AN INDUSTRIAL AND ACADEMIC JOINT EXPERIMENT ON AUTOMATED VERIFICATION OF A SECURITY PROTOCOL

OLIVIER HEEN
IRISA, Lande Project, Rennes, France

THOMAS GENET
IRISA, Lande Project, Rennes, France

STEPHANE GELLER
ENS Cachan, Rennes, France

NICOLAS PRIGENT
Thomson R&D France, Security Lab, Rennes, France

This paper relates the collaboration between industrial and academic teams for the design and the verification of a security protocol. The protocol is about trust establishment in large communities of devices where infrastructure components are not always reachable. The collaboration covers the writing of formal specifications up to their verification, using both manual and automated verification methods embedded in the AVISPA [1] and SPAN [7] tools. At each stage, the use of the visualization and protocol animation facilities of SPAN is key to the mutual understanding of working teams. As a result, we obtain much more confidence in the security of the final protocol. We also demonstrate the usefulness of some embedded countermeasures.

1. Introduction

As they expand, digital transmissions require increased security. Sometimes a straightforward adaptation of widely known solutions (SSL, IPSEC, PGP…) is adequate. Sometimes a new protocol must be specifically designed. In this case, many sources of error exist:

- The protocol is an answer to a recent security problem: all the aspects of the problem may already not be known.
- The protocol is an answer to a time critical situation: the design time may be very short.

- The protocol is designed for commercial use in a competitive domain: some details may not be published too early, and thus no external review is possible.

Because of such error factors there is an urge need for formal verification. Moreover, if the formal verification can be automated, this gives a chance to systematically verify the protocol after each update and to deal with the complexity of some protocols that cannot be managed by hand in practice.

This paper relates the collaboration of industrial and academic teams about specifying and verifying one protocol. For convenience in the rest of the paper, we call our protocol LCDP for *Large Community of Device Protocol*. It is an adaptation of a symmetric key authentication scheme [4] to the case of large communities of devices. This collaboration illustrates the advantages of formal verification from a practical point of view: at first, it leads to a more precise specification of LCDP; then it gives more confidence on the final version, since no tools found attacks. This is especially true for the parts of the protocol that are completely new, and thus not heavily reviewed. At last, we check downgraded versions of LCDP where some countermeasures are disabled: this leads to the discovering of non-trivial attacks, and thus provides better justifications for the chosen countermeasures.

At each stage (formal specification, modeling, verification), the visualization and the interactive use of execution diagrams is key for the mutual understanding of both teams: the industrial team developing the LCDP, and the academic team leading the formal specification and verification effort. Throughout the paper, we provide some of the message sequence charts obtained with SPAN.

The section 2 gives the basics of automated protocol verification in the Dolev-Yao model. The tools that we use are also described. The section 3 gives motivations for LCDP and its description. The section 4 relates the whole verification process.

2. About automated protocol verification

This section is a short introduction to the automated verification of protocols in the Dolev-Yao model (that was introduced in [5]). The reader familiar with automated protocol verification can skip this section.

2.1. *Verification of Diffie-Hellman*

We briefly introduce the verification on a well-known example: the Diffie-Hellman key agreement protocol [6]. It is presented below, using the so-called

``Alice & Bob" notation. The agents are denoted by A and B and the established key is denoted by K. This key is used in the final step to encrypt a secret message msg sent by A to B.

1. $A \rightarrow B: g^{Na}$
2. $B \rightarrow A: g^{Nb}$, A and B compute key $K=(g^{Na})^{Nb}=(g^{Nb})^{Na}$
3. $A \rightarrow B: \{msg\}_K$

At step 1, A generates the *nonce* (a random number) N_a and computes g^{Na} where g is a public number. Then A sends g^{Na} to the agent B. At step 2, the agent B also generates a number N_b and computes g^{Nb} and $K=(g^{Na})^{Nb}$. The former is sent to A and the latter stands for the symmetric key shared between A and B.

As soon as A receives g^{Nb} from B, it computes $(g^{Nb})^{Na}$ and considers it as the symmetric key shared with B. Indeed, according to the algebraic properties of the exponentiation, $K=(g^{Na})^{Nb}=(g^{Nb})^{Na}$. Finally, the message $\{Msg\}_K$ is sent by A to B in which Msg is a datum standing to be secret between A and B, and $\{\}_K$ denotes the use of a symmetric encryption algorithm using the key K.

Security protocols can be attacked in several ways. If the keys or the algorithms used for ciphering messages are not robust enough, the content of the messages can be obtained or modified by an attacker. Such attacks are more related to cryptanalysis and can generally be avoided with a careful choice of keys and algorithms when implementing the protocol. The attacks we are interested in are based on a malicious use of the protocol itself. The Dolev-Yao model is particularly well suited for this kind of attacks. The intruder can read, block, store, modify and send messages over the network. It is said that *the intruder is the network.*

Hereafter, we show the well known man-in-the-middle attack against the Diffie-Hellman protocol. The notation I(A) means that the intruder pretends to be A.

1. $A \rightarrow I: g^{Na}$
2. $I(A) \rightarrow B: g^{Ni}$
3. $B \rightarrow I: g^{Nb}$, B and I compute the key $K_{IB}=(g^{Ni})^{Nb}=(g^{Nb})^{Ni}$
4. $I(B) \rightarrow A: g^{Ni}$, A and I compute the key $K_{IA}=(g^{Ni})^{Na}=(g^{Na})^{Ni}$
5. $A \rightarrow I: \{Msg\}_{K_{IA}}$

Roughly, the intruder establishes two keys: $K_{IA}=(g^{Na})^{Ni}$ with A at Steps 1 and 4, and $K^{IB}=(g^{Nb})^{Ni}$ with B at Steps 2 and 3. He then acts as a proxy between A and B. At Step 5, the agent A sends the secret data to B using the key K_{IA} shared with the intruder. The intruder then extracts the secret data.

2.2. *Verification tools*

Due to the intrinsic complexity of real-life protocols, their formal verification is unlikely to be performed by hand. For instance, the Needham-Schroeder Public Key protocol [10] was proved correct by hand [3] though it revealed to be flawed [9] when carefully analyzed using formal methods.

The verification in the Dolev-Yao model has been implemented in several tools. The AVISPA framework [1] is one such tool. It is very convenient, especially when completed with SPAN [7] for the visualization and animation parts. AVISPA and SPAN both use formal specifications of the protocols, written in the language of AVISPA called HLPSL (High Level Protocol Specification Language). Hereafter is a specification of the Diffie-Hellman protocol using HLPSL.

```
role alice(A,B:agent, G:text, Snd,Rcv:channel(dy)) played_by A def=
    local State:nat, Na,Msg:text, X,K:message init State:=1
transition
1.  State=1 /\ Rcv(start) =|> State':=2 /\ Na':=new() /\ Snd(exp(G,Na'))
2.  State=2 /\ Rcv(X') =|> State':=3 /\ K':=exp(X',Na) /\ Msg':= new() /\
    Snd({Msg'}_K')
end role

role bob(B,A:agent, G:text, Snd,Rcv:channel(dy)) played_by B def=
    local State:nat, Y,K:message, Nb,Msg:text init State:=1
transition
1.  State=1 /\ Rcv(Y') =|> State':=2 /\ Nb':=new() /\ K':=exp(Y',Nb') /\
    Snd(exp(G,Nb'))
2.  State=2 /\ Rcv({Msg'}_K) =|> State':=3
end role
```

The specification is based on role descriptions, i.e. finite state automata, where transitions are fired when a message is sent or received. Contrary to ``Alice & Bob" notation, HLPSL imposes explicit definition of roles, nonce generation, message emission and reception, etc. In HLPSL, `=|>` stands for the transition relation and `/\` stands for the usual conjunction symbol. The HLPSL is based on a notation *à la* TLA where the meaning of a primed variable `X'` depends on the location of this variable. Indeed, if `X'` occurs in a message pattern of the left-hand side of a transition then a new value is obtained for `X` by matching the message pattern on received messages. Then this value is accessible by `X'` in the same transition.

Hereafter is an attack automatically found by one AVISPA tool called ATSE. Note that this attack is not the usual man-in-the-middle attack. We believe that this is due to ATSE providing the shortest attack first.

```
SUMMARY       UNSAFE
DETAILS       ATTACK_FOUND TYPED_MODEL
PROTOCOL      Diffie-Hellman.if
GOAL          Secrecy attack on (n2(Msg))
BACKEND       CL-AtSe
STATISTICS    [...]
ATTACK TRACE
  i -> (a,3): start
  (a,3) -> i: exp(g,n1(Na))
  i -> (a,3): g
  (a,3) -> i: {n2(Msg)}_(exp(g,n1(Na))) & Secret(n2(Msg),set_53);
              Add a to set_53; Add b to set_53;
```

Figure 1 is a visualization of the man-in-the-middle attack built using SPAN. Starting for the HLPSL specification, SPAN lets the operator choose any possible transition, until there is no more possible transition.

In figure 1, the operator has chosen a transition sequence that leads to the attack. See [2] for more details about the formal specification of the Diffie-Hellman protocol and its verification using AVISPA and SPAN.

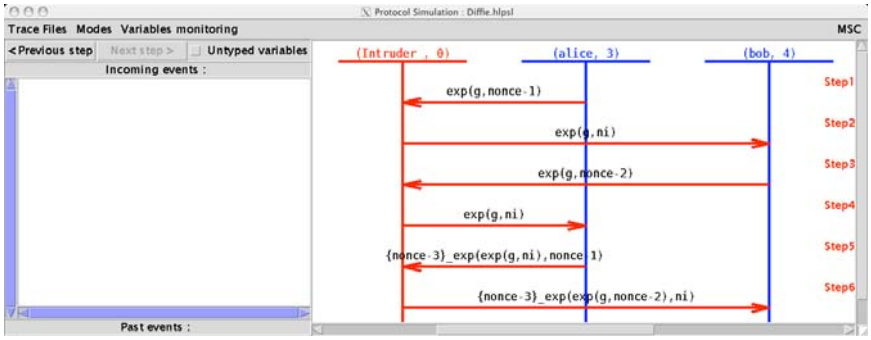

Figure 1: Visualization of the man in the middle attack with SPAN

3. Brief description of LCDP

This section describes LCDP and gives indication on how it was specified. The focus of the paper is the verification in section 4, but we first provide some rational for understanding LDCP. LCDP is meant to establish trust within large communities of devices. Here, the trust is the mutual authentication between community devices and, ultimately, the sharing of a secret key (as in [8] for instance). We believe that such communities will increasingly exist because of the generalization of cheap devices with network connectivity and the move towards ubiquitous computing [13]. In these communities the infrastructure

components (internet server, home center, WiFi base…) are not always reachable, and thus should not be used often, in an opportunistic manner. One typical example is a cloud of home devices and personal devices willing to collaborate, but not always connected to a central home server. Moreover, the use of asymmetric cryptography on devices must be sparse because of their low computation capabilities. Refer to [8,11] for previous work by two authors on secure communities of devices.

3.1. *Symmetric key authentication*

LCDP is an adaptation of an existing symmetric key authentication scheme [4] in the case of large communities of devices. The principle is to allow devices to progressively acquire a set of trusted other devices. Once trust is established, devices can securely communicate on a one-to-one basis. To achieve this, LCDP uses a mix of several techniques and components. The components are: one authentication server, at least one directory server and the devices. The techniques are: Public Key Infrastructure (PKI) between devices and authentication server, symmetric cryptography between devices and directory servers, symmetric cryptography again between devices.
The notations used to describe LCDP are: AS for the authentication server, DS for the directory server, $D_1 \ldots D_i$ for devices having identities $Id_1 \ldots Id_i$ respectively.
In a typical session, a device D_1 first connects to AS, using its PKI credentials. If D_1 is authorized, then AS sends it a cryptographic ticket t_1. D_1 presents t_1 to DS in order to register. DS informs D_1 about other devices $\{D_i,\ldots\}$ willing to communicate. For each D_i, DS also provides a specific ticket $t_{1,i}$. Only then may D_1 securely communicate with D_i by first presenting $t_{1,i}$.

3.2. *LCDP details*

The PKI part of LCDP can be a very common X509 architecture. The authentication server is typically a SSL server. It acts as a guard against non-authenticated devices. It also enforces the revocation of devices. The devices connect the authentication server only when they want to enter into a LCDP session. This happens typically once every day. Since we are in ad hoc context, we do not assume that the authentication server is always reachable. Moreover, the use of a PKI can be no more that opportunistic. For instance, the revocation information will be received by device only on a best effort basis.

The directory server is contacted by devices whenever they need references for other devices. This can happen often (e.g. every minute for every device)

especially if there are many devices, hence the use of symmetric cryptography for this part. As discussed in [4], this is also a way to mitigate the relative weakness of the directory server being a single point of failure.

Because it is difficult to keep control over large communities of devices and because LCDP uses some long terms tickets (see 3.3), one global control mechanism is also added. This is one global symmetric key, chosen by the authentication server and shared with the directory server. It is denoted KoD for "*Key of the Day*", stressing the fact that the authentication server regularly changes it, typically on a daily basis. It is used as a mandatory component in the calculation of tickets. Changing KoD invalidates all the previously calculated tickets. This mechanism provides an authoritative way to globally discard all previous trust relations and to force devices acquiring fresh tickets everyday. Also, it can be useful in case of a mass revocation of devices that should quickly become effective. There is one last advantage of using KoD: we do not want to use the time in LCDP, mainly because some other devices do not have reliable time mechanism. In particular it could be difficult to set up tickets with specific time duration. Instead, KoD provides a way to obsolete all tickets in a single operation.

Because “devil lies in the details” we are very suspicious about the security of the KoD part of the protocol. The verification of it is one important motivation for the collaboration with academic team and for the intensive use of formal tools.

3.3. *Putting it altogether*

The specification of LCDP is provided here in the following formats: one generic execution diagram in figure 2 and one ``Alice & Bob" trace including the formulas for the computation of tickets. In the trace, we assume that a device D_i already is in a LCDP session. Also, we assume an existing secure channel between AS and DS.

1. AS → DS: KoD. This happens regularly, for instance once a day, trough a secure channel.
2. D_1 → AS: Id_1. D_1 attempts a SSL connection to AS.
3. AS → D_1: $K_1,Id_1,t_1=\{K_1,Id_1,n_1\}_{KoD}$. If AS authenticates D_1, it gives it ticket t_1 for communicating with DS. n_1 is a nonce used for anti-replay. KoD is used to prevent generation of false tickets or the replay of old tickets.
4. D_1 → DS: Id_1,t_1. D_1 authenticates to DS with its ticket. Decrypting the ticket, DS learns the key K_1.

5. DS → D_1: $\{Id_i,K_{1,i},t_{1,i}\}=\{Id_1,Id_i,K_{1,i},n_{1,i}\}_{Ki}\}_{K1}$. For each D_i willing to communicate, DS gives D_1 a specific ticket.
6. $D_1 \rightarrow D_i$: $Id_1,t_{1,i}$. D_1 presents its ticket to D_i.
7. $D_i \rightarrow D_1$: $\{message\}_{K1,i}$. D_i can now securely send a secret message to D_1.

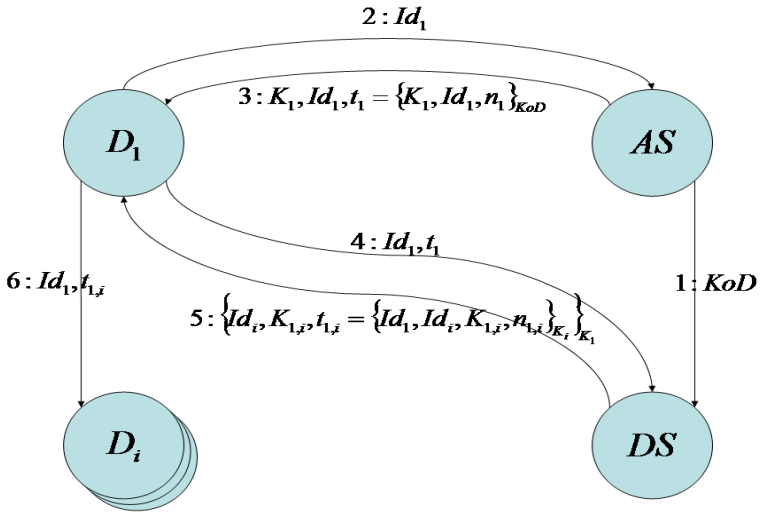

Figure 2: Generic execution diagram of LCDP.

4. Verification of LCDP

4.1. *Formal specification of LCDP*

Before we run manual and automated verifications from AVISPA and SPAN we first need a sufficient specification of LCDP: most of the complexity must be embedded within the specification, otherwise we will miss attacks. On the other hand it shall not become too complex so that tools can handle it correctly and in a reasonable time. As a compromise, we make the following assumptions:

SSL is reliable at least in our context. Thus, at the end of the SSL session between D_i and AS, we can suppose that both parties share one secret key K_{SSLi}.

Low number of devices we set-up LCDP sessions with only a few devices at the same time. Moreover, we restrict the set of devices that will communicate with another to one device at most. This case corresponds to devices that want to communicate by pairs rather than by groups.

One directory server this is an important simplification at the cost of missing attacks that specifically happen with many directory servers.

Eluded details some parts are not formally specified, like the details of the communication between AS and DS, some mechanisms to mitigate consequences of changing KoD, the way devices publish there willingness to communicate with some other.

Remark that these assumptions altogether represent a rather restrictive compromise. Thus it is unlikely that we find an attack, or it will be a very serious one. Our intention is to start with these restrictive assumptions, and then possibly relax some of them, until attacks are eventually found. Doing so, we seek more formal justifications for some protocol features (see end of 4.2).

Taking all above assumptions, we get the HLPSL specification hereafter:

```
Role device1(D1,Di,A,M:agent,Kssl1:symmetric_key,Id1:message,
SND,RCV:channel(dy)) played_by D1 def=
    local State:nat,N1,N2,N:text,Idi,Cred,Ticket:message,K1,K1i:symmetric_key
    init State:=0
transition
1.  State=0 /\ RCV(start) =|> State':=1 /\ N':= new() /\SND({Id1.N'}_Kssl1)
2.  State=1 /\ RCV({K1'.Id1.N.Cred'}_Kssl1) =|> State':=2 /\ SND(Id1.Cred')
3.  State=2 /\ RCV({Idi'.K1i'.Ticket'}_K1) =|> State':= 3 /\ SND(Id1.Ticket')
4.  State=3 /\ RCV({N2'}_K1i) =|> State':= 4
end role

role devicei(Di,D1:agent,Kssli:symmetric_key,Idi:message,SND,RCV:channel(dy))
played_by Di def=
    local State:nat,Msg,N2,N:text,Id1,Cred,Tcred:message,Ki,K1i:symmetric_key
    init State:=0
transition
1.  State=0 /\ RCV(start) =|> State':=1 /\ N':= new() /\SND({Idi.N'}_Kssli)
2.  State=1 /\ RCV({Ki'.Idi.N.Cred'}_Kssli) =|> State':=2 /\ SND(Idi.Cred')
3.  State=2 /\ RCV(Id1'.{Id1'.Idi.K1i'}_Ki) =|> State':=3 /\ Msg':= new() /\
    SND({Msg'}_K1i') /\ secret(Msg',secret_msg,{D1,Di})
end role

role as( A:agent,Kssl1,Kssli,KoD:symmetric_key,SND,RCV:channel(dy))
played_by A def=
    local State:nat,N:text,K:symmetric_key,Adr:message
    init State:=0
transition
1.  State=0 /\ RCV({Adr'.N'}_Kssli) =|>
    State':=1 /\ K':= new() /\ SND({K'.Adr'.N'.{K'.Adr'}_KoD}_Kssli)
2.  State=1 /\ RCV({Adr'.N'}_Kssl1) =|>
    State':=2 /\ K':= new() /\ SND({K'.Adr'.N'.{K'.Adr'}_KoD}_Kssl1)
end role

role ds( M:agent,KoD:symmetric_key,SND,RCV:channel(dy))
played_by M def=
    local State:nat,Idi,Id:message,K,Ki,K1i:symmetric_key
    init State:=0
transition
1.  State=0 /\ RCV(Idi'.{K'.Idi'}_KoD) =|> State':=1
2.  State=1 /\ RCV(Id'.{Ki'.Id'}_KoD) =|> State':=2 /\ K1i':= new() /\
    SND({Id'.K1i'.{Id'.Idi.K1i'}_K}_Ki')
end role
```

```
  role session(D1,Di,A,M:agent,Kssl1,Kssli,KoD:symmetric_key,Id1,
Idi:message) def=
    local SD1,SDi,SA,SM,RD1,RDi,RC3,RC4,RA,RM:channel(dy)
composition
    device1(D1,Di,A,M,Kssl1,Id1,SD1,RD1) /\ devicei(Di,D1,Kssli,Idi,SDi,RDi)
/\  as(A,Kssl1,Kssli,KoD,SA,RA) /\ ds(M,KoD,SM,RM)
end role

role environment() def=
    const d1,di,i,as,ds:agent,
    oldKssli,newKssl1,newKssli,oldKsslintruder,oldKoD,newKoD:symmetric_key,
    id1,idi,idintruder:message, secret_msg:protocol_id
    intruder_knowledge={i,d1,di,as,ds,id1,idi,idintruder,oldKsslintruder}
composition
    session(i,di,as,ds,oldKsslintruder,oldKssli,oldKoD,idintruder,idi) /\
    session(d1,di,as,ds,newKssl1,newKssli,newKoD,id1,idi)
end role
goal secrecy_of secret_msg end goal
```

4.2. *Verifications using AVISPA and SPAN*

We start the verification of LCDP with a single KoD, hence modeling a "one day" usage. This corresponds to a normal use of LCDP when devices first authenticate, then build trust, and then communicate together. For simulating SSL we first preset symmetric keys between the participating devices and the authentication server. Then, devices systematically use theses keys and a nonce for communicating with AS. The nonce expresses the anti-replay feature of SSL. Hence steps 2 and 3 of LCDP specifications are replaced by:

2. $D_1 \rightarrow AS: \{Id_1,n_1\}_{KSSL1}$.
3. $AS \rightarrow D_1: \{K_1,n_1\}_{KSSL1},Id_1,t_1=\{K_1,Id_1,n_1\}_{KoD}$.

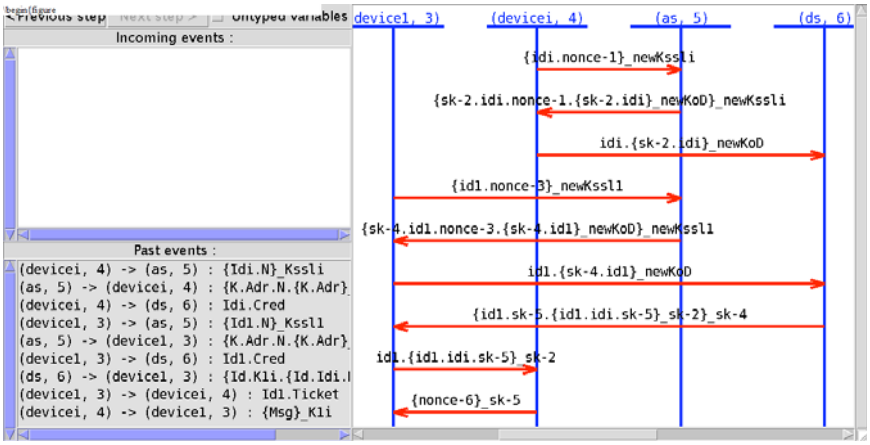

Figure 3: Simple session of LCDP.

The figure 3 shows the corresponding execution with SPAN. We use two of the verification tools from the AVISPA framework: OFMC and ATSE. Note that other tools like SATMC and TA4SP also exist but do not lead to additional result in the context of the present experiment. The verification goal is the secrecy of the messages that are sent between devices. No tool reveals attacks, with respect to the assumptions of 4.1.

We now verify LCDP when a KoD change happens. As indicated in 3.2 we particularly care about potential vulnerabilities in this case. We still use the same method to simulate SSL and we set-up a multiple session using the HLPSL declaration below.

```
session(d1,di,as,ds,oldKssl1,oldKssli,oldKoD,idintruder,idi) /\
session(d1,di,as,ds,newKssl1,newKssli,newKoD,id1,idi)
```

Note that the same devices **d1** and **di** appear in both sessions. The KoD changes from **oldKoD** in the first session to **newKoD** in the second session. Neither OFMC nor ATSE found attacks in this case.

We also verify the case where one device is valid in the first session, but revoked in the second session. In this case, a device could try to keep its privileges even after revocation and KoD change. We code it by making the intruder **i** explicitly playing the role of a valid device in the first session. In the second session, the intruder is not considered anymore as a valid participant.
However, he is still able to manipulate the network and take advantage of the information gathered during the first session. Here again, no attacks are found. This reads:

```
session(i,di,as,ds,oldKsslintruder,oldKssli,oldKoD,idintruder,idi) /\
session(d1,di,as,ds,newKssl1,newKssli,newKoD,id1,idi)
```

At last, one can argue about the real need of SSL at the beginning of LCDP sessions. Maybe this can be replaced by a lighter mechanism and permanent secret keys. We have tried several ways to downgrade this part. This generally leads to attacks, some of them non trivial. Figure 4 shows such a possible attack, when the anti-replay mechanism at the beginning of LCDP is discarded. We do not give the rational of the attack here, the important fact being the existence of at least one attack when the countermeasure is weakened.

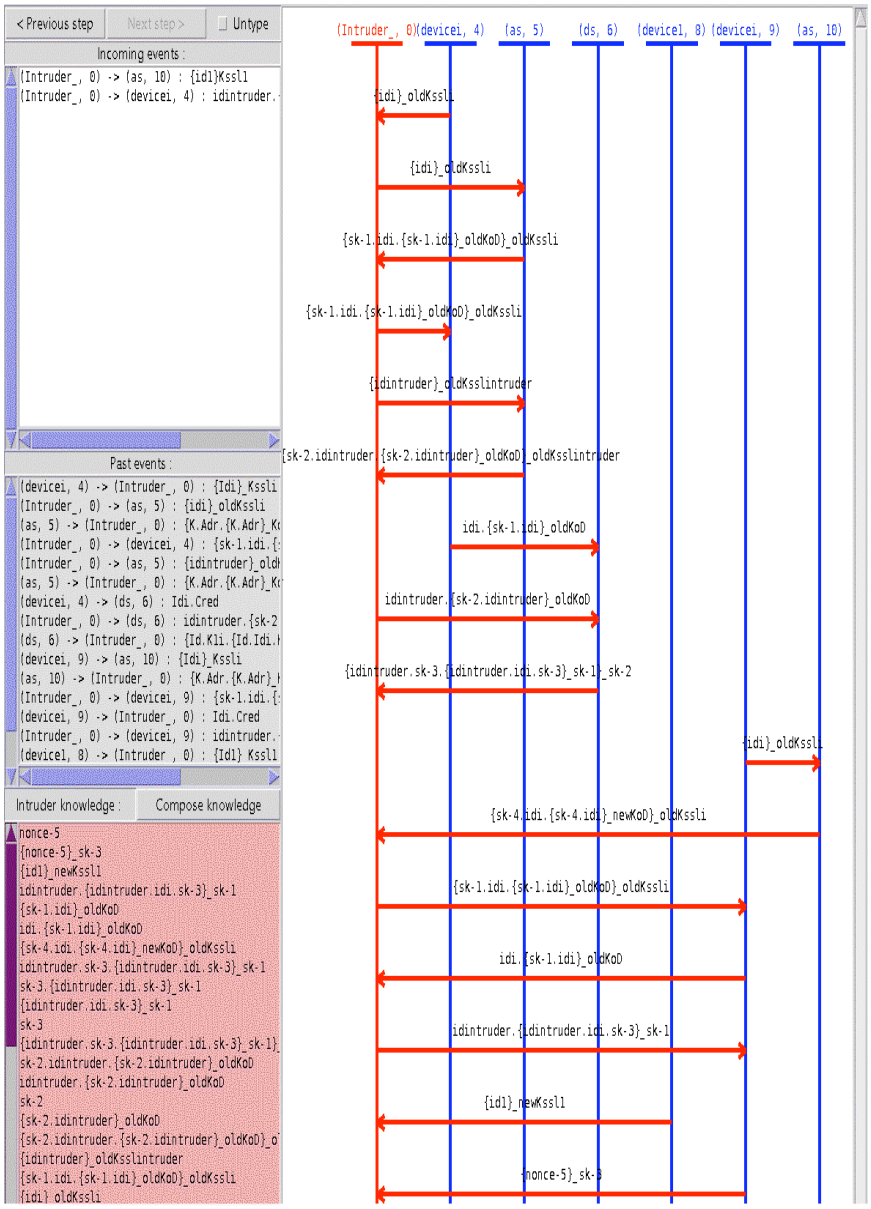

Figure 4: Attack against a slightly downgraded version of LCDP.

Conclusion

Both the academic team and the industrial team observe that this experiment has positive results:

- It brings better confidence in the security of LCDP in the simple cases. Neither OFMC nor ATSE found attacks in the test cases. Because ATSE is proven complete [12] this means that, under the simplifying assumptions of section 4.1, there is no attack on a finite number of sessions in the Dolev-Yao model.
- It provides precise justifications for countermeasures that may otherwise be embedded on a rather prophylactic basis. By just removing some countermeasures, we easily get corresponding attacks.
- It produces precise specification and execution diagrams. Both are useful for the understanding and further implementation of LCDP.

Of course, because of many simplifying assumptions, we are still far away from a complete proof of the protocol. The next step is to progressively relax some of the assumptions, especially the one about the number of directory servers. We are also in the course of simulating more devices and larger sets of trusted devices, at the cost of more intensive computation.

References

1. A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuellar, P. Hankes Drielsma, P.-C. Héam, O. Kouchnarenko, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santos Santiago, M. Turuani, L. Viganò, and L. Vigneron. The AVISPA Tool for the automated validation of internet security protocols and applications, In K. Etessami and S. Rajamani, editors, 17th International Conference on Computer Aided Verification, CAV'2005, volume 3576 of Lecture Notes in Computer Science, pages 281-285, Edinburgh, Scotland, 2005. Springer.
2. Y. Boichut, T. Genet, Y. Glouche, and O. Heen. Using Animation to Improve Formal Specifications of Security Protocols. In Joint conference SAR-SSI, 2007.
3. M. Burrows, M. Abadi, and R. Needham. A logic of authentication. ACM Trans. Comput. Syst., 8(1):18-36, 1990.
4. B. Crispo, B. Popescu, and A. Tanenbaum. Symmetric key authentication services revisited, 2004.
5. D. Dolev and A. Yao. On the security of public key protocols. In Proc. IEEE Transactions on Information Theory, pages 198-208, 1983.
6. W. Diffie and M. Hellman. New directions in cryptography. IEEE Transactions on Information Theory, IT-22(6):644-654, 1976.

7. Y. Glouche and T. Genet. SPAN - a Security Protocol ANimator for AVISPA - User Manual. IRISA / Université de Rennes 1, 2006. 20 pages. http://www.irisa.fr/lande/genet/span/.
8. O. Heen, J.P. Andreaux, and N. Prigent. Improving secure device insertion in home ad-hoc networks. In IFIP SEC, pages 381-394, 2004.
9. G. Lowe. Breaking and Fixing the Needham-Schroeder public-key protocol using FDR. In Tools and Algorithms for the Construction and Analysis of Systems (TACAS), volume 1055, pages 147-166. Springer-Verlag, Berlin Germany, 1996.
10. R. M. Needham and M. D. Schroeder. Using Encryption for Authentication in Large Networks of Computers. Communications of the ACM, 21(12):993-999, 1978.
11. N. Prigent, C. Bidan, J.P. Andreaux, and O. Heen. Secure long term communities in ad hoc networks. In SASN '03: Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks, pages 115-124, New York, NY, USA, 2003. ACM.
12. M. Turuani. Security of Cryptographic Protocols: Decidability and Complexity. PhD thesis, Université of Nancy 1, 2003.
13. M. Weiser. The computer for the 21st century. SIGMOBILE Mob. Comput. Commun. Rev., 3(3):3-11, 199