From Protocol Specifications to Flaws and Attack Scenarios: An Automatic and Formal Algorithm

M. Debbabi M. Mejri N. Tawbi I. Yahmadi

Computer Science Department,
Laval University, Quebec, G1K 7P4,
Canada.
{debabi,mejri,tawbi,yahmadi}@ift.ulaval.ca

Abstract

We present a new approach to the verification of authentication protocols. This approach is formal, fully automatic and does not necessitate any specification of any protocol property or invariant. It takes as parameter the protocol specification and generates the set of flaws, if any, as well as the corresponding attack scenarios. This approach involves three steps. First, protocol roles are extracted from the protocol specification. Second, the intruder abilities to perform communications and computations are generated from the protocol specification. In addition to the classical known intruder computational abilities such as encryption and decryption, we also consider those computations that result from different instrumentations of the protocol. The intruder abilities are modeled as a deductive system. Third, the extracted roles as well as the deductive system are combined to perform the verification. The latter consists in checking whether the intruder can answer all the challenges uttered by a particular role. If it is the case, an attack scenario is automatically constructed. To exemplify the usefulness and efficiency of our approach, we illustrate it on the Woo and Lam authentication protocol.

1 Motivations and Background

In secure distributed systems it is mandatory to have some mechanism whereby principals (persons, hosts, computers, etc.) can prove their identities to each other. The act of proving the identity over networked systems is called authentication. Typically, cryptographic protocols are used to ensure authentication and related purposes.

It is well known that the design of authentication cryptographic protocols is error prone. Several protocols have been shown flawed in computer security literature. Consequently, a surge of interest has been expressed in the development of formal methods and tools for the specification, design and analysis of cryptographic protocols. A complete bibliography and a comparative study of these methods can be found in [1, 2, 3, 4, 5].

In this paper we present a new approach for the analysis of authentication cryptographic protocols. This approach is fully automatic, formal and does not necessitate any specification of any protocol property or invariant. The analysis of a given cryptographic protocol is structured in three main steps: First, starting from a classical protocol description, we extract what we call protocol roles. Actually, a role is a protocol abstraction where the emphasis is put on only one principal. Such an abstraction allows us to point out, for each principal, his interpretation of the exchanged protocol messages. Second, the intruder abilities are automatically extracted. Apart from the well known abilities, we consider all the computational abilities provided by the protocol itself. In our approach, such computation abilities are captured as a finite proof system in which the inference is based on narrowing i.e. rewriting modulo syntactic unification. Third, the roles together with the proof system are combined to perform the protocol verification. The latter consists in checking whether one can instrument a particular role by answering all the challenges uttered by this role.

The main contributions of this work are:

- A new approach for the analysis of cryptographic protocols. This approach is formal, fully automatic and does not necessitate any specification of any protocol property or invariant.

- An illustration of the usefulness and efficiency of our approach, on the analysis of the Woo and Lam authentication protocol.

We consider here a subclass of authentication cryptographic protocols. The approach deals with protocols that use symmetric-key cryptography, at most one server and where the only exchanged messages are constructed over nonces, principal identities and symmetric-keys. The rest of this paper is structured as follows: In Section 2 we recall briefly the one-way authentication protocol of Woo and Lam. Section 3 is devoted to an informal presentation of the verification algorithm. A few concluding remarks and a discussion of further research are ultimately sketched as a conclusion in Section 4.

2 The Woo and Lam Protocol

In what follows, we will recall the one-way Woo and Lam authentication protocol as presented in [6, 7]. This protocol relies on symmetric-key cryptography. The protocol is given in Table 1. Here N_b is a nonce, a random number generated by B specially for this protocol run, S is a server and k_{as} and k_{bs} are keys that A and B share initially with S. The protocol runs as follows when the principal A wants to prove his identity to the principal B: (1) A initiates the protocol and claims his identity to B; (2) B replies by sending the nonce N_b and asking A to encrypt it under k_{as} in order to prove what he claimed; (3) A returns the nonce N_b encrypted under k_{as}; (4) B forwards the response encrypted, together with A's identity, under k_{bs} for verification; (5) S decrypts the received message using B's key, extracts the encrypted component and decrypts it using A's key and re-encrypts under B's Key. If S replies by $\{N_b\}_{k_{bs}}$, then B will find N_b after decrypting it and he should be convinced that A is really running this session with him. The next section is devoted to an informal presentation of our verification algorithm. The latter will be illustrated over the Woo and Lam authentication protocol described above.

3 Informal Presentation

Starting from a protocol description, the algorithm operates in three main steps:

(a) Role extraction
(b) Proof system generation
(c) Verification

3.1 Role Extraction

Roles are protocol abstractions where the emphasis is put each time on a particular principal. A role reflects the way by which some principal perceives the protocol messages. For instance, in the case of the Woo and Lam protocol of Table 1, three roles could be extracted: A, B and S. The principal, playing the role A, participates in the protocol through three main steps: First, A sends his identity to the principal B. Second, he receives a nonce N_b from B. Third, he sends the message $\{N_b\}_{k_{as}}$ to B. Hence, the role associated to A could be written as the following sequence of actions:

$$Role(A) = \langle !, A, B\rangle\langle ?, N_b, B\rangle\langle !, \{N_b\}_{k_{as}}, B\rangle$$

An action is a triple of the form $\langle dir, m, P\rangle$ where dir is a direction symbol (either ? meaning input or ! meaning output), m is a message and P is a principal identifier. Similarly, the roles associated with B and S could be written as follows:

$$\begin{aligned} Role(B) &= \langle ?, A, A\rangle\langle !, N_b, A\rangle \\ &\quad \langle ?, \{N_b\}_{k_{as}}, A\rangle \\ &\quad \langle !, \{A, \{N_b\}_{k_{as}}\}_{k_{bs}}, S\rangle \\ &\quad \langle ?, \{N_b\}_{k_{bs}}, S\rangle \\ Role(S) &= \langle ?, \{A, \{N_b\}_{k_{as}}\}_{k_{bs}}, B\rangle\langle !, \{N_b\}_{k_{bs}}, B\rangle \end{aligned}$$

At that point, we would like to explain the second step of our algorithm i.e. the proof system generation.

3.2 Proof System Generation

In cryptographic protocol design, we always assume the presence of an intruder that has a complete control over the communication network. Accordingly, the intruder is able to intercept, modify, store, retrieve and send any circulating message in the network. Moreover, we assume that the intruder has the usual encryption and decryption abilities. In this work, we consider another valuable source of information available to the intruder: the protocol itself. Actually, the protocol may be viewed by the intruder as a computation resource. The latter may be used to synthesize and deliver some particular information that makes possible a fatal attack. Here, we capture all these intruder abilities by a deductive proof system. We associate to each protocol step an inference rule. Each inference rule captures a class of possible instrumentations of the protocol. The general form of an inference rule is:

$$\frac{p_1 \ldots p_n}{m}\ c$$

$Message\ 1.$	A	$\longrightarrow$	B	:	A
$Message\ 2.$	B	$\longrightarrow$	A	:	N_b
$Message\ 3.$	A	$\longrightarrow$	B	:	$\{N_b\}_{k_{as}}$
$Message\ 4.$	B	$\longrightarrow$	S	:	$\{A, \{N_b\}_{k_{as}}\}_{k_{bs}}$
$Message\ 5.$	S	$\longrightarrow$	B	:	$\{N_b\}_{k_{bs}}$

Table 1: The Woo and Lam Authentication Protocol

where $p_1, \ldots, p_n, m$ are messages and c is a boolean formula. Here is the way such an inference rule should be read: the intruder could supply the protocol with the messages $p1, \ldots, p_n$ and get the message m from the protocol provided that side condition c holds. In fact, the side condition refers to those freshness conditions dictated by the protocol. Furthermore, we will endow each inference rule with a sequence of protocol steps (a scenario) showing *how* the intruder could instrument the protocol with the information $p_1, \ldots, p_n$ so as to get the message m.

Now, let us see how this applies to the Woo and Lam protocol of Table 1. For the lack of space, we present here the inference rules together with the associated scenarios without explaining how they are generated.

The first step in the Woo and Lam protocol of Table 1 is:

$$1.\ A \longrightarrow B : A$$

The inference rule associated with this protocol step is:

$$\frac{}{A}$$

meaning that the intruder could get from the protocol the identity of any principal wishing to initiate a protocol session. Here is the scenario that makes this possible:

$$1.\ A \longrightarrow I(B) : A$$

This sequence stipulates that the intruder could get the identity of a principal A by intercepting the message sent by this principal when wishing to initiate a new protocol session.

The second step in the Woo and Lam protocol of Table 1 is:

$$2.\ B \longrightarrow A : N_b$$

The inference rule associated with this protocol step is:

$$\frac{A}{N_b}\ Fresh(N_b)$$

meaning that the intruder could get a fresh nonce (generated by B) from the protocol just by supplying it with an agent identity. The side condition $Fresh(N_b)$ stipulates the freshness[1] of the information N_b. Here is the scenario that illustrates such a situation:

1.	$I(A)$	$\longrightarrow$	B	:	A
2.	B	$\longrightarrow$	$I(A)$	:	N_b

The third step in the Woo and Lam protocol of Table 1 is:

$$3.\ A \longrightarrow B : \{N_b\}_{k_{as}}$$

The inference rule associated with this protocol step is:

$$\frac{X}{\{X\}_{k_{as}}}$$

meaning that the intruder could supply the protocol with any information X and get it encrypted under the secret key shared between the server S and a principal A. At this level, we would like to pinpoint one important feature in our verification algorithm. The reader should notice that we used X in the inference rule instead of N_b. The rationale underlying this choice is that the principal A, at the third step of the protocol, replies with the message received at the second step (i.e. N_b) encrypted by k_{as}. Notice that N_b is a nonce generated and uttered by B and sent to A. Now, let us mention the following facts:

- The principal A has no preliminary knowledge on the effective value of the expected message at the second step of the protocol. Accordingly, A has no means to verify the value of the received message at that step.

[1] The symbol $Fresh$ denotes a unary predicate that holds whenever its argument is fresh.

- The freshness of N_b is a local property. In other words, the freshness of the nonce cannot be attested by any principal other than B. So, A cannot require the freshness of the message received at the second step of the protocol.

- For the sake of generality, we assume that the exchanged messages are not typed. Consequently, A is unable to verify the type of the message received at the second step of the protocol.

Owing to these three facts, A can verify neither the value, the freshness nor the type of the message received at the second step of the protocol. Consequently, A will accept any message at that step and will reply by sending the encrypted version of this message under the key k_{as}. This explains the rationale underlying the use of a variable message X instead of N_b. Here is the scenario that exhibits how the intruder could instrument the protocol so as to get the message $\{X\}_{k_{as}}$ provided that he supplies the protocol with X:

$$\begin{array}{llcll} 1. & A & \longrightarrow & I(B) & : A \\ 2. & I(B) & \longrightarrow & A & : X \\ 3. & A & \longrightarrow & I(B) & : \{X\}_{k_{as}} \end{array}$$

Now, let us give some general comments regarding the inference rules and the associated scenarios. First, the agent identifiers as well as the introduced variables are implicitly universally quantified. Second, the generated scenarios constitute *proofs* of the *correction* of the inference rules. Actually, the scenarios are propagated during the deduction process.

The fourth step in the Woo and Lam protocol of Table 1 is:

$$4.\ B \longrightarrow S : \{A, \{N_b\}_{k_{as}}\}_{k_{bs}}$$

The inference rule associated with this protocol step is:

$$\frac{A \quad X}{\{A, X\}_{k_{bs}}}$$

meaning that the intruder could supply the protocol with a principal identifier, say A, and a known message X so as to get an encrypted version of the composition of these two messages under the key k_{bs}. Here is the scenario that illustrates such a situation:

$$\begin{array}{llcll} 1. & I(A) & \longrightarrow & B & : A \\ 2. & B & \longrightarrow & I(A) & : N_b \\ 3. & I(A) & \longrightarrow & B & : X \\ 4. & B & \longrightarrow & I(S) & : \{A, X\}_{k_{bs}} \end{array}$$

The fifth step in the Woo and Lam protocol is:

$$5.\ S \longrightarrow B : \{N_b\}_{k_{bs}}$$

The inference rule associated with this protocol step is:

$$\frac{\{A, \{X\}_{k_{as}}\}_{k_{bs}}}{\{X\}_{k_{bs}}}$$

meaning that the intruder could supply the protocol with a message of the form $\{A, \{X\}_{k_{as}}\}_{k_{bs}}$ so as to get the message $\{X\}_{k_{bs}}$. Here is the scenario that illustrates such a situation:

$$\begin{array}{llcll} 4. & I(B) & \longrightarrow & S & : \{A, \{X\}_{k_{as}}\}_{k_{bs}} \\ 5. & S & \longrightarrow & I(B) & : \{X\}_{k_{bs}} \end{array}$$

3.3 Verification

In this section, we show how to combine roles and inference rules so as to perform the verification of a given cryptographic authentication protocol. To keep the presentation simple, we will confine ourselves to the analysis of authentication properties.

A principal A wishing to prove his identity to a principal B, has to answer all the challenges uttered by B. Hence, the intruder can impersonate the principal A if he can answer all those challenges uttered by B to A. Thus, the verification principal consists in checking whether the intruder, with all its computation abilities, can answer such challenges. Actually, the intruder computation abilities consist of:

- An initial knowledge generally made of the keys that the intruder shares with other principals, the server identity and other principal identities. Notice that this initial knowledge must be given explicitly within the protocol specification.

- The usual encryption, decryption, composition and decomposition abilities. Indeed, the intruder could encrypt and decrypt any message under known keys. In addition, he has the ability to compose (catenate) and decompose messages.

- Those computation abilities given by the protocol itself and captured by the deductive proof system.

Now we come to the explanation of the verification procedure. Starting from the protocol description, we extract the roles. Now, for each role, the intruder will attempt to answer all the challenges uttered so as to perform a masquerade. The intruder will progressively follow the role in a stepwise way. For a

given step, if the role sends a message, the intruder will merely store it and hence extending his knowledge. If the role is waiting for receiving some particular message, the intruder will attempt to generate it using his current computation abilities. In other words, the intruder will try to synthesize the required message using his current knowledge modulo some composition/decomposition and encryption/decryption steps, and also up to the use of the deductive proof system.

Now, let us see how this applies to the unidirectional Woo and Lam authentication protocol of Table 1. Notice that the only concerned role is B.

Actually, before starting to take up the various challenges uttered by B, we have first to proceed to what we call the *generalization* of the role B. This operation aims to replace some messages by message variables as done in the deductive proof system. The rationale underlying such substitutions is that these messages could not be verified from the content, structure, freshness and type standpoint. The role B is rewritten into:

$$\begin{aligned} Role(B) \quad = \quad & \langle ?, A, A \rangle \\ & \langle !, N_b, A \rangle \\ & \langle ?, X, A \rangle \\ & \langle !, \{A, X\}_{k_{bs}}, S \rangle \\ & \langle ?, \{N_b\}_{k_{bs}}, S \rangle \end{aligned}$$

Here, we replace the message $\{N_b\}_{k_{as}}$ by a message variable X since B does not know the key k_{as}, hence he could accept any other information.

First of all, let us initialize the knowledge of the intruder, noted KI, with the intruder initial knowledge. In other words, KI is set to $\{k_{is}, A, B, S\}$. This means that the intruder initially knows the identity of the server S, his shared key with S, the role A and the role B.

In the first step of the role B, the principal is waiting for a message identifier, say A. To take up this challenge, the intruder has to generate such an identifier from KI modulo some composition/decomposition and encryption/decryption steps, and also up to the use of the deductive proof system. Such a constraint is written as:

$$(E_1) \qquad KI \models_R A$$

where R stands for the set of inference rules previously mentioned together with the usual composition/decomposition and encryption/decryption rules.

At the second step, B sends the nonce N_b over the network. Hence, the message becomes available to the intruder who extends his current knowledge to become $KI \cup \{N_b\} = \{k_{is}, A, B, S, N_b\}$.

At the third step, the intruder must supply B with X. Accordingly, this second constraint will be written as:

$$(E_2) \qquad KI \cup \{N_b\} \models_R X$$

At the fourth step, B offers the message $\{A, X\}_{k_{bs}}$ to the intruder. The latter updates his knowledge to become $KI \cup \{N_b, \{A, X\}_{k_{bs}}\}$.

Finally, the intruder has to supply the role B with $\{N_b\}_{k_{bs}}$ in order to achieve the masquerade. This last constraint will be written as:

$$(E_3) \qquad KI \cup \{N_b, \{A, X\}_{k_{bs}}\} \models_R \{N_b\}_{k_{bs}}$$

At this level, we are ready to check whether the intruder can or cannot impersonate the role A. This amounts to check the existence of solutions to the constraint set $\{E_1, E_2, E_3\}$.

In the case of the constraint set $\{E_1, E_2, E_3\}$, the algorithm produces many flaws and attack scenarios. Most of these attacks are new and completely different from those known up to now. In what follows, we present in Table 2 one among the many elegant attacks yielded by the algorithm.

This attack could be read as follows: First, the intruder begins a session with B in order to prove to B that its identity is A. Second, the intruder I intercepts the nonce N_b generated by B and sent to A. Third, I responds by a message *anything* which may be any message since B cannot do any verification. Fourth, B must react according to the protocol by sending the message $\{A, anything\}_{K_{bs}}$. This message will be intercepted by the intruder. Meanwhile, the server S has no idea about what is happening. To finish this protocol run, the intruder ultimately needs the message $\{N_b\}_{k_{bs}}$ which will be sent to B at the step (1.5). The goal of the sessions 2,3 and 4 is to generate that final required message. Finally, the intruder finishes the first session with B by sending the message $\{N_b\}_{k_{bs}}$ as if it is the response of S. Hence, B is convinced that the initiator of the session one is A.

4 Conclusion

We reported in this paper a new algorithm for the formal automatic verification of authentication protocols. This algorithm does not necessitate any specification of any protocol property or invariant. It takes as parameter the protocol specification and generates the set of flaws, if any, as well as the corresponding attack scenarios. To exemplify the usefulness and

$$
\begin{array}{llcll}
1.1 & I(A) & \longrightarrow & B & : A \\
1.2 & B & \longrightarrow & I(A) & : N_b \\
1.3 & I(A) & \longrightarrow & B & : anything \\
1.4 & B & \longrightarrow & I(S) & : \{A, anything\}_{k_{bs}} \\
 & & & & \\
2.1 & C & \longrightarrow & I(D) & : C \\
2.2 & I(D) & \longrightarrow & C & : N_b \\
2.3 & C & \longrightarrow & I(D) & : \{N_b\}_{k_{cs}} \\
 & & & & \\
 & 3.1 & C \longrightarrow & I(E) & : C \\
 & 3.2 & I(E) \longrightarrow & C & : C, \{N_b\}_{k_{cs}} \\
 & 3.3 & C \longrightarrow & I(E) & : \{C, \{N_b\}_{k_{cs}}\}_{k_{cs}} \\
 & & & & \\
 & & 4.1 \quad I(C) \longrightarrow B & & : C \\
 & & 4.2 \quad B \longrightarrow I(C) & & : N'_b \\
 & & 4.3 \quad I(C) \longrightarrow B & & : \{C, \{N_b\}_{k_{cs}}\}_{k_{cs}} \\
 & & 4.4 \quad B \longrightarrow S & & : \{C, \{C, \{N_b\}_{k_{cs}}\}_{k_{cs}}\}_{k_{bs}} \\
 & & 4.5 \quad S \longrightarrow I(B) & & : \{C, \{N_b\}_{k_{cs}}\}_{k_{bs}} \\
 & & & & \\
 & & & 5.4 \quad I(B) \longrightarrow S & : \{C, \{N_b\}_{k_{cs}}\}_{k_{bs}} \\
 & & & 5.5 \quad S \longrightarrow I(B) & : \{N_b\}_{k_{bs}} \\
1.5 & I(S) & \longrightarrow & B & : \{N_b\}_{k_{bs}}
\end{array}
$$

Table 2: An Attack of the Woo and Lam Authentication Protocol

efficiency of our approach, we illustrated it on Woo and Lam authentication protocol. A prototype of this verification algorithm has been implemented in BNR-Prolog.

As future work, we intend to lift this verification algorithm to deal with key-exchange cryptographic protocols. Actually, the same principles apply to these protocols. The two first steps of our verification algorithm remain unchanged. The only step that need to be accommodated is the third verification step.

References

[1] U. Carlsen. *Formal Specification and Analysis of Cryptographic Protocols.* PhD thesis, Université de PARIS XI, France, October 1994.

[2] J. Clark and J. Jacob. A Survey of Authentication Protocol Literature. Unpublished article available at `http://dcpu1. cs. york. ac. uk/j̃eremy`.

[3] A. Liebl. Authentication in Distributed Systems: A Bibliography. *Operating Systems Review*, 27(4):122–136, October 1993.

[4] C. Meadows. Formal Verification of Cryptographic Protocols: A Survey. In *Proceedings of Asiacrypt 96*, 1996.

[5] A. D. Rubin and P. Honeyman. Formal Methods for the Analysis of Authentification Protocols. Technical Report Technical report 93–7, Technical Report, Center for Information Technology Integration, 1993. University of Michigan. Internal Draft.

[6] T. Y. C. Woo and S. S. Lam. Authentication for Distributed Systems. *Computer*, 25(1):39–52, January 1992.

[7] T. Y. C. Woo and S. S. Lam. A Lesson on Authentication Protocol Design. *Operating Systems Review*, pages 24–37, 1994.