CVS at Work: A Report on New Failures upon Some Cryptographic Protocols

Antonio Durante[1], Riccardo Focardi[2], and Roberto Gorrieri[3]

[1] Dipartimento di Scienze dell'Informazione, Università di Roma "La Sapienza", e-mail: durante@dsi.uniroma1.it

[2] Dipartimento di Informatica, Università Ca' Foscari di Venezia e-mail: focardi@dsi.unive.it

[3] Dipartimento di Scienze dell'Informazione, Università di Bologna e-mail: gorrieri@cs.unibo.it

Abstract. CVS is an automatic tool for the verification of cryptographic protocols, we have presented in [9, 10], that uses a non-interference based analysis technique which has been successfully applied to many case-studies, essentially most of those belonging to the Clark & Jacob's library [4]. In this paper we report some new failures we have found. More precisely, we have been able to detect attacks upon two unflawed (to the best of our knowledge) protocols: Woo & Lam public key one-way authentication protocol and ISO public key two-pass parallel mutual authentication protocol; and new failures upon two flawed protocols: Encrypted Key Exchange and Station to Station protocols.

1 Introduction

Many cryptographic protocols have been proposed in the literature to achieve several goals, e.g. authentication and secrecy; these protocols are supposed to succeed even in the presence of malicious agents, usually called intruders or enemies.

In recent years, many attacks have been discovered over well known protocols (see e.g., [2, 19]). It is often the case that these attacks are based on unexpected execution sequences, where the enemy intercepts messages and combines them in order to obtain new faked messages. Sometimes, many parallel sessions of the protocol are exploited by the enemy in order to carry out the attack, and this complicates even further the task of detecting such sequences, if they exist. For these reasons, there has been an increasing interest in computer-assisted verification tools for the analysis of cryptographic protocols (see, e.g., [23, 20]). Those tools that seem more useful to detect new attacks are essentially model-checkers (see, e.g., the new attacks reported in [23, 9, 6]). On the other hand, theorem provers, such as [21, 24], give better guarantees if a proof is found.

One of the various model checkers is the CVS/CoSeC technology, we have developed in the last three years. It is composed of:

– the compiler CVS [7, 9, 10] from a high-level protocol specification notation (a value passing process algebra with explicit name declarations, called VSP) to a basic process algebra (called SPA), and

– the tool CoSeC [13] which takes as input the SPA protocol specification and checks on that process a non interference-like property, called *Non Deducibility on Compositions* (NDC) we have proposed for this aim [13, 15, 14].

This paper reports some new failures we have found with the help of CVS and CoSeC. More precisely, we have been able to show attacks upon two unflawed (to the best of our knowledge) protocols: Woo & Lam public key one-way authentication protocol [27] and ISO public key two-pass parallel authentication protocol [18]. Moreover, we have also found new failures upon three flawed protocols: Encrypted Key Exchange [3], Station to Station [5] and Woo & Lam symmetric key one-way authentication protocol [27]. (The last failure is not reported here as it already appeared in [9].) Moreover, most of those belonging to the Clark' & Jacob's library [4] has been successfully analyzed, obtaining the known attacks.

Note that the four protocols mentioned above have been already studied extensively. Hence, on the one hand, our results might appear very surprising. On the other hand, the anomalies we have found are rather subtle and it may be not obvious that they are really harmful. For instance, the *entity authentication* attack to the Woo & Lam public key one-way authentication protocol may be considered a real attack depending on the modeling assumptions (packed-switched network vs connection based network [17]). It is thus important, when a particular anomaly is detected, to discuss a possible scenario (i.e., the particular context where the protocol is supposed to run) where the anomaly can show its potential danger (see [8] for a case study fully analyzed in this way). We will discuss this point in the next sections, when presenting the various protocol failures we have found.

2 Non interference for the analysis of security protocols

CVS/CoSeC has been implemented to support the Non-Interference based analysis of security protocols we have proposed in [13, 10, 14].

Non-Interference [16] has been originally proposed for modeling multilevel security and in particular to solve the problem of *covert channels*, i.e., indirect ways of transmitting information from a security level to a lower one. In recent work [12, 13], two of the authors have studied the many non interference-like properties proposed so far, by defining all of them uniformly in a common process algebraic setting, producing the first taxonomy of these security properties reported in the literature.

Among the many NI-like properties, we advocated a property called *Non Deducibility on Composition* (NDC for short) that, informally, can be expressed as follows: Given two groups of users H (high) and L (low), there is no information flow from H to L iff there is no way for H to modify the behaviour of L.

Analogously, we may think that L is the set of the honest participants to a protocol and H is the external, possibly malicious, environment, i.e. the set of possible intruders (or enemies). Following the analogy, no information flow from high to low means that the intruders have no way to change the low behaviour of the protocol. Of course, this statement may hold depending on what we mean by *low behaviour* of the protocol. We should assume that an intruder may have complete control of the network, and so it is reasonable to assume that all the activities on the public channels is considered a high behaviour. The low behaviour is therefore related to extra observable actions included into the specification of the protocol to specify the security properties of interest. For instance, to model some form of authentication as in [19], it is enough to include into the specification of the protocol special start/commit actions for all the honest participants; the property holds if the enemy is not able to destroy the precise correspondences among these start/commit actions.

The interesting point is that many security properties in the literature can be characterized as specialization of the NDC property, with a suitable insertion of extra observable low actions: secrecy , some forms of authentication, message authenticity, some forms of non repudiation and fair-exchange [15, 14]. More interestingly, we have proved that, once the extra observable events have been fixed, NDC implies all the other security properties characterized by the extra observable actions. This means that every attack which is based on certain observable events is for sure detected by our NDC check. Thus, if we observe all the events corresponding to a set of typical interesting properties, we will in a sense verify all of them in just one NDC check [10]. Moreover, we can include a rather generous set of extra observable actions into the protocol specification (containing at least the extra actions of the properties of interest) that may reveal some suspicious behaviour that we should analyze, as it may reveal a potential anomaly, failure or really dangerous attack. This is indeed the point that mostly differentiates the NDC-based approach from the usual ones: we do not fix in advance the specific security property that we want to check. As a matter of fact, it is quite typical with this approach to find "unexpected" protocol failures that were not considered as important in the initial modeling of the protocol. In our opinion, the NDC-based approach offers more opportunities of finding new attacks with respect to classic approaches where, often, the analyzer sets the tool (and the security property) in search for an attack that he has already guessed. This paper is intended to substantiate this claim.

3 The methodology

Analyzing a security protocol applying NDC consists of the following steps (see Figure 1).

1. We specify the protocol honest roles in a value-passing version of the Security Process Algebra (SPA [13]), called VSP [7, 9, 10], that includes name declarations (e.g., process names), crypto-functions (e.g., symmetric and asymmetric encryption, hash functions), and other features that are useful to provide a fully-fledged specification of the protocol. In this phase, the actual instance of the protocol is to be specified defining the analyzed session (e.g., who is playing which role, possibly in how many instances), including some parameters (e.g., initial knowledge) that initialize the structure of the enemy which is then generated automatically by the compiler CVS.

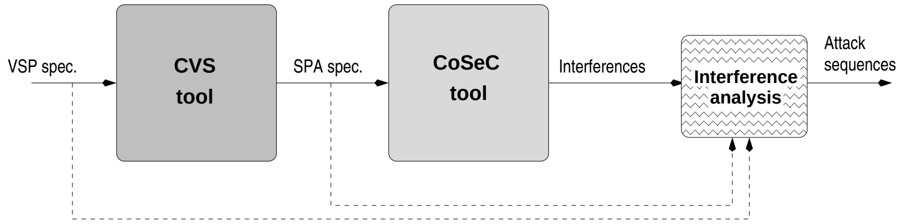

Fig. 1. The CVS tool and the NDC-based protocol analysis.

2. The second step is the definition of the extra observable actions, i.e. those actions that are relevant for specifying and checking the security properties of interest and that can reveal if the enemy has interfered with the correct execution of the protocol. As already stated, a choice of a quite large set of observable events will reveal simultaneously a number of different, potentially dangerous, attack sequences. These actions are suitably inserted into the VSP specification of the parties.
3. The tool CVS [7] is used to compile a VSP specification into SPA code, where the enemy specification is provided in full detail. SPA is a slight modification of Milner's CCS [22] where actions have been partitioned into two security levels, in order to analyze Non-Interference properties. It basically allows for the specification of protocols as the parallel composition of the various parties with the enemy. Each party is defined by the sequence of messages it can send or receive. The execution of the protocol is then obtained by considering all the possible interleavings of the parties which may also synchronize in order to perform a message exchange.
4. The tool CoSeC [13] takes as input the SPA specification representing the protocol that we want to analyze and checks the NDC property on the finite automaton representing the semantics of that process. If the protocol verifies NDC successfully, then we are sure that the enemy cannot interfere with the correct execution of the protocol in the analyzed session; otherwise we observe some *interferences*, i.e., execution traces that were not possible without enemy's intervention.
5. The interference traces are then (manually) analyzed. These interferences may reveal some actual attacks upon the protocol.

The next sections are devoted to the presentation of some new attacks we have found with the CVS/CoSeC technology which implements our verification analysis technique based on NDC. We will not give the details of how the analysis can be performed. The interested reader can refer to [9, 10, 8] for details on how to use CVS/CoSeC. Indeed, the main aim of this paper is to show that our approach can be useful, by giving some examples of new failures that we have found in some well-known (and deeply analyzed) protocols.

In each analysis we do, we set in the VSP specification as observable actions the start and the correct commit of the protocol.

4 ISO Public Key Two-Pass Parallel Mutual Authentication Protocol

This is one of the protocols proposed in [18] by the ISO/OSI committee. The goal of this protocol is to achieve mutual authentication (in a parallel way). Its formal description is reported below, where the notation $A \to B : msg$ represents the sending of message msg from Alice (A) to Bob (B):

Message 1	$A \to B :$	A, N_A	`i_start_A_B`
Message 1′	$B \to A :$	B, N_B	`i_start_B_A`
Message 2	$B \to A :$	$N_B, N_A, A, SK_B(N_B, N_A, A)$	`r_commit_B_A`
Message 2′	$A \to B :$	$N_A, N_B, B, SK_A(N_A, N_B, B)$	`r_commit_A_B`

The protocol consists of four messages, pairwise symmetric. It is based on public key cryptography: PK_U denotes the public key of a generic user U while SK_U is his secret key. Moreover, N_U represents a *nonce* (i.e., a number used only once) generated by user U. Both A and B prove their identity by signing the respective challenges N_A and N_B with their secret keys.

Each message is decorated by an extra observable action that we have inserted into the specification for analysis purposes. For instance what we observe by executing message 1 is the action `i_start_A_B` meaning that A is initiating a session with B. Similarly, in message 2, `r_commit_B_A` signals that B ended the exchange with A as a responder.

For this protocol we have found the particular failure sequence `i_start_A_A`, `r_commit_A_A`, that can be justified by the following attack (we denote the enemy which is impersonating the entity U with $E(U)$ in the position of the sender, and the enemy that intercepts the message directed to U with $E(U)$ in the position of the receiver):

Message 1	$A \to E(A) :$	A, N_A	`i_start_A_A`
Message 1′	$E(A) \to A :$	A, N_A	
Message 2′	$A \to E(A) :$	$N_A, N_A, A, SK_A(N_A, N_A, A)$	
Message 2	$E(A) \to A :$	$N_A, N_A, A, SK_A(N_A, N_A, A)$	`r_commit_A_A`

This corresponds to a parallel session attack and can happen when an entity A needs to authenticate itself on a remote computer. The enemy in the middle intercepts all the communications re-sending the messages to the initiator A. At the end of the run the entity A is convinced to communicate with herself. This self-authentication attack might be dangerous when the protocol is used by A, e.g., to transfer files between accounts on different computers both owned by A. This failure is similar to the one reported in [6] for the ISO public key two-pass mutual authentication protocol, where the protocol we analyze here is listed as unflawed.

Note that a simple way for A to avoid this potential attack is simply to verify, in every protocol execution, that the two (signed) nonces in Message 2 are different.

5 Woo-Lam Public Key One-Way Authentication protocol

In [27], Woo and Lam proposed two protocols for achieving one-way authentication. These protocols differ in the cryptosystem they use. Here, we examine the one based on public-key cryptography. The protocol is composed of five steps:

Message 1	$A \rightarrow B :$	A	`i_start_A_B`
Message 2	$B \rightarrow A :$	N_B	`r_start_B_A`
Message 3	$A \rightarrow B :$	$SK_A(N_B)$	
Message 4	$B \rightarrow S :$	A	
Message 5	$S \rightarrow B :$	$A, SK_S(A, PK_A)$	`r_commit_B_A`

Message 1 represents the authentication request form A to B. In this message A sends her name as plaintext to B. In *Message 2*, B replies to A by sending a freshly generated nonce N_B. Once A has received the nonce N_B, she encrypts (signs) it with her secret key SK_A and sends the result to B. B cannot decrypt this message because he does not know the public key of A. Thus, in *Message 4*, B requests to S a certificate containing the public key of A. The Server retrieves a certificate $SK_S(A, PK_A)$ from the key database and sends it to B. B decrypts it using the public key of S, recovers A, PK_A and uses PK_A to decrypt the *Message 3*. Then, he verifies if N_B is equal to N_B of *Message 2*.

For this protocol we have found the following failure sequence:

Message 1.a	$A \rightarrow E$	: A	`i_start_A_E`
Message 1.b	$E(A) \rightarrow B$	: A	`r_start_B_A`
Message 2.b	$B \rightarrow E(A)$	: N_B	
Message 2.a	$E \rightarrow A$	: N_B	
Message 3.a	$A \rightarrow E$	: $SK_A(N_B)$	
Message 3.b	$E(A) \rightarrow B$	: $SK_A(N_B)$	
Message 4.b	$B \rightarrow S$	: A	
Message 5.b	$S \rightarrow B$	: $A, SK_S(A, PK_A)$	`r_commit_B_A`

The failure is very similar to the one we have found for the version based on symmetric key cryptography [9, 10]. The actions performed by the enemy leads the responder B to achieve authentication with A that has never sent any authentication request to him. It is a typical man-in-the-middle attack: indeed, A starts the protocol with E and E is able to exploit this session (denoted by a) with A in order to impersonate A with B in a parallel session (denoted by b).

The fact that this failure is dangerous or not strictly depends on the model assumptions. For a detailed discussion see, e.g., [17]. However, such a failure can be avoided by simply including in the signature of Message 3, the identifier of the intended recipient, i.e., Message 3 becomes $SK_A(N_B, \mathbf{B})$. With this modification, E will not be able to forward Message 3.a (now $SK_A(N_B, E)$) to B as it contains the identifier of the intended recipient E. As a matter of fact, B would immediately recognize that such a message is just the replay of a message sent to E.

6 The EKE protocol

The protocol considered is the Encrypted Key Exchange - EKE protocol proposed in [3]. It provides both security and authentication on computer networks, using both symmetric and public key cryptography. The EKE protocol is proposed for the negotiation of a new session key between two users that share a common password K_P.

6.1 First analysis: EKE based on Diffie-Hellman

We consider the version of EKE that is based on the Diffie-Hellman protocol. It is thus useful to briefly summarize how the Diffie-Hellman protocol works: Before starting the protocol, Alice and Bob agree on a large prime n and on a number g, such that g is a primitive element mod n. The prime n is chosen such that it is infeasible to calculate the discrete logarithm mod n. These two integers do not have to be secret; usually they are common among a group of users [25]. The Diffie-Hellman protocol is used for establishing a new session key between Alice and Bob and is composed of the following two messages:

$$\begin{array}{lll} \textit{Message } 1 & A \to B: & X = g^x \ mod \ n \\ \textit{Message } 2 & B \to A: & Y = g^y \ mod \ n \end{array}$$

Alice and Bob respectively choose two random large integers x and y, and send one another $X = g^x \ mod \ n$ and $Y = g^y \ mod \ n$. Alice and Bob can then compute the same session key as $Y^x \ mod \ n = g^{xy} \ mod \ n = X^y \ mod \ n$. No one listening on the channel can compute such a value as it does not know x and y and can neither calculate them through logarithm (see, e.g., [25] for more details). At the end of the protocol, each pair of users shares a unique secret key, and no prior communication between the two users is required.

We will denote with $DH_{X,y}$ the Diffie-Hellman function, which is used to compute the key shared between the two principals, i.e., $DH_{X,y}$ will represent the exponentiation $X^y \ mod \ n$. The version of EKE based on Diffie-Hellman protocol follows:

$$\begin{array}{llll} \textit{Message } 1 & A \to B: & A, X & \texttt{i\_start\_A\_B} \\ \textit{Message } 2 & B \to A: & K_P(Y), DH_{X,y}(N_B) & \texttt{r\_start\_B\_A} \\ \textit{Message } 3 & A \to B: & DH_{Y,x}(N_A, N_B) & \\ \textit{Message } 4 & B \to A: & DH_{X,y}(N_A) & \texttt{i\_commit\_A\_B}, \texttt{r\_commit\_B\_A} \end{array}$$

In the first two messages Alice and Bob negotiate the new session key. Indeed, Alice sends her identifier together with the value $X = g^x \ mod \ n$, where x is freshly generated. Then, Bob replies with the value $Y = g^y \ mod \ n$ encrypted with the shared key K_P, together with the encryption with the Diffie-Hellman session key $DH_{X,y}$ of a nonce N_B. In the last two messages Alice and Bob use the obtained session key to authenticate the run through a typical challenge-response mechanism. The encryption of Y with K_P should guarantee that Y has been generated by B since only A and B know such a shared key.

In the analyzed session Alice plays at the same time both the initiator and the responder role. Alice begins two runs with Bob, the first as an initiator and the second as a responder and she completes both. We describe in detail the actions performed by Alice due to the enemy presence:

Message 1.a	$A \rightarrow E(B)$:	A, X	`i_start_A_B`
Message 1.b	$E(B) \rightarrow A$:	B, X	`r_start_A_B`
Message 2.b	$A \rightarrow E(B)$:	$K_P(Y), DH_{X,y}(N_A)$	
Message 2.a	$E(A) \rightarrow A$:	$K_P(Y), DH_{X,y}(N_A)$	
Message 3.a	$A \rightarrow E(B)$:	$DH_{Y,x}(N_A, N_A)$	
Message 3.b	$E(B) \rightarrow A$:	$DH_{Y,x}(N_A, N_A)$	
Message 4.b	$A \rightarrow E(B)$:	$DH_{X,y}(N_A)$	`r_commit_A_B`
Message 4.a	$E(B) \rightarrow A$:	$DH_{X,y}(N_A)$	`i_commit_A_B`

Alice starts the run a as an initiator with Bob, message 1 is intercepted by the enemy that starts a run b with her pretending to be Bob. The enemy in the middle acts as Bob, intercepting and forwarding the message from Alice to Alice. Alice completes both the runs, thinking in the first that Bob acts as a responder, and in the second that Bob acts as an initiator. However, Bob has never started any run with Alice and could even not be present on the net.

This kind of weakness is not serious as the negotiated key is not compromised, but it leads to a denial of service attack, as Alice will be convinced to share the (useless) key $DH_{X,y}$ with Bob. However, this execution sequence shows that the protocol should never be used for entity authentication only (no key-exchange). In such a case the attack would be valid as E would be able to completely impersonate B. A way for repairing the protocol is to encrypt with K_P also the identifier of the intended recipient (as similarly done for the previous attack), i.e., message 2 becomes $K_P(Y, \mathbf{A}), DH_{X,y}(N_B)$. This allows A to discover that message 2.a is indeed a replay.

6.2 Second Analysis: EKE based on RSA

We have analyzed another version of the EKE protocol where the public key cryptography is implemented with the RSA algorithm. Even in this case the two users share a common password K_P and use the challenge response mechanism to perform the authentication goal.

Message 1 $A \rightarrow B$: $A, K_P(PK_A)$
Message 2 $B \rightarrow A$: $K_P(PK_A(K))$
Message 3 $A \rightarrow B$: $K(N_A)$
Message 4 $B \rightarrow A$: $K(N_A, N_B)$
Message 5 $A \rightarrow B$: $K(N_B)$

With the first two messages, the parties negotiate the new session key K generated by the responder B. The key exchange is performed using the new public key PK_A that A generates. The last three messages achieve the authentication task. We consider the protocol in a single session, where each agent plays just one role at a time. The failure sequence due to the enemy presence is the following:

$$
\begin{array}{llll}
\textit{Message } 1.a & A \rightarrow E: & A, K_{Pe}(PK_A) & \texttt{i\_start\_A\_E} \\
\textit{Message } 1.b & E \rightarrow B: & E, K_{Pe}(PK_E) & \texttt{r\_start\_B\_E} \\
\textit{Message } 2.b & B \rightarrow E: & K_{Pe}(PK_E(K)) & \\
\textit{Message } 2.a & E \rightarrow A: & K_{Pe}(PK_A(K)) & \\
\textit{Message } 3.a & A \rightarrow E: & K(N_A) & \\
\textit{Message } 3.b & E \rightarrow B: & K(N_A) & \\
\textit{Message } 4.b & B \rightarrow E: & K(N_A, N_B) & \\
\textit{Message } 4.a & E \rightarrow A: & K(N_A, N_B) & \\
\textit{Message } 5.a & A \rightarrow E: & K(N_B) & \texttt{i\_commit\_A\_E\_K} \\
\textit{Message } 5.b & E \rightarrow B: & K(N_B) & \texttt{r\_commit\_B\_E\_K}
\end{array}
$$

The agent E simply acts as a responder running the protocol with A in the run a, and as an initiator running the protocol with B in the run b. E re-negotiates in the run a the key K negotiated with B in the run b. None of the principals thinks to be connected to an unexpected principal. The anomaly consists in the fact that A and B share K, at the same time, with E. Indeed, a key-exchange protocol should guarantee the very basic security property that different pairs of principals have different shared keys. The effect of this anomaly is that the attacker E creates an implicit communication channel between A and B passing through its location. This kind of failure is common to many cryptographic protocols designed to create a secure channel between two agents A and B without the intervention of a trusted third party.

Of course, the enemy is interested in sharing a common password with both A and B only if this gives some advantage to him. An interesting case could be one where E gets credit for something which is indeed done by A. For example, suppose that B represents a "teacher server" where students can log-in and have on-line interactive examinations. Suppose also that E is able to create a clone of the login interface of B on another site (E itself), and to convince some students that such site is the correct one. Now consider a student A that is logging on E thinking it is the teacher server. E is now able, through the sequence above, to establish a key K which is shared between A, B and E. This allows E to leave A and E freely interact (with no delays due to man-in-the-middle encryptions and decryptions). A will notice nothing strange since she indeed wanted to talk to the teacher. On the other side, B will give credit to E of all the (hopefully good) answers of A to the various questions. Moreover E is able to passively track every communication between the two in order to know what is happening. For a discussion about credit and authentication see, e.g., [1].

7 The Station to Station protocol

The Station to Station protocol was proposed in [5] with the aim of establishing a secure communication channel between each pair of users of a community using the Diffie-Hellman protocol.

The Diffie-Hellman key exchange is known to be vulnerable to a man in the middle attack. One way to prevent this problem is to have Alice and Bob signing their message each other, as proposed in [5]. This protocol assumes that each pair of users have a certificate containing the public key of the other users of the community. The modified Diffie-Hellman protocol follows:

$$\begin{array}{lll}
\textit{Message } 1 & A \rightarrow B: & X \\
\textit{Message } 2 & B \rightarrow A: & Y, DH_{X,y}(SK_B(X,Y)) \\
\textit{Message } 3 & A \rightarrow B: & DH_{Y,x}(SK_A(X,Y))
\end{array}$$

Alice and Bob agree on the values X and Y, and they sign such values in messages 2 and 3 with their private key to avoid the man in the middle attack.

A strange execution sequence for this protocol has been detected by CVS in a multiple session analysis, when Alice and Bob play two times the Initiator and the Responder role, respectively. In detail, the sequence of actions performed by Alice and Bob in the presence of the enemy is the following:

$1.a$	$A \rightarrow E(B):$	$X1$	`i_start_A_B_a`
$1.b$	$A \rightarrow E(B):$	$X2$	`i_start_A_B_b`
$1.a'$	$E(A) \rightarrow B \quad :$	$X2$	`r_start_B_A_a`
$1.b'$	$E(A) \rightarrow B \quad :$	$X1$	`r_start_B_A_b`
$2.a$	$B \rightarrow E(A):$	$Y1, DH_{X2,y1}(SK_B(X2,Y1))$	
$2.b$	$B \rightarrow E(A):$	$Y2, DH_{X1,y2}(SK_B(X1,Y2))$	
$2.a'$	$E(B) \rightarrow A \quad :$	$Y2, DH_{X1,y2}(SK_B(X1,Y2))$	
$2.b'$	$E(B) \rightarrow A \quad :$	$Y1, DH_{X2,y1}(SK_B(X2,Y1))$	
$3.a$	$A \rightarrow E(B):$	$DH_{Y2,x1}(SK_A(X1,Y2))$	`i_commit_A_B_X1_Y2_a`
$3.b$	$A \rightarrow E(B):$	$DH_{Y1,x2}(SK_A(X2,Y1))$	`i_commit_A_B_X2_Y1_b`
$3.a'$	$E(A) \rightarrow B \quad :$	$DH_{Y1,x2}(SK_A(X2,Y1))$	`r_commit_B_A_X2_Y1_a`
$3.b'$	$E(A) \rightarrow B \quad :$	$DH_{Y2,x1}(SK_A(X1,Y2))$	`r_commit_B_A_X1_Y2_b`

Alice, acting as an initiator, begins two runs a, b with Bob (acting as a responder), and she terminates both with him. The enemy acts in the middle intercepting and exchanging the order of the messages between the two runs. At the end of the runs Alice and Bob agree on two different values of X and Y. In the run a Alice agrees on $X1, Y2$ (message 3.a), and Bob agrees on $X2, Y1$ (message 3.a') and vice-versa for the run b. The agents are mutually authenticated but they disagree upon the exchanged values, so upon the new negotiated session key. This kind of weakness does not compromise the secrecy of the negotiated key, but leads to a denial service.

As a matter of fact, in a real implementation of the protocol a pair of users will presumably establish a (non-secure) connection before starting the protocol, e.g., a TCP connection. Suppose that this is done in a preliminary message where A and B decide to start the protocol. In the case of the execution sequence above, we would have two (TCP) connections between the two pairs of users that are willing to communicate. We call (A_1, B_1) and (A_2, B_2) such two pairs. Basically, the enemy is now able to force the "crossed" pairs (A_1, B_2) and (A_2, B_1) sharing two different secret keys, but A_1 will never be able to communicate with B_2 since they are on two different (TCP) connections. A_1 will indeed try to use the secret key with B_1 with, of course, no success.

This situation is perhaps a typical example in which the refinement of a protocol specifications leads to undesired executions, i.e., denial of service. It may happen in a number of existing protocols, and may be avoided only by including in the messages some information about the low level connections, e.g., IP address and port. An attack which is somehow similar to the denial of service above has been found in Otway-Rees protocol in [11], although in such an attack there is only one session where A and B commit on two different sessions keys. Such an attack is thus effective also at an abstract specification level (without considering low level connections).

8 Conclusion

We think that these case studies support our claim that CVS, and the NDC-based analysis technique for cryptographic protocols it implements, is potentially useful in order to detect new attacks. It is therefore desirable to start investigating more complex protocols, used in real-life applications (such as SET [26]) to see if our technique can uncover possible unknown failures. We are currently working to improve our tools in order to handle such larger protocols. The main issue is the state-space explosion problem for which we are implementing some techniques for reducing the state space, on the one hand, and for managing more efficiently the state memorization, on the other hand.

We are also implementing the automatic filtering of the interference traces which correspond to attacks over some already known security properties like, e.g., those reported in [15, 14]. This will allow us to automatically extract the interferences that correspond to well known (interesting) typologies of attacks. It will then be possible to focus the analysis on the remaining interferences in order to observe if new strange failures are present in the analyzed protocol.

References

[1] M. Abadi. "Two Facets of Authentication". In *Proceedings of 11th IEEE Computer Security Foundations Workshop*, pages 25–32. IEEE press, June 1998.

[2] M. Abadi and R. Needham. "Prudent Engineering Practice for Cryptographic Protocols". *IEEE Transactions on Software Engineering*, 22(1), January 1996.

[3] S. M. Bellovin and M. Merritt. "Encrypted Key Exchange: Password-Based Protocols Secure Against Dictionary Attacks". In *Proceedings of 1992 IEEE Computer Society Conference on Research in Security and Privacy*, pages 72–84, 1992.

[4] John Clark and Jeremy Jacob. "A Survay of Authentication Protocols Literature: version 1.0". Available at http://www.cs.york.ac.uk/ jac/papers/drareview.ps.gz, November 1997.

[5] W. Diffie, P. C. van Oorschot, and M. J. Wiener. "Authentication and Authenticated Key Exchanges". *Design, Codes and Cryptography*, 2:107–125, 1992.

[6] B. Donovan, P. Norris, and G. Lowe. "Analizing a Library of Security Protocols Using Casper and FDR". In *Proceedings of the FLoC Workshop Formal Methods and Security Protocols*, Trento, July 1999.

[7] A. Durante. "Uno strumento automatico per la verifica di protocolli crittografici". Master's thesis, Università degli Studi di Bologna (sede Cesena), luglio 1998.

[8] A. Durante. "Analysis of cryptographic protocols: the environment and the application". In *Proceedings of* 15^{th} *IFIP SEC 2000*, August 2000.

[9] A. Durante, R. Focardi, and R. Gorrieri. "CVS: a tools for the analysis of cryptographic protocols". In *Proceedings of* 12^{th} *IEEE Computer Security Foundation Workshop*, pages 203–212, July 1999.
[10] A. Durante, R. Focardi, and R. Gorrieri. "A Compiler for Analysing Cryptographic Protocols Using Non-Interference". To appear in *ACM Transaction on Software Engeneering and Methodology* (TOSEM), 2000.
[11] F. J. T. Fabrega, J. C. Herzog, and J. D. Guttman. "Honest Ideals on Strand Spaces". In *Proceedings of* 11^{th} *IEEE Computer Security Foundation Workshop*, pages 66–77, June 1998.
[12] R. Focardi and R. Gorrieri. "A Classification of Security Properties for Process Algebras". *Journal of Computer Security*, 3(1):5–33, 1995.
[13] R. Focardi and R. Gorrieri. "The Compositional Security Checker: A Tool for the Verification of Information Flow Security Properties". *IEEE Transactions on Software Engineering*, 23(9):550–571, September 1997.
[14] R. Focardi, R. Gorrieri, and F. Martinelli. "Non Interference for the Analysis of Cryptographic Protocols". In *Proceedings of ICALP'00*, pages 744–755. LNCS 1853, July 2000.
[15] R. Focardi and F. Martinelli. "A Uniform Approach for the Definition of Security Properties". In *Proceedings of World Congress on Formal Methods in the Development of Computing Systems (FM'99), LNCS 1708*, pages 794–813, Toulouse (France), September 1999. Springer-Verlag LNCS.
[16] J. A. Goguen and J. Meseguer. "Security Policy and Security Models". In *Proceedings of the 1982 Symposium on Security and Privacy*, pages 11–20. IEEE Computer Society Press, April 1982.
[17] D. Gollman. "On the Verification of Cryptographic Protocols - A Tale of Two Committees". In *Workshop on secure architectures and information flow*, volume 32 of *ENTCS*, 2000.
[18] International Organization for Standardization. Information technology - Security techniques - Entity authentication mechanism; Part 3: Entity authentication using a public key algorithm. ISO/IEC 9798-3, 1995.
[19] G. Lowe. "Breaking and Fixing the Needham-Schroeder Public-key Protocol using FDR". In *Proceedings of Second International Workshop on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'96)*, pages 146–166, Passau (Germany), March 1996. Springer-Verlag, LNCS 1055.
[20] G. Lowe. "CASPER: A Compiler for the Analysis of the security protocols". In *Proceedings Tenth IEEE Computer Security Foundation Workshop, (CSFW'97)*, pages 18–30, Rockport Massachusetts (USA), June 1997. IEEE Press.
[21] C. Meadows. "The NRL Protocol Analyzer: An Overview". *Journal of Logic Programming*, 26(2):113–131, 1996.
[22] R. Milner. *Communication and Concurrency.* Prentice-Hall, 1989.
[23] J. C. Mitchell, M. Mitchell, and U. Stern. "Automated Analysis of Cryptographic Protocols Using Murϕ". In *Proceedings of the 1997 IEEE Symposium on Research in Security and Privacy*, pages 141–153. IEEE Computer Society Press, 1997.
[24] Lawrence C. Paulson. "The Inductive Approach to Verifying Cryptographic Protocols". In *Proceedings Tenth IEEE Computer Security Foundation Workshop, (CSFW'97)*, Rockport Massachusetts (USA), June 1997. IEEE Press.
[25] B. Schneier. *Applied Cryptography.* John Wiley & Sons, Inc., 1996. second edition.
[26] SET Secure Electronic Transaction LLC. The SET Standard Specification. Available at the URL, http://www.setco.org/set_specifications.html, May 1997.
[27] T.Y.C. Woo and S.S. Lam. "Authentication for distributed systems". *Computer*, 25:10–10, March 1992.