Validating Integrity for the Ephemerizer's Protocol with CL-Atse

Charu Arora[1] and Mathieu Turuani[2]

[1] Indian Institute of Technology, Delhi, India
charu.arora7@gmail.com
[2] Loria-INRIA, Vandoeuvre-lès-Nancy, France
turuani@loria.fr

Abstract. It is usually very difficult in Computer Science to make an information "disappear" after a certain time, once it has been published or mirrored by servers world wide. This, however, is the goal of the IBM ephemerizer's protocol by Radia Perlman. We present in this paper the general structure of the CL-Atse protocol analysis tool from the AVISPA's tool-suite, and symbolic analysis of the ephemerizer's protocol and its extensions using CL-Atse. This protocol allows transmitting a data which retrieval is guarantied to be impossible after a certain time. We show that this protocol is secure for this property plus the secrecy of the data, but is trivially non secure for its integrity. We model a standard integrity check as a first extension to this protocol, which is natural and close to common usage, and we present a second extension for integrity that is much less obvious and deeply integrated in the structure of the ephemerizer's protocol. Then, we show that while the first extension guaranty the basic integrity property under certain conditions, the second one is much stronger and allows faster computations.

1 Introduction

It is a known difficult problem to ensure that a data is completely destroyed, say after a given amount on time: whatever it is transmitted by email, placed on a web server, etc.., a data is expected to be copied or archived in a way that we cannot truly control. To solve this problem, and to guaranty expiration times on certain messages, Radia Perlman proposed the so called ephemerizer's protocol [20], a solution where a unique, not completely trusted server manage the keys used to encrypt those messages. Since these keys are only known by the ephemerizer, deleting one when its expiration time is reached makes the data "disappear". It is the responsibility of the ephemerizer to provide keys for the protocol and delete them at the appropriate time. Moreover, Perlman's protocol has the extra advantage to use a so called triple encryption, that guaranty the secrecy of the data even when the ephemerizer is dishonest. However, it is kind of obvious that this protocol does not guaranty the integrity of the data.

In this paper, we propose an automatic analysis of this protocol w.r.t its security properties, as well as some extensions validating integrity. Many decision procedures have been proposed to decide security properties of protocols w.r.t. a bounded number of sessions [1,9,21,19] in the so called Dolev-Yao model of intruder [17], the dominating formal security model in this line of research (see [18] for an overview of the early history of protocol analysis). In particular, among the different approaches the symbolic ones [19,12,14] have proved to be very effective on standard benchmarks [13] and discovered new flaws on several protocols. Here, we uses the CL-Atse tool [22] to analyze the Ephemerizer's protocol and its extensions. The modularity and performance of this tool appeared to be very useful for analyzing protocols from the AVISPA [2] project in which CL-Atse is involved since a few years (with OFMC [6], SATMC [3] and TA4SP [7]), as well as for the RNTL Prouvé project. The CL-Atse tool can be freely used, either by binary download on the CL-Atse web page[3], or through on-line execution on the AVISPA web page[4]. It allows automatic formal analysis of cryptographic protocols with the single (necessary) restriction of a bounded number of sessions. These analysis are done on a symbolic level, i.e. bit-strings are replaced by terms in a language of messages, and we assume that all cryptographic primitives are perfect. As usual in such cases, the protocol is run in presence of an active intruder with all capacities of the Dolev-Yao intruder (i.e. he can intercept or block any message, impersonate agents, or use any legal cryptographic operation).

Paper overview First, we present the details of the version of the ephemerizer's protocol that is analyzed here (section 2), along with the security properties. This includes two versions of the integrity. Second, we give a general overview of the CL-Atse tool used to analyze this protocol (section 3). Then, in section 4 we show that while the protocol validates the standard security properties, a simple extension for data integrity fails and should never be used in practice. Instead, two extensions are proposed, one quite natural and the other one less obvious. We show that while the natural extension satisfy the basic data integrity property, the second one is much stronger and may even be faster in practice. We conclude in section 5. Also, note that the protocol models presented here are publicly available at [4].

2 The ephemerizer's protocol

The term signature allowed by the analysis tool and used to model the Ephemerizer's protocol is the following :

$$\begin{aligned}\mathscr{T}erm = \mathscr{A}tom \,|\, \mathscr{V}ar \,|\, \mathscr{T}erm.\mathscr{T}erm \,|\, inv(\mathscr{T}erm) \\ |\, \{\mathscr{T}erm\}^s_{\mathscr{T}erm} \,|\, \{\mathscr{T}erm\}^a_{\mathscr{T}erm} \\ |\, Sig_{\mathscr{T}erm}(\mathscr{T}erm) \,|\, HMAC(\mathscr{A}tom, \mathscr{T}erm) \\ |\, \mathscr{T}erm \oplus \mathscr{T}erm \,|\, Exp(\mathscr{T}erm, \mathscr{P}roduct) \\ \mathscr{P}roduct = (\mathscr{T}erm)^{\pm 1} \,|\, (\mathscr{T}erm)^{\pm 1} \times \mathscr{P}roduct\end{aligned}$$

Terms can be atoms, variables, concatenations (or pairing), and symmetric or asymmetric encryption (marked by s or a). Also, $inv(k)$ is the inverse of k for asymmetric encryption. Note that if k is a (random) term, then $inv(k)$ exists but is unknown

[3] `http://www.loria.fr/equipes/cassis/softwares/AtSe/`

[4] `http://www.avispa-project.org/web-interface/`

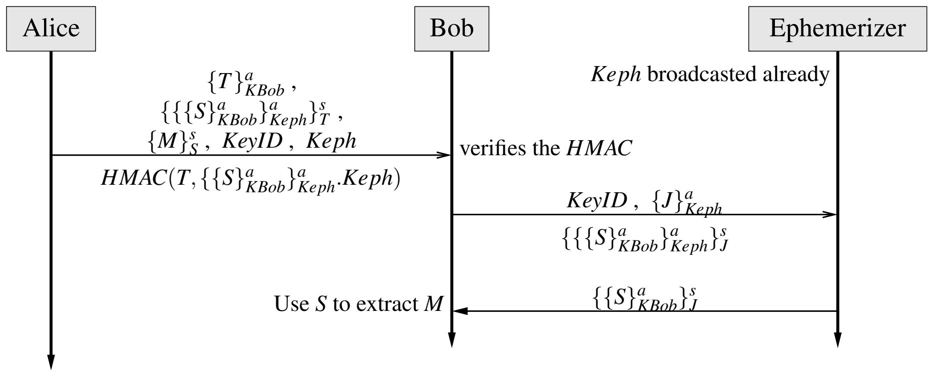

Alice
Bob
Ephemerizer
$Keph$ broadcasted already
$\{T\}^a_{KBob}$, $\{\{\{S\}^a_{KBob}\}^a_{Keph}\}^s_T$, $\{M\}^s_S$, $KeyID$, $Keph$
$HMAC(T, \{\{S\}^a_{KBob}\}^a_{Keph}.Keph)$
verifies the $HMAC$
$KeyID$, $\{J\}^a_{Keph}$
$\{\{\{S\}^a_{KBob}\}^a_{Keph}\}^s_J$
Use S to extract M
$\{\{S\}^a_{KBob}\}^s_J$

Fig. 1. Ephemerizer Scheme proposed by Radia Perlman (using triple encryption). T, S, and J are symmetric keys generated during the protocol.

to every agent. $Sig_k(m)$ represents the message m plus a signature on m with key k. $HMAC(k,m)$ represents m plus a MAC on message m with key k. In the tool this is coded as $\{h,m\}^a_k$ with some (optional) header h to differentiate multiple operators: on a formal point of view, the only difference between signature and asymmetric encryption is the agents who knows the key k or it's inverse $inv(k)$. The $\oplus$ and $Exp(..)$ operators model the xor and exponentiation operators. A $\mathscr{P}roduct$ represents a product of bit-strings (modeled by terms) to be used as an exponent for the $Exp(..)$ operator. Thus, each term in the product is equipped with $^{+1}$ or $^{-1}$, in order to model usual properties such that $a^{+1} \times a^{-1} \times b^{+1} = b^{+1}$. The intruder capabilities in CL-Atse match the Dolev-Yao model [17], extended for xor and exponentiation as in [10,11].

However, the Ephemerizer's protocol relies on neither $\oplus$ nor exponentiation, and uses only atomic keys in its design. Therefore, we simplify a bit the term signature by allowing the following shortcuts to present the protocol and its analysis. Note however that this does not restrict the tool analysis in any way.

Notations: Following the notations of Radia Perlman with small differences, we note $u.v$ the concatenation of messages u and v; $\{M\}_K$ the encryption of M by K (symmetric or asymmetric depending on K's type); $\{M\}_{inv(KAlice)}$ the signature of M with $Alice$'s private key. We also assume the existence of a subset $AKeys \subseteq Atom$ containing the public keys for asymmetric encryption or signature. Note that for the analysis tool as well as for the modeling in the tool's language, a signature is equivalent to an encryption with a private key, and a MAC is equivalent to an encryption with a public key which private key is unknown to everybody, including the intruder.

Description: The ephemerizer's protocol is a communication protocol that allows an agent, say $Alice$, to send one message (or more) protected by an expiration time. While the recipient (Bob) shall be able to retrieve the message(s) before the expiration time, this must become impossible after the time is reached. To do so, a trusted third party is required to provide an ephemeral key $Keph$, i.e. a public encryption key linked with an expiration time, that is used to encrypt the data sent to Bob. Then, Bob must ask the ephemerizer for a decryption key that he will get only if the expiration time is not reached yet. This ensures the expected ephemeral property. Moreover, by using multiple encryption with single-use symmetric keys, the protocol also ensures that the message remains secret for anybody except *Alice* and *Bob*, even if the ephemerizer is dishonest. The protocol is displayed in figure 1, with *KBob* being *Bob*'s public key, and with *T*, *S*, *J*, *M* being nonces, i.e. atoms freshly generated at run time, and represents respectively three symmetric keys and the message from *Alice*. Here, *S* is the symmetric key protecting *M* that *Bob* must acquire from the ephemerizer. *S* is sent to *Bob*, too, but protected by *Bob*'s public key so that only he can get it, and protected by the ephemeral key *Keph* to ensure that *Bob* don't get it if the expiration time is reached. It is then protected (again!) by *Bob*'s public key (through *T* for efficiency) to ensure that only *Bob* can query the ephemerizer. This is the so called triple encryption. *Bob*'s query to the ephemerizer simply consists in *Bob* asking him to remove the protection of the ephemeral key *Keph*.

The initial state of the protocol matches the expects: all public keys of agents are known by everybody including the intruder, as well as *Keph* and *KeyID* (ID of *Keph*), and the *HMAC* function; private keys are known by their owner only; and other atoms (*T*, *S*, *M* and *J*) are generated during the execution.

Security properties: This protocol was designed to guarantee both the ephemeral property on *M* (i.e. bob cannot obtain *M* after the expiration time), and the confidentiality of *M* (only *Alice* and *Bob* can obtain *M*). According to the formal analysis of this protocol that we performed with CL-Atse (see Section 4), these properties are always satisfied for at most two sessions, and for all the 3-sessions scenarios that we could run. However, it appears immediately that this protocol does not guaranty the integrity of *M*: the intruder can impersonate *Alice* to send his own message to *Bob*. However, integrity of *M* is a basic property that many users may need. Therefore, in this paper we add the two following properties to the previous basic ones:

1. Integrity of the message: it is impossible for an intruder to corrupt, change or replace *M* during the transfer;
2. Integrity of the protocol run: it is impossible for an intruder to corrupt, change or replace any of the temporary keys of the protocol, i.e.. *T*, *S*, *J* or *Keph*.

In order to guaranty the property 1 above, a user would certainly simply sign *M* with *Alice*'s private key, assuming that *Bob* knows her public key already. Along with the proof of destination guarantied by the confidentiality property of the Ephemerizer's protocol, this proof of origin "seems" to guaranty integrity. This approach is very classical in the real world, where users or agents usually compose protocols (or cryptographic methods) with limited security properties to reach stronger ones. We will see in the analysis in Section 4 how limited this approach can be. But for now, we simply remark that we cannot rely on the Ephemerizer's nonces to guaranty the security of this combined protocol w.r.t. multiple sessions: if an agent plays *Alice* twice, then the intruder can exchange the messages of the two sessions. Therefore, one need to include either the official recipient's name or a sessions ID of *Alice* in the message transmitted, i.e. the protocol that we consider initially is the following:

Modified Ephemerizer (root version): In the original Ephemerizer protocol, we replace *M* by $\{M\}_{inv(KAlice)}.IDCheck$, i.e *Alice*'s signature on *M* joint with some *IDCheck* atom to identify each session. Depending on which variant of the integrity property we consider, *IDCheck* will be either *Bob* (Bob's name, for weak integrity) or a *SiD* (a session ID, for strong integrity). These two variants are defined as follows.

To define the integrity properties that we consider, we need the differentiate roles and agents. Now, *Alice* and *Bob* used before are in fact only roles, i.e. pieces of protocols or services run by real agents *a* or *b*, e.g. real computers or humans. Agents can run many roles, or sessions, in parallel. In our opinion, there are two possible variants for the integrity property that the final user may additionally require for this protocol, namely the weak and strong integrity :

Integrity Variant n°1 (weak integrity): A data like *M* or *T* is corrupted between *a* and *b* playing *Alice* and *Bob* when *b* receives a value for *M* that has never been sent by *a* in any of the (multiple) sessions she plays with *b*. That is, we allow messages of one session to reach an other session as long as the agents are the same. To ensure this, we uses *Bob* as *IDCheck*'s value.

Integrity Variant n°2 (strong integrity): Same as above, but a message is also corrupted if it is accepted in an other session (no crossing). To ensure this, adding the recipient's name is not enough. Therefore, we uses *SID* as *IDCheck*'s value, with *SID* a public, unique, number (like a port number) identifying *Alice*'s session playing with *Bob*. We assume that *Bob* (and the intruder!) knows *SID* from the start of the protocol.

Both variants will be checked for validity against the protocol (and its two patches) in Section 4. Note that the encryption with *S*, as well as the triple encryption over *S*, should prevent any modification of *SID* or *Bob*'s name. While these modifications may look obvious at first, and may even be performed in practice since it is only a modification of *M*, we will see that it is still possible for the intruder to combine multiple sessions in order to corrupt the message *M*.

3 Overview of CL-Atse

The protocol analysis methods of CL-Atse have their roots in the generic knowledge deduction rules from CASRUL [12] and AVISPA. However, a lot of optimizations and major extensions have been integrated in the tool, like prepossessing of the protocol specifications of extensions to manage the algebraic properties of operators like xor or exponentiation. In practice, the main characteristics of CL-Atse are:

- A general protocol language: CL-Atse can analyze any protocol specified as a set of IF rewriting rules (no restriction, see [2] or the documentation on AVISPA's web page for IF details). The figure 2 shows the standard process of protocol analysis using the AVISPA tools, from a specification in HLPSL (role-based, same idea as strands) to any of the four tools available at the moment.
- Flexibility and modularity: CL-Atse structure allows easy integration of new deduction rules and operator properties. In particular, CL-Atse integrates an optimized version of the well-known Baader & Schulz unification algorithm [5], with modules for xor, exponentiation, and associative pairing. To our knowledge, CL-Atse is the only protocol analysis tool that includes complete unification algorithms for xor and exponentiation, with no limitation on terms or intruder operations.

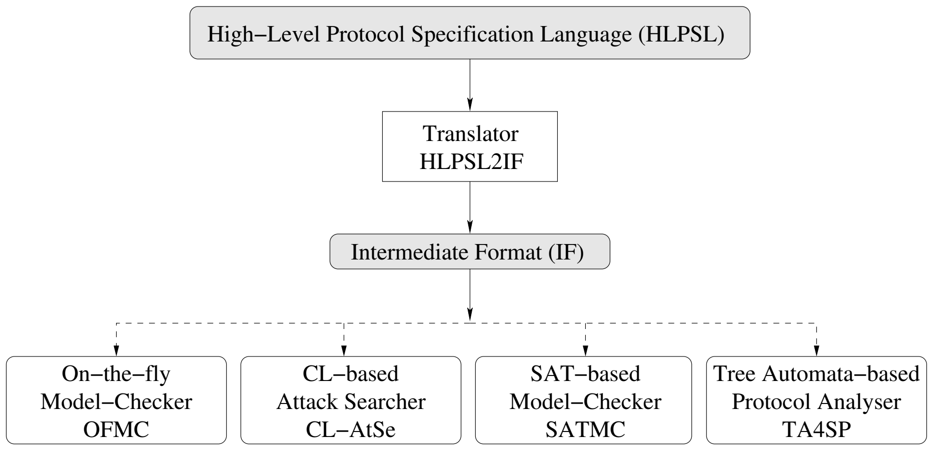


Fig. 2. Structure of AVISPA's analysis tool

- Efficiency: CL-Atse takes advantage of many optimizations, like simplification and rewriting of the input specification, or optimizations of the analysis method.
- Expressive language for security goals: CL-Atse can analyze any user-defined state-based property specified in AVISPA IF format.

Since protocol security is undecidable for unbounded number of sessions, the analysis is restricted to a fixed but arbitrary large number of sessions (or loops, specified by the user). Other tools provide different features. The closest to CL-Atse are:

The OFMC tool [6], also part of AVISPA, solves the same problem as CL-Atse except that loops and sessions are iterated indefinitely. However, OFMC proposes a different method to manage algebraic properties of operators: instead of hard-coding these properties in the tool, a language of operator properties is provided to the user. Equality modulo theories is solved through modular rewriting instead of direct unification with state-of-the-art algorithms for CL-Atse. However, since this language covers all theories, termination is only obtained by specifying bounds on message depths and number of intruder operations used to create new terms. Hence, completeness cannot be ensured. CL-Atse does not provide such flexibility on properties, but it also does not have any limitation for the theories it can handle (xor, exponentiation, etc...). Moreover, thanks to modularity in the unification algorithm and in knowledge deduction rules, it is quite easy to include new algebraic (or cryptographic) properties directly in the tool.

The Corin-Etalle [14] constraint-based system, which improves upon one developed by Millen & Schmatikov, relies on an expressive syntax based on strands and some efficient semantics to analyze and validate security protocols. Here, strands are extended to allow any agent to perform explicit checks (i.e. equality test over terms). This makes a quite expressive syntax for modeling protocols, that is however subsumed by IF rules. Moreover, to our knowledge no implementation for xor and exponential is provided.

The SCYTHER tool [15], recently developed by Cas Cremers, is dedicated to unbounded protocol analysis. However, unlike other tools for unbounded protocols which restrictions, heuristics or approximations also apply to the case of a bounded number of sessions, this one do not suffer from this limitation. This would be a nice alternative for analyzing the Ephemerizer's protocol, especially for an unbounded number of sessions, assuming that the user-defined predicates and properties used here are not problematic for analyzing an unbounded number of sessions (but they should not be).

These other tools could have very well been used for analysis in this paper with equivalent results. Choosing CL-Atse had the advantage to allow cheating only one protocol model understandable by all tools in the AVISPA's project. Also, various algorithms are implemented in CL-Atse to simplify and optimize the input protocol specification, and also to guide the protocol analysis. However, these methods require working on a protocol specification with some special features. Listing these would be quite technical, but the most important ones are that all protocol steps and roles must be local to only one participant, and that CL-Atse must eliminate all honest agent's knowledge by converting them into a small set of equality and inequality constraints over terms with global variables. This allows CL-Atse to compute closures of the participant's or intruder knowledge, unforgeable terms, sets or facts, and to optimize each role instance accordingly (prepossessing). This prepossessing has two main axes :

Protocol simplifications: They reduce the overall size of the protocol, and specifically the number of steps, by merging some protocol steps together, or tagging others with execution directives (e.g. tag a protocol step to be run as soon or as late as possible). This is a generic process in CL-Atse's algorithm, thus not limited to the Ephemerizer's protocol in this paper. Also, these tags are not heuristics, in the sense that opposite choices are never tried. Thus, CL-Atse tag a protocol step only when it was able to prove statically that if an attack exists, then there exists one validating the tag. In practice, this occurs quite often.

Optimizations: Protocol optimizations aim at rewriting automatically some parts of the protocol in order to accelerate the search for attacks. The acceleration can be significant, and the protocol structure can be changed deeply but equivalently. The idea is to track all possible origins of cipher-texts that the intruder must send but cannot create himself (i.e. necessarily obtained from an agent). By building an exhaustive list of origins for such terms, CL-Atse can reduce the future work of the analysis algorithm by unifying these terms with each of their possible origins and generate minimal choice points accordingly. Analysis acceleration comes from a reduction of redundancy in the steps execution. Moreover, this strategy also fixes the time when steps holding such cipher terms must be run in an attack, thus reducing interleaving.

Once all prepossessing are done, the analysis algorithm implemented in CL-Atse symbolically executes the protocol in any possible step ordering. While the maximum number of symbolic executions build by the tool remains finite (exp. bounded in the size of the protocol specification), each one represents an infinite number of protocol traces and intruder actions. Note that no bound is assumed on the intruder (neither the number of actions nor the size of terms it outputs). Also, this analysis relies on a (generic) unification algorithm modulo the properties of the operators, like xor or exponentiation, that provide all term-specific computations.

4 Symbolic analysis with CL-Atse

We modeled the Ephemerizer's protocol and all its variants in the HLPSL language, input of the AVISPA's protocol analysis platform in which CL-Atse is a back-end. Even if not particularly complex and quite readable, the technical design of the modeling in this language would be too long to describe in this paper. However, all modeling in HLPSL used in this paper are publicly available and can be found at [4]. We refer to the AVISPA's user manual (google avispa-project) for deeper concerns about HLPSL.

Formal Modeling of the integrity properties: The integrity is modeled as usual in HLPSL using the *witness* and *request* or *wrequest* predicates defined initially for authentication, and the property shortcuts provided in HLPSL. E.g. for weak integrity:

- the `witness(Alice,Bob,m,M1)` predicated is released when *Alice* send $M1$ to *Bob*;
- the `wrequest(Bob,Alice,m,M2)` predicate is released when *Bob* receives $M2$;
- and the protocol must guaranty that for each *wrequest* there exists a matching *witness*.

Formal Modeling of the ephemeral property: The Ephemeral property however cannot rely on any predicate or property already defined in HLPSL or in the tool. Therefore, user-defined predicates and security properties must be written in the modeling specifically for this protocol. Hopefully, the analysis tool is able to check a wide range of user-defined properties, written in an property language in HLPSL based on LTL formula. Thus, we defined the following predicates :

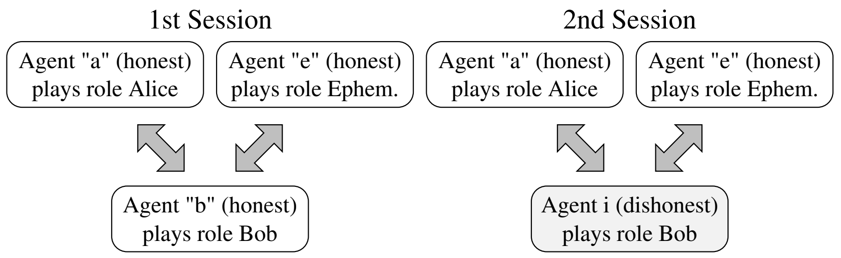
- `message_decr_bob(A, B, E, KeyID, KEph)` is activated by *Bob* when he become able to decrypt $\{M\}_S$ associated to key $KEph$. This is equivalent to *Bob* knowing M.
- `message_not_decr_bob(A, B, E, KeyID, KEph)` is activated by *Bob* when he is denied receiving S by the Ephemerizer. This is not truly equivalent to *Bob* knowing M since he could have performed an other request to the Ephemerizer before the expiration time.
- `no_expiry_eph(A,B,E,KeyID,KEph)` is activated by the Ephemerizer as long as the key *Keph* did not expire.
- `expiry_eph(A,B,E,KeyID,KEph)` is activated by the Ephemerizer when the key *Keph* expire. Note that the instant when the key expire is not deterministic, i.e. the protocol must be secured independently of the key life-time.

Using these predicates, the modeling of the Ephemerizer's security property in HLPSL is quite simple, using the LTL notation :

```
[] ( [-] expiry_eph(A,B,E,KeyID,KEph)
      => message_not_decr_bob(A,B,E,KeyID,KEph) )
[] (     message_decr_bob(A,B,E,KeyID,KEph)
      => no_expiry_eph(A,B,E,KeyID,KEph) )
```

This can be read as: ”At any moment, if somewhere in the past the key $KEph$ expired, then *Bob* must be denied retrieving the decryption key associated to $KEph$”; ”At any moment if *Bob* is allowed retrieving the decryption key associated to $KEph$, then $KEph$ must not have expired”.

An integrity attack on *M*: During the analysis of this protocol with CL-Atse, many attacks were found on the integrity of any of the internal data of this protocol (M, T, S, $Keph$). The most complex ones showed integrity flaws of either S or T. However, since the central data in this protocol is M only, we choose to present here a simple integrity attack on M w.r.t. the simple protocol extension presented above, for the strong integrity. The same attack also works for weak integrity. The scenario is the following, with a, b, e three agents and i the intruder:


1st Session
2nd Session
Agent "a" (honest) plays role Alice
Agent "e" (honest) plays role Ephem.
Agent "a" (honest) plays role Alice
Agent "e" (honest) plays role Ephem.
Agent "b" (honest) plays role Bob
Agent i (dishonest) plays role Bob

We write X_n the object X in session n. First, the session 1 is run normally, thus adding $\{M_1\}_{inv(KAlice)}$ to the intruder's knowledge. Then, the intruder can simply impersonate a in session 2 using $\{M_1\}_{inv(KAlice)}$ instead of $\{M_2\}_{inv(KAlice)}$: the $Alice$'s signature is the only thing that the intruder cannot create himself. However, b cannot differentiate M_1 from M_2, so M_1 is accepted and the integrity is lost.

Patch n°1, signing *M* and *Sid*: The previous attack occurs for the single reason that the official receiver of M_1 could reuse the signature for $Alice$ on it in an other session of the protocol. To prevent that, we can naturally include SID, or *Bob*'s name, in the signature: $\{M,SID\}_{inv(KAlice)}$ instead of $\{M\}_{inv(KAlice)}.SID$. While it may not be obvious at first that we also need to protect SID or *Bob*'s name with the signature, this modification guaranty the integrity of M in the ephemerizer's protocol. However, here we also want to guaranty the integrity of the local keys, i.e. S, T, J and $Keph$. Hopelessly, the signature on M is unable to prevent the intruder from modifying or replacing any of these local variables: there exists many attacks on the integrity of these keys, including very complex ones. These can be seen in the model and analysis files [4] associated to this work, which include Alice-Bob description of each attack in the tool's output.

Patch n°2, signing *S* and *Sid*: The problem of guarantying the integrity of all M, S, T, J and $Keph$ here is that we just cannot sign everything. For example, the analysis shows that our goal would be reached if we could sign M, T and J (the key generated by *Bob*), and that omitting to sign at even one of these objects allows the intruder to perform an attack. But, in practice it would not be affordable to add more than one signature. Moreover, signing only the HMAC could look like a good idea, since it contains data that depends on S, T, and $Keph$. But still, some attacks remain which can be seen in the tool's output in [4]. In fact, it appeared during the analysis process of this protocol that the only way to guaranty the integrity of all local keys (plus M) is to sign S directly (along with Sid): this is actually the central key of all the transmission, and signing it prevents any modification on M (since S encrypts M), on T (since nobody except Bob can retrieve $\{\{S\}_{KBob}\}_{Keph}$ which is encrypted with T), and on $Keph$ (since T's integrity is guarantied). Therefore, the modification w.r.t the original protocol is the following :

$$\{\{S\}_{KBob}.Sid\}_{inv(KAlice)} \text{ replaces } \{S\}_{KBob} \text{ everywhere}$$

It is remarkable that the signature must be placed *inside* the triple encryption: if placed outside, that is if $Alice$ sends $\left\{\left\{\{\{S\}_{KBob}\}_{Keph}\right\}_T.Sid\right\}_{inv(KAlice)}$ to reduce encryption time, then an attack still exists on the integrity of M. Similarly, there also exists an attack if $Alice$ signs $\{\{S\}_{KBob}\}_{Keph}$ only.

On the point of view of the encryption time, this is very interesting: we can keep encrypting only the "small" message S with $KBob$ (slow, asymmetric encryption), while we must encrypt $\{S\}_{KBob}$ and the signature with only $KEph$, T and J (fast, symmetric encryption). Moreover, this may even be faster than signing M, since $\{S\}_{KBob}$ is probably much smaller and faster to sign than M.

Successful analysis: For all analyzed scenario, no attacks were found on the ephemerizer's protocol with the signature on $\{S\}_{KBob}$ described above, for all the properties described in this paper (including secrecy of M and integrity of M,S, T, $Keph$ and J), and for any of the weak or strong integrity properties (with Bob's name or Sid respectively). Alternatively, signing S directly gives the same result. Also, signing M with the correction of patch n°1 still guaranty the integrity of M (alone). For all variants of this protocol, we analyzed as many execution scenarios as we could, including all relevant scenarios where honest agents plays at most two roles (that is, scenarios that are not trivially secure), plus some scenarios with three or four roles per honest agent. This is actually the limit of the analysis tool for this protocol: with more sessions, no answer comes in a reasonable time (less than an hour). While it may be interesting to use parallel computing to raise this limit, we think that the analyzed scenarios are the most relevant ones for this protocol. While only CL-Atse were used during the modeling process and the generation of all scenario variants to be analyzed, other tools from the AVISPA project can be run to confirm the final results: OFMC [6] and SAT-MC [3], giving similar result. Note that OFMC may require small adjustments for the user-defined predicates of the Ephemerizer's property. However, SAT-MC don't, and its speed greatly increased recently as shown in [16], thus allowing it to go a bit farther in increasing the number of sessions. We would however not expect new attacks from that.

5 Conclusion

In this paper, we presented an analysis of the ephemerizer's protocol by Radia Perlman with CL-Atse and the AVISPA's tool-suite. The analysis has three main results: first, it confirmed that the original protocol is secure against the ephemeral property and the secrecy of M (even if the ephemerizer is dishonest); second, to reach the integrity of M

(the transmitted data), it showed that we cannot count on the encryption by S to prevent modifications of M: at least the patch n°1 is required; and third, it showed that while signing M is a partial solution for a non-modifiable implementation of the ephemerizer's protocol, it is in fact much better, and more secure, to sign S or $\{S\}_{KBob}$ instead (patch n°2): it is faster for large M, and it guaranty that no run of this protocol can deviate from the specification (meaning integrity of the protocol execution). For future work, it would be interesting to have a security proof for the second extension of this protocol for an unbounded number of sessions, either manually created, or automatically generated by a tool in a restricted model of protocol: best candidates are TA4SP [7], part of AVISPA, and ProVerif [8] for over-approximation methods; and SCYTHER [15] for complete characterization method. Also, a comparition of the state-of-the-art analysis tools can be found in .

Acknowledgments

The work presented in this paper was partially supported by the FP7-ICT- 2007-1 Project no. 216471, "AVANTSSAR: Automated Validation of Trust and Security of Service-oriented Architectures" (www.avantssar.eu).

References

1. R. Amadio, D. Lugiez, and V. Vanackère. On the symbolic reduction of processes with cryptographic functions. *Theor. Comput. Sci.*, 290(1):695–740, 2003.
2. The AVISPA Team. The Avispa Tool for the automated validation of internet security protocols and applications. In *Proceedings of CAV 2005, Computer Aided Verification*, LNCS 3576, Springer Verlag.
3. A. Armando, L. Compagna. An Optimized Intruder Model for SAT-based Model-Checking of Security Protocols. In *Proceedings of the Workshop on Automated Reasoning for Security Protocol Analysis (ARSPA 2004)*, ENTCS 125(1):91-108, 2005.
4. C. Arora. The Ephemerizer's specification files in HLPSL. `http://www.loria.fr/~turuani/Ephemerizer_models.zip`
5. F. Baader and K.U. Schulz. Unification in the Union of Disjoint Equational Theories: Combining Decision Procedures. In *Journal of Symbolic Computing*. 21(2): 211-243 (1996).
6. D. Basin, S. Mödersheim, L. Viganò. OFMC: A symbolic model checker for security protocols. In *International Journal of Information Security* 4(3):181–208, 2005.
7. Y. Boichut, P.-C. Héam, O. Kouchnarenko. Automatic Verification of Security Protocols Using Approximations. *INRIA Research Report* RR-5727, October 2005. `http://www.inria.fr/rrrt/rr-5727.html`
8. B. Blanchet. An Ecient Cryptographic Protocol Verier Based on Prolog Rules. In *Proceedings of 14th IEEE Computer Security Foundations Workshop (CSFW)*. IEEE Computer Society, 2001.
9. M. Boreale. Symbolic trace analysis of cryptographic protocols. In *Proceedings of the 28th ICALP'01*, LNCS 2076, pages 667–681. Springer-Verlag, Berlin, 2001.
10. Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. An NP decision procedure for protocol insecurity with xor. In *Proceedings of LICS 2003*, 2003.

11. Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. Deciding the Security of Protocols with Diffie-Hellman Exponentiation and Products in Exponents. In *Proceedings of the Foundations of Software Technology and Theoretical Computer Science (FSTTCS'03)*, LNCS 2914, Springer-Verlag, December 2003.
12. Y. Chevalier and L. Vigneron. A Tool for Lazy Verification of Security Protocols. In *Proceedings of the Automated Software Engineering Conference (ASE'01)*. IEEE CSP, 2001.
13. J. Clark and J. Jacob. A Survey of Authentication Protocol Literature: Version 1.0, 17. Nov. 1997. URL: `www.cs.york.ac.uk/~jac/papers/drareview.ps.gz`.
14. R. Corin and S. Etalle. An improved constraint-based system for the verification of security protocols. In *SAS*, LNCS 2477:326–341, Springer-Verlag, 2002.
15. Cas J.F. Cremers. Unbounded verification, falsification, and characterization of security protocols by pattern refinement. In *Proceedings of the 15th ACM conference on Computer and Communications Security*, ACM, 2008.
16. Cas Cremers and Pascal Lafourcade. Comparing State Spaces in Automatic Protocol Verification. In *Proceedings of the Seventh International Workshop on Automated Verification of Critical Systems (AVoCS'07)*, Elsevier ScienceDirect, 2007.
17. D. Dolev and A.C. Yao. On the Security of Public-Key Protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.
18. C. Meadows. Open issues in formal methods for cryptographic protocol analysis. In *Proceedings of DISCEX 2000*, pages 237–250. IEEE Computer Society Press, 2000.
19. J. Millen and V. Shmatikov. Symbolic protocol analysis with products and Diffie-Hellman exponentiation. In *Proceedings of the 16th IEEE Computer Security Foundations Workshop (CSFW'03)*, pages 47–61, 2003.
20. R. Perlman. The Ephemerizer: Making Data Disappear. Technical report, Sun Labs, 2005. At `http://www.research.sun.com/techrep/2005/smll-tr02005-140.pdf`
21. M. Rusinowitch and M. Turuani. Protocol Insecurity with Finite Number of Sessions is NP-complete. In *14th IEEE Computer Security Foundations Workshop (CSFW-14)*, pages 174–190, 2001.
22. M. Turuani. The CL-Atse Protocol Analyser. In Proceedings of RTA, 2006.