Feasibility of Multi-Protocol Attacks

Cas Cremers
Eindhoven University of Technology
Department of Mathematics and Computer Science
P.O. Box 513, NL-5600 MB Eindhoven, The Netherlands
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. We identify two likely attack patterns and sketch a tagging scheme to prevent multi-protocol attacks.*

1. Introduction

In recent years, a number of successful formal methods and corresponding tools have been developed to analyse security protocols, e.g. [2, 3, 21, 24]. 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.

However, the assumption that a protocol is the only protocol run over the untrusted network is not realistic. Unfortunately, when multiple protocols are used over a shared untrusted network, the problem of verifying security properties becomes significantly harder. The cause of this is the fact that 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 necessarily involves more than one protocol, is called a *multi-protocol attack*. That such attacks exist has been established by Kelsey, Schneier and Wagner in [18]. 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. Some examples have been explained in [1, 27]. Because the protocols in these papers are constructed especially for the purpose of breaking some other protocol, they seem contrived, and it seems unlikely that they would occur in practice.

At the other end of the spectrum, sufficient conditions for compositionality have been established by e.g. Guttman and Thayer in [15]. Alternative requirements can be found in [5, 6, 14]. If *all* protocols that use the same network and key infrastructure satisfy certain requirements, e.g. sufficiently different message structures, compositionality of the individual security properties is guaranteed. In that case, in order 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.

A third approach to the compositionality problem are methods that aim to establish that a given set of protocols does not interfere. A recent example of such an approach is e.g. the invariant based approach in [12].

The question that is answered here is: Are multi-protocol attacks a realistic threat which we should consider when using protocols from literature? This question has not been addressed before, because verification methods for security protocols traditionally 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.

Modifying semantics and tools for such analysis was already pointed out as an open research question in [9], and has been addressed partially in e.g. [23, 26]. A semantics which can handle multi-protocol analysis in an intuitive way, is the operational semantics for security protocols by

0-7695-2567-9/06 $20.00 © 2006 IEEE

Cremers and Mauw in [10]. Based on these semantics, the Scyther tool [8] has been developed. Using this tool, 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.

We proceed as follows. In Section 2 we will define Multi-Protocol attacks and we describe the conducted experiments. The results of the experiments are analysed in Section 3, and two feasible attack patterns are analysed in Section 4. We discuss preventive measures in Section 5 and draw conclusions in Section 6.

Acknowledgements The author would like to thank Sjouke Mauw for suggesting the research problem in the first place, and Gijs Hollestelle for his work on classifying the attacks.

2. Multi-Protocol Attacks

In order to define what a multi-protocol attack is, we assume that there exists some model to describe a security protocol and its properties. In particular, we assume that there exists a predicate attack(ps, c) for all sets of protocols ps and all security properties c, that is true if and only if there is an attack possible on the security property c, in an environment where only the protocols in the set ps are being executed. As stated in the introduction, there are many formalisms and tools that can help in establishing the value of attack(ps, c) for all singleton sets ps. For sets with more than one element, we define Multi-Protocol attacks.

Definition 1 *A Multi-Protocol Attack is a tuple (ps, c), where ps is a set of protocols, and c is a security property, iff the following property holds:*

$$|ps| > 1 \;\wedge\; \mathit{attack}(ps, c) \;\wedge\; \forall ps' \subset ps : \neg \mathit{attack}(ps', c)$$

In other words: the set ps contains more than one protocol, there exists an attack on the security property c in an environment with the protocols in ps, and furthermore all protocols in ps are required for the attack.

Note that this definition does not put any constraints on the intruder behaviour. Therefore, it is more general than the definition given in [13], where only replaying a message from one protocol into another is allowed.

The question we set out to address was:

Given a set of security protocols, investigate whether multi-protocol attacks occur. If they occur, assess their feasibility and try to extract any typical attack patterns.

The set of protocols included in our test is shown in Table 1. The protocols were selected from literature: the Clark and Jacob library in [7], the related SPORE library at [25], and the work on protocols for authentication and key establishment in [4] by Boyd and Mathuria. This resulted in a set of 30 protocols. For these experiments, we have only considered three security properties: secrecy and two forms of authentication: non-injective agreement and non-injective synchronisation. Formal definitions of these properties can be found in [10, 11].

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. (Because of complexity, verifying the existence of multi-protocol attacks with four or more protocols is future work.) 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.

The verification results also depend on a *matching* parameter, that 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. In total, over 14000 tests were performed to obtain these results.

The experiments have been conducted using the Scyther tool [8]. This tool uses a hybrid theorem-proving/model-checking algorithm, which is an improved version of the algorithm by Song for the Athena tool in [24]. Given a description of a (set of) protocols, it tries to construct a counterexample for each security 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 role instances. Contrary to Athena, Scyther can verify authentication properties such as synchronisation from [11], and can handle non-atomic keys and multiple key structures.

3. 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 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, which is often called a *nonce*. It has

Table 1. Protocol list

Protocols with Multi-Protocol Attacks	
Bilateral Key Exchange	NS symm.
Boyd key agreement	NS symm. amended
Denning-Sacco shared key	SOPH
Gong (nonce)	Splice-AS Hwang and Chen[a]
Gong (nonce) v2	Wide Mouthed Frog (Brutus)
ISO ccitt 509 (BAN)	Woo and Lam pi f
Kao-Chow	Woo-Lam mutual auth.
Kao-Chow v2	Yahalom
Kao-Chow v3	Yahalom (BAN)
KSL	Yahalom-Lowe
Needham-Schroeder	Yahalom-Paulson
Needham-Schroeder-Lowe	

Protocols for which we found no Multi-Protocol Attacks	
Andrew Secure RPC (BAN)	Otway-Rees
Andrew Secure RPC (Lowe)	Splice-AS
ISO IEC 11770 2-13	Splice-AS Hwang and Chen[b]
TMN	

[a] Modified version 1

[b] Modified version 2 (Clark and Jacob)

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: (1) only terms of the correct type, e.g. the set of nonces, and thus no type-flaws can occur, or (2) all basic terms, but not tuples or encrypted terms, allowing for what we call basic type-flaws, or (3) any term, in which case all type-flaw attacks can occur. We discuss each of these categories separately.

No 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. Random values revealed in this way can be used to break authentication.

An interesting fact is that the vast majority of these attacks involve variants of the Woo-Lam Pi protocol from [28]. The reason for this is that these protocols contain a pattern which can be used as a so-called encryption oracle. An encryption oracle is a protocol mechanism that allows an intruder to encrypt arbitrary values with some key, in this case the symmetric key shared by an agent and the server. This enables many attacks on other protocols that involve this shared key.

Table 2. Influence of read semantics

No type-flaws possible	17 attacks
Basic type-flaws possible	40 attacks
All type-flaws possible	106 attacks

The remainder of the attacks involves something that is similar to what we call *ambiguous authentication*, which will be explained in detail in Section 4.

Basic type-flaws

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.

All type-flaws

As expected, the situation gets worse when any kind of type-flaw attacks is possible. Typically, random values can now be mistaken for any tuple term or encrypted term. This enables 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 all type-flaws.

We also found examples of three-protocol attacks. For example, the Yahalom-Lowe claim of the secrecy of the received session key, is correct in isolation, is correct in combination with any other protocol from the test, but can be broken in the presence of the Denning-Sacco shared key and Yahalom-BAN protocols if all type-flaws are possible.

To conclude, we have summarized the influence of the read semantics (and thus the susceptibility to type-flaw attacks) in Table 2.

Attack example

As an example of a basic type-flaw two protocol attack, we show an attack on the Woo-Lam mutual authentication protocol from [28] and the Yahalom-Lowe protocol from [22]. Before we explain the example attack in detail, we first give a brief introduction to the way we model 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. There exists a claimed proof of correctness for the isolated version of Yahalom-Lowe.

The protocol in this example uses 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.

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).

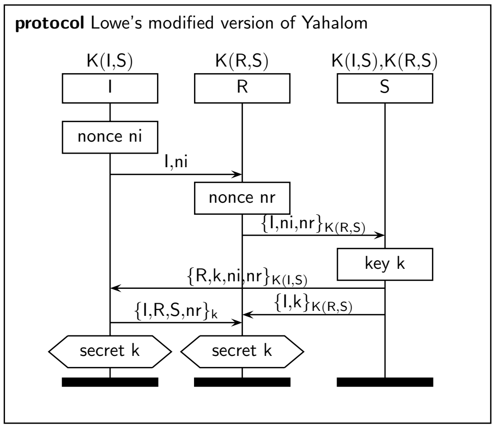

Figure 1. Yahalom-Lowe

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 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 also model an intruder, which has full control over the network. 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. The full details of the operational semantics that we use are described in [10].

We now return to the attack example. The attack involves the Yahalom-Lowe protocol, and also the Woo-Lam mutual authentication protocol, which is shown on the left in Figure 2. Both protocols use symmetric encryption and a trusted server to generate a fresh session key. 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.

On the right hand side in Figure 2 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.

The attack proceeds as follows. 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 $\{a, \text{Key}, n2, \text{Nonce}\}_{K(a,s)}$, 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 then claims 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.

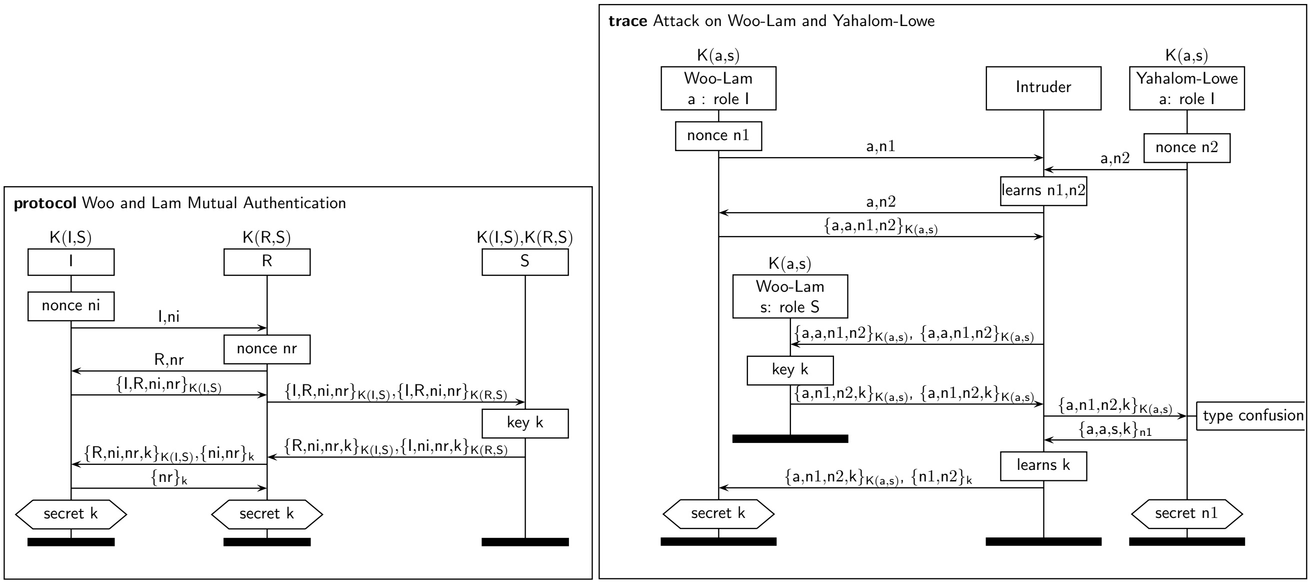
protocol Woo and Lam Mutual Authentication
K(I,S)
K(R,S)
K(I,S),K(R,S)
I
R
S
nonce ni
I,ni
nonce nr
R,nr
$\{I,R,ni,nr\}_{K(I,S)}$
$\{I,R,ni,nr\}_{K(I,S)},\{I,R,ni,nr\}_{K(R,S)}$
key k
$\{R,ni,nr,k\}_{K(I,S)},\{I,ni,nr,k\}_{K(R,S)}$
$\{R,ni,nr,k\}_{K(I,S)},\{ni,nr\}_k$
$\{nr\}_k$
secret k
secret k
trace 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
learns 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
$\{a,a,s,k\}_{n1}$
learns k
$\{a,n1,n2,k\}_{K(a,s)}$, $\{n1,n2\}_k$
secret k
secret n1

Figure 2. Attack on two protocols

4. Attack patterns

The experiments have revealed that although many multi-protocol attacks can occur, their scope is limited if type-flaw attacks are prevented. But even if such attacks are excluded, two main scenarios remain in which multi-protocol attacks are likely to occur. We discuss the scenarios in this section, and we discuss preventive measures in the next section.

Protocol updates

The experiments have shown that multi-protocol attacks are likely on protocols that use similar messages. We describe here a feasible scenario where such similarities arise in practice.

It often occurs that security protocols are broken in some way, and that this is discovered after the protocol is deployed. The problem can be fixed by issuing a security update. This is effectively a second protocol, that shares the same key structure, which is very similar to the first one. Such a situation makes multi-protocol attacks very likely.

As an example, we show in Figure 4 a broken authentication protocol. It is susceptible to a man-in-the-middle attack, similar to the one described in [20], which we will not show here. For our purposes, we only assume this protocol has been distributed to and is being used by clients, and that we need to update it with a security fix. The easiest way to fix the protocol is to replace the name in the first message, resulting in the protocol in Figure 4. This protocol is also known as the Needham-Schroeder-Lowe protocol (typically abbreviated to NSL), and can be proven to be correct when run in isolation.

If the broken protocol is updated in such a way that the old version of the protocol can still be running as well on some clients, then there is a multi-protocol attack possible on the new protocol, as shown in Figure 3. In this attack, the intruder uses two instances of the old protocol (denoted by “Broken I” and “Broken R”) to learn the value of a nonce na. Then, an instance of the responder role of the new protocol (“NSL R”) is completed using the initiator role of the old protocol. Thus, an agent completing the responder role of new protocol claims that the nonces na and nb are secret. Because the nonce na was already leaked by the old protocol version, this claim is false.

The cause of the problems is that the messages of the updated protocol often closely resemble the messages of the original protocol. Because of this, many possibilities are available for an intruder to insert messages from one protocol at unforeseen places in the other protocol, which opens the door for multi-protocol attacks.

Ambiguous authentication

Authentication protocols are often used to set up session keys for other protocols. The resulting protocol then consists of the sequential composition of the authentication protocol and the protocol that uses the session key. Often the same protocols is used for authentication, which is then composed with different follow-up protocols. In such cases ambiguous authentication can occur: although the authentication protocol is correct in isolation, there can be a multi-protocol attack involving different follow-up protocols.

We give an example of this phenomenon. Consider the protocol pattern “Service 1”, as shown in Figure 4. In the

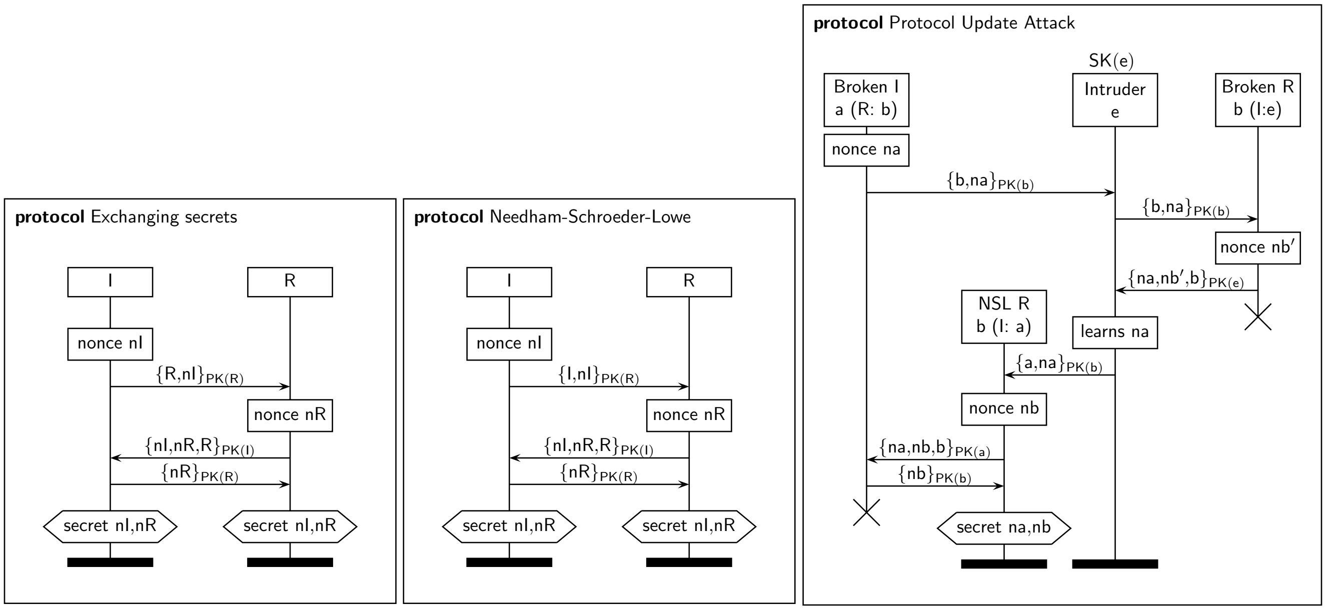

Figure 3. NSL attack using the broken variant

left of this figure, there is a large rectangle denoted as protocol P. For this rectangle, we can substitute any protocol that authenticates the partners and generates a fresh shared secret. (For example we could insert here the Needham-Schroeder-Lowe protocol from Figure 4, and take either of the nonces as the fresh secret x.) This protocol P is then extended by a single message, that sends the secret Y, encrypted with the fresh secret value from the protocol P, from the initiator I to the responder R. Given that the protocol P is correct, we can prove that the complete protocol for Service 1 is correct.

Now we re-use the protocol P to implement a protocol for Service 2, as in the centre of the figure. Here we again use the same base protocol, but we extend it by sending a session identifier and some message m. For the session identifier, we use the fresh random value x from the base protocol. (If we substitute Needham-Schroeder-Lowe for P, the protocol for Service 2 is secure in isolation.)

If we run Service 1 in parallel with Service 2, the combined protocols are broken. The attack is shown in on the right in Figure 4. In this attack, the intruder simply re-routes the initial messages from Service 1 to Service 2. After this initial phase a is halfway into Service 1, and b is halfway into Service 2. Therefore b will now use the random value x as a session identifier, effectively revealing it to the intruder. Then, when a uses this value x as a session key for the secret Y, the intruder can decrypt it. Thus the security claim of Service 1 is violated.

5. Preventing Multi-Protocol Attacks

The experiments have also revealed what is required to effectively prevent multi-protocol attacks. We discuss two methods: context-aware tagging, which completely prevents multi-protocol attacks, and type-flaw prevention, which partially prevents such attacks. We furthermore discuss verification of multi-protocol attacks.

CATS: context-aware tagging scheme

As has been shown in [15], if two protocols are sufficiently different, their composition will not introduce new attacks. Ensuring that two protocols are different can be easily enforced by introducing e.g. a unique tagging scheme. As an abstract mechanism, this is sufficient to prevent all multi-protocol attacks. In realistic settings however, it can be unclear how to enforce uniqueness for the tags.

In particular, there are two elements that are easily overlooked in a tagging scheme, which can be derived directly from the attack patterns in the previous section. The first element is the *protocol version*: if a protocol is updated, the new version of the protocol often runs in parallel with the old version. Because their messages are very similar, there is a high probability that multi-protocol attacks exist. A second element is the *context*, uniquely identifying the follow-up protocol as already described in Section 4. Base protocols such as the ISO standard protocols are often used in multiple contexts: they often occur prefixed before several different follow-up protocols. This can easily lead to multi-protocol attacks. A good tagging scheme should in-

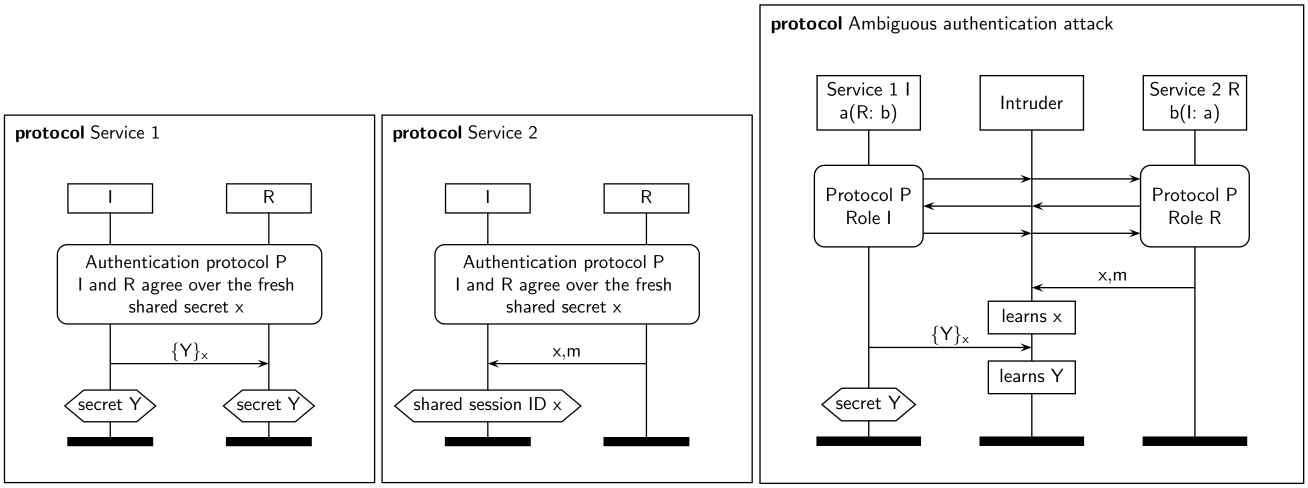

Figure 4. Attack on combined services

corporate these two elements. We propose a Context-Aware Tagging Scheme (CATS), that explicitly incorporates these two elements, as well as the protocol identifier.

Definition 2 *A CATS tag string is defined as*

$$tag = [\mathit{identifier}, \mathit{version}, \mathit{context}]$$

where identifier *uniquely identifies a protocol (e.g. "Needham Schroeder public key with Server"),* version *signifies the version of the protocol as used here, and* context *denotes the context in which the protocol is used (e.g. "Authentication for service 1"), thereby uniquely identifying the follow-up protocol. We require that the three fields of the tag are either of fixed length, or that they are clearly terminated by some end-of-field data item.*

We transform a security protocol into a CATS tagged protocol by means of the following procedure. Given a protocol, we transform it by modifying all term encryptions. At each point in the protocol where a term m is encrypted by a key k, we substitute it by an encryption of the tuple (tag, m) by the key k. This will ensure the requirements are met for *disjoint encryption* as described in [15], and ensures no multi-protocol attacks can occur. However, we must warn that in a multi-protocol environment, any single protocol can break the security of all others. Thus, it must be ensured the CATS tagging scheme is enforced everywhere where the same key infrastructure is used.

Although this scheme prevents all multi-protocol attacks, it can be costly in terms of increased message size. We therefore also discuss a second option, which prevents type-flaw attacks.

Strict type detection

As noted in [16], it is possible to prevent type-flaw attacks by adding type information to the messages occurring in a protocol. This significantly reduces the number of possible attacks on a single security protocol, and is therefore advisable even when not considering a multi-protocol environment.

The experiments detailed here have shown that making sure no type-flaw attacks can occur prevents many multi-protocol attacks. In fact, 84 percent of the attacks in the test set can not occur in a setting without type-flaw errors. Ensuring all messages are typed is therefore also a good preventive measure for multi-protocol attacks. For details of preventing type-flaw attacks we refer the reader to [16, 19].

Verification

In some cases it is undesirable to modify a set of protocols, unless it can be proven that a vulnerability exists. In such cases verification of (multi-protocol) attacks is a feasible option. We have shown here that using suitable tools such as Scyther [8] it is possible to perform automated analysis of concurrent protocols.

6. Conclusion

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. 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.

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. Many of the attacks that 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 a large amount of 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, as initiated in [17]. From the data we identified two likely attack patterns: protocol updates and ambiguous authentication.

We have sketched a context aware tagging scheme (CATS), that can be used to prevent multi-protocol attacks if it is consistently applied to all protocols that share the same key infrastructure. The proposed tagging scheme addresses issues raised by the identified attack patterns.

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. Alves-Foss. Multiprotocol attacks and the public key infrastructure. In *Proc. 21st National Information Systems Security Conference*, pages 566–576, Arlington, Oct 1998.

[2] A. Armando, D. Basin, Y. B. Y. Chevalier, L. Compagna, L. Cuellar, P. Drielsma, P. Heám, O. Kouchnarenko, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Viganò, and L. Vigneron. The AVISPA tool. In *Proc. of CAV'2005*, LNCS 3576. 2005.

[3] B. Blanchet. An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In *14th IEEE Computer Security Foundations Workshop (CSFW-14)*, pages 82–96, Cape Breton, June 2001. IEEE Computer Society.

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

[5] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067.

[6] R. Canetti, C. Meadows, and P. Syverson. Environmental requirements for authentication protocols, 2002.

[7] J. Clark and J. Jacob. A survey of authentication protocol literature. Technical report.

[8] C. Cremers. Scyther documentation.

[9] C. Cremers. Compositionality of security protocols: A research agenda, 2004.

[10] C. Cremers and S. Mauw. Operational semantics of security protocols. In S. Leue and T. Systä, editors, *Scenarios: Models, Transformations and Tools Workshop 2003, Revised Selected Papers*, volume 3466 of *LNCS*. Springer, 2005.

[11] C. Cremers, S. Mauw, and E. de Vink. Defining authentication in a trace model. In T. Dimitrakos and F. Martinelli, editors, *FAST 2003*, pages 131–145, Pisa, September 2003. IITT-CNR technical report.

[12] A. Datta, A. Derek, J. C. Mitchell, and D. Pavlovic. Secure protocol composition. In *Proc. of Mathematical Foundations of Programming Semantics*, volume 83 of *ENTCS*, 2003.

[13] X. Didelot. Cosp-j: A compiler for security protocols, 2003.

[14] L. Gong and P. Syverson. Fail-stop protocols: An approach to designing secure protocols. In *Proc. of the 5th International Working Conference on Dependable Computing for Critical Applications (DCCA-5)*, pages 44–55, 1995.

[15] J. Guttman and F. Thayer. Protocol independence through disjoint encryption. In *PCSFW: Proc. of the 13th Computer Security Foundations Workshop*. IEEE, 2000.

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

[17] G. Hollestelle. Systematic analysis of attacks on security protocols, Nov 2005. Master's Thesis.

[18] J. Kelsey, B. Schneier, and D. Wagner. Protocol interactions and the chosen protocol attack. In *Security Protocols Workshop*, pages 91–104, 1997.

[19] Y. Li, W. Yang, and C.-W. Huang. On preventing type flaw attacks on security protocols with a simplified tagging scheme. *J. Inf. Sci. Eng.*, 21(1):59–84, 2005.

[20] G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *Proceedings of TACAS*, volume 1055, pages 147–166. Springer Verlag, 1996.

[21] G. Lowe. Casper: A compiler for the analysis of security protocols. In *Proc. CSFW '97, Rockport*. IEEE, 1997.

[22] G. Lowe. Towards a completeness result for model checking of security protocols. In *PCSFW: Proc. of the 11th Computer Security Foundations Workshop*. IEEE, 1998.

[23] M. Maffei. Tags for multi-protocol authentication. In *Proc. SECCO 2004*, Electronic Notes in Theoretical Computer Science, August 2004. To appear.

[24] D. Song, S. Berezin, and A. Perrig. Athena: A novel approach to efficient automatic security protocol analysis. *Journal of Computer Security*, 9(1/2):47–74, 2001.

[25] Security protocols open repository (SPORE).

[26] F. Thayer, J. Herzog, and J. Guttman. Mixed strand spaces. In *Proc. of the 1999 Computer Security Foundations Workshop*, page 72. IEEE, 1999.

[27] W. Tzeng and C. Hu. Inter-protocol interleaving attacks on some authentication and key distribution protocols. *Inf. Process. Lett.*, 69(6):297–302, 1999.

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