HAL
archives-ouvertes.fr

Adding Integrity to the Ephemerizer's Protocol

Charu Arora, Mathieu Turuani

▶ **To cite this version:**

Charu Arora, Mathieu Turuani. Adding Integrity to the Ephemerizer's Protocol. Stephan Merz and Tobias Nipkow. Sixth International Workshop on Automatic Verification of Critical Systems - AVOCS'06, Sep 2006, Nancy/France, pp.146-151, 2006. <inria-00091660>

HAL Id: inria-00091660
https://hal.inria.fr/inria-00091660

Submitted on 6 Sep 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Adding Integrity to the Ephemerizer's Protocol

Charu Arora[1] Mathieu Turuani[2]

LORIA-INRIA, Vandoeuvre-lès-Nancy, France

Abstract

We present a symbolic analysis of the ephemerizer's protocol by Radia Perlman, using the CL-Atse tool from the AVISPA's tool-suite. This protocol allows transmitting a data that will "disappear" (i.e. cannot be retrieved) after a certain time. We show that this protocol is secured for this property plus the secrecy of the data, but is trivially non secured for it's integrity. Therefore, we present two extensions of this protocol, one natural and probably already done in practice, the other one much less obvious. We shows that while the first extension guaranty the basic integrity property under certain conditions, the second one is much stronger and even allows faster computations.

Keywords: protocol analysis, cryptographic protocols, secrecy, integrity

1 Introduction

It is a 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 [5], 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 it's 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 short paper, we present a series of formal analysis over the ephemerizer's protocol and some of it's variants. These analyses were done using the CL-Atse tool

[1] This paper is an extract of Charu's work as student in Loria's CASSIS team.

[2] Email: Mathieu.Turuani@loria.fr

[7], which is part of the AVISPA [6] European Project's tool-suite, and that allows automatic formal analysis of cryptographic protocols with the single (necessary) restriction of a bounded number of sessions. These analyses 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. See [4] for an overview of formal methods used in protocol analysis. As usual in such cases, the protocol is run in presence of an active intruder with all capacities of the Dolev-Yao intruder [3] (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. A simple extension of the protocol for data integrity is modeled. However, in section 3 we show that it 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 4. Also, note that the protocol models presented here are publicly available along with the tools, so that anybody can check or extend the present work (see [1]).

2 The ephemerizer's protocol

Notation

Here, the notations follow the ones from Radia Perlman, with small differences due to our analysis tools. In particular, 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; and $HMAC\,(T, M)$ the MAC of M with key T.

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 encryptions 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. Here, S is the (new) 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

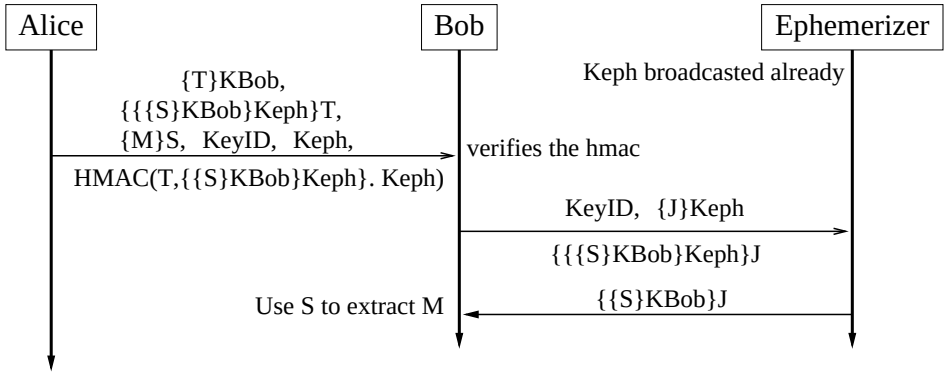

Fig. 1. Ephemerizer Scheme proposed by Radia Perlman (using triple encryption). T, S, and J are symmetric keys generated during the protocol.

(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, 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:

(i) Integrity of the message: it is impossible for an intruder to corrupt, change or replace M during the transfer;

(ii) 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. While this works for single sessions, this does not for multiple ones: 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 session ID of *Alice* in the message transmitted. Adding one or the other depends on the required strength of integrity. We consider two options:

- Option 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 replace M by $\{M\}_{inv(KAlice)}.Bob$, i.e. *Alice*'s signature on M joint with *Bob*'s name.

- Option 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 replace M by $\{M\}_{inv(KAlice)}.SID$, 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.

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 in fact possible for the intruder to combine multiple sessions in order to corrupt the message M.

3 Symbolic analysis with CL-Atse

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, c three agents and i the intruder:

	Alice		*Bob*		*Ephemerizer*
1st session :	a (honest)	$\longleftrightarrow$	i (dishonest)	$\longleftrightarrow$	e (honest)
2nd session :	a (honest)	$\longleftrightarrow$	b (honest)	$\longleftrightarrow$	e (honest)

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 exist many attacks on the integrity of these keys, including very complex ones.

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 depend on S, T, and $Keph$. But still, some attacks remain. In fact, it appears 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 secured), 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. 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.

4 Conclusion

In this short 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 tree main results: first, it confirmed that the original protocol is secured 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 secured, 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 create, or automatically generated by an automatic tool in a restricted model of protocol, like TA4SP [2], if it fits.

References

[1] Charu Arora. The ephemerizer's models and analysis tools. "http://www.loria.fr/~turuani/Ephemerizer_models.zip".

[2] Y. Boichut, P.-C. Héam, and 0. Kouchnarenko. Automatic verification of security protocols using approximations. Research report, INRIA, 2005.

[3] Danny Dolev and Andrew Chi-Chih Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–207, 1983.

[4] Catherine Meadows. Open issues in formal methods for cryptographic protocol analysis. In *MMM-ACNS*, page 21, 2001.

[5] Radia Perlman. The ephemerizer: Making data disappear. Technical report, Sun Labs, 16 Network Circle, Menlo Park, CA 94025, USA, 2005.

[6] The AVISPA team. The avispa tool for the automated validation of internet security protocols and applications. In *CAV*, pages 281–285, 2005.

[7] Mathieu Turuani. The cl-atse protocol analyser. In *RTA*, 2006. to be published.