Verification of Multi-Protocol Attacks

C.J.F. Cremers
Eindhoven University of Technology,
Department of Mathematics and Computer Science,
P.O. Box 513, NL-5600 MB Eindhoven, The Netherlands,
e-mail: ccremers@win.tue.nl

Abstract

Formal modeling and verification of security protocols typically assumes that a protocol is executed in isolation, without other protocols sharing the network. We investigate the existence of multi-protocol attacks on protocols described in literature. Given two or more protocols, that share key structures and are executed in the same environment, are new attacks possible? Out of 30 protocols from literature, we find that 23 are vulnerable to multi-protocol attacks.

1 Introduction

A number of successful formal methods have been developed to analyse security protocols in recent years. These methods are generally limited to verification of protocols that run in isolation: For a protocol that is used over an untrusted network, the formal models generally assume that there is only one protocol that is using the network. For such protocols, there are various methods to prove correctness or to find attacks.

The assumption that a protocol is the only protocol run over the untrusted network is not realistic. However, when multiple protocols are used over a shared untrusted network, the problem of verifying properties becomes significantly harder. Verification in such multi-protocol environments is a much more difficult task, because security properties are not compositional. When two protocols, that are correct when run in isolation, are used over the same network, new attacks might be introduced.

An attack that involves more than one protocol, is called a *multi-protocol attack*. That such attacks exist has been established in [1] by Kelsey, Schneier and Wagner. They show that given a correct protocol, it is possible to construct a specially tailored protocol that is also correct. When these two protocols are run over the same network, the intruder can use messages from one protocol to attack the other protocol.

At the other end of the spectrum, sufficient conditions for compositionality have been established in [2] by Guttman and Thayer. If all protocols that use the same network satisfy certain requirements, e.g. sufficiently different message structures, compositionality is guaranteed. In that case, to prove correctness of the system it suffices to prove correctness of the protocols in isolation. However, the standard protocols found in literature do not meet these requirements, and thus the theoretical possibility of multi-protocol attacks remains.

The question that is answered here is: Are multi-protocol attacks a realistic threat which we should consider when using protocols from literature?

Our contributions is the scalability of the methods used for verification. This applies to the size of the models, as well as the expressivity of the semantics. Verification methods for security protocols traditonally do not scale well. This holds for manual methods (e.g. proofs by hand) as well as (semi-)automatic methods. It is feasible to verify small to medium-sized protocols in isolation, but large and/or composed protocols have been outside the scope of most formal methods. This lack of scalability has also led to a limited expressiveness of security protocol semantics of formal models: most models only allow for modeling a single protocol and its requirements.

By using recent formal semantics and verification methods, we have been able to verify two- and three-protocol composition of 30 protocols from literature. The tests have revealed a significant number of multi-protocol attacks. Because this particular problem has not been addressed before, all attacks we find are previously unreported.

2 Multi-protocols

Before we explain our experiments in detail, we first give a brief introduction to modeling security protocols. We define a security protocol by means of a Message Sequence Chart, as in Figure 1. This protocol is known as the Yahalom-Lowe protocol. The protocol in this example is based on symmetric encryption. We use the notation $\{m\}_k$ to denote the encryption of a message m with a key k. For a symmetric key k, we have that $m = \{\{m\}_k\}_k$. We assume perfect encryption: a message can only be decrypted by someone who has the key.

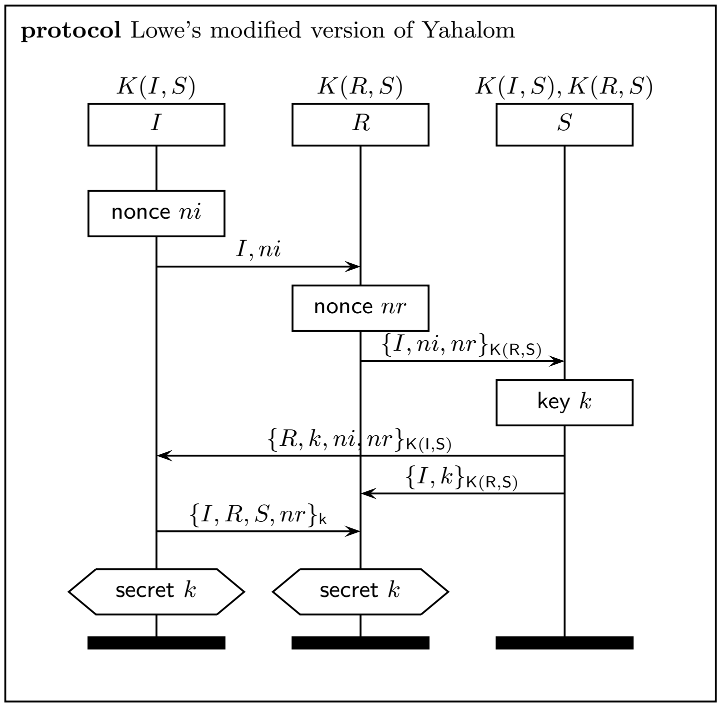

protocol Lowe's modified version of Yahalom
$K(I,S)$
$K(R,S)$
$K(I,S), K(R,S)$
I
R
S
nonce ni
I, ni
nonce nr
$\{I, ni, nr\}_{\mathsf{K(R,S)}}$
key k
$\{R, k, ni, nr\}_{\mathsf{K(I,S)}}$
$\{I, k\}_{\mathsf{K(R,S)}}$
$\{I, R, S, nr\}_{\mathsf{k}}$
secret k
secret k

Figure 1: Yahalom-Lowe

The protocol has three roles. There is a server role S, which shares symmetric keys of the form $K(I,S)$ with all the agents in the system. Such assumed initial knowledge is mentioned above the protocol role. The objective of this protocol is to have the server generate and distribute a secret session key for two agents. The two agents execute the roles I (initiator) and R (responder).

Communication of a message is represented by an arrow. Because agents only execute instances of single roles, and communication is asynchronous, this is interpreted as two separate events: the arrow represents the event of sending of the message into the network by the sender, and also the event of reading a message from the network by the receiver. The receiver verifies messages to see whether they match with the information he already has.

We also model an intruder in the network, that has full control over the network. Thus, any message that is sent can be intercepted by the intruder, and the intruder can construct messages and inject them into the network. The intruder has some initial knowledge, which can include the knowledge of compromised agents. When the intruder intercepts a message, he extends his knowledge with the message and anything he can derive from it by e.g. decryption.

We use the hexagons to denote security claim events: when an agent executes a protocol role up to such an event, the claim must hold. Thus, in this example, when an agent completes the I role, the key k must be secret and remain so.

We use the term multi-protocol environments to denote environments in which more than one protocol is using the network. In particular, we mean protocols that share key infrastructures. In such environments multi-protocol attacks can occur.

With the increased use of cryptographic libraries, their application on limited resource devices (e.g. Smart-cards), and the need to have a minimal key infrastructure, multi-protocol environments with shared key infrastructures are becoming more and more common.

3 Experiments

A large number of experiments were performed to verify the existence of multi-protocols on security protocols from literature. We have chosen a set of protocols. and we have verified the security properties of combinations of these protocols. When such a test yielded an attack, it was verified automatically whether the attack actually required multiple protocols, or could be mounted against a single protocol. In this section we elaborate on some of the details of these tests.

The majority of the protocols in our test was taken from literature: the Clark and Jacob library in [3], the related SPORE library at [4], and the work on protocols for authentication and key establishment in [5] by Boyd and Mathuria. This resulted in a set of 30 protocols. To this we added a set of 22 specially crafted protocols, which we used to verify a number of tailored protocol attacks.

Modeling the protocols has been done using the Operational Semantics of Security Protocols from [6]. This semantics allows for the modeling of multiple concurrent protocols and their security requirements. Security properties are expressed as local claim events, which reduces composition of security requirements to composition of the protocol specifications. The protocol roles were tested for three properties: secrecy of values, and two forms of authentication. These forms are agreement and synchronisation, as defined in [7].

The computational costs of verifying properties in multi-protocol environments are exponential w.r.t. the number of protocols. It is therefore currently impossible to verify an environment with all these protocols in parallel. Instead, we have chosen to test all possible combinations of two or three protocols from this set. Using this method, it was possible to find multi-protocol attacks that involve two or three protocols. To verify the existence of multi-protocol attacks with four or more protocols is possible future work.

The verification results are dependent on a *matching* parameter. This parameter is used to make assumptions about the semantics of the read events in the system, and expresses which classes of so-called *type-flaw* attacks are possible. We explain this in more detail in the next section. All tests were conducted three times, one time for each possible value of the matching parameter.

To give an indication of the number of tests, for combinations of two protocols we have $3 \times \binom{52}{2}$ tests, and for three protocols we have $3 \times \binom{52}{3}$ tests. Furthermore, some additional (simpler) tests were required to check whether the attacks could not have been mounted using less protocols. This amounts to over 70000 tests, which took a few days on a current desktop computer.

The experiments have been conducted using the Scyther tool (See [8]). This tool uses a hybrid theorem proving/model checking algorithm, which is an improved version of the algorithm developed by Song for the Athena tool in [9]. Given a description of a (set of) protocols, it tries to construct a counterexample (attack) for each claim. Verification of small protocols usually takes less than a second, either resulting in an attack description or a proof of correctness in most cases. In the remaining cases, the user can choose to bound the search space to e.g. a finite number of nonces. Furthermore, the tool is able to find so-called type-flaw attacks. The input language is based on the notation of the semantics described in [6].

4 Results

The tests reveal that there is a large number of multi-protocol attacks possible on the selected set of protocols. Out of the 30 protocols from literature, 23 had security claims that are correct in isolation, but had attacks when put in parallel with one or two other protocols from the set.

We classify the results into three groups, based on type-flaw categories. A type-flaw (See e.g. [10]) attack makes use of the fact that in many cases, agents cannot verify the values that they receive. As an example, suppose an agent is expecting to receive a random value. It has never encountered this value before, and therefore it cannot be compared to previous knowledge. In the formal modeling of such a read event, we discern three possibilities for the sets of terms that the agent accepts. The agent possibly accepts

- Only terms of the correct type, e.g. the set Nonce of all random values. (No type-flaws)
- All basic terms: E.g. random values, agent names, keys, but not tuples or encrypted terms. (Basic type-flaws)
- Any term. (Full type-flaws)

We start off with the most restricted model, in which it is assumed that the agents can somehow check the type of the data they receive, and thus only accept terms of the correct type. For the protocols from literature we found 17 two-protocol attacks, and no three-protocol attacks. We found attacks violating authentication as well as secrecy requirements. The main cause of these attacks is the way in which challenge-response mechanisms for authentication are constructed. An agent proves his identity to another agent or server by applying a key that only he knows. This can be done in roughly two ways: either by decrypting a challenge, or encrypting a challenge. These two methods can interfere with each other. As a result, these attacks commonly break secrecy of the random values. Such revealed random values can then be used to break authentication.

If a random value that is read can contain any basic term, the number of possible attacks for the intruder increases dramatically. Attacks in this category are called basic type-flaw attacks. Specifically, many attacks can now use the fact that keys and random values can mistakenly be accepted in each others' place, which enables attacks revealing the session keys. This can also cause new authentication attacks. In our tests, we found 40 attacks using basic type-flaw mistakes.

The situation gets worse for full type-flaw attacks. Typically, random values can now be mistaken for any tuple term or encrypted term. This enables many type-flaw attacks where large parts of messages are read as random values, and revealed at some other point. In fact, we found 106 multi-protocol attacks based on full type-flaws.

5 Attack examples

To give an indication of the type of attacks found, we give two examples. First we give an example of a two protocol attack on Yahalom-Lowe and Woo-Lam mutual authentication. The second example is a three protocol attack on the Yahalom-Lowe, Yahalom BAN and Denning-Sacco shared key protocols.

5.1 Yahalom-Lowe and Woo-Lam mutual authentication

As an example of a 2-protocol attack, we show an attack on the well-known Woo-Lam mutual authentication protocol (from [11]) and the Yahalom-Lowe protocol. There is a claimed proof of correctness for the isolated version of Yahalom-Lowe in [12].

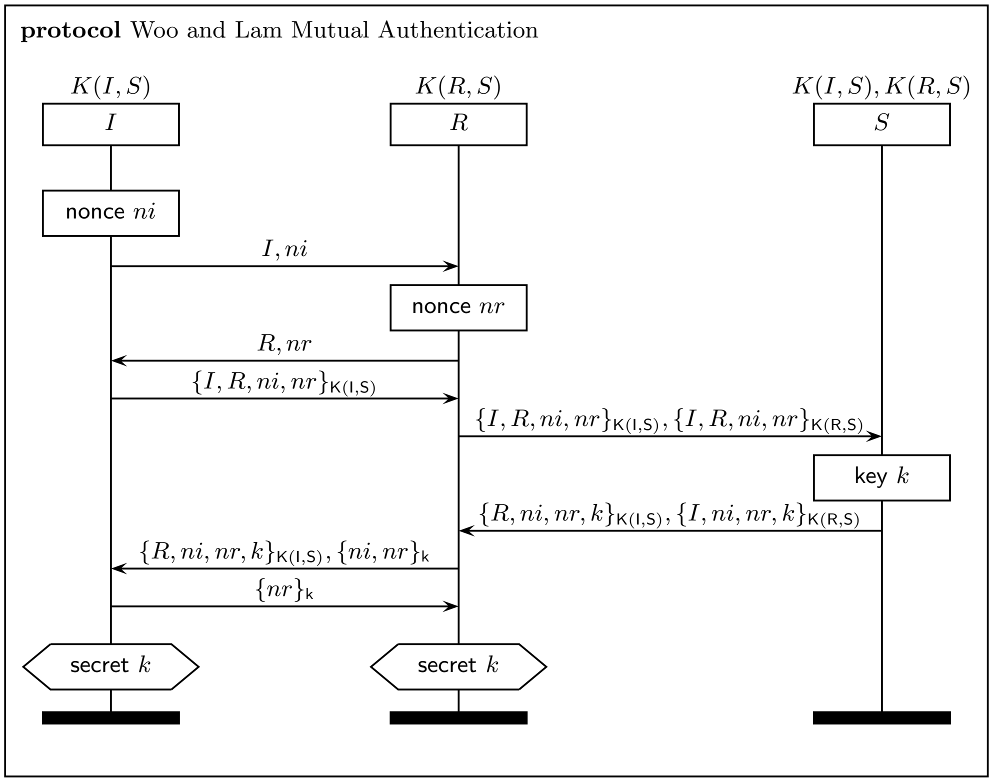
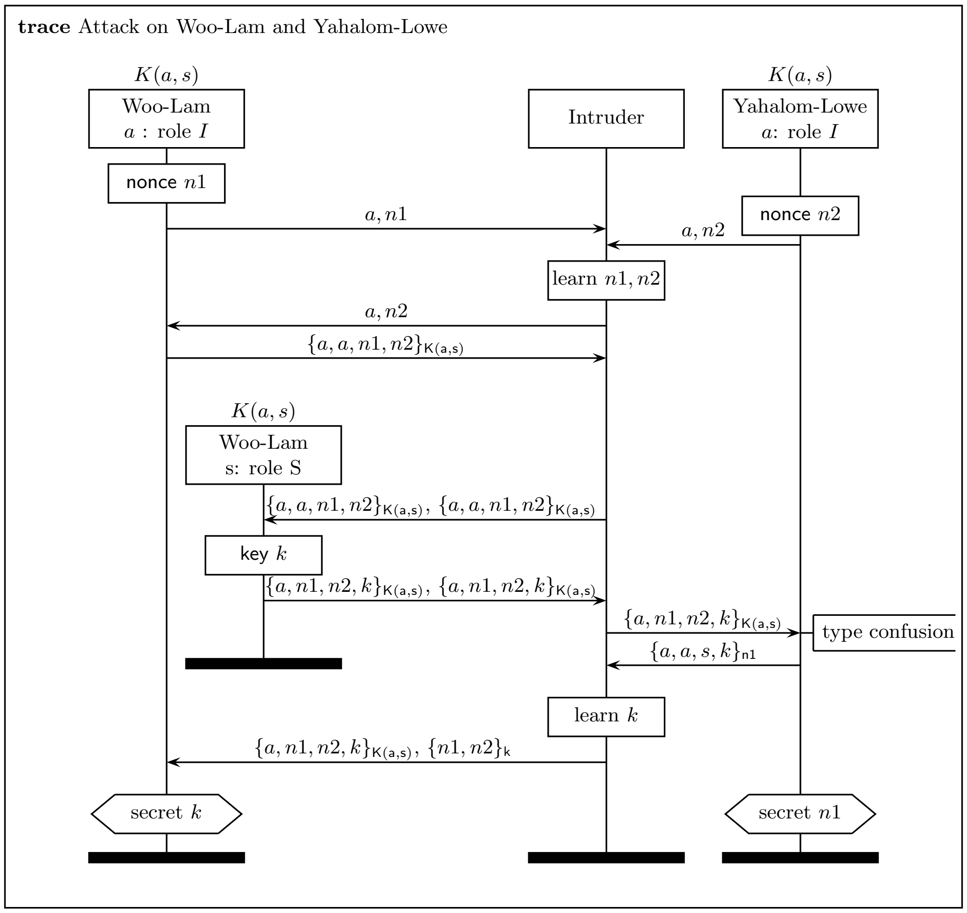

protocol Woo and Lam Mutual Authentication
$K(I,S)$
I
$K(R,S)$
R
$K(I,S), K(R,S)$
S
nonce ni
I, ni
nonce nr
R, nr
$\{I, R, ni, nr\}_{\mathsf{K(I,S)}}$
$\{I, R, ni, nr\}_{\mathsf{K(I,S)}}, \{I, R, ni, nr\}_{\mathsf{K(R,S)}}$
key k
$\{R, ni, nr, k\}_{\mathsf{K(I,S)}}, \{I, ni, nr, k\}_{\mathsf{K(R,S)}}$
$\{R, ni, nr, k\}_{\mathsf{K(I,S)}}, \{ni, nr\}_{\mathsf{k}}$
$\{nr\}_{\mathsf{k}}$
secret k
secret k

Figure 2: Woo-Lam mutual authentication

Both protocols use symmetric encryption and a trusted server to generate a fresh session key. The protocols are shown in Figures 1 and 2. They operate in a similar way: the initiator I and responder R both create a nonce (a fresh random value), which they send to the server. The server creates a new session key k, and distributes the key, combined with the nonces, back to I and R. They check the nonces, to confirm that the key is indeed fresh.



trace Attack on Woo-Lam and Yahalom-Lowe
$K(a,s)$
Woo-Lam a : role I
Intruder
$K(a,s)$
Yahalom-Lowe a: role I
nonce $n1$
$a, n1$
nonce $n2$
$a, n2$
learn $n1, n2$
$a, n2$
$\{a, a, n1, n2\}_{K(a,s)}$
$K(a,s)$
Woo-Lam s: role S
$\{a, a, n1, n2\}_{K(a,s)}$, $\{a, a, n1, n2\}_{K(a,s)}$
key k
$\{a, n1, n2, k\}_{K(a,s)}$, $\{a, n1, n2, k\}_{K(a,s)}$
$\{a, n1, n2, k\}_{K(a,s)}$
type confusion
$\{a, a, s, k\}_{n1}$
learn k
$\{a, n1, n2, k\}_{K(a,s)}$, $\{n1, n2\}_{k}$
secret k
secret $n1$

Figure 3: Attack on two protocols

In Figure 5.1 we show a multi-protocol attack on these protocols, exploiting a basic type flaw. This attack is possible if the agent cannot distinguish between a session key and a nonce, assuming that he has not encountered either before.

An agent a starts the Woo-Lam protocol in the I role, wants to communicate with another instance of a, and sends a fresh nonce $n1$. The intruder intercepts the nonce. The agent starts a Yahalom-Lowe session in parallel, in the I role. a creates and sends a second nonce $n2$. This is also intercepted by the intruder.

The intruder now sends the nonce $n2$ to a in the Woo-Lam protocol, as if it was sent by a Woo-Lam responder role. The agent responds with a server request with the names of both agents and the nonces $\{a, a, n1, n2\}_{K(a,s)}$. This message is intercepted, concatenated with itself, and sent to the Woo-Lam server s. The server generates a fresh session key and sends back two (identical) messages $\{a, n1, n2, k\}_{K(a,s)}$. One of these is redirected to the Yahalom-Lowe I role. This role is expecting a message of the form $\{a, \mathrm{Key}, n2, \mathrm{Nonce}\}_{K(a,s)}$, where Key is a new key and Nonce is a nonce, which he has not encountered before. Thus, he cannot tell the difference. Because of type confusion, he accepts the message, under the assumption that $n1$ is the fresh session key, and that k is the responder nonce.

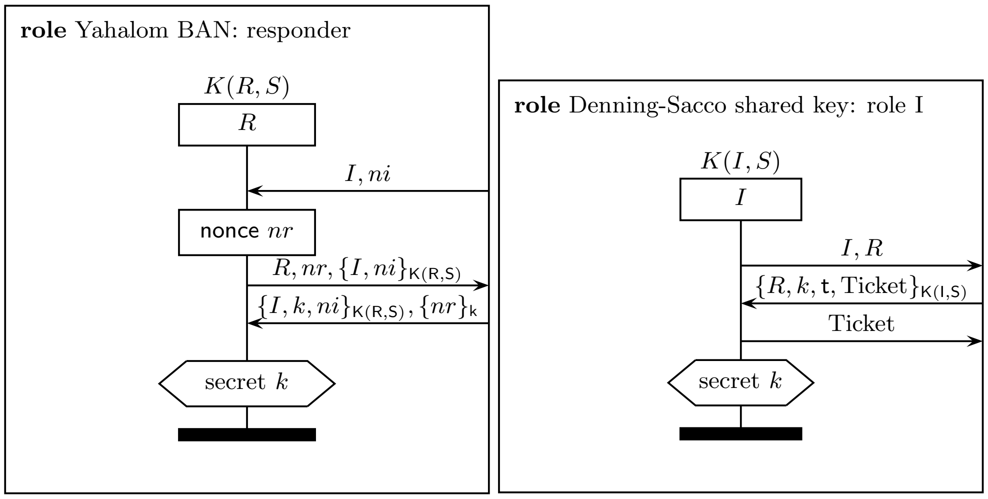
Thus, he encrypts the key using the nonce, sends $\{a, a, n2, k\}_{n1}$ and claims that $n1$ is secret. Because the intruder knows $n1$, this is clearly not the case. This is an attack on the Yahalom-Lowe I role.

However, we can continue the attack. The intruder intercepts this last message. Because he knows $n1$, he can decrypt the message, and learns the key k. This enables him to create the last message that is expected by the Woo-Lam I role. This role will then claim secrecy of k, which is also known to the intruder.

This basic type-flaw attack enables an intruder to break two protocols at the same time. Furthermore, it is an attack against Yahalom-Lowe, for which no attack had been known previously.

5.2 Yahalom-Lowe, Yahalom-BAN and Denning-Sacco shared key

As an example of a 3-protocol attack we will show one on Yahalom-Lowe, Yahalom BAN (from [13]) and Denning-Sacco shared key (from [14]). In Figure 5.2 we have depicted the parts of Yahalom BAN and Denning-Sacco shared key that are relevant for this attack.



role Yahalom BAN: responder
$K(R,S)$
R
I, ni
nonce nr
$R, nr, \{I, ni\}_{K(R,S)}$
$\{I, k, ni\}_{K(R,S)}, \{nr\}_k$
secret k
role Denning-Sacco shared key: role I
$K(I,S)$
I
I, R
$\{R, k, t, \text{Ticket}\}_{K(I,S)}$
Ticket
secret k

Figure 4: Two protocols

The Denning-Sacco shared key protocol is also a session key distribution protocol based on a trusted server and symmetric cryptography. It is based on timestamps rather than nonces. The initiator role, shown in the figure, receives an encrypted message from the server. In this message is the name of the responding agent, a fresh session key k, a timestamp t and a *Ticket*. This Ticket is supposed to be a message from the server for the responder, encrypted with their shared key. As a result, the initiator cannot validate the contents of the Ticket, and simply sends it to the responder. This will be exploited in the attack.

In Figure 5.2 an example of a 3-protocol attack can be found. Note that this attack involves four different type-flaws.

An agent a starts the Denning-Sacco protocol as initiator, wanting to communicate with b. The intruder intercepts the first message, and generates a session key ke and a timestamp t. He sends a, ke to b in the Yahalom-BAN responder role. For this protocol, a nonce is expected, but the key ke is accepted instead (this the first type-flaw.) Agent b responds by sending his name, a fresh nonce $n1$, and an encrypted message for the server: $\{a, ke\}_{K(b,s)}$. The intruder will use this message

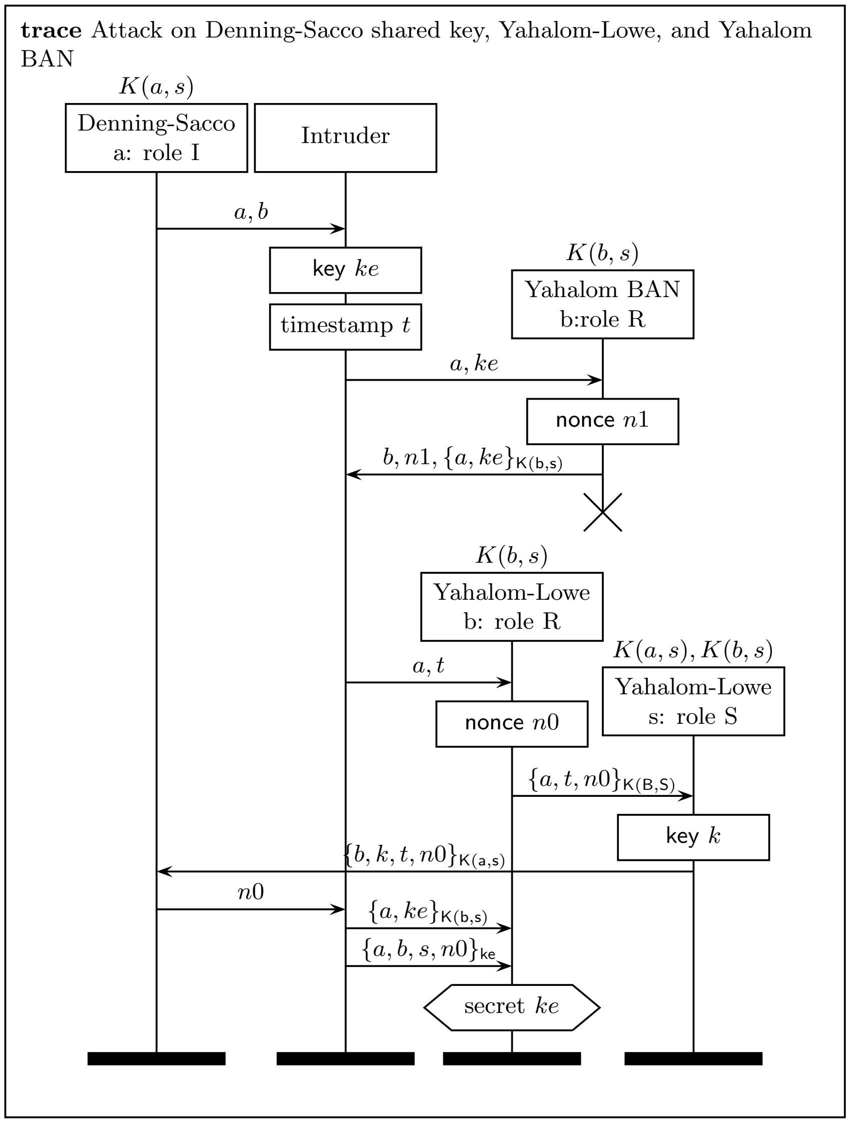


Figure 5: Attack on three protocols

later.

The intruder now sends a, t to b for the Yahalom-Lowe responder role. b expects a name and a nonce, and mistakingly accepts the timestamp t as a nonce (type-flaw nr. 2.) After generating a fresh nonce $n0$, b sends an encrypted message to the server, containing the initiator name and the two nonces. Because of the type-flaw, he in fact sends $a, t, n0$. The server expects nonces, but accepts the timestamp (type-flaw nr. 3.) He generates a fresh session key k, and sends back the key and the nonces, encrypted with the key shared with a. This is forwarded by the intruder to the Denning-Sacco initiator role of a.

The agent a expects a message with a ticket inside, but cannot verify whether or not this is the case. He mistakenly accepts the nonce $n0$ as the ticket (this is the fourth type-flaw). Effectively he decrypts $n0$ from the message, acting as an oracle, and thus $n0$ is learnt by the intruder.

Now the intruder has enough information to convince b in the Yahalom-Lowe role that a secret session key was generated by the server. He sends the message he intercepted earlier, $\{a, ke\}_{K(b,s)}$, and sends it to b, fooling b into thinking that ke is a secret key. Now, he can construct the final message, because he knows $n0$ from the interaction with the Denning-Sacco protocol. Upon accepting this message, b will claim that ke is a secret session key. This completes the attack.

6 Related Work

The formal analysis and verification of multi-protocols has only been the subject of recent investigations. The first results in this area are surfacing at the moment.

As already mentioned in the introduction, the main result that multi-protocol attacks exist has been given in [1]. Some detailed examples for public-key infrastructures have been given in [15].

A large part of the research in this area deals with specifying sufficient conditions for security protocols, in order guarantee that the composition of protocols that satisfy these conditions does not introduce any new attacks. Fairly strong requirements, or explicit design constraints, to ensure that protocol composition does not introduce new attacks, were given in [16]. This situation was improved significantly when less strict requirements were proven to be sufficient in [2], based on an adapted version of the Strand Space formalism in [17].

From a cryptographic point of view, a general approach to the composition problem can be found in [18]. A more detailed analysis of the environmental requirements for correct composition of authentication authentication protocols has been given in [19].

An attempt to develop a logic for secure protocol composition can be found in [20].

All these approaches have in common that they give sufficient but not necessary requirements for protocol composition. Thus, there exist protocols that can be composed safely, but do not meet the requirements of these methods. Thus, if we are looking for an optimal solution (e.g. minimal message size or number of protocol steps) for some multi-protocol environment with limited resources, these methods cannot help us. It is in fact in such environments with limited resources, that key structures are often shared between protocols.

There have not been many attempts on the actual verification of multi-protocol environments. A notable exception can be found in [21], where the interaction of sub-protocols in the Internet Key Exchange protocol was investigated using the NRL protocol analyzer.

In [22] a tool is presented for detecting possible authentication problems in multi-protocol environments. The tool statically checks whether protocols meet a set of sufficient conditions. However, the sufficient conditions that can be checked are specifically tailored for a very weak form of authentication, and do not allow for ensuring e.g. full agreement or synchronisation.

7 Conclusions and Future Work

By conducting these experiments, we have found 163 multi-protocol attacks. This shows that multi-protocol attacks on protocols from literature exist in large numbers, and are feasible. All attacks found here are previously unreported. We have discovered many more attacks than we had expected: the possibility of multi-protocol attacks is therefore a much larger threat than we assumed.

Some of the security claims of the protocols are correct in isolation, and are even correct when put in parallel with any other protocol from the set, but are broken by a 3-protocol attack. This proves that it is not sufficient to check for 2-protocol attacks only.

The problem of multi-protocol attacks is not limited to a small subset of the protocols. Out of the 30 protocols, we found that 23 of them had security claims that are correct in isolation but for which multi-protocol attacks existed. Furthermore, we found multi-protocol attacks on combinations of protocols for which no attacks were known previously.

Many attacks are intricate and we would not have been able to find them without tool support. Using formal models and tools has proven invaluable to assess the feasibility of these attacks, and has allowed us to conduct such large scale tests.

The tests have yielded much data on possible attacks, and we consider mining all possible information from this data to be future work. This involves e.g. automated classification of attacks.

The main conclusion to be drawn here, is that for multi-protocol environments, it is absolutely necessary to address the interaction between the protocols. This can only be done by looking at all the protocols in the environment: a single protocol can cause all others to break. Taking protocols from literature, that have been proven to be correct in isolation, gives no guarantees at all for multi-protocol environments.

References

[1] J. Kelsey, B. Schneier, D. Wagner, Protocol interactions and the chosen protocol attack, in: Security Protocols Workshop, 1997, pp. 91–104.
URL `citeseer.ist.psu.edu/kelsey97protocol.html`

[2] J. Guttman, F. Thayer, Protocol independence through disjoint encryption, in: PCSFW: Proceedings of The 13th Computer Security Foundations Workshop, IEEE Computer Society Press, 2000, http://citeseer.ist.psu.edu/guttman00protocol.html.

[3] J. Clark, J. Jacob, A survey of authentication protocol literature, Tech. Rep. 1.0 (1997).
URL `citeseer.nj.nec.com/clark97survey.html`

[4] Security protocols open repository, `www.lsv.ens-cachan.fr/spore`.

[5] C. Boyd, A. Mathuria, Protocols for Authentication and Key Establishment, Information Security and Cryptography, Springer, 2003, iSBN: 3-540-43107-1.

[6] C. Cremers, S. Mauw, Operational semantics of security protocols, in: Scenarios: Models, Algorithms and Tools (Dagstuhl 03371 post-seminar proceedings), LNCS, 2005, to appear.
URL `http://www.win.tue.nl/~ecss/downloads/CrMa04b.pdf`

[7] C. Cremers, S. Mauw, E. de Vink, Defining authentication in a trace model, in: T. Dimitrakos, F. Martinelli (Eds.), FAST 2003, Proceedings of the first international Workshop on Formal Aspects in Security and Trust, IITT-CNR technical report, Pisa, 2003, pp. 131–145.
URL `http://www.win.tue.nl/~ecss/downloads/cmv_defining_authentication.ps`

[8] C. Cremers, Scyther documentation, `www.win.tue.nl/~ccremers/scyther`.

[9] D. Song, Athena: a new efficient automatic checker for security protocol analysis, in: Proceedings of the 1999 IEEE Computer Security Foundations Workshop, IEEE Computer Society, 1999, p. 192.

[10] J. Heather, G. Lowe, S. Schneider, How to prevent type flaw attacks on security protocols., Journal of Computer Security 11 (2) (2003) 217–244.

[11] T. Woo, S. Lam, A lesson on authentication protocol design, SIGOPS Oper. Syst. Rev. 28 (3) (1994) 24–37.

[12] G. Lowe, Towards a completeness result for model checking of security protocols, in: PCSFW: Proceedings of The 11th Computer Security Foundations Workshop, IEEE Computer Society Press, 1998.
URL `citeseer.ist.psu.edu/lowe99towards.html`

[13] M. Burrows, M. Abadi, R. Needham, A logic of authentication, ACM Transactions on Computer Systems 8 (1990) 18–36.

[14] D. Denning, G. Sacco, Timestamps in key distribution protocols, Commun. ACM 24 (8) (1981) 533–536.

[15] J. Alves-Foss, Multiprotocol attacks and the public key infrastructure, in: Proc. 21st National Information Systems Security Conference, Arlington, Va., 1998, pp. 566–576.

[16] L. Gong, P. Syverson, Fail-stop protocols: An approach to designing secure protocols, in: Proceedings of the 5th International Working Conference on Dependable Computing for Critical Applications (DCCA-5), 1995, pp. 44–55.
URL `citeseer.ist.psu.edu/gong95failstop.html`

[17] F. Thayer, J. Herzog, J. Guttman, Mixed strand spaces, in: Proceedings of the 1999 IEEE Computer Security Foundations Workshop, IEEE Computer Society, 1999, p. 72.

[18] R. Canetti, Universally composable security: A new paradigm for cryptographic
protocols, Cryptology ePrint Archive, Report 2000/067, `eprint.iacr.org/` (2000).

[19] R. Canetti, C. Meadows, P. Syverson, Environmental requirements for authentication protocols (2002).
URL `citeseer.ist.psu.edu/canetti02environmental.html`

[20] A. Datta, A. Derek, J. C. Mitchell, D. Pavlovic, Secure protocol composition, in: Proceedings of Mathematical Foundations of Programming Semantics, Vol. 83 of Electronic Notes in Theoretical Computer Science, 2003.

[21] C. Meadows, Analysis of the Internet Key Exchange protocol using the NRL protocol analyzer, pp. 48–61.
URL `citeseer.ist.psu.edu/article/meadows99analysis.html`

[22] M. Maffei, Tags for multi-protocol authentication, in: Proc. SECCO 2004, Electronic Notes in Theoretical Computer Science, 2004, to appear.

A List of protocols

The full list of 30 protocols from literature included in the tests is given below:

Protocol name	**Multi-protocol attack**
Andrew Secure RPC (BAN version)	No*
Andrew Secure RPC (Lowe modified BAN version)	No*

Protocol name	**Multi-protocol attack**
Bilateral Key Exchange	yes
Boyd key agreement	yes
Denning-Sacco shared key	yes
Gong (nonce based)	yes
Gong (nonce based, version 2)	yes
ISO ccitt 509 (BAN version)	yes
ISO IEC 11770 2-13	no
Kao-Chow	yes
Kao-Chow (version 2)	yes
Kao-Chow (version 3)	yes
KSL (based on Kerberos)	yes
Needham-Schroeder mutual authentication	yes
Needham-Schroeder-Lowe mutual authentication	yes
Needham-Schroeder symmetric	yes
Needham-Schroeder symmetric (amended)	yes
Otway-Rees	no*
SOPH Secret Out, Public Home	yes
Splice-AS	no*
Splice-AS Hwang and Chen modified	no*
Splice-AS Hwang and Chen modified (Clark Jacob)	yes
TMN	no*
Wide Mouthed Frog (Brutus version)	yes
Woo and Lam pi f (unilateral authentication)	yes
Woo-Lam mutual authentication	yes
Yahalom	yes
Yahalom (BAN)	yes
Yahalom Lowe modified	yes
Yahalom Paulson strengthened	yes

(*) There are attacks for these protocols when running in isolation: multi-protocol attacks do not introduce any new attacks on claims that were correct in isolation.