Towards Verification with no False Attack of Security Protocols in First-order Logic

一階述語論理に於ける誤検知の無いセキュリティプロトコル検証に向けて

Reynald Affeldt[*] Hubert Comon-Lundh
アフェルト レナルド コモン-ルンド ウーベル
Nat. Inst. of Advanced Industrial Science and Technology, Research Center for Information Security
産業技術総合研究所 情報セキュリティ研究センター

Keywords: Security protocols, first-order logic, resolution, rigid variables, false attacks
キーワード：セキュリティプロトコル、一階述語論理、導出、厳正な変数、誤検知

It is possible to model security protocols and their properties using first-order logic. Out of such models, it is even possible to get automatically security proofs using theorem provers for first-order logic. Yet, this approach sometimes fails to produce security proofs despite the correctness of the protocol. This is because of the detection of false attacks that originate from abstractions made for the purpose of modeling. We show that we get rid of some false attacks by using the notion of "rigid variable" to model protocols. Thanks to a simple translation to first-order logic, it turns out that automatic verification with rigid variables can be implemented using standard techniques.

1 Security Protocols in First-order Logic

To verify a security protocol one first needs to model an intruder. The standard model is the so-called Dolev-Yao intruder: it can initiate and divert communications, and send to participants fake messages built out of public or leaked information. In logic, the (knowledge of the) intruder is modeled as a predicate I and its capabilities by logical implications, or rules (see Fig. 1). For example, the rule ($I4$) models the capability of the intruder to decrypt the cipher $[x]_y$ if it knows the key y.

$$\begin{array}{ll}
I(x) \wedge I(y) \rightarrow I(x,y) & (I0) \\
I(x,y) \rightarrow I(x) & (I1) \\
I(x,y) \rightarrow I(y) & (I2) \\
I(x) \wedge I(y) \rightarrow I([x]_y) & (I3) \\
I([x]_y) \wedge I(y) \rightarrow I(x) & (I4)
\end{array}$$

Figure 1: Intruder Model

The model of the security protocol to be analyzed extends the Dolev-Yao intruder with more rules that model the input/output behavior of the participants. In those rules, logical variables $(x, y, z, \ldots)$ can be instantiated by the intruder with whatever (well-formed) message. Let us explain such a model by commenting on the sample protocol of Fig. 2. The goal of this protocol is to produce a secret known only of two participants that already share a secret key K. The initiator A uses this key to start the protocol by sending an encrypted nonce N_0—rule ($P0$). Upon reception, the responder B sends back two nonces N_1, N_2—rule ($P1$). Eventually, A publicly revealed the nonce N_1 (this peculiarity is for illustrative purpose)—rule ($P2$). The claimed security properties is that, even though one nonce is revealed, the pair is not known of the intruder. This claim is modeled by stating its negation (Fig. 3). $\neg I(N_1, N_2)$ is intended to mean that the intuder should not be able to derive both the knowledge of N_1 and N_2 by using the intruder rules and the protocol rules.

$$\begin{array}{ll}
I([A, N_0]_K) & (P0) \\
I([A, x]_K) \rightarrow I([B, x, N_1]_K, [B, x, N_2]_K) & (P1) \\
I([B, N_0, y]_K, [B, N_0, z]_K) \rightarrow I(y) & (P2)
\end{array}$$

Figure 2: Sample protocol

Verification amounts to an exhaustive enumeration of all the logically derivable consequences by a procedure known as *resolution*. Derivation of a contradiction shows a (potential) attack; the absence of contradiction is a proof of security.

$$\neg I(N_1, N_2) \quad (S)$$

Figure 3: Security Goal

[*]E-mail: `reynald.affeldt at aist.go.jp`

Detection of a False Attack The use of first-order logic actually introduces several approximations regarding the order of execution of rules or their replay. This is safe since the absence of attacks implies the absence of attacks for any refined model. However, these approximations can lead to false attacks, i.e., derivation of a contradiction that, after analysis, can be shown to correspond to no real attack.

$$
\begin{array}{llll}
(P0) + (P1) & \Rightarrow & I([B, N_0, N_1]_K, [B, N_0, N_2]_K) & (R0) \\
(R0) + \overline{(P2)} & \Rightarrow & I(N_1) & (R1) \\
(R0) + \overline{(I1)} & \Rightarrow & I([B, N_0, N_1]_K) & (R2) \\
(R0) + (I2) & \Rightarrow & I([B, N_0, N_2]_K) & (R3) \\
(R2) + (R3) + (I0) & \Rightarrow & I([B, N_0, N_2]_K, [B, N_0, N_1]_K) & (R4) \\
(R4) + \overline{(P2)} & \Rightarrow & I(N_2) & (R5) \\
(R1) + \overline{(R5)} + (I0) & \Rightarrow & I(N_1, N_2) & (S') \\
(S) + (S') & \Rightarrow & \textsf{contradiction} &
\end{array}
$$

Figure 4: A False Attack

Fig. 4 shows a false attack for our sample protocol. It comes from the replay of the protocol rule ($P2$). This is not a real attack because in a concrete implementation the protocol participants would have an internal state that they move forward so as to avoid such replays during the same session.

2 Avoid False Attacks using Rigid Variables

The above false attacks can be avoided by introducing *rigid variables*. Intuitively, rigid variables are variables that can participate in only one logical derivation. This turns out to be precisely what we need to model security protocols: the Dolev-Yao intruder still has the freedom to send fake messages to participants, but one it has sent a message, it cannot revoke it. Unfortunately, rigid variables complicate the resolution procedure [1]. Our contribution is a simple way to implement resolution for rigid variables, as we now sketch.

First, we translate the model of the security protocol with rigid variables so as to eliminate them while preserving logical satisfiability. Fig. 5 concretely shows the result of our translation. The idea is to gather together all the (rigid) variables occuring in protocol rules and to prepend a vector of these variables to the predicate I. For protocol rules, this causes a capture: the variables in the prepended vector and the variables in the I predicate are the same. For the intruder rules (not displayed here) and the security goal, there is no capture. In Fig. 6 one can confirm that the derivation of a contradiction is indeed avoided.

$$
\begin{array}{ll}
I(\mathsf{x}, \mathsf{y}, \mathsf{z}, [A, N_0]_K) & (P0) \\
I(\mathsf{x}, \mathsf{y}, \mathsf{z}, [A, x]_K) \to I(\mathsf{x}, \mathsf{y}, \mathsf{z}, [B, x, N_1]_K, [B, x, N_2]_K) & (P1) \\
I(\mathsf{x}, \mathsf{y}, \mathsf{z}, [B, N_0, y]_K, [B, N_0, z]_K) \to I(\mathsf{x}, \mathsf{y}, \mathsf{z}, y) & (P2) \\
\neg I(\mathsf{x}, \mathsf{y}, \mathsf{z}, N_1, N_2) & (S)
\end{array}
$$

Figure 5: Translation of the Sample Protocol of Fig. 2,3

The translation above is only one part of the verification procedure. In fact, nothing a priori guarantees that the resulting fragment of first-order logic is decidable. In [2], we exhibit a refinement of resolution for which one can show completeness and termination.

$$
\begin{array}{llll}
(P0) + (P1) & \Rightarrow & I(\mathsf{N_0}, \mathsf{y}, \mathsf{z}, [B, N_0, N_1]_K, [B, N_0, N_2]_K) & (R0) \\
(R0) + \overline{(P2)} & \Rightarrow & I(\mathsf{N_0}, \mathsf{N_1}, \mathsf{N_2}, N_1) & (R1) \\
(R0) + \overline{(I1)} & \Rightarrow & I(\mathsf{N_0}, \mathsf{y}, \mathsf{z}, [B, N_0, N_1]_K) & (R2) \\
(R0) + (I2) & \Rightarrow & I(\mathsf{N_0}, \mathsf{y}, \mathsf{z}, [B, N_0, N_2]_K) & (R3) \\
(R2) + (R3) + (I0) & \Rightarrow & I(\mathsf{N_0}, \mathsf{y}, \mathsf{z}, [B, N_0, N_2]_K, [B, N_0, N_1]_K) & (R4) \\
(R4) + \overline{(P2)} & \Rightarrow & I(\mathsf{N_0}, \mathsf{N_2}, \mathsf{N_1}, N_2) & (R5) \\
(R1) + \overline{(R5)} + (I0) & \Rightarrow & \textsf{not possible} & (S')
\end{array}
$$

Figure 6: The False Attack of Fig. 4 avoided

Conclusion The verification procedure sketched above allows for a more precise analysis of security protocols since it avoids replays (this is under the restriction that only a bounded number of sessions is considered for verification, see [2] for details). One can further extend it so as to eliminate false attacks due to the absence of ordering of rules. This approach is appealing in practice because the underlying machinery is based on standard resolution for which there already exist numerous results and implementations.

References

[1] S. Delaune, H. Lin, and C. Lynch. Protocol verification via rigid/flexible resolution. In LPAR'07.

[2] R. Affeldt and H. Comon-Lundh. A Note on First-order Logic and Security Protocols. In FCS-ARSPA-WITS'08.