Chosen-name Attacks

Ceelen, Pieter; Mauw, Sjouke; Radomirovi, Saša

Published in:
Electronic Notes in Theoretical Computer Science

DOI:
10.1016/j.entcs.2007.12.015

Publication date:
2008

Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 197 (2008) 31–43
www.elsevier.com/locate/entcs

Chosen-name Attacks: An Overlooked Class of Type-flaw Attacks

Pieter Ceelen[2] Sjouke Mauw[3] Saša Radomirović[1,4]

Université du Luxembourg
Faculté des Sciences, de la Technologie et de la Communication
6, rue Richard Coudenhove-Kalergi
L-1359 Luxembourg

Abstract

In the context of Dolev-Yao style analysis of security protocols, we consider the capability of an intruder to dynamically choose and assign names to agents. This capability has been overlooked in all significant protocol verification frameworks based on formal methods. We identify and classify new type-flaw attacks arising from this capability.
Several examples of protocols that are vulnerable to this type of attack are given, including Lowe's modification of KSL. The consequences for automatic verification tools are discussed.

Keywords: security protocols, automatic verification, type-flaw attacks, semantics.

1 Introduction

Security protocols have been the subject of study for a long time. Consequently, there have been many frameworks developed to verify security protocols, for instance [8,26,11,12]. Common to most frameworks is the assumption that the adversary has complete control over the network and is only limited by the constraints of cryptographic operations. The first attempt to precisely formulate this idea was done in the early 1980s by Dolev and Yao [13].

Dolev and Yao's threat model assumes that the adversary is *a legitimate user of the network*, able to *be a receiver to any user on the network*, *obtain any message passing through the network*, and trying *everything he can in order to discover the plaintext* of an encrypted message.

[1] Supported in part by a Centre de Recerca Matemàtica Postdoctoral Fellowship.
[2] Email: ceelen.p@gmail.com
[3] Email: sjouke.mauw@uni.lu
[4] Email: sasa.radomirovic@uni.lu

doi:10.1016/j.entcs.2007.12.015

Not all formal methods adopt all of these requirements. For instance, BAN Logic [8] does not consider insider attacks. This leads the three-line Needham-Schroeder public-key authentication protocol, shown in Figure 1, to be provably correct in BAN Logic in spite of the existence of an interleaving attack on it, as was first shown by Lowe [21]. Most modern formal methods do implement Dolev and Yao's intruder model but differ in their interpretation of it. For example, one major difference among the frameworks is that some of them allow the adversary to carry out *type-flaw* attacks and others do not.

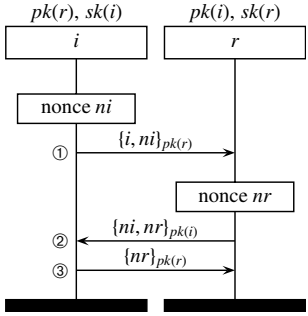

Fig. 1. Needham-Schroeder public-key authentication protocol

A type flaw occurs when the type of a message can be misinterpreted [7], for example, when one agent's encrypted term may be interpreted as a nonce by another agent. The adversary's attempt to use such a misinterpretation to his own advantage is called a *type-flaw attack*.

In fact, if we assume the order of the message components as in Figure 1, the Needham-Schroeder protocol is susceptible to the type-flaw attack where a malicious agent uses his name instead of a nonce in the first message in order to impersonate another agent. The attack requires two parallel sessions and is shown in detail in Figure 2. We write $a : r(b, a)$ to denote that agent a executes the responder role r assuming that the corresponding initiator role i is executed by agent b. Similarly, $eve(b) : i(b, a)$ means that the intruder *eve* impersonates b executing the initiator role in a session with agent a executing the responder role. There are mechanisms to ensure correctness of types throughout the execution of a protocol, for instance by message tags [15], therefore the absence of type-flaw considerations in some formalizations can be justified.

In this paper, we introduce and discuss new intruder abilities concerning the dynamic selection of names for compromised agents and potentially even for honest agents. In contrast to the well-known type-flaw attacks, such as the one shown on the Needham-Schroeder protocol, which are based on the assumption that agent names are static, we allow the adversary to choose dynamically created terms as the name of an agent. Dynamic agent naming is typically not being considered by protocol verification frameworks and tools.

While it is easy to craft secure protocols that are susceptible to what we call the *chosen-name type-flaw attacks*, we don't expect these attacks to be found frequently in real-life protocols. However, to prove their existence, we also show a chosen-

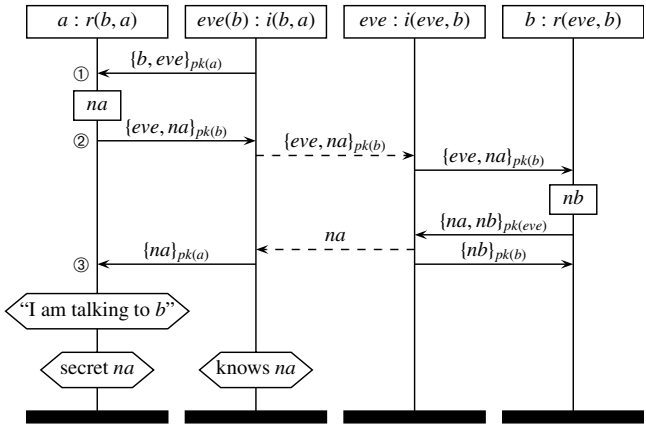

Fig. 2. Type-flaw attack on Needham-Schroeder

name type-flaw attack on a published protocol upon which previously no attacks were known. In the attack, using a dynamically created name, the intruder takes advantage of a type flaw to learn a shared key. This type-flaw attack would not be possible with a static name.

The new attacks and intruder capabilities presented here were discovered during a systematic analysis of the Cremers-Mauw semantics [11] and have, to the best of our knowledge, not been taken into account in any formalization.

The paper is structured as follows. We introduce the general notion of chosen-name attack and present our main results in Section 2, we discuss related work in Section 3, and we conclude in Section 4.

2 Chosen-name attacks

2.1 Preliminaries

A *chosen-name attack* occurs when the intruder violates a security property by generating a suitable name or identity for an agent. Clearly, the less restricted an agent's name space is, the more likely this type of attack is to occur.

We distinguish between two types of chosen-name attacks. In a *selected-name attack*, the intruder can select arbitrary names for compromised agents only, while in an *assigned-name attack* the intruder may additionally assign arbitrary names to uncompromised, i.e. honest agents. Aside from the fact that assigned-name attacks are much more specialized than selected-name attacks, the two classes also have distinct targets. In selected-name attacks malicious agents choose their name to attack other agents, hence the veracity of security properties for these agents may be ignored, while in assigned-name attacks the victim may very well be the agent that is being assigned a name. Note that while the attacker may assign a name to an honest agent in order to attack another victim, in general these attacks can also be executed as selected-name attacks.

Although attacks that would fit our definition of chosen-name attacks have been known and described in the literature, they have not been treated as a class of

their own, but have been rather occurring as instances of other classes, such as impersonation attacks, man-in-the-middle attacks, or relay attacks. The *chosen-name type-flaw* attacks, however, are new and did not receive attention before. In the rest of this section we will therefore restrict ourselves to chosen-name type-flaw attacks. We will discuss existing chosen-name attacks in Section 3.

2.2 Selected-name attacks

We consider attacks where the intruder dynamically selects the names of conspiring agents in such a way that a security claim fails due to a type flaw. We split these selected-name attacks into two subclasses. The first class consists of those attacks where an agent's selected name will be accepted without further scrutiny, while the second class requires the name to be confirmed or accepted by a third party, for instance by a certificate authority or a key server.

We begin by presenting and discussing a protocol vulnerable to an attack from the first class and then discuss Lowe's modification of the KSL protocol which is vulnerable to an attack from the second class. In both cases we only consider secrecy, even though our methods can be used to attack any security property.

2.2.1 A flawed key-establishment protocol

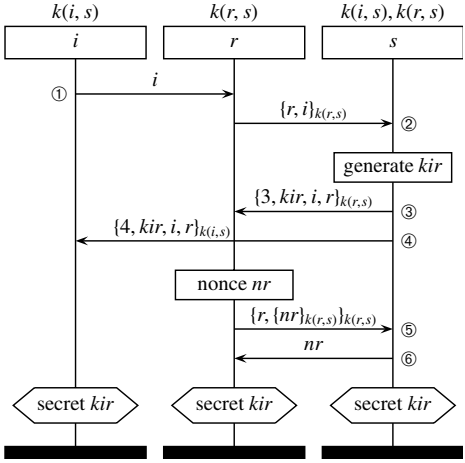

Fig. 3. Key-establishment protocol

Consider the protocol in Figure 3. It is a fictitious key establishment protocol combined with a liveness check. The premise is that initiator i and server s share the secret key $k(i, s)$ and similarly responder r and server s share the secret key $k(r, s)$. On i's communication request to r in message ①, r contacts the server s in ②, who then distributes a fresh secret key kir to r and i in messages ③ and ④. When r receives the new key kir he generates a nonce nr to check liveness of the server in ⑤ and ⑥. To demonstrate the selected-name attack, the liveness check is implemented in a non-standard manner. The security claim of this protocol for all three roles is that kir is secret, as indicated by the hexagons at the end of the protocol.

Since this is a specially crafted protocol, we first sketch why the protocol achieves its security goals in the absence of type-flaw attacks. The key kir is freshly produced by s, and transmitted only in messages ③ and ④, encrypted with $k(r, s)$ and $k(i, s)$, respectively. In both messages, kir is concatenated with i and r and encrypted with keys known only to s and one of i and r. Thus each of the messages binds kir, i, r, and s together. It follows that kir is secret for i, since all messages containing kir are bound to the intended agents, must have been produced by the intended agent, and are only readable by the intended agents. The same reasoning can be applied for r and s's secrecy claim.

Next, we consider conventional type flaws. It is evident that messages ②, ③, and ④ cannot interfere with each other due to the message identifiers contained in messages ③ and ④. Messages ② and ⑤ are supposed to be distinguishable by the fact that one contains an agent name and the other one an encryption term. In a conventional type-flaw attack, an adversary may attempt replacing message ⑤ in a certain run by message ②, possibly from another run. In the present protocol, this attack would be futile, as it would, at best, lead to an encryption term $\{i\}_{k(r,s)^{-1}}$. This term would not be useful for the adversary to break the secrecy of kir, since it does not fit the type of any other message.

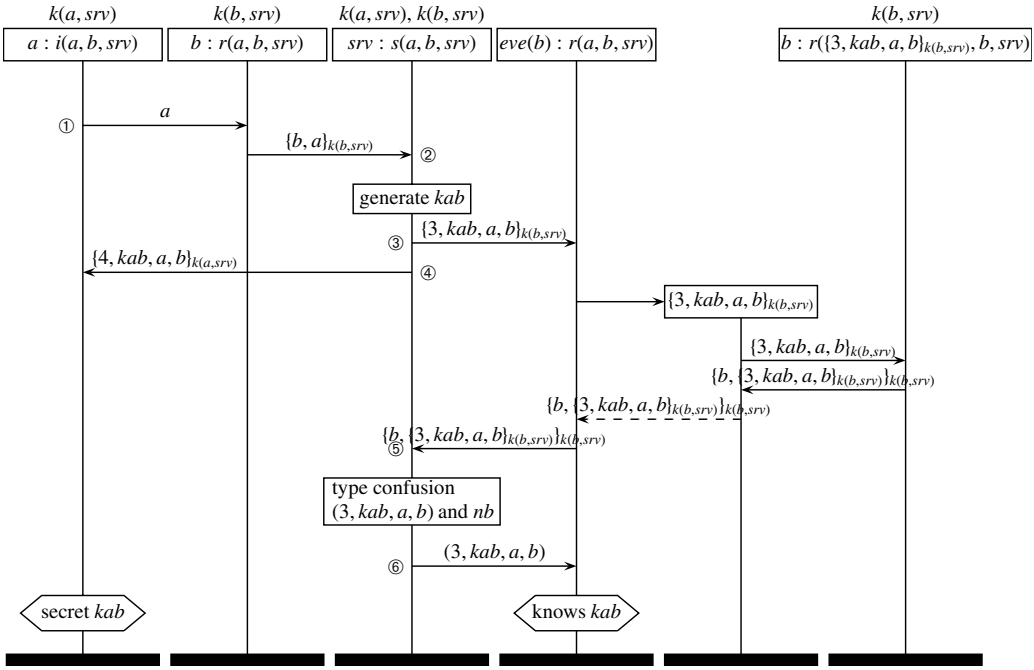

Fig. 4. Attack on key-establishment protocol

Finally, in a selected-name attack, the replacement can be attempted in the other direction, too, i.e. message ② in a certain run may be replaced by message ⑤ from another run. Figure 4 shows a trace demonstrating this attack. Assume that the adversary controls two agents, an agent called *eve* who pretends to be b and an agent who will be named $\{3, kab, a, b\}_{k(b,srv)}$. Agent *eve* listens to a conversation

between the honest agents a, b, and srv. When message ③ is sent from srv to b, eve intercepts this message and the agent with the name $\{3, kab, a, b\}_{k(b,srv)}$ is created. This agent initiates a session with a second run of b. Following the protocol, agent b constructs the message $\{b, \{3, kab, a, b\}_{k(b,srv)}\}_{k(b,srv)}$ which he sends to srv. The adversary intercepts this message and injects it into the first session as message ⑤, impersonating b to srv. Agent srv tries to decrypt the message and obtains $(3, kab, a, b)$ due to a type confusion. This quadruple is sent back in the clear allowing the adversary to learn kab.

2.2.2 Lowe's modified KSL

The selected-name attack presented in the previous section only required the adversary to select names for the agents he controls. Some selected-name attacks additionally require the conspiring agent to have his selected name accepted by a third party, for instance when obtaining a symmetric key associated with the name from a key server or a certificate from a certificate authority. The ability to obtain key material or certificates for a chosen name is plausible in identity-based encryption and signature schemes and in systems where users may have one or more pseudonyms.

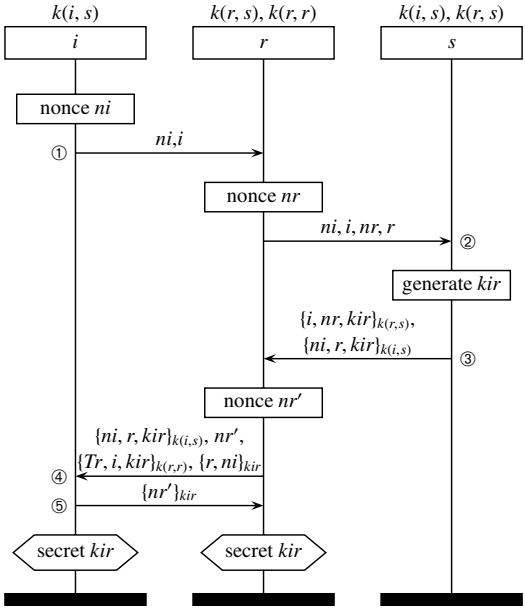

Fig. 5. Lowe modified KSL

As an example of a protocol vulnerable to a selected-name attack under the assumption that the chosen name is accepted, we consider the KSL protocol [16] including the modifications suggested by Lowe in [20]. In [22] an exact modeling of Lowe's modifications is provided. We will focus on the authentication phase of this protocol, shown in Figure 5, and omit the reauthentication protocol.

This protocol is similar to the key-establishment protocol discussed in the previous example in that in messages ① through ③ i contacts r, who in turn contacts the

key server s to obtain shared secret keys. Here, however, nonces are generated and sent in the first two messages, and neither of the first two messages is encrypted. Further, the server s does not deliver the encrypted shared secret key to i directly, but rather sends it to r in message ③, who forwards the encrypted key along with another fresh nonce nr', a ticket $\{Tr, i, kir\}_{k(r,r)}$, and i's original nonce encrypted under the shared secret key to i. Finally, i sends back nr' encrypted with the received shared secret key kir. The ticket in message ④, is only used for the reauthentication protocol which we have omitted. It is encrypted with the key $k(r, r)$ known only to r and contains a generalized time stamp, Tr, made with respect to r's local clock. The fact that the key $k(r, r)$ is only known to r prevents everybody but r to tamper with the ticket or create such a ticket. In the reauthentication protocol r uses Tr to check the validity of the ticket.

Until now, there have been no attacks known on this protocol. In fact, if our chosen-name attacks are disregarded, then the secrecy claims of the protocol can be shown to be correct using, for instance, the Cremers-Mauw semantics [11].

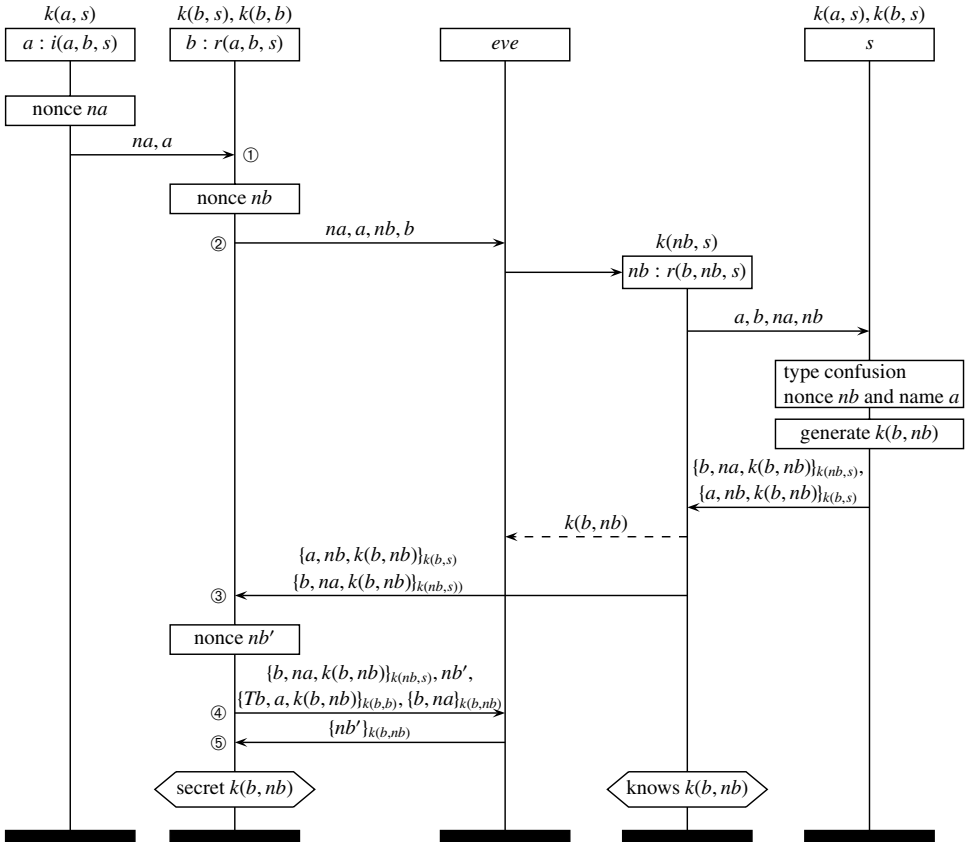

Fig. 6. Attack on Lowe modified KSL

To carry out a selected-name attack, as described in Figure 6, the attacker waits for a to initiate a session with b and s. The adversary then creates an agent with the name nb which he observed in message ②. This agent obtains a valid key $k(nb, s)$ and pretends that b has initiated a session with him by sending the message (a, b, na, nb)

to s. In this message s interprets a as a nonce and nb as a name and responds with a newly generated key, $k(b, nb)$, for b and nb. Agent nb can decrypt the first part of the message to learn the key $k(b, nb)$. He then reverses the order of the two parts of the message and forwards them to b. Agent b decrypts $\{a, nb, k(b, nb)\}_{k(b,s)}$ and thinks that $k(b, nb)$ is the freshly generated key that he should use in his session with a. He then forwards the ticket $\{b, na, k(b, nb)\}_{k(nb,s)}$ together with a newly created nonce nb' to a. The adversary intercepts this message and respond to it by encrypting the nonce nb' with the key $k(b, nb)$ and impersonating a.

2.3 Assigned-name attacks

So far we have considered the adversary's ability to select the names of conspiring agents. In some settings, however, the adversary might even be able to assign names to honest agents. One example would be a compromised naming authority, another possibly more realistic example, would be a compromised DHCP server. In the latter scenario, a protocol which uses IP-addresses to identify agents could be vulnerable to an assigned-name attack.

Consider a variant of the Needham-Schroeder-Lowe (NSL) protocol where the nonces in the second message have been swapped as shown in Figure 7. The NSL protocol is a mutual authentication protocol that has originally been shown to be correct by Lowe [21] and since then by several other authors as well. The swapping of the two nonces has no influence on the correctness of the protocol even when conventional type flaws are taken into consideration. For simplicity, we are restricting ourselves to the secrecy claims of the protocol.

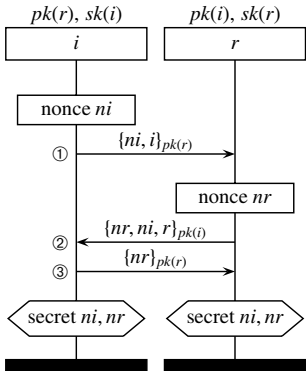

Fig. 7. A variant of NSL

Figure 8 demonstrates an assigned-name attack on the NSL variant. An honest agent b starts a conversation with a malicious agent eve by sending $\{nb, b\}_{pk(eve)}$. The adversary then assigns the name (nb, e) to another honest agent. This honest agent starts a conversation with b and produces an encryption term of the form $\{nnbe, (nb, e)\}_{pk(b)}$. The conversation between the two honest agents continues and at the end of the protocol, (nb, e) and b agree on a secret value $nnbe$. The adversary takes the first message of this conversation and inserts it into the running session between b and eve. Agent b receives this message and confuses the name (nb, e)

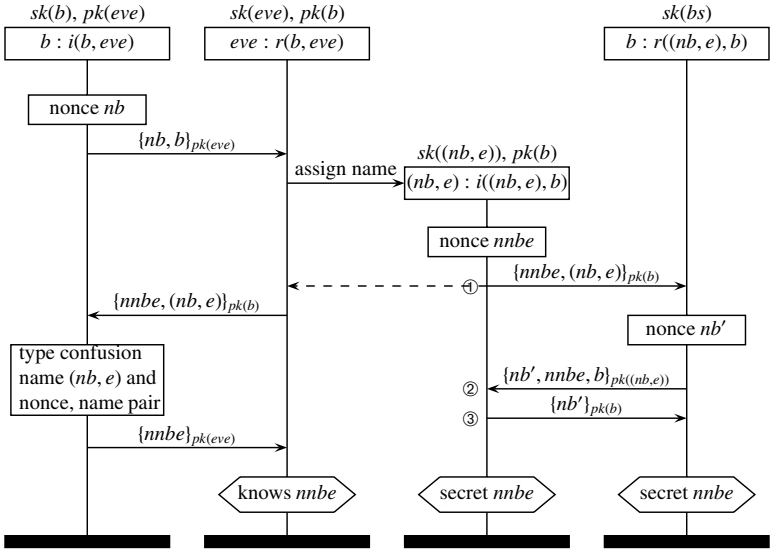

Fig. 8. Assigned-name attack

with nonce nb and name eve and responds with the message $\{nnbe\}_{pk(eve)}$ which enables the adversary to learn the value $nnbe$. Thus, the secrecy of $nnbe$ claims of the honest agents (nb, e) and b are falsified by this attack.

This attack can be modified to impersonate b to nb and invalidate both secrecy claims of nb as follows. When (nb, e) sends out the first message of the protocol, the adversary can block the communication between the agents (nb, e) and b and inject the message $\{nnbe, (nb, e)\}_{pk(b)}$ into his run with b to learn $nnbe$. He then picks a nonce ne to construct the message $\{ne, nnbe, b\}_{pk((nb,e))}$. The adversary now knows both nonces and has furthermore impersonated b to nb. The security claims of b are not invalidated though, since b does not finish the protocol.

3 Related Work

The attacks we have described in this paper belong to the intersection of two classes, namely chosen-name attacks and type-flaw attacks.

Chosen-name attacks have been known and described in the literature in various forms. For instance, it is known that in public key infrastructures a malicious or sloppy certificate authority would make it possible for an attacker to impersonate any user by registering under the user's name or a slight variation of the user's name. A particular instance of this attack, which is known as the *homograph* or *unicode* attack [14], is the registration of Internet domain names resembling well-known domain names. This attack became particularly popular when internationalized domain names became available, since, for instance, several Cyrillic characters are identical to Latin characters allowing two distinct Internet domain names to have the same appearance.

A cryptographic impersonation attack, due to flawed key certification schemes, has been described by Lenstra and Yacobi [17]. In principle, such attacks can also

be carried out on identity-based encryption schemes if the private key generation algorithm is weak. For instance, in Boneh and Franklin's scheme, it is easy to see that the possibility of a chosen-name attack hinges on the quality of the cryptographic hash function H_1 [6, Section 4.1].

Another source of chosen-name attacks are man-in-the-middle attacks on authentication protocols. A malicious agent seeking access to a resource would wait for an honest agent to initiate a vulnerable authentication protocol and consequently select the honest agent's name to perform the attack. In fact, the attack in Figure 2 is another example of a chosen-name man-in-the-middle attack. The attacker chooses and impersonates the agent b to obtain access to a. Similarly, in relay attacks, for instance on protocols running on radio frequency devices, a rogue device would forward the authentication challenge it receives to any victim it can find in the vicinity.

While all these attacks are well-known and have been extensively studied, they are different from the type-flaw attacks considered in this paper, either in that they are not type-flaw attacks at all, or in that the chosen name is *static*.

Since the introduction of type flaws in security protocol analysis [7] various approaches have been used to detect and prevent type flaws. In [15] a tagging scheme is presented that prevents simple type flaws. Simple type flaws occur when one variable is unified with a complex term or a variable of another type.

More complex type-flaw attacks are described in [23]. These attacks emerge when tags are confused with terms or when parts of a term are confused with another term. The detection of complex type flaws is formalized in [23,24,18,19]. Research in this area focuses on the transitions from abstract message specification into concrete bit strings and vice versa.

Some of the formal frameworks aiming at verification of security protocols have included the concept of simple type-flaw attacks in their models, for instance [26,1,11]. We have investigated whether the tools based on these models, namely ProVerif [4], Scyther [10], the constraints solver in prolog [25], and the four tools of the Avispa project (CL-Atse[27], OFMC[3], SATMC [2], TA4SP [5]), are able to detect chosen-name attacks. These tools cover most of the modern techniques used in protocol verification, such as model checking, constraint solving, SAT-solving, and approximation. Since all of the selected tools provide a specification of the NSL protocol, only minor modifications were necessary to test the NSL variant in Figure 7. None of the selected tools were able to detect the selected-name attack described in Figure 8.

For Scyther and the Avispa tools it is easy to see why the attack could not be found. Scyther has a fixed domain out of which the names of agents are picked. The reason why none of the Avispa tools was able to find an attack is related to their input language, HLPSL. This language requires the user to define a set of concrete sessions under consideration. This set typically only contains sessions between agents with normal names. In order to find a chosen-name attack, one has to set up a session where the name of one of the agents is a concrete run term. Since the set of concrete run terms is infinite, it is not possible to list all

potential chosen-name attack scenarios. This implies that for an Avispa tool using the HLPSL input language to find a chosen-name attack, the attack has to be known in advance. OFMC cannot find chosen-name type-flaw attacks, even in its native input language, due to an over-optimizing design choice in its symbolic session generation algorithm [3, §6.3].

We could not pinpoint the exact reason why the constraint solver in prolog did not find the assigned-name attack, as it seems that this formalism does not require a special domain for the names of honest agents. This formalism does however limit the names of the attacker, as a constant ϵ is used to represent his name, and thus precludes the detection of selected-name attacks. In ProVerif the default implementation of NSL uses the public key of an agent to identify the agent. Instead of sending $\{na, nb, b\}_{pk(a)}$ the second message is modeled by $\{na, nb, pk(b)\}_{pk(a)}$. Another way to model agent names in ProVerif is via the $host()$ function, but even in that case, the attack could not be found.

Most formal models underlying tools for verification of security protocols can be extended to express chosen-name attacks. However, it will not necessarily be easy to extend the tools themselves. Especially tools that search through the state space of a given finite scenario will face the problem of having to choose appropriate agent names from an infinite domain.

4 Conclusion

In this paper we have presented an intruder ability which was overlooked in common interpretations of the Dolev-Yao threat model and we demonstrated how this ability can be used to construct a special class of type-flaw attacks. We have identified a structure related to this intruder ability and classified the newly found attacks.

We have shown that Lowe's modified KSL protocol is vulnerable to a selected-name attack and that a mere reordering of two nonces renders the Needham-Schroeder-Lowe protocol vulnerable to an assigned-name attack.

Type-flaw attacks on a protocol are intimately related to the implementation of the protocol. The attacks presented in this paper are infrequent, but as realistic as any other type-flaw attack and therefore should be taken into account by those tools and models which attempt to detect type flaws. Protocols vulnerable to this new class of attacks can be corrected like protocols vulnerable to type-flaw attacks by rearranging fields in messages, by adding extra information in vulnerable messages, such as was for instance done in messages ③ and ④ of the fictitious key-establishment protocol in Section 2.2.1, or by using tagging schemes such as those proposed in [15]. A way to prevent chosen-name type-flaw attacks in particular, is to precisely define the agent's name space and enforce strict name checking.

This work shows that the common Dolev-Yao interpretation is not complete with respect to the requirement that the adversary *tries everything he can* in order to learn a certain message. For instance, in [9] it is shown that from any attack on a secrecy claim involving n agents, an attack can be constructed which involves only two agents, assuming that agents may talk to themselves. The construction

essentially maps all dishonest agents to one agent and all honest agents to the other agent. The attacks introduced in this paper indicate that the security of a protocol can depend on the names of the agents. It is possible to construct protocols where an attack requires the adversary to select several names for dishonest agents. If one agent can only have one name, such an attack requires more than two agents. This shows that the results in [9] do not hold under the present intruder model. It is conceivable that there are other subtle assumptions made in the common interpretation of Dolev-Yao.

Acknowledgement

We thank Cas Cremers, David Basin, Kristian Gjøsteen, and Suzana Andova for discussions and helpful comments.

References

[1] A. Armando, D. Basin, M. Bouallagui, Y. Chevalier, L. Compagna, S. Mödersheim, M. Rusinowitch, M. Turuani, L. Viganò, and L. Vigneron. The AVISS Security Protocol Analysis Tool. *Proceedings of CAV*, 2:349–354, 2002.

[2] A. Armando and L. Compagna. An Optimized Intruder Model for SAT-based Model-Checking of Security Protocols. *Proc. of ARSPA*, 2004, 2004.

[3] David A. Basin, Sebastian Mödersheim, and Luca Viganò. OFMC: A symbolic model checker for security protocols. *Int. J. Inf. Sec.*, 4(3):181–208, 2005.

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

[5] Y. Boichut, P.-C. Héam, O. Kouchnarenko, and F. Oehl. Improvements on the Genet and Klay technique to automatically verify security protocols. In *Proc. Int. Ws. on Automated Verification of Infinite-State Systems (AVIS'2004), joint to ETAPS'04*, pages 1–11, Barcelona, Spain, April 2004.

[6] Dan Boneh and Matt Franklin. Identity-based encryption from the Weil pairing. *Lecture Notes in Computer Science*, 2139:213–229, 2001.

[7] C. Boyd. Hidden assumptions in cryptographic protocols. *Computers and Digital Techniques, IEE Proceedings-*, 137(6):433–436, 1990.

[8] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. In *Practical Cryptography for Data Internetworks*. IEEE Computer Society Press, 1996. Reprinted from the Proceedings of the Royal Society, volume 426, number 1871, 1989.

[9] H. Comon-Lundh and V. Cortier. Security properties: two agents are sufficient. *Science of Computer Programming*, 50(1-3):51–71, 2004.

[10] C.J.F. Cremers. *Scyther — Semantics and Verification of Security Protocols*. PhD thesis, Eindhoven University of Technology, 2006.

[11] C.J.F. Cremers and S. Mauw. Operational semantics of security protocols. In S. Leue and T.J. Systä, editors, *Scenarios: Models, Algorithms and Tools (Dagstuhl 03371 post-seminar proceedings, September 7–12, 2003)*, volume 3466 of *LNCS*, pages 66–89, 2005.

[12] A. Datta, A. Derek, J. C. Mitchell, and A. Roy. Protocol composition logic (PCL). *Electronic Notes in Theoretical Computer Science*, 2007. Gordon D. Plotkin Festschrift, to appear.

[13] D. Dolev and A.C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, IT-29(12):198–208, March 1983.

[14] Evgeniy Gabrilovich and Alex Gontmakher. The homograph attack. *Commun. ACM*, 45(2):128, 2002.

[15] James Heather, Gavin Lowe, and Steve Schneider. How to prevent type flaw attacks on security protocols. *J. Comput. Secur.*, 11(2):217–244, 2003.

[16] Axel Kehne, Jürgen Schönwälder, and Horst Langendörfer. A nonce-based protocol for multiple authentications. *Operating Systems Review*, 26(4):84–89, 1992.

[17] Arjen K. Lenstra and Yacov Yacobi. User impersonation in key certification schemes. *J. Cryptology*, 6(4):225–232, 1993.

[18] Benjamin W. Long. Formal verification of type flaw attacks in security protocols. In *APSEC '03: Proceedings of the Tenth Asia-Pacific Software Engineering Conference Software Engineering Conference*, page 415, Washington, DC, USA, 2003. IEEE Computer Society.

[19] B.W. Long, C.J. Fidge, and D.A. Carrington. Cross-layer verification of type flaw attacks on security protocols. In G. Dobbie, editor, *Proceedings of the 30th Australasian Computer Science Conference (ACSC 2007)*, pages 171–180, 2007.

[20] G. Lowe. Some new attacks upon security protocols. *Proceedings of the 9th IEEE Computer Security Foundations Workshop*, pages 162–169, 1996.

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

[22] LSV, ENS Cachan. Security Protocols Open Repository. http://www.lsv.ens-cachan.fr/spore.

[23] C. Meadows. Identifying potential type confusion in authenticated messages. In *Proceedings of Foundations of Computer Security*, 2002.

[24] C. Meadows. A procedure for verifying security against type confusion attacks. In *16th IEEE Computer Security Foundations Workshop (CSFW-16 2003)*, pages 62–72, 2003.

[25] J. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. *Proceedings of the 8th ACM conference on Computer and Communications Security*, pages 166–175, 2001.

[26] F.J. Thayer Fàbrega, J.C. Herzog, and J.D. Guttman. Strand spaces: Why is a security protocol correct? In *Proc. 1998 IEEE Symposium on Security and Privacy*, pages 66–77, Oakland, California, 1998.

[27] M. Turuani. The CL-Atse Protocol Analyser. *17th international conference on term rewriting and applications-rta*, pages 277–286, 2006.