Int. J. Inf. Secur. (2015) 14:103–121
DOI 10.1007/s10207-014-0253-x

SPECIAL ISSUE PAPER

An adaptive threat model for security ceremonies

Jean Everson Martina · Eduardo dos Santos · Marcelo Carlomagno Carlos · Geraint Price · Ricardo Felipe Custódio

Published online: 7 August 2014

Abstract Ever since Needham and Schroeder introduced the notion of an active attacker, significant research has been conducted regarding protocol design and analysis to verify that the protocols' goals are robust against this type of attacker. Nowadays, the Dolev–Yao threat model is the most widely accepted attacker model for the analysis of security protocols. Consequently, there are several security protocols considered secure against an attacker under Dolev–Yao's assumptions. With the introduction of the concept of ceremonies, which extends protocol design and analysis to include human peers, we can potentially find and solve security flaws that were previously not detectable. In this paper, we discuss that even though Dolev–Yao's threat model can represent the most powerful attacker possible in a ceremony, the attacker in this model is not realistic in certain scenarios, especially those related to human peers. We propose a dynamic threat model that can be adjusted according to each ceremony and consequently adapt the model and the ceremony analysis to realistic scenarios. We demonstrate the feasibility of our approach with a support implementation using first-order logic and an automatic theorem prover.

Keywords Threat models · Security ceremonies · Security protocols · Formal verification and specification

Supported by CNPq and FINEP Brazil.

J. E. Martina (✉) · R. F. Custódio
Departamento de Informática e de Estatística, Universidade Federal de Santa Catarina, INE Campus Universitário, Florianópolis, Brazil
e-mail: everson@inf.ufsc.br

R. F. Custódio
e-mail: custodio@inf.ufsc.br

E. dos Santos
Centre for Doctoral Training in Cyber Security, Department of Computer Science, University of Oxford, Parks Road, Oxford OX1 3PW, UK
e-mail: eduardo.dossantos@cybersecurity.ox.ac.uk

M. C. Carlos · G. Price
Information Security Group, Royal Holloway, University of London, Egham TW20 0EX, UK
e-mail: marcelo.carlos.2009@rhul.ac.uk

G. Price
e-mail: geraint.price@rhul.ac.uk

1 Introduction

Security protocols are generally secure against passive attackers who eavesdrop the communication medium. However, since Needham and Schroeder introduced the idea of an active attacker in 1978 [22], a lot of research has been conducted in this area in order to prove protocols' claims. Needham and Schroeder's attacker model assumed that the attacker could alter, copy, replay and create messages (or parts of messages) in all communication paths. Dolev and Yao [18] further developed this attacker model by formalising it and adding new assumptions. In general, we can say that the Dolev–Yao attacker has complete control of the network, but is not able to perform cryptanalysis.

Nowadays, the Dolev–Yao threat model is the most widely accepted model to analyse security protocols [9]. Consequently, there are several security protocols considered secure against Dolev–Yao's assumptions. In general, we assume that if a protocol is secure against this powerful attacker, it is secure against less powerful variations.

However, recent research [16,19,20] shows that even protocols verified under Dolev–Yao threat model assumptions might be susceptible to attacks when implemented. Several security protocols, when implemented in practice, are used by humans. The non-deterministic nature of human behaviour can produce situations where an unexpected (but plausible) operation is performed. In general, these problems happen not due to a design flaw or user's misconduct, but due to the implementation of a protocol assumption. Ellison [19] introduced the concept of a "ceremony" which focuses on the context surrounding security protocols. In his definition, a ceremony is an extension of the network protocol; its nodes may be humans or computers, and the communication channels are not limited to the network.

By extending protocol analysis to ceremony analysis, we can potentially find and solve security flaws that were previously not detectable. However, the formal verification of ceremonies is not a straightforward process. In ceremonies, we have new communication mediums, new nodes and consequently new attacker variations. For that reason, an appropriate threat model must be designed to fit into this new architecture. We argue that, even though Dolev–Yao's threat model can represent the most powerful attacker possible in a ceremony, the attacker in this model is unrealistic in some scenarios. One of the reasons that certain protocols fail when implemented is that their assumptions are either not well specified or not realistic, forcing implementations to create mechanisms to circumvent these problems. Consequently, these workarounds may introduce security problems, making the implementation of the protocol flawed in certain contexts. This is due to an inaccurate assumption forced by an unrealistic threat model, despite the fact that the problem was created during the implementation. In this paper, we revisit Dolev–Yao's threat model so we can have a tailored threat model for ceremonies, allowing more aligned design and implementation components for ceremonies.

In Sect. 2, we present a comparison between ceremonies and protocols. We describe protocols' threat models in Sect. 3. Our premises for the proposed threat model for ceremonies are discussed in Sect. 4. The proposed threat model is presented in Sect. 5. We then introduce some concepts concerning the Bluetooth protocol in Sect. 6 which will be needed in our scenarios in Sect. 7. Section 8 discusses our formal specification of the threat model using first-order logic and the automatic theorem prover SPASS. In Sect. 9, we verify some of the security goals of the ceremony proposed in Sect. 6, and in Sect. 10, we discuss the gains of describing ceremonies under a realistic threat model. Finally, we conclude our paper with our final thoughts and future expectations in Sect. 11.

2 Ceremony versus protocol

Security protocols can be defined as a prescribed sequence of interactions between principals designed to achieve certain goals [24]. Their goals include authentication, key distribution, secrecy and anonymity, amongst others. Security ceremonies [19] are a superset of security protocols (as we can see in Fig. 1). Ceremonies, when compared to protocols, include additional node types, communication channels and operations that were previously out-of-bounds. As an example of these out-of-bound operations, we have user interaction and pre-key distribution.

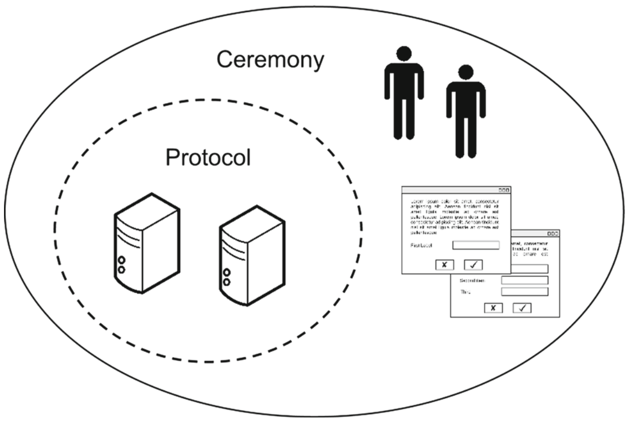

Fig. 1 Protocol and ceremony association

The main difference, and probably the most challenging in terms of design and analysis, is the addition of human nodes into the specification. In a protocol specification, we usually have devices (e.g. computers) communicating with other devices via a network medium. By including a human node, we have to define and use extra mediums such as user interfaces for human–device interaction and a human medium to represent speech, gestures, etc., for human–human [1] interaction.

In a protocol specification, we define assumptions as a representation of out-of-bound operations. In ceremonies, we break down these assumptions into smaller and better defined assumptions. The inclusion of human interaction, behaviour and cognitive processes adds a considerable amount of complexity to the design and verification tasks. However, their inclusion also enriches the details and coverage of the analysis.

In the same way that a ceremony allows a more detailed analysis of a protocol, the capabilities of an attacker, or the threat they pose within the ceremony, also requires more detailed analysis. The assumptions made by Needham and Schroeder [22] and extended by Dolev and Yao [18] are the current standard for protocol analysis. However, for ceremonies, they are not always consistent with real-world threats. For example, the notion that an attacker would be capable of modifying (or replaying) a "speech" packet in a human–human medium is unrealistic if this communication were to happen face-to-face.

We anticipate that by specifying and verifying security ceremonies, we will be able to encompass a more human-

[1] Human–human interaction in this paper means face-to-face and in-person communication.

centric security view. The extensions to Dolev–Yao's threat model will help us design more realistic ceremonies that will assist the human peer in assessing the threat they are subject to. By not overstating assumptions, we inherently make them plausible and achievable. An example of how a protocol can assist a human peer in assessing the threat model they are subject to can be seen in Sect. 7.

3 Abstract threat models for protocols

Security protocols were initially designed to be safe against a "passive" attacker; that is, an attacker able to eavesdrop the communication channels and make use of its contents [18]. The idea of the existence of an "active" *attacker*, also called *intruder*, *saboteur* or *spy*, was first mentioned in the classical Needham–Schroeder paper [22]. They assumed that an attacker could intervene between parties in all communication channels. Such an attacker is capable of altering, copying, replaying and creating messages. Needham and Schroeder claim that, although it is a very pessimistic scenario, the only way an authentication protocol can be considered safe is if it resists such a powerful attacker. Subsequently, this definition became widely accepted and applied when discussing whether a protocol is correct or not. In summary, assuming that the machines involved in the protocol are safe and that cryptography cannot be broken by brute force or cryptanalysis, Needham and Schroeder also assume that an attacker can:

– obtain any message that goes through the communication channel;
– modify messages and its subcomponents;
– copy messages and its subcomponents;
– replay messages;
– create messages.

In the following years, Dolev and Yao [18] formalised Needham–Schroeder's attacker and described the attacker capabilities in more detail, in order to allow protocols to be analysed more precisely, with fewer assumptions regarding the attacker's behaviour. They added the following assumptions to the attacker's capabilities:

– The attacker is a valid agent of the network and can therefore initiate communication with any other agent.
– The attacker can prevent the recipient from receiving a message.
– The attacker, despite being unable to read the content of an encrypted message (for which they do not know the key), can forward its contents to another agent.
– The attacker can perform any operation over a message except cryptanalysis. For example, they can encrypt and decrypt messages using a key they know.

This threat model is known as Dolev–Yao and is a de facto standard for symbolic protocol analysis at present [9]. In summary, in this model, the attacker controls the communication channels. From the Dolev–Yao threat model, we have seen the development of two lines of research. The first considers that the Dolev–Yao model should be extended because such an attacker cannot perform cryptanalysis. They take a probabilistic reasoning-based approach, which has been described by Bellare and Rogaway [11]. This research thread has evolved over the years in parallel with the Dolev–Yao research line [4], although there are a few efforts to reconcile them [1,6]. The second follows the idea that if a protocol is secure against a Dolev–Yao attacker, it is secure against a less powerful variation. Furthermore, the latter argues that subtleties of protocol attacks can still be found even after a protocol is proved correct against Dolev–Yao. By adjusting the powers of the attacker to adhere to the real world and using the symbolic approach, such subtleties may be discovered [4].

Our main focus is on the second line of research. A first example of a threat model that considers variations of Dolev–Yao's attacker capabilities is *BUG* [8]. In *BUG*, the agents are no longer classified as attacker and non-attacker; instead, they are partitioned into three groups: *Bad, Ugly and Good*. *Bad* agents may (or may not) collude with each other and attempt to break the protocol for their own (illegal) benefits. *Ugly* agents have an intermediate type of behaviour. They may follow the protocol or may deliberately not do so, letting the *Bad* principals exploit them. *Good* are principals who follow the protocol and its rules. The *BUG* threat model is interesting because of the novelty of having attackers which may not share their knowledge and may change their behaviour during the protocol run, depending on the risks of retaliation. In *BUG*, the Bad and Ugly peers operate with the full set of capabilities of a single Dolev–Yao attacker. *BUG* does not change the capabilities of the Dolev–Yao attacker, but changes their power by changing how collusion works.

A direct derivation from *BUG* is the *Rational Attacker* [4]. The *Rational Attacker* model drops the separate distinction of the agent types and simplifies *BUG* by considering that any principal makes a cost/benefit decision at any time whether to behave according to the protocol specification or not. The attacker must decide whether the gain outweighs the risks of being caught. The *Rational Attacker* was followed up by the *General Attacker* [4]. In this model, the cost/benefit function is removed, but any agent may still behave as a Dolev–Yao attacker. In fact each peer is a potential attacker that can use the information lawfully acquired to subvert the protocol. This threat model is more realistic than the *BUG* and *Rational Attacker* models in an Internet scenario and has the benefit of not having to deal with the gain/loss function.

Finally, there is a refinement of the *General Attacker* model, called *Multi-Attacker* [5]. In this model, each principal may behave as a Dolev–Yao attacker but will never reveal his long-term secrets to other agents. The *Multi-Attacker* adds some rationality to the *General Attacker* in a way that it avoids trivial impersonation attacks.

The *BUG* family of threat models can be seen as a relevant attempt to represent protocols' execution environments in a more accurate manner. In the human-centric security, and ubiquitous computing areas, we see works from Stajano [26], Balfanz [7] and Creese et al [15] as interesting starting points where we can see variations of the Dolev–Yao attacker applied to the analysis of human-interactive security protocols. Their models include two different communication channels to encompass different threats and environments. In general, they consider a traditional network channel that is susceptible to a Dolev–Yao attacker and a restricted channel where a weakened version of the attacker is used to represent the threat model involving human agents and devices. These examples are relevant because they introduce the notion that communication channels involving human peers and human communication need to be analysed against realistic attacker actions.

In our work, we will discuss how to modify the attacker capabilities to analyse security ceremonies under a realistic threat model. This realistic and dynamic threat model will enable us to devise stronger or weaker attackers (threat models) depending on the scenarios the ceremony is subject to. Although initially simple, such a threat model hides subtleties that can validate (or invalidate) claims regarding the achievement of security goals. Attacks can be seen as the marriage between weak goal achievements and the misunderstanding of the correct threat model, and therefore, the definition of a realistic and accurate threat model is extremely important.

4 Premises for ceremonies and ceremonies threat modelling

It seems obvious that developing a ceremony that is secure against a Dolev–Yao attacker will imply that the same ceremony will be secure against any weaker real-world attacker. However, it is often the case that, to guarantee that a certain ceremony is secure against such a powerful attacker, we have to include very complex mechanisms, which can often lead to the degradation of usability. By doing that, a new threat is introduced, which is the fact that the user will try to circumvent the security mechanisms in order to accomplish his/her tasks.

The premises presented below are derived from the work of Carlos and Price [14]. We especially took into account their thorough analysis of the human–protocol interaction and their recommendations for designing security ceremonies. We did not include all their recommendations, but only those that directly affect the composition of our proposed threat model.

If we consider a more realistic threat model, which might be vulnerable to a Dolev–Yao attacker, but is safe against a real-world attacker, we can prevent the user from being overloaded. As a consequence, we should be able to make the ceremony more usable, and as a result, secure. A realistic threat model would be a threat model that accurately represents the threat environment in which the ceremony will be used, and at the same time, requires only a set of peer-verifiable assumptions to be available during the execution.

One important premise for a reasonable threat model for security ceremonies involving human peers is that **no being is omnipotent in human-to-human channels**. This premise is easily verifiable by humans. The detection of powers beyond usual human capability is straightforward in the setting of security ceremonies. The impact of such a premise is that, depending on the situation, the presence of an active attacker is not realistic. We previously mentioned the example that replaying or blocking "speech" in a human-to-human channel will involve the use of powers that are not feasible for a human peer.

Following a similar pattern, we have a premise that **omnipotence in the human-to-device channel is not always realistic**. Although we have scenarios where we expect that an attacker has full control over the human-to-device channel (therefore a Dolev–Yao attacker), in some specific situations, such a powerful attacker does not represent reality. An example of such situation is a ceremony that makes use of single-purpose devices (e.g. one-time password generators). When these devices are used, the capabilities of the attacker over the human-to-device channel are limited. Thus, the threat model used on such channels is ceremony and context-specific.

Next, we have the premise that **a threat model including human peers should be constrained by the laws of physics**. It is unrealistic to assume an omnipresent attacker in human-to-human channels. The implications of such a premise will allow human peers to properly choose a location to execute their ceremonies, taking into account the verifiable presence of a potential attacker. This can be exemplified by a ceremony in a physical context where human peers have strict physical access control. A real-world example of this premise is the execution of security ceremonies for PKIs in safe rooms with strict physical and electromagnetic controls.

Another important premise for security ceremonies is that **humans are capable of performing basic information recall or mathematical operations**. Some security protocols and their related security ceremonies are designed to encompass unrealistic human capabilities regarding the recall of information or the execution of mathematical operations. In a realistic threat model, human peers are required to recall just fresh information and to execute basic mathematical operations. This premise impacts how the personification of the attacker in the human-to-human channel behaves.

Without support from a device, a human peer has limited memory and limited mathematical capabilities. The presence of external aids is detectable and can be used to verify expected behaviour. An example of such a premise is the verification of possession of a device in an authentication scenario to generate one-time passwords.

Finally, we have the premise that **one should never use more crypto than needed**.

Using more crypto than needed can potentially create usability problems, or introduce inaccurate assumptions about the human-to-device interaction expected in a real-world context. Although this is not a ceremony specific problem, as noted by Anderson and Needham [3], it could lead to problems in related security ceremonies. The addition of extra layers of crypto, which do not directly address threats specific to the expected threat model, may induce the user who is taking part in the ceremony to misunderstand the threat level to which they are exposed. An example of such an extra layer not addressing the threat model is the usage of one-time password devices by banks. Most banks let their costumers believe they are safer because they posses such tokens, when, in fact, the real threat models they are subject to allows the attacker to perform active man-in-the-middle attacks and not only password eavesdropping. The extra layer of crypto does not address the active man-in-the-middle attacks, but establishes a very strong device possession premise. The artefact protects from password reuse, but does not provide transaction authentication.

With the above premises in mind, we propose a threat model that encompasses the characteristics of each specific channel. For every channel, we start from the premises of Dolev–Yao, but we weaken the attacker to meet real-world conditions.

5 Proposed threat model for ceremonies

The proposition of a new threat model for security ceremonies is justified because no protocol is executed without context. It is known that even if a protocol is proven secure against a powerful attacker (e.g. Dolev–Yao), it might still fail, the reason for which may include:

– Usability problems: despite the fact that user interaction is usually part of a protocol's assumptions (and not an explicit part of the specification), in some cases, when these assumptions are implemented, they may require an unrealistic set of user capabilities to achieve the expected goals. Therefore, a user may not be able to perform their tasks correctly and/or not be able to execute them at all.
– The assumptions are too big/strong or generic: it is often necessary to assume that previous steps were successfully performed, or that the user is capable of performing some kind of operation. However, in some scenarios, converting an assumption into an implementation that achieves the same (expected) security properties might be extremely challenging. One example of such strong and generic assumptions is the identity guarantees yielded by digital certificates in SSL/TLS protocols. It is mostly true for certificates issued by Certification Authorities, but when the user is allowed to accept self-signed certificates, the assumption becomes weaker than necessary.

While those reasons are not directly related to the network channel, and therefore one could state that the protocol achieves its security goals, we cannot make the same statements when the protocol is implemented. When put in practice, assumptions that involve human-to-device and human-to-human interaction have to be implemented somehow. These assumptions must be replaced by dynamic user interactions. In doing so, we introduce two new possible communication channels. In this case, we cannot ensure that the expected security properties assumed in the protocol design will hold in the ceremony.

Another important issue regarding threat models for security ceremonies is the fact that humans make different decisions regarding their security, based on a dynamic evaluation of the environmental threat level to which they are exposed [23]. An example of such an embodied decision-making strategy is the evolutionary pressure humans suffered when considering the trade-offs of whether to engage in attacks to become hunters or to maintain a gatherer way of life, and thus be exposed to less risk [2]. This inherent faculty of human nature is usually not taken into account when we always assume the worst-case scenario as in a Dolev–Yao setting. Some attacks may be thwarted by using a very pessimistic threat model, but inherently this action may provoke human nature into acting and finding an easier and plausible solution if the user does not consider the alternative path as risky.

In taking the above into consideration, we stress that the threat model for a security ceremony must be adaptive. Even the same protocol might be used in the context of different threat models and achieve its goals in different but still reasonable ways. Considering the worst case is not always the best option as it degrades usability. The adaptive model we are proposing applies mostly to the human-to-device and human-to-human channels. For network communication (device-to-device channel), in the majority of cases, we will assume a Dolev–Yao attacker since it is the de facto standard. This is important since it is very well studied and developed.

In addition, having different threat models for different environments can potentially "teach" users to be more aware of the threats they face, and to better (intuitively) understand the threat model for each circumstance. For example, a user pairing two Bluetooth devices at home is subject to different threats than when at an airport. The same may be said for an ATM cash withdrawal ceremony differing between America and Europe, or even when the ceremony takes place in the same location but at different times. All of these scenarios can be subtly different.

Based on that, the most challenging step in designing and analysing ceremonies is to define the threats and the conditions under which the ceremonies will be used. A threat model for ceremonies must be ceremony and context-dependent. However, we can define a limited set of threats which encompass the great majority of those cases. The existence of a standardised threat model scenario is paramount to the establishment of security goals of ceremonies and for the comparison between the efficiency of different ceremonies.

Therefore, our proposal for a threat model for ceremonies is based on the set of capabilities of Dolev–Yao's attacker. We list and describe each capability, and we dynamically add and remove them from the threat model we define. For example, we might use a threat model where the attacker may have the *eavesdrop* and *initiate* capabilities only to enable the fulfilment of our premises. Our final goal is to measure the security of ceremonies against a realistic attacker, whose set of capabilities may be a subset of the abilities of Dolev–Yao's attacker. This approach will also help us reuse some of the abstract verification techniques and tools already in use for security protocols.

To describe our threat model approach, we first create a set of the potential capabilities of an attacker based on the Dolev–Yao model. By using this set and defining a threat model, we will be able to design and verify ceremonies that are secure against a realistic attacker with different capabilities on different channels (e.g. Dolev–Yao attacker on the network channel, while an attacker with *eavesdrop*, *initiate* and *block* exists on the human-to-device channel, and finally a passive attacker, who only *eavesdrops* on the human-to-human channel).

We will use a simple notation to describe the powers of the attacker on a specific channel. We begin with no threat model, or simply, a model where the attacker has "no capabilities". From that point on, we add the desired capabilities for each attacker. For example, E for *eavesdrop* only, or EB for *eavesdrop* and *block* only. Another way of approaching the notation is by presenting a weakened version of Dolev–Yao's attacker model instead of using a composition of capabilities. We will consider that "DY" is a Dolev–Yao attacker, and everything followed by a "–" symbol will represent the capabilities removed from this attacker. For example, a "DY–BR" means a Dolev–Yao attacker without the *blocking* and *replaying* capabilities. We would like to stress that this is more a matter of notation, and if we use both strategies, we will always achieve equivalence. However, before making use of the secondary notation, we would need a formal and precise definition of Dolev–Yao's set of capabilities, which we will not include here. We will therefore use only the first type of notation in this paper.

Using the same initial assumptions of Dolev and Yao about the attacker—that is, that the attacker is a valid agent of the network and cannot perform cryptanalysis—we list below the attacker's set of capabilities which we include in our adaptive threat model. All the capabilities we list are based on Dolev–Yao's attacker, and their informal definitions are shown. We start with the definition for the Eavesdrop capability as shown below:

Definition 1 (Eavesdrop – E)

$$\forall X \in M.\ A \rightarrow B : X \Rightarrow X \in knows(I)$$

Our definition for the eavesdrop capability reads as follows: for all messages X in the set of messages M, if the agent A sends to B a message X, this implies that the intruder I will learn X. All the logical connectives have their usual meaning and the set knows (Y), represent the set of knowledge of an agent Y in the protocol.

Another important capability of a Dolev–Yao attacker is the capacity of initiating a new communication with another peer using the knowledge the attacker possesses. The Definition 2 describes such capability:

Definition 2 (Initiate – I)

$$\forall X \in knows(I).\ I \rightarrow B : X$$

Definition 2 reads as follows: for all messages X in the knowledge set of the intruder I, the intruder I can initiate a communication with a peer B and send message X.

Next, we define the capability that enables the attacker to break a message down into its atomic parts. This capability is relevant so that the intruder can use atomic components of previously learned messages to produce new ones. The definition for the Break Down capability is shown in Definition 3:

Definition 3 (Atomic Break Down – A)

$$\forall \{X, Y\} \in knows(I). \Rightarrow$$
$$\{X\} \in knows(I)\ \wedge\ \{Y\} \in knows(I)$$

The atomic break down definition is: for all pairs composed by some X and Y elements in the knowledge of the intruder I, the element X and Y are also in the knowledge of the intruder I individually.

The cryptographic capabilities of the attacker are presented in the Definition 4:

Definition 4 (Crypto – C)

$$\forall \{X\}_k \in M \wedge\ k \in knows(I).\ A \rightarrow B :$$
$$\{X\}k \Rightarrow X \in knows(I)$$

This definition is described as: for all X ciphered using a key k in the set of messages M, and the key k is in the knowledge of the intruder I, if an agent A sends a message X encrypted with the key k to an agent B, the unencrypted version of X will also be in the knowledge of the intruder I.

The attacker capability of blocking messages, that is, preventing the receiver from learning the contents of a message sent to them, is presented in the Definition 5:

Definition 5 (Block – B)

$$\forall X \in M.\ A \rightarrow B : X \Rightarrow X \notin knows(B)$$

This definition is read as: for all messages X in the set of possible messages M, if an agent A sends a message X to agent B, the agent B will not have the message X in its set of knowledge $knows(B)$.

An important capability of the attacker is the use of a publicly known function to fabricate new messages. Examples of such functions can be cryptographic hashes, public-key encryption or any other function publicly available within the ceremony. Fabricate may be an n-ary function, differently than presented below. The capability of fabricating messages is presented in the Definition 6:

Definition 6 (Fabricate – F)

$$\forall X \in knows(I) \Rightarrow F(X) \in knows(I)$$

This definition is read as: for all messages X in the set of knowledge of the intruder I, the result of the application of a publicly available function $F()$ is also in the knowledge set of the intruder I.

Spoofing messages is an attacker's capability where he is able to send a message to an agent pretending to be some other agent. The definition of spoof is presented below:

Definition 7 (Spoof – S)

$$\forall X \in knows(I).\ Spoof(I, A) \rightarrow B : X$$

This definition is read as: for all messages X in the knowledge of the intruder I, an intruder I can spoof the identity of an agent A to the agent B and send a message X. Spoof differentiates from initiate in deliberately not allowing the attacker to be an internal agent in the execution of the ceremony.

The final capability that we can selectively detach from the original Dolev–Yao attacker model is the capability of re-ordering messages, as in Definition 8:

Definition 8 (re-Order – O)

$$\forall X, Y \in M.\ A \rightarrow B : X\ \wedge\ C \rightarrow B : Y$$
$$\Rightarrow Y \in knows(B) \wedge \cdots \wedge X \in knows(B)$$

It reads as: for all messages X and Y in the set of possible messages M, so that A first sends to B the message X, and at a later point in time C sends the message Y to B, it will imply that the attacker can add those messages to B's set of knowledge in a different order to the one in which they were sent, and possibly with some other messages in between. An important aspect of this capability is that it is described from the receiver's point of view, since there are many different ways in which the intruder may achieve this re-ordering. We deliberately show this on Definition 8 by using two different senders. Nevertheless, this can also be achieved with $A = C$.

Some of the well-known attacker capabilities (based on the Dolev–Yao threat model) are not directly shown here, since they can be achieved by the combination of our definitions. For example, the capability of **Modifying (M)** messages on the communication channels can be defined as the use of **Block** + textbfInitiate or more precisely **Block** + **Fabricate**, while **Replaying (R)** messages can be represented as **Eavesdrop** + **Initiate**. It is important to note that the list we present is necessarily incomplete, simply because new attacks and/or attacking tactics can be discovered or exist under different scenarios that we do not initially cover. Nevertheless, our dynamic and adaptive threat model inherently allows additions to the set of attacker's capabilities.

The adaptive threat model we propose here can be applied to all the communication channels in a ceremony. Although we recommend that the device-to-device channel should be verified in the majority of cases against a Dolev–Yao attacker, it is important to always consider whether the Dolev–Yao attacker is realistic for every channel, even for device-to-device communication.

6 Bluetooth pairing ceremony

Bluetooth is a short-range communications system intended to replace the cables connecting portable and/or fixed electronic devices [12]. The establishment of such communication is temporary in nature, created for data exchange between the devices. Bluetooth has been designed by focusing on robustness, low power consumption and low cost.

Bluetooth devices work in two modes of operation, namely discoverable and non-discoverable. When operating in discoverable mode, the device responds to enquiries made by other (unknown) devices. On the other hand, when in non-discoverable mode, a device only responds to enquiries from devices with whom it has previously set up communication.

When two devices are communicating for the first time, they do not have a common link key. Therefore, a key is created. To create a new key, a procedure called "pairing" is used. There are two procedures for pairing. The first is the legacy pairing protocol and the second is secure simple pairing [12].

The legacy pairing protocol, in use from Bluetooth versions 1.0–2.0 (known as **legacy pairing**), uses a user input to establish the connection. Users of both devices in the pairing protocol are asked to type a conventional PIN that is used as part of connection establishment. For devices with limited input capabilities (e.g. headsets), a fixed PIN is used (e.g. 0000), whereas for advanced devices, such as mobile phones or computers, a numeric or alphanumeric PIN is used. Usually, at least one of the devices executing the Bluetooth protocol will have some sort of input capability, so that the pairing with a chosen device can be achieved.

For recent versions of Bluetooth, a different pairing mechanism is defined. This new pairing procedure is called "Secure Simple Pairing" (SSP) and is used from version 2.1 + EDR (Enhanced Data Rate) onwards. SSP was designed to solve several problems found in earlier versions of the pairing protocol. Firstly, it simplifies the pairing process from the user's point of view, offering different pairing options and requiring fewer and simpler interactions. In addition to the usability improvements, it adds increased protection against passive (eavesdropping) and active (man-in-the-middle) attacks. This additional protection solves flaws found on earlier versions that allowed attackers to deploy man-in-the-middle (MITM) attacks [12,13,21,25].

SSP presents four distinct association modes designed to cover most device types. The **just works** model is designed for devices with limited display and input capabilities, possibly without them altogether. In this mode, the associating devices exchange public keys, nonce and a commitment value but nothing is displayed to the user, except in some implementations where the user might be asked whether to accept the connection or not. The **numeric comparison** association mode, which is our focus in this paper, makes use of the same protocol as the just works mode. The difference between them is that the numeric comparison mode is designed for devices capable of displaying digits (a six digit number) and accepting user inputs ("yes" or "no"). Because of the additional capability of the device, the protocol includes an additional authentication step, which is performed by the user. Both devices display a number (based on the nonce and public keys shared between the devices), and the users have to check whether the numbers shown are the same on both devices. If they are the same, the pairing is successful. The third association mode is **out of band (OOB)**, and it is designed for scenarios where an out of band mechanism is used for discovering devices as well as exchanging the cryptographic information required for the pairing process. The last mode is the **passkey entry**, which is designed for situations where the pairing devices have different input and display capabilities; for example, where one device has only input capabilities, such as a keyboard, the other has only display capabilities, such as a screen [13].

The SSP protocol is divided into five phases, and phases one, three, four and five are the same for all modes/protocols. Phase two, which is most important stage for the authentication process between the devices, is unique for each association mode used. Our focus in this paper is on the SSP protocol under the numeric comparison (NC) mode. Therefore, we will concentrate on phase two of the SSP protocol using the NC mode and assume that all other phases are correct.

Phase two is run as soon as phase one is concluded. In phase one, the initiating device (A) sends its public key (PK_a) to the non-initiating device (B). Next, B replies to A, sending its public key (PK_b), and phase one is finished. Phase 2 under the NC mode, considering that N_a and N_b are nonces and the function $f1$ is a function to generate the 128-bit commitment value C_b, is described as follows:

1. $B \longrightarrow A : C_b = f1(PK_B, PK_A, N_b, 0)$
2. $A \longrightarrow B : N_a$
3. $B \longrightarrow A : N_b$

There are only 3 steps in phase two in the NC mode. However, there is one very important step missing in the description above. After message 3, two confirmation values are calculated by the devices A and B. These values are V_a and V_b which are generated via a known function g using the values PK_A, PK_B, N_a, N_b as parameters which were shared between the devices during the previous protocol messages. V_a and V_b are values that will be shown on the device A and B displays, respectively. Finally, the users of these two devices will then compare whether V_a and V_b are the same and confirm by (usually) pressing a button on the device. Without this user verification of V_a and V_b, this protocol is vulnerable to MITM attacks.

As we can see, there is more involved in this protocol than a classic protocol analysis is able to cover. There is human verification and interaction with the devices and humans communicating with each other. This makes the SSP protocol (and even the legacy mode) an excellent case study for ceremony analysis.

When described as a ceremony, phase two of SSP includes two more agents U_A and U_B, and two additional communication channels HD and HH, to represent the human-to-device channel and human-to-human channel, respectively (in this ceremony, we will identify the network channel used in the protocol description as the DD—device-to-device—channel).

Thus, phase two of SSP described using the NC mode, as a ceremony, is described as follows:

$$\begin{aligned}
&M_1.B \xrightarrow[DD]{} A: && C_b = f1(PK_B, PK_A, N_b, 0)\\
&M_2.A \xrightarrow[DD]{} B: && N_a\\
&M_3.B \xrightarrow[DD]{} A: && N_b\\
&M_4.A \xrightarrow[HD]{} U_A: && V_a = g(PK_A, PK_B, N_a, N_b)\\
&M_5.B \xrightarrow[HD]{} U_B: && V_b = g(PK_A, PK_B, N_a, N_b)\\
&M_6.U_A \xrightarrow[HH]{} U_B: && V_a\\
&M_7.U_B \xrightarrow[HH]{} U_A: && V_b
\end{aligned}$$

When compared to the protocol description, we have four additional steps in the ceremony version. These steps are M_4 that represents the device A sending the value V_a to the user U_A; M_5 that represents the device B sending the value V_b to the user U_B; M_6 that represents the user U_A sending the value V_a to the user U_B; and finally M_7 that represents the user U_B sending the value V_b to the user U_B. It is important to notice that in our case study, we are assuming two humans pairing two devices. The main reason for using this assumption is that it allows us to introduce the idea of the HH channel in a realistic scenario. However, there is also the possibility of a single human pairing two devices, which would remove the need for the HH channel. In that case, we would just assume that messages sent by both devices via the HD channel would be sent directly to the same human who would then compare their values.

By adding these four additional steps, the pairing ceremony encompasses the whole pairing process and there is nothing left uncovered. Furthermore, when analysing the specifications, we find that both legacy mode and SSP are designed under assumptions that the attacker possesses a restricted set of capabilities during the pairing process. Usually, protocols are designed and analysed against a very powerful attacker model defined by Dolev and Yao [17].

In this paper, we will formally analyse SSP using the NC mode against a Dolev–Yao attacker, and then against less powerful (but more realistic) variations.

7 Example scenario: bluetooth pairing protocol

In our analysis, we consider the ceremony using the **numeric comparison** (NumComp) mode under different variations of the threat model. The theorems below present the results of our analysis.

Theorem 1 **(NumComp+DY)** *If the protocol messages M_1 to M_7 are run against a DY attacker, the attacker can prevent U_A from learning V_a or V_b and U_B from learning V_b or V_a, forcing them to learn V_i instead.*

$$\frac{M_{1\ldots7} \cup DY}{\begin{aligned} V_a \wedge V_b \wedge V_i \in knows(I) \wedge \\ V_a \notin knows(A) \wedge V_b \notin knows(A) \wedge \\ V_b \notin knows(B) \wedge V_a \notin knows(B) \wedge \\ V_i \in knows(U_A) \wedge V_i \in knows(U_B) \end{aligned}}$$

Proof Due to space limitations, we will skip the initial steps of the proof and assume that the intruder I, acting as a man-in-the-middle, initiated two simultaneous pairing sessions with A and B during messages M_1 to M_3. The authentication from A to B starts in M_4 where the value V_a is sent to U_A. The equivalent message from B to U_B occurs in M_5. A DY intruder I, by using his block (B) and initiate (I) capabilities, can prevent message M_4 and M_5 from being delivered to U_A and U_B, respectively, and instead, send them any chosen value V_i. In M_6 and M_7, A and B would complete the protocol by sending V_i to each other, successfully concluding the pairing and allowing the man-in-the-middle attack to be deployed.

Using an alternative threat model, which we term the Adaptive Threat Model V1, we assume the attacker can only eavesdrop the HD channel. In the specific case of the Bluetooth pairing, the assumption is that the device is free from malware and the display is presenting the correct information.

Theorem 2 **(NumComp + Ad. Threat Model V1)** *If the protocol messages M_1 to M_3 are run against a DY attacker; the messages M_4 to M_5 are run against an attacker with E capability only; and messages M_6 to M_7 are run against a DY attacker, the attacker can prevent U_A from learning V_b and U_B from learning V_a, forcing them to learn the repetition (replay) of V_a and V_b (respectively) instead.*

$$\frac{(M_{1\ldots3} \cup DY) \wedge (M_{4\ldots5} \cup E) \wedge)(M_{6\ldots7} \cup DY)}{V_a \wedge V_b \in knows(I) \wedge V_a \notin knows(B) \wedge V_b \notin knows(A)}$$

Proof Again, due to space limitations, we will skip the initial steps of the proof and assume that the intruder I, acting as a man-in-the-middle initiated two parallel pairing sessions with A and B during messages M_1 to M_3. The authentication from A to B starts in M_4 where the value V_a is sent to U_A. The equivalent message from B to U_B occurs in M_5. In this case, an intruder with only E capability can only learn the values V_a and V_b. In M_6 and M_7, A and B complete the protocol by sending V_a and V_b, respectively, to each other. A DY attacker in messages M_6 and M_7 can perform a similar attack to the one described in Theorem 1. By preventing V_a from being delivered from U_A to U_B in M_6 and V_b from U_B to U_A in M_7, and then replaying the values V_b and V_a (respectively) instead (by making use of Blocking, Replaying and Spoof capabilities), the protocol run would be successfully concluded and would allow a man-in-the-middle attack to be deployed.

Finally, when using an alternative threat model, which we call the Adaptive Threat Model V2, the attacker can eavesdrop both the HD and the HH channels, that is, the attacker can eavesdrop on any communications in the pairing process that involve the humans.

Theorem 3 (NumComp + Ad. Threat Model V2) *If the protocol messages M_1 to M_3 are run against a DY attacker and the messages M_4 to M_7 are run against an attacker with E capability only the attacker cannot produce any relevant attack.*

$$\frac{(M_{1\ldots3} \cup DY) \wedge (M_{4\ldots7} \cup E)}{\emptyset}$$

Proof Once again, due to space limitations, we will skip the initial steps of the proof and assume that the intruder I, acting as a man-in-the-middle, initiated two parallel pairing sessions with A and B during messages M_1 to M_3. The authentication from A to B starts in M_4 where the value V_a is sent to U_A. The equivalent message from B to U_B occurs in M_5. In this case, an intruder with only E capability can only learn the values V_a and V_b. In M_6 and M_7, A and B complete the protocol by sending V_a and V_b, respectively, to each other. In messages M_6 and M_7, an intruder with only E capability can only learn the values V_a and V_b, and in this case, V_a received by U_B in M_6 and V_b received by U_A in M_7 would not match the V_b and V_a in $knows(B)$ and $knows(A)$, respectively, not allowing the attack to succeed.

Although the attack described in Theorem 1 is plausible in real-world scenarios, it is very difficult to deploy. An attacker would have to corrupt both devices as well as start parallel sessions with both users during a short period of time. By removing capabilities B and I from the attacker, we can analyse the protocol further, and possibly find other (more) relevant attacks.

The second attack, found in Theorem 2, is completely unrealistic. To be deployed in practice, the attacker would have to block communication between two humans and then replay some data over a channel where the user would easily notice whether some other party wanted to spoof the identity of the sender. In this case, the attack does not exist in practice.

The idea is similar for the other association modes. Each one of them must consider the real-world scenario and define the threat model. In addition to that, it should not be possible to use an association mode under a different threat model to the one specified.

8 Specifying the ceremony

The processes of formalisation and verifications were carried out in first-order logic (FOL) with the help of the automated theorem prover SPASS version 3.5 [27]. We chose to use first-order logic because of its automation and completeness. Basically, our formalisation follows the same structure as the one presented by Weidenbach [28], with the addition of logic elements particular to our protocol and ceremony. Furthermore, the main difference between Weidenbach's work and ours relies on an extended version of threat models as well as on an extra number of communication channels.

The formalisation process consisted of translating all the steps of the Bluetooth protocol into FOL formulae. A formula in this logic system makes use of the following elements: quantifiers, predicates, functions and connectives. Quantification in FOL only occurs over functions of 0-arity (constants). Predicates return a Boolean value, either $true$ or $false$, whereas functions return an element from the universe of discourse.

Throughout the formalisation, we adopt the following syntax: quantifiers (∀ – "for all" and ∃ – "there exists"); connectives (¬ – "not", ∧ – "and", ∨ – "or", $\implies$ – "implies", ⇔ – "if and only if", and = – "equality"). In addition, predicate names start with an upper-case letter and function names start with a lower-case letter. Names of variables always start with an "x". Parentheses define the relationship between elements and brackets limit quantifiers' scope.

Before moving to describe the protocol in logic formulae, we must firstly explain the meaning of each predicate and function used. This is a useful step towards a better understanding of our formalisation.

- **Agent predicates.** These predicates define what are agents and subclasses of agents. For example: $Agent(x)$ (x is an ordinary agent), $Honest(x)$ (x is an honest agent) and $Initiator(x)$ (x is an agent that may start the protocol execution).
- **Knowledge predicate.** This predicate indicates the knowledge belonging to a certain agent. That is, $Knows(x, y)$ (the knowledge x belongs to agent y).
- **Communication channels predicates.** The communication channels used in our threat model. They are: $MDD(x)$ (for communication between devices), $MHD(x)$ (for communication between a human user and a device or vice-versa) and $MHH(x)$ (for communication between humans). In all these predicates, the variable x refers to the message being exchanged on the channel.
- **Message exchange function.** this function indicates a message exchange operation between two agents: $sent(x, y, z)$. The variable x points to the agent that sends the message (the issuer) and variable y points to the agent to which the message is being sent (the receiver). The message content is represented by variable z. Moreover, this function should be used together with the predicates of communication channels.
- **Cryptographic key functions.** these functions establish possession relationships between cryptographic mater-

ial and their corresponding owner agents. For example: $krkey(x, y)$ (private keys), $kukey(x, y)$ (public keys) and $nonce(x, y)$ (nonces). In these functions, the cryptographic material x belongs to the agent y. Also, there is a function intended to group a private key with its corresponding public key in a key pair: $kp(x, y)$.
- **Human agent function.** This function returns the human user associated with a certain agent $x : human(x)$.
- **Bluetooth specific functions.** these functions represent the Bluetooth protocol's internal cryptographic functions: $c\_f1(x, y, z, x1)$ (for function $C_b = f1(\ldots)$) and $v\_g(x, y, z, x1, x2)$ (for function $V_a = g(\ldots)$). In function $v\_g(\ldots)$, the additional fifth variable refers to the agent who created V. We will provide further explanation of this addition to the specification. Functions C_b and V_a are formalised together to $f1$ and g because, otherwise, it would bring unnecessary redundancy to our logic problem. We recommend reading Bella's Goal Availability principle for a detailed explanation of this redundancy elimination technique [10].
- **Agents constants.** The agents themselves. For example: a and b (for the first and second agents, respectively).
- **Keys constants.** The sensitive material (private keys, public keys and nonces). Respectively, kra, kua, and na (for agent a); and krb, kub and nb (for agent b).
- **Connection results constants.** The final result of a Bluetooth pairing: ok (in case of success) and $fail$ (in case of failure).
- **The zero constant.** This number is used in the calculation of function C_b (see Sect. 6).

We would like to note the formulas presented below are a direct translation from our SPASS implementation. Thus, some redundancies in pre-conditions are present to help SPASS to reason faster. This was done for the sake of fidelity with our experiments.

Finally, the whole logic problem is modelled. It is divided into three sub-models: *Preliminary Operations*, *The Main Logic Model*, and *The Intruder Model*. The first sub-model, *Preliminary Operations*, contains basic formulae available throughout the specification. Next, the second sub-model, *Main Logic Model*, formalises the Bluetooth protocol's message flow. The last sub-model, *Intruder Model*, formalises the operations that an arbitrary intruder (attacker) can make use of in their attempt to compromise the protocol.

8.1 Preliminary operations

This sub-model defines general-purpose operations available throughout the protocol. These operations refer to the relationship between agents, cryptographic keys and communication channels. Their main objective is to avoid needless repetition of conditions and in the formulae of the main logic model.

Formula 1 states that if something is an agent, its corresponding human being will also be an agent. The same follows for honest agents in Formula 2.

1. $\forall xa[$
   $Agent(xa) \implies Agent(human(xa))]$
2. $\forall xa[$
   $Honest(xa) \implies Honest(human(xa))]$

We also specify two formulae for dealing with keys. Formula 3 states that, if a certain agent has a public-private key pair in his knowledge, then he will also have each key in his knowledge individually. The inverse is also possible under the equivalence operator ($\Leftrightarrow$). In addition, Formula 4 states that the order of keys in a key pair does not matter.

3. $\forall xkukey, xkrkey, xagent[$
   $Agent(xagent) \implies$
   $(Knows(kp(kukey(xkukey, xagent),$
   $krkey(xkrkey, xagent)), xagent) \Leftrightarrow$
   $(Knows(kukey(xkukey, xagent), xagent) \wedge$
   $Knows(krkey(xkrkey, xagent), xagent))))]$
4. $\forall xkukey, xkrkey, xagent[$
   $Agent(xagent) \implies$
   $(Knows(kp(kukey(xkukey, xagent),$
   $krkey(xkrkey, xagent)), xagent) \Leftrightarrow$
   $Knows(kp(krkey(xkrkey, xagent),$
   $kukey(xkukey, xagent)), xagent)))]$

When a message is sent in the protocol, we assume it will always arrive at the intended recipient. Therefore, the message receiver will automatically be able to put it in his knowledge set. Formulae 5, 6 and 7 represents this behaviour for channels DD (device–device), HD (device–human) and HH (human–human), respectively. Later, we will formalise situations in which the recipient is prevented from receiving the message through a block operation made by the attacker.

In this context, it is worth noting that we cannot make any assumption about the message's issuer. An intruder impersonating an honest agent might have sent the message. In this circumstance, what the receiver sees is the honest agent as the message author, not the intruder himself. Thus, we cannot enforce knowledge of the issuer when a message is sent, otherwise we would inadvertently break the protocol structure. Nevertheless, under our basic logic framework, an agent will only be able to engage in creating a message using components he already possesses.

5. $\forall xa, xb, xm[$
   $MDD(sent(xa, xb, xm)) \implies$
   $Knows(xm, xb)$
6. $\forall xa, xb, xm[$
   $MHD(sent(xa, xb, xm)) \implies$
   $Knows(xm, xb)]$
7. $\forall xa, xb, xm[$
   $MHH(sent(xa, xb, xm)) \implies$
   $Knows(xm, xb)]$

8.2 Main logic model

The second sub-model in our logic problem involves the description of the Bluetooth ceremony. To do so, we firstly formalise the agents and their initial knowledge bases. Then, we formalise the Bluetooth protocol and ceremony.

In Formula 5, we introduce the first agent in the protocol (Agent A). That is, the constant a is introduced as an agent. It is also an honest agent (Formula 6) and it may initiate the protocol (Formula 7). In its initial knowledge base, there is the possession of its keypair and nonce (specified by Formulae 8 and 9). Formulae 10–12 establish equality between key constants and their corresponding key functions.

5. $Agent(a)$
6. $Honest(a)$
7. $Initiator(a)$
8. $Knows(kp(krkey(kra, a), kukey(kua, a)), a)$
9. $Knows(nonce(na, a), a)$
10. $kra = krkey(kra, a)$
11. $kua = kukey(kua, a)$
12. $na = nonce(na, a)$

The second agent in the protocol is introduced in the same way (Formulae 13–19).

13. $Agent(b)$
14. $Honest(b)$
15. $Knows(kp(krkey(krb, b), kukey(kub, b)), b)$
16. $Knows(nonce(nb, b), b)$
17. $krb = krkey(krb, b)$
18. $kub = kukey(kub, b)$
19. $nb = nonce(nb, b)$

We now advance to the Bluetooth protocol specification. In general, our formal model follows a simple pattern. Each step in the protocol is represented by a single formula containing an implication ($\implies$), which means that its pre-conditions must be satisfied in order to have its post-conditions evaluated (causality). Pre-conditions are on the left side of the implication arrow, whereas post-conditions are on the right side. Usually, the pre-conditions involve checking what the agent has in its knowledge set at the current step. It is expected that the agent has accumulated knowledge during the protocol execution but has not created new knowledge inadvertently. To ensure an ordered execution, the pre-conditions test not only the $Knows$ predicate, but also the messages previously exchanged on the communication channel. That is, to acquire a certain message ($Knows$ predicate), the agent must have previously received it at some point during the execution.

In the first step (Formula 20), the following conditions must be observed. There must be two agents, where one of them must be allowed to initiate the protocol and this agent must also have his public key in its $Knows$ predicate. If all of these conditions are true, the initiating agent sends his public key over the DD channel to the other agent.

20. $\forall xa, xb, xkua[($
    $Agent(xa) \wedge Agent(xb) \wedge Initiator(xa) \wedge$
    $Knows(kukey(xkua, xa), xa))$
    $\implies MDD(sent(xa, xb, kukey(xkua, xa)))]$

In the second step (Formula 21), the sequence of events is basically the same, but with a few differences. Besides checking whether he is in possession of his own public key, the current agent also verifies whether he received the public key from the initiating agent in the previous step. This check is done on both the $Knows$ and MDD predicate.

21. $\forall xa, xb, xkua, xkub[($
    $Agent(xa) \wedge Agent(xb) \wedge Knows(kukey(xkub, xb),$
    $xb) \wedge Knows(kukey(xkua, xa), xb) \wedge$
    $MDD(sent(xa, xb, kukey(xkua, xa))))$
    $\implies MDD(sent(xb, xa, kukey(xkub, xb)))]$

The same agent who performed the second step performs the third step. To be able to generate C_b (represented logically as function $c\_f1(\ldots)$), the agent must have both public keys and his nonce in its predicate $Knows$.

22. $\forall xnb, xb, xkub, xkua, xa[($
    $Agent(xa) \wedge Agent(xb) \wedge Initiator(xa) \wedge$
    $Knows(nonce(xnb, xb), xb) \wedge$
    $Knows(kukey(xkub, xb), xb) \wedge$
    $Knows(kukey(xkua, xa), xb) \wedge$
    $Knows(kukey(xkub, xb), xa) \wedge$
    $MDD(sent(xb, xa, kukey(xkub, xb))))$
    $\implies MDD(sent(xb, xa, c\_f1(xkub, xkua, xnb, 0)))]$

The fourth step consists of, upon receipt of $C_b(c\_f1(\ldots))$, the initiator agent sends its nonce. Formula 23 illustrates this.

23. $\forall xna, xa, xb, xkub, xkua, xnb[($
$Agent(xa) \wedge Agent(xb) \wedge$
$Knows(nonce(xna, xa), xa) \wedge$
$Knows(kukey(xkua, xa), xa) \wedge$
$Knows(kukey(xkub, xb), xa) \wedge$
$Knows(c\_f1(xkub, xkua, xnb, 0), xa) \wedge$
$MDD(sent(xa, xb, kukey(xkua, xa))) \wedge$
$MDD(sent(xb, xa, c\_f1(xkub, xkua, xnb, 0))))$
$\Longrightarrow MDD(sent(xa, xb, nonce(xna, xa)))]$

In the fifth step (Formula 24), the second agent sends his own nonce at the moment it receives the nonce of the initiator. This is illustrated in Formula 24.

24. $\forall xnb, xb, xa, xna, xkua, xkub[($
$Agent(xa) \wedge Agent(xb) \wedge$
$Knows(nonce(xnb, xb), xb) \wedge$
$Knows(nonce(xna, xa), xb) \wedge$
$Knows(kukey(xkub, xb), xb) \wedge$
$Knows(kukey(xkua, xa), xb) \wedge$
$MDD(sent(xb, xa, kukey(xkub, xb))) \wedge$
$MDD(sent(xb, xa, c\_f1(xkub, xkua, xnb, 0))) \wedge$
$MDD(sent(xa, xb, nonce(xna, xa))))$
$\Longrightarrow MDD(sent(xb, xa, nonce(xnb, xb)))]$

In Formulae 25 and 26, agents forward to their corresponding human users the value of function V_x. A significant difference in these formulae is the use of the predicate MHD (channel human to device) and the function $human(x)$.

25. $\forall xa, xb, xkua, xkub, xna, xnb[($
$Agent(xa) \wedge Agent(xb) \wedge$
$Knows(nonce(xna, xa), xa) \wedge$
$Knows(nonce(xnb, xb), xa) \wedge$
$Knows(kukey(xkua, xa), xa) \wedge$
$Knows(kukey(xkub, xb), xa) \wedge$
$Knows(c\_f1(xkub, xkua, xnb, 0), xa) \wedge$
$MDD(sent(xb, xa, kukey(xkub, xb))) \wedge$
$MDD(sent(xb, xa, c\_f1(xkub, xkua, xnb, 0))) \wedge$
$MDD(sent(xb, xa, nonce(xnb, xb))))$
$\Longrightarrow MHD(sent(xa, human(xa), v\_g(xkua, xkub,$
$xna, xnb, xa)))]$

26. $\forall xa, xb, xna, xnb, xkua, xkub[($
$Agent(xa) \wedge Agent(xb) \wedge$
$Knows(nonce(xna, xa), xb) \wedge$
$Knows(nonce(xnb, xb), xb) \wedge$
$Knows(kukey(xkua, xa), xb) \wedge$
$Knows(kukey(xkub, xb), xb) \wedge$
$MDD(sent(xa, xb, kukey(xkua, xa))) \wedge$
$MDD(sent(xb, xa, c\_f1(xkub, xkua, xnb, 0))) \wedge$
$MDD(sent(xa, xb, nonce(xna, xa))) \wedge$
$MDD(sent(xb, xa, nonce(xnb, xb))))$
$\Longrightarrow MHD(sent(xb, human(xb), v\_g(xkua, xkub,$
$xna, xnb, xb)))]$

After receipt of V_x from their corresponding devices, each human user exchanges it with his counterpart. Formula 27 shows this in a general way (i.e. the formula does not take the agents order into account). The result of this formula is a message containing V_x to be sent over channel HH to the opposite human user.

27. $\forall xa, xb, xkua, xkub, xna, xnb, xag, xag2[($
$Agent(xa) \wedge Agent(xb) \wedge Agent(xag) \wedge Agent(xag2) \wedge$
$Knows(nonce(xna, xa), xag) \wedge$
$Knows(nonce(xnb, xb), xag) \wedge$
$Knows(kukey(xkua, xa), xag) \wedge$
$Knows(kukey(xkub, xb), xag) \wedge$
$Knows(v\_g(xkua, xkub, xna, xnb, xag),$
$human(xag)) \wedge$
$MHD(sent(xag, human(xag), v\_g(xkua, xkub, xna,$
$xnb, xag))))$
$\Longrightarrow MHH(sent(human(xag), human(xag2),$
$v\_g(xkua, xkub, xna, xnb, xag)))]$

V_x is a function that will take into account the four variables and generate a truncated number that is manageable for the human agent. Once human users have exchanged V_x between each other, they must verify its integrity. To do so, they compare the digits output by their own function $v\_g(\ldots)$ against the other digits received. The V-function's fifth variable is not compared because it is only used as a mark to indicate the agent who generated the V-function into consideration. This is just a technical necessity for the implementation and does not matter for the ceremony. Without this marking strategy, we would not be able to distinguish between two different Vs. In the case in which both values match, then a successful connection is established. Otherwise, something went wrong.

The integrity verification described above is shown in Formula 28. This formula can be divided into two parts. Part one (represented by the first implication) enforces the causality property seen so far in our logic model. This is done by checking whether the predicate MHH was used to exchange v_g between humans. Next, part two (represented by the second and third implications) tries to determine whether the Vs exchanged are equal or not. This is done by comparing each element that makes part of V_g. Here, the second and third implications make an if-then-else statement in FOL. In case the comparison is successful, the connection result *ok* is sent to the device over channel MHD. Otherwise, $fail$ is sent.

28. $\forall xa, xb, xkua1, xkua2, xkub1, xkub2, xna1, xna2,$
$xnb1, xnb2, xag[($
$Agent(xa) \wedge Agent(xb) \wedge$
$Knows(v\_g(xkua1, xkub1, xna1, xnb1, xb),$
$human(xa)) \wedge$
$MHH(sent(human(xb), human(xa),$
$v\_g(xkua1, xkub1, xna1, xnb1, xb))) \wedge$
$Knows(v\_g(xkua2, xkub2, xna2, xnb2, xa),$
$human(xb)) \wedge$
$MHH(sent(human(xa), human(xb),$
$v\_g(xkua2, xkub2, xna2, xnb2, xa))))$
$\Longrightarrow$
$((($
$(xkua1 = xkua2) \wedge$
$(xkub1 = xkub2) \wedge$
$(xna1 = xna2) \wedge$
$(xnb1 = xnb2))$
$\Longrightarrow$
$($
$MHD(sent(human(xa), xa, ok)) \wedge$
$MHD(sent(human(xb), xb, ok)))$
$\wedge\ ($
$\neg($
$(xkua1 = xkua2) \wedge$
$(xkub1 = xkub2) \wedge$
$(xna1 = xna2) \wedge$
$(xnb1 = xnb2))$
$\Longrightarrow$
$(MHD(sent(human(xa), xa, fail)) \wedge$
$MHD(sent(human(xb), xb, fail)))]$

The formulae that make part of our ceremony formalisation can be grouped in the set Γ_{main} as:

$$\Gamma_{main} = \{ \text{Formula } 1, \ldots, \text{Formula } 28 \}.$$

8.3 Intruder model

The intruder model (also known as the attacker model) is the set of formulae expressing actions an intruder can take in his attempt to break our ceremony. Some new logic elements are added to help us better describe the intruder's capabilities. They are the following:

- **Intruder predicate.** This predicate defines which elements are intruders: $Intruder(x)$. This predicate works in the same way as $Agent(x)$ and $Honest(x)$.
- **Block predicate.** This predicate is a support predicate for dealing with the intruder's capability of preventing a message from arriving at its intended receiver. In principle, blocking messages is an overly powerful capability. An intruder capable of blocking messages may be able to prevent the entire protocol or ceremony from executing in real life, but in our model, we must limit this in some way so that our solver can complete. Therefore, we must limit it in some way. We adopt the criteria that the intruder can block a maximum of one message for each agent in the whole logic problem. For this, we use predicate $BlocksAgentOnce(x, y)$ (x and y are, respectively, the message and agent blocked). Dealing with this attacker capability is rarely seen in protocol formalisations, since it will always lead to the detection of Denial of Service attacks.
- **Intruder agent constant.** As well as other agents, we created a constant to represent the intruder. In this case, i.
- **Intruder keys constants.** Intruder's cryptographic material: kri (private key), kui (public key) and ni (nonce). These are necessary so that the intruder is able to impersonate an honest agent in the protocol.

We want to emphasise the necessity of having a block predicate in our threat model. Our proposition was to be able to represent all the derivative properties of the Dolev–Yao model, so being able to perform at least some sort of blocking is important to create certain composite properties, such as Modifying.

In a similar way to other agents, we specify the intruder in Formulae 29–31. We highlight some key points: the intruder is considered an active agent in the ceremony (Formula 30); the human users corresponding to the intruder device is also considered an intruder (Formula 31); and the intruder's cryptographic material is defined in Formulae 32 and 33.

29. $Intruder(i)$
30. $Agent(i)$
31. $\forall xi[Intruder(xi) \implies Intruder(human(xi))]$
32. $Knows(kp(krkey(kri, i), kukey(kui, i)), i)$
33. $Knows(nonce(ni, i), i)$
34. $kri = krkey(kri, i)$
35. $kui = kukey(kui, i)$
36. $ni = nonce(ni, i)$

The most basic intruder capacities consist of eavesdropping and spoofing messages. Formulae 37–39 express the eavesdrop and Formulae 40–42 express the sending of faked messages (spoof). In both cases, the operations are replicated for each communication channel. These operations are carried out for honest agents only since it does not make much sense in our threat model that an intruder can eavesdrop and spoof against himself or other intruders.

37. $\forall xa, xb, xm[$
$((Honest(xa) \wedge Honest(xb) \wedge$

$MDD(sent(xa, xb, xm))$
$\implies Knows(xm, i)))]$

38. $\forall xa, xb, xm[$
$((Honest(xa) \wedge Honest(xb) \wedge$
$MHD(sent(xa, xb, xm))$
$\implies Knows(xm, i)))]$
39. $\forall xa, xb, xm[$
$((Honest(xa) \wedge Honest(xb) \wedge$
$MHH(sent(xa, xb, xm))$
$\implies Knows(xm, i)))]$
40. $\forall xa, xb, xm[$
$((Honest(xb) \wedge Knows(xm, i)$
$\implies MDD(sent(i, xb, xm))))]$
41. $\forall xa, xb, xm[$
$((Honest(xb) \wedge Knows(xm, i)$
$\implies MHD(sent(i, xb, xm))))]$
42. $\forall xa, xb, xm[$
$((Honest(xb) \wedge Knows(xm, i)$
$\implies MHH(sent(i, xb, xm))))]$

With the eavesdrop capacity set up, the intruder can accumulate knowledge over time. He can then try to break the Bluetooth computing functions C_x and V_x out of the eavesdropped messages. This is possible provided that he learns the sensitive cryptographic material (keys and nonces) exchanged between honest agents. Formulae 43 and 44 illustrates the fabrication of C_x and V_x, respectively. Furthermore, the intruder can use his own keys and nonces as inputs to these formulae.

43. $\forall xkua, xkub, xnb, xa, xb[$
$($
$Agent(xa) \wedge Initiator(xa) \wedge Agent(xb) \wedge$
$\neg Initiator(xb) \wedge Knows(kukey(xkua, xa), i) \wedge$
$Knows(kukey(xkub, xb), i) \wedge$
$Knows(nonce(xnb, xb), i))$
$\implies Knows(c\_f1(xkub, xkua, xnb, 0), i)]$

44. $\forall xkua, xkub, xna, xnb, xa, xb, xag[$
$($
$Agent(xa) \wedge Agent(xb) \wedge Agent(xag) \wedge$
$Knows(kukey(xkua, xa), i) \wedge$
$Knows(kukey(xkub, xb), i) \wedge$
$Knows(nonce(xna, xa), i) \wedge$
$Knows(nonce(xnb, xb), i))$
$\implies Knows(v\_g(xkua, xkub, xna, xnb, xag), i)]$

Formula 45 explicitly defines that an intruder can initiate a message exchange with an honest agent over channel HD.

45. $\forall xb, xm[$
$($
$Honest(xb) \wedge Knows(xm, i))$
$\implies (MHD(sent(i, human(xb), xm)) \wedge Initiator(i))]$

As seen before, we limit the number of blocking operations that an intruder can perform to one block per honest agent only. We use the $BlocksAgentOnce(x, y)$ predicate to do so. Formulae 46–47 illustrates the blocking operations for channels HD and HH.

46. $\forall xa, xb, xm[$
$($
$Honest(xa) \wedge Honest(xb) \wedge MHD(sent(xa,xb,xm)) \wedge$
$\neg\exists xblock[BlocksAgentOnce(xblock, xb)])$
$\implies (\neg Knows(xm, xb) \wedge BlocksAgentOnce$
$(xm, xb))]$
47. $\forall xa, xb, xm[$
$($
$Honest(xa) \wedge Honest(xb) \wedge MHH(sent(xa,xb,xm)) \wedge$
$\neg\exists xblock[BlocksAgentOnce(xblock, xb)])$
$\implies (\neg Knows(xm, xb) \wedge BlocksAgentOnce$
$(xm, xb))]$

The intruder specification ends here. The formulae used in the intruder model can be grouped into the set $\Gamma_{intruder}$ as.

$$\Gamma_{intruder} = \{ \text{Formula } 29, \ldots, \text{Formula } 47 \}.$$

Finally, we also define the set $\Gamma_{extended}$ containing our whole logic model. It is represented as a union of all the two previous sets.

$$\Gamma_{extended} = \{\Gamma_{main} \cup \Gamma_{intruder}\}.$$

We acknowledge that representing this problem with a more expressive formalisation such as HOL would enrich the details we were able to prove. Nevertheless, the effort in constructing proofs in HOL outweigh its benefits at the moment. We would also like to note that converting the ceremony to the formulae below is a laborious and error-prone work. In the future, we plan to automate such transcription tasks as well as the use of a more expressive formalism.

9 Results

In this section, we show the main results of our formalisation. Some of these results are merely informational, serving to assure that the protocol and ceremony were specified correctly; others are full proofs for theorems.

To find a proof, SPASS negates the conjecture (refutation), and then, it exhaustively generates all possible clauses. Due to space limitations, the output generated by SPASS is not included in this paper.

9.1 Saturation of main logic model

The first fact obtained in the formal analysis is the saturation of our main logic model (Γ_{main}). This is achieved by running the problem without conjectures. If it terminates, it means that SPASS can derive a finite set of clauses from it. Otherwise, if running the problem without conjectures would cause the theorem prover to run indefinitely, this would mean that it was undecidable due to the nature of first-order logic. Saturation is a strategy for enumerating the unsatisfiability problem in FOL, squatting it into the decidable portion of the logic.

We must keep in mind that saturation is not straightforward. During the formalisation process, we faced several issues: for example, SPASS entering in infinite loops, deriving invalid clauses from the practical viewpoint. We had to continually refine our logic description until we arrived at its current version. We would also like to emphasise that although the model saturates, it represents potentially infinite parallel executions with a potentially infinite set of agents engaging on different runs.

9.2 Normal execution

To evaluate the correctness of our specification, we tested a conjecture to verify whether the main logic model runs as specified; in other words, its last step (represented by Formula 28) is reachable by the derivations embedded in the model. This conjecture basically tests whether both honest agents (a and b) have the *ok* connection confirmation in their $Knows$ predicate. That is:

$$\Gamma_{main} \vdash [Knows(ok, a) \wedge Knows(ok, b)]$$

The proof analysis shows us that all formulae are used in the proof. This is of extreme importance because no steps should be left out. This theorem also concerns our description of the Bluetooth NC SSP as a ceremony. It demonstrates that our multi-media communication scheme is able to distribute knowledge properly to different types of peers during protocol execution.

9.3 Abnormal execution attempt

Alternatively, we ran a conjecture for the hypothetical scenario of agents being unable to establish a connection on normal circumstances (no intruder present). That is:

$$\Gamma_{main} \nvdash [Knows(fail, a) \vee Knows(fail, b)]$$

The execution of this conjecture only causes SPASS to saturate the problem, and it is unable to find any proof. This demonstrates our logic model follows the expected behaviour. This result further assures the correctness of our specification. This is the expected behaviour as the protocol should not fail without the interference of an intruder.

We now begin to examine the logic model in conjunction with an intruder. This will help us to further elaborate on the specifications, since the ceremony will be tested against a proper threat model.

9.4 Saturation of extended logic model

The extended logic model, combining the main logic model plus the intruder model, also saturates. Thus, SPASS can also derive a finite set of clauses from the larger problem. Due to the considerations of the attacker capabilities we now have the ceremony executing in its threat model environment.

$$\Gamma_{extended} \vdash [Knows(ok, a) \wedge Knows(ok, b)]$$

One important thing to note here is that the size of the problem increases dramatically by adding the threat model. The number of clauses deviated by SPASS increases significantly. The saturation here also demonstrates that our implementation works. During the validation phase, we proved some smaller lemmas regarding the capabilities embedded in the threat model. We took special care to verify how knowledge was being distributed to peers across the different media.

9.5 Proof of Theorem 1

The first theorem involves executing the whole model under a Dolev–Yao threat model. Its original description is:

"If the protocol messages M_1 to M_7 are run against a DY attacker, the attacker can prevent U_A from learning V_a or V_b and U_B from learning V_b or V_a, forcing them to learn V_i instead".

In logic notation, we have:

$$\begin{gathered}\Gamma_{extended} \vdash [Knows(v\_g(kua, kub, na, nb, a), i) \wedge \\ Knows(v\_g(kua, kub, na, nb, b), i) \wedge \\ Knows(v\_g(kua, kui, na, ni, i), i) \wedge \\ Knows(v\_g(kui, kub, ni, nb, i), i) \wedge \\ \neg Knows(v\_g(kua, kub, na, nb, a), b) \wedge \\ \neg Knows(v\_g(kua, kub, na, nb, b), a) \wedge \\ Knows(v\_g(kua, kui, na, ni, i), human(a)) \wedge \\ Knows(v\_g(kui, kub, ni, nb, i), human(b))]\end{gathered}$$

The conjecture above strictly lists which V_i we want the intruder to fake. Formula 44 says that the intruder can try any possible combination of public keys and nonces in order to generate some V. Therefore, we must restrict the possible values of V_i to $v\_g(kua, kui, na, ni, i)$ and

$v\_g(kui, kub, ni, nb, i)$. In the first case, a initiates a connection with i and, in the second case, i initiates a connection with b. The proof of this theorem is automatically checked using the theorem prover SPASS.

An important result is shown here: the Bluetooth Pairing Protocol in Numeric Comparison mode is insecure when analysed under a Dolev–Yao threat model. This also happens for the related ceremony as shown above.

9.6 Proof of Theorem 2

The second theorem tests what happens if we restrict the threat model in channel HD by only allowing eavesdropping. Its original description is:

"If the protocol messages M_1 to M_3 are run against a DY attacker; the messages M_4 to M_5 are run against a N+E attacker; and messages M_6 to M_7 are run against a DY attacker, the attacker can prevent U_A from learning V_b and U_B from learning V_a, forcing them to learn the repetition (replay) of V_a and V_b (respectively) instead".

To do so, we must remove Formulae 41 and 46 (Block and Spoof in channel HD, respectively). The channels DD and HH continue using a Dolev–Yao threat model. In logic notation, we have:

$$\begin{gathered}\Gamma_{extended} - \{\text{ Formula } 41, \text{ Formula } 46 \\ \} \vdash [Knows(v\_g(kua, kub, na, nb, a), i) \wedge \\ Knows(v\_g(kua, kub, na, nb, b), i) \wedge \\ \neg Knows(v\_g(kua, kub, na, nb, a), b) \wedge \\ \neg Knows(v\_g(kua, kub, na, nb, b), a)]\end{gathered}$$

The theorem above shows us a nuance of the threat model which the Bluetooth SPP ceremony is exposed to. When removing the block and spoof capabilities of the attacker in the HD channel only, we still find unrealistic attacks against the ceremony. We were able to prove that the attacker would be able to break the ceremony attacking the HH channel, using his blocking and spoof capabilities. This is an example of an attack that would surface using the full Dolev–Yao capabilities, but that in a real-world setting is unrealistic. As we demonstrate, our specification and verification framework is capable of capturing this nuance.

9.7 Proof of Theorem 3

The last theorem has a similar purpose to the previous one. It replaces the Dolev–Yao model with a weaker version, but now on both channels HD and HH. Its original description is:

"If the protocol messages M_1 to M_3 are run against a DY attacker and the messages M_4 to M_7 are run against an N+E attacker the attacker cannot produce any relevant attack".

Although the original statement of the theorem is open, we decided to specify it as the attacker's capability to fool the human behind the screen. To achieve that, we propose a man-in-the-middle attack happening in the Human–Human channel. Again, to do so, we remove Formulae 41, 46, 42 and 47 from $\Gamma_{extended}$. In logical notation, we then have:

$$\begin{gathered}\Gamma_{extended} - \{\text{ Formula } 41, \text{ Formula } 42, \text{ Formula } 46, \\ \text{Formula } 47 \} \vdash [\neg(\\ MHH(sent(human(i), human(a), \\ v\_g(kua, kui, na, ni, i))) \\ \wedge \\ MHH(sent(human(i), human(b), \\ v\_g(kui, kub, ni, nb, i))))]\end{gathered}$$

The proof generated by SPASS, of the negation of a man-in-the-middle attack, shows us that the attacker is indeed unable to compromise the ceremony when he only possesses the eavesdrop capability on channels HD and HH. Thus, we have proved that the man-in-the-middle attack does not happen in this setting.

10 Gains of ceremony description under a realistic threat model

If the correct threat model for each association mode in the Bluetooth protocols was misunderstood, it would lead us to two types of incorrect conclusions:

- The protocol (and related ceremony) is not secure because they do not cope with an overly pessimistic threat model. Usually this would lead the protocol/ceremony designer to add features that would assume non-plausible assumptions and/or degraded usability.
- The protocol is secure, but the user misunderstands the threat he is subject to. This is again related to the costs of the added security measures to protect against all threats, even the unrealistic ones.

The ceremony for the Bluetooth pairing can be described avoiding the above conclusions. The ceremony could enforce the correct choice of threat model at the implementation level by restricting the modes available to the strongest type possible for that specific device; or at application level, where the application would dynamically allow/block association modes depending on the environment.

For example, the Bluetooth protocol could be implemented with the dynamic threat model in mind. Bluetooth devices are capable of sensing their surroundings, and in some cases—for example, in the case of smart phones—they can pinpoint the exact location they are at that moment. With these capabilities, the implementation of the ceremony could avoid the user misunderstanding the threat model he is subject to. As examples we can cite that a pairing ceremony with a fixed-pin device could only happen after sensing the radio and not detecting the presence of any other devices around, thus sticking to the designed threat model. Or, when pairing with other semi-constrained devices, this pairing does not happen in a known public place, as this is not covered by the actual threat model the ceremony was design for. In a more precise way, the ceremony should take into account, for example, whether there are more Bluetooth-enabled devices around before allowing pairing under the just works mode. If there is more than one, the just works mode should not be available, as the weakened threat model requires that, if only one device is found, the just works mode could be securely used, since it respects the threat model specified for its use.

This kind of ceremony potentially leads to a positive side effect; it trains users to detect different threat models for those situations. In the same example, the user is able to learn through experience that to be more protected in public environments they need to use stronger authentication mechanisms, while in private they are subject to different threats.

Although the examples discussed in this paper focus on the Bluetooth pairing protocol, the idea of using variations of the threat model can be applied for several different ceremonies, varying from ATM authentication ceremonies to TLS handshake protocol implementations and its variations.

11 Conclusions

The existence of a single worst-case scenario threat model is justifiable in security protocol scenarios. However, the same cannot be said for security ceremonies. Human agents executing security ceremonies are constrained by the laws of physics and usual abilities expected from human beings. The existence of such a powerful agent in a setting involving human-to-human communication is not plausible and is likely to demand solutions that are not tailored to reality.

Our approach for describing a threat model for security ceremonies is based on a well-established model for security protocols. In our approach, we weaken the attacker to conform to the premises governing human–device interaction and human-to-human interaction. This strategy seems plausible because it will help security protocols and ceremony designers to develop ceremonies with reasonable assumptions and tailored to the real capacities of the attacker.

In this paper, we presented our proposal and demonstrated using examples that the goals of our proposal are already embedded in industry deployed protocols. We described a fraction of a ceremony where a dynamic threat model using the weakened versions of the all-powerful attacker is used. Additionally, we showed that we can design a ceremony that helps the user by ensuring the correct assumptions are made by protocol designers. We also presented a set of formulae that provide support for mechanised and automated verification of security ceremonies. To specify these formulae, we used the theorem prover SPASS and first-order logic. We extended a scheme presented by Weidenbach [28] that performs the verification using an unbounded number of parallel runs and an unbounded number of peers per run. An important characteristic of the formulae we produced is that they are generic, so we can easily reuse them in the verification and specification of other ceremonies.

Our formulae also include the specification and verification of the variable threat model proposed. We managed to dismantle the threat model into a set of formulae that can easily be added or removed from the specification. With this strategy, we can represent all the threat model nuances proposed for the Bluetooth Pairing Protocol in Numeric Comparison mode. This attacker model can also be re-used in the verification of other ceremonies with little effort.

Finally, we were able to formally verify that the SSP ceremony under the NC mode holds the expected properties when the threat model is correctly specified. We specified the protocols and the ceremony together in a scheme that can trace the distribution of knowledge between peers. These results match the empirical analysis performed. Our framework can also easily build a proof path, which will teach the ceremony designer or verifier how to reproduce any attack found. This is an implicit characteristic inherited from Weidenbach's specification.

Our contributions are at least threefold. Firstly, we are able to mechanise a scheme capable of specifying and formally verifying security ceremonies. Secondly, we created a deconstruction of the Dolev–Yao attacker so that it can be used following the ideas stated in earlier sections. Finally, our mechanised scheme is able to teach the ceremony designer or verifier the exact steps he should follow to construct a real-world attack if possible.

In addition to showing that a tailored threat model is important to realistically analyse security ceremonies, we could perceive how our model can be useful to detect attacks to ceremonies. A straightforward example is that the attacks found on the legacy mode could be easily detected by our implementation if we describe this protocol as a ceremony. An attacker with the capability of eavesdropping on the HD or HH channel would be able to learn the PIN (e.g. hearing/reading the PIN value), which should be secret. The attacker, in possession of this value, would be able to decode all the messages between the devices.

The next steps for research in this area clearly include implementing and refining the formulae by using them to verify other ceremonies, especially those that make more use of encryption and digital signatures schemes (the SSP does not directly uses encryption and digital signatures). A natural next step is the implementation of our formal model in a more expressive logic, such as higher-order logic (HOL). Although, it will probably be more difficult to implement in the beginning, if we specify this current framework in HOL, some more conclusive findings can be asserted.

References

1. Abadi, M., Rogaway, P.: Reconciling two views of cryptography (the computational soundness of formal encryption). J. Cryptol. **20**(3), 395–395 (2007)
2. Alexander, R.D.: The evolution of social behavior. Annu. Rev. Ecol. Syst. **5** (1974). http://www.jstor.org/stable/2096892
3. Anderson, R., Needham, R.: Robustness principles for public key protocols. In: CRYPTO '95. Springer (1995)
4. Arsac, W., Bella, G., Chantry, X., Compagna, L.: ARSPA-WITS. Lecture Notes in Computer Science. In: Degano, P., Viganò, L. (eds.) Validating Security Protocols Under the General Attacker, pp. 34–51. Springer (2009). doi:10.1007/978-3-642-03459-6
5. Arsac, W., Bella, G., Chantry, X., Compagna, L.: Multi-attacker protocol validation. J. Autom. Reason. **46**(3–4) (2011). doi:10.1007/s10817-010-9185-y
6. Backes, M., Pfitzmann, B.: Relating symbolic and cryptographic secrecy. IACR Cryptology ePrint Archive 2004, 300 (2004), http://eprint.iacr.org/2004/300
7. Balfanz, D., Smetters, D.K., Stewart, P., Wong, H.C.: Talking to strangers: Authentication in ad-hoc wireless networks. In: NDSS'02. San Diego (2002)
8. Bella, G., Bistarelli, S., Massacci, F.: Retaliation: can we live with flaws? In: IACS, vol. 6. IOS Press (2006)
9. Bella, G.: Formal Correctness of Security Protocols. Information Security and Cryptography. Springer, Berlin (2007)
10. Bella, G.: Formal Correctness of Security Protocols, Information Security and Cryptography, vol. XX. Springer, Berlin (2007)
11. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: CRYPTO' 93, LNCS, vol. 773. Springer (1994)
12. Bluetooth Special Interest Group: Bluetooth specifications 1.0 - 2.1+EDR. Technical specifications. http://www.bluetooth.com (1999–2007)
13. Bluetooth Special Interest Group: Simple pairing whitepaper v10r00. Technical report (2006)
14. Carlos, M.C., Price, G.: Understanding the weaknesses of human-protocol interaction. In: Proceedings of the 16th International Conference on Financial Cryptography and Data Security. pp. 13–26. FC'12, Springer, Berlin, Heidelberg (Mar 2012)
15. Creese, S., Goldsmith, M., Roscoe, A.W., Zakiuddin, I.: The attacker in ubiquitous computing environments: formalising the threat model. In: FAST'03 (2003)
16. Dhamija, R., Tygar, J.D., Hearst, M.: Why phishing works. In: SIGCHI'06. ACM, New York (2006). doi:10.1145/1124772.1124861
17. Dolev, D., Yao, A.: On the security of public key protocols. IEEE Trans Inf Theory **29**(2), 198–208 (Mar 1983)
18. Dolev, D., Yao, A.C.: On the security of public key protocols. IEEE Trans. Inf. Theory **29**(2), 198–208 (1983)
19. Ellison, C.: Ceremony Design and Analysis. Cryptology ePrint Archive. Report 2007/399 (Oct 2007)
20. Jakobsson, M.: The human factor in phishing. In: Priv. Secur. Consum. Inf. '07 (2007)
21. Jakobsson, M., Wetzel, S.: Security weaknesses in bluetooth. In: CT-RSA 2001, LNCS, vol. 2020. Springer (2001)
22. Needham, R.M., Schroeder, M.D.: Using encryption for authentication in large networks of computers. Commun. ACM **21**(12) (1978)
23. Parker, G.: Assessment strategy and the evolution of fighting behaviour. J. Theor. Biol. **47**(1) (1974). http://www.sciencedirect.com/science/article/pii/0022519374901118
24. Ryan, P., Schneider, S.: Modelling and Analysis of Security Protocols, 1st edn. Addison Wesley, Boston (2001)
25. Shaked, Y., Wool, A.: Cracking the bluetooth pin. In: MobiSys '05s. ACM, New York (2005)
26. Stajano, Anderson: The resurrecting duckling: security issues for ad-hoc wireless networks. In: IWSP: International Workshop on Security Protocols, LNCS (1999)
27. Weidenbach, C.: SPASS Input Syntax Version 1.5. Max-Planck-Institut fur Informatik
28. Weidenbach, C.: Towards an automatic analysis of security protocols in first-order logic. In: 16th International Conference on Automated Deduction, pp. 314–328. Springer, London, UK (1999)