J. Cryptology (1991) 3: 81–98

Journal of Cryptology
© 1991 International Association for Cryptologic Research

Applying a Formal Analysis Technique to the CCITT X.509 Strong Two-Way Authentication Protocol[1]

Klaus Gaarder
Norwegian Telecom Research Department, P.O. Box 83,
N-2007 Kjeller, Norway
gaarder@odin.nta.no

Einar Snekkenes
Alcatel STK Research Centre, Secure Information Systems,
P.O. Box 60, Økern, N-0508 Oslo 5, Norway
snekkene@stku01.uucp.no

Abstract. In the quest for open systems, standardization of security mechanisms, framework, and protocols are becoming increasingly important. This puts high demands on the correctness of the standards. In this paper we use a formal logic-based approach to protocol analysis introduced by Burrows *et al.* [1]. We extend this logic to deal with protocols using public key cryptography, and with the notion of "duration" to capture some time-related aspects. The extended logic is used to analyse an important CCITT standard, the X.509 Authentication Framework. We conclude that protocol analysis can benefit from the use of the notation and that it highlights important aspects of the protocol analysed. Some aspects of the formalism need further study.

Key words. Formal methods, Cryptographic protocols, Authentication.

1. Introduction

Burrows *et al.* recently published a paper suggesting a notation for the formal analysis of protocols [1]. A modified and extended version appears in [3]. The notation given in [1] is restricted to Symmetric Key Crypto Systems (SKCS) and excludes the explicit mentioning of time.

Public Key Crypto Systems (PKCS) [7], such as RSA [14], seem to be appropriate when considering authentication in open systems. This is recognized by ISO and CCITT [4]. Thus, we would expect a formalism for analysing authentication protocols not to be restricted to SKCS.

We have extended [1] to cater for PKCS and a slightly generalized notion of

[1] Date received: February 15, 1990. Date revised: October 26, 1990. This research was sponsored by the Royal Norwegian Council for Scientific and Industrial Research under Grant IT 0333.22222, and was performed while K. Gaarder was at Alcatel STK Research Centre.

time-stamp. The use of the extended formalism is demonstrated on a nontrivial example, namely a version of CCITT X.509, The Directory—Authentication Framework, Strong Two-Way Authentication.

Our first attempt to prove some properties of X.509 using [1] extended with notation to cater for PKCS failed. To prove some reasonable goals of X.509, we had to make unreasonable assumptions. Our problem occurred as a result of limitations in [1] making us unable to capture the time-stamp-related mechanisms of X.509.

The works of Burrows *et al.* [1]–[3] attempt to give general goals of authentication protocols. In many cases this seems reasonable. We emphasize the need to specify the intended goals of protocols as expressed by the protocol designers. This is illustrated in our analysis.

The rest of this section gives a summary of the formalism introduced in [1], presenting the notation for stating assumptions, goals, and messages, the rules for proving properties of assumptions and goals, and the rules for constructing proofs. Section 2 extends the formalisms to cover PKCS and certain aspects of time. In Section 3 we give an informal description of relevant parts of X.509. Section 4 gives a semiformal analysis of the strong two-way authentication protocol using the notation presented in the preceding sections. Finally, we give some concluding remarks.

1.1. *Goals, Assumptions, and Messages*

We may express the goal of our protocol analysis as some theorem of the form $\{X\}S\{Y\}$ where S is the list of protocol steps, X is the collection of initial assumptions, and Y is the goal of the protocol. Goals, assumptions, and messages consist of formulae constructed by means of the symbols listed below.

It is convenient to treat our approach to protocol analysis as an application of formal logic. Well-formedness and syntactic derivation rules are given below. As with first-order predicate calculus, it is possible to write down an inconsistent set of formulae giving rise to potential trivial and uninteresting derivations, see [13]. It does not seem reasonable to infer that from the existence of uninteresting derivations we have an ill-defined logic.

Let X denote some formula and, for all i, let U_i denote some communicating entity. Below we list the well-formed formulae together with their informal semantics:

$U_i \models X$	U_i *"believes"* X. If $U_i \models X$ holds, then U_i will behave as if X is true in her interpretation.
$U_i \mid\sim X$	U_i *"once said"* X. $U_i \mid\sim X$ typically holds if U_i has sent a message Y having X as a subformula.
$U_i \Rightarrow X$	U_i *"has jurisdiction over"* X. If you believe U_i has jurisdiction over X and that U_i believes X, then you ought to believe X.
$U_i \lhd X$	U_i *"sees"* X. Typically U_i sees X if she has received a message Y, where X is a subformula of Y.
$\#(X)$	X *"is fresh."* A formula is fresh if it has not previously occurred as the subformula of some sent formula. A formula generated for the purpose of being fresh is usually called a *nonce.*

$U_i \overset{K}{\leftrightarrow} U_j$	U_i and U_j "*share the good crypto key*" K, where K is a symmetric crypto key.
$\{X\}_K$ *signed* U_i	X "*encrypted under the symmetric key*" K "*by*" U_i. Notice that although both parties sharing the key may produce a signed message, the formula only holds if U_i was in fact the one which signed the message.

1.2. *Axioms and Inference Rules*

Below we list the inference rules of the original logic. Assume $X_1, \ldots, X_n$, Y are formulae as above, then the rules below have the following format:

$$\text{Rm.} \quad \frac{X_1, \ldots, X_n}{Y},$$

read as "if X_1 and ... and X_n, then Y." Basically, the rules make explicit the intuitive consequences of semantics given above.

It is worth noting that we assume the existence of an implicit rule:

$$\text{R0.} \quad \frac{\Psi}{\Psi^L_{L'}},$$

where L and L' are permuted lists of formulae, and Ψ is some formula containing the formula list L, and $\Psi^L_{L'}$ is Ψ with L' substituted for L.

Let U_i, U_j denote communicating entities, let X, Y denote formulae, and let K be a symmetric crypto key. Assuming you see a message encrypted under some good key of an SKCS, then you ought to believe that the unencrypted message was uttered by the only other entity knowing the key. This is formalized as

$$\text{R1.} \quad \frac{U_i \models U_j \overset{K}{\leftrightarrow} U_i,\ U_i \lhd (\{X\}_K \textit{ signed } U_j)}{U_i \models U_j \mid\!\sim X}.$$

Note that entities are not allowed to change their beliefs during a run (R2):

$$\text{R2.} \quad \frac{U_i \models \#(X),\ U_i \models U_j \mid\!\sim X}{U_i \models U_j \models X}.$$

If you believe U_j has jurisdiction over X, and you believe that U_j believes X, then you believe X (R3):

$$\text{R3.} \quad \frac{U_i \models U_j \Rightarrow X,\ U_i \models U_j \models X}{U_i \models X}.$$

Rules R4–R7 and R10 make it possible to break up and aggregate formulae. Their intuitive justification should be obvious:

$$\text{R4.} \quad \frac{U_i \models X,\ U_i \models Y}{U_i \models (X, Y)},$$

$$\text{R5.} \quad \frac{U_i \models (X, Y)}{U_i \models X},$$

R6. $\dfrac{U_i \models U_j \models (X, Y)}{U_i \models U_j \models X}$,

R7. $\dfrac{U_i \models U_j \mid\!\sim (X, Y)}{U_i \models U_j \mid\!\sim X}$,

R10. $\dfrac{U_i \lhd (X, Y)}{U_i \lhd X}$.

Clearly, if you have not seen any message X, you cannot have seen any messages where X occurred as a subformula. In particular we have

R12. $\dfrac{U_i \models \#(X)}{U_i \models \#(X, Y)}$.

There are other rules, but the above are the most commonly used ones.

1.3. *Protocol Annotation*

When analysing a protocol, we first derive an *idealized* version of the protocol S, then state assumptions A and goals B, and finally attempt (and hopefully succeed) to prove $\{A\}S\{B\}$ as a theorem using the annotation rules below. An idealized protocol consists of a sequence of steps, each of the form $(U_1 \rightarrow U_2 : X)$, where U_1, U_2, X is sender, recipient, and message, respectively. The message X is expressed as a formula in the above notation.

There are four annotation rules (A1–A4). The annotation rules are presented in the familiar Hoare style [10]. The first rule states that the effect of executing a protocol step is that the message sent becomes visible to the recipient:

A1. $\vdash \{Y\}(U_i \rightarrow U_j : X)\{Y, (U_j \lhd X)\}$.

The collection of annotated protocols is in a certain sense transitive, as shown below:

A2. $\dfrac{\vdash \{X\}S_1 \cdots \{Y\},\ \vdash \{Y\}S_1' \cdots \{Z\}}{\vdash \{X\}S_1 \cdots \{Y\}S_1' \cdots \{Z\}}$.

Intermediate assertions and the conclusion may be weakened (A3):

A3. $\dfrac{\vdash \{W\}S_1 \cdots \{X_1\} \cdots \{X_n\},\ X_i \vdash X_i'}{\vdash \{W\}S_1 \cdots \{X_1\} \cdots \{X_i'\} \cdots \{X_n\}}$.

We may always make consequences of the assumption explicit:

A4. $\dfrac{\vdash \{X\}S \cdots \{Y\},\ X \vdash X'}{\vdash \{X, X'\}S \cdots \{Y\}}$.

2. The Extensions

In this section we describe the modifications needed in the logic to be able to reason about PKCS and certain aspects of time. For each, we give the necessary symbols, their informal semantics, and the corresponding inference rules.

2.1. *Public Key Crypto Systems*

In a PKCS we assume the existence of a nonempty collection of communicating entities. Each entity has associated one public and one private key. We only consider signature properties, leaving secrecy issues aside. To check that a message m was signed by U, it is sufficient to know U's public key. When using this approach, it is paramount that U's public key is genuine. However, to sign some message with U's signature, we must be in possession of U's private key. Usually, only U knows U's private key. Below we list the symbols together with their informal semantics:

$\mathscr{PK}(K, U)$	The entity U "*has associated the good public key*" K. Thus, there exists some unique key corresponding to K.
$\Pi(U)$	The entity U "*has associated some good private key.*" Consequently, the key value is only known by U.
$\sigma(X, U)$	The formula X "*signed with the private key belonging to*" U.

The above informal semantics lead to the rule:

$$\text{R13.}\quad \frac{U_i \models \mathscr{PK}(p_j, U_j),\ U_i \models \Pi(U_j),\ U_i \lhd \sigma(X, U_j)}{U_i \models U_j \mid\sim X}.$$

Thus, to believe that U_j has once said X, it is sufficient to believe that we have U_j's public key, that U_j's secret key is good, and we must see X signed with U_j's private key.

The content of a signed message can always be made visible:

$$\text{R14.}\quad \frac{U_i \lhd \sigma(X, U_j)}{U_i \lhd X}.$$

2.2. *Time*

Using the notion of freshness described above, we may well have a fresh formula which was generated at some arbitrary time in the past. Thus, a fresh message might still be very old. Similarly, a very recent message is only fresh if it has not been sent before. Thus, we stipulate that recency and freshness are different concepts. The authors of [1] claim that their logic benefits from avoiding the explicit notion of time. As previously stated, we have to include the notion of time to be able to prove certain properties of X.509. Thus, insisting on using [1] to certify protocols removes an important mechanism from the protocol designers repertoire, with the effect that protocols may become unnecessarily complex and hard to design.

A time-stamp added to a message usually states the time the message was generated [6]. It seems reasonable to assume that a time-stamped message was in some sense *good* at the time of generation. From this we develop the notion of a duration-stamp. A message might thus be tagged with the duration-stamp denoting the time interval during which its creator claims the message is good.

The symbols needed are listed below together with their informal semantics:

$(\Theta(t_1, t_2), X)$	X "*holds in the interval*" t_1, t_2. Thus, the creator which uttered the time-stamped message X, claims that X is, or was, good in the time interval between t_1 and t_2.

$\Delta(t_1, t_2)$ t_1, t_2 "*denotes a good interval.*" Thus, the local unique Real Time Clock (RTC) shows a time in the interval between t_1 and t_2.

Obviously we may model a time-stamp, namely $(\Theta(t, t), X)$. Our semantics above is based on the following assumptions:

1. A process making the explicit reference to some RTC in some message using a duration-stamp, does so with respect to his own unique local RTC.
2. The duration of any protocol run must be short.
3. When tagging a message subsequently uttered in a protocol step using a duration-stamp, e.g., $(\Theta(t_1, t_2))$, we commit ourselves to believe the message X to be correct in the time interval specified by t_1 and t_2.

Recall that formulae of [1] are only assumed to stay true for the duration of a single run of the protocol. However, using the above, we are able to commit ourselves for future runs. Later, it is shown that this is exactly what is needed when reasoning about the certificates of X.509.

Based on the assumptions above, we give a rule for reasoning about duration-stamps:

$$\text{R15.}\quad \frac{P \models Q \models \Delta(t_1, t_2),\ P \models Q \mid\!\sim (\Theta(t_1, t_2), X)}{P \models Q \models X}.$$

The rule states that when uttering a duration-stamped message, we commit ourselves to believe the message for the interval specified by the duration-stamp. If we use Coordinated Universal Time (UTC time),[2] or synchronized clocks, the first premise of R15 may be established by inspecting the local RTC.

3. An Informal Description of X.509

The central items in the X.509 messages are the data structures known as *certificate* and *token*, which are the basis of authentication together with *distinguished name.*

Certification Authority (CA). A trusted entity maintaining security information in the directory[3] (issuing certificates, etc.).

Distinguished Name. Each entity has a unique *distinguished name* which is allocated by the *naming authority.*

Signature (as defined in the X.509 standard). Signing a message M is performed by encrypting a hashed version of M under the signer's *private* key and appending it to the message M,

$$\langle M \rangle_U \overset{\text{def}}{=} M \cdot E(Z_U, h(M)), \tag{1}$$

[2] See CCITT Recommendations X.208 and X.209.

[3] The X.500 Directory is a database.

where "·" denotes concatenation, E is an encryption function, h is a hash function, and Z_U is the private key of the signer U.[4]

Certificate. The certificate is a data structure signed by the *certification authority*, connecting a distinguished name to a *public key* of some PKCS. Its structure is as follows: let A be an algorithm identifier, let I be the name of the *issuer*, let δ be a *duration* period consisting of two dates (from, to), let U be the name of the *owner*, and let P_U be the *public key* of U, then the certificate of U is

$$C_U = \langle (A, I, \delta, U, P_U) \rangle_I. \tag{2}$$

Token. The token is a signed data structure, where t_P, P, R_P, Q, R_Q denotes the *generation time* and *expiry date* in UTC time, name of the *sender*, a random number generated by P, name of the *receiver*, and a random number generated by Q, respectively. The token in its most general form contain two further pieces of data: *sgnData* and $E(p_Q, \mathit{encData})$ (*encData* encrypted with the receiver's public key). *encData* may be a secret key for some SKCS, typically a DES key. The token has two forms:[5]

$$G_A = \langle (t_A, R_A, B, \mathit{sgnData}, E(p_B, \mathit{encData})) \rangle_A \tag{3}$$

or

$$G_B = \langle (t_B, R_B, A, R_A, \mathit{sgnData}, E(p_A, \mathit{encData})) \rangle_B. \tag{4}$$

The form (4) of the token is used in the reply to the first form (3).

Basic Idea of X.509. By possessing the public key of the certification authority, the entities in question will be able to verify the signature and timeliness of the certificates and thus be convinced that it contains the good public key of the desired communication partner. By using this public key each entity will verify the token received from the other (which was produced using the sender's private key), thus authenticating the sender.

The precise role of the certificates is thus to supply trusted information to the authenticating parties (*their respective public keys*), which in turn will enable them to do the signature verification which constitutes the authentication proper.

We have chosen to view the certificate exchange between A and B as part of the X.509 protocol, mainly for two reasons:

- Completeness; certificates are an integral part of authentication in this framework.
- Certificates are part of the messages being transferred.

Some workers have chosen to ignore the initial handling of certificates [2], assuming they have been acquired and checked previously. We argue that important aspects of X.509 are then lost.

[4] The hash function suggested for use in [4] has recently been shown to contain weaknesses [5].
[5] See X.509 9.2.2 and X.509 9.3.5.

We consider only what is known as "two-way strong authentication." The other modes are "one-way" and "three-way." Without loss of generality the certification path is considered to be of length one, since a certification path is simply a sequence of certificates.[6] Then each user needs only one certificate, and this is supplied to both by the same certification authority (CA).

3.1. *Strong Two-Way Authentication*

The participants in the protocol are the parties seeking to authenticate (A, B) and the Certification Authority (C). Subsequently any item subscripted by P is to be interpreted as in some sense belonging to P. The following messages are passed:

Message 1: $A \rightarrow C$: (A, B). This is the *certificate request*.

Message 2: $C \rightarrow A$: (C_A, C_B). C forwards A's and B's certificates to A. A checks the signature of C on C_B by using the public key of C. If the signature check succeeds, then A proceeds.

Message 3: $A \rightarrow B$: (C_A, G_A). A forwards her certificate to B together with a *token* of the form (3) above. On receipt of this message B will check the signature of C_A, and if it is correct proceed to check the signature on G_A. If B successfully applies A's public key (received in C_A) to verify the signature on G_A, he will proceed to check the validity of the contents.

Message 4: $B \rightarrow A$: (G_B). This is B's reply, a token of the form (4). A performs checks as B did above (also checking the returned part of her own token).

4. A Semiformal Analysis of the Two-Way X.509 Protocol

Evidently any protocol has a number of possible interpretations. The X.509 recommendation is no exception. The analysis presented in [2] takes a view differing from ours, by assuming that certificates have been distributed to the participants in advance and that they have been checked and accepted as valid. We choose to assume only that each participant has acquired the public key of a common certification authority (a common point of trust), and take the checking of certificates for participants as an integral part of the authentication protocol. This forces us to consider the logic's ability to reason about the certificates and their contents. In the general case of communicating with someone for the first time this is what must happen. At this point our approach differs from previous analyses of X.509 using this logic.

We choose to arrive at the goals of an authentication protocol from the description of the protocol and the statements from the authors about what it should achieve. Originally [2] proposed some generic goals involving shared secret keys.

[6] A certification path is defined in [4], to establish a path of trusted points in the DIT, see Section 7.5 of [4].

The important thing is not to prove that a protocol achieves some generic goals but that it achieves the goals set out by the designers, since this is basically what the users of the protocol can expect.

4.1. *The Assumptions*

In a protocol we often include *names* of the principals in certain messages, like in

$$A \rightarrow B\colon (B, X), \tag{5}$$

where *the semantics of the protocol itself* is such that the inclusion of the name B in the message is interpreted as "B is the intended recipient of X." (In the X.509 protocol this is explicitly stated as one of the goals achieved by the protocol.) Note that this is a semantic property of the *combination* of a name (B) and a message (X). The original BAN logic has no construction to handle this and we see no means of expressing it in the language available.

To remedy this we introduce a special symbol for the notion of "recipient" of a message. We write

$$\mathscr{R}(P, X) \tag{6}$$

to say that "*P is the intended recipient* of a message X." Normally this will appear in the following form:

$$P \rightarrow Q\colon \mathscr{R}(Q, X), X, \tag{7}$$

where P sends to Q a message X together with the *tag* $\mathscr{R}(Q, X)$ telling that Q is the intended recipient of X.

From the X.509 text it is clear that each principal must be assumed to have some kind of jurisdiction over statements involving this construction, at least in the following sense. If Q believes that P has said $(\mathscr{R}(Q, X), X)$ in this protocol run this will lead Q to believe that X is indeed meant for her. However we will not allow Q to conclude anything if what P said was $\mathscr{R}(O, X)$ (provided $O \neq Q$). That is, you only accept as meant for you the messages which contain your name in the recipient field. Thus, you cannot conclude that a message is not meant for you.[7]

As noted above $\mathscr{R}$ may be thought of as a "tag" on a parcel. The reasons for introducing $\mathscr{R}$ can be illustrated with an example. If you receive an invoice in your mailbox (be it electronic or otherwise) your further action will clearly depend upon whether you believe you are the intended recipient of this invoice or not! Adding $\mathscr{R}$ to the logic, we may still use the inference rules described above. In particular, the freshness propagation also applies to $\mathscr{R}$.

We recognize seven assumptions in the description of the X.509 protocol. All but a few assumptions are easy to read directly from the text. α_2 reflects the jurisdiction over intended recipients as noted in the introduction of $\mathscr{R}$. α_7 makes explicit the time dependency of the certificates.

All participants believe that *everyone keeps their private key private*, including their own, X.509-6.4:

$$\alpha_1\colon \qquad \{A \models \Pi(B), B \models \Pi(A)\}. \tag{8}$$

[7] Since the logic has no negation connective "$\neg\mathscr{R}(U, X)$" is not a well formed formula.

A and B trust each other on telling who is the intended recipient (X.509-9.1.2.a.1, 9.1.2.b.1),

$$\alpha_2: \qquad \{A \models B \Rightarrow \mathcal{R}(A, X), B \models A \Rightarrow \mathcal{R}(B, X)\}. \tag{9}$$

Both A and B believe they have the correct public key of C, the authority (X.509-7.1):[8]

$$\alpha_3: \qquad \{A \models \mathcal{PK}(p_C, C), B \models \mathcal{PK}(p_C, C)\}. \tag{10}$$

A and B trust the certification authority on (*public-key*, *name*) pairs (X.509-7.1, X.509-7.2):

$$\alpha_4: \qquad \{B \models C \Rightarrow \mathcal{PK}(p_A, A), A \models C \Rightarrow \mathcal{PK}(p_B, B)\}. \tag{11}$$

A (B) believes in the freshness of the opponent's time-stamp (X.509-9.3.6.c, 9.3.3.d, respectively):

$$\alpha_5: \qquad A \models \#(t_B), \tag{12}$$

$$\alpha_6: \qquad B \models \#(t_A). \tag{13}$$

The principals believe that the certification authority believes that the duration δ_P of P's certificate is still good, i.e., that C will not deliver certificates with invalid duration periods, X.509-7.2. This is a critical assumption:

$$\alpha_7: \qquad \{A \models C \models \Delta(\delta_B), B \models C \models \Delta(\delta_A)\}. \tag{14}$$

Comments on Assumption α_7. Since the certificate does not contain any *time-stamp* or random number making it a *nonce* (i.e., having the "fresh" property) we are not able to use the rule of jurisdiction to deduce

$$P \models \mathcal{PK}(p_Q, Q)$$

from assumptions α_1–α_6. Thus, the X.509 protocol would seem not to achieve its primary goals. There are at least two ways to remedy this.

One way is to insert a time-stamp in the certificate in addition to the duration, making it necessary to fetch the certificates from the CA, with a fresh time-stamp, each time they are needed. Essentially this means generating a new certificate each time a request is made. This is exactly the problem we seek to avoid for performance reasons. To go for this solution would be to say that *the formalism* as it stands captures every aspect of the protocol, and claim that there is a deficiency in *the protocol.*

Another possible solution is the one we have outlined. We argue that the duration period certainly increases the security of the protocol, and that the formalism could not capture this notion of time dependence. Therefore we chose to include assumption α_7, and the concept of a "good duration period," $\Delta(\delta)$. Since all participants are assumed to use UTC time, it is easy to check $\Delta(\delta)$. Note that *clocks have to be trusted if this is to provide additional security.*

[8] We suppress the fact of how A and B got hold of p_C. This is clearly not a part of the X.509 protocol.

When trying to prove Γ_1–Γ_8 without the Δ relation, we might be misled in assuming that certificates have the "fresh" property. This is wrong since certificates are expected to be used in a large number of protocol runs.

It also reflects the fact that the "fresh" notion is not basically a measure of *time*, just a stating of the fact that something has not been said in any previous run of the protocol.

4.2. *The Idealized Protocol*

The most critical part of protocol analysis using a method of this kind is the formalization of the protocol steps. It is a feature of the method that we cannot carry out any formal proofs to justify the correctness of our (rather subjective) idealization. The exact syntactic form we give the messages will entirely determine the outcome of the proofs. This is simply the nature of a formal method: proofs are devoid of semantics and reduce to purely syntactical manipulations on the formulae available. To be able to use the inference rules we are dependent upon having available formulae which may be used as premises in these rules. This is in turn completely decided by the syntactic form of these formulae, not by their intended meaning.

Certificate. The formalization of the certificate and token are critical aspects of the analysis. The key to the formalization is the implicit meaning of each message and its contents.

The important features of the certificate are:

- the signature of the certification authority,
- the duration,
- the name of the owner,
- the public key of the owner.

If $\delta_P = (t_1^P, t_2^P)$ is the duration of C_P we write $\Theta(\delta_P)$ for $\Theta(t_1^P, t_2^P)$. Recall that $(\Theta(\delta_P), X)$ is intended to mean that X holds in the interval δ_P. Then we give the certificate the formalization

$$\sigma((\Theta(\delta_P), \mathscr{PK}(p_P, P)), C). \tag{15}$$

Token. As with the certificate the important features are reflected in the formalization, these are:

- time-stamp,
- random number,
- name of the intended recipient,
- signed cleartext data, X_A, X_B (optional),
- secret data, Y_A, Y_B (optional),
- signature of the issuer.

It is clear that it is the intention of the designers that the time-stamp and (or) random number should give the token the property of a nonce.

Let T_A (3) and T_B (4) be defined as

$$T_A = (t_A, R_A, X_A, \{A \overset{Y_A}{\leftrightarrow} B\}_{p_B}), \tag{16}$$

$$T_B = (t_B, R_B, R_A, X_B, \{B \overset{Y_B}{\leftrightarrow} A\}_{p_A}). \tag{17}$$

Here X_Q (Y_Q) denotes the "sgnData" ("encData") of Q. In the idealized protocol the token takes the form of a signed message with a "tag" telling who is the intended recipient, $\mathscr{R}(T_A, B)$, $\mathscr{R}(T_B, A)$.

The formalization of messages 2 (X.509-7.7) and 3 (X.509-9.3.1, 9.3.2) are given by (15) and (16) above, message 4 (X.509-9.3.4, 9.3.5) is given by the form of the token in (17).

Message 1. Exactly how we formalize message 1 (X.509-7.7) is of little importance to the subsequent analysis:

$$A \rightarrow C: (A \mid\sim (A, B)). \tag{18}$$

Message 2.

$$C \rightarrow A: \sigma((\Theta(\delta_A), \mathscr{PK}(p_A, A)), C), \sigma((\Theta(\delta_B), \mathscr{PK}(p_B, B)), C). \tag{19}$$

Message 3. The idealization of this message is critical, and requires some explaining. Concerning checking of certificates, [4] states that A will check the validity of both A's and B's certificate before proceeding with message 3. She would *not* proceed if B's certificate was invalid, so if she proceeds this is an implicit statement of the following belief:

$$A \models \mathscr{PK}(p_B, B),$$

which is in fact conveyed to B via A's token giving

$$A \rightarrow B: \sigma((\Theta(\delta_A), \mathscr{PK}(p_A, A)), C), \sigma((T_A, \mathscr{R}(B, T_A), \mathscr{PK}(p_B, B)), A). \tag{20}$$

Message 4. An argument similar to above yields

$$B \rightarrow A: \sigma((T_B, \mathscr{R}(A, T_B), \mathscr{PK}(p_A, A)), B). \tag{21}$$

4.3. *The Goals of Authentication in X.509*

Burrows *et al.* suggest a set of goals for authentication which any protocol is evaluated against. This might lead us to believe that it is possible to formulate general goals of authentication regardless of the mechanisms used to achieve it. The goals presented by [1] are strictly related to SKCS, since they are formulated as statements about *share symmetric crypto keys.* Such a notion neither exists in a PKSC nor in a Zero Knowledge Interactive Proof system (ZKIP), see, e.g., [8], and [9]. In fact, in a ZKIP-based protocol no key in the usual sense need exist at all.

We have chosen a slightly different approach. Every protocol is designed to achieve certain goals, and the user of any protocol should be aware of this. So if a protocol design claims to achieve only this much, then we cannot expect it to achieve more, but we do expect it to achieve what it claims! Until we have an agreed version of the meaning of *authentication*, authentication protocols may be expected to achieve slightly differing goals. We have approached the analysis of X.509 by formalizing the goals which [4] claims to achieve, and trying to prove these. This approach results in a rather long list of goals.

Formal Goals of X.509.[9] These are the goals of the X.509 strong two-way authentication:

- Γ_1–Γ_2. After a run of the protocol each of the parties should believe that they have a valid public key belonging to the other.
- Γ_3–Γ_4. The beliefs of the previous goals should be mutual.
- Γ_5–Γ_6. These goals are less intuitive. They state the following fact: A should end up believing that B recently said "A is the intended recipient of token T_B," i.e., that the token received was produced for A by B during the current run (Γ_5). Similarly for B (Γ_6).
- Γ_7–Γ_8. Both should end up believing they are the legitimate receivers of the respective tokens.
- Γ_9–Γ_{10}. Optionally A (B) should end up believing that B (A) believes X_B (X_A).
- Γ_{11}–Γ_{12}. Optionally A (B) should end up believing in the mutual secrecy of part of the token.

We formalize these goals as:

$$\Gamma_1: \quad A \models \mathscr{PK}(p_B, B), \tag{22}$$

$$\Gamma_2: \quad B \models \mathscr{PK}(p_A, A), \tag{23}$$

$$\Gamma_3: \quad A \models B \models \mathscr{PK}(p_A, A), \tag{24}$$

$$\Gamma_4: \quad B \models A \models \mathscr{PK}(p_B, B), \tag{25}$$

$$\Gamma_5: \quad A \models ((B \mid\sim (\mathscr{R}(A, T_B), T_B)), \#(\mathscr{R}(A, T_B))), \tag{26}$$

$$\Gamma_6: \quad B \models ((A \mid\sim (\mathscr{R}(B, T_A), T_A)), \#(\mathscr{R}(B, T_A))), \tag{27}$$

$$\Gamma_7: \quad A \models \mathscr{R}(A, T_B), \tag{28}$$

$$\Gamma_8: \quad B \models \mathscr{R}(B, T_A), \tag{29}$$

$$\Gamma_9: \quad B \models A \models X_A, \tag{30}$$

$$\Gamma_{10}: \quad A \models B \models X_B, \tag{31}$$

$$\Gamma_{11}: \quad A \models A \overset{Y_B}{\leftrightarrow} B, \tag{32}$$

$$\Gamma_{12}: \quad B \models B \overset{Y_A}{\leftrightarrow} A. \tag{33}$$

We notice the similarity of goals Γ_1–Γ_4 with the goals presented in [1].

It turns out that we are not able to prove Γ_{11} and Γ_{12}, the reason being that the sender of the secret does not provide any evidence of his (her) knowledge of the

[9] The formulation of these goals may be found in the sections of [4] as follows: Γ_1–Γ_4: 7.1; Γ_5: 9.1.2.b.1, 9.1.2.b.2; Γ_6: 9.1.2.a.1, 9.1.2.a.3; Γ_7: 9.1.2.b.1; Γ_8: 9.1.2.a.2; Γ_9: 9.1.2.a; Γ_{10}: 9.1.2.b; Γ_{11}, Γ_{12}: 9.1.2.b.3.

value of the secret. A similar result is obtained in [3]. For the purpose of this analysis, we consider the notions of shared secrets and shared keys as equivalent (Γ_{11}, Γ_{12}). We note that [3] distinguishes between the two concepts.

4.4. *Proof Outline*

Conducting the proof itself is rather easy once the assumptions, goals, and idealized protocol are established. At each step only a few inference rules are applicable, making it easy to decide which rules to apply.

The all important thing to note in the proof outline is where and how the different goals appear, and the use of the assumptions. It turns out that the assumptions concerning time aspects are critical. As we have seen, the use of time in X.509 is one of the primary causes for the need to extend the logic.

We state the following claim:

Claim 1. *With assumptions α_1–α_7 the above idealized protocol attains goals Γ_1–Γ_{10}.*

Proof. *Goals Γ_1–Γ_2.*

Γ_1: $$A \models \mathcal{PK}(p_B, B),$$

Γ_2: $$B \models \mathcal{PK}(p_A, A).$$

These goals are proven using the result of messages 2 and 3, which are

$$A \lhd \sigma((\Theta(\delta_B), \mathcal{PK}(p_B, B)), C), \tag{34}$$

$$B \lhd \sigma((\Theta(\delta_A), \mathcal{PK}(p_A, A)), C) \tag{35}$$

by using assumptions $\alpha_1, \alpha_3, \alpha_4$, and α_7. Without rule R15 and the assumption that the certification authority does not hand out certificates with bad duration (α_7) we could not have proved this.

Let

$$M_A = (T_A, \mathcal{R}(B, T_A), \mathcal{PK}(p_B, B)),$$

$$M_B = (T_B, \mathcal{R}(A, T_B), \mathcal{PK}(p_A, A)),$$

then we have the following lemma:

Lemma 1. *From assumptions α_1–α_7 the above idealized protocol attains*

γ_A: $$A \models B \models M_B,$$

γ_B: $$B \models A \models M_A.$$

Proof. Message 4 gives us

$$A \lhd \sigma(M_B, B). \tag{36}$$

Using rules R13 and R14 with assumptions $\alpha_1, \alpha_5, \alpha_6$ we first prove

$$A \models B \mid\sim M_B. \tag{37}$$

Then by R12 and α_1 we get

$$A \models \#(M_B). \tag{38}$$

Finally, by R2 we have

γ_A:
$$A \models B \models M_B. \tag{39}$$

Similarly, by using message 3 we can obtain a proof of γ_B. □

By using the lemma above and the projection of beliefs (R6) we obtain

Γ_3:
$$A \models B \models \mathcal{PK}(p_A, A), \tag{40}$$

Γ_4:
$$B \models A \models \mathcal{PK}(p_B, B). \tag{41}$$

From (37) and the freshness propagation over $\mathcal{R}$ we obtain

Γ_5:
$$A \models ((B \mid\!\sim (\mathcal{R}(A, T_B), T_B)), \#(\mathcal{R}(A, T_B))) \tag{42}$$

by the belief aggregation rule (R4). Similarly, we can obtain a proof of Γ_6. Again, by the belief projection rule (R6) and the lemma above, followed by the jurisdiction rule (R3) with α_2 we obtain

Γ_7:
$$A \models \mathcal{R}(A, T_B),$$

Γ_8:
$$B \models \mathcal{R}(B, T_A).$$

Finally, by the belief projection rule and the lemma we obtain

Γ_9:
$$B \models A \models X_A,$$

Γ_{10}:
$$A \models B \models X_B.$$ □

Note that we were unable to prove Γ_{11} and Γ_{12}. In particular, from the lemma (by R6) we can easily obtain

$$A \models B \models \{B \overset{Y_B}{\leftrightarrow} A\}_{P_A}. \tag{43}$$

Even if $\{B \overset{Y_B}{\leftrightarrow} A\}_{P_A}$ is contained in a signed message, the encryption makes it impossible for A to deduce that B is in possession of the cleartext (Y_B). It turns out that neither Γ_{11} nor Γ_{12} can be established unless we modify the protocol (e.g., by including the plaintext Y_i as input to the hash function, see [11] for details) and introduce more assumptions (such as trust in the others ability to generate good secrets; i.e., $A \models B \Rightarrow A \overset{Y_B}{\leftrightarrow} B$ and $B \models A \Rightarrow \{A\}_{Y_A}$ *signed* B).

5. Conclusions

We have shown that formal proofs in protocol analysis using the formalism presented is feasible, even without machine support. Also, the proofs has given additional insight into the workings of the protocol.

Using a formal notation, we have clarified very important assumptions for X.509 to succeed: in particular, the requirement of having knowledge about the current time of the certification authority. Also our idealization of the certificates emphasize the fact that the certification authority *must* commit himself when signing the certificates. The suggestion to use blacklists in [4] reflects the view that this commitment might be too strong.

The crucial issue of formalizing goals, assumptions, and idealizing the protocol was not always intuitively obvious. This has also been noted by Meadows [12]. However, the formalization forced us to consider details which would otherwise have been ignored.

As we have shown in the preceding sections, we can divide the task of protocol design into four distinct activities:

- Identify the set of goals (Γ).
- Identify the set of assumption (α).
- Find a protocol (P) which is intended to satisfy goals (Γ) whenever the assumptions (α) holds.
- Construct a correctness argument.

When P is shown to be correct, the designers task is to ensure that the explicit assumptions α and the implicit assumptions (built into the logic) can be made to hold. Furthermore, they must ensure that the protocol idealization correctly reflects the state of affairs. Thus, the BAN approach to protocol analysis is consistent with what has become good software engineering practice, namely identifying several layers of abstraction.

The approach taken is rather abstract, thus we do not increase confidence in the actual mechanisms. Consequently we have *not* shown that the mechanisms used to implement X.509 cannot be compromised. Thus, our results are not inconsistent with [5].

In our presentation we have focused on the use of the notation rather than the notation itself. The formal semantics of the notation is the topic for further study.

We hope our extensions have widened the scope of the notation, making its application an interesting approach for the protocol analyst and designer.

Acknowledgments

We would like to thank the anonymous referees and our colleagues at Alcatel STK Research Centre, Secure Information Systems for helpful suggestions. We would also like to thank Michael Burrows and Martín Abadi for helpful comments and for making us aware of their recent work [2]. Special thanks to Mr. Kåre Presttun at the research centre for his encouragement and continued support.

Appendix. The Formal Proof of Two Goals

The proof proceeds line-by-line as described below:

$$\varphi \quad \beta \quad \textit{comment} \tag{44}$$

means φ is proved using the rules or previous theorems referred to in the string β. If $\beta = n_1, n_2, \ldots, n_k, Rj$, then φ is proved by taking the formulae in lines $n_1, \ldots, n_k$ as premisses in the rule Rj. From one line to the next only one inference rule is used at a time.

We note that the proof layout differs from the annotations of [2], but this is only a cosmetic difference. Our proof may easily be recast into that form.

A.1. *The Proof.*

The formulae $A \models \mathcal{PK}(B, p_B)$, $B \models \mathcal{PK}(A, p_A)$ have been proved in Section 4. Recall that we have introduced the abbreviation

$$M_A = (T_A, \mathcal{R}(T_A, B), \mathcal{PK}(p_B, B)).$$

From message 3 we obtain by use of annotation rule A1 (ε denotes the empty sequence)

$$B \lhd \sigma(M_A, A) \quad \varepsilon \quad \text{A1}, \tag{1}$$

$$B \models \mathcal{PK}(p_A, A) \quad \Gamma_2 \quad \text{proved above}, \tag{2}$$

$$B \models \Pi(A) \quad \alpha_6 \quad \text{assumption}, \tag{3}$$

$$B \models A \mid\!\sim (M_A) \quad 1, 2, 3, \text{R13} \quad \text{sign. verif.}, \tag{4}$$

$$B \models \#(t_A) \quad \alpha_6 \quad \text{fresh “time-stamp,”} \tag{5}$$

$$B \models \#(T_A) \quad 5, \text{R12} \quad \text{propagating } \#, \tag{6}$$

$$B \models \#(M_A) \quad 6, \text{R12} \quad \text{propagating } \#, \tag{7}$$

$$B \models A \models M_A \quad 7, 4, \text{R2}. \tag{8}$$

The way we have defined M_A lets us obtain the rest by projection,

$$B \models A \models T_A \quad 8, \text{R5}, \tag{9}$$

$$B \models A \models \mathcal{R}(T_A, B) \quad 8, \text{R5}, \tag{10}$$

$$B \models A \Rightarrow \mathcal{R}(T_A, B) \quad \alpha_2 \quad \text{instance}, \tag{11}$$

$$B \models \mathcal{R}(T_A, B) \quad 10, 11, \text{R3} \quad = \Gamma_8, \tag{12}$$

$$B \models A \models X_A \quad 9, \text{R5} \quad = \Gamma_9, \tag{13}$$

$$B \models A \models \{A \overset{Y_A}{\leftrightarrow} B\}_{p_B} \quad 9, \text{R5}. \tag{14}$$

This last line is as close as we get to proving Γ_{12} without more assumptions or a changed protocol.

References

[1] M. Burrows, M. Abadi, and R. Needham. Authentication: A Practical Study in Belief and Action. Technical Report 138, University of Cambridge Computer Laboratory, 1988.

[2] M. Burrows, M. Abadi, and R. Needham. A Logic of Authentication. Technical Report 39, Digital Systems Research Center, 1989.

[3] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. *ACM Transactions on Computer Systems*, **8**(1): 18–36, February 1990.

[4] CCITT. *CCITT Blue Book, Recommendation X.509 and ISO* 9594-8, *Information Processing Systems—Open Systems Interconnection—The Directory—Authentication Framework*. Geneva, March 1988.

[5] D. Coppersmith. Analysis of ISO/CCITT Document X.509 Annex D. IBM Thomas J. Watson Research Center, Yorktown Heights, June 1989.

[6] D. E. Denning and G. M. Sacco. Timestamps in key distribution protocols. *Communications of the ACM*, **24**(28): 533–536, 1981.

[7] W. Diffie and M. E. Helleman. New Directions in cryptography. *IEEE Transactions on Information Theory*, **22**(6), 1976.

[8] U. Feige, A. Fiat, and A. Shamir. Zero-knowledge proofs of identity. *Journal of Cryptology*, **1**(2): 77–94, 1988.

[9] S. Goldwasser, S. Micali, and C. Rackoff. Knowledge complexity of interactive proof systems. *SIAM Journal of Computing*, **18**(1): 186–208, 1989.

[10] C. A. R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, **12**(10): 576–580, 1969.

[11] C. l'Anson and C. Mitchell. Security defects in CCITT recomendation X.509—the directory authentication framework. *Computer Communication Review*, **20**(2): 30–34, April 1990.

[12] C. Meadows. Using narrowing in the analysis of key management protocols. In *IEEE Computer Society Symposium on Security and Privacy*, p. 138–147, 1989.

[13] D. M. Nessett. A critique of the Burrows, Abadi and Needham logic. *Operating System Review*, **24**(2), April 1990.

[14] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public key crypto systems. *Communications of the ACM*, **21**(2): 120–126, 1978.