification is relative to a set of security properties specifying the goals of the authentication that we will illustrate first.

E. Security Properties and Verification Process

Given the implementation of the protocol as described in the previous section, we can now use the inbuilt features of Avispa to verify security in a push-button manner. Avispa provides two features for protocol verification: secrecy of keys and authentication. The secrecy of the server and client keys and authentication of client and server are given as verification commands to Avispa as follows.

```
goal
  secrecy_of sec_c_Kcs, sec_s_Kcs
  authentication_on kcs
end goal
```

The meaning of these two formulas can be illustrated more closely by inspecting their translation into the IF format. We apply all four back-ends OFMC, CLAtSE, SATMC, and TA4SP of the Avispa tool to the Radius-256 specification. For the full IF representation and the performance details of the analysis see [11]. The main observation is that the original security guarantees shown for Radius can be carried over to the protocol Radius-SHA256 by *simply replacing* the hash function MD5 by SHA-256 in the specification. The above secrecy and authentication properties verify just the same.

To understand the effect that the choice of a particular hash function, i.e. MD5, SHA-256, or any other cryptographic hash function has on the security guarantees, we need to inspect the IF version in more detail. First of all, a hash function application in HLSPL like `SHA256(Kcs)` is translated into IF as `apply(SHA256,Kcs)`. According to the Avispa semantics [2], this `apply` operator is reserved for the application of hash functions which manifests itself in the following type.

```
apply(F,Arg) apply: message × message → message
```

However, there seems to be no further semantics attached to the type. The defining properties of a cryptographic hash function are provided implicitly by defining the intruder knowledge for hashes as follows.

```
step gen_apply (PreludeM1,PreludeM2) :=
     iknows(PreludeM1).iknows(PreludeM2) ⇒
     iknows(apply(PreludeM1,PreludeM2))
```

Since the apply-operator can produce a hash, the intruder can apply a hash himself but Avispa's intruder semantics provides no rule to inverse a hash function nor any rule enabling collision detection for the intruder.

F. Evaluation and Generalization

Wrapping up the discussed security verification we see that the verification of Radius-SHA256 yields exactly the same guarantees as the classical Radius of RFC 2865/6. In this final section, we show up the consequences of this mechanized verification.

Primarily, the re-engineered modelling and verification for the Radius-256 protocol in Avispa shows that the guarantees of secrecy of keys and mutual authentication that have already been shown for the classical Radius version MD5 equally hold for Radius-256.

Next, our construction process reveals that the exchange of MD5 by another hash function in the Avispa model is simply replacing one (presumably) secure cryptographic hash function by another. As we have observed in the previous section, the Avispa semantics of a hash models hash functions abstractly. Thus, we observe that the verification depends only on the general assumption that *some* hash function is used in the protocol. Therefore the derived result can be generalized to *all* secure hash functions.

Theorem 1: The Avispa guarantees of secrecy of keys and authentication of the Radius protocol hold for all secure cryptographic hash functions.

Note, however, that this verification does presuppose a secure hash function. That is, the proved result is not valid if the assumed cryptographic strength of the hash function is flawed, like in the case of MD5.

Since the Avispa model cannot cover the implicit part of the hash function security proof, the analysis does not reveal possible attacks. However, the aforementioned attack on SSL [16] could be used as a guideline to produce a similar attack on the classical Radius protocol based on MD5. On the other hand, the generalization presented in this paper is not trivial: its proof relies on the re-engineering of the Radius for SHA-256 and the observation that this re-engineering is applicable to any secure hash function.

IV. Secure Simple Protocol

A. Transport Protocols and Future Network Environments

We are witnessing an explosive change in terms of the different types of wireless networks being developed and deployed. Thus, future network environments will primarily consist of multiple local wireless networks, each with a different Quality-of-Service (QoS). Users will be always connected by switching between available networks using vertical handover techniques [8].

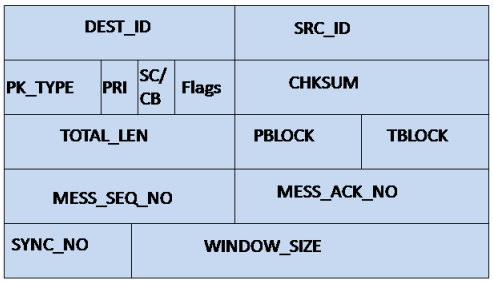


Figure 6. The structure of Ycomm/SP.

In this new world, a connection between two devices may therefore involve several network interfaces. This reality is beyond the scope of many TCP implementations and protocols such as SCTP [12] have been developed to replace TCP.

Another approach being explored is to keep TCP as a Wide Area transport protocol, but use another protocol to handle communication on local networks. Such an approach is being adopted by the Y-Comm Group [9], [21]. There are other factors favoring this approach: firstly, the common tack of tuning TCP to deal with different types of local networks has not been as effective as initially hoped. TCP is, in fact, a relatively complicated protocol as it needs to support different transport features including connection management, reliability, streaming and congestion evaluation and response. This means that it has been difficult for TCP to adjust to handover issues [3] without substantial support from device interfaces; we believe a local transport protocol should be simpler to use. Secondly, in order make use of higher speeds that are generally available in local environments, transport window sizes must be substantially larger by default. Thirdly there is a need to support different Qualities-of-Service is an explicit and flexible way. Finally, the issue of local area network security in terms ensuring that resources are properly balanced among several users must be urgently addressed to ensure improved user experience.

B. The Simple Protocol

The Simple Protocol (SP) [10] is being developed by the Y-Comm Group for Local area communications. The SP diagram is shown in Figure 6 with a brief explanation of the various fields shown in Figure 7.

As can be seen from Figures 6 and 7, SP is a message-based protocol where messages are broken down into a number of blocks. Unlike TCP, SP supports the concept of packet types and uses explicit phases: connection setup, data exchange and close; this allows the protocol to be quickly processed. It also supports a 4 MB window size, leading to improved transfer rates for large data exchanges. In SP,

- DEST_ID (16) – identifying remote end
- SRC_ID (16) – from source end
- PK_TYPE (4) - packet type
- PRI (2) - supports 4 priority levels
- SC (2) – destination scope
- CB (2) – supports ECN
- CHKSUM (16) – sixteen bit checksum
- TOTAL_LEN (16) – total packet length
- PBLOCK (8) – the present block
- TBLOCK (8) – the total number of blocks
- MESS_SEQ_NO (16) – last message sent
- MESS_ACK_NO (16) – last message received
- SYNC_NO (8) – the last ACK received
- WINDOW_SIZE (24) – the window size

Figure 7. Explaining what the fields mean.

acknowledgments can also be monitored leading to faster recovery times.

C. Support for Servers

Because SP is a message based protocol, it allows the application and the protocol stack to operate in a more asynchronous manner using an event-driven interface for receiving packets. Hence the interaction between the application and SP for reception revolves around the application being made aware of four key events:

1) A new connection has been established
2) A new message has been received on a connection
3) The connection has been reset by the network or other side
4) The other side has closed its connection, i.e. it is finished sending all its data.

Since it is possible to attach call back functions to each event, it is possible to build local servers that are totally event-driven. This leads to much more efficient server implementations. In addition, the application can tune SP so as to optimize the local environment. Thus a local transaction server can tune SP to suppress transport level acknowledgment requests since the reply by the server of the client's request will also be interpreted by the client transport system that the original request was correctly received. This too is a significant optimization for efficient server implementations. SP is therefore a powerful local protocol and hence the need to look at a secure version of SP.

D. Security Mechanism in SP

Because SP uses a connection phase, we can use this to improve the security of the system and to enhance communications. When connecting to servers, the concept of scope is supported. So a server can only be accessed using a scope which is defined by its functionality [1]. Four scopes are supported using 2 bits: Scope 00 means that the server can only be connected to processes on the same machine. Scope 01 means that the server can only be accessed by machines on the same LAN, while scope 10 indicates that access to the server is only accessible by machines on the same site while a scope of 11 shows that the server may be globally accessed. The connection phase of SP can be used to set up keys which are formed using local area parameters to ensure secure communication between machines in the local area.

The protocol SP consists of two parts: the connection part and the data transmission. The connection part establishes a communication between processes A and B to prepare a data transmission according to these established connection parameters. Thereby, a "connected" state is reached during which data may be transmitted before the connection is closed again. During data transmission, SP uses synchronization numbers (`SYNC_NO`) for each message and acknowledgments replying those message numbers to ensure safe transmission. This sequential message numbering can be used as well to secure the protocol against replay attacks, i.e. resending of previously intercepted messages by adversaries. However, to ensure this security, we need to keep the message numbers secret. To do that, we establish a session key in the connection part of SP. We assume that a global public key infrastructure provides certified identities, that is for every principal X on the network we have a signed pair $(K_X, X)_{K_C^{-1}}$ of a public key K_X associated to the principal's identity (for example the MAC of his device). This key-identity pair is signed with the secret key of the certification authority K_C^{-1} and can be verified by both parties A and B even off-line.

Now given this setup, the secure-SP connection part extends the basic exchange of request and reply (`REQ`, `REP`) by additional time stamps T, nonces N (where indices $\in \{A, B\}$ indicate the sender and receiver), sender, and a symmetric session key K_S for the future data transmission. The contents of the following two messages are encrypted using the public keys K_A and K_B so that only the intended recipient A or B can read the message contents.

$$\begin{aligned} A \mapsto B &: \texttt{REQ} + \{\texttt{SYNC\_NO}_A, T_A, A, N_A\}_{K_B} \\ B \mapsto A &: \texttt{REP} + \{\texttt{SYNC\_NO}_B, T_B, B, N_B, N_A, K_S\}_{K_A} \end{aligned}$$

If this two step challenge response protocol succeeds, a connection between A and B is established. In the course of that connection, A and B can now exchange messages whose `SYNC_NO` and shared secrets N_B and T_A are cryptographically protected by the symmetric key K_S that has been exchanged.

$$A \mapsto B : \{\texttt{SYNC\_NO}_A, N_B, T_A, A\}_{K_S} + \text{data message}$$

Note, that the authentication of A to B is only complete after the third step, i.e. the first data transmission, where A shows possession of the private key K_A^{-1} by decrypting and re-encrypting N_B, T_A, and K_S. This protocol has been formalized and successfully verified with Avispa (for details see the following section). Confidentiality and integrity of the data communication part holds as long as the session keys are not broken. This additional assumption is necessary and explicit in Avispa: it is beyond the scope of the protocol verification since we abstract from key length and duration of use. The same applies for the above mentioned public key infrastructure.

The secured SP protocol's communication part bears a strong resemblance to the (corrected) Needham-Schroeder asymmetric authentication protocol. This is no surprise, as the NS-asymmetric protocol (NSPK) is the essence of remote authentication.

E. Formalizing and Verifying SP in Avispa

The SP protocol resembles a lot the improved version (not susceptible to man-in-the-middle attacks) of the NSPK protocol.

1. $A \rightarrow B : \{N_A, A\}_{K_B}$
2. $B \rightarrow A : \{N_A, N_B, B\}_{K_A}$
3. $A \rightarrow B : \{N_B\}_{K_B}$

Due to this resemblance and the fact that NSPK is available in the Avispa library, we can *reuse* in large parts the Avispa formalisation of that NSPK protocol as a starting point adapting it to SP.

The formalization has two roles alice and bob. We first consider alice.

```
role alice (A,B : agent,
    Ka,Kb : public_key,
            Ks: symmetric_key,
    Snd,Rcv : channel (dy)) played_by A def=
local
State : nat,
Na,Nb : text,
Ta,Tb : text,
REQ,REP : message,
SN,MN,AMN : text
init State := 0
transition
1. State = 0  ∧ Rcv(start) ⇒
   State':= 1 ∧ Na' := new()
              ∧ Snd(REQ.SN.Ta.A.Na'_Kb)
      ∧ witness(A,B,na,Na')
      ∧ secret(Na',na,A,B)
2. State = 1  ∧ Rcv(REP.SN.Tb.B.Nb'.Na'.Ks_Ka) ⇒
   State':= 2 ∧ Snd(SN.Nb.Ta.A_Ks)
              ∧ wrequest (A,B,nb,Nb')
end role
```

Alice's role specifies the protocol from here point of view, i.e., it comprises the initial sending step as a send, followed by waiting to receive the step two, and finally sending out the third step that already used a symmetric key for efficient transport level encryption. Complementing this the role for bob is as follows.

```
role bob (B,A : agent,
  Kb,Ka : public_key,  Ks: symmetric_key,
  Snd,Rcv : channel (dy)) played_by B def=
local
State : nat,
Na,Nb : text,
Ta,Tb : text,
REQ,REP : message,
SN,AMN   : text
init State := 0
transition
1. State = 0  ∧ Rcv(REQ.SN.Ta.A.Na'_Kb) ⇒
   State':= 1 ∧ Nb' := new()
              ∧ Snd(REP.SN.Tb.B.Nb'.Na'_Ka)
      ∧ witness(B,A,nb,Nb')
      ∧ secret(Nb',nb,A,B)
2. State = 1 ∧ Rcv(SN.Nb.Ta.A_Ks) ⇒
   State':= 2 ∧ wrequest(B,A,nb,Nb)
end role
```

Similar to before for the Radius protocol, the two roles are now instantiated into a session.

```
role session (A,B: agent,
      Ka, Kb : public_key, Ks : symmetric_key)
def=
local SA, RA, SB, RB: channel (dy)
composition
   alice(A,B,Ka,Kb,Ks,SA,RA)
∧ bob(B,A,Kb,Ka,Ks,SB,RB)
end role
```

Also an environment is defined that sets up the intruder.

```
role environment() def=
const ta,tb,sn,mn,amn : text,
a, b, i      : agent,
na, nb : protocol_id,
ka, kb, ki   : public_key,
        ks,ksi: symmetric_key
intruder_knowledge =
      {a,b,i,ka,kb,ki,inv(ki),ta,tb,mn,amn,sn}
composition
session(a,b,ka,kb,ks) ∧ session(a,i,ka,ki,ksi)
end role
```

The goal we check is as follows (equal to the guarantees in NSPK).

```
  secrecy_of na, nb
  authentication_on alice_bob_nb
  authentication_on bob_alice_na
```

The checking of this goal succeeds and provides a fully automated verification of the authentication process of our specified SP protocol. The engineering of this secure SP protocol has been a process of reuse (reusing the NSPK-Avispa specification in large parts) and also composition: the SP protocol – as such rather a transport protocol – has been composed into secure SP by prepending the above authentication steps. This is a composition process.

V. Conclusions

In this paper we have shown that an adapted version of the Radius protocol using SHA-256 instead of MD5 provides exactly the same security guarantees as the RFC version based on MD5. The verification is a fully automatic analysis in the Avispa toolkit, a specialized model checker for security protocols. We could generalize this result to guarantee security for Radius protocols using secure hash functions, even other than SHA-256. We furthermore illustrated on the example of the simple protocol SP that modelchecking can be used to stepwisely introduce security to a transport layer protocol. The verification process has shown the feasibility of model checking as an engineering tool.

Although the authors of [15] provide a model for the Radius protocol as defined in the RFC, they have failed to sufficiently generalize their results. In some sense, our approach resembles a *refactoring* of the formal model: refactoring is a technique from software engineering supporting the change in software without affecting desired properties; we change the formal model of Radius by replacing MD5 by SHA-256 *without losing desired security properties*. In the process of following the earlier design, we discovered that the model is by no means limited to the classical Radius but can indeed be generalized to a more secure Radius-SHA256, and that this generalization can be extended to arbitrary hashes.

The generalization or refactoring could be an interesting concept to explore because for the working security engineer it provides an easy to use extension making the rather complex model checking process easy to access and provide a practical tool to allow more flexibility in network security engineering. Apart from facilitating the process of protocol engineering, this could also advocate the use of formal specification and automated model checking in the domain of network security.

References

[1] Mahdi Aiash, Glenford Mapp, Aboubaker Lasebae and Raphael Phan. Exploring the Concept of Scope to Provide Better Security for Internet Services. *Proceedings at the First Global Conference on Communication, Science and Information Engineering.* Middlesex University, London, 2011.

[2] Avispa v1.1 User Manual. Available at http://www.avispa-project.org, 2006.

[3] D. Cottingham and P. Vidales. Is Latency the Real Enemy in Next Generation Networks? *Proceedings of First International Workshop on Convergence of Hetergeneous Wireless Networks*. July 2005.

[4] F. Kammüller, G. Mapp, S. Patel. A. S. Sani. Engineering Security Protocols with Modelchecking – Radius-SHA256 and Secured Simple Protocol. *International Conference on Internet Monitoring and Protection, ICIMP'12*, 2012.

[5] I.-G. Kim and J.-K. Choi. Formal Verification of PAP and EAP-MD5 Protocols in Wireless Networks: FDR Model Checking. *AINA*. 2004.

[6] Y. Kirsal-Ever. Development of Security Strategies using Kerberos in Wireless Networks. PhD Thesis, Middlesex University, 2011.

[7] G. Lowe. Breaking and Fixing the Needham-Schroeder Security Protocol. *Information Processing Letters*, 1995.

[8] G. Mapp, F. Shaikh, M. Aiash, R. Vanni, M. Augusto and E. Moreira. Exploring efficient imperative handover mechanisms for heterogeneous networks. *Proceedings of the International Symposium of Emerging Ubiquitous and Persuasive Systems*. Indianapolis, Ind, USA, August 2009.

[9] G. Mapp, F. Shaikh, J. Crowcroft, D. Cottingham, and J. Baliosian. Y-Comm: A Global Architecture for Heterogeneous Networking (Invited Paper). *3rd Annual International Wireless Internet Conference (WICON)*, 2007.

[10] yRFC2: The Simple Protocol (SP) Specification, 15.9.2012. http://www.mdx.ac.uk/research/areas/software/ycomm_research.aspx

[11] S. Patel. Implementation and Analysis of Radius Protocol using Avispa. Master's Thesis. Middlesex University, 2011.

[12] RFC 4960 - Stream Control Transmission Protocol. *IETF.* September 2007.

[13] R. Rivest. Message-Digest MD5. *Network Working Group, RFC: 1321*. http://www.kleinschmidt.com/edi/md5.htm. 1992.

[14] A. S. Sani. Verifying the Secured Simple Protocol in AVISPA. Master's Thesis, Middlesex University, 2012.

[15] V. Sankhla. Formalisation of Radius in Avispa. http://www.avispa-project.org/library/RADIUS-RFC2865, University of Southern California, 2004.

[16] Rogue CA certificate signed by a commercial Certification Authority. http://www.win.tue.nl/hashclash/rogue-ca/\#sec71 Presented at the 25th Chaos Communication Congress, Berlin 2008.

[17] X. Wang and H. Yu. How to Break MD5 and Other Hash Functions. *Advances in Cryptology, Eurocrypt 2005.* LNCS **3439**, Springer 2005.

[18] Wikipedia RADIUS, http://en.wikipedia.org/wiki/RADIUS, 2012.

[19] Remote Authentication Dial In User Service (RADIUS). RFC 2868. Internet Engineering Task Force (IETF). http://tools.ietf.org/html/rfc2865, accessed 18th December 2012.

[20] Common Remote Authentication Dial In User Service (RADIUS) Implementation Issues and Suggested Fixes. RFC 5080. Internet Engineering Task Force (IETF). http://tools.ietf.org/html/rfc5080, accessed 18th December 2012.

[21] The Ycomm Framework. Offical Web-Site, Middlesex University. http://www.mdx.ac.uk/research/areas/software/ycomm_research.aspx.

Mitigating Distributed Service Flooding Attacks with Guided Tour Puzzles

Mehmud Abliz*, Taieb Znati*†, and Adam J. Lee*
*Department of Computer Science
†Telecommunication Program
University of Pittsburgh
Pittsburgh, Pennsylvania 15260 USA
{mehmud, znati, adamlee}@cs.pitt.edu

***Abstract*—Various cryptographic puzzle schemes have been proposed as defenses against Denial of Service (DoS) attacks. However, these schemes have two common shortcomings that diminish their effectiveness as a DoS mitigation solution. First, the DoS-resilience that these schemes provide is minimized when there is a large disparity between the computational power of malicious and legitimate clients. Second, the legitimate clients also have to perform the same heavy puzzle computations that do not contribute to any useful work from the clients' perspective. In this article, we introduce *guided tour puzzles* (GTP), a novel puzzle scheme that addresses these shortcomings. GTP uses latency — as opposed to computational delay — as a way of forcing a sustainable request arrival rate on clients. Measurement results from a large-scale network test-bed shows that the variation in the puzzle solving times is significantly smaller than the puzzle solving time variation of computation-based puzzles. As attackers have much less control over the round-trip delays than they do over the computational power, a latency-based puzzle scheme, such as GTP, provides significantly better protection against strong attackers. Meanwhile, we show that GTP minimizes useless computations required for the client computers. We evaluate the effectiveness of guided tour puzzles in a realistic simulation environment using a large-scale Internet topology, and show that GTP provides a strong mitigation of DoS request flooding attacks and puzzle solving attacks.**

***Keywords*-denial of service; availability; tour puzzles; proof of work; client puzzles; cryptography.**

I. INTRODUCTION

A Denial of Service (DoS) attack is an attempt by malicious parties to prevent legitimate users from accessing a service, usually by depleting the resources of the server, which hosts that service. DoS attacks may target resources such as server bandwidth, CPU, memory, storage, or any combination thereof. These attacks are particularly easy to carry out if a significant amount of server resource is required to process a client request that can be generated trivially. Cryptographic puzzles have been proposed to defend against DoS attacks with the aim of balancing the computational load of the server relative to the computational load of the clients [1] [2] [3] [4] [5] [6].

In a cryptographic puzzle scheme, a client is required to solve a moderately hard computational problem, referred to as *puzzle*, and submit the solution as a proof of work before the server spends any significant amount of resource on its request. Solving a puzzle typically requires performing significant number of cryptographic operations, such as hashing, modular multiplication, etc. Consequently, the more a client requests service from the server, the more puzzles it has to solve, further expending its own computational resources. Puzzles are designed so that their construction and verification can be achieved with minimum server computational load in order to avoid DoS attacks on the puzzle scheme itself. Attacks aimed at the puzzle scheme itself are thereafter referred to as *puzzle solving attacks*.

Originally, cryptographic puzzles were proposed to combat spams [7]. They have then been extended to defend against other attacks, including DoS [2] [3] [6] [8] [9] and Sybil attacks [10] [11]. Furthermore, different ways of constructing and distributing puzzles have been explored [6] [12] [13] [14] [15]. Unfortunately, existing puzzle schemes have shortcomings that limit their effectiveness in defending against DoS attacks.

First, the effectiveness of computation-based puzzles decreases, as the variation in the computational power of clients increases. To illustrate this limitation, consider a system composed of a server whose capacity is R requests per second, N_l legitimate clients whose clock frequency is f, and N_m malicious clients whose clock frequency is $a \cdot f$, where a is a *disparity factor* that represents the degree of disparity between the CPU powers of malicious and legitimate clients. Furthermore, assume that legitimate clients can tolerate a maximum puzzle difficulty of D_{max}, expressed in terms of the number of instructions. The maximum protection the server can achieve against a DoS attack is by setting the puzzle difficulty to D_{max}. During an attack, the total load on the server is the sum of the loads generated by the legitimate and malicious clients, which can be expressed as $N_l \frac{f}{D_{max}} + N_m \frac{af}{D_{max}}$ (without loss of generality, we assume that when solving puzzles clients use their full CPU capacity). Therefore, to carry out a DoS attack against the server, an attacker must at least induce a load on the server that exceeds the server's full capacity, i.e., $N_l \frac{f}{D_{max}} + N_m \frac{af}{D_{max}} \geq R$. Using simple deductions, it is clear that the minimum number of malicious clients required to cause denial of service should satisfy the inequality $N_m \geq \frac{RD_{max} - N_l f}{af}$. Consequently, the minimum number of

malicious clients required to stage a successful DoS attack against the server becomes smaller as the disparity factor a increases, decreasing the effectiveness of a puzzle-based defense in mitigating the DoS attacks.

Second, existing puzzle schemes may exact heavy computational penalty on legitimate clients, when the server load becomes heavy and the need to increase the computational complexity of the puzzle becomes necessary to prevent overloading the server. The negative impact of such a penalty is further compounded by the fact that the puzzle-induced computation does not usually contribute to the execution of any task that is useful to the client, thereby further wasting client resources and limiting the client's ability to carry out other computational activities.

In this article, we propose a novel, latency-based puzzle scheme, referred to as *Guided Tour Puzzle (GTP)*, to address the shortcomings of current cryptographic puzzle schemes in dealing with DoS attacks. The guided tour puzzle scheme is the first to use the concept of latency, as opposed to computational delay, to control the rate of client requests and prevent potential DoS attacks on the server. The main contributions of the proposed work include:

- A comprehensive study of the cryptographic puzzles to derive a list of basic requirements;
- The introduction of two novel puzzle properties, namely *puzzle fairness* and *minimum interference*, that are essential to the effectiveness of puzzle-based defense against DoS attacks;
- Design and analysis of the GTP scheme, and showing that the proposed scheme meets the list of basic requirements while achieving puzzle fairness and minimum interference;
- A thorough evaluation of the guided tour puzzle effectiveness against distributed DoS attacks, using a realistic simulation framework.

The rest of the article is organized as follows. Section II provides a survey of cryptographic puzzle based DoS prevention schemes. Section III describes the system model and the threat model used. Section IV discusses the design goals of GTP scheme. Section V introduces the GTP scheme. In Section VI, we use analysis and measurement to show that guided tour puzzles meet the design goals of GTP scheme. The effectiveness of the GTP in mitigating DDoS attacks is evaluated in Section VII. A conclusion and future plans for extending the puzzle framework are presented in Section VIII.

II. Related Work

Currently, there are many different type of DoS and DDoS defense mechanisms such as filtering based [16] [17], traceback and pushback based [18] [19], capability based [20] [21] and cryptographic puzzle based defense mechanisms. One approach that is similar to GTP scheme is a system called *speak-up* that encourages legitimate hosts to increase their request sending rate during application layer DoS attacks [22]. This method uses a bandwidth based puzzle and is different from the latency based puzzle we proposed. WebSOS [23] is similar to GTP in that both builds a strong distributed network of protection points in front of a DoS vulnerable server. However, key differences exist. The overlay network in WebSOS is used as a tool to hide the IP addresses of the secret nodes that are permitted to send traffic to the protected server, whereas the set of tour guides act as a "delay box" to let the client wait between requests. The WebSOS is designed to protect private services whose users are known a priori, whereas the GTP scheme can be used by both private and public services.

Due to the enormity of various DoS defense solutions, here we limit our survey only to the cryptographic puzzle based mechanisms.

A. Client Puzzles

Dwork and Noar [7] were the first to introduce the concept of requiring a client to compute a moderately hard but not intractable function, in order to gain access to a shared resource. However this scheme is not suitable for defending against the common form of DoS attack due to its vulnerability to puzzle solution pre-computations.

Juels and Brainard [2] introduced a hash function based puzzle scheme, called *client puzzles*, to defend against connection depletion attack. Client puzzles addresses the problem of puzzle pre-computation. Aura et al. [4] extended the client puzzles to defend DoS attacks against authentication protocols, and Dean and Stubblefield [5] implemented a DoS resistant TLS protocol with the client puzzle extension. Wang and Reiter [6] further extended the client puzzles to prevention of TCP SYN flooding, by introducing the concept of *puzzle auction*. Price [24] explored a weakness of the client puzzles and its above mentioned extensions, and provided a fix for the problem by including contribution from the client during puzzle generation.

Waters et al. [9] proposed outsourcing of puzzle distribution to an external service called *bastion*, in order to secure puzzle distribution from DoS attacks. However, the central puzzle distribution can be the single point of failure, and the outsourcing scheme is also vulnerable to the attack introduced by Price [24].

Wang and Reiter [8] used a hash-based puzzle scheme to prevent bandwidth-exhaustion attacks at the network layer. Feng [25] argued that a puzzle scheme should be placed at the network layer in order to prevent attacks against a wide range of applications and protocols. And Feng and Kaiser et al. [3] implemented a hint-based hash reversal puzzle at the IP layer to prevent attackers from thwarting application or transport layer puzzle defense mechanisms.

Portcullis [26] by Parno et al. used a puzzle scheme similar to the puzzle auction by Wang [6] to prevent denial-of-capability attacks that prevent clients from setting up

capabilities to send prioritized packets in the network. In Portcullis, clients that are willing to solve harder puzzles that require more computation are given higher priority, thus potentially giving unfair advantage to powerful attackers.

In all of proposals above, finding the puzzle solution is parallelizable. Thus an attacker can obtain the puzzle solution faster by computing it in parallel using multiple machines. Morever, they all suffer from the resource disparity problem, and interferes with the concurrently running user applications. In comparison, guided tour puzzles are non-parallelizable, and addresses the problems of resource disparity and interference with user applications.

B. Non-Parallelizable Puzzles

Non-parallelizable puzzles prevents a DDoS attacker that uses parallel computing with large number of compromised clients to solve puzzles significantly faster than average clients. Rivest et al. [27] designed a *time-lock puzzle*, which achieved non-parallelizability due to the lack of known method of parallelizing repeated modular squaring to a large degree [27]. However, time-lock puzzles are not very suitable for DoS defense because of the high cost of puzzle generation and verification at the server.

Ma [14] proposed using *hash-chain-reversal puzzles* in the network layer to prevent against DDoS attacks. Hash-chain-reversal puzzles have the property of non-parallelizability, because inverting the digest i in the chain cannot be started until the inversion of the digest $i+1$ is completed. However, construction and verification of puzzle solution at the server is expensive. Furthermore, using a hash function with shorter digest length does not guarantee the intended computational effort at the client, whereas using a longer hash length makes the puzzle impossible to be solved within a reasonable time.

Another hash chain puzzle is proposed by Groza and Petrica [15]. Although this hash-chain puzzle provides non-parallelizability, it has several drawbacks. The puzzle construction and verification at the server is relatively expensive, and the transmission of a puzzle to client requires high-bandwidth consumption.

More recently, Tritilanunt et al. [28] proposed a puzzle construction based on the subset sum problem, and suggested using an improved version [29] of *LLL lattice reduction* algorithm by Lenstra et al. [30] to compute the solution. However, the subset sum puzzles has problems such as high memory requirements and the failure of LLL in dealing with large instance and high density problems.

Although the non-parallelizable puzzles addresses one of the weaknesses of client puzzles discussed in Section II-A, they still suffer from the resource disparity problem and interferes with the concurrently running user applications on client machines. Guided tour puzzles, on the other hand, address these two weaknesses of non-parallelizable puzzles.

C. Memory-Bound Puzzles

Abadi et al. [12] argued that memory access speed is more uniform than the CPU speed across different computer systems, and suggested using memory-bound function in puzzles to improve the uniformity of puzzle cost across different systems. Dwork et al. [13] further investigated Abadi's proposal and provided an abstract memory-bound function with an amortized lower bound on the number of memory accesses required for the puzzle solution. Although these techniques seem promising, there are several issues that need to be resolved regarding memory-bound puzzles.

First, memory-bound puzzles assume a upper-bound on the attacker machine's cache size, which might not hold as technology improves. Increasing this upper-bound based on the maximum cache size available makes the memory-bound puzzles too expensive to compute by average clients. Secondly, deployment of proposed memory-bound puzzle schemes require fine-tuning of various parameters based on a system's cache and memory configurations. Furthermore, puzzle construction in both schemes is expensive, and bandwidth consumption per puzzle transmission is high. Last, but not least, clients without enough memory resources, such as PDAs and cell phones, cannot utilize both puzzle schemes, hence require another service that performs the puzzle computation on their behalf.

III. System Model

In this section, we introduce our system model, including a system overview, a model of cryptographic puzzle protocol, and a threat model.

A. System Overview

We consider an Internet-scale distributed system of clients and servers. A *server* is a process that provides a specific service, for example a Web server or an FTP server. A *client* is a process that requests service from a server. The term *client* and *server* are also used to denote the machines that run the server process and the client process respectively. Clients are further classified as *legitimate clients* that do not contain any malicious logic and *malicious clients* that contain malicious logic. In the denial of service context, a malicious client attempts to prevent legitimate clients from receiving service by flooding the server with spurious requests. An *attacker* is a malicious entity who controls the malicious clients. We refer to a *user* as a person who uses a client machine.

B. Threat Model

The attacker attempts to disrupt service to the legitimate clients by sending apparently legitimate service requests to the server to consume its computational resources. We consider DoS attacks that flood the server with large amount of requests and attacks that attempt to thwart puzzle defense using massive computational resources. It is assumed that

network resources are large enough to handle all traffic, and the resource under attack is server computation.

Our threat model assumes a stronger attacker than previous schemes do. First, we assume that the attacker may possess hardware resources that are several orders of magnitude more powerful than that of the average legitimate clients. Meanwhile, the attacker can take maximum advantage of her resources by perfectly coordinating all of her available computation resources. Next, the attacker can eavesdrop on all messages sent between a server and any legitimate client. We assume that the attacker can modify only a limited number of client messages that are sent to the server. This assumption is reasonable since if an attacker can modify all client messages, then it can trivially launch a DoS attack by dropping all messages sent by all clients to the server. Finally, the attacker may launch attacks on the puzzle scheme itself, including puzzle construction, puzzle distribution, or puzzle verification.

IV. Design Goals

The design goal of GTP scheme is twofold. First, it aims to achieve the general puzzle properties that have been discussed in the existing literature. Second, it must meet the puzzle fairness and minimum interference requirements that are proposed by us to address the limitations of existing puzzle schemes. These requirements and puzzle properties are explained next.

A. General Properties

Computation guarantee. The computation guarantee (also referred to as "bounds on cheating" [31]) means a cryptographic puzzle guarantees a lower and upper bound on the number of cryptographic operations spent by a client to find the puzzle answer. In other words, a malicious client should not be able to solve a puzzle by expending significantly less operations than required. This is discussed in [7].

Efficiency. The construction, distribution, and verification of a puzzle by the server should be efficient in terms of CPU, memory, bandwidth, hard disk, etc. Specifically, puzzle construction, distribution, and verification should add minimal overhead to the server to prevent the puzzle scheme itself from becoming an avenue for denying service [7] [4] [3].

Adjustability of difficulty. This property is also referred to as *puzzle granularity* [28]. Adjustability of puzzle difficulty means the cost of solving the puzzle can be increased at a fine granularity from zero to impossible [4]. Adjustability of difficulty is important, because finer adjustability enables the server to achieve better trade-off between blocking attackers and the service degradation of legitimate clients.

Correlation-free. A puzzle is considered correlation-free if knowing the solutions to all previous puzzles seen by a client does not make solving a new puzzle any easier [4]. If a puzzle is not correlation-free, then it allows malicious clients to solve puzzles faster by correlating previous answers.

Stateless. A puzzle is said to be stateless if it requires constant memory at the server for storing client information or puzzle-related data. This property is discussed in [4].

Tamper-resistance. A puzzle scheme should limit replay attacks over time and space. Puzzle solutions should not be valid indefinitely and should not be usable by other clients [4] [3].

Non-parallelizability. Non-parallelizability means a puzzle solution cannot be computed in parallel using multiple machines [28]. Non-parallelizable puzzles can prevent attackers from distributing computation of a puzzle solution to a group of machines to obtain the solution quicker.

B. Puzzle Fairness and Minimum Interference

Puzzle Fairness. Puzzle fairness means that a puzzle should take approximately the same amount of time to compute by any client, regardless of the CPU power, memory size, and bandwidth available to that client. If a puzzle scheme achieves fairness, then a malicious client with very strong computational resources will effectively be reduced to a legitimate client. Without puzzle fairness, few powerful malicious clients can solve puzzles at a higher rate than many computationally weaker legitimate clients, and may lead to the occupation of most of the server's capacity by few malicious clients.

Minimum Interference. This property requires that puzzle computation at the client should not interfere with the normal usage of the client computer by its users. If a puzzle scheme consumes too many resources, then it interferes with users' normal computing activity. For example, if computing a hash reversal puzzle expends most of the CPU cycles, then a user may feel a very slow response in using other applications that are running concurrently on the client machine. Consequently, users may avoid using any service that deploys such a puzzle scheme.

V. Guided Tour Puzzle

This section presents the GTP scheme. We start out with the main idea behind the GTP scheme, and describe a very basic puzzle protocol. Then, the limitations of the basic protocol is discussed and a solution is given to address each limitation.

A. The Basic Protocol

When a server suspects that it is under attack or its load is above a certain threshold, it asks all clients to solve a puzzle prior to receiving service. In the GTP protocol, the puzzle is simply a tour that needs to be completed by the client via taking round-trips to a set of special nodes, called *tour guides*, in a sequential order. We call this tour a *guided tour*, because the client should not know the order of the tour a priori, and each tour guide must direct the client towards

Table I: A summary of notations.

N	Number of tour guides in the system
G_j	j-th tour guide ($1 \leq j \leq N$)
k_S	Secret key only known to the server
k_{Sj}	Shared key between the server and G_j
$k_{i,j}$	Shared key between G_i and G_j ($i \neq j$)
L	Length of a guided tour
A_x	Address of client x
i_s	Index of the s-th stop tour guide ($1 \leq i_s \leq N$)
t_s	Coarse timestamp at the s-th stop of the tour
R_s	Client puzzle solving request at s-th stop
B	Size of the $hash$ digest in bits

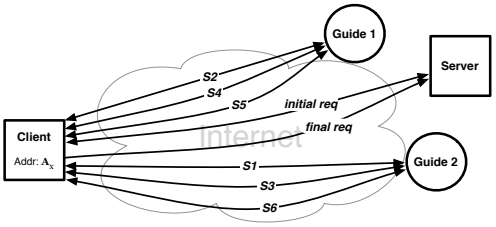


Figure 1: Example of a guided tour; the tour length is 6, and the order of visit is: $G_2 \rightarrow G_1 \rightarrow G_2 \rightarrow G_1 \rightarrow G_1 \rightarrow G_2$.

the next tour guide. Each tour guide may appear zero or more times in a tour, and the term *stop* is used to represent a single appearance of a tour guide in a given tour.

Figure 1 shows an example of a guided tour with two tour guides and 6 stops. The tour guide at the first stop of a tour is randomly selected by the server, and will also be the last stop tour guide, i.e., a guided tour is a closed-loop tour. The tour guide at each stop randomly selects the next stop tour guide. Starting from the first stop, the client contacts the tour guide at each stop and receives a reply. Each reply contains a token that proves to the next stop and the last stop that the client has visited this stop. Prior to sending its reply, the tour guide at each stop verifies that the client visited the previous stop tour guide, so that the client cannot contact multiple tour guides in parallel. After completing $L-1$ stops in a L-stop tour, the client submits the set of tokens it collected from all previous stops to the last stop tour guide (which is also the first stop tour guide), which will issue the client a proof of tour completion. The client then sends this proof to the server, along with its service request, and the server grants the client service if the proof is valid.

There are several issues concerning the basic protocol. First of all, a security mechanism must be in place to enforce the sequentiality of a single tour. Second, as a guided tour does not create a computational or bandwidth bottleneck at the client machine, an attacker may take many tours simultaneously, thereby qualifying itself for more resources of the server. Third, an attacker may cause DoS on the server indirectly by attacking the tour guides and the puzzle scheme itself. In the following subsections, we address each of these challenges individually.

B. Ensuring Sequential Guided Tour

We set up N tour guides in the system, where $N \geq 1$. The server keeps a secret k_S that only it knows, and a set of keys $k_{S1}, k_{S2}, \ldots, k_{SN}$ are shared between the server and each tour guide. Each tour guide G_i maintains a pairwise shared key $k_{i,j}$ with every other tour guide G_j, where $i \neq j$ and $1 \leq i, j \leq N$. The total number of keys need to be maintained by each tour guide or the server is N, and this key management overhead is acceptable since N is usually a small number in the order of 10 or less. The tour length L is decided by the server to adjust the puzzle difficulty. Notations are summarized in Table I.

The four steps of the GTP protocol is as follows.

1) Service request: A client x sends a service request to the server. If the server load is normal, the client's request is serviced as usual; if the server is overloaded, then it proceeds to the next step.

2) Initial puzzle generation: The server replies to the client x with a message that informs the client to complete a guided tour. The reply message contains $\{L, i_1, t_0, h_0, m_0\}$, where i_1 is the uniform-randomly selected index of the first stop tour guide, t_0 is a coarse timestamp, h_0, m_0 are message authentication codes that provide message integrity. h_0 and m_0 are computed as follows:

$$h_0 = hmac(k_S,\ (A_x||L||i_1||t_0)) \quad (1)$$

$$m_0 = hmac(k_{Si_1},\ (A_x||L||i_1||t_0||h_0)) \quad (2)$$

where, $||$ denotes concatenation, A_x is the address (or any unique value) of the client x, and $hmac$ is a cryptographic hash-based message authentication code (HMAC) [32]. Since m_0 is computed using the key k_{Si_1} that is shared between the first stop tour guide G_{i_1} and the server, it enables G_{i_1} to do integrity checking later on.

3) Puzzle solving: After receiving the puzzle information, the client visits the tour guide G_{i_s} at each stop s, where $1 \leq s \leq L$, and receives a reply. Each reply message contains $\{h_s, m_s, i_{s+1}, t_s\}$, where i_{s+1} is the uniform-randomly selected index of the next stop tour guide, t_s is the timestamp at stop s, and h_s, m_s are computed as follows:

$$h_s = hmac(k_{i_s,i_1},\ (h_0||A_x||L||s||i_s||i_{s+1})) \quad (3)$$

$$m_s = hmac(k_{i_s,i_{s+1}},\ (m_{s-1}||A_x||L||s||i_s||i_{s+1}, t_s)) \quad (4)$$

At each stop s, the client sends a puzzle solving request message R_s that contains $\{h_0, L, s, t_{s-1}, m_{s-1},\ i_1, i_s\}$ to the tour guide G_{i_s}, and the tour guide G_{i_s} replies to the client only if m_{s-1} is valid. In other words, each stop enforces that the client correctly completed the previous stop of the tour.

At the $(L-1)$-th stop, the tour guide $G_{i_{L-1}}$ knows that the next stop is the last stop, and replaces i_{s+1} with i_1 (recall that the first stop i_1 is also the last stop) when computing

h_s and m_s. After completing the $(L-1)$-th stop, the client computes h_L as follows

$$h_L = h_1 \oplus h_2 \oplus \ldots \oplus h_{L-1} \quad (5)$$

where $\oplus$ means exclusive or, and submits $\{h_0, h_L, L, m_{L-1}, i_1, i_2, \ldots, i_L\}$ to the first stop tour guide G_{i_1}. Using these information, G_{i_1} can compute $h_1, h_2, \ldots, h_{L-1}$ using formula (3), and subsequently h_L using formula (5). Note that only G_{i_1} can compute values h_1 to h_{L-1}, since only it knows the keys k_{i_1,i_2} to $k_{i_1,i_{L-1}}$ that are used in the HMAC computations.

If the h_L submitted by the client matches the h_L computed by G_{i_1} itself, then G_{i_1} sends back the client a token h_{sol} that can prove to the server that the client did complete a tour of length L. The token h_{sol} is computed as follows:

$$h_{sol} = hmac(k_{Si_1}, (h_0||A_x||L||t_L)) \quad (6)$$

4) Puzzle verification: The client submits to the server $\{h_0, h_{sol}, t_0, t_L, i_1\}$ along with its service request, and the server checks to see if h_0 and h_{sol} that it computes using formulas (1) and (6) matches the h_0 and h_{sol} submitted by the client. If both hash values match, the server allocates resources to process the client's request.

C. Thwarting Simultaneous Tours

The sequential completion of a single tour can be achieved via cryptographic hash chains, as shown above. However, a malicious client can still take multiple tours simultaneously. To prevent simultaneous tours, the GTP scheme limits the number of tours that can be carried out within each time interval. The details are described next.

The time in the GTP protocol is divided into time intervals of length Δ, and T_i is used to denote the i-th time interval. The token that the client receives during the time interval T_i for completing a tour can only be used during the time intervals T_i and T_{i+1}. This restriction can be achieved easily by using a clock tick value of Δ for the coarse timestamp t_s used in the GTP protocol. The interval length Δ must be set to a value that provides enough time for completion of at least a single tour.

Acquiring the token in T_i and using it in T_{i+1} eliminates the additional service delay incurred by using GTP scheme. But, there must be limits to how many tokens a client can acquire per interval and how much service a single token can buy. The policy concerning the per-token resource allocation at the server can be decided by the owner of the service, hence it is beyond the scope of this article. However, it is worth noting that reuse of a token can be prevented by caching only the verified tokens in a Bloom filter [33], and check for the existence in the Bloom filter whenever a new token arrives.

To limit how many tokens a client can acquire per interval, we propose the following solution. During each time interval T_i, a tour guide keeps a pair of counting Bloom filters (in a counting Bloom filter the array positions, or buckets, are extended from being a single bit, to an n-bit counter), one for each interval T_{i-1} and T_i. The tour guide counts the number of tours a client x is taking during each time intervals using the bloom filter, and ignores the client if it has already taken C_{T_i} tours for that period. The value of C is decided by the server for each time interval based on its load, and it is included in the hash chain computation to protect against manipulation by the client. The number of tours are accounted for two time intervals (T_{i-1} and T_i), as a single tour may span two time intervals and using a single counter does not capture that situation. The GTP scheme uses the server timestamp t_0 to decide, which interval a tour belongs to; if $t_0 = T_{i-1}$, the tour belongs to the interval T_{i-1}, and vice versa. This scheme for limiting tours requires the clocks of the tour guides and the server to be synchronized. However, the accuracy of the synchronization is coarse-grained enough (in the order of seconds) such that synchronizing the server and the tour guide clocks independently with a time server using NTP protocol [34] suffices.

D. Increasing Tour Guide Robustness

To prevent attackers from indirectly launching DoS on the server by attacking one of the tour guides, tour guides should be robust against attacks on themselves. As tour guides perform very simple operation, i.e., computing a hash function, they are very light-weight and far less susceptible to DoS attacks. Also due to their simple operation, securing the tour guides against compromise attempts also becomes much simpler. Furthermore, the basic guided tour puzzle scheme is designed to localize the impact of a compromised tour guide. Due to the all-pair pair-wise shared keys, compromising one tour guide only gives the attacker a free ride for the leg of the tour that starts with the compromised tour guide, and the attacker still has to complete the majority of the tour.

In terms of DoS attacks on the tour guides, we propose a simple solution to thwart DoS attacks. In this solution, whenever a tour guide receives a puzzle solving request from the client, it checks to see if the client's request is already in its service queue (the priority queue when the puzzle solving time adjustment mechanisms is used), and it simply drops the request if another request from the same client is already in the queue. This will prevent a malicious client from unfairly taking up more than one slot in the tour guide's service queue, and subsequently minimizes the effect of a DoS attack on legitimate clients.

Although the tour guides are highly immune to DoS attacks, it is still possible for a tour guide to be down due to internal failure or a very strong DoS attack that involves millions of nodes. To operate gracefully even when one of the tour guides is down, all tour guides exchange heartbeat messages with each other and with the server, such that unavailability of a tour guide is immediately known by the

server and other tour guides, and forwarding of clients to the failing or unavailable tour guide is avoided. The heartbeat messages should be adequately protected to thwart attacks on the tour guides.

VI. Analysis

In this section, we use analytical reasoning and experimental results to demonstrate that guided tour puzzles meet all of our proposed design goals.

A. General Puzzle Properties

For each property, we briefly explain how that property is achieved in guided tour puzzle.

Computation guarantee. Each client is required to complete L round-trips in order to obtain a token that proves to the server that it has completed a length L guided tour. A client cannot skip any one of the required round-trips, because doing will be detected by the tour guides immediately. Therefore, guided tour puzzles achieve a strict computation guarantee that enforces the same number of operations for computing the same puzzle answer at all clients.

Efficiency. In guided tour puzzle, construction of a puzzle takes only two hash operations (to compute h_0 and m_0) at the server, and verification of a puzzle answer also takes two hash operations (to compute h_0 and h_L). This efficiency can be further improved at the cost of a small fixed size memory for caching h_0 values. Transferring of puzzle from server to the client requires $2 * B/8$ plus few extra bytes, where the size of hash digest B is usually $160 \sim 256$ bits.

Adjustability of difficulty. The difficulty of a tour puzzle is adjusted by adjusting the tour length L, which can be increased or decreased by one as needed. So, guided tour puzzle provides linear adjustability of difficulty.

Correlation-free. Attackers may try to correlate previousy seen puzzles and puzzle solution to directly obtain a new valid puzzle solution or compute new puzzle solutions after obtaining the secret key used in the hash computations. In guided tour puzzle scheme, resistance to such correlation attacks are achieved through the pre-image resistance and collision resistance properties of the cryptographic hash function used. It is important to pick a hash function that is proven in practice to be secure against various cryptanalytic attacks.

Stateless. Guided tour puzzle does not require the server to store any client or puzzle related information, except for the cryptographic keys that are used for the hash calculation. Puzzle answer verification using memory lookup require few megabytes of memory, but the size of this memory is constant and does not increase as the number of clients increase. The Bloom filter memory used by tour guides is also roughly constant in size and small enough such that it is not susceptible to memory depletion attacks.

Tamper-resistance. The server can guarantee a limited validity period of a puzzle answer, by checking if the difference between the t_L and t_0 is within an acceptable value t_D, i.e. $t_L - t_0 \leq t_D$. Recall that t_0, t_L are submitted by the client to the server in the verification phase of the puzzle protocol, and the client cannot change these values since they are protected by the hash values h_0 and h_L. Meanwhile, the puzzle answer computed by one client cannot be used by any other client, since a value unique to each client is included in the computation of h_0 and h_s.

Non-parallelizability. A guided tour puzzle cannot be completed in parallel, because at each stop s, the tour guide G_{i_s} requires the client to submit the hash values m_{s-1} from the previous stop, and replies to the client if only m_{s-1} is valid. As such, the puzzle scheme strictly enforces sequential completion of a guided tour and achieves non-parallelizability.

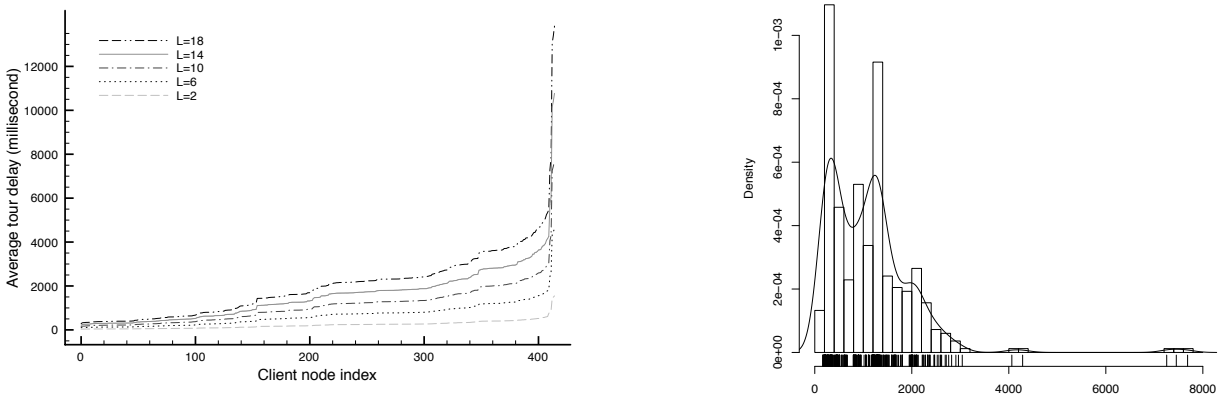
B. Achieving Puzzle Fairness

The guided tour puzzle scheme is not affected by the disparity in the computational powers. It is because the round trip delays that consist the puzzle solving time of a guided tour puzzle are mostly decided by the intermediate network between the client and the tour guides, and the clients' CPU, memory, or bandwidth resources have minimal impact. In terms of the uniformity of puzzle solving time across clients, the guided tour puzzle scheme provides a better guarantee compared to the computation-based puzzle schemes as shown in next subsections.

1) Experimental Analysis: In the following, we use *tour delay* to refer to the sum of all round trip delays occurred in a single tour. We use a two-week long measurement data collected on PlanetLab [35] to show that the variation in tour delay across clients is within a small factor for a large distributed system. PlanetLab has a collection of over $1,000$ nodes distributed across the globe, and provides a realistic network testbed that experiences congestion, failures, and diverse link behaviors [35]. We used about 40% (over 400) of the nodes that had complete measurement data available throughout the two-week period.

We first randomly chose 20 nodes, out of the 400 selected nodes, as candidates for tour guides. The remaining nodes are used as client nodes. The number of tour guides N is varied from 4 to 20, and the tour length L is varied from 2 to 18. For each (N, L) pair, guided tours are generated for all client nodes. The tour delay at a given time is computed based on the round trip delays for corresponding time periods.

To give a better idea of how the tour delays vary across clients on average, we averaged tour delays of all clients over two-week period. To find the average tour delay of a client for a specific (N, L) setting, all tour delays of the client for a given (N, L) configuration is averaged over the two-week period to get the average tour delay of a client



(a) Average tour delays of two-week period, $N = 4$.

(b) PDF of tour delay (unit: millisecond) when $N = 4$

Figure 2: The tour delays of clients when 4 tour guides are used.

for that (N, L) setting. Next, the average tour delays are sorted by least-to-most to provide a better view of the delay variation across clients. Figures 2a shows the average tour delays computed using this method for all client nodes when N = 4. Results for other values of N are skipped due to the space limitation, but they are very similar to the results shown here. The ratio of the largest and the smallest tour delays is around 5, when outliers are excluded. This disparity is several orders of magnitude smaller when compared to the disparity in available computational power (which can be in thousands [12] [13]). Figure 2b shows that majority of tour delays are clustered within a tight area of delay and the distribution of tour delays closely simulates a normal distribution.

2) Analytical Analysis: Although the PlanetLab data provides a somewhat good approximation of the delay characteristics of the Internet, it certainly has limitations. Due to the fact that the majority of the PlanetLab machines are connected to the Internet through campus networks, the delay data may not sufficiently reflect the diverse access network technologies that are used for connecting end hosts to the Internet. Next, we use latency data from the existing literature to show that even when clients are connected to the Internet using access technologies that provide very different delay properties, the disparity in their end-to-end round trip delays will be still smaller than the disparity in the computational power.

Let us take four very common access network technologies with very different delay characteristics: 3rd generation mobile telecommunications (3G), Asymmetric Digital Subscriber Line (ADSL), cable, and campus Local Area Network (LAN). The average access network delays are $200ms$ for 3G [36], $15 \sim 20ms$ for ADSL and cable [37], [38], and in the order of $1ms$ or negligible for campus LANs. Here, we refer to the access network delay as the round-trip delay between the end host and the edge router of the host's service provider. This latency is usually measured by measuring the round-trip delay to the first pingable hop. Based on the measurement analysis of the Internet delay space [39], the delay space among edge networks in the Internet can be effectively classified into three major clusters with average round trip propagation delays of about $45ms$ for the North America cluster, $135ms$ for Europe cluster, and $295ms$ for Asia cluster. Using these edge to edge propagation delay values and the average access network delay values, we can compute an average end to end round trip delays of 245, 335, and 495 ms for 3G hosts, 65, 155, and 315 ms for DSL & cable hosts, and 45, 135, and 295 ms for campus LAN hosts. The biggest disparity occurs between the hosts in the Asia cluster that connect through 3G and the hosts in the US cluster that connects through campus LAN, and the ratio of their round trip delays is $495ms/45ms = 11$. This disparity is about 4 times smaller than the low estimate of computational disparity provided in [26]. The round trip delays may get higher than $495ms$ due to congestion and high queuing delays in the intermediate routers. However, these congestions and high queuing delays affects all packets, regardless of whether they are from malicious clients or legitimate clients. Being able to persistently decrease the tour delay requires the attacker to compromise majority of the intermediate routers between itself and the tour guides, which is hard compared to minimizing computational puzzle solving time by adding more computing power.

C. Achieving Minimum Interference

In guided tour puzzle scheme, a client only has to send/receive packets to/from tour guides. To complete a guided tour puzzle with tour length L, a client only needs to send and receive a total of $2 \times L$ packets with about less than few hundred bytes (depending on the output size of the cryptographic hash function) of data payload. Since L is usually a small number in the order of tens, this creates negligible CPU and bandwidth overhead even for small

devices such as cellular phones.

D. Limitation

As with any other solution, GTP scheme has its limitation. First of all, GTP is a mitigation scheme that reduces the impact of DDoS, hence does not eliminate the effects of the DDoS entirely. The optimal defense that is achievable without being able differentiate malicious requests or detect malicious clients is to service all clients equally. The GTP scheme aims at being very close to such optimal defense. Secondly, the GTP scheme only works at the application/service layer, and relies on other solutions to provide protection against bandwidth flooding DoS attacks. Lastly, the tour delay seen by different clients for the same tour length can be different in the GTP scheme, and this may lead to different levels of service response time experience by different clients. However, note that this disparity in the tour delay does not discriminate against clients that possess less computational resources, and both malicious clients and legitimate clients are equally likely to experience longer tour delays. Having more computational power does not particularly help malicious clients complete a guided tour faster.

VII. Study of DDoS Defense Efficacy

In this study, we focus our evaluation on the ability of guided tour puzzles in mitigating the application layer DDoS attacks. We show that the guided tour puzzle scheme provides an optimal defense against request flooding attacks and a near optimal defense against puzzle solving attacks, when the server does not have the capability to differentiate the malicious clients from the legitimate ones.

A. Simulation Setup

To evaluate the effectiveness of guided tour puzzle in a practical simulation environment, we used Network Simulator 2 (NS-2) [40]. To create a network topology that can closely simulate large-scale wide area networks, such as the Internet, a topology with $5,000$ nodes is generated using the Internet Topology Generator 3.0 (Inet-3.0) [41]. The bandwidth and the link delay values are calculated based on the Inet-3.0 generated link distance values. Note that the link and queueing delays are different from one link to another, therefore the round trip delays, and consequently the tour delays, of different clients will be different.

Since client nodes, tour guides, and server nodes will be located in the edge in real networks, we use only degree-one nodes from the generated topology as client, server, and tour guide nodes. From a total of $1,922$ degree-one nodes, we randomly choose a degree-one node as the server node and another 20 degree-one nodes as potential tour guides. The remaining $1,901$ degree-one nodes are all used as client nodes, which includes both legitimate and malicious client nodes. The percentage of malicious client nodes is varied

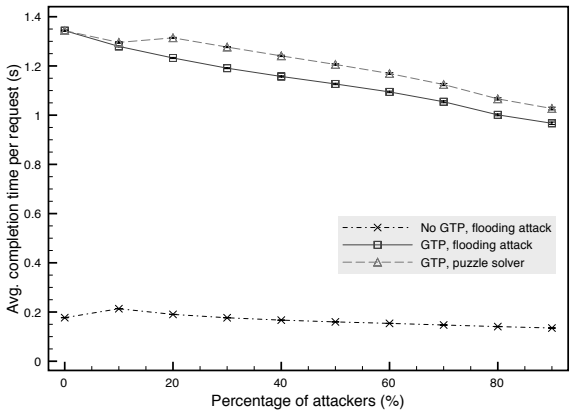
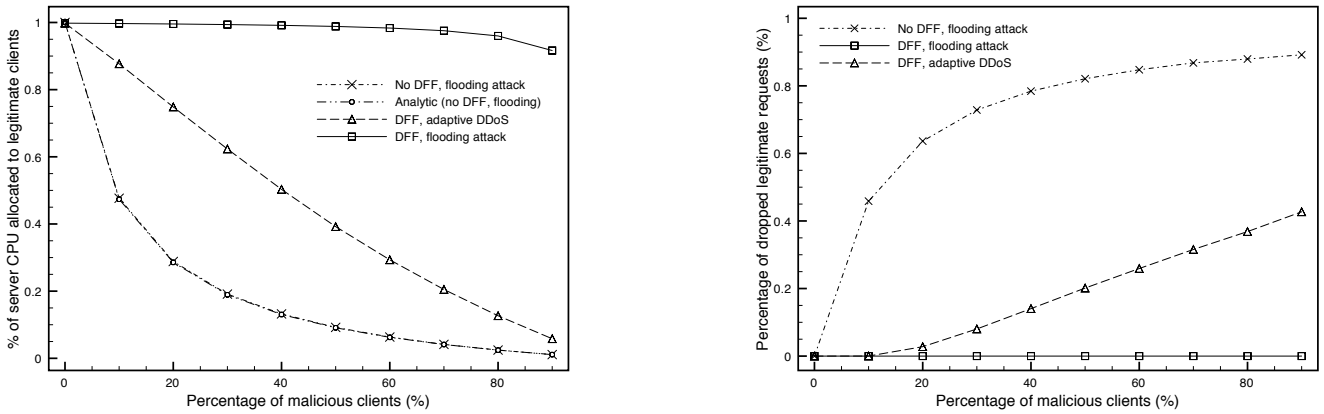


Figure 4: The cost of guided tour puzzles in terms of request completion times.

from 0% to 90% with an increment of 10%, and the server load is increased from 0.96 to 8.74 correspondingly. The server load is calculated as the ratio of the number of incoming requests per second to the server CPU capacity in requests per second.

A simulation model of the guided tour puzzle scheme is developed in NS-2. Clients in the simulation model generate self-similar traffic to closely mimic the Internet traffic. Since the self-similar traffic can be produced by multiplexing ON/OFF sources that have fixed rates in the ON periods and heavy-tail distributed ON/OFF period lengths [42] [43], each client application is implemented as an ON/OFF source with ON/OFF period lengths are taken from a Pareto distribution with shape parameter α equals to 1.5 (NS-2 default). The average ON and OFF times are set to 2 seconds. Each legitimate client sends at an average rate of 8000 bits per second. The average client request size is set to 1000 bytes, thus each legitimate client essentially sends requests at one request per second during on times, and 0.5 request per second on average. The average ON and OFF times, the client request size, and the server response size values selected based on the Web workload model introduced by Barford and Crovella [44].

The same client application model is used for malicious clients except that 1) malicious clients can choose to follow or not to follow the puzzle process, whereas legitimate clients always follow the puzzle process; and 2) malicious clients send requests at a higher rate than legitimate clients. We experimented with two different types of attacks — the flooding attack and the attack against the puzzle scheme. In a flooding attack, a malicious client sends requests at a high rate and ignores the server's request for solving puzzles. In the attack against the puzzle scheme, a malicious client solves puzzles as fast as they can to send requests at the maximum speed possible. The latter is a much stronger attack, since a server that deploys guided tour puzzle scheme can trivially filter out a malicious request that contains an



(a) Percentage of CPU allocated to legitimate clients

(b) Percentage of dropped legitimate requests

Figure 3: The effectiveness of guided tour puzzle against flooding attacks and puzzle solving attacks (N=4, L=8).

incorrect puzzle solution, while a malicious request that includes a correct puzzle solution consumes significantly more resources at the server.

The server's behavior is modeled as described in Section V. Since NS-2 does not provide a CPU model, the server's CPU is modeled as a link between the server node and a dummy node that is connected only to the server. When a client request arrives at the server, the server injects a packet with its size equals to the server response size into the link toward the dummy node. And when the packet is pinged back by the dummy node (which implies the completion of the server processing of the request), the server sends a response to the client. The capacity of this link is set to 80 Mbps to simulate a CPU processing capacity of $\frac{80Mbps}{8\times 10{,}000bytes} = 1{,}000$ requests per second, where the $10,000$ bytes is the average size of a server response. The server capacity of $1,000$ requests per second is used so that the server's full capacity can be reached when all clients are legitimate, and the server load can be increased by 100% with each increase of the percentage of malicious clients. A round-robin queue is used to model the CPU's round-robin process scheduling.

Using the average estimated client request rate of 0.5 request per second and the server CPU rate of $1,000$ requests per second, we can compute that the expected utilization of the server is $\frac{0.5\times 1901}{1000} = 0.9505$ when all the clients are legitimate clients. We achieved a utilization of 0.9656 for this setting in our experiments, which validates the correctness of our simulation setup.

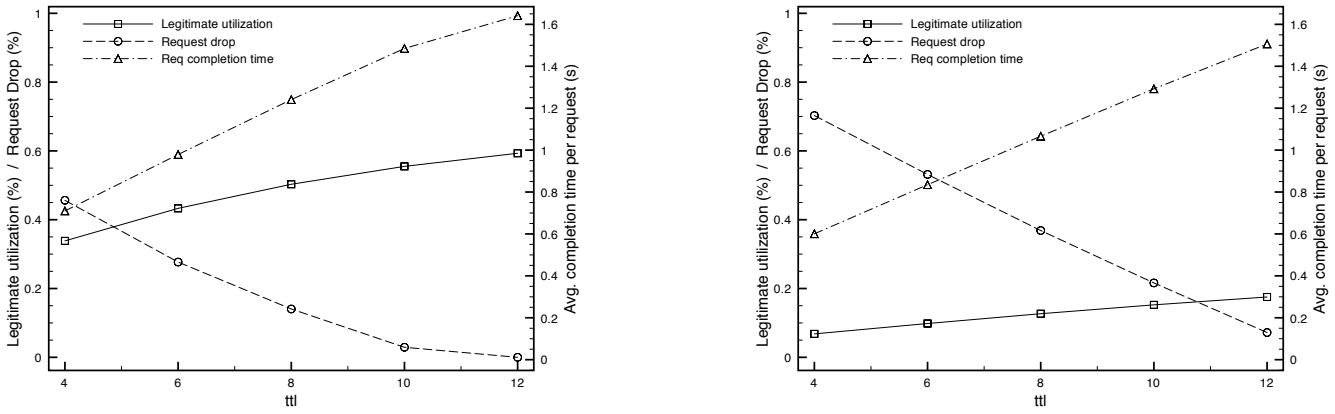
To keep the simulation simple, instead of using an adaptable tour length, a fixed tour length is used within a single run of the simulation. For each solved puzzle, clients are granted service for a single request. We may achieve significantly better protection against the denial of service attack by dynamically adapting the tour length and the number of granted requests per completed puzzle.

We use three evaluation metrics — average completion time of a single legitimate request, percentage of the server CPU allocated to legitimate requests, and percentage of dropped legitimate requests. The average completion time is calculated by recording the time spent between sending of a request and the receiving of its response, which includes the time spent on solving puzzles, for all completed requests of all the legitimate clients and taking the average. The percentage of the server CPU allocated to legitimate requests is computed as the fraction of the time the server's CPU is processing the requests of legitimate clients. The percentage of dropped legitimate requests is computed by dividing the total number of dropped legitimate requests by the total number of legitimate requests sent. For the results we report here, we set the simulation length of each run to 1000 seconds. For each simulation a warmup period of 100 seconds is used, after which recording of the evaluation metric measurement is started. Each experiment is repeated 10 times, and the average of 10 runs is reported along with a 99% confidence interval.

B. Simulation Results

The first set of simulations are conducted with a fixed tour length of 8 and using 4 tour guides. The results are reported in Figure 3 and 4.

1) Server CPU utilization: Figure 3(a) illustrates the improvement in the percentage of the server's effective CPU capacity that is allocated to processing the requests of legitimate clients. As the line "No GTP, flooding" (GTP means guided tour puzzle) indicates, the legitimate clients' share of the server's CPU capacity drops rapidly as the percentage of attackers increases when no guided tour puzzle is used. The percentage of server CPU allocated to processing legitimate requests in this case is predominantly decided by the ratio of total number of legitimate requests to the total number of requests. This can be validated by computing the percentage



(a) 40% attacker, N = 4 (b) 80% attacker, N = 4

Figure 5: The effect of the tour length on the effectiveness of the guided tour puzzle defense.

of legitimate requests for different percentage of malicious clients using the following formula.

$$\frac{r \times (1-x) \times N_c}{r \times (1-x) \times N_c \; + \; 10 \times r \times x \times N_c} = \frac{1-x}{1+9x} \qquad (7)$$

where, r denotes the request rate of legitimate clients, N_c is the total number of client nodes, and x is the percentage of malicious nodes. The line "Analytic (no GTP, flooding)" is then computed using the Formula (7), and it overlaps perfectly with the experiment results from the NS-2 simulation for the case of "No GTP, flooding attack".

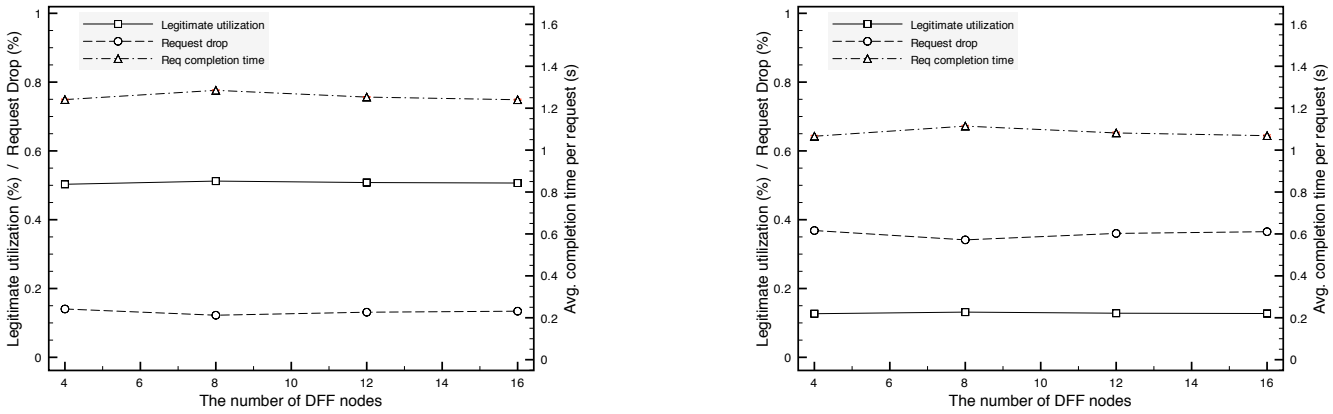
The top line "GTP, flooding attacker" in the Figure 3(a) shows that using guided tour puzzle eliminates the impact of flooding attackers entirely. In this scenario, the malicious clients do not solve any puzzle, but send requests that include fake puzzle solutions at a high rate in an attempt to consume as much server CPU capacity as possible. The slight decrease in the legitimate clients' utilization of the server CPU as the percentage of attackers increases is due to the increase in the percentage of server's CPU capacity allocated to verifying puzzle solutions. We intentionally used a low estimate of 10^6 hash operation per second as the server's hash computation rate to highlight the cost of puzzle solution verification.

The last line "Puzzle, solver" in Figure 3(a) is corresponding to the attack targeted at the guided tour puzzle scheme itself. It shows that the percentage of server CPU allocated to legitimate clients is roughly equal to the percentage of legitimate clients in the system when the guided tour puzzle scheme is used. We argue that without being able to differentiate legitimate clients from the malicious ones, the best a DoS prevention scheme can achieve is to treat every client equally and fairly allocate the server CPU to all the clients that are requesting service. Therefore, the optimal protection that a defense mechanism can provide without being able to differentiate malicious clients is to guarantee the legitimate clients the amount of server CPU that is equal to the percentage of legitimate clients in the system.

2) Request drops: Figure 3(b) shows the percentage of dropped legitimate requests. When no guided tour puzzle is used, the flooding attack caused legitimate clients to drop most of their requests as the line "No puzzle, flooding" indicates. When the percentage of attacker is increased to 90%, almost all legitimate requests are dropped as a result of the flooding attack. After switching to use guided tour puzzles (line "Puzzle, flooding"), the percentage of dropped requests becomes zero under the flooding attack even when the 90% of the clients are malicious. In the case of puzzle solving attacks, guided tour puzzle scheme reduces the legitimate request drops by more than half in all cases, and reduces the request drops to zero in some cases. In fact, the legitimate request drops can be eliminated entirely even in the case of puzzle solving attacks, as the simulation results in "effect of tour length" subsection show.

3) Request completion time: Of course, the benefit of using the guided tour puzzle scheme comes at the cost increased average request completion time, similar to any other "proof of work" based DoS defense mechanism. This cost is shown in the Figure 4. When guided tour puzzle is utilized, the average completion time of a request increased significantly in both flooding attack and puzzle solver attack cases, due to the extra delay introduced by the puzzle solving process. Nonetheless, the increase in the request completion time is within an acceptable range of degradation of service quality. Moreover, the guided tour puzzle scheme provides an easy way to achieve a better trade-off between two mutually restricting sets of quality of service goals by varying the tour length.

4) Effect of tour length: The tour length in guided tour puzzles is critical for the optimality of the guided tour puzzle defense, especially for the legitimate clients' utilization of server CPU in the case of puzzle solving attacks. The next set of simulation experiments are conducted to measure the



(a) 40% attacker, L = 8 (b) 80% attacker, L = 8

Figure 6: The effect of the number of tour guides on the effectiveness of the guided tour puzzle defense.

effect of tour length on utilization, request completion time, and request drops in the case of puzzle solving attacks. Configurations of 40% and 80% malicious clients are used in these experiments, and the number of tour guides N is set to 4.

The response of various metrics to the change in tour length is illustrated in Figure 5. As the tour length increases, the CPU allocated to legitimate clients ("Legitimate utilization") and the request completion time ("Req completion time") increase while the percentage of dropped legitimate requests ("Request drop") decreases. After increasing the tour length to 12, the percentage of dropped legitimate requests becomes zero, and the server CPU allocated to legitimate clients becomes optimal in both cases of 40% and 80% malicious clients. Here the optimal means legitimate clients are granted the amount of server CPU capacity that is equal to the percentage of legitimate clients in the system. Further increasing the tour length does not improve the utilization and request drop metrics and decreases the total utilization of the server CPU, while increasing the request completion time. The increase in the request completion time is evident since larger tour length means more round trips between clients and tour guides. These observations tell us that choosing the right tour length is important in achieving optimal DoS prevention results and providing better trade-off between mutually restricting metrics.

5) Effect of the number of tour guides: The last set of experiments are conducted to determine the effect of the number of tour guides on the effectiveness of guided tour puzzles. The 40% and 80% malicious clients are used, while the tour length L is set to 8. As the results in Figure 6 show, increasing the number of tour guides in the system does not produce any significant change in terms of all three metrics we are measuring. We can conclude from these results that guided tour puzzle can provide a good protection against the DDoS attack with just a few tour guides. Since the tour guides have a single function, which is replying to every request with the hash of the input message contained in the request, it is much easier to protect and maintain. The cost of hardware devices that can be used as tour guides likely to be significantly cheaper than over-provisioning by adding new servers. Moreover, a set of tour guides can be used to protect multiple servers, which further minimizes the cost per server by amortizing the total cost of tour guides over multiple servers.

VIII. Conclusion and Future Work

In this article, we showed that most existing cryptographic puzzle schemes do not consider the resource disparity between clients. We argued that resource disparity reduces or even nullifies the effectiveness of cryptographic puzzle schemes as a defense against denial of service attacks. To this end, we introduced the guided tour puzzle scheme, and showed that it achieves the desired properties of an effective and efficient cryptographic puzzle scheme. In particular, we showed how guided tour puzzles achieve puzzle fairness, minimum interference properties, and how it can achieve better defense against denial of service attacks. Meanwhile, using extensive simulation studies we showed that guided tour puzzle is very effective in mitigating distributed denial of service attacks, and that it is a practical solution to be adopted.

As future work, we would like to further improve the guided tour puzzle scheme in terms of the following. First, we would like to eliminate the need for the server's involvement in the puzzle generation process. Second, further investigation is needed to find out optimal ways to position tour guides in the network. Last but not least, adopting guided tour puzzle to defend against other application layer attacks, such as Sybil attack and spam, is also desirable.

References

[1] M. Abliz and T. Znati, "New approach to mitigating distributed service flooding attacks," in *the 7th International Conference on Systems (ICONS '12)*, Reunion Island, 2012.

[2] A. Juels and J. Brainard, "Client puzzles: A cryptographic countermeasure against connection depletion attacks," in *NDSS '99*, San Diego, CA, 1999, pp. 151–165.
[3] W. Feng, E. Kaiser, and A. Luu, "The design and implementation of network puzzles," in *IEEE INFOCOM '05*, 2005.
[4] T. Aura, P. Nikander, and J. Leiwo, "DoS-resistant authentication with client puzzles," in *8th International Workshop on Security Protocols*, vol. 2133, 2000, pp. 170–181.
[5] D. Dean and A. Stubblefield, "Using client puzzles to protect TLS," in *10th USENIX Security Symposium*, 2001, pp. 1–8.
[6] X. Wang and M. K. Reiter, "Defending against denial-of-service attacks with puzzle auctions," in *IEEE Symposium on Security and Privacy*, Oakland, 2003, pp. 78–92.
[7] C. Dwork and M. Naor, "Pricing via processing or combatting junk mail," in *CRYPTO '92*, 1992, pp. 139–147.
[8] X. Wang and M. K. Reiter, "Mitigating bandwidth-exhaustion attacks using congestion puzzles," in *CCS '04*, 2004.
[9] B. Waters, A. Juels, J. A. Halderman, and E. W. Felten, "New client puzzle outsourcing techniques for dos resistance," in *11th ACM CCS*, 2004, pp. 246–256.
[10] N. Borisov, "Computational puzzles as sybil defenses," in *P2P '06: Proceedings of the Sixth IEEE International Conference on Peer-to-Peer Computing*, Washington, DC, USA, 2006, pp. 171–176.
[11] H. Rowaihy, W. Enck, P. Mcdaniel, and T. L. Porta, "Limiting sybil attacks in structured p2p networks," in *the IEEE INFOCOM '07*, 2007, pp. 2596–2600.
[12] M. Abadi, M. Burrows, M. Manasse, and T. Wobber, "Moderately hard, memory-bound functions," in *NDSS '03*, 2003.
[13] C. Dwork, A. Goldberg, and M. Naor, "On memory-bound functions for fighting spam," in *CRYPTO '03*, 2003.
[14] M. Ma, "Mitigating denial of service attacks with password puzzles," in *International Conference on Information Technology: Coding and Computing*, vol. 2, Las Vegas, 2005, pp. 621–626.
[15] B. Groza and D. Petrica, "On chained cryptographic puzzles," in *3rd Joint Symposium on Applied Computational Intelligence (SACI '06)*, Timisoara, Romania, 2006.
[16] D. G. Andersen, "Mayday: Distributed filtering for Internet service," in *4th USENIX Symposium on Internet Technologies and Systems*, Seattle, WA, 2003.
[17] R. Thomas, B. Mark, T. Johnson, and J. Croall, "Netbouncer: Client-legitimacy-based high-performance DDoS filtering," in *3rd DARPA Information Survivability Conference*, 2003.
[18] S. M. Bellovin, M. Leech, and T. Taylor, "ICMP traceback messages," IETF Draft, 2003.
[19] S. Savage, D. Wetherall, A. Karlin, and T. Anderson, "Practical network support for IP traceback," in *ACM SIGCOMM '00*, vol. 30(4), Stockholm, Sweden, 2000, pp. 295–306.
[20] A. Yaar, A. Perrig, and D. Song, "SIFF: A stateless Internet flow filter to mitigate DDoS flooding attacks," in *IEEE Symposium on Security and Privacy*, 2004, pp. 130–143.
[21] X. Yang, D. Wetherall, and T. Anderson, "A DoS-limiting network architecture," in *ACM SIGCOMM*, 2005, pp. 241–252.
[22] M. Walfish, M. Vutukuru, H. Balakrishnan, D. Karger, and S. Shenker, "DDoS Defense by Offense," in *Proceedings of the SIGCOMM '06*, 2006, pp. 303–314.
[23] A. Stavrou, D. L. Cook, W. G. Morein, A. D. Keromytis, V. Misra, and D. Rubenstein, "WebSOS: An overlay-based system for protecting web servers from denial of service attacks," *Elsevier Journal of Computer Networks*, vol. 48, 2005.
[24] G. Price, "A general attack model on hash-based client puzzles," in *9th IMA Conference on Cryptography and Coding*, vol. 2898, Cirencester, UK, 2003, pp. 319–331.
[25] W. Feng, "The case for TCP/IP puzzles," in *ACM SIGCOMM Future Directions in Network Architecture*, 2003.
[26] B. Parno, D. Wendlandt, E. Shi, A. Perrig, B. Maggs, and Y. Hu, "Portcullis: Protecting connection setup from denial-of-capability attacks," in *ACM SIGCOMM*, 2007, pp. 289–300.
[27] R. L. Rivest, A. Shamir, and D. A. Wagner, "Time-lock puzzles and timed-release crypto," MIT, Cambridge, Massachusetts, Tech. Rep., 1996.
[28] S. Tritilanunt, C. Boyd, E. Foo, and J. M. González, "Toward non-parallelizable client puzzles," in *6h International Conference on Cryptology and Network Security*, 2007, pp. 247–264.
[29] M. J. Coster, A. Joux, B. A. Lamacchia, A. M. Odlyzko, C. Schnorr, and J. Stern, "Improved low-density subset sum algorithms," *Computational Complexity*, vol. 2(2), 1992.
[30] A. K. Lenstra, H. W. Lenstra, and L. Lovász, "Factoring polynomials with rational coefficients," *Mathematische Annalen*, vol. 261(4), pp. 515–534, 1982.
[31] M. Jakobsson and A. Juels, "Proofs of work and bread pudding protocols," in *the IFIP TC6/TC11 Joint Working Conference on Secure Information Networks*, 1999, pp. 258–272.
[32] H. Krawczyk, M. Bellare, and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication," RFC 2104 (Informational), Internet Engineering Task Force, Feb. 1997.
[33] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13(7), pp. 422–426, 1970.
[34] D. Mills, J. Martin, J. Burbank, and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification," RFC 5905, Internet Engineering Task Force, Jun. 2010.
[35] "About planet lab," Planet Lab, [Accessed: Dec. 20, 2012]. [Online]. Available: http://www.planet-lab.org/about
[36] J. Huang, Q. Xu, B. Tiwana, Z. M. Mao, M. Zhang, and P. Bahl, "Anatomizing application performance differences on smartphones," in *MobiSys '10*, 2010, pp. 165–178.
[37] M. Dischinger, A. Haeberlen, K. P. Gummadi, and S. Saroiu, "Characterizing residential broadband networks," in *IMC '07*, 2007, pp. 43–56.
[38] M. Yu, M. Thottan, and L. Li, "Latency equalization as a new network service primitive," *Networking, IEEE/ACM Transactions on*, vol. PP, no. 99, p. 1, May 2011.
[39] B. Zhang, T. S. E. Ng, A. Nandi, R. H. Riedi, P. Druschel, and G. Wang, "Measurement-based analysis, modeling, and synthesis of the internet delay space," *IEEE/ACM Trans. Netw.*, vol. 18, no. 1, pp. 229–242, 2010.
[40] VINT, "The network simulator - ns-2," 2009.
[41] J. Winick and S. Jamin, "Inet-3.0: Internet topology generator," University of Michigan, Tech. Rep. CSE-TR-456-02, 2002.
[42] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson, "On the self-similar nature of ethernet traffic," *IEEE/ACM Transactions on Networking*, vol. 2, no. 1, pp. 1–15, 1994.
[43] V. Paxson and S. Floyd, "Wide-area traffic: The failure of poisson modeling," *IEEE/ACM Transactions on Networking*, vol. 3, pp. 226–244, 1995.
[44] P. Barford and M. Crovella, "Generating representative web workloads for network and server performance evaluation," *SIGMETRICS Perform. Eval. Rev.*, pp. 151–160, 1998.

A Distributed Hash Table Assisted Intrusion Prevention System

Zoltán Czirkos, Márta Rencz, and Gábor Hosszú
Department of Electron Devices
Budapest University of Technology and Economics
Hungary, H-1117 Budapest, Magyar tudósok körútja 2.
{czirkos,rencz,hosszu}@eet.bme.hu

***Abstract*—Using collaborative intrusion detection to sense network intrusions comes at a price of handling an enormous amount of data generated by detection probes, and the problem of properly correlating the evidence collected at different parts of the network. The correlation between the recorded events has to be revealed, as it may be the case that they are part of a complex, large-scale attack, even if they manifested at different parts of the network. In this paper we describe the inner workings a peer-to-peer network based intrusion detection system, which is able to handle the intrusion detection data efficiently while maintaining the accuracy of centralized approaches of correlation. The system is built on a distributed hash table, for which keys are assigned to each piece of intrusion data in a preprocessing step. The network traffic requirements of such a system, and the load balancing that can be achieved by using the Kademlia peer-to-peer overlay network are discussed as well.**

Keywords-collaborative intrusion detection; attack correlation; peer-to-peer; distributed hash table; Kademlia.

I. Introduction

In the earliest days of the Internet, services on the network were all based on trust. As e-commerce emerged, network hosts became victim of a wide range of everyday attacks. Due to the high amount of confidental data and resources that can be exploited, the possibilites and open nature of the Internet opened serious security questions as well.

The attacks network administrators fight against are both human and software initiated. They get more and more sophisticated, originating or targetting ocassionally multiple hosts at the same time. A large number of nodes can be simultaneously scanned by attackers to find vulnerabilities. Automatized worm programs replicate themselves to spread malicious code to thousands of vulnerable systems, typically of home users. Others compromise hosts to build botnets, which can deliver millions of spam e-mails per day.

As the manifestation of attacks, e.g., the evidence that can be observed is spread across multiple hosts, these large-scale attacks are generally hard to detect accurately. To recognize such, one has to first collect or *aggregate* the evidence, then *correlate* the pieces of information collected [1]. A collaborative intrusion detection system has to analyze the evidence from multiple detector *probes* located at different hosts, and even on different subnetworks [2]. However, this poses several problems to solve:

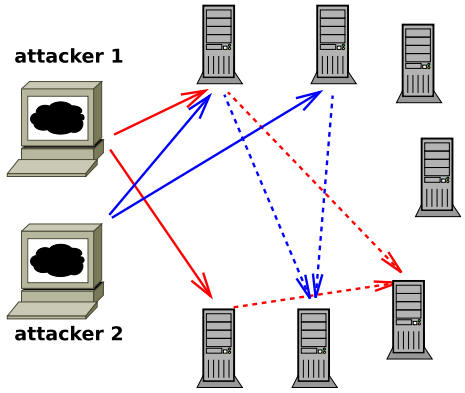


Figure 1. Messages carrying attack information in the *Komondor* system. If any probes in the network detect a suspicious event (solid lines), it sends a report to the DHT (dashed lines). The nodes of the DHT act as correlation units as well, and are able to collect these reports.

- large quantities of possible evidence collected,
- including inadequate data for precise decision making,
- communication and reliability problems,
- frequent change of intrusion types and scenarios.

Some of these troubles are specific for the isolated, host-based detection systems, while others occur only in case of the network scale intrusion detection. Despite of all these difficulties it is still worth collecting and correlating evidence available at different locations for the efficiency and accuracy boost of both detection and protection [3].

In this paper, we present a collaborative intrusion detection system, which organizes its participants to a *peer-to-peer (P2P) overlay network.* For intrusion data aggregation, a *distributed hash table* (DHT) is used, which is built on the Kademlia topology. This is used to balance the load of both aggregation and correlation of events amongst the participants. The organization of nodes in the overlay network is automatic. Should some nodes quit or their network links fail, the system will reorganize itself.

The rest of this paper is organized as follows. In Section II, we first review existing research of collaborative intrusion detection systems. Then we present the architecture of our intrusion detection solution based on the Kademlia DHT overlay in Section III. The results of our intrusion

detection method and statistics of detection are highlighted in Section IV. Research is concluded in Section V.

II. Related Work

Attackers use various ways for intrusion of computer network systems depending on their particular goals. These methods leave different tracks and evidences, called the *manifestation of attacks* [4]. To discuss the internals of a collaborative intrusion detection system, we use the following terms [2]:

- *Suspicious events* are primary events, that can be detected at probes. Not necessarily attacks by themselves, but can be part of a complex attack scenario.
- *Attacks* are real intrusion attempts, which are used to gain access to a host or disturb its correct functioning. Usually these are made up from several suspicious events at once.

The activity of an SSH (Secure Shell, a remote login software) worm program can be seen as an example of an attack. These worms use brute-force login attempts using well-known user names and simple passwords [5], directed against a single host. The attempts are events that make up the attack in this case. Multiple failed login attempts usually indicate an attack, while a single failed attempt is usually only a user mistyping his password.

A. Centralized Collaborative Intrusion Detection

Generally, large-scale attacks can only be detected by *collecting* and *correlating* events from a number of detector probes. The collection of evidence has to be extended to suspicious events as well, which otherwise do not necessarily suggest attacks themselves. In order to achieve this, various collaborative intrusion detection systems (CIDS) have been proposed, for which a detailed overview can be found in [3].

The earliest collaborative detection systems used a centralized approach for *collecting* the events, as seen on Figure 2. The Internet Storm Center *DShield* project collects firewall and intrusion detection logs from participants, uploaded either manually or automatically [6]. The log files are then analyzed centrally to create trend reports.

The *NSTAT* system [7] on the other hand is more advanced, since its operation completely real-time. In NSTAT, the detection data generated by the probes is preprocessed and filtered before being sent to a central server for correlation. This system analyzes the order of events using a state transition mechanism with predefined scenarios to find out the connection between them.

The advantage of centralized methods is that the server is able to receive and process all data that could be gathered, i.e.,it has all the information necessary to recognize the intrusion attempt. The correlation can be carried out with several different methods. SPICE [8] and CIDS [9] group events by their common attributes. The LAMBDA system tries to fit events detected into pre-defined and known scenarios [10]. The JIGSAW system maps prerequisites and consequences of events to find out their purposes [11].

Centralized solutions have two drawbacks to address. The first one of these is scalability – the high amounts of data to be aggregated and correlated for large networks cannot be handled by a single *correlation unit*. The second one is that the correlation unit is a single point of failure, being even a possible target of attack for shutting down the whole intrusion detection system.

B. Hierarchical and P2P Collaborative Intrusion Detection

By using hierarchical approaches, the scalability problem of centralized intrusion detection systems can be handled. The *DOMINO* system is used to detect virus and worm activity. It is built on an unstructured P2P network with participants grouped into three levels of hierarchy [12]. The nodes on the lowest level generate statistics hourly or daily, therefore they induce only a small network traffic.

The *PROMIS* protection system (and its precedessor, Netbiotic) uses the JXTA framework to build a partly centralized overlay network to share intrusion evidence [13]. The nodes of this system generate information for other participants about the frequency of detected suspicious events. This information is used to fine-tune the security settings of the operating system and the web browser of the nodes. This creates some level of protection against worms, but also decreases the usability of the system.

The *Indra* system is built on the assumption that attackers will try to compromise several hosts by exploiting the same software vulnerability [14]. If any attempts are detected by any participant of the Indra network, it alerts others of the possible danger. Participants can therefore enhance their protection against recognized attackers, rather than developing some form of general protection.

The scalability and single point of failure problems of centralized solutions can also be solved by using structured P2P application level networks. The P2P communication model enables one to reduce network load compared to the hierarchical networks presented above.

The CIDS system [9] is a publish-subscribe application of the Chord overlay network [15]. Nodes of this system store IP addresses of suspected attackers in a blacklist, and they subscribe in the network for notifications that are connected to these IPs. If the number of subscribers to a given IP address reaches a predefined threshold, they are alerted of the possible danger. The Chord network ensures that the messages generated in this application will be evenly distributed among the participants.

The *BotSpot* system aims to discover traffic patterns generated by botnets in recorded NetFlow data [16]. By dropping specific IP addresses from the data to be analyzed, anonymity can also be ensured for its users. The *Spamwatch* system aims filtering of spam messages [17]. It uses a Tapestry-based peer-to-peer network to store data of mail

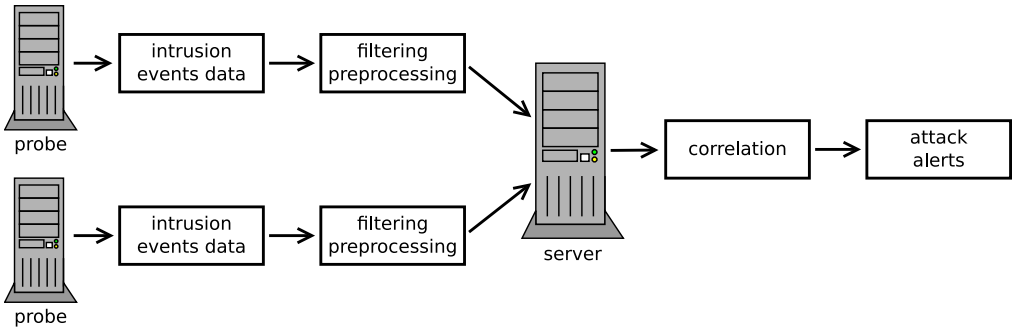


Figure 2. Collaborative detection system with centralized collection and correlation of data from probes. Every piece of information is sent to a server, which handles the correlation, and has the responsibility of alerting participants when an attack is detected.

messages that are tagged as spam by the users of the Spamwatch community [18]. Other users' mail applications can then automatically delete known spam messages.

C. Structured P2P Networks

The intrusion detection systems mentioned above use various P2P substrate networks. By selecting a proper substrate, the traffic generated in a specific application of the network can be reduced.

Structured P2P networks generally implement *distributed hash tables* [19]. DHTs store $\langle key; value \rangle$ pairs and allow the quick and reliable retrieval of any $value$ if the key associated to that is known precisely. This is achieved by using a *hash function* and mapping all data to be stored to the nodes selected by the distance of the hashed keys and their NodeIDs, which are chosen from the same address space. The connections between nodes are determined by their NodeID selected upon joining the network. They are selected so that the number of steps between any two node is usually in the order of $\log N$, where N is the count of all nodes.

DHTs all implement routing between their nodes on the application level to build the topology desired. For the small network diameter however, only some of these are feasible. The *Chord* DHT, for example, arranges its nodes into a ring [15]. To reduce the number of hops required for sending a message, it uses auxiliary network connections, which enable nodes to send message to the opposite side of the ring, and it divides the network to smaller pieces, which are half of the original in every step.

The *Kademlia* network uses a binary tree topology [20], as seen on Figure 3. All Kademlia nodes have some degree knowledge of the successively smaller subtrees of the network they are *not* part of. For any of these subtrees they have routing tables called k-buckets, which store IP addresses of nodes that reside in distant subtrees. When a node looks up a selected destination, it successively queries other nodes, which are step by step closer to the destination. The queried nodes answer by sending their k-buckets to the source. As

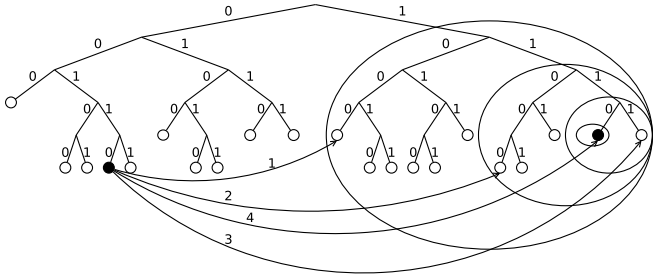


Figure 3. Sequence of lookup messages in the Kademlia overlay network. The node initiating the message successively queries nodes closer to the destination, so it finally receives its IP address for direct communication. (For details of the lookup procedure, see [20].)

nodes closer to the destination have greater knowledge of their neighbors, the lookup will get closer every step, as discussed in [20]. The distance in the binary tree is halved with every message, so the number of messages is $\log_2 N$ with N being the number of nodes in the tree.

DHTs map all $\langle key; value \rangle$ pairs to the nodes, which have their NodeIDs closest to the hashed value of the key. The distance function used depends on the topology of the network. Kademlia uses the XOR function to calculate the distance, which captures the topology of the binary tree well, as the magnitude of the distance calculated with $d(A, B) = A \otimes B$ is the height of the smallest subtree containing them both. The k-buckets are sorted by decreasing distance. The advantage of Kademlia is great flexibility: for the correct functioning of the lookup procedure, any nodes can be put in any of the k-buckets, as long as they are in the correct subtree.

III. The Komondor System Architecture

In this section, our intrusion detection system named *Komondor* is presented. Its most important novelty is that it uses the Kademlia DHT as a substrate network to store intrusion data and to disseminate information about detected events [1]. Having analyzed the collected events, *Komondor*

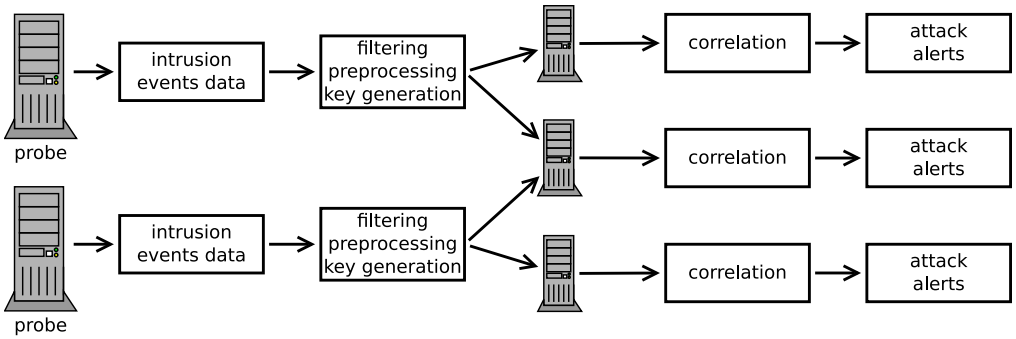


Figure 4. Distributed collection and distributed correlation of intrusion evidence from various probes. The *Komondor* system assigns keys to pieces of evidence so that data can be stored efficiently in a DHT. By using these keys, computational load of correlating can be distributed among several units.

correlation units may start an alert procedure notifying other nodes of the possible danger if necessary.

A. Distributing Load Among Multiple Correlation Units

The *Komondor* peer-to-peer application level network consists of multiple nodes. All nodes have the *responsibility* of collecting and correlating intrusion data. They also report attacks discovered to other nodes of the network, as seen in Figure 1. All participants of the *Komondor* network serve as intrusion detection units and correlation units as well.

The *Komondor* network is designed to enable the correlation methods mentioned in Section II to be used in a distributed manner:

- Pieces, which are correlated should be sent to the same correlation unit, so that it can gather all the information about the attack.
- Pieces of evidence, which are part of distinct ongoing attacks should preferably be sent to different correlation units. This reduces load and improves overall reliability of the system.

Komondor achieves this goal by *assigning keys to preprocessed intrusion data*, as seen in Figure 4 (cf. Figure 2). Keys assigned are used as storage keys in the DHT as well. For different attackers or attack scenarios, different keys are selected, and this way data is aggregated at different nodes of the *Komondor* overlay.

This is different from other P2P distributed intrusion detection networks, in which only one attack correlation method is used. In Komondor, attack correlation and event aggregation is decoupled by the means of selecting a key in an early phase of correlation, and using it as a DHT key for storage. The *Komondor* system is essentially a *middle layer inserted into the intrusion detection data path.*

Correct key selection is critical, since pieces of evidence, which might be correlated to each other must be assigned the same key and sent to the same *Komondor* node for correlation. Note that these pieces do not have to be detected by the same probe, yet they can be aggregated by the same correlation unit. The nodes of the DHT are the correlation units, which have to implement the same correlation methods as their centralized counterparts. The correlation procedure is started as soon as the preprocessing stage with the key selection, and it is finalized at the correlation units.

The detected and preprocessed data of suspicious events is stored in the *Komondor* overlay. In this system, the key assigned at the preprocessing stage of detection is used as a *key for DHT operations* as well. The value parts of the $\langle key; value \rangle$ pairs stored are any other data, which might be useful for detection or protection. As all nodes use the same key selection mechanism and the same hash function, events related to each other will be stored at the same node, as seen in Figure 1. This way the algorithm ensures that the aggregator node has perfect knowledge of all events related to the attack in question, and is able to recognize the attack as well.

The reason why a structured overlay was selected for the *Komondor* system is that it combines the advantages of both the distributed and centralized detection systems. Event data collected has to be sent to a single collector node only (this would not be possible with an unstructured overlay, as those have no global rule to map a key to a node.) Moreover, when *Komondor* nodes are under multiple but unrelated attacks, the network and computational load of both aggregation and correlation is distributed among nodes. Moreover, the *Komondor* system does not have a single point of failure: the responsibility of correlating particular events is transferred to another node in this case. The overlay can also be used to disseminate other type of information as well, for example the attack alerts, which enable nodes to create protection.

B. Kademlia as the DHT Topology of Komondor

The nodes of *Komondor* create a *Kademlia DHT* overlay. This is the topology, which can adapt its routing tables to the dynamic properties of traffic generated by the intrusion detection probes. As discussed below, other DHTs wouldn't be able to adapt their routing tables to the dynamic properties of this kind of traffic.

Storing information of events generated by the probes

Table I
NUMBER OF MESSAGES IN STRUCTURED OVERLAYS FOR INTRUSION DETECTION

Overlay	Chord	Kademlia
Routing algorithm	recursive	iterative
Node lookup	0	$\log_2 N$
First event stored	$\log_2 N$	$1 + \log_2 N$
n **events with the same key**	$n \cdot \log_2 N$	$n + \log_2 N$
Average number of messages per event	$(n \cdot \log_2 N)/n$	$(n + \log_2 N)/n$
Average number of messages with $n \to \infty$	$\log_2 N$	1

generates significant overlay traffic, which will load not only detector and collector nodes, but other nodes along the path from the former to the latter one as well, as routing between nodes is handled on the application level. If the *events are in correlation* with the same attack, the *key chosen is likely to be the same*, making the distribution of keys highly uneven. However, by using Kademlia, network traffic can be significantly reduced in this scenario. The reason for this is that the routing algorithm of Kademlia is very flexible: any node can be put to the routing tables of any other node while still obeying the rules of the routing protocol. Routing tables of other DHT overlays like CAN or Chord are much more rigid, and therefore the routing algorithm of those cannot optimize the number of messages for the store requests with the same key.

Table I compares the number of messages generated in intrusion detection for Kademlia and Chord, with the latter being an example for having rigid routing tables. Chord uses a *recursive routing mechanism,* which means that messages are forwarded by overlay nodes along the path from the source to the destination of the message, as seen on Figure 5. If *Komondor* would be built on Chord, the number of messages generated in the overlay would be in the order of $\log_2 N$ for each detected event, where N is the node count of the overlay.

Contrary to Chord, Kademlia uses an iterative algorithm. To store a $\langle key; value \rangle$ pair, a Kademlia node first looks up the IP address of the destination node by successively querying nodes closer to the destination. After finding out its address, data is sent directly from the source and the destination. This also implies that the payload of the message is contained in every message for Chord, and only in the last message for Kademlia. For Kademlia, the node has to first look up the address of the destination, which also takes $\log_2 N$ messages. Having done that, it requires one more message $(+1)$ to send the payload as well. If multiple events are to be stored, which are detected by the same probe (this is a likely scenario for a node that is under attack), the *lookup procedure can be optimized away*, as the key and therefore the collector node is the same, too. For sending data of n events, the number of messages generated is only $n + \log_2 N$ for Kademlia and $n \cdot \log_2 N$ for Chord, which is worse at the factor of n for the latter one. The limit of messages per event drops to 1 for Kademlia in this common intrusion detection scenario.

The above optimization is made possible by the fact, that any node can be inserted to the routing tables of any other arbitrarily selected node in Kademlia, while still obeying the selection rules of the protocol. The k-buckets of the nodes cover the whole NodeID space of the binary tree, and the exact selection of nodes do not affect the correctness of the lookup mechanism, only its latency properties. The original Kademlia paper [20] suggests that nodes with long session uptimes are selected for routing, which is feasible in file sharing applications to enhance reliability. *Komondor* nodes, which are selected by attack events to be stored, should be selected to reduce network traffic.

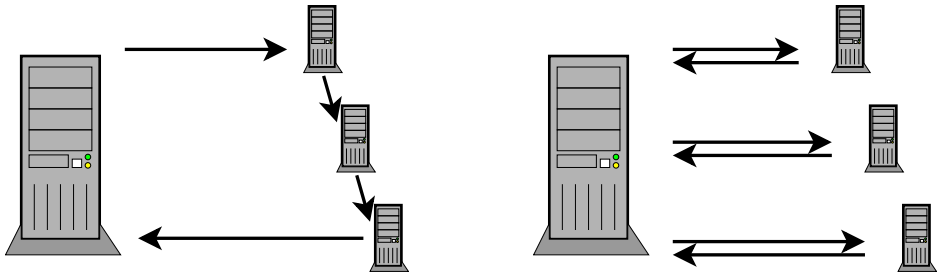
The *Komondor* system uses does not use the data lookup mechanism (looking up a value associated with a specific key) of the DHT as other applications do. Only the data store mechanism is used. Stored events are never looked up, rather the node, which stores them has to process incoming events to recognize attackers. The collector nodes have the responsibility to start a broadcast algorithm [21], if an attack is recognized. The broadcast message must contain data, which can be used by participants to create their own protection.

C. Selection of Keys in the Komondor System

The accuracy of detection, also network and computational load balancing depends on the proper selection of keys. If, at preprocessing stage, the correct key is failed to be chosen, pieces of evidence may mistakenly end up at different correlation units, and therefore the attack may remain unnoticed. Detection efficiency can be increased by assigning more keys, should an event be suspected to be a candidate for being part of different attacks or attack scenarios. One can also implementing several correlation algorithms simultaneously. However, every subsequent key increases network traffic as well.

Examples for key selection include the source or destination IP addresses of offending packets. For every large-scale network scan scenario, a different key selection mechanism is feasible. Consider the network scan types categorized in [12]:

- *Horizontal port scan.* Different hosts are scanned by a attackers, but the port number, e.g., the vulnerability searched for is the same. In this case, a blacklist of attackers can be built using the collection and correlation of detected attempts. The key for the *Komondor* overlay in this case is the identifier of the vulnerability, or the port number.
- *Vertical scan.* A single host is under attack. The attack originates from a single host, too. If this is the case,



(a) recursive (b) iterative

Figure 5. Routing methods in DHT overlays. In overlays with recursive routing, messages are forwarded from node to node. The iterative method requires nodes to look up the address of the destination of the message themselves.

the attacker is known and hosts can protect themselves against it, should it try to attack another friendly hosts. The DHT key should be the IP address of the attacker.

- *Mixed scan.* Multiple attackers use their network capacity to launch an attack against a single host or a subnetwork. This is the usual scenario for the well known DDoS (distributed denial of service) attacks [22], the goal of which is to disrupt some service of an on-line service provider by overloading its network or computational capacities. The key for the evidence storage in the DHT in this case is the subnetwork address attacked. By analyzing the data collected in this scenario, hosts can automatically detect the fact of the network scale attack, e.g., they can discover that the problem is not only related to a single host but a complete subnetwork or organization.

Apparently, the achievable benefit of the collaborative detection for these scan methods also varies with their type and intent.

IV. Results and Discussion

In this section, we present statistics of intrusion attempts detected using the implemented *Komondor* system. The statistics are evaluated to show which types of attacks this system can be used to detect.

The implementation used for testing was written in C++, and run on various versions of Ubuntu, Debian Linux and OpenBSD operating systems. The systems protected provided HTTP, SSH, mail, SQL and other services to their users. The number of probes in the system varied from 7 to 10, each with their own, public IP address. The overlay created was not limited to a single subnetwork.

The present *Komondor* implementation used the open-source *Snort intrusion detection system* [23] to detect intrusion events, and it could collaborate with other host-based intrusion detection solutions as well. The key selected for each event was the *IP address* of the attacker, as found in the Snort log file. It was also used for correlation. We selected common event types from the Snort database and also tagged events with a severity score. Intrusion alert was triggered when the sum of these scores reached a threshold level. This simple correlation method enabled us to determine the efficiency and reliability of the *Komondor* system for known attack types. Data presented here was collected in a three year interval. During this time, 17,088 attacks were detected, with the maximum number of attacks originating from a single IP address being 811. The number of individual events for a single attack reached as much as 80,000 events for some of the worm attacks recorded. The number of nodes in the small *Komondor* test overlay was around 7 and 15 nodes, with most of them being on the same subnetwork.

One of the nodes of the test overlay was assigned special logging tasks. This was achieved by fixing the NodeID of that node to the hexadecimal value 0x00000001. (Our implementation used 32-bit NodeIDs, rather than using the full 160-bit space as usual in Kademlia networks.) The attack storage method used in other nodes was modified to send all data to this node as well, besides sending the events to the nodes as selected by the keys. This anchor node generated statistics, and provided us with a monitoring interface accessible through a web browser.

A. Attack Intervals and Number of Events

Figure 6 shows invalid passwords detected for SSH login attempts on various hosts [5]. Every dot on the graph is an individual attack. The y axis shows the number of events or the number of invalid passwords detected. The duration of an attack is the time interval between the first and the last event detected, and is on the x axis. Several attackers were detected by multiple *Komondor* probes, because the SSH worm that was trying to gain access to the subnetwork tried to login all on-line hosts it found. The number of probes, which detected an attack in question is shown by the color of the dots. (In the case of multiple probes detecting an attacker, the event number on axis x is an average per probe.)

Apparently the attacks, which were detected by one probe only (black dots) have much less events associated to them. The 1,100 attacks shown on the graph have as much as 450 of them stacked up in the $(1; 1)$ point. These evidently came from human interaction. Attacks detected by multiple probes

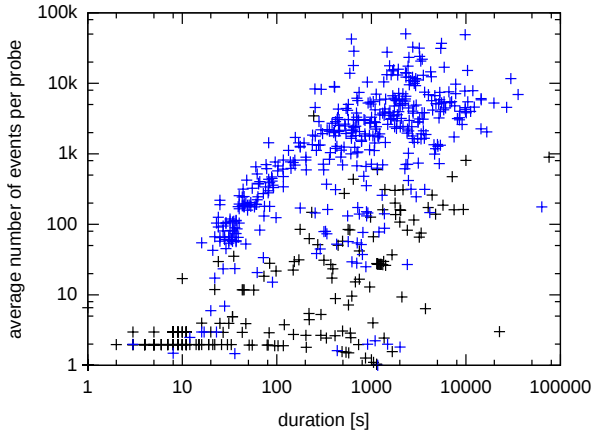


Figure 6. Number of invalid password events detected for various attacks (y axis) plotted by the duration of the attack (x axis), as detected by the *Komondor* test overlay network. The color of the dots represent the number of probes a specific attacker was detected by.

usually suggest automatic worm programs using dictionary attacks against the detector hosts.

This experience suggests that distributed intrusion detection can benefit from the advantages of DHTs:

- Attackers could be detected by several probes at the same time. When multiple hosts are attacked, recognizing an attacker using any evidence from any probe of the *Komondor* network, several hosts could be protected using firewalls at the same time, which might promptly be attacked, too.
- Attack evidence came from multiple probes. One attack is likely to be associated to thousands or tens of thousands of events, which must be stored and processed in the overlay. This type of load can be dealt with the DHT fairly well, as it can select different collector nodes for each individual attack and therefore balance the load.
- When detecting an event, which generates the same key, the Kademlia DHT can significantly reduce network traffic, as the IP address of the collector nodes have to be looked up only once. When the IP address is obtained, the system works as if it were using a centralized approach with the same benefits as those.

B. Attack Types and Confidence

Table II shows various attack types and the efficiency for the *Komondor* system regarding protection. The column *Protection* shows the number of attacks for each type, for which the attack continued after it was blocked on the firewall, and the activity of the attacker was detected by another *Komondor* node of the same subnetwork. For these attacks, the collaborative intrusion detection can greatly enhance the protection of hosts.

Figure 7 shows event numbers and attack durations for different worms attacking SQL servers. The y axis has two scales for each graph. The scales of the left hand side show attack durations (red plot), and the right hand side scale shows the number of events (blue plot). Attacks are sorted by duration. Every value on the x axis is an attack for which the duration and the number of events is shown right under each other.

Table II
NUMBER OF ALL ATTACKS AND ATTACKS FOR WHICH PROTECTION COULD BE BUILT BY *Komondor*, FOR EACH ATTACK TYPES.

Type of attack	Attacks	Protection	Ratio
phpMyAdmin scan	107	71	66%
MSSQL overflow	4355	15	0%
SSH connection lost	490	321	65%
SSH failed password	546	219	40%
SSH invalid user	51	47	92%
FTP failed login	46	2	4%

A worm, which scanned the Web servers for vulnerabilities via HTTP requests is shown on the right hand side subfigure. For any event detected, the IP address of the attacker can be recognized by the correlation units. The left hand side graph presents the properties of the Slammer worm, which penetrates outdated MSSQL servers. This worm does not issue more attempts in a short time interval to the same host, and selects IP addresses of victims randomly. For detecting this type of attacks, the PROMIS and CIDS systems could be used more effectively.

Figure 8 is similar to Figure 7, showing the attack interval and the number of events for attacks. However, invalid SSH login attempts are visualized on this one. The figure shows real attacks and mistyped passwords as well. The left hand side subfigure shows the invalid user name events, and the right hand side subfigure the invalid password events. The usual user interfaces of SSH remote login software show the login names to the users as they type, while the password is hidden for security reasons. This implies that mistyped login names rarely come from authorized users, as they would correct it before sending it to the server. Almost all of this type of attacks are conducted automatically by worm software. However, 40% of detected mistyped password attacks have only one event, and supposedly come from authorized users. These are all false alarms in an automatic intrusion protection system like *Komondor*.

C. Load Balancing Potential of Using Hash Functions for IP Addresses as Keys

Figure 9 shows the distribution of events in IP address space and in overlay key (NodeID) space. The figure shows only the events related to SSH worms.

The IP addresses on the top part of the figure were mapped to the rectangular area using a Hilbert space filling curve. This mapping renders the 32 bit address space in such a way,

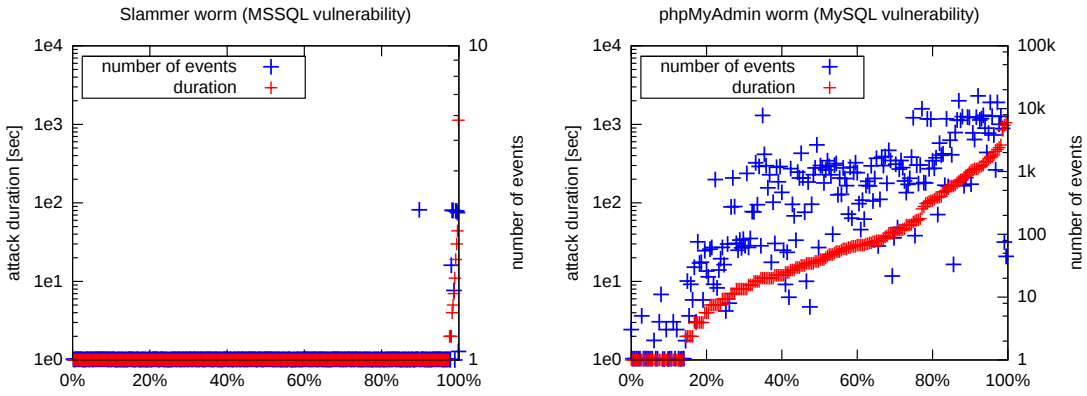


Figure 7. Attack intervals and number of events for different worm activities detected by the *Komondor* system. The left hand side shows a worm, which scanned our Web servers via HTTP in order to find a phpMyAdmin installation to gain access to MySQL databases. On the right hand side the activity of the infamous Slammer worm is shown, which penetrates MSSQL servers.

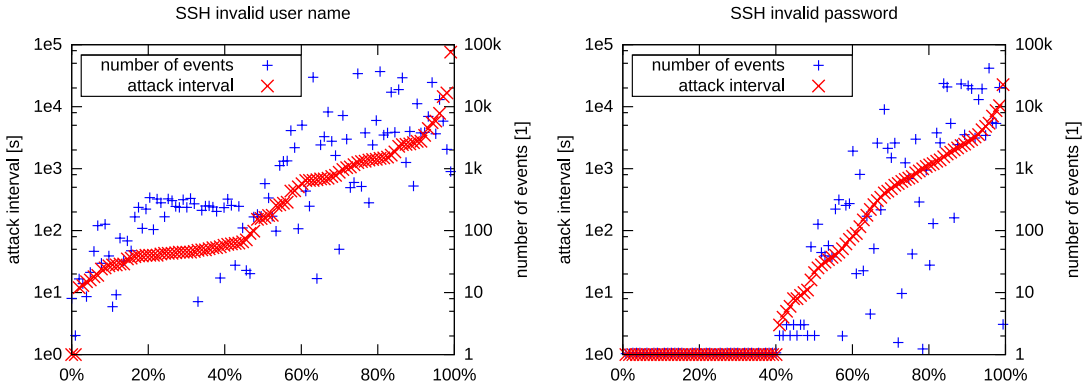


Figure 8. Attack intervals and number of events for SSH login attempts, as detected by the *Komondor* network. An invalid login name almost inevitably suggests an attack, while an invalid password may come from an otherwise authorized user.

that addresses close to each other (therefore, addresses in the same subnetwork) are close to each other. For example, the 0/8 to 63/8 range is in the upper left quarter square, and the 0/16 to 15/16 range in the upper left sixteenth. The first octet of the address determines the numbered square, and the second octet was used to select the place inside every small square similarly, so that dots do not cover each other needlessly.

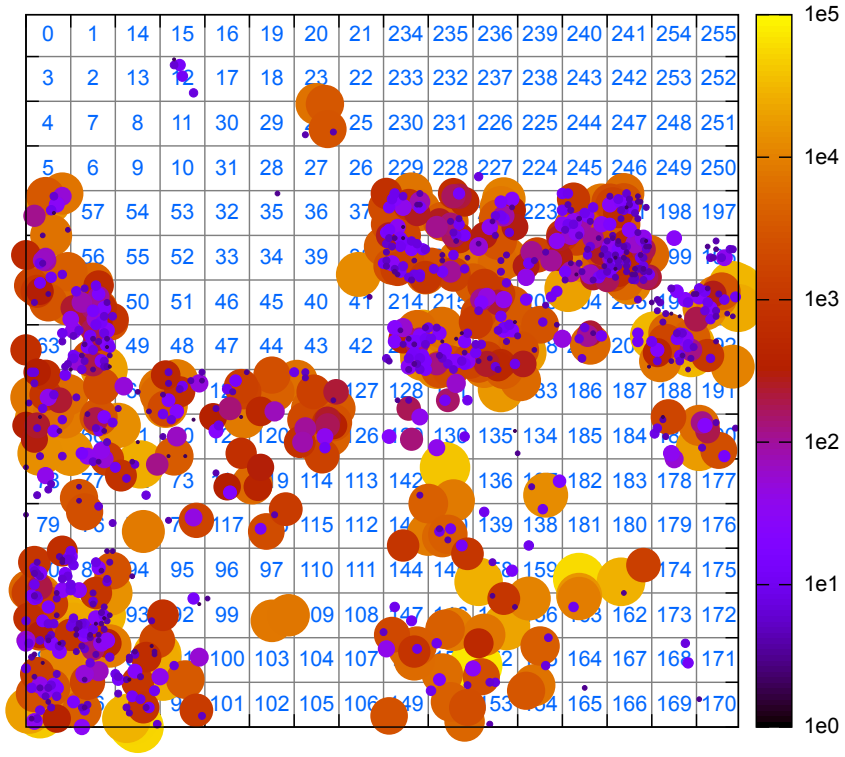
The size and color of the nodes show the number of events for each attack. The number of events related to each attack is quite different for every attack, the difference has a magnitude of about four. If we are using IP addresses for correlation, the hash functions used in the structured overlays for data to node mapping can significantly reduce this, as seen on the bottom side subfigure.

The *Komondor* reference implementation used 32-bit hashed addresses. The bottom side plot on Figure 9 shows the number of events by their hashed values, the first octet of which values are used for the x coordinate, and the next eight bits for the y coordinate. The magnitude of the difference between the highest and lowest number of messages that are related to a single attack could be reduced by 1.85, i.e., about 70 times lower.

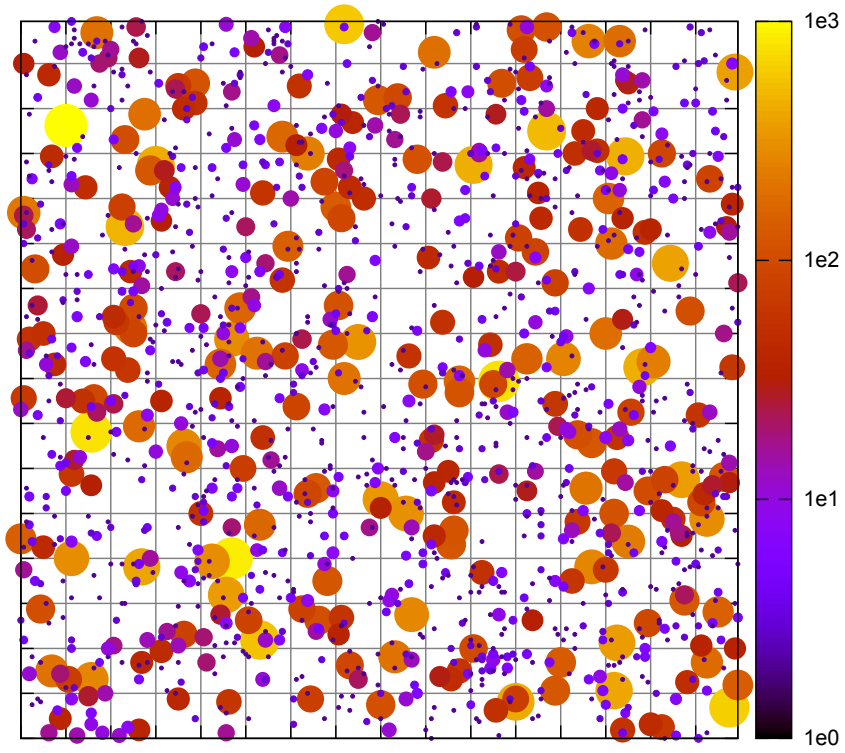
Of course, a single detector node can still detect and send many events to the same collector node, when being under attack. This load imbalance can not further be reduced by hashing the keys, but rather by properly selecting the DHT topology, as discussed in Subsection III-B.

V. Conclusion

Attacks on the Internet mean constantly growing problem for network administrators. Sophisticated attacks have evidence spread across multiple hosts and subnetworks. To detect these attacks promptly and correctly, data must be aggregated and analyzed automatically. In this article, the novel *Komondor* intrusion detection system is presented, which enables current attack correlation methods to be upgraded to work in a distributed environment. This is achieved by inserting a middle layer into the intrusion detection data path, which utilizes the Kademlia DHT overlay. As it is possible to optimize the data storage traffic to $O(1)$ message per attack event, Kademlia is the most feasible



(a) Number of events in IP address space



(b) Number of messages sent to nodes in NodeID space

Figure 9. Network traffic distribution in the structured overlay

choice of DHT topology for a wide area deployment of an intrusion detection system.

The novelty of the method presented is attaching a key to the detected events, which key is then used to send the events for correlating to several correlation units that are organized as a DHT. This mechanism can be used to reduce network and computational load and increase reliability of the system, while still retaining the advantages of centralized approaches of intrusion detection. By mapping the detected events to nodes in the system, all nodes are assigned the same level of responsibility as well. Our ongoing research is focusing on considering the different computational and network capacity of nodes to prevent those with slow connections or CPUs from being overloaded by intrusion detection data.

ACKNOWLEDGEMENT

The work reported in the paper has been developed in the framework of the project "Talent care and cultivation in the scientific workshops of BME". This project is supported by the grant TÁMOP - 4.2.2.B-10/1–2010-0009.

REFERENCES

[1] Z. Czirkos, M. Rencz, and G. Hosszú, "Improving attack aggregation methods using distributed hash tables," in *ICIMP 2012, The Seventh International Conference on Internet Monitoring and Protection*, 2012, pp. 82–87.

[2] H. Debar, "Intrusion Detection Systems-Introduction to Intrusion Detection and Analysis," *Security and privacy in advanced networking technologies*, p. 161, 2004.

[3] C. Zhou, C. Leckie, and S. Karunasekera, "A Survey of Coordinated Attacks and Collaborative Intrusion Detection," *Computers & Security*, vol. 29, no. 1, pp. 124–140, 2010.

[4] D. Mutz, G. Vigna, and R. Kemmerer, "An Experience Developing an IDS Stimulator for the Black-Box Testing of Network Intrusion Detection Systems," in *In Annual Computer Security Applications Conference, Las Vegas, NV*, 2003, pp. 374–383.

[5] C. Seifert, "Analyzing Malicious SSH Login Attempts," http://www.symantec.com/connect/articles/analyzing-malicious-ssh-login-attempts, Nov. 2010, retrieved: March, 2012.

[6] "Internet Storm Center," http://www.dshield.org/, retrieved: March, 2012.

[7] R. Kemmerer, "NSTAT: A Model-based Real-time Network Intrusion Detection System," *University of California-Santa Barbara Technical Report TRCS97*, vol. 18, 1997.

[8] A. Valdes and K. Skinner, "Probabilistic Alert Correlation," *Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection*, pp. 54–68, October 2001.

[9] C. V. Zhou, S. Karunasekera, and C. Leckie, "A Peer-to-Peer Collaborative Intrusion Detection System," in *Networks, 2005. 13th IEEE International Conference on*, vol. 1.

[10] F. Cuppens and R. Ortalo, "LAMBDA: A language to model a database for detection of attacks," in *Recent advances in intrusion detection*. Springer, 2000, pp. 197–216.

[11] S. Templeton and K. Levitt, "A Requires/provides Model for Computer Attacks," in *Proceedings of the 2000 workshop on New security paradigms*. ACM, 2001, pp. 31–38.

[12] V. Yegneswaran, P. Barford, and S. Jha, "Global Intrusion Detection in the DOMINO Overlay System," in *Proceedings of NDSS*, vol. 2004, 2004.

[13] V. Vlachos and D. Spinellis, "A PRoactive Malware Identification System based on the Computer Hygiene Principles," *Information Management and Computer Security*, vol. 15(4), pp. 295–312, 2007.

[14] R. Janakiraman, M. Waldvogel, and Q. Zhang, "Indra: A Peer-to-peer Approach to Network Intrusion Detection and Prevention," in *Enabling Technologies: Infrastructure for Collaborative Enterprises. WET ICE 2003.* IEEE, 2003, pp. 226–231.

[15] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications," *ACM SIGCOMM Computer Communication Review*, vol. 31, no. 4, pp. 149–160, 2001.

[16] P. Kenyeres, A. Szentgyörgyi, T. Mészáros, and G. Fehér, "BotSpot: Anonymous and Distributed Malware Detection," *Recent Trends in Wireless and Mobile Networks*, pp. 59–70, 2010.

[17] J. S. Kong, P. O. Boykiny, B. A. Rezaei, N. Sarshar, and V. P. Roychowdhury, "Scalable and reliable collaborative spam filters: harnessing the global social email networks," 2005.

[18] B. Zhao, J. Kubiatowicz, and A. Joseph, "Tapestry: An infrastructure for fault-tolerant wide-area location and routing," *Computer*, vol. 74, no. 11-20, p. 46, 2001.

[19] S. Androutsellis-Theotokis and D. Spinellis, "A Survey of Peer-to-peer Content Distribution Technologies," *ACM Computing Surveys (CSUR)*, vol. 36, no. 4, pp. 335–371, 2004.

[20] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the xor metric," *Peer-to-Peer Systems*, pp. 53–65, 2002.

[21] Z. Czirkos, G. Bognár, and G. Hosszú, "Pseudo Reliable Broadcast in the Kademlia P2P System," in *Computer Science and Communication Devices: Proceedings of Int. Conf. EDC 2012, CSA 2012, SPC 2012, ACE 2012.*, 2012.

[22] F. Lau, S. Rubin, M. Smith, and L. Trajkovic, "Distributed denial of service attacks," in *Systems, Man, and Cybernetics, 2000 IEEE International Conference on*, vol. 3. Ieee, 2000, pp. 2275–2280.

[23] "Snort – Open-source Intrusion Detection System," http://www.snort.org/, retrieved: March, 2012.

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, CLOUD COMPUTING, COMPUTATION TOOLS, ENERGY, COLLA, IMMM, INTELLI, SMART, DATA ANALYTICS

issn: 1942-2679

International Journal On Advances in Internet Technology

ICDS, ICIW, CTRQ, UBICOMM, ICSNC, AFIN, INTERNET, AP2PS, EMERGING, MOBILITY, WEB

issn: 1942-2652

International Journal On Advances in Life Sciences

eTELEMED, eKNOW, eL&mL, BIODIV, BIOENVIRONMENT, BIOGREEN, BIOSYSCOM, BIOINFO, BIOTECHNO, SOTICS, GLOBAL HEALTH

issn: 1942-2660

International Journal On Advances in Networks and Services

ICN, ICNS, ICIW, ICWMC, SENSORCOMM, MESH, CENTRIC, MMEDIA, SERVICE COMPUTATION, VEHICULAR, INNOV

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, CLOUD COMPUTING, COMPUTATION TOOLS, IMMM, MOBILITY, VEHICULAR, DATA ANALYTICS

issn: 1942-2628

International Journal On Advances in Systems and Measurements

ICQNM, ICONS, ICIMP, SENSORCOMM, CENICS, VALID, SIMUL, INFOCOMP

issn: 1942-261x

International Journal On Advances in Telecommunications

AICT, ICDT, ICWMC, ICSNC, CTRQ, SPACOMM, MMEDIA, COCORA, PESARO, INNOV

issn: 1942-2601