Master research Internship

Bibliographic report

Verification of security protocols: distance bounding protocols

Domain: Cryptography and Security

Author:
Alexandre Debant

Supervisor:
Stéphanie Delaune
EMSEC

Abstract: The symbolic verification of cryptographic protocols is a well studied domain. It enables automatic proofs of security properties on some protocols, but cannot be applied to some recent ones, called distance bounding protocols. In common models distances cannot be expressed. Thus, new models must be proposed or adapted to analyze them.

Contents

1 Introduction

Nowadays cryptographic protocols are often used for critical applications. For example, the HTTPS protocol is often used for the internet communications to ensure mutual authentication. An other example is remote voting: it is more and more used to make the vote more easier for voters. Flaws in these protocols may have important consequences. Therefore we can be interested in proving their correctness.

The correctness of all these protocols depends on the definition of "secure": in payment on the internet, secure means "with mutual authentication", in the electronic voting, secure means "with confidentiality". Common security properties are authentication, integrity, privacy and non-repudiation. Many others can be expressed depending on the domain and the application of the protocol.

Protocols are more and more complex. The more they are complex the more designers will make mistakes designing them. Therefore, since they are used in critical domains, tools to make automatic proofs are needed. Such tools require the use of models. They can be split into two families: the computational models and the symbolic models. The first one is extremely precise modeling messages as bitstrings and participants as probabilistic polynomial time Turing machine. In the second one, messages are modeled by abstracted terms and participants are modeled depending on the messages a participant can send and/or receive.
Both have pros and cons. The computational model is very precise and this precision has consequences: making proofs in these models can be very hard. On the opposite, the symbolic model is a little less precise but proofs tend to be easier or at least can be automated more easily.
For these two kinds of models, there exist tools to make automatic proofs. In the computational model we can mention CryptoVerif [3] and in the symbolic model ProVerif [2].

These tools have been developed to analyze and prove what we will call standard protocols. They are protocols in which no physical properties are taken into account. For example, the locations of participants in the network is not defined: they can always be located everywhere. Such protocols are for example HPPS, TLS, SSL... But for several years, contactless devices have become more and more used. New uses of such devices in which flaws can be critical have appeared: for example, payment with credit cards or smart keys. All these new ways of using protocols naturally lead us to define a new kind of security properties: the physical proximity. When a payment is done with a terminal, enforcing the physical closeness between the credit card and the terminal is an efficient way to guarantee the security. It is the same thing for smart keys. Specific algorithms have been designed to achieve this goal.
Moreover, this new devices may have huge consequences in case of flaws in the implemented protocol. Therefore it is needed to prove the correctness of such protocols. Because of their specificity of using physical parameters, standard techniques to make proofs do not suite well. New techniques and tools will have to be studied and developed.

State of the art. This report presents a state of the art in symbolic verification of distance bounding protocols. It starts presenting the domain of the symbolic verification. Then it describes what is a distance bounding protocols and how they are currently analyzed. Finally it talks about

the limitations of standard models of this domain and presents what will be the high lines of the internship.

2 State of the art in symbolic verification of standard protocols

Standard protocols have already been studied using symbolic verification methods. This technique enabled to find new flaws called logical attacks. To study this protocols, a lot of symbolic models have been proposed. Each model has its own specificities but they share some common features that are briefly discussed in this section. From these models some results of decidability for security properties have been proved and several tools have been developed.

2.1 Logical attacks on an example

People often think that using well-designed cryptographic primitives, like the RSA for encryption and signature or SHA-2 as hash function, will make their protocol secure. It's wrong. Of course, this condition is important but not sufficient. There exist flaws called logical attacks. These attacks break the security properties using the structure of the protocol, the sequence of actions executed by the agents. G. Lowe describes in [11] such an attack against the Needham-Schroeder protocol [13].

Needham-Schroeder protocol. Needham-Schroeder is an authentication protocol: at the end, an agent A, Alice, wants to be sure that she is communicating with an agent B, Bob. The protocol presented in figure 1 proceeds as follows: first, Alice sends to Bob $\{(A, N_A)\}_{pkB}$, her identity and a fresh nonce encrypted with Bob's public key. Receiving this message, Bob can decrypt it and send back $\{(N_A, N_B)\}_{pkA}$, the received nonce and a fresh nonce N_B encrypted with Alice's public key. Finally Alice can decrypt this message and send back $\{N_B\}_{pkB}$ to prove that she has received Bob's answer.

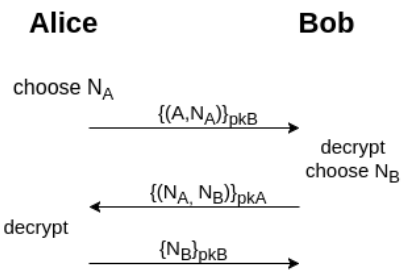

Figure 1: Needham-Schroeder protocol

After the execution, Bob knows that he is communicating with a agent knowing N_B. Since this nonce has been encrypted with Alice's public key and Alice is the only agent who can decrypt such an encrypted message, Bob can deduce that the other agent is her. Intuitively this protocol seems to ensure Alice's authentication because the secrecy of N_B seems to be ensured. Similarly, this protocol seems to ensure Bob's authentication. The authentification of both agents is called mutual authentification.

Lowe's attack on the Needham Schroeder protocol

At the end, we would like to say that Alice and Bob can be sure that they are communicating together but it is wrong. G. Lowe showed that even under perfect cryptography assumption, which means that without the corresponding private key it is impossible to decrypt a message encrypted with a public key, there exists an attack which is presented in figure 2. This attack consists in a man-in-the-middle attack: the attacker Charlie, C, intercepts and relays messages from Alice to Bob decrypting received messages and encrypting them with Bob's public key. Finally Bob thinks

having talked with Alice but he really talked with Charlie. Alice's authentication is broken.
This attack doesn't depend on the primitive used to encrypt messages. Charlie never tries to break the algorithm of encryption but he can anyway break the mutual authentication property just playing with exchanged messages.

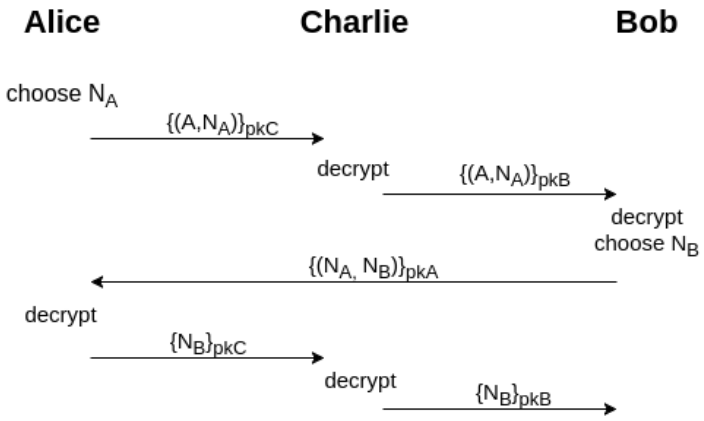

Figure 2: Attack against Needham-Schroeder protocol

2.2 Symbolic models

To formally describe protocols we need to fix a model. A lot a different models have been proposed in the literature. Most of them have been derived from the original Dolev-Yao model [8]. Therefore, even if they are different they have a lot of common features. We will present here the general aspects of such models.

Messages. Messages are defined as first order terms. The atomic ones represent for example constants and nonces. The non-interpreted function, represent, for example, cryptographic primitives or tuples. To do that they are abstracted by uninterpreted symbols of function; cryptographic primitives are modeled as blackboxes. To model the messages exchanged in the Needham-Schroeder protocol the following signature may be used:

$$\mathcal{F} = \{< \_, \_ >, \{\_\}_{pub(\_)}, pub(\_), priv(\_)\} \cup \mathcal{A}$$

where $\mathcal{A}$ represents atomic messages.
For example, the term $\{< A, N_A >\}_{pub(B)}$ models the first message sent in the protocol.

Given a set of messages M, an agent can derive new messages. For example knowing a public key $pub(k)$ and a message m, it can derive the message $\{m\}_{pub(k)}$. Inversely, knowing an encrypted message $\{m\}_{pub(k)}$ and the corresponding private key, an agent can derive m. Finally, if an agent knows a pair then it can derive both elements. We can define $DM(M)$, the set of derivable messages from M, as the smallest set satisfying all the rules presented in figure 3. Moreover, we assume that each agent will be able to derive a set of nonces unknown from another agent.

Protocol. A protocol is a set of roles (typically 2 or 3) and each role is a sequence of actions describing how an agent must behave depending on which messages he receives. An action is non

$$\frac{k \in \mathcal{A}}{pub(k) \in DM(M)}\ PUB \qquad \frac{< m, n > \in M}{n \in DM(M)}\ RPAIR$$

$$\frac{m \in M \quad n \in M}{< m, n > \in DM(M)}\ PAIR \qquad \frac{m \in M \quad pub(k) \in M}{\{m\}_{pub(k)} \in DM(M)}\ ENC$$

$$\frac{< m, n > \in M}{m \in DM(M)}\ LPAIR \qquad \frac{\{m\}_{pub(k)} \in M \quad priv(k) \in M}{m \in DM(M)}\ DEC$$

Figure 3: Derivation rules

ground term which can be for example send or receive. A non ground term is a term which can contain variables representing, for example, a message or a nonce.
In the Needham-Schroeder protocol, the roles of Alice and Bob are the sequences:

$$role_{Alice} = [send(\{< A, N_A >\}_{pub(x)});\ receive(\{< N_A, \mathbf{y} >\}_{pub(A)});\ send(\{\mathbf{y}\}_{pub(x)})]$$

$$role_{Bob} = [receive(\{< \mathbf{x}, \mathbf{y} >\}_{pub(B)});\ send(\{< \mathbf{y}, N_B >\}_{pub(x)});\ receive(\{N_B\}_{pub(B)})]$$

Attacker. In the model, an attacker is very powerful. We assume that he is ubiquitous. Intuitively, the attacker is the network. An attacker can always intercept all messages sent by other agents. Therefore as soon as a message is sent in the network, it is known by the attacker; the attacker can derive it. In addition to intercept messages, an attacker can trigger receive action using any message that it is derivable from his current knowledge. Such receive can contain any messages derivable by the attacker.
In such a model of attacker we assume that all messages go through the attacker before being received.

Traces. In our model, a trace is a sequence of events defined as follows: at each step we try to apply the first behavior of each role of the agents. If several are possible one is chosen and is removed to the role of the agent. At each step we can also make the attacker sending a Receive message as described above. The non deterministic choices at each step between all possible actions and the receives triggered by the attacker makes the set of different traces noted $Tr(proto)$.

Properties. Finally, in our models, properties can be expressed as a reachable issue. For example, the weak secrecy which corresponds to ensure that an attacker cannot know a specific message s can be expressed by exploring all valid traces $Tr(proto)$: is it possible to reach a state where the attacker can derive s? Authentication properties can be expressed similarly.

2.3 Symbolic verification: results and tools

Even for simple secrecy properties and when only using simple cryptographic primitives, the verification of security properties is, in general, undecidable [10]. This is due to several sources of unboundedness. First, a protocol can be played an arbitrary number of times; we say that the

number of sessions is unbounded. Then the number of agents and the size of messages are not bounded either. Finally, an attacker can send any messages he can derive at each step and can generate as many nonces as he wants. All these infinite sets make the model too complex. But despite its undecidability, several tools have been developed to make automatic proofs. Such tools use techniques like abstractions to make the proofs. In [11] G. Lowe proposes a variation of the Needham-Schroeder protocol to fix the attack presented before. When answering, Bob encrypts also his identity: the message $\{< N_A, N_B >\}_{pkA}$ becomes $\{< N_A, N_B, B >\}_{pkA}$. The correctness of this new protocol has been proven using for example ProVerif. This proof does not only prove that the presented attack is no longer possible but proves also that there does not exist other logical attacks.

M. Rusinowitch and M. Turuani proved in [14] that the secrecy is decidable in the class of protocols with a bounded number of sessions. Even if the bounded number of sessions hypothesis eliminates many infinite sets (they become finite), an attacker can still build messages of arbitrary size. Therefore this result of decidability is not straightforward.
Actually, proving the absence of attacks with a bounded number of agents provide a certain confidence in the protocol. Similar results of decidability have also been proved under various hypotheses: more complex cryptographic primitives, other type of security properties (e.g. equivalence-based properties).

3 Distance bounding protocols

The omnipresence of wireless communication led to the rise of new contactless devices. Today we can pay with a contactless credit card or we can use a keyless entry system for our car. Both are examples of critical devices. Any attack against such device could have huge impacts. Therefore to be able to use them with great serenity, confidently, we have to be able to prove some security properties. The new one (in comparison with standard protocols) is a distance property: we need to ensure *physical proximity*. The credit card paying something should be close to the terminal. To open a car, the key should be physically close to it; the key must not be farther to three or five meters. This new security goal led to the design of distance bounding protocols which can be split into two main families. Finally, studies have been done to find attacks on them.

3.1 Overview

To ensure the property of physical proximity, the notion of "time out" (described more precisely after) has been implemented in existing protocols. However it has been shown that without some constraints it cannot enforce physical proximity. Therefore new protocols, called distance bounding protocols, have been designed. Distance bounding protocols are specific cryptographic protocols based on the notion of time and manipulating also classical primitives.

Before studying distance bounding protocols let us discuss about the "time out" notion. In most of contactless devices the proximity notion is currently ensured by that. At the beginning of the protocol a timer is started and at the end it is stopped. If the process took too much time then the protocol is aborted. The notion of "too much time" must be defined: the developer must fix

a threshold. This one cannot be too small to enable devices with a small computing power to communicate with the terminal. But it cannot be too large because an attacker may compute very quickly; a large threshold enables him to be far from the terminal. A compromise must be found but it is not easy and sometimes impossible. Moreover, if an answer can be anticipated then an attacker can reduce its measured time and therefore can be farther to the terminal.
These two aspects lead to constraints about messages exchanged during the time measurement. This is why specific protocols are designed.

Formally, the goal of a distance bounding protocol is to enable a device, called the *Verifier*, to establish an upper bound on its physical distance to an other device, called the *Prover*.
The general shape of a distance bounding protocol is presented in Figure 4. We can see that there are three phases:

1. **Setup phase:** the Prover P and the Verifier V exchange information without measuring time;
2. **Time measurement:** they perform a quick phase measuring time, sending one or several challenges;
3. **Finalizing phase:** they verify the good execution of the protocol.

Since the time measured in the second phase must represent the distance from P to V, answers should not be anticipated and the computation time should be negligible. During this phase, encryption, signature or commitment should not be computed; they must be computed before. Only operations that can be done very quickly can be computed.
In the Figure 4, the finalizing phase only contains one round of challenge/response. Currently, there is a number n of such tests.

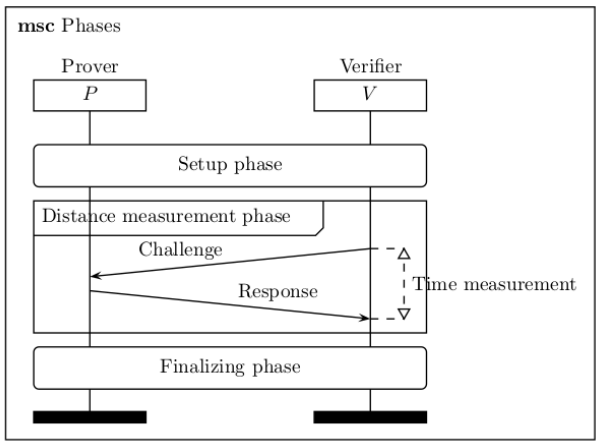

Figure 4: General shape of a distance bounding protocol

The time measurement phase starts at the date t_S when the Verifier sends its first challenge and finishes at the date t_R when the Verifier receives the last one. The security property is: $dist_{phys}(P, V) \leq \frac{c}{2^k}(t_R - t_S)$ where $dist_{phys}$ represents the physical distance, c represents the speed of communications and k the number of challenges sent.

In the literature many distance bounding protocols have been proposed (see e.g. [5] for a recent survey) A classification in two families is often used: the descendants of Brands and Chaum [4] and the descendants of Hancke and Kuhn [9]. The first family is based on a commitment which is revealed during the finalizing phase. On the contrary, the second family does not need exchange during the last phase; the Verifier only needs to make some computations.

3.2 Some examples

Before presenting both families of distance bounding protocols, we will present a naive distance bounding protocol to enable an easier understanding of different notions.

A naive protocol. As presented is [4], let us define the simplest and most intuitive distance bounding protocol: let N_V be a fresh nonce generated by the Verifier V, V sends it to the Prover P, P sends it back to V (see figure 5). To compute its distance to P, V only needs to measure the duration between the first sending by V and the reception, by V, of the reply.
Since the nonce sent by the Verifier is fresh, the Prover cannot anticipate its answer. Therefore, we can deduce that when the Verifier receives the message coming from the Prover then the duration from its initial sending corresponds to twice the distance between P and V.

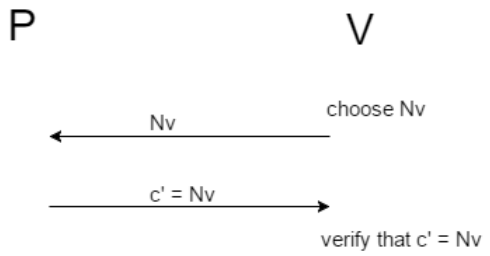

Figure 5: Simplest protocol

However, this protocol is not secure. Since the Verifier has no way to authenticate the Prover, an attacker can impersonate the Prover. Therefore even if the Prover is far from the Verifier, an attacker close enough can anticipate the challenge response and can make the Prover accepted.

Brands and Chaum protocol. In 1993, Brands and Chaum proposed to overcome this flaw with the distance bounding protocol presented in figure 6. This protocol follows the same structure as the protocol presented before but a commitment is sent before and is revealed at the end. The commitment is sent during the setup phase and the signature is computed during the finalizing phase thus the duration of the computation has no impact on the time measured during the second phase.

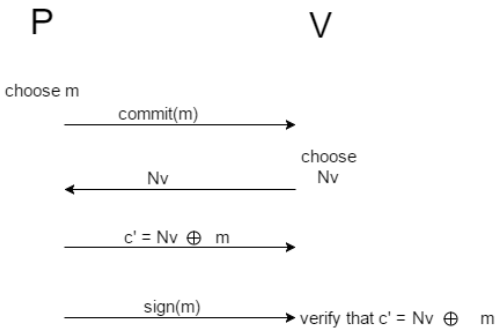

Figure 6: Brands & Chaum protocol

Hancke and Kuhn protocol. In 2005 Hancke and Kuhn presented in [9] a new distance bounding protocol that does not need to process a finalizing phase (see figure 7).
In this protocol the Verifier and the Prover share a secret key K and a pseudo-random function h. The verifier starts by sending a nonce N_v to the Prover. This one computes a secret value $h(K, N_v)$, which cannot be computed by an attacker, and split it into (R_0, R_1). The verifier starts the distance measurement phase by sending a bit b to the Prover. It answers with $c' = R_b$. The Verifier can verify the correctness of the answer by computing $h(K, N_v) = R_0||R_1$. No extra message has to be exchanged.

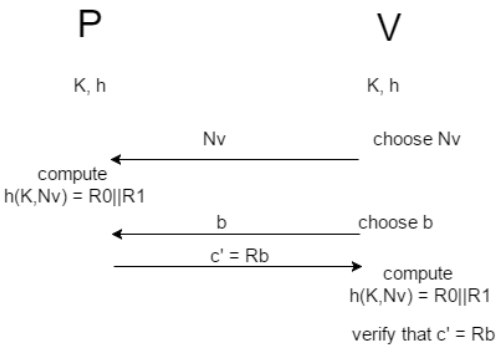

Figure 7: Hancke and Kuhn protocol

3.3 Some common attacks

To analyze a distance bounding protocol three classes of attacks are traditionally considered. All these attacks have the same goal: make the Verifier V believe that the Prover P is physically closer to the Verifier than it really is. The main difference between these classes is which entities carry out the attack and their mutual relationships. We can distinguish three entities: the Verifiers, the Provers and the external attackers. Provers and Verifiers can be honest or dishonest: an honest agent executes the protocol correctly while a dishonest can anticipate answers, share its private secrets, or communicate with other dishonest agents or external attackers.

Distance Fraud attack. In a *Distant Fraud* attack only two agents are involved: a Verifier supposed to be honest and a Prover dishonest. The goal of the dishonest Prover is to shorten the distance measured by the Verifier. This attack is possible when the Prover can answer the challenge before receiving it. In this way the measured distance can clearly be shortened.
In both protocols presented before, this class of attacks is not possible because the nonce is supposed to be fresh. The Prover cannot guess it.

Mafia Fraud attack. In a *Mafia Fraud* attack, three agents are considered: a Verifier, a Prover and an external attacker. This last one performs the attack and both the Verifier and the Prover are supposed to be honest. Typically the attacker will be closer to the Verifier than the Prover is and will try to use this advantage to shorten the distance computed before.
The attack described on the naive protocol in section 3.2 belongs to this class.

Terrorist Fraud attack. A *terrorist Fraud* attack involves 3 agents: an honest Verifier and a dishonest Prover (or at least not entirely honest) but can also communicate with an external attacker. We assume that it does not want to reveal its private knowledge; for example a secret key. Possible reasons to this unwillingness is impersonation: after having shared its personal key for a signature, everyone can sign on its behalf. Without this restriction the Verifier could not distinguish

between an external attacker and the Prover; therefore these attacks would be equivalent to a *Mafia Fraud* attack.

In 2012 only the three previous classes were studied to prove the security of a distance bounding protocol. In [7], Hijacking attacks have been discovered and cannot be reduced to the three previous classes. This class of attacks corresponds to a dishonest Prover which will try to shorten its distance to the Verifier using another honest Prover.

Hijacking attack. Cremers et al. showed in [7] that many protocols considered secure previously, are vulnerable to this kind of protocols. Often these attacks consist in running protocol in an honest way between the Verifier and the honest Prover P but the dishonest Prover P' will replace all messages identifying P by its own. For example it will replace all MACs or signatures.

The protocol presented in Figure 6 is not secure. A Prover P close to V can start the protocol with a Verifier V. During the finalizing phase, a dishonest Prover P', the attacker, too far from the Verifier can intercept the decommitment sent by P and re-sign it with its signature. This attack is described in figure 8. In this way the Verifier will think that it is talking with P' and therefore will locate P' at the location of P. The mutual authentication if therefore correct.

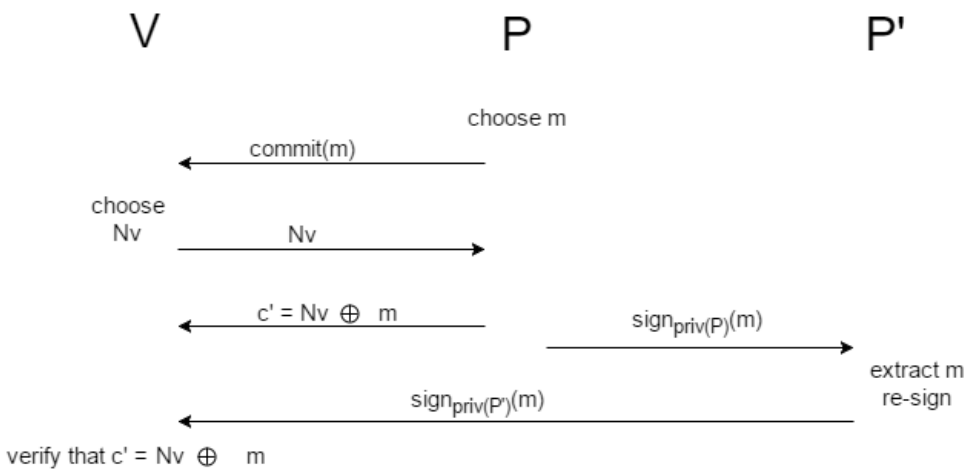

Figure 8: Hijacking attack against the Brands and Chaum protocol

4 Towards symbolic and automated verification of distance bounding protocols

Nowadays to analyze a distance bounding we focus on the four attacks presented before. We said that it is secure if it resists to these four classes of attacks. But we are not sure that these four classes contain all possible attacks against distance bounding protocols. Until 2012 only the three first types of attacks were considered in the study of the security of a protocol. Cremers et al. showed that several protocols supposed to be secure were vulnerable to this new Hijacking attack.

In the next years new attacks might be found. Therfore, there is a need to develop results and verification tools to allow one to perform a formal analysis of distance bounding protocols.

4.1 Limitations of standard models

Nowadays, automatic cryptographic protocol verifiers, like ProVerif, often use symbolic models derived from the Dolev-Yao model. None of them can express a notion of time. In such models, an attacker is very powerful: without a time notion, messages are assumed to be sent instantly: when an agent sends a message, another agent can immediately receive it. Therefore, an attacker sending messages instantly can break all notions of physical distance in the network and unrealistic attacks may appear. In this model an attacker is ubiquitous. A Verifier will not be able to define the distance of a Prover because even if it is very far from the Verifier it can send a message very quickly. Without notion of time or distance, it is impossible to model distance bounding protocols. Therefore tools commonly used today are not suitable to model distance bounding protocols.

4.2 A specific model: Basin's model

From this observation, several models have been proposed to take into account a notion of physical time or distance. In 2007, Meadows et al. proposed in [12] a new logic to describe distance bounding protocols but their model cannot detect attacks under user collusion like *Terrorist Fraud* attacks. Thus this model is not well adapted to prove properties of proximity on distance bounding protocols.

In 2011 Basin et al. have presented in [1] another model. This is a very precise model in which two notions of distance are defined: the distance in the network, $dist_{Net}$, and the physical distance $dist_{Phys}$. There is just one constraint on $dist_{Net}$: information cannot propagate in the network faster than the speed of light. The consequence is that $dist_{Net}$ is always greater than $dist_{Phys}$. Having a notion of distance, this model seems to be relevant and takes into account the four classes of attacks presented before.
Even if the model proposed in [1] is syntactically different from the protocol presented in section 2, they share many features. The most important difference is that a notion of time can be expressed. We will present all these differences and similarities.

Time. The model defines two kinds of agents: the *Honest* and the *Dishonest*. A location $loc_A \in \mathbb{R}^3$ is associated to each agent A.
The duration for a message to go from an Agent A to another B is defined by the distance between both and the speed of communications. The last one is supposed known.
We define two notions of distance: $dist_{phys}$ is the line-of-sight distance and $dist_{net}$ is the distance taking into account the specificity of the environment. If there is an obstacle between two agents A and B, $dist_{net}(A, B)$ may be equal to $\perp$. These two distances are defined with the following equation where c represents the speed of light:

$$dist_{phys}(A, B) = \frac{|loc_A - loc_B|}{c} \leq dist_{net}(A, B).$$

Messages. As presented in the description of symbolic models, a message is a term. It can either be atomic or composed. Composed messages are the same as presented in section 2 but in the atomic ones, a set of constants must to be added. This one represents the time in the protocol.
From a message an agent can derive new messages. Common rules dealing with encryption, description and pairs are kept but a new one is added:

$$\frac{}{Time\ t \in DM(M)}\ TIME.$$

This rule just says that an agent always knows the time in the protocol; the time can always be derived.

Protocol and attacker. Protocols are still defined by set of roles. But, in this timed model, we no longer can say that it is the attacker which adds *Receive* messages in the network. Indeed, since an attacker has a location, it cannot receive messages instantly. Passing through its to add *Receive* will hence increase the delay to send a message between two agents.
The attacker presented in the Basin model looks like the one presented in section 2: it can always send one of the messages it can derive. However, since an attacker is considered as an agent, it has a location and therefore messages are received and sent with delays.

Traces. As in the model presented in section 2, a trace is a set of events but since a notion of time is present in the Basin model, a date is added for each action. A trace is therefore a timed sequence.
Moreover, to define the set of all correct traces, it is not sufficient to use rules defined by all the roles. A new rule must be defined to add *Receive* messages in the system (as explained before the attacker cannot play this role). This rule is the following:

$$\frac{\begin{array}{c} tr \in Tr(proto) \quad \mathbf{t_R} \geq \mathbf{maxtime(tr)} \\ (t_S, Send(A, m, L) \in tr \\ \mathbf{dist_{Net}(A, B) = t_{AB}} \\ t_{AB} \neq \perp \quad \mathbf{t_R} \geq \mathbf{t_S} + \mathbf{T_{AB}} \end{array}}{tr.(t_R, Recv(B, m)) \in Tr(proto)}\ NET.$$

This rule expresses the fact that to receive a message, an agent B must have waited at least a delay t_{AB} which corresponds to the sending of the messages (sent by A). To enforce that the time in a trace increases, the condition $T_R \geq maxtime(tr)$.

Properties. Properties are still properties of reachability. To express them Basin chose to add a new event called $Claim(A, m)$". Such an event expresses a belief or a conclusion made by a protocol participant and formalized as a message. Such a message could be $< B, \frac{c}{2^k}(t_R - t_S) >$ where c represents the speed of communications and k the number of challenges sent. Hence, the proximity property can be expressed as: if a valid trace contains an event $Claim(A, < B, d >)$ then we must have $d \geq \frac{c}{2^k}|loc(A) - loc(B)|$.

4.3 Limitations and possible improvements

The Basin model is very interesting for our purpose: it models precisely the distance and has a realistic model of attacker that seems to be precise enough for our purposes. But, as presented in

[1], it seems to be too much complex to imagine use it in an automatic tool. Basin et al. present several proof of security of distance bounding protocols but they only apply their model on simple protocols. To make their proofs easier they present several general results independent from the protocol. For example, they have shown that if the first event of a trace $(t_A, Send(A, m_A))$ in which m_A contains a nonce N_A and if the same trace contains an event $(t_B, Send(t_B, m_B))$ in which m_B contains N_A too then $t_B - t_A \geq dist_{phys}(A, B)$. A similar result is also proved for *Send* and *Receive* events: if there are a *Send* and a *Receive* event sharing the same nonce then the delay between both is greater than the distance between both agents. However they seem not to be sufficient to make an automation of the possible proofs.

Therefore, we will try to establish new results to make the automatic proofs of distance bounding protocols possible. We will try to establish some reduction results to reduce the size of search space. For example an interesting result would be to find a bound on the number of agents needed to abstract all the possible attacks or to reduce the number of different physical organization of the network (locations of honest and dishonest agents). A reduction like the last one has been presented by Cortier et al. in [6] for routing protocols. Hence, we may be optimistic to find such a result for distance bounding protocols.

5 Conclusion

In conclusion, we can say that distance bounding protocols are protocols based on a notion of time. The aim is to locate an agent in a network and it is done by measuring the time spent during a specific phase of the protocol.
The symbolic verification has been studied a lot for standard protocols. We have a lot of tools for automatic proofs. But, the models used behind these tools cannot be used to make proofs on distance bounding protocols: they do not model the notion of time or distance and therefore they cannot model a realistic attacker.
To model the distance bounding protocols, Basin has proposed a very interesting symbolic model: a notion of time is expressed and this enables to represent the delay between the sending and the reception of a message. This is the main issue to model distance bounding protocols. However, this model is to complex to be used in automatic proofs.
To automate the proofs we need to find results simplifying the search space and making the proofs less complex. An interesting solution can be to find reduction results as presented in section 4.3.

References

[1] David Basin, Srdjan Capkun, Patrick Schaller, and Benedikt Schmidt. Formal reasoning about physical properties of security protocols. *ACM Transactions on Information and System Security (TISSEC)*, 14(2):16, 2011.

[2] Bruno Blanchet. An automatic security protocol verifier based on resolution theorem proving (invited tutorial). In *20th International Conference on Automated Deduction (CADE-20)*, Tallinn, Estonia, July 2005.

[3] Bruno Blanchet. A computationally sound mechanized prover for security protocols. In *IEEE Symposium on Security and Privacy*, pages 140–154, Oakland, California, May 2006.

[4] Stefan Brands and David Chaum. Distance-bounding protocols. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 344–359. Springer, 1993.

[5] Agnès Brelurut, David Gerault, and Pascal Lafourcade. Survey of distance bounding protocols and threats. In *International Symposium on Foundations and Practice of Security*, pages 29–49. Springer, 2015.

[6] Véronique Cortier, Jan Degrieck, and Stéphanie Delaune. Analysing routing protocols: four nodes topologies are sufficient. In *International Conference on Principles of Security and Trust*, pages 30–50. Springer, 2012.

[7] Cas Cremers, Kasper B Rasmussen, Benedikt Schmidt, and Srdjan Capkun. Distance hijacking attacks on distance bounding protocols. In *2012 IEEE Symposium on Security and Privacy*, pages 113–127. IEEE, 2012.

[8] Danny Dolev and Andrew Yao. On the security of public key protocols. *IEEE Transactions on information theory*, 29(2):198–208, 1983.

[9] Gerhard P Hancke and Markus G Kuhn. An RFID distance bounding protocol. In *First International Conference on Security and Privacy for Emerging Areas in Communications Networks (SECURECOMM'05)*, pages 67–73. IEEE, 2005.

[10] Nevin Heintze and J Doug Tygar. A model for secure protocols and their compositions. *IEEE transactions on software engineering*, 22(1):16–30, 1996.

[11] Gavin Lowe. Breaking and fixing the needham-schroeder public-key protocol using fdr. In *International Workshop on Tools and Algorithms for the Construction and Analysis of Systems*, pages 147–166. Springer, 1996.

[12] Catherine Meadows, Radha Poovendran, Dusko Pavlovic, LiWu Chang, and Paul Syverson. Distance bounding protocols: Authentication logic analysis and collusion attacks. In *Secure localization and time synchronization for wireless sensor and ad hoc networks*, pages 279–298. Springer, 2007.

[13] Roger M Needham and Michael D Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, 1978.

[14] Michaël Rusinowitch and Mathieu Turuani. Protocol insecurity with a finite number of sessions and composed keys is np-complete. *Theoretical Computer Science*, 299(1-3):451–475, 2003.