UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

Taciane Martimiano

DISTRIBUTED ATTACKER: AN ATTACKER TYPE PROPOSAL FOR SECURITY CEREMONIES

Florianópolis

Taciane Martimiano

DISTRIBUTED ATTACKER: AN ATTACKER TYPE PROPOSAL FOR SECURITY CEREMONIES

Dissertação de Mestrado submetida à Ciência da Computação para a obtenção do Grau de Mestre em Ciência da Computação.
Orientador: Prof. Dr. Jean Everson Martina (UFSC)
Coorientador: Prof. Dr. Ricardo Alexandre Reinaldo de Moraes (UFSC)

Florianópolis

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Martimiano, Taciane
Distributed Attacker: An attacker type proposal for security ceremonies / Taciane Martimiano ; orientador, Jean Everson Martina ; coorientador, Ricardo Alexandre Reinaldo de Moraes. - Florianópolis, SC, 2017.
89 p.

Dissertação (mestrado) - Universidade Federal de Santa Catarina, Centro Tecnológico. Programa de Pós-Graduação em Ciência da Computação.

Inclui referências

1. Ciência da Computação. 2. Security Ceremonies. 3. Attacker Types. 4. Threat Models. 5. Formal Analysis. I. Martina, Jean Everson. II. Reinaldo de Moraes, Ricardo Alexandre. III. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Ciência da Computação. IV. Título.

Taciane Martimiano

DISTRIBUTED ATTACKER: AN ATTACKER TYPE PROPOSAL FOR SECURITY CEREMONIES

Esta dissertação foi julgada adequada para obtenção do título de mestre e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

Florianópolis, 22 de Fevereiro de 2017.

_____
Profª. Carina Friedrich Dorneles, Drª.
Coordenadora do Programa

_____
Prof. Ricardo Alexandre Reinaldo de Moraes, Dr.
Universidade Federal de Santa Catarina
Coorientador

Banca Examinadora:

_____
Prof. Jean Everson Martina, Dr.
Universidade Federal de Santa Catarina
Orientador

_____
Prof. Giampaolo Bella, Dr.
Università di Catania (Videoconferência)

_____
Profª. Carla Merkle Westphall, Drª.
Universidade Federal de Santa Catarina

_____
Profª. Jerusa Marchi, Drª.
Universidade Federal de Santa Catarina

To my parents, for all the love, help, faith and patience.

ACKNOWLEDGEMENTS

I am very thankful to my supervisor, Jean Martina, for all the guidance provided since my graduation until now. I extend my thanks to the remaining professors and colleagues of the LabSEC/UFSC who helped me make this work possible.

I also would like to express my most sincere gratitude to my parents Almir and Loreci Martimiano, and boyfriend Renan Netto. They have always been an endless source of support, love and encouragement in my life. Without them I would certainly have not reached this far.

ABSTRACT

Security ceremonies are extensions of security protocols, including all that is out-of-bounds for protocols. Nowadays we lack a base description language and a detailed threat model for security ceremonies in order to be able to use symbolic evaluation methods and verify claims embedded in ceremonies. Our goal is to contribute with a syntax and detailed threat model for ceremonies description in order to establish our proposal for a new attacker type named Distributed Attacker (DA in brief). Moreover, we also developed a strategy for symbolic evaluation of our attacker model using First-Order Logic (FOL) and an automatic theorem prover. Lastly, we present scenarios formally analysed with our methodology, including cases we could not have with standard Dolev-Yao or Multi-Attacker models. For instance, our most interesting scenario is when several attackers gather only pieces of an user's credentials and, by putting together their knowledge, collude to attack this user's email account.

Keywords: Security Ceremonies, Socio-Technical Security, Threat Models, Attacker Types, Formal Analysis

RESUMO

Protocolos de segurança são subconjuntos das chamadas cerimônias de segurança. Atualmente não se tem uma linguagem de descrição e um modelo de ameaça detalhado para cerimônias de segurança, necessários para o uso de métodos de avaliação simbólica e verificação de suposições presentes em cerimônias. O objetivo desta dissertação é contribuir com uma sintaxe para descrição de mensagens de cerimônias e apropriado modelo de ameaça a fim de estabelecer a proposta para um novo tipo de atacante (nomeado Atacante Distribuído). Adicionalmente, uma estratégia para execução de avaliação simbólica também foi desenvolvida, utilizando lógica de primeira ordem e um provador de teoremas automático. Por fim, cenários formalmente analisados com o modelo de atacante proposto são exibidos, incluindo casos não passíveis de serem simulados com modelos padrão como Dolev-Yao ou Multi-Attacker. Por exemplo, o caso mais interessante é o que apresenta vários atacantes com conhecimento apenas de partes das credenciais de um usuário, mas que ao colaborar entre si conseguem atacar a conta de email desse usuário.

Palavras-chave: Cerimônias de segurança, Segurança sócio-técnica, Modelos de ameaça, Tipos de atacante, Análise formal

RESUMO ESTENDIDO

INTRODUÇÃO

Um dos problemas da área de protocolos de segurança é que mesmo protocolos massivamente testados ainda falham quando usados pelos usuários finais (BELLA; LONGO; PAULSON, 2003). A maioria dessas falhas está relacionada a suposições feitas pelo *designer* que acabam por não ocorrer da forma prevista após a implementação do protocolo. Tal problema leva a uma nova abordagem que considere o usuário na hora de criar as premissas necessárias: cerimônias de segurança.

As cerimônias de segurança - vistas como extensões dos protocolos - incluem as ações humanas de forma explícita, como por exemplo através da modelagem das interações entre humanos, e entre humanos e máquinas (ELLISON, 2007). Mesmo assim, algumas hipóteses ainda são necessárias. Aqui pode-se citar o conhecimento inicial de cada agente humano presente na cerimônia.

Atualmente, a falta de uma linguagem de descrição e de um modelo de ameaça padronizados para cerimônias de segurança - capazes de expressar as sutilezas do comportamento humano - dificulta o desenvolvimento de uma avaliação simbólica para cerimônias.

É importante providenciar uma sintaxe de descrição para cerimônias e uma especificação minuciosa dos modelos de ameaça padrões para ser possível comparar diferentes cenários de uma mesma cerimônia, e determinar qual é a mais segura.

Outra dificuldade encontrada é a falta de modelos automáticos que chequem se uma cerimônia realmente alcança os objetivos esperados, devido à complexidade da análise de cerimônias de segurança. Com uma ferramenta formal de análise, o teste e mecanização da comparação de cerimônias passa a ser viável. Adicionalmente, tal análise automatizada facilita o entendimento do modelo de ameaça ao qual os agentes do sistema estão sujeitos.

OBJETIVOS

O objetivo geral deste trabalho é estabelecer uma avaliação simbólica para cerimônias de segurança sociotécnicas. Primeiramente, é elaborada uma linguagem de descrição para representar os agentes, canais e mensagens das cerimônias de segurança. A seguir, é proposto um novo tipo de atacante, Atacante Distribuído (sigla DA em inglês), através do qual foram modelados alguns casos de estudo ao longo da dissertação. Por fim, com a automatização do modelo DA, foram analisadas formalmente as cerimônias dos casos de estudo por meio do provador de teoremas SPASS.

Objetivos específicos

- Descrever cerimônias de segurança com a sintaxe de notação proposta;
- Usar o *framework* de Bella et al. (2014) e relacionar cada camada das cerimônias com um modelo de ameaça (atacantes e habilidades), como sugerido por Carlos et al. (2013);
- Avaliar o comportamento do modelo de Atacante Distribuído proposto nesta dissertação;
- Mecanizar e formalizar os canais de comunicação, mensagens e agentes das cerimônias de segurança.

METODOLOGIA

O primeiro passo é entender as cinco camadas da metodologia Concertina de Bella et al. (2014). A primeira delas, camada 1, concentra-se na troca de mensagens pela rede (Internet) executada por dois usuários (agentes comunicantes). A camada 2 representa o protocolo (executado em nome da interface) que encaminha e recebe dados da rede. A terceira camada estabelece a conexão entre máquina e ser humano, caracterizada pelo usuário interagindo com a interface de seu computador e/ou celular. Esta camada é de crucial importância para que a cerimônia alcance os objetivos estimados. A camada 4 já se preocupa com o humor do usuário ao se relacionar com a interface, enquanto a camada 5 reflete a influência da sociedade sobre o comportamento do usuário. À ciência da computação cabe se concentrar nas camadas de 1 a 3, envolvendo

máquina e humano. As camadas 4 e 5 são bem voltadas para a área social e, portanto, não são abordadas no presente trabalho.

O segundo passo é saber como aplicar o modelo adaptativo de Carlos et al. (2013) para os modelos de ameaça. Segundo esse modelo, é plausível ter atacantes com um subconjunto de todas as capacidades de um atacante Dolev-Yao (DOLEV; YAO, 1983). Tal conjunto é originalmente composto pelas seguintes capacidades: *Eavesdrop, Initiate, Atomic Break Down, Block, Crypto, Fabricate, Spoof, Re-order, Modifying* e *Replaying*. Basicamente, o atacante é capaz de aprender o conteúdo das mensagens trocadas pela rede, bloquear e criar mensagens, iniciar novas comunicações com outros agentes do sistema, fazer uso de funções criptográficas de conhecimento público, modificar e replicar mensagens.

A contribuição central desta dissertação é o modelo de Atacante Distribuído (DA), que se dá juntamente com contribuições menores tais como: descrição apropriada de cerimônias, uso das camadas da metodologia Concertina aplicadas ao modelo adaptativo, e a formalização do modelo DA no provador de teoremas SPASS.

A grande novidade da proposta para Atacante Distribuído é a flexibilidade que os atacantes têm de poder compartilhar conhecimento (caso assim desejem). Os atacantes podem fazer associações entre si de maneira em que cada atacante obtém apenas parte das informações totais necessárias, e juntos são capazes de realizar o ataque com muito menos esforço individual.

Por fim, apresentamos diversos cenários que ilustram nosso modelo DA e os analisamos formalmente - por meio da tradução e verificação da nossa metodologia através do SPASS.

RESULTADOS

A maior contribuição deste trabalho é o modelo de atacantes distribuídos (DA). O DA é um modelo poderoso ao permitir que os atacantes possuam conhecimento sobre camadas que eles não estão atacando ativamente, por meio das associações entre os atacantes. Por meio dessa cooperação entre os atacantes, o modelo DA flexibiliza ataques em cerimônias que usufruam de *2-step verification* (foco da dissertação). Diferentemente de modelos como *Doley-Yao* (DY) e *Multi-Attacker* (MA) [ver (ARSAC et al., 2010)], os

atacantes DA são livres para atuarem em várias camadas ao mesmo tempo, inclusive com capacidades diferentes para cada camada.

Como mostrado os estudos de caso, o uso do celular com o código do "segundo passo de autenticação", além da senha habitual do usuário, contribui para a segurança do usuário ao envolver mais dispositivos na cerimônia. Contudo, o modelo DA permite que dois ou mais atacantes reúnam parte das informações (nesse caso, a senha e o código) cada um e juntos ataquem o sistema. Capturar a senha ou o código separadamente e atacar em conjunto é mais fácil do que apenas um atacante agindo sozinho ter de capturar ambos senha e código do celular.

É relevante ressaltar a origem dos modelos empregados neste trabalho para entender melhor suas filosofias. O modelo DY data da época da guerra fria, focando na propriedade do sigilo. Já o modelo MA se origina da era da Internet e sua mudança na postura dos atacantes. Considera que qualquer agente do sistema pode se tornar um atacante DY (permitindo que hajam vários atacantes simultaneamente).

Hoje em dia, ao usarmos celular ou cartão de crédito deixamos "traços" que podem ser rastreados, como alerta Snowden em entrevista após liberar informações sigilosas da NSA (SNOWDEN…, 2014). Motivado por esse contexto, o modelo DA considera atacantes com mais capacidades do que em modelos clássicos, adaptando-se à era de espionagem em que vivemos.

Por último, foi apresentada a formalização do modelo DA em SPASS, juntamente com a análise dos estudos de caso com autenticação em 2 passos (a qual não é suportada pelos demais modelos inclusos). Os estudos de caso servem para mostrar a viabilidade do nosso modelo, por meio de diversos ataques diferentes, e para ilustrar que métodos formais podem auxiliar na compreensão de problemas práticos em cerimônias de segurança.

Palavras-chave: Cerimônias de segurança, Segurança sociotécnica, Modelos de ameaça, Tipos de atacante, Análise formal.

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS LIST

SYMBOLS LIST

CONTENTS

1 INTRODUCTION

Nowadays, one of the problems of the security protocol community is that even massively deployed protocols which are deeply verified (proved secure by automated protocol analysis tools) still fail (BELLA; LONGO; PAULSON, 2003). Most of these failures are related to assumptions taken by the protocol designer that are not fulfilled when the protocol is implemented (for instance, an unexpected user behaviour). These problems call for a new approach that better describe assumptions and take the user into account. We see then the appearance of the socio-technical area of security ceremonies (MARTINA; CARLOS, 2010).

We define security ceremonies as a sequence of interactions among entities, designed to achieve a given security goal (such as entity authentication, key distribution, secrecy, etc). The distinction between ceremonies and protocols, as described by Ellison (2007), is that ceremonies are a super-set of security protocols. Ceremonies include as explicit interactions all assumptions considered out-of-scope in protocols, being good examples the interactions between humans and devices, and between humans and humans. Undoubtedly, ceremonies still need assumptions, such as the initial knowledge of human peers. Nevertheless, these assumptions tend to be more precisely described, fine grained and realistic (CARLOS et al., 2013).

As in ceremonies we have to handle the daunting challenge of adding human nodes into the specification, we rely on extra communication channels: human-device and human-human channels (CARLOS et al., 2013). For the present work we apply the "Security Ceremony Concertina Traversal" methodology to understand these channels as layers (BELLA et al., 2014). The Concertina layers are broader in view and help us mitigate cause and place of attacks during the ceremony execution.

The Ceremony Concertina consists of a ceremony model that links technology to society through a number of layers, ranging from computer processes to user personas. The first three layers represent the information pathway: the interaction between network and operating system, then between operating system and process running on behalf of the interface and, lastly, between interface and user. At this point, the information reaches the user expressing a given persona and the last layer stands for the influence of society over this user's demeanour.

Bella et al. (2014) argue that a ceremony can be layered and

analysed in the specific sections of its description, enabling the comprehension of important aspects. Moreover, they say that we need relevant descriptions to verify whether we can achieve our goals. However, they do not assert a precise description syntax for security ceremonies. They also do not establish a threat model where one can conduct formal analysis. Thus, we find our motivation for defining the threat models present in our case study ceremony scenarios and analyse them using symbolic evaluation techniques.

It is useful to clarify the distinction between ceremony and scenario. We may have several ceremonies to solve a given problem. However, when we fix one of those ceremonies we start talking about scenarios for this fixed ceremony. Ceremony scenarios are different versions of the same ceremony with a slight change in the threat model of one of the ceremony messages.

Now moving to the attacker types considered for this dissertation, we have the Dolev-Yao attacker (DY in brief) [see (DOLEV; YAO, 1983)], the Multi-Attacker (MA) [see (ARSAC et al., 2010)] and our proposal for a Distributed attacker (DA).

The idea of representing an active attacker in a security protocol comes from Needham and Schroeder (NEEDHAM; SCHROEDER, ACM Press, 1978). Later, Dolev and Yao (DOLEV; YAO, 1983) formalised Needham-Schroeder attacker and described in more details the capabilities of the attacker. The capabilities of the attacker are in fact what validate or not Needham-Schroeder claims, as shown by Lowe (1996).

A Dolev-Yao attacker, in turn, controls the channel - being capable of altering, copying, replaying and creating messages (DOLEV; YAO, 1983). In this context, the only restraint is that the attacker cannot perform cryptanalysis and guess random numbers. A protocol considered secure against such an attacker is secure against less powerful ones in the point of view of security protocols analysis and verification.

As a variant of the DY attacker, we have the Multi-Attacker model. The MA model allows each participant agent to be a potential attacker by behaving as a DY attacker. We may have several DY attackers in this model, where neither collude nor share knowledge with each other (ARSAC et al., 2010).

Our proposal for a more flexible attacker type (DA) is explained later on in full detail - as it is our main contribution.

Regarding threat modelling, we notice that perceiving the threshold between a realistic and secure ceremony, and an overly protected one is also challenging. If we overestimate the attacker capabilities in a ceremony we will probably end up designing complex and difficult

ceremonies. However, if we underestimate the attacker, we might have a flawed ceremony (CARLOS et al., 2013). To approach such a task, Carlos et al. (2013) proposed an adaptive threat model for security ceremonies. In their work, the ceremony may start with a powerful attacker such as a DY. Then, the designer can adjust the set of attacker capabilities to make the attacker more realistic. The full set of DY capabilities (DY_{cap}) includes[1]: *Eavesdrop*, *Initiate*, *Atomic Break Down*, *Block*, *Crypto*, *Fabricate*, *Spoof*, *Re-order*, *Modifying*, and *Replaying* (CARLOS et al., 2013).

Carlos et al. (2013) establish that an over-powerful attacker may be unrealistic in a human-human ceremony setting. They say that even though the Dolev-Yao threat model can represent the most powerful attacker possible in a ceremony, such an attacker is unrealistic in certain scenarios, especially those related to human peers. Carlos et al. (2013) believe that having an attacker which can overcome the laws of physics and interfere with human speech or direct human action is usually above reason for most cases. It other words, adjusting the set of capabilities is generally used for those layers involving humans. This way, their adaptive threat model leaves the well known Dolev-Yao attacker as the standard setting for device-device channels (once a DY suits the network layer perfectly).

Modelling and connecting the Ceremony Concertina layers to the classic DY attacker and to the Multi-Attacker is a topic not discussed yet. We will combine these attacker types (MA, DY and our DA) and adaptively set the threat model, so that we can perform tests to compare (among different sets) which ceremony is the most secure to implement.

1.1 JUSTIFICATION

With ceremonies we are able to evaluate human peers along with other peers of the system (CARLOS et al., 2013). Today, however, two main points hinder the symbolic evaluation of security ceremonies: lack of a standard notation for security ceremonies, which encompass the subtleties of human peers, and lack of a standard threat model.

The lack of a standard description language makes it difficult to envisage the creation of a symbolic evaluation tool for ceremonies. To be able to analyse security ceremonies, we need a language capable of collecting the description of human peers actions and interaction within

[1] We define each of these DY capabilities on our background chapter.

the system, as well as the threat models that appear in each scenario. Only with that we will be able to sketch a symbolic evaluation strategy for ceremonies analysis. Claims will only be verifiable within a specific scenario (primarily described by the threat model within each ceremony layer). The lack of a standardised threat model makes any reasonable claim imprecise.

Another difficulty we face is the lack of automated models (due to the complexity of security ceremonies design) for checking that indeed a given ceremony achieves its claimed security goals. With a formal analysis tool, we can test and compare ceremonies, being able to distinguish between the secure from the flawed ones. Furthermore, such analysis can also accomplish a better understanding of the threat models the peers of the system are subject to.

1.2 MOTIVATION

We see that currently employed attacker types (as DY and MA) do not allow information sharing. Bearing in mind that "Every time you pick up the phone, dial a number, write an email, make a purchase, travel on the bus carrying a cell phone, swipe a card somewhere, you leave a trace [...]" (SNOWDEN..., 2014), we find ourselves in a new reality in terms of ceremony threat models. It has even more impact when thought from the point of view of the user. Users act naively in many cases or avoid security measures (attempting to bypass the system) by not knowing which threat model they are subject to.

In fact, powerful agencies have all kinds of information easily on hands and by any means necessary. In here we must cite The Five Eyes alliance and its mass surveillance - in part evidenced by documents released by Edward Snowden [2]. Within this context, we are considering attackers with clearly more abilities than the standard models assume. Therefore, we are proposing a more powerful attacker type which is suitable for the espionage era we currently live in: our Distributed Attacker.

We provide the mechanisation of the case study scenarios we analyse throughout this dissertation as a means of validation of our methodology. This is important because we need formalisation to compare ceremony scenarios. However, the relevance of formally analysing a ceremony goes beyond the comparison among scenarios. With formalisation, we are able to perform fast and automated tests to assess

[2] More info available at: https://www.privacyinternational.org/node/51

scenarios for further classification into those that achieve the ceremony goal and those that do not. As a result, we are able to choose the best ceremony scenario from the tested set.

1.3 OBJECTIVES

Our goal is to contribute with a notation and detailed threat model for ceremony description in order to establish our proposal for a new attacker type named Distributed Attacker (DA in brief).

1.3.1 Specific Objectives

- Describe ceremonies with our proposed syntax notation;
- Use the framework of Bella et al. (2014) and correlate each layer with a threat model, as described by Carlos et al. (2013);
- Evaluate the behaviour of our proposed Distributed Attacker;
- Mechanise and formalise communication channels, messages and peers of security ceremonies, showing the formalisation steps applied to our ceremony scenarios.

1.4 PUBLICATIONS

We would like to stress that most part of the contents of this dissertation are derived from the publications listed below. Additionally, we present part of a work we submitted to the Formal Aspects of Computing - CryptoForma Special Issue which is still being evaluated for publication.

- Ceremony Analysis Meets Verifiable Voting: Individual Verifiability in Helios (Securware, 2015) - Qualis B3;
- Threat Modelling Service Security as a Security Ceremony (ARES, 2016) - Qualis B1.

1.5 STRUCTURE OF THIS WORK

Chapter 2 brings ceremony threat modelling background. Following, chapter 3 presents our first accepted paper (as listed in 1.4). In this chapter we show the beginning of our analysis, with informal and somewhat primitive proofs. Chapter 4 has part of the contents of our second accepted paper (also listed in 1.4), including our DA proposal and our case study scenarios which exemplify our model. The formalisation of the discussed scenarios is given in chapter 5, along with our mechanisation strategy. Finally, chapter 6 concludes this dissertation.

2 BACKGROUND

This chapter presents all needed background on security ceremonies, threat models and ceremony analysis to ease the comprehension of the following materials.

2.1 SECURITY CEREMONIES

Ceremonies extend protocols by including human peers to the system specification. In protocols, all human actions are modelled as assumptions and, when the protocol is implemented, these assumptions may result in user interactions that are unrealistic or not well-specified (ELLISON, 2007).

In ceremonies, additional channels are available to model human interaction: human-human (HH) and human-device (HD) channels. We have the device-device (DD) channel for the subset of protocols itself. Taking these channels into account, we analyse ceremonies using the adaptive threat model (CARLOS et al., 2013)[1].

The distinction between ceremonies and protocols, as described before, is that ceremonies are a super-set of security protocols. The interactions between humans and devices (interfaces) and between humans and humans (human-human communication and transfers of physical objects that carry data) are good examples of out-of-scope situations for protocols, all of which are easily instantiated with the use of security ceremonies (ELLISON, 2007).

Ceremonies, however, still need some assumptions as for example the initial knowledge of human peers. Moreover, the gains from designing and analysing security ceremonies are linked to the quality and accuracy with which the involved components are described (CARLOS et al., 2013).

Ellison (2007) says that "[...] A secure ceremony is secure against both normal attacks and social engineering. However, some secure protocols imply ceremonies that cannot be made secure. [...] The problem comes with modelling a human node. Like a computer protocol node, the human node has state and a state machine. It receives and emits messages which cannot be programed as done with device nodes. We must instead learn the human state machine empirically, by observing actual human behaviour.".

[1] We use HD and DD channels in our first publication, in chapter 3.

In ceremonies, differently than in protocols, human behaviour is neither considered predictable nor deterministic. We now move to the specifics of the model we employed to address human and device nodes for the ceremony analysis carried in our second publication - presented in chapter 4.

In this point of our work we move from Carlos et al. (2013) channels (HH, HD and DD) to Bella et al. (2014) layers, as the latter are more expressive in terms of the human role played in ceremonies, and define in a more detailed manner the information pathway.

2.2 CEREMONY CONCERTINA LAYERS

The Ceremony Concertina approach establishes security and privacy in the presence of humans. It links technology to society through the following five layers (BELLA et al., 2014):

- Layer 1 (Informational) concerns the security protocol running between computer processes. It aims to secure the messages exchanged between two users communicating over a potentially insecure network.

- Layer 2 (Operating System) is an intermediate level expressing the communication between the process that executes the security protocol on behalf of the user and the process that runs the interface presented to that user;

- Layer 3 (Human-Computer Interaction) stands for the socio-technical protocol whereby an user interacts with a graphical interface displayed in his communicating device. This layer is crucial for the protocol to reach its end and achieve its goals;

- Layer 4 (Personal) is related to the user expression of a given persona while interacting to the interface. The user's persona may differ accordingly to which technology or service the user needs to deal with for a given task and to his own mood during the execution of that task;

- Layer 5 (Communal) shows the society influence over the user demeanour.

From all five layers, protocols just focus in the first one as they only handle network traffic. Ceremonies allow us to study the three

first layers, once they include both human and device peers. Thus, we cover up to the third layer in our work (standing in computer science research area). As stated in the work of Bella et al. (2014), layers 4 and 5 are strongly related to social science (which is out of our scope) as such layers deal with the non-deterministic nature of the human being.

Layer 4 does not appear in our modelling because it involves more than just the simple interaction between user and device. It concerns the mindset of the user while interacting with the system. We envisage Pirandellian masks as one interesting strategy to address this layer. Pirandellian masks are theatrical tool used by Luigi Pirandello, where characters search for the actors to be performed (PIRANDELLO, 1922). With that we ought to be able to design ceremonies that protect for common user behaviour by having the meaningful masks people wear when executing ceremonies.

Lastly, layer 5 represents the influence of the society over the user's attitudes, being clearly out of our reach as computer scientists (alone). This way, both layers 4 and 5 remain for future work, as we believe that such tremendous challenge (as modelling human actions) may be manageable by joined work with social science experts.

2.3 ATTACKER TYPES

A ceremony can be designed and analysed with variants of the mature methods already in use for a network protocol (ELLISON, 2007). In this section, we enlist the attacker types we used throughout our work. As such, we start with the standard attacker model for security protocols: a Dolev-Yao (DY) attacker (DOLEV; YAO, 1983).

A Dolev-Yao attacker controls the channel, being capable of altering, copying, replaying and creating messages. However, the attacker is not allowed to perform cryptanalysis or guess random numbers. In the point of view of security protocol analysis and verification, a protocol considered secure against this attacker is secure against less powerful ones.

Along with the DY model, we use one of its variants: the Multi-Attacker model (MA) (ARSAC et al., 2010). Any peer is a potential attacker, behaving as a Dolev-Yao and never revealing his long-term secrets to others. In this sense, each attacker neither colludes nor shares knowledge with any other attacker. For this dissertation, we adapted the MA model to also be able to act on layer 2 (besides layer 1, which both MA and DY attacker types are able to attack).

2.4 THREAT MODELS

The adaptive threat model proposed by Carlos et al. (2013) uses the set of capabilities of a Dolev-Yao attacker, by dynamically adding or removing capabilities from the whole set. By doing so, we are able to avoid cases where overly stringent requirements are placed on users. While such requirements are motivated by security concerns, they are likely to negatively impact usability.

The Dolev-Yao full set of capabilities (DY_{cap}) is defined as follows (CARLOS et al., 2013):

- *Eavesdrop* (E) – The attacker learns the contents of any message sent through the communication channel, just by listening to the channel.
- *Initiate* (I) – The attacker can use his own knowledge to initiate a new communication with another peer of the system.
- *Atomic Break Down* (A) – The attacker can break any message in its sub-components and learn each of the sub-components contents individually.
- *Block* (B) – The attacker is able to block a message, preventing its intended receiver from learning the message.
- *Crypto* (C) – Capability where the attacker performs cryptographic operations, once he has the needed knowledge to do so. For instance, if a given attacker knows a specific cryptographic key he is able to decrypt any message encrypted with such key.
- *Fabricate* (F) – Represents the use of publicly known functions to fabricate new messages. For this purpose, the attacker may use his initial knowledge or the contents he learnt through any other capability.
- *Spoof* (S) – Attacker's capability of sending a message to an agent pretending to be another agent. It is distinct from Initiate by not allowing the attacker to be an internal agent in the execution of the ceremony.
- *Re-order* (O) – The attacker is able to re-order the messages, so the receiver will learn the contents in different order than originally sent.

- *Modifying* (M) – This capability can be seen as the combination of Block and Initiate capabilities. The receiver does not learn the intended contents, learning a modified version (sent by the attacker) instead.

- *Replaying* (R) – This capability can be seen as the combination of Eavesdrop and Initiate capabilities.

The analysis process begins with the establishment of which channels are present at the ceremony of interest. This consists of, first, a list of all human and device nodes. Secondly, we identify which pairs of these nodes exchange messages. Then we associate each pair of nodes to the respective type of channel representing their communication (i.e., HH, HD or DD). This way we can proceed to the threat modelling of the ceremony being analysed, where we vary attacker types and capability sets to assess the impact on security.

The attacker's goal is to learn the contents being exchanged among nodes. The DY attacker type defines capabilities that allow the attacker to achieve his goal. Thus, we observe how the attacker may use his capabilities in order to obtain a realistic threat model to each channel - following the adaptive threat model. Understanding the threat model the user is subject to when interacting in a ceremony will prevent him from being overloaded with unrealistic scenarios, and guarantee that important security properties hold (CARLOS et al., 2013).

2.5 CEREMONY DESIGN AND VERIFICATION

Since ceremonies are a super-set of protocols and protocol design and verification has been a well researched topic, we approach ceremony formalisation by borrowing some ideas from the protocol research field. During the initial phase of protocol verification in the 1980's and mid-1990's, protocol verification was carried out informally. This is also the case for security ceremonies.

Informal verification as a first step in ceremony formalisation is important because it helps us understand the semantics behind the messages. Due to its informality, it is often simpler to find minor flaws. No complicated or extensive reasoning is usually involved in this stage, making the outcome easy to comprehend.

From mid-1990's, we started seeing an increased interest in the usage of formal tools to assist the verification of security protocol models (MEADOWS, 1996). We can cite efforts such as:

- Belief Logic (BURROWS; ABADI; NEEDHAM, 1990), which first represented formally the beliefs that peers running the protocol can derive during the execution;
- Paulson's Inductive Method, a formalism that allows us to prove the existence of security properties via structural induction over an unbounded protocol model (PAULSON, 1998; BELLA, 2007). It is assisted by a very powerful tool called Isabelle, a theorem prover for higher-order logic (HOL) (NIPKOW; PAULSON; WENZEL, 2002).

Formal verification techniques have been aimed to ensure the correct communication among devices in security protocols. We now face a different new challenge of including everything else into the mix and conducting symbolic evaluation of security ceremonies as we do with protocols. Some works have already tried to address ceremony design and verification. Carlos et al. (2012) pursued these formalisation ideas using Isabelle/HOL, being able to capture some of the subtleties of ceremony design and verification. However, they seem to have abandoned the idea due to the lack of a symbolic evaluation threat model and to the difficulties of dealing with HOL.

Martina et al. (2015) further expanded Carlos et al. (2013) by demonstrating how to conduct symbolic evaluation with the adaptive threat model, and using first-order logic (FOL) and a theorem prover. We based our mechanisation in Martina et al. work, as FOL is easily molded and we were able to test the attacker types we employed with different sets of capabilities.

We now move to chapter 3 which brings the contents of our first accepted and published paper. It has the Helios voting system as example ceremony. The purpose of the next chapter is to demonstrate our very first steps regarding ceremony analysis with informal proofs (basically just mathematical reasoning). Voting systems are not our area of expertise, so Helios serves us simply as our case study.

3 INFORMAL CEREMONY ANALYSIS FOR HELIOS VOTING SYSTEM

We start this chapter by describing how Helios voting system works, and then we present our informal analysis over two proposals for interface usability improvement on Helios.

In order to engender voter trust in electronic voting, cryptographic voting systems that offer verifiability while maintaining vote secrecy have been proposed, and continue to gain ground.

Helios, open-source verifiable Internet-based voting system, stands out for its continued use, primarily in academic contexts, for example, in 2009, to elect the university president at the Université Catholique de Louvain (ADIDA, 2008; ADIDA et al., 2009). It was also used in the 2013 Princeton University undergraduate student government elections, and to elect the Board of Directors of the International Association for Cryptologic Research (IACR) (PRINCETON, 2013; IACR, 2013).

Helios assumes that voters will verify their votes to ensure vote integrity (ADIDA, 2008). While it is not known whether this assumption is true for the elections where Helios has been used, findings from expert reviews and user studies suggest that this is not likely to be the case, due to the cumbersome nature of the verification process (KARAYUMAK et al., 2011a, 2011b).

Usability improvements to the Helios voting interfaces, with a specific focus on the verification aspect, have been proposed to ensure that this assumption can be met. Voters can now verify their vote (through the support of independent trusted institutes) in two forms: either in the institutes' web page, or via download and installation of an app on the voters' smartphone (maintained by those same institutes). We analyse the security of such improvements, a recommended practice for usable security (SASSE; FLECHAIS, 2005). The focus is on verifiability and integrity.

Several extensions to the Helios voting protocol have been proposed, focusing on providing everlasting privacy (DEMIREL; GRAAF; ARAúJO, 2012), privacy and correctness (CORTIER et al., 2013), and preventing attacks against privacy (CORTIER; SMYTH, 2011). Tsoukalas et al. (2013) proposed Zeus, which is a verifiable voting and counting system, developed based on the Helios version in (ADIDA, 2008). The authors propose that the voter enters an audit code to indicate that a submitted vote should be verified. However, no analysis of the security implications of these modifications is provided.

Cortier et al. (2013) came up with a variant of Helios that prevents ballot box stuffing. Comparatively, the research we report in this chapter analyses the security of two proposals made to improve the usability of verification in Helios, in order to ensure that voters can indeed verify that their ballots are cast and counted in the final tally, as intended[1].

We apply ceremony analysis [see Ellison (2007)] using the adaptive threat model provided by Carlos et al. (2013), which is appropriate to analyse the human-device and human-human channels. Following standard practice, a Dolev-Yao [see (DOLEV; YAO, 1983)] adversary is assumed on the device-device channels.

Our findings show that:

1. No threats to secrecy are present when the voter uses the smartphone app to verify;

2. Reputation attacks might be carried out to undermine the institutes participating in elections. In such cases, voter education on necessary steps is required;

3. Semi-formal verification can be applied to election ceremonies.

We discuss the implications of these findings for voting and verification in Helios.

3.1 HELIOS VOTING PROTOCOL

We focus here only on those aspects that are relevant to verifiability and that are necessary to understand the proposals presented in later sections.

To vote using Helios, the voter downloads the Helios ballot preparation system (BPS) onto his web browser (ADIDA, 2008). He indicates his choice on the ballot. The BPS encrypts these choices (i.e. the vote) and commits to this encryption by displaying a hash value, which we refer to as a check-code. The voter should record the check-code displayed if he plans to verify it. At this point, the voter makes a choice to either submit this encrypted vote to the public web bulletin board or to challenge the voting system, verifying whether the vote has been correctly encrypted.

[1] These proposals to improve Helios usability come from the PhD research of one of the authors of our first paper (reported here in this chapter): Maina M. Olembo.

If the voter decides to verify his vote, he interacts with the Helios ballot verifier system (BVS). The BPS displays the choice and the randomness used for encryption. The voter selects and copies this information to the voting device clipboard and pastes it into the BVS, which BPS opens in a new web browser window. The BVS encrypts the corresponding choice and generates the hash value of this encryption. This hash value is displayed together with the choice contained in the vote received earlier.

In order to complete the verification process, the voter needs to confirm that the check-code displayed by the BVS matches the one displayed earlier by the BPS. Additionally, he needs to confirm that the vote is correct. If both these conditions are met, the voter is assured that the system correctly encrypted the vote in this instance. He can repeat the verification process several times until he is satisfied that the system is behaving correctly.

Once votes are verified they can no longer be submitted to the public web bulletin board as the voter could easily prove how he voted using the revealed randomness. Thus, new randomness is required. As the BVS learns the content of the encryption, the use of test votes that differ from the final vote has been recommended, to avoid the BVS computing intermediate results (KARAYUMAK et al., 2011b).

If the voter chooses to submit his vote to the public web bulletin board, he is prompted to authenticate himself, and his encrypted vote is then posted on the public web bulletin board together with the check-code. To verify that the encrypted vote is correctly stored on the voting server or public web bulletin board, the voter needs to confirm that the check-code appears on the public web bulletin board next to his name, or some pseudonym (ADIDA, 2008; ADIDA et al., 2009). It is only necessary to do this once as the Helios threat model assumes that auditors continuously observe the bulletin board preventing malicious behaviour.

3.1.1 Assumptions

We present assumptions of the original Helios system, as well as our assumptions regarding the entities involved in the ceremony, and the ceremony itself.

The *attacker* lies only on the channel as is the usual Dolev-Yao assumption. Further, he cannot control more than one device-device channel.

Any participating *verification institute* is trustworthy as any malicious behaviour would lead to a loss of reputation. We are not considering denial of service attacks.

The *voter* is an honest peer in the ceremony as a dishonest voter can easily prevent a ceremony from concluding correctly. We also do not consider coercion. Thus, the cases where the attacker is the voter are not included.

Finally, the *ceremony* is assumed to have only one entry and one exit point, so the voter is expected to follow all the steps provided in the ceremony he is executing.

3.1.2 Security criteria

As the adaptive threat model will be applied to the electronic voting context, we define necessary security criteria adapted from Neumann, Budurushi e Volkamer (2013).

A number of security properties are considered relevant in the electronic voting context. In this chapter, we focus on verifiability and integrity, likely the most important properties for elections conducted over a remote channel.

Integrity: The sum of all participating voters' submitted votes (votes submitted to and stored on the voting server or the public web bulletin board) matches the declared election result.

Integrity violations must not go undetected (LANGER et al., 2010). From this requirement, we obtain the definition of verifiability.

Verifiability: Property in which the voter assures himself of the integrity of the individual vote and the public is assured of the integrity of the election result. Verifiability consists of evidence of the following aspects being provided:

- The vote correctly represents the voter's choice;
- The vote has been stored on the voting server or public web bulletin board as it was cast by the voter;
- All valid votes on the public web bulletin board are tallied without modification.

3.1.3 Notation

The traditional protocol description of Needham e Schroeder (ACM Press, 1978) consists of denoting each step of the protocol in one line. Each line is numbered and shows the direction of the information flow, generally from left to right. The sender is named on the left and the receiver is named on the right. Lastly, we have the message payload itself.

Figure 1 shows an example of this notation for a simple protocol of just two messages. First step stands for message "Show Email", sent from sender A to receiver S. Step number two has the answer "Learn Meeting Date" sent from S back to A.

1.	A	$\rightarrow$	S	:	Show Email
2.	S	$\rightarrow$	A	:	Learn Meeting Date

Figure 1 – Protocol notation

For the ceremonies in this chapter we add text below the arrows to identify the channel through which the message is being sent, as Carlos et al. (2013) do in their work. For instance, HD1 is a Human-Device channel with number 1 as ID. The same logic is applied for the DD channels. We enumerate them because each different pair of communicating devices forms a new channel (with its respective ID).

3.2 INSTITUTES WEBSITE PROPOSAL

We summarise the processes that voters would carry out using a web-based ballot verifier provided by the trusted institutes. We then analyse these processes using the adaptive threat model, and briefly compare our results to those obtained in the case of a DY attacker - closing this section with a discussion of the results.

3.2.1 Web-based verification and vote submission

The message flow for the web-based proposal is showed in Figure 2. Involved entities are represented as follows: V refers to Voter, Bo to Booth, Inst to Institute, App to the smartphone application and BB

to Bulletin Board.

The voting process is similar to that described in section 3.1. A relevant difference for this ceremony is that the voter enters the URL into the address bar of his browser in order to open the election website. He receives the voting credentials via postal mail, rather than clicking on a URL in the invocation email as in the original Helios.

We therefore concentrate on the verification processes where differences emerge between the original Helios and across the proposals discussed in this chapter. In order to verify that his vote is correctly encrypted on the voting device, a voter first needs to record the check-code displayed by the ballot preparation system (BPS) - see message 9 in Figure 2. This message presents the check-code contents which is a hash function (H) of the encryption of user's vote plus the randomness information (Rand) used. This encryption is done with the election public key (denoted by E_{pk}).

1.	V	$\xrightarrow[HD_1]{}$	Bo	:	URL
2.	Bo	$\xrightarrow[HD_1]{}$	V	:	Instructions Page
3.	V	$\xrightarrow[HD_1]{}$	Bo	:	"*Proceed to ballot*"
4.	Bo	$\xrightarrow[HD_1]{}$	V	:	Ballot Page
5.	V	$\xrightarrow[HD_1]{}$	Bo	:	Vote
6.	V	$\xrightarrow[HD_1]{}$	Bo	:	"*Review vote*"
7.	Bo	$\xrightarrow[HD_1]{}$	V	:	Review Page
8.	V	$\xrightarrow[HD_1]{}$	Bo	:	"*Encrypt vote*"
9.	Bo	$\xrightarrow[HD_1]{}$	V	:	$H\{\{Vote + Rand\}_{Epk}\}$
10.	V	$\xrightarrow[HD_1]{}$	Bo	:	"*Check encrypted vote*"
11.	Bo	$\xrightarrow[HD_1]{}$	V	:	Verifying Options Page
12.	V	$\xrightarrow[HD_1]{}$	Bo	:	Inst
13.	Bo	$\xrightarrow[DD_1]{}$	$Inst$	:	Vote + Rand
14.	$Inst$	$\xrightarrow[HD_2]{}$	V	:	$H\{\{Vote + Rand\}_{Epk}\}$ + Vote

Figure 2 – Verifying vote using website of independent institutes

The voter then expresses to BPS his intention to verify (in message 10), views the verification institutes that are available (in message 11), then selects an institute that he trusts (in message 12). He is re-directed to the verification web page of the selected institute, in a

new browser window. The BPS transmits the information necessary for verification, that is "Vote + Rand" and accompanying proofs to the selected institute (in message 13). Since the vote is transmitted to the institute, the case considered here is that the voter verifies a 'test vote', that is, one that is not equivalent to the final vote that he will cast.

The institute will compute the check-code using the information it receives from the booth, and displays this, along with the vote it received, to the voter (message 14). The voter now needs to confirm that the two check-codes match (the original check-code, produced by the BPS, must be equal to the check-code just computed by the institute), and that the vote displayed by the institute matches his selection on the ballot.

1.	V	$\xrightarrow[HD_1]{}$	Bo	:	URL
2.	Bo	$\xrightarrow[HD_1]{}$	V	:	Instructions Page
3.	V	$\xrightarrow[HD_1]{}$	Bo	:	"*Proceed to ballot*"
4.	Bo	$\xrightarrow[HD_1]{}$	V	:	Ballot Page
5.	V	$\xrightarrow[HD_1]{}$	Bo	:	Vote
6.	V	$\xrightarrow[HD_1]{}$	Bo	:	"*Review vote*"
7.	Bo	$\xrightarrow[HD_1]{}$	V	:	Review Page
8.	V	$\xrightarrow[HD_1]{}$	Bo	:	"*Encrypt vote*"
9.	Bo	$\xrightarrow[HD_1]{}$	V	:	$H\{\{Vote + Rand\}_{Epk}\}$
10.	V	$\xrightarrow[HD_1]{}$	Bo	:	"*Log in to cast vote*"
11.	Bo	$\xrightarrow[HD_1]{}$	V	:	Login Page
12.	V	$\xrightarrow[HD_1]{}$	Bo	:	VoterID + Password
13.	Bo	$\xrightarrow[DD_3]{}$	BB	:	$\{Vote\}_{Epk}$
14.	Bo	$\xrightarrow[HD_1]{}$	V	:	Vote is submitted
15.	V	$\xrightarrow[HD_1]{}$	Bo	:	Inst
16.	Bo	$\xrightarrow[DD_1]{}$	$Inst$	:	$H\{\{Vote + Rand\}_{Epk}\}$
17.	$Inst$	$\xrightarrow[HD_2]{}$	V	:	$H\{\{Vote + Rand\}_{Epk}\}$ + VoterID

Figure 3 – Submitting final vote using website of independent institutes

Next, we describe the process for the voter to verify that his vote is correctly stored on the voting server or public web bulletin board. This message flow is showed in Figure 3. The voter records the check-code displayed to him (in message 9). He then logs in to submit his

vote (in messages 10 - 12). Upon successful authentication, the booth submits the vote to the bulletin board (in message 13). The voter would select an institute from several options displayed after he submits his vote (in message 15). A new web page would open. He would enter the recorded check-code information and view the result returned by the institute (in message 17).

Note that the institute additionally needs to display the Voter ID in its response to the voter, in order to prevent a successful clash attack (KüSTERS; TRUDERUNG; VOGT, 2012). A clash attack occurs when different voters are showed the same check-code when they verify that their vote has been stored on the bulletin board. This can only be detected if the Voter ID is returned as it is a unique value.

3.2.2 Analysis

We apply the adaptive threat model of Carlos et al. (2013) in our ceremonies so we can conclude which scenarios are realistic and whether the attacker succeeds or not in his attempts to corrupt the system.

3.2.2.1 Preliminaries

With the full Dolev-Yao (DY) attacker capabilities in mind and applying the framework proposed by Carlos et al. (2013), we can evaluate our ceremonies against a less powerful and more realistic variation of such an attacker. We describe the adaptive threat model for each Helios ceremony and compare the results to the DY threat model. In the latter case, all the communication channels are under a full DY attacker.

In the adaptive threat model, only the device-device (DD) channel is under a DY attacker. Regarding the human-device (HD) channel, we assume there is a human being (and not a machine pretending to be a human) communicating with a device. Thus, the voter interacts with his device (for example, typing in the keyboard of his computer or looking at his computer screen). In this setting, we consider the HD channel as under a DY-E attacker. DY-E means that the attacker has all DY capabilities except the Eavesdrop (E) capability.

We are assuming controlled environments, where the voter does not need to check around his shoulders for someone eavesdropping his

actions. As the attacker will not be able to Eavesdrop on the user, we excluded such ability from his set of capabilities. This way, a DY-E attacker is not able to compromise the secrecy of the messages sent through HD channels. Once the attacker is not able to learn any voter's information, he can only apply his capabilities over the knowledge he already has. So, as he can only use his initial knowledge, he poses no threat to the voting and verifying ceremony.

We now present proofs for our Helios ceremonies making use of the following notation:

- $M_{1...10,12}$ represents the group of message consisting of the first message up to the tenth message, plus message 12, for example;
- $M_9 \cup DY$ means that message 9 is under the DY threat model, for example;
- Each peer's knowledge is showed with the predicate knows(Y), which stands for the knowledge set of agent Y[2]. When we say "x $\in$ knows(Y)" it means that agent Y knows message x. In other words, x belongs to the set of knowledge of peer Y;
- The $\wedge$ symbol stands for the logic connective AND.

Based on Figure 2, we check that the test vote is correctly encrypted on the voting device:

If the messages M1 to M12, and message M14 are run against a DY-E attacker, and message M13 is run against a DY attacker: the attacker (Att) can prevent the institute Inst from learning "Vote + Rand" and instead send $Vote_{att}$ + Rand instead, where $Vote_{att}$ is chosen by the attacker.

$$\frac{(M_{1...12,14} \cup DY - E) \wedge (M_{13} \cup DY)}{\begin{array}{r} Vote \wedge Vote_{att} \in knows(Att) \wedge \\ Vote \in knows(B) \wedge \\ (Vote + R) \notin knows(Inst) \wedge (Vote_{att} + R) \in knows(Inst) \end{array}}$$

Assume the attacker *Att* initiated two simultaneous pairing sessions between the booth Bo and the institute Inst in message 13. *Att* uses his block, atomic breakdown, fabricate and initiate capabilities in

[2]This predicate comes from the work of Carlos et al. (2013)

this message, preventing Inst from learning the correct vote and randomness information, that is (Vote+R), forcing it to receive (Vote$_{att}$ +R) instead.

Based on Figure 3, we check that the final vote is correctly stored on the voting server or public web bulletin board:

If the messages M1 to M12, M14, M15 and M17 are run against a DY-E attacker, and messages M13 and M16 are run against a DY attacker, the attacker (Att) can prevent the bulletin board BB from receiving the correct $\{Vote\}_{Epk}$. *Att can also prevent the institute Inst from learning* $H\{\{Vote + Rand\}_{Epk}\}$. *The attacker uses his crypto capability to generate the* $\{Vote_{att}\}_{Epk}$ *information and his fabricate capability to generate* $H_{att}\{\{Vote_{att}\}+R_{att}\}_{Epk}\}$. *Then, he sends these altered information* $\{Vote_{att}\}_{Epk}$ *and* $H_{att}\{\{Vote_{att}+Rand_{att}\}_{Epk}\}$ *to the bulletin board BB and the institute Inst, respectively, where* $Vote_{att}$ *is chosen by the attacker.*

$$\frac{(M_{1...12,14,15,17} \cup DY - E) \wedge (M_{13,16} \cup DY)}{\begin{array}{r}\{Vote\}_{Epk} \wedge \{Vote_{att}\}_{Epk} \wedge H\{\{Vote + Rand\}_{Epk}\}\wedge \\ H_{att}\{\{Vote_{att} + Rand_{att}\}_{Epk}\} \in knows(Att)\wedge \\ Vote \in knows(B) \wedge \{Vote\}_{Epk} \notin knows(BB)\wedge \\ \{Vote_{att}\}_{Epk} \in knows(BB)\wedge \\ H\{\{Vote + Rand\}_{Epk}\} \notin knows(Inst)\wedge \\ H_{att}\{\{Vote_{att} + Rand_{att}\}_{Epk}\} \in knows(Inst)\end{array}}$$

We assume the attacker *Att* initiated two simultaneous pairing sessions between booth Bo and the public web bulletin board BB maintained by institute Inst. The attacker Att uses his block, fabricate and initiate capabilities (message 13 in Figure 3) and sends $\{Vote_{att}\}_{Epk}$ to the BB - instead of the correct $\{Vote\}_{Epk}$ the BB was supposed to receive. Similarly, for message 16, we have that Inst receives the altered information $H_{att}\{\{Vote_{att} + Rand_{att}\}_{Epk}\}$ instead of $H\{\{Vote + Rand\}_{Epk}\}$. As a result, the user receives (in turn) the altered hash in message 17. This user, then, believes the institute is untrustworthy given that the hash in message 17 is different than the original hash previously displayed to him in message 9.

Note that the attacker can know the existing votes and the public key of the election, however the attacker cannot know the randomness information Rand.

3.2.3 Results

If we consider scenarios with the DY threat model, the attacker would have total control over the ceremonies above and would be able to manipulate the voter in the whole voting process. In these scenarios, the attacker intercepts all original sent messages, sending messages in his knowledge instead to booth Bo, institute Inst, and bulletin board BB.

At the same time, the attacker displays to the voter the right contents (those he blocked before reaching the intended receivers), pretending to be the legitimate entities. Therefore, the voter is led to believe that his vote was encrypted, submitted and stored as intended when this is not the case.

Nevertheless, such a scenario is highly unlikely to happen in real world situations, as the HD channel limits the attacker's actions. Furthermore, by involving the human peer, it is difficult for the attacker to control this latter channel, and the information being exchanged through it without being noticed.

A scenario that is realistic and feasible involves the attacker intercepting messages on the DD channel only. Therefore, the institute receives altered information and calculates a different check-code from the one expected by the voter. The voter then no longer trusts the institute, believing it to be unreliable. This outcome highlights the need for multiple institutes to be available, providing verification services to voters. The voter is free to verify with several other institutes. If these subsequent checks also return a failed result, he can then contact the election commission.

Analysing the two ceremonies presented above, we can see the message (Vote+R) being sent without any encryption in the first ceremony (for the test vote) while the second one contains the vote encrypted with the public key of the election (E_{pk}). From this, we can conclude that secrecy does not hold in the ceremony for the test vote, represented in Figure 2. Secrecy does hold when the voter decides to cast his final vote in the second ceremony, represented in Figure 3.

Our conclusions are given the fact that even when the DY attacker intercepts message 13 in the DD channel, he only sees a check-code that gives no information about the vote or the randomness information used. In other words, the attacker does not know the voter's final vote in the second ceremony, which is more secure than the test vote ceremony.

3.3 INSTITUTES APP PROPOSAL

We describe here the processes that voters would carry out to verify with a verifier installed as an app on their smartphones. With this proposal, the voter has a way to verify which is separate from the voting device. Thus, it uses a device that is in the voter's possession and that he likely trusts (with respect to secrecy and integrity). We analyse these ceremonies using an adaptive threat model, and compare our findings to those using a DY attacker. We conclude this section with the results of our analysis.

3.3.1 Using a smartphone app and QR codes

The message flow for this proposal is showed in Figure 4. The ballot preparation system (BPS) displays a QR code containing the check-code in addition to the human-readable value, in message 9. The voter opens the smartphone app and scans the QR code containing the check-code, in messages 10 and 11. This check-code will be stored by the app for later use during the verification process. The voter then expresses his intention to verify to the voting booth, in messages 12-14.

He scans a second QR code, in message 15, and the app computes the check-code and compares it to the first check-code. It then informs the voter that the check-codes match (in a success scenario), and prompts him to confirm that the displayed vote matches his initial input on the ballot, in messages 16 - 17. This is an implementation of a *forcing function*, preventing the voter from proceeding without confirming that his vote is correct (NORMAN, 2002).

In order to verify that a vote is correctly stored on the voting server or public web bulletin board (see Figure 5), the voter scans the first QR code containing the check-code as in the previous ceremony, in message 10. He logs in (messages 12 - 14) and the booth submits his vote upon successful authentication, in message 15. The voter instructs the app to check the public web bulletin board for the stored check-code, in message 17. The app does this by querying the public web bulletin board for the check-code, in message 18.

In a success scenario, the app displays a message to the voter that the check-code was stored on the public web bulletin board. To prevent clash attacks, the app should return the accompanying voter ID as well (KüSTERS; TRUDERUNG; VOGT, 2012). In a failure scenario, the voter will be informed that the check-code was not found. He can use

other apps for verification. To finalise the ceremony, the voter will be directed to contact the election commission in case of multiple checks with various apps return failed results.

1.	V	$\xrightarrow[HD_1]{}$	Bo	:	URL
2.	Bo	$\xrightarrow[HD_1]{}$	V	:	Instructions Page
3.	V	$\xrightarrow[HD_1]{}$	Bo	:	"*Proceed to ballot*"
4.	Bo	$\xrightarrow[HD_1]{}$	V	:	Ballot Page
5.	V	$\xrightarrow[HD_1]{}$	Bo	:	Vote
6.	V	$\xrightarrow[HD_1]{}$	Bo	:	"*Review vote*"
7.	Bo	$\xrightarrow[HD_1]{}$	V	:	Review Page
8.	V	$\xrightarrow[HD_1]{}$	Bo	:	"*Encrypt vote*"
9.	Bo	$\xrightarrow[HD_1]{}$	V	:	Check-code + $QRCode^*$
10.	V	$\xrightarrow[HD_3]{}$	App	:	"*Open/Scan*"
11.	Bo	$\xrightarrow[DD_2]{}$	App	:	QRCode
12.	V	$\xrightarrow[HD_1]{}$	Bo	:	"*Check encrypted vote with app*"
13.	Bo	$\xrightarrow[HD_1]{}$	V	:	QRCode Page
14.	V	$\xrightarrow[HD_3]{}$	App	:	"*Check that vote is correctly encrypted*"
15.	Bo	$\xrightarrow[DD_2]{}$	App	:	$QRCode^{**}$
16.	App	$\xrightarrow[HD_3]{}$	V	:	"*Check-codes match. Is this your vote?*" + Vote
17.	V	$\xrightarrow[HD_3]{}$	App	:	Confirm (Yes)
18.	V	$\xrightarrow[HD_3]{}$	App	:	"*Close*"

*Check-code is the hash $H\{\{Vote + Rand\}_{Epk}\}$. In messages 9 and 11, the QRCode has the check-code contents.

**In messages 13 and 15, the QRCode has $(Vote + Rand + E_{pk})$ information.

Figure 4 – Verifying vote using institutes' app

1.	V	$\xrightarrow[HD_1]{}$	Bo	:	URL
2.	Bo	$\xrightarrow[HD_1]{}$	V	:	Instructions Page
3.	V	$\xrightarrow[HD_1]{}$	Bo	:	"*Proceed to ballot*"
4.	Bo	$\xrightarrow[HD_1]{}$	V	:	Ballot Page
5.	V	$\xrightarrow[HD_1]{}$	Bo	:	Vote
6.	V	$\xrightarrow[HD_1]{}$	Bo	:	"*Review vote*"
7.	Bo	$\xrightarrow[HD_1]{}$	V	:	Review Page
8.	V	$\xrightarrow[HD_1]{}$	Bo	:	"*Encrypt vote*"
9.	Bo	$\xrightarrow[HD_1]{}$	V	:	Check-code + $QRCode^*$
10.	V	$\xrightarrow[HD_3]{}$	App	:	"*Open/Scan*"
11.	Bo	$\xrightarrow[DD_2]{}$	App	:	QRCode
12.	V	$\xrightarrow[HD_1]{}$	Bo	:	"*Log in to cast vote*"
13.	Bo	$\xrightarrow[HD_1]{}$	V	:	Login Page
14.	V	$\xrightarrow[HD_1]{}$	Bo	:	VoterID + Password
15.	Bo	$\xrightarrow[DD_3]{}$	BB	:	$\{Vote\}_{Epk}$
16.	Bo	$\xrightarrow[HD_1]{}$	V	:	Vote is submitted
17.	V	$\xrightarrow[HD_3]{}$	App	:	"*Check that vote is on bulletin board*"
18.	App	$\xrightarrow[DD_4]{}$	BB	:	$H\{\{Vote + Rand\}_{Epk}\}$
19.	BB	$\xrightarrow[DD_4]{}$	App	:	Confirmation + VoterID

*Check-code is the hash $H\{\{Vote + Rand\}_{Epk}\}$. In messages 9 and 11, the QRCode has the check-code contents.

Figure 5 – Submitting final vote using institutes' app

3.3.2 Analysis

Before we present the analysis and results, we first review necessary considerations for the adaptive threat model.

3.3.2.1 Preliminaries

Although DD channels are usually under a DY attacker, it is not realistic for channel DD_2 (seen in message 11 - Figure 4, for instance). This channel characterises what we call a 'visual channel', once there is no Bluetooth or connection of any kind between the involved devices.

In this case, we have the voter using his smartphone app to scan the QR code which is being displayed by the booth running in his computer. We consider the ideal case where the smartphone has no virus or worms, for simplicity.

Both smartphone and computer are considered to be in voter's possession, and not under the attacker's control. Hence, we have a similar situation as that of a HD channel (as described in section 3.2.2.1).

It is clear that the attacker cannot block any contents passing through DD_2 channel as this would imply the attacker literally blocking the computer screen from the voter and his smartphone. Hardly any attempt of an attack would succeed as the attacker's capabilities are very limited. The attacker cannot put his capabilities in motion without being noticed by the voter or without the possession over the voter's devices. The latter is only feasible if the voter leaves the devices unattended in the middle of the vote casting process. Nonetheless, instructions regarding the safety for the completion of the ceremony could be given to the voter. So, the voter would be aware of the threat model he is subject to and decide whether to continue or quit the remaining proceedings.

Therefore, we consider the DD_2 channel as being DY - E, similar to the HD channel. Any weakened variation of the DY - E attacker (capability set of the full DY attacker less the Eavesdrop capability) can be used because this attacker will not be effective. This is due to the fact that Eavesdrop is the only capability which can compromise the secrecy of the voter's vote.

We now move to the analysis of each ceremony involving the app individually, using the adaptive threat model, and the DY threat model, respectively.

Based on Figure 4, we verify that the vote is correctly encrypted on the voting device:

If all messages M1 to M18 are run against a DY-E attacker, the attacker cannot perform any significant attack with respect to secrecy and integrity.

$$\frac{M_{1\ldots18} \cup DY - E}{\emptyset}$$

For this ceremony, we consider that the DD_2 channel (in messages 11 and 15 of Figure 4) is not under a full DY attacker. This means that this channel is a weakened variation of the DY threat model, be-

cause the scenario involves a computer displaying a QR code to be scanned by the voter's mobile device. Considering that the attacker is unable to Eavesdrop on the communication, all the other capabilities the attacker has will not have an effect on the secrecy of the voter's vote in the ceremony. We justify this assumption once the environment can be controlled or the voter himself could watch over his shoulders to guarantee he is not being eavesdropped. Moreover, as the attacker cannot possess the voter's devices and he cannot know the voter's vote, he can perform no significant attack at all.

Based on Figure 5, we verify that the final vote is correctly stored on the voting server or public web bulletin board:

Consider messages M1 to M14, M16 and M17 are run against a DY-E attacker, and messages M15, M18 and M19 are run against a DY attacker. The attacker (Att) *can prevent the bulletin board BB from learning the correct values of* $\{Vote\}_{Epk}$ *and* $H\{\{Vote + Rand\}_{Epk}\}$.

$$\frac{(M_{1\ldots14,16,17} \cup DY - E) \wedge (M_{15,18,19} \cup DY)}{\begin{array}{r} Vote \wedge Vote_{att} \wedge \{\text{Vote}\}_{Epk} \wedge \\ \{\text{Vote}_{att}\}_{Epk} \wedge \text{H}\{\{\text{Vote} + \text{Rand}\}_{Epk}\} \wedge \\ \text{H}_{att}\{\{Vote_{att} + Rand_{att}\}_{Epk}\} \in knows(Att) \wedge \\ Vote \in knows(B) \wedge \\ \{\text{Vote}\}_{Epk} \wedge \text{H}\ \{\{\text{Vote} + \text{Rand}\}_{Epk}\} \notin knows(BB) \wedge \\ \{\text{Vote}_{att}\}_{Epk} \wedge \text{H}_{att}\{\{Vote_{att} + Rand_{att}\}_{Epk}\} \in knows(BB) \end{array}}$$

We assume the attacker *Att* initiated two simultaneous pairing sessions between the booth Bo and the bulletin board BB (message 15 in Figure 5) and between App and BB (messages 18 and 19 in Figure 5). We continue to assume that the DD_2 channel has a DY-E attacker since it is a visual channel. The attacker *Att* uses his capabilities of block, fabricate and initiate in messages 15, 18 and 19 where *Att* sends to the bulletin board BB a different value of the encrypted vote and a different value of the check-code, instead of the original ones.

3.3.3 Results

The ceremony seen in Figure 4 is more secure due to the presence of the visual channel, which limits the attacker's actions, once it has the

same behaviour as on the HD channels. If we consider the DY threat model, the attacker can trick the voter by manipulating the information displayed to him. Such a situation can be considered realistic (see Figure 5). However, it is highly unlikely to happen due to the fact that HD channels are secure under our assumption that the environment is controlled.

A very important contribution of the app proposal is that both test and final votes are secret, when compared to the proposal that uses the institutes (where the test vote is sent in clear through a full DY channel). Such a contribution means that the security property of secrecy holds in the app ceremony and, as the messages are no longer interrupted and modified, we can conclude this ceremony also ensures integrity.

The attacker cannot control more than one DD channel in the ceremonies for verification using the app showed above (CARLOS et al., 2013). Therefore, either the attacker controls the message between the booth Bo and the bulletin board BB (message 15 of Figure 5) or he controls the messages between App and BB (messages 18 and 19 of Figure 5). When the attacker succeeds, the bulletin board BB does not display to the voter the expected confirmation in message 19 of Figure 5. In such a situation, the voter would be advised to contact the election commission.

3.4 FINAL CONSIDERATIONS

The proofs presented in this chapter (contents of our first accepted and published paper) were subject to formal verification using the theorem prover SPASS (WEIDENBACH, 2007) and are available at: https://github.com/tacianem/HeliosSpass[3]

The first verification proposal consists of a web-based verifier provided by a trusted institute. Our results show the possibility of secrecy violations when the voter verifies that his vote is correctly encrypted on the voting device. Integrity violations were found when the voter also verifies that his vote is correctly submitted to the bulletin board. Integrity violations, on the other hand, take the form of 'reputation' attacks, resulting in the voter mistrusting the institute as he detects that it displays incorrect information.

The situation improves with regard to the smartphone app. First,

[3]At the time, this formalisation was made by one of the authors of our paper - strictly following Martina et al. (2015) ideas.

secrecy is maintained due to the presence of a visual channel, and because information is transmitted in encrypted form. Our results also show that no significant attack can occur when the voter verifies that his vote is correctly encrypted on the voting device. Integrity violations, as in the previous case, are reputation attacks which lead to a mistrust of the participating institute.

One can argue that the impact of the reputation attacks is low due to the availability of mitigating strategies. We highlight that integrity assurances in both verification proposals rest on the distribution of trust. It is, there are several options, whether web-based verifiers or smartphone apps from trusted institutes, available for the voter to use.

Should the verification process fail in one case, the voter can verify using other sources. Indeed, voters who do not want to trust any of the available institutes can use all the provided verification mechanisms. We acknowledge that this is not an ideal case with regard to usability, however, we note that it places less of a burden on the voter than would be the case if a more powerful adversary was considered.

Throughout this chapter, we were able to verify early properties desired for our Helios ceremony scenarios. Nevertheless, informal proofs are very limited and result in a great amount of time spent on short ceremonies. So, it is unfeasible to perform manual tests over long ceremonies. Having this in mind, we walk towards an automated ceremony analysis to quickly run tests in a reliable way. Starting on the next chapter, we have our proposal for a Distributed Attacker and its mechanisation in order to establish the symbolic evaluation of our 2-step verification scenarios.

4 PROPOSED DISTRIBUTED ATTACKER MODEL

Moving from our informal proofs to a strategy for security ceremony formalisation, this chapter brings two of the contributions of our second published paper.

We present our notation for the description of security ceremonies (4.1), based on the traditional protocol message description proposed by Needham e Schroeder (ACM Press, 1978), as suggestion to be standardised.

In addition, we demonstrate how our Distributed Attacker works and what are its main characteristics. We exemplify our attacker proposal through the threat modelling of our 2-step verification case study ceremony.

4.1 PROPOSED SYNTAX NOTATION

To be able to properly discuss properties on security ceremonies, we must establish the basis for the notation representing them. The notation we chose started with the early paper from Needham e Schroeder (ACM Press, 1978) and was refined over time. We augment this notation by adding the Ceremony Concertina layers, representing the information pathway (BELLA et al., 2014).

Recalling Figure 1, we continue to have enumerated steps corresponding to the ceremony messages, entities (first entity is the sender, and the second is the receiver), and message payload. We add the Concertina layer information (in capital letter) below the arrow in each step.

When describing ceremonies, independently of the approach chosen, it is necessary to show the layer involved in each message due to the presence of human peers along with devices and interfaces. Besides, linking messages and layers is important to demonstrate the information flow - as each message crosses the Concertina layers from one end to the other.

Our notation steps, however, are in a dotted version. This stands for a same given message crossing the Concertina layers (L3 through L1 or vice-verse). Thus, the contents of a message appears three times, each relating to a different Concertina layer. For instance, step 1.1 is the first message payload related to layer 1, as step 1.2 relates the same message to layer 2, and 1.3 to layer 3.

Our next step is to set a threat model for each layer. We believe that a notation is only fully comprehensible when put in the context of the threat model it is used with. We describe the threat model between parenthesis, after the layer information, stating the capabilities the attacker has for each layer. Such capabilities are either the Dolev-Yao full set of capabilities or a subset of this DY full set - accordingly to the adaptive threat model of Carlos et al. (2013) stated in 2.4.

We have the following four different threat model cases for a given layer:

- "N": represent safe model, meaning the absence of threat model or attacker;
- In case of a subset of the DY full set of capabilities containing just one capability, we write this capability in capital letter (e.g. "E");
- In case of a subset of the DY full set of capabilities (with size greater than one), we use "+" to split the capabilities. For example: "E+B" stands for the subset containing Eavesdrop and Block capabilities.
- "DY_{cap}": full set of the DY attacker capabilities;

To finalise, we add an attacker type to the set of capabilities, it is: DY for Dolev-Yao, MA for Multi-Attacker or DA for Distributed Attacker - our proposal for a distinct attacker type which is the major contribution of our work.

The DY attacker type is an attacker which has DY full capability set and follows strictly the DY threat model. This means that a DY attacker is always linked to the full set of the DY capabilities ($(DY_{cap})_{DY}$), and never to a subset of these capabilities.

On the other hand, we adapted the MA to be associated to different attacker capability sets. Its main difference from the DY model is the possibility of having more than only attacker. For this reason, we give each attacker a number to serve as his unique identification (e.g. MA_1).

Messages are described on the right side of each step, after the colon symbol (":"). Message components are separated by a comma (","). We denote the usage of public key encryption for messages crossing the network by involving the message contents with curly braces ("{" and "}"), and showing the public key itself at the end of the message.

Figure 6 illustrates our notation. Its two sample messages are the same as those used in the protocol example of Figure 1). Now we detail our example ceremony to show the difference of our syntax to the protocol description.

1.1	U_A	$\xrightarrow[L3_{(N)}]{}$	S	:	Show Email
1.2	U_A	$\xrightarrow[L2_{(E)_{MA_1}}]{}$	S	:	Show Email
1.3	U_A	$\xrightarrow[L1_{(DY_{cap})\,DY}]{}$	S	:	$\{\text{Show Email}\}_{Spk}$
2.1	S	$\xrightarrow[L1_{(DY_{cap})\,DY}]{}$	U_A	:	$\{\text{Learn Meeting Date}\}_{Spk}$
2.2	S	$\xrightarrow[L2_{(E)_{MA_1},(DY_{cap})_{MA_2}}]{}$	U_A	:	Learn Meeting Date
2.3	S	$\xrightarrow[L3_{(N)}]{}$	U_A	:	Learn Meeting Date

Figure 6 – Proposed ceremony notation

The first message can be read as: the user A (U_A) sends a message to server S requesting access to his email. This message starts with the user interacting with his device's graphical interface and then crosses through the following lower layers until it reaches the server.

The user interface is characterised by layer 3 (L3) of the Ceremony Concertina (step 1.1). We consider no threats for this layer.

This same first message then flows through the operating system of the user's device (L2). For this layer we present the possibility of a keylogger installed. We represent a keylogger by the Eavesdrop capability, as it allows the attacker to see and learn all contents typed in the infected device. Thus, we have the threat model for this step (1.2) denoted by E - subset of DY capabilities consisting only of the Eavesdrop capability.

Attacker type has a Multi Attacker with ID 1 (MA_1) for layer 2. Again, we add a number to differentiate MAs as they may be several and may attack more than one layer simultaneously.

We chose MAs (besides the well-known Dolev-Yao attacker type) to exemplify attackers acting in more than only one channel. As standard for protocol messages, a DY only acts on messages being transmitted in the network (layer 1 of the Concertina methodology). It is convenient to emphasise that both models do not allow the attacker to share information with other attackers in any protocol or ceremony.

Step 1.3 shows the message reaching the server through the insecure network (layer 1) which is under standard security protocol setting. This setting is a DY full capability set under a DY attacker type. We use the server public key (S_{pk}) for encrypting of messages crossing layer 1.

The second message brings the answer from server S to user A. This answer comes through the network (layer 1) first, assuming the same full set of capabilities under a DY attacker (step 2.1).

This message follows to layer 2 representing the possibly compromised operating system (OS) of the user's device (step 2.2). In here, we added one more attacker to show several attackers at the same time in the same layer. We may have as many attackers as we want for the ceremony layers.

The second MA (MA_2) with DY full set of capabilities (meaning a virus installed on the user's device). A virus is able to completely compromise this device, that is why we use all DY capabilities to represent it on layer 2.

With this new attacker we also want to bring attention to the change in the threat model throughout the ceremony execution. The user's computer only got infected during the second message (the answer from the server). We highlight here that our notation is able to capture such subtleties of the threat models and describe variations at exact points during the ceremony execution. This way we increase the number of scenarios we can test for a given ceremony.

Regarding the expressiveness of our proposed syntax notation we have a much more complete notation than the one used in Carlos et al. (2013), for instance. They just include the ceremony channel information for each step, leaving behind attacker types, threat models, and their changes during the ceremony execution.

Finally, the last step (2.3) of our example shows the answer from the server reaching the user through the interaction with his devices' interface (assumed secure - N).

4.2 THREAT MODELLING WITH THE DISTRIBUTED ATTACKER APPROACH

In this section, we demonstrate that our notation is comprehensive enough to describe subtleties of a real ceremony based on our 2-step verification email log-in case study.

Alongside attacker types DY and MA, we developed a new one

based on a distributed and flexible approach. We call our attacker type proposal "Distributed Attacker" (DA in brief), which can be subject to tests for analysis in all possible scenarios as the MA is.

In our DA model, we allow several attackers to be distributed throughout the layers of a security ceremony - even more than one per layer. It is, we may have any number of attackers distributed as they like on the ceremony layers.

A DA attacker may either control more than one ceremony layer or control exactly one specific layer in the whole ceremony (simulating the DY attacker behaviour). He is free to adapt himself to the best occasion, inclusive by changing his behaviour. If he decides to attack another layer, our notation is able to capture this alteration precisely when it happens.

Moreover, we consider that information sharing makes more sophisticated attacks feasible. A DA decides whether to share knowledge with other attackers or not (in the latter, attempting to corrupt the system by himself as in standard models). This attacker can have various associations and act in several layers with different powers.

In Table 1, we compare the attacker types used for this dissertation in terms of information sharing, capabilities and layers.

Attacker type	Share knowledge	Same capabilities	Different layers
DY	No	Yes	No
MA	No	Yes	Yes
DA	**Yes**	**No**	**Yes**

Table 1 – Comparison among attacker types

As we can see, the DY attacker type does not share knowledge (information sharing goes against its principles). A DY has always the full set of capabilities, so there is no change in its capabilities whatsoever. The major contrast here is that a DY appears only in the network layer (layer 1) - no different layers, just one.

The MA also does not share knowledge and has always the same capability setting (behaving as a DY attacker). On the other hand, he may appear in more than one layer - being able to attack both layers 1 and 2 (network and operating system, respectively).

Our DA model encourages the information sharing, although it is a decision the attacker makes during the ceremony execution. A DA can have different capability sets in each layer. Finally, our DA appears in any ceremony layer (ranging from layer 1 to 3). To summarise, our DA has the capacity of appearing in several layers with varied powers

- sharing knowledge whenever he judges advantageous.

A point to highlight about sharing knowledge under our threat model (DA) is that DAs can choose what information to share. So, for example, if a DA has capabilities of Eavesdrop and Initiate, he is able to decide whether to share only what he is eavesdropping or creating and sending to other peers, or even both information (up to all his knowledge).

In the next section, we apply our notation and attacker type proposals in the threat modelling of our 2-step verification email log-in case study ceremony. We present several scenarios, discussing our choices for the threat models of each ceremony layer.

4.3 2-STEP VERIFICATION CASE STUDY CEREMONY

We focus on the 2-step verification factor to exemplify the behaviour of our proposed DA model. We start with the default scenario for our 2-step verification email log-in case study.

The entities involved are C and P as the user's computer and phone, respectively. The interface between the user and his devices are then represented by U_C and U_P, while S stands for the server of the email. Italic messages stands for the user's s credentials. The remaining messages indicate the flow of the ceremony, such as the requests sent to the server and its responses.

For completeness, we now describe all the steps in our 2-step verification ceremony. They stand for the following communication between user and server: First of all, the user enters the email site on the browser of his computer. The user then waits for the email log-in page to be displayed back. When it does, the user enters his information of email address and password. Email server answers with the 2-step verification page to the user's browser, also sending an authentication code for the current session to the user's cell phone. The user reads this code from his phone and types it to the computer. Finally, the user successfully logs-in to his email account (given that he typed the code correctly and before the session expired).

Although the communication in our scenarios is between the user's devices and the server directly, they represent the user interacting with such devices. It is relevant to state that the user's actions are the key for the ceremony to achieve its goals.

4.3.1 Scenario I - Default

Our first scenario in Figure 7 represents the protocol for 2-step verification log-in itself (which is by definition also a ceremony). This means that we have the protocol threat model setting for this scenario: layer 1 (Internet) with standard DY threat model (DY full set of capabilities) under a DY attacker type as only threat. It shows that the network is insecure and subject to all kinds of conceivable attacks.

On the other hand, layers 2 and 3 are safe (no threat). No virus or keylogger is installed on user's devices, and no attacker is in the same environment as the user (e.g. attempting to eavesdrop him). Both computer and cell phone are safe in this basic scenario.

The first message (steps 1.1, 1.2 and 1.3) in Figure 7 has as sender U_C (user's computer) and receiver S (email server). Messages 3 and 6 (and their corresponding steps) follow the same structure as the first message. They are also sent from the user's device to the server (layer 3 to 1)

At some point, the server answers the request coming from the user. As such, messages 2, 4 and 7 (and their respective steps) stand for the responses from the server to the user - where the layers are crossed in contrary order (layers 1 to 3).

Message 5 (composed of steps 5.1 and 5.2) is a communication between the server and the user's cell phone U_P. We do not consider layer 1 for this message as no Internet is required. The phone simply receives the 2-step verification code sent by the email server as an SMS. Therefore, we have layer 2 standing for the operating system of the phone and layer 3 for the user interaction with his phone (to open the text message containing the code).

1.1	U_C	$\xrightarrow[L3_{(N)}]{}$	S	:	Email server URL
1.2	U_C	$\xrightarrow[L2_{(N)}]{}$	S	:	Email server URL
1.3	U_C	$\xrightarrow[L1_{(DY_{cap})DY}]{}$	S	:	$\{\text{Email server URL}\}_{Spk}$
2.1	S	$\xrightarrow[L1_{(DY_{cap})DY}]{}$	U_C	:	$\{\text{Email log-in page}\}_{Spk}$
2.2	S	$\xrightarrow[L2_{(N)}]{}$	U_C	:	Email log-in page
2.3	S	$\xrightarrow[L3_{(N)}]{}$	U_C	:	Email log-in page
3.1	U_C	$\xrightarrow[L3_{(N)}]{}$	S	:	*email,password*
3.2	U_C	$\xrightarrow[L2_{(N)}]{}$	S	:	*email,password*
3.3	U_C	$\xrightarrow[L1_{(DY_{cap})DY}]{}$	S	:	$\{email,password\}_{Spk}$
4.1	S	$\xrightarrow[L1_{(DY_{cap})DY}]{}$	U_C	:	$\{\text{2-step verification}\}_{Spk}$
4.2	S	$\xrightarrow[L2_{(N)}]{}$	U_C	:	2-step verification
4.3	S	$\xrightarrow[L3_{(N)}]{}$	U_C	:	2-step verification
5.1	S	$\xrightarrow[L2_{(N)}]{}$	U_P	:	Code message
5.2	S	$\xrightarrow[L3_{(N)}]{}$	U_P	:	Code message
6.1	U_C	$\xrightarrow[L3_{(N)}]{}$	S	:	*authentication code*
6.2	U_C	$\xrightarrow[L2_{(N)}]{}$	S	:	*authentication code*
6.3	U_C	$\xrightarrow[L1_{(DY_{cap})DY}]{}$	S	:	$\{authentication\ code\}_{Spk}$
7.1	S	$\xrightarrow[L1_{(DY_{cap})DY}]{}$	U_C	:	$\{\text{User's email page}\}_{Spk}$
7.2	S	$\xrightarrow[L2_{(N)}]{}$	U_C	:	User's email page
7.3	S	$\xrightarrow[L3_{(N)}]{}$	U_C	:	User's email page

Figure 7 – Default 2-step verification email log-in ceremony

4.3.2 Scenario II - Eavesdrop on layer 3

For our second scenario (Figure 8), we introduce a subtle powerful threat which consists of the Eavesdrop capability. The simple fact of having an attacker on the same environment as the user ("shoulder-surfing" him) may compromise significantly the insurance of a given ceremony. This is so because most people do not watch their back while entering critical credentials in their devices.

We now exemplify our first proposed attacker type DA. We enumerated our DAs (as well as the MAs) once we allow several attackers to be present over the layers, and we need to identify each attacker uniquely.

An interesting point we want to highlight is that an DA attacker which appears in layer 3 can have visual of both user's computer and cell phone devices. This way, the DA attacker controls two different communication channels over the same layer (L3).

In Figure 8, DA_1 is able to eavesdrop the user entering information in his computer and also on his cell phone. It happens because the user and the attacker are in the same environment while the user is switching his attention between his devices. Thus, the attacker is perfect capable of eavesdropping on the information being exchanged by the user on both devices.

It is important to clarify that, although user interactions with both computer and phone are represented by layer 3, they stand for different channels. The interaction between user and computer needs to be understand as independent from the interaction between user and his phone. For example, eventually only one of these channels may be compromised (say by a virus), none or both - all possibilities should be considered for a complete view of the threats present at a given scenario.

The messages for our scenarios are always the same, we just vary the threat model in order to analyse how each change impacts the final security level of the ceremony in question. In this example, the DA is not able to share information even if he is willing to do so because the only other attacker present in this context is a DY, which by definition does not share his knowledge.

1.1	U_C	$\xrightarrow[L3_{(E)_{DA_1}}]{}$	S	:	Email server URL
1.2	U_C	$\xrightarrow[L2_{(N)}]{}$	S	:	Email server URL
1.3	U_C	$\xrightarrow[L1_{(DY_{cap})DY}]{}$	S	:	$\{\text{Email server URL}\}_{Spk}$
2.1	S	$\xrightarrow[L1_{(DY_{cap})DY}]{}$	U_C	:	$\{\text{Email log-in page}\}_{Spk}$
2.2	S	$\xrightarrow[L2_{(N)}]{}$	U_C	:	Email log-in page
2.3	S	$\xrightarrow[L3_{(E)_{DA_1}}]{}$	U_C	:	Email log-in page
3.1	U_C	$\xrightarrow[L3_{(E)_{DA_1}}]{}$	S	:	*email,password*
3.2	U_C	$\xrightarrow[L2_{(N)}]{}$	S	:	*email,password*
3.3	U_C	$\xrightarrow[L1_{(DY_{cap})DY}]{}$	S	:	$\{\textit{email,password}\}_{Spk}$
4.1	S	$\xrightarrow[L1_{(DY_{cap})DY}]{}$	U_C	:	$\{\text{2-step verification}\}_{Spk}$
4.2	S	$\xrightarrow[L2_{(N)}]{}$	U_C	:	2-step verification
4.3	S	$\xrightarrow[L3_{(E)_{DA_1}}]{}$	U_C	:	2-step verification
5.1	S	$\xrightarrow[L2_{(N)}]{}$	U_P	:	Code message
5.2	S	$\xrightarrow[L3_{(E)_{DA_1}}]{}$	U_P	:	Code message
6.1	U_C	$\xrightarrow[L3_{(E)_{DA_1}}]{}$	S	:	*authentication code*
6.2	U_C	$\xrightarrow[L2_{(N)}]{}$	S	:	*authentication code*
6.3	U_C	$\xrightarrow[L1_{(DY_{cap})DY}]{}$	S	:	$\{\textit{authentication code}\}_{Spk}$
7.1	S	$\xrightarrow[L1_{(DY_{cap})DY}]{}$	U_C	:	$\{\text{User's email page}\}_{Spk}$
7.2	S	$\xrightarrow[L2_{(N)}]{}$	U_C	:	User's email page
7.3	S	$\xrightarrow[L3_{(E)_{DA_1}}]{}$	U_C	:	User's email page

Figure 8 – Eavesdrop on layer 3 in 2-step verification email log-in ceremony

4.3.3 Scenario III - Eavesdrop combined with virus

For our third scenario, depicted in Figure 9, we added a new DA for the user's computer (namely DA_2). Attacker DA_2 is able to eavesdrop through an active (real time) keylogger he installed in the user's computer, for instance. Both DA_1 and DA_2 are eavesdropping, but in different ways as to adapt to the layers they are acting upon. Now, we have the possibility of information sharing between DA_1 and DA_2.

We consider the capability of the attacker to retain what he eavesdrops only doable to some point. For example, too complicated passwords can be difficult for the attacker to memorise or even copy to a device of his own. This way, human cognitive limitations are properly taken into consideration in our analysis.

In this scenario, if DA_1 happens to miss the user's password (e.g. because it is too long), DA_2 is able to retrieve such an information through the installed keylogger. In exchange, DA_1 provides the code sent by the server to the user's cell phone (as such code is usually small and practical to remember). Due to the fact that DA_1 is in the same environment as the user, the probability of him being able to learn at least one of the user's credentials is high.

It is relevant to notice that although DA_1 may catch all needed information to impersonate the user, DA_2 has no access to the code sent to the user's cell phone at all. We consider attacker DA_2 to be remotely controlling the contents being exchanged in the user's computer only.

For the user's cell phone, we added a third DA with a virus (DA_3).. By doing so we just expanded the possibilities of collusion among the (current) three DA attackers. DA_3 has the piece of information DA_2 does not (which is the authentication code). If they work together, each one can easily get his part of the user's credentials and then they can log-in into the user's email account together.

We already mentioned the possibility of collaboration between DA_1 and DA_2 some paragraphs above. Now, for when DA_1 and DA_3 cooperate with each other, DA_1 would be responsible for memorising the user's password while DA_3 would retrieve the authentication code. Although DA_1 probably would get the code by himself (with no further help), maybe he still would be interested in the partnership for his own reasons. In all described alliances, the attackers involved are able to successfully steal user's credentials.

In case they decide not to share information with each other, they corrupt the system in their own ways. Again, the DY does not

1.1	U_C	$\xrightarrow[L3_{(E)_{DA_1}}]{}$	S	:	Email server URL
1.2	U_C	$\xrightarrow[L2_{(E)_{DA_2}}]{}$	S	:	Email server URL
1.3	U_C	$\xrightarrow[L1_{(DY_{cap})_{DY}}]{}$	S	:	$\{\text{Email server URL}\}_{Spk}$
2.1	S	$\xrightarrow[L1_{(DY_{cap})_{DY}}]{}$	U_C	:	$\{\text{Email log-in page}\}_{Spk}$
2.2	S	$\xrightarrow[L2_{(E)_{DA_2}}]{}$	U_C	:	Email log-in page
2.3	S	$\xrightarrow[L3_{(E)_{DA_1}}]{}$	U_C	:	Email log-in page
3.1	U_C	$\xrightarrow[L3_{(E)_{DA_1}}]{}$	S	:	*email,password*
3.2	U_C	$\xrightarrow[L2_{(E)_{DA_2}}]{}$	S	:	*email,password*
3.3	U_C	$\xrightarrow[L1_{(DY_{cap})_{DY}}]{}$	S	:	$\{email,password\}_{Spk}$
4.1	S	$\xrightarrow[L1_{(DY_{cap})_{DY}}]{}$	U_C	:	$\{\text{2-step verification}\}_{Spk}$
4.2	S	$\xrightarrow[L2_{(E)_{DA_2}}]{}$	U_C	:	2-step verification
4.3	S	$\xrightarrow[L3_{(E)_{DA_1}}]{}$	U_C	:	2-step verification
5.1	S	$\xrightarrow[L2_{(DY)_{DA_3}}]{}$	U_P	:	Code message
5.2	S	$\xrightarrow[L3_{(E)_{DA_1}}]{}$	U_P	:	Code message
6.1	U_C	$\xrightarrow[L3_{(E)_{DA_1}}]{}$	S	:	*authentication code*
6.2	U_C	$\xrightarrow[L2_{(E)_{DA_2}}]{}$	S	:	*authentication code*
6.3	U_C	$\xrightarrow[L1_{(DY_{cap})_{DY}}]{}$	S	:	$\{authentication\ code\}_{Spk}$
7.1	S	$\xrightarrow[L1_{(DY_{cap})_{DY}}]{}$	U_C	:	$\{\text{User's email page}\}_{Spk}$
7.2	S	$\xrightarrow[L2_{(E)_{DA_2}}]{}$	U_C	:	User's email page
7.3	S	$\xrightarrow[L3_{(E)_{DA_1}}]{}$	U_C	:	User's email page

Figure 9 – Eavesdrop combined with virus in 2-step verification email log-in ceremony

share information with any other attacker in the system.

4.3.4 Scenario IV - Dynamic change in the threat model

When moving from one scenario to the next, we do small changes to the threat models to assess how our notation and model deal with each change, highlighting important aspects of our proposals.

In Figure 10, we illustrate a dynamic change in the initial ceremony threat model. At the beginning, layer 2 had only attacker DA_2 eavesdropping on the user's computer for the first two messages of the ceremony. Now suppose the user is with several tabs opened in his browser and he happens to install a virus in other tab than the one in which our ceremony is running. Such virus is now on the user's computer and may compromise our ceremony from this point on.

For this scenario, we use an MA (namely MA_1) with DY full set of capabilities. Multi Attackers, as well as DY attackers, do not share knowledge with other attackers. From the third message until the last ceremony message, we see that the initial threat model for layer 2 (user's computer) changed and currently has two attackers: MA_1 and DA_2.

It is worthy to recall that the Ceremony Concertina allows the compression of layers (e.g. layers 1 and 2 seen as one). However, we do not compress layers because it could hide the dependence between each layer and the next one. Showing the threat models for all messages enables us to understand if anything has changed throughout the ceremony execution, and precisely in which step the changes took place. If the threat model remains intact for a given layer, we know that the communication through such layer was not altered.

1.1	U_C	$\xrightarrow[L3_{(E)_{DA_1}}]{}$	S	:	Email server URL
1.2	U_C	$\xrightarrow[L2_{(E)_{DA_2}}]{}$	S	:	Email server URL
1.3	U_C	$\xrightarrow[L1_{(DY_{cap})DY}]{}$	S	:	$\{\text{Email server URL}\}_{Spk}$
2.1	S	$\xrightarrow[L1_{(DY_{cap})DY}]{}$	U_C	:	$\{\text{Email log-in page}\}_{Spk}$
2.2	S	$\xrightarrow[L2_{(E)_{DA_2}}]{}$	U_C	:	Email log-in page
2.3	S	$\xrightarrow[L3_{(E)_{DA_1}}]{}$	U_C	:	Email log-in page
3.1	U_C	$\xrightarrow[L3_{(E)_{DA_1}}]{}$	S	:	*email,password*
3.2	U_C	$\xrightarrow[L2_{(DY)_{MA_1},(E)_{DA_2}}]{}$	S	:	*email,password*
3.3	U_C	$\xrightarrow[L1_{(DY_{cap})DY}]{}$	S	:	$\{\textit{email,password}\}_{Spk}$
4.1	S	$\xrightarrow[L1_{(DY_{cap})DY}]{}$	U_C	:	$\{\text{2-step verification}\}_{Spk}$
4.2	S	$\xrightarrow[L2_{(DY)_{MA_1},(E)_{DA_2}}]{}$	U_C	:	2-step verification
4.3	S	$\xrightarrow[L3_{(E)_{DA_1}}]{}$	U_C	:	2-step verification
5.1	S	$\xrightarrow[L2_{(DY)_{DA_3}}]{}$	U_P	:	Code message
5.2	S	$\xrightarrow[L3_{(E)_{DA_1}}]{}$	U_P	:	Code message
6.1	U_C	$\xrightarrow[L3_{(E)_{DA_1}}]{}$	S	:	*authentication code*
6.2	U_C	$\xrightarrow[L2_{(DY)_{MA_1},(E)_{DA_2}}]{}$	S	:	*authentication code*
6.3	U_C	$\xrightarrow[L1_{(DY_{cap})DY}]{}$	S	:	$\{\textit{authentication code}\}_{Spk}$
7.1	S	$\xrightarrow[L1_{(DY_{cap})DY}]{}$	U_C	:	$\{\text{User's email page}\}_{Spk}$
7.2	S	$\xrightarrow[L2_{(DY)_{MA_1},(E)_{DA_2}}]{}$	U_C	:	User's email page
7.3	S	$\xrightarrow[L3_{(E)_{DA_1}}]{}$	U_C	:	User's email page

Figure 10 – Threat model dynamic change in email 2-step verification log-in ceremony

4.3.5 Scenario V - A complex scenario

It is important to state that the ceremony depicted in Figure 11 is one of many. We are assuming the threat models and the communication flow so that the ceremony is achievable and yield the security goals. To evaluate any claims regarding our ceremony it is necessary to evaluate all its variants.

For our last scenario, we set the subset of capabilities Eavesdrop and Block (E+B) as the threat model for layer 3. We consider that the user will not let his computer unattended, so the contents the user is entering into the device will not be tampered by anybody else without the user's notice. This assumption allows us to remove the remaining capabilities as they rely on the attacker manipulating the user's devices himself. Once the devices are under user possession only, the only remaining feasible capabilities are Eavesdrop and Block.

Regarding the attacker type for layer 3, we set a DA attacker (DA_1). Once DA_1 is in the same environment as the user, DA_1 also appears in message 5 (where the user interacts with his phone).

In fact, literature has not yet established how the DY set of capabilities is manifested in the context of human-human communication. It is open to discussion how the capabilities may occur in the user environment and whether they remain realistic or not.

We believe Block is a feasible capability for layer 3. We see Block as an attempt of the attacker to delay the operations the user is attempting to perform, or disturb the surroundings of the user with noise of some kind - aiming to gain time or any advantage for an attack. Such circumstances are certainly more plausible then the attacker literally blocking the user's device screen out of no reason (which could also happen).

For layer 2, we consider a second DA (namely DA_2) with Eavesdrop (E) only - a keylogger. For layer 1 we consider the standard protocol setting DY as the threat mode, with a DY attacker.

Now, moving to the layers between server and cell phone: for step 5.1, layer 2 involves attacker MA_1. He does not share knowledge with any other attacker, and is controlling the user's cell phone via a virus. We have a second attacker type associated for this step. A third DA (DA_3) is controlling layer 2 also under a DY threat model, sharing or not his knowledge with others.

Besides the possible combinations of the three DA attackers mentioned before, they may also decide to cooperate among themselves. In this case, DA_2 and DA_3 could promptly manage to get the user's pass-

word and code, while DA_1's capability of Block would come in handy by talking to the user or delaying his actions somehow. This way, attackers DA_2 and DA_3 have the necessary time to access the user's account before him, invalidating the authentication code received. The user would need to get a new code for his next attempt to log-in to his (now compromised) email.

Our work is a proposal for a new attacker type, so it is liable of being tested. If the reader is not satisfied with our particular threat model choices, we encourage him/her to explore another scenarios and change the threat models to ascertain the results.

We would like to stress that once we can analyse such ceremonies with an automatic tool we may vary parameters, such as attacker powers and attacker types. This analysis would result in a set of secure ceremony scenarios for which these message exchanges hold security guarantees. In this sense, the intention is to produce ceremonies that satisfy security properties for their intended usage scenario.

1.1	U_C	$\xrightarrow[L3_{(E+B)_{DA_1}}]{}$	S	:	Email server URL
1.2	U_C	$\xrightarrow[L2_{(E)_{DA_2}}]{}$	S	:	Email server URL
1.3	U_C	$\xrightarrow[L1_{(DY_{cap})DY}]{}$	S	:	$\{\text{Email server URL}\}_{Spk}$
2.1	S	$\xrightarrow[L1_{(DY_{cap})DY}]{}$	U_C	:	$\{\text{Email log-in page}\}_{Spk}$
2.2	S	$\xrightarrow[L2_{(E)_{DA_2}}]{}$	U_C	:	Email log-in page
2.3	S	$\xrightarrow[L3_{(E+B)_{DA_1}}]{}$	U_C	:	Email log-in page
3.1	U_C	$\xrightarrow[L3_{(E+B)_{DA_1}}]{}$	S	:	*email,password*
3.2	U_C	$\xrightarrow[L2_{(E)_{DA_2}}]{}$	S	:	*email,password*
3.3	U_C	$\xrightarrow[L1_{(DY_{cap})DY}]{}$	S	:	$\{\textit{email,password}\}_{Spk}$
4.1	S	$\xrightarrow[L1_{(DY_{cap})DY}]{}$	U_C	:	$\{\text{2-step verification}\}_{Spk}$
4.2	S	$\xrightarrow[L2_{(E)_{DA_2}}]{}$	U_C	:	2-step verification
4.3	S	$\xrightarrow[L3_{(E+B)_{DA_1}}]{}$	U_C	:	2-step verification
5.1	S	$\xrightarrow[L2_{(DY)_{MA_1},(DY)_{DA_3}}]{}$	U_P	:	Code message
5.2	S	$\xrightarrow[L3_{(E+B)_{DA_1}}]{}$	U_P	:	Code message
6.1	U_C	$\xrightarrow[L3_{(E+B)_{DA_1}}]{}$	S	:	*authentication code*
6.2	U_C	$\xrightarrow[L2_{(E)_{DA_2}}]{}$	S	:	*authentication code*
6.3	U_C	$\xrightarrow[L1_{(DY_{cap})DY}]{}$	S	:	$\{\textit{authentication code}\}_{Spk}$
7.1	S	$\xrightarrow[L1_{(DY_{cap})DY}]{}$	U_C	:	$\{\text{User's email page}\}_{Spk}$
7.2	S	$\xrightarrow[L2_{(E)_{DA_2}}]{}$	U_C	:	User's email page
7.3	S	$\xrightarrow[L3_{(E+B)_{DA_1}}]{}$	U_C	:	User's email page

Figure 11 – Complex threat models for email 2-step verification log-in ceremony

5 DISTRIBUTED ATTACKER FORMALISATION

For this dissertation, we used the automated theorem prover SPASS [see (WEIDENBACH, 2007)] for our formalisation in First-Order Logic (FOL). We developed a script to translate ceremonies described with our proposed notation into symbolic formulae verifiable with SPASS. We adapted the Ceremony Concertina layers, along with the threat models and attacker types described in our work for our translation. Our mechanisation is based in the work of Martina et al. (2015), where they also employed SPASS as verification tool.

Our formalisation process consisted in translating all steps of our 2-step verification scenarios into FOL formulae. The attacker model was differentiated from Martina et al. (2015) by the creation of three different intruder predicates, representing the Dolev-Yao (DY), the Multi-Attacker (MA) and our Distributed Attacker (DA).

The DY() predicate is unary and represents the only almighty Dolev-Yao attacker. The MA() predicate takes as only argument the MA enumeration. Our contribution is the DA() predicate which also takes the attacker number (ID) as parameter. The option of willing to share information with other DAs can be made directly in the conjectures. In our specification, we represent the layers of the Ceremony Concertina and the DY capabilities. We created predicates for the layers by concatenating each layer to a capability in the DY set. For instance, L3_E() stands for the capability of Eavesdrop on layer 3 and has as parameters a sent predicate and the attacker identification.

The sent() function illustrates a message being sent from one peer to another, and as such has as parameters a sender, receiver and message payload[1]. This way, a message sent through layer 3 with attacker DA_1 eavesdropping is translated to the following formulae:

- formula(L3_E(sent(a,b,m), DA1)).

- ```
  formula( forall([xa, xb, xm, xatt],
          implies(
                  and(
                          Agent(xa),
                          Agent(xb),
                          Honest(xa),
                          Honest(xb),
                          Attacker(xatt),
  ```

[1]The sent() function is from Martina et al. (2015) model.
  ```

```
                Knows(xa, xm),
                L3_E(sent(xa,xb,xm),xatt)
            ),
            and(
                Knows(xb, xm),
                Knows(xatt, xm),
                L3_Sender(xa,xm)
            )
        )),
    Eavesdrop_L3).
```

The first formula above states that message m was sent from sender a to receiver b through layer 3 with DA_1 eavesdropping. It works based on the rule "Eavesdrop_L3" seen in the second item above. We defined such rule as: if xa and xb are honest agents (predicates Agent() and Honest() are valid for both xa and xb) - meaning xa and xb are not attackers; there is an attacker xatt; xa knows a given message xm[2] and xa sends message xm through the L3_E() predicate, then both receiver xb and attacker know (as the attacker is eavesdropping) message m. Finally, we set the sender for this message to be xa (we have a Sender() predicate only for precaution in case of a Spoof attempt).

Likewise, we have similar definitions for the remaining capabilities in the set of a DY attacker for layers 2 and 1. In other words, we created predicates for all capabilities in all layers. For instance, considering layer 3 our predicates are: L3_E (demonstrated earlier), L3_B, L3_S, L3_I, L3_C, L3_O and L3_F. Same predicates are also present for layers 2 and 1. As capabilities Modifying (M) and Replaying (R) are a combination of others we do not have specific predicates for them, we just use the already existing ones.

Our implementations for a given capability are the same for all layers. So our Eavesdrop capability (explained before) has the same implementation as predicates "Eavesdrop_L2" and "Eavesdrop_L1", because for SPASS it suffices to mechanise the idea for each capability. The conditions for the capabilities to be realistic for each layer must be defined by the ceremony designer when performing tests and evaluating the results.

Regarding the remaining capabilities, we also implemented them accordingly to their definitions described in our background (chapter 2) - as we can see with the Eavesdrop description above. Although we do not enter in further detail in here, we leave the link for our repository where our implementation is available: github.com/tacianem/CryptoForma.

[2]Knows() predicate is also from Martina et al. (2015).

It is important to emphasise that all mentioned predicates until here constitute the basis of our mechanisation, and as such are common for all scenarios. We now move to the specifics of each scenario (its individual threat modelling).

5.1 MECHANISATION OF SCENARIOS

We apply our symbolic evaluation strategy to each scenario (presented in the previous chapter) to support our claims. For scenario I, we simply proved that the DY attacker knows all messages sent through layer 1. However, he is not able to learn these messages contents as they are encrypted under the public key of the server. The conjecture for this proof is:

- formula(
  and(
  KnowsEncr(dy,encr(email_server_url,public_server)),
  KnowsEncr(dy,encr(email_log_in_page,public_server)),
  KnowsEncr(dy,encr(pair(email,password),public_server)),
  KnowsEncr(dy,encr(2step_verification,public_server)),
  KnowsEncr(dy,encr(authentication_code,public_server)),
  KnowsEncr(dy,encr(users_email_page,public_server))
  ),
  dy_knowledge).

In fact, this formula is present in all scenarios as we fixed the DY attacker for layer 1. Function encr() receives the message contents and the encryption key. It indicates that the given message is encrypted. The predicate KnowsEncr() links an encrypted message to the knowledge set of a peer of the ceremony, in this case the attacker. Declarations for functions and predicates are part of the SPASS structure, but irrelevant for the comprehension of our results.

For scenario II, we have also the set of conjectures for the knowledge of the DA_1 attacker:

- formula(
  and(
  Knows(da1,email_server_url),
  Knows(da1,email_log_in_page),
  Knows(da1,email),
  Knows(da1,password),
  Knows(da1,2step_verification),
  Knows(da1,code_message),
  Knows(da1,authentication_code),
  Knows(da1,users_email_page)

```
        ),
    da1_knowledge).
```

We can see that DA_1 knows the "code_message", which the DY attacker has no access to. The and() function operates similarly to the logical AND operation.

The DA_1 knowledge conjectures remain for scenario III, and we have a new set including the knowledge of attackers DA_2 and DA_3:

-
```
formula(
        and(
                Knows(da2,email_server_url),
                Knows(da2,email_log_in_page),
                Knows(da2,email),
                Knows(da2,password),
                Knows(da2,2step_verification),
                Knows(da2,authentication_code),
                Knows(da2,users_email_page),
                Knows(da3,code_message)
        ),
    da2_and_da3_knowledge).
```

It is noticeable that the information the attacker DA_3 lacks in knowledge is in the knowledge set of attacker DA_2 and vice-verse. This example encourages collaboration between these two DAs.

For scenario IV we have that MA_1 has almost the same knowledge as attacker DA_2, as he entered in layer 2 few steps after the very beginning of the ceremony:

-
```
formula(
        and(
                Knows(ma1,email),
                Knows(ma1,password),
                Knows(ma1,2step_verification),
                Knows(ma1,authentication_code),
                Knows(ma1,users_email_page)
        ),
    ma1_knowledge).
```

And, finally, for scenario V we have MA_1 with the same knowledge set of attacker DA_3:

-
```
formula(
        Knows(ma1,code_message)
    ma1_knowledge).
```

Sharing allows a flexible and powerful attack where each of the attackers involved does not need to get all the information to be successful in an attempt to stole user's credentials. Knowing part of the information is sufficient in this case. We are able to prove it in SPASS by just adding more formulae so that the set of the knowledge of both attackers (when summed) correspond to all needed credentials for the attack.

As example of how we do this, in the list of conjectures we present formulae which prove the knowledge of each peer of the system. We are adding the credentials known by attackers DA_2 and DA_3 below. As mentioned before, together they have both password and code. This way SPASS is able to prove our formulae and we conclude that together they are able to attack the user.

- formula(Knows(da2, password), da2_knows_password).

- formula(Knows(da3, auth_code_msg), da3_knows_code).

5.2 IMPLEMENTATION

In our repository on github, we have a complete description for each of the ceremony scenarios presented in the previous chapter - in files with extension ".tex". Theses files consist simply of the ceremony flows described with mathematical libraries for latex (exactly as we used to generate the figures presented throughout this dissertation).

At first, we tried to create a python[3] program to directly adapt our latex descriptions to equivalent ".dfg" files (readable by the SPASS theorem prover). However, it turned out that the latex files were not easily translated and the code got a bit cumbersome due to the regular expressions (regex) we needed to employ. Thus, we switched to json[4], as it is clean and straightforward. We kept the latex descriptions just for completion, as our focus changed to the ".json" files.

It is practical to map json to python and vice-verse, as python gives support to json. We, then, successfully developed our python program for the translation of json ceremony structures to SPASS readable files. Each json file has the following fields for each ceremony step:

- Sender: sender of the message corresponding to the given step;

[3]Available at: https://www.python.org/
[4]Available at: http://www.json.org/

- Layer: indicates the current layer the message is crossing;
- Attackers: Array of attackers. Each object in the array contains two properties - attacker and capability. The attacker field has the attacker's ID (DY, MAx or DAx, where x is the number which composes the ID). The capability property stands for the set of capabilities of the attacker. We use DY for the full set of capabilities or a combination formed by a subset of these capabilities, themselves if only one or we separate them by the symbol "+" - e.g. E and E+B, respectively). In case of no threat (no attackers whatsoever), this entry is empty;
- Receiver: receiver of the intended message;
- Message: contents of the message itself.

Now we move to the specifics about our python program ("ceremony_json_to_spass.py"). It iterates over each of the ceremonies descriptions in ".json" and automatically generates their corresponding descriptions in .dfg (SPASS extension). In order to do it, all the information present in each step is stored: agents, message contents, which layer the message is current related to, and the attackers and their respective powers for such a layer.

After all important data is stored, our program follows accordingly to our ceremony specification model designed in SPASS ("ceremony_model.dfg") to create a particular ".dfg" file for each ceremony scenario. Our model contains basic and generic information, which can be applied for any ceremony being analysed, such as our attacker types and layers predicates.

After our python program creates a SPASS file for each json one, it fills each SPASS file with the specifics of each scenario. It declares the agents and messages as variables. The program also links the agents to their initial knowledge. Afterwards, it states the messages in the order they happened, along with the layer information and the attackers capabilities. Then, the program creates a list of conjectures based on what knowledge the agents should have gained over the messages sent.

Finally, the python program calls the terminal to run each ceremony description created in ".dfg" in SPASS with the flag "-DocProof". The results for each ceremony are put in a file of same name but extension ".txt" for further reading. In these ".txt" files we can observe all the reasoning made by SPASS over the listed conjectures. It is possible to see whether the theorem prover SPASS actually found a proof for each conjecture, or if it just ran all feasible derivations and got

nothing. As expected, we were able to prove all our conjectures, successfully linking the contents of the messages to the attackers present at the compromised ceremony layers.

5.3 FINAL CONSIDERATIONS

With our approach, we believe that the usage of the cell phone contributed to the safety of the ceremony as it is certainly more difficult for the attackers to handle both devices than just one. If we consider a situation where the user is not using his personal computer (if he is at work or in a public environment, for instance), it increases the chance of this computer being compromised. In this untrustworthy environment, an ttacker may have access to the user's account by stealing the user's password with a keylogger for instance. In here the 2-step verification is crucial, where the user (in possession of his cell phone) has a second credential to which the attacker has no access (ideally). This example shows a scenario where it is feasible to attack without the 2-step verification, and it is not when the user has 2-step verification activated. This can even be proven with our mechanisation, although we are only showing our particular scenarios in here.

Now considering a virus in the untrustworthy computer the user is using, the attacker would know in real time the user's password. For 2-step verification scenarios, then, this attacker would have to steal the code sent to the user's cell phone in some other way (which makes the attack harder). Our formalisation is also able to capture that.

In a DA scenario, even with the 2-step verification it is less difficult for cooperating attackers to log-in the user's account pretending to be him. Suppose we have a DA on the Internet layer (L1) of a user's computer and another DA in the operating system (L2) of this same user's cell phone. Consider the first DA happens to steal the user's password by a broken HTTPS link or other means, and the second DA is able to check the code sent to the user's cell phone via a virus. Therefore, it is clear that threats involving the user environment while he interacts with his devices may be more devastating than we expect.

This way, it is possible to exchange information between them in order for both to have knowledge of the password and the code, being able to successfully attack the user. If the user logs in before them, the captured code will no longer be valid. Nevertheless, this mischievous attack can be repeated later in time. Among the current threat models and attacker types, we do not have such a possibility.

With our proposal, we are able to formally analyse complex scenarios like this one.

6 CONCLUSIONS

In the beginning of our work, we informally analysed the security of two verification proposals made to improve the usability of the Helios voting system. We considered the verification processes in Helios as ceremonies in order to perform our analysis. This first part of our research clearly shows that informal methods are already capable of showing us some insights about the security of the system. However, this manual work takes too much time even for short ceremonies. Given that, we continued our formalisation study where we started developing a more precise description syntax for security ceremonies. In our syntax there is space for describing the agents involved, messages and the layers they traverse, as well as the threat models and attacker types related to each layer.

As our syntax comes directly as a variation of the notation used for protocols, it is easy to be put in practice. A complex and completely different notation from the ones currently in use would complicate the migration from protocol to ceremony description. Besides, our proposed syntax notation is clear and efficient in linking all needed information for each ceremony step.

We proposed a new attacker type, Distributed Attacker, where the attacker may act in several layers at once. A DA can have different capabilities in each layer or be the same in all layers under his domain. Furthermore, he may or not share his knowledge with other attackers in order to perform more sophisticated attacks. Such an approach is powerful in allowing the attackers to have information about more layers than usual. Moreover, we have combined Dolev-Yao and Multi-Attacker models to our proposal for a Distributed Attacker. All three attacker types are under the well known Dolev-Yao threat model or the adaptive threat model.

It is relevant here to recall the differences in the origin of each of the attacker types we employ in our work and that we just cited above. The DY model dates back to the Cold War, and represents the most powerful protocol attacker until nowadays. On the other hand, the MA model follows from the Internet era and its change in the attacker prototype. As such, any agent (internal or not) is allowed to attack the protocol. In this setting, we may have several MAs behaving as DY attackers: not colluding or sharing information with other attackers. Our DA proposal comes in a "Post-Snowden" era, where users are becoming more aware of the threat models they are subject to.

As current attacker types do not take into account information sharing, they are not suitable to describe and assess important ceremonies (such as our 2-step verification case study). The main gain in using our DA model comes for ceremonies with more than only one user credential. That justifies our choice for a 2-step verification example ceremony, where we have two credentials: user's password and server code sent to the user's cell phone.

We encourage information sharing and cooperation among attackers as it decreases the number of credentials required for each attacker to gather. In other words, instead of having one attacker acting alone and pursuing both password and code, we propose two (or more) attackers acting together: one gets the password and the others gets the code (for instance). This way, our model reflects the espionage era we live in, and allows us to design more complex attacks.

Furthermore, we brought the adaption of a formalisation technique from Martina et al. (2015), which enabled us to verify our claims and to experiment with the subtleties of security ceremonies. Our mechanisation now encompasses our proposed notation, the Security Ceremony Concertina methodology, our DA proposal (along with DY and MA attacker types), and the threat models used.

We showed several ceremony scenarios to exemplify our attacker type, discussing feasible threat models for each ceremony layer. We represented various types of infection of the user's devices, and tested the outcomes using our symbolic evaluation. We believe that the usage of formal methods helps us understand practical problems in security ceremonies.

Our next step is to study how to expand the applicability of our approach, through the creation of push button systems to do ceremony verification. Our plan is to have a fast and automatic analysis for any scenario, under our syntax and threat modelling. For example, we can extend our model by adding predicates and/or functions such as Triple() and Quadruple() - for now we only support the pair() function. Other interesting path for future work is the study of how to apply Pirandellian masks to shape the ceremony accordingly to the user's behaviour.

We also envisage the specification of our notation as a grammar in order to do the classification of security ceremonies, given the security properties they hold. In this case, we would classify a group of ceremonies in sets where each set contains the subset of ceremonies that guarantees the property represented by the set. This way, the ceremony designer has the advantage of choosing which ceremony better

fits his necessities. He can also assess the big picture of all achievable properties that ceremonies can grant for the problem in question.

REFERENCES

ADIDA, B. Helios: Web-based open-audit voting. In: *Proceedings of the 17th Conference on Security Symposium*. Berkeley, CA, USA: USENIX Association, 2008. (SS'08), p. 335–348.

ADIDA, B. et al. Electing A University President using Open-Audit Voting: Analysis of Real-World Use of Helios. In: *Proceedings of the 2009 Conference on Electronic Voting Technology/Workshop on Trustworthy Elections*. [S.l.]: Usenix Association, 2009. (EVT/WOTE'09), p. 10–10.

ARSAC, W. et al. Multi-attacker protocol validation. *Journal of Automated Reasoning*, Springer, v. 46, n. 3-4, p. 353–388, 2010. ISSN 0168-7433. <http://dx.doi.org/10.1007/s10817-010-9185-y>.

BELLA, G. *Formal Correctness of Security Protocols*. [S.l.]: Springer, 2007. 274 p. (Information Security and Crypto.).

BELLA, G. et al. A socio-technical methodology for the security and privacy analysis of services. In: *COMPSACW*. [S.l.: s.n.], 2014.

BELLA, G.; LONGO, C.; PAULSON, L. C. Is the verification problem for cryptographic protocols solved? In: CHRISTIANSON, B. et al. (Ed.). *Security Protocols Workshop*. Springer, 2003. (Lecture Notes in Computer Science, v. 3364), p. 183–189. ISBN 3-540-28389-7. <http://dx.doi.org/10.1007/11542322$_2$3>.

BURROWS, M.; ABADI, M.; NEEDHAM, R. A logic of authentication. *ACM Trans. Comput. Syst.*, ACM, New York, NY, USA, v. 8, n. 1, p. 18–36, 1990. ISSN 0734-2071.

CARLOS, M. C. et al. An Updated Threat Model for Security Ceremonies. In: *ACM Symposium on Applied Computing*. [S.l.]: ACM, 2013. (SAC '13).

CARLOS, M. C. et al. A proposed framework for analysing security ceremonies. In: *SECRYPT*. [S.l.: s.n.], 2012. p. 440–445.

CORTIER, V. et al. *A Generic Construction for Voting Correctness at Minimum Cost - Application to Helios*. 2013. Cryptology ePrint Archive, Report 2013/177.

CORTIER, V.; SMYTH, B. Attacking and Fixing Helios: An Analysis of Ballot Secrecy. In: . [S.l.]: IEEE Computer Society, 2011. (CSF '11), p. 297–311.

DEMIREL, D.; GRAAF, J. V. D.; ARAúJO, R. Improving Helios with Everlasting Privacy Towards the Public. In: . [S.l.]: USENIX Association, 2012. (EVT/WOTE'12).

DOLEV, D.; YAO, A. C. On the Security of Public Key Protocols. *IEEE Transactions on Information Theory*, v. 29, n. 2, p. 198–208, 1983.

ELLISON, C. *Ceremony Design and Analysis*. 2007. Cryptology ePrint Archive, Report 2007/399.

IACR. *IACR Board of Directors 2013 Election and Referendum on Bylaws Amendments*. 2013. Accessed on 30th June, 2015. <https://vote.heliosvoting.org/helios/elections/b36cbf0c-250a-11e3-89f4-46d2afa631be/view>.

KARAYUMAK, F. et al. Usability Analysis of Helios - An Open Source Verifiable Remote Electronic Voting System. In: *Proceedings of the 2011 International Conference on Electronic Voting Technology/Workshop on Trustworthy Elections*. [S.l.]: USENIX Association, 2011. (EVT/WOTE'12).

KARAYUMAK, F. et al. User Study of the Improved Helios Voting System Interfaces. In: *STAST*. [S.l.: s.n.], 2011. p. 37–44.

KüSTERS, R.; TRUDERUNG, T.; VOGT, A. Clash Attacks on the Verifiability of E-Voting Systems. In: *IEEE Symposium*. [S.l.: s.n.], 2012. p. 395–409.

LANGER, L. et al. A Taxonomy Refining the Security Requirements for Electronic Voting: Analyzing Helios as a Proof of Concept. In: IEEE. *ARES'10*. [S.l.], 2010.

LOWE, G. Breaking and fixing the needham-schroeder public-key protocol using fdr. In: . Springer-Verlag, 1996. (TACAs '96), p. 147–166. ISBN 3-540-61042-1. <http://dl.acm.org/citation.cfm?id=646480.693776>.

MARTINA, J. E.; CARLOS, M. C. Why should we analyse security ceremonies. In: *First CryptoForma Workshop*. [S.l.: s.n.], 2010.

MARTINA, J. E. et al. An adaptive threat model for security ceremonies. *IJIS*, Springer Berlin Heidelberg, v. 14, n. 2, p. 103–121, 2015. ISSN 1615-5262. <http://dx.doi.org/10.1007/s10207-014-0253-x>.

MEADOWS, C. Formal verification of cryptographic protocols: A survey. In: *Asiacrypt 96*. [S.l.: s.n.], 1996.

NEEDHAM, R. M.; SCHROEDER, M. D. Using Encryption for Authentication in Large Networks of Computers. ACM Press, 1978.

NEUMANN, S.; BUDURUSHI, J.; VOLKAMER, M. Analysis of Security and Cryptographic Approaches to Provide Secret and Verifiable Electronic Voting. In: _____. [S.l.]: IGI Global, 2013.

NIPKOW, T.; PAULSON, L. C.; WENZEL, M. *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*. [S.l.]: Springer, 2002. LNCS Tutorial 2283.

NORMAN, D. A. *The Design of Everyday Things*. [S.l.]: Basic Books, Inc., 2002.

PAULSON, L. C. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, v. 6, p. 85–128, 1998.

PIRANDELLO, L. *Sei personaggi in cerca d'autore*. Gutenberg Project, 1922. <http://www.gutenberg.org/ebooks/18457>.

PRINCETON. *Princeton Undergraduate Elections*. 2013. Accessed on 30th June, 2015. <https://princeton.heliosvoting.org/>.

SASSE, M. A.; FLECHAIS, I. Usable Security: Why Do We Need It? How Do We Get It? In: *Security and Usability: Designing Secure Systems That People Can Use*. [S.l.]: O'Reilly, 2005.

SNOWDEN Interview: Transcript. 2014.

TSOUKALAS, G. et al. From Helios to Zeus. *EVT/WOTE'13*, Usenix Association, 2013.

WEIDENBACH, C. *SPASS Input Syntax Version 1.5*. [S.l.], 2007.