Classification of Security Properties

(part 2: network security)

Riccardo Focardi

Dipartimento di Informatica, Università Ca' Foscari di Venezia,

Second International School On Foundations

Of Security Analysis And Design

— FOSAD 2001 —

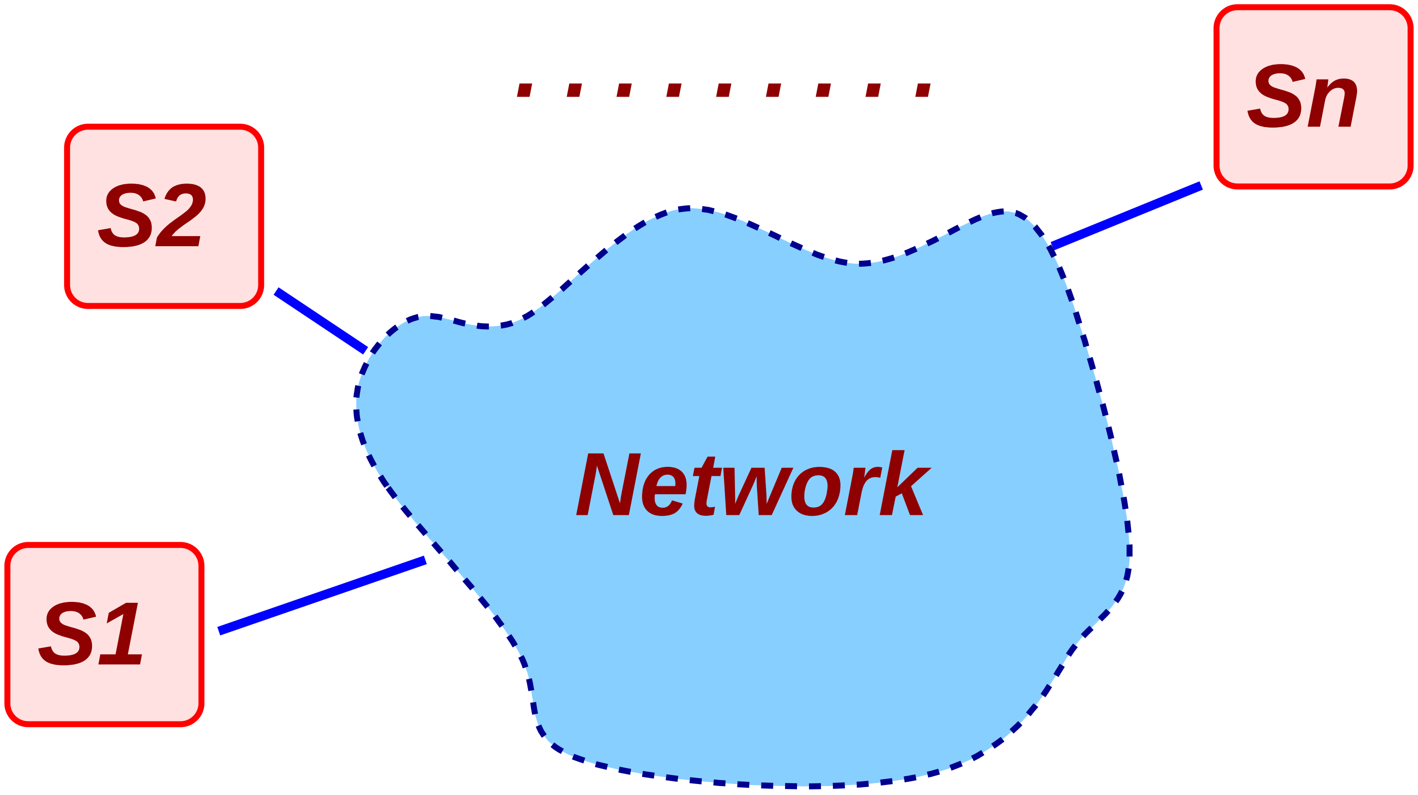
Bertinoro, September 2001

Introduction

First part of the course: You have seen the formalization of Information Flow Security inside systems!

- Try to detect any (direct and indirect) flows from High level to Low one.
- Main idea: Information Flow $\equiv$ Interference
- Question: Can these definitions and ideas be applied also to network security?

System vs Network security



- Inside systems some control is possible, what about networks?

System vs Network security

System	Network
Identification mechanism (e.g., login-passwd, smartcard, . . .)	Usually easy to have complete access to a part of the network (e.g., plug on ethernet and monitor every single packet)
Access control policies (e.g. Unix file system, multilevel security, . . .)	No access control is possible
Security through access control	Security through cryptography

Worst case assumption: Everything bad could happen to the data you send across the network!

Different levels of attack

Two main different kinds of attacks:

- Cryptanalysis: exploit some possible weakness of cryptographic algorithms; it is the “counterpart” of cryptography.
- Crypto protocol analysis: exploit some possible weakness of a cryptography-based protocol. No need of breaking cryptography to perform the attack.

We can also have mixed attacks:

- exploit both the aspect above, i.e., protocols may use cryptography in a way that facilitate cryptanalysis.

Cryptanalysis (simple example)

(already seen in Cryptography course)

Substitution cipher: each character of the plain text is replaced with a different one.

- the key K is a permutation of the alphabet e.g.:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
QBFTRPSONAGUMCXVHKZWJEDLIY
```

- Big key space: there are 26! different keys.
- But: if you know the "language" of the plaintext it is straightforward to attack! How?

Cryptanalysis of substitution cipher

- Do a frequency analysis of character occurrences in the ciphertext and compare it with a frequency table of the language;
- Use also frequency analysis of pairs of characters and so on;
- Use a dictionary to stop when you find the right key.

Crypto protocol attack (simple example)

(Trivial) money transfer protocol:

- Suppose Alice have a cryptographic mechanism for signing a message in a way that nobody can forge a signed message and everybody can check the validity of Alice's signature.
- Consider the following protocol:

$$A \longrightarrow Bank : \{\text{“transfer 1000\$ to E”}\}_{Sign_A}$$

- There is a very simple (non cryptographic) attack. Can you find it out?

The simple attack

E eavesdrops the message and replays it later:

$$\begin{array}{llll} A \longrightarrow & E & \longrightarrow Bank : & \{\text{“transfer 1000\$ to E”}\}_{Sign_A} \\ & E(A) & \longrightarrow Bank : & \{\text{“transfer 1000\$ to E”}\}_{Sign_A} \end{array}$$

$\Rightarrow$ No attack on the signature!

- Also in this case, there are more sophisticated attacks that are not so easy to detect.

We focus on crypto protocol attacks

We consider cryptography as a **secure underlying primitive**, e.g.:

- Symmetric encryption: Given a message M,

  $$\{M\}_K$$

  represents the encryption of M with key K.

  $\Rightarrow$ Can be generated and decrypted only knowing K

Secure Cryptographic Primitives (ctd.)

- Asymmetric encryption (public key): Alice has a secret key SK_A and a corresponding public key PK_A. Then

  $$\{M\}_{PK_A}$$

  $\Rightarrow$ Can be generated by everyone and can be decrypted only by Alice (through secret key SK_A)

- Electronic Signature (through secret key):

  $$\{M\}_{SK_A}$$

  $\Rightarrow$ Can be generated only by Alice and can be verified by anyone (through public key PK_A)

Which are the consequences of this assumption?

$\Rightarrow$ We detect attacks that can be done without even breaking cryptography!

- Soundness: If an attack is revealed, it will be an attack also in the implementation of the protocol.

  Example: the trivial money transfer example.

- No completeness: Even if no attacks are found, new attacks could arise on the cryptographic side.

  Example: If the signature is breakable, whatever money transfer protocol we consider, an attack will always be possible.

$\Rightarrow$ Improve completeness by integrating protocol models with cryptographic models (see, e.g., [Mitchell et al. 1999]).

Classification

GNDC (Generalized NDC): a generalization of NDC proposed for capturing a number of (system and network) security properties with the following aims:

- *formal comparison*: as the definitions are now given in a uniform style, it is easier to compare the relative merits;

  Example: many variants of authentication like Message Authenticity, Entity Authentication, . . .

- *one check for all*: As all the properties are defined in the same Non-Interference (NDC) style, by observing enough "actions" of the protocol we can check all of them in just one NDC analysis.

  (we will see examples of this)

Network security properties

- we now informally present some typical (network) security properties through some simple examples.
- All these properties have been defined for different aims and have been formalized using various models.

$\Rightarrow$ Through the example, we will identify a *general common idea* behind all of these properties (leading to GNDC!)

A simple key-exchange protocol

- *KDC* (Key Distribution Center): a process on a remote host which is devoted to the distribution of the public keys.
- A asks *KDC* for B's public key
- A sends to B a session key encoded with the public key of B.
- Such session key will be used for every further communication, until the two users decide to establish a new session with a new session key

Please, Don’t try to find attacks
to the protocol, (for the moment)

The protocol

Message 1	A	$\rightarrow$	KDC	:	A, B
Message 2	KDC	$\rightarrow$	A	:	PK_B
Message 3	A	$\rightarrow$	B	:	$\{K_{sess}\}_{PK_B}$
Message 4	B	$\rightarrow$	A	:	$\{M\}_{K_{sess}}$

Which security properties could we require?

Which properties are of interest?

(i) *Authenticity of M*: message M should be authentic from B since only A and B should know the session key at the end of the protocol;

(ii) *Entity authentication*: if A receives the last message encrypted with the correct key, then A should be guaranteed that B has run the protocol with her (or at least is “alive”);

(iii) *Secrecy*: at the end of the protocol, the session key and the message M should be known only by A and B.

Message authenticity [Abadi and Gordon, 1999]

- Idea: consider all the possible runs of the protocol and require that, in such runs, A always receives the correct message M, i.e. the message B wanted to send to A

$\Rightarrow$ If this is true, we can conclude that the protocol is indeed guaranteeing that no one is able to force A accepting a faked message M'!

- Is this really enough? what do we exactly mean by “all the possible runs”?

Hostile environments

- We have to consider all the possible executions of the protocol in every possible (potentially hostile) environment.

$\Rightarrow$ Without a malicious enemy which tries to send a faked M' no attack is indeed possible.

- We thus summarize Message Authenticity as:

  "Whatever hostile environment is considered, A will never receive (as part of Message 4) during all her possible runs a message different from M".

Formalizing Message Authenticity

Message 1	A	$\rightarrow$	KDC	$: A, B$
Message 2	KDC	$\rightarrow$	A	$: PK_B$
Message 3	A	$\rightarrow$	B	$: \{K_{sess}\}_{PK_B}$
Message 4	B	$\rightarrow$	A	$: \{M\}_{K_{sess}}$

$$
\begin{aligned}
A(k) &\stackrel{\text{def}}{=} \overline{c_1}\,(A,B)\,.\,c_2(y)\,.\,\overline{c_3}\,\{k\}_y\,.\,c_4(w)\,.\\
&\qquad [\langle w,k\rangle \vdash_{dec} j]F(j)\\
B(m) &\stackrel{\text{def}}{=} c_3(y)\,.\,[\langle y, SK_B\rangle \vdash_{dec} z]\,\overline{c_4}\,\{m\}_z\\
KDC &\stackrel{\text{def}}{=} c_1(x,y)\,.\,\overline{c_2}\,PK_y\\
P(M) &\stackrel{\text{def}}{=} A(k_{sess}) \parallel B(M) \parallel KDC
\end{aligned}
$$

Message Authenticity in the spi-calculus style (1)

- Define a "secure specification":

$$
\begin{aligned}
A_{spec}(k, \underline{\mathbf{m}}) &\overset{\text{def}}{=} \overline{c_1}\,(A, B)\,.\,c_2(y)\,.\,\overline{c_3}\,\{k\}_y\,.\,c_4(w)\,.\\
&\qquad [\langle w, k\rangle \vdash_{dec} j]F(\underline{\mathbf{m}})\\
P(M) &\overset{\text{def}}{=} A(k_{sess}, \underline{\mathbf{M}}) \parallel B(M) \parallel KDC
\end{aligned}
$$

- **Remark:** The only message that can be delivered by A_{spec} is $M \Longrightarrow$ Guarantees *Authentication by construction*.

Message Authenticity in the spi-calculus style (2)

- For all continuations F, and for all messages M:

$$S \simeq S_{spec}$$

- where: $S \simeq S_{spec}$ iff there exists no “test” T, i.e. hostile environment, that is able to distinguish the two processes.
- **Remark:** the tester is both an *observer* and an *attacker*!

A simpler definition of Message Authenticity

In [Focardi, Gorrieri, Martinelli AMAST'00] we have proposed a simpler definition:

- consider $F(j) = \overline{rec}\, m$, i.e., event $\overline{rec}\, m$ correspons to the fact that A is receiving message m.

  For all the protocols of this form we state that:

  > "$P(M)$ guarantees message authenticity if whatever hostile environment is considered, an event $\overline{rec}\, M'$ with $M' \neq M$ can never occur".

- Theorem: The two defs are equivalent (for a class of protocols)

OK!

Now you can find the attacks!

Does the protocol guarantee message authenticity?

Message 1	A	$\rightarrow$	KDC	$: A, B$	
Message 2	KDC	$\rightarrow$	A	$: PK_B$	
Message 3	A	$\rightarrow$	B	$: \{K_{sess}\}_{PK_B}$	
Message 4	B	$\rightarrow$	A	$: \{M\}_{K_{sess}}$	event $\overline{rec}\, M$

The message authenticity attack

We denote with $E(U)$ the enemy which is impersonating U:

$$\begin{array}{lrcll}
1. & A & \rightarrow & KDC & : A, B \\
2. & KDC & \rightarrow & \mathbf{E}(A) & : PK_B \\
2'. & \mathbf{E}(KDC) & \rightarrow & A & : \mathbf{PK_E} \\
3. & A & \rightarrow & \mathbf{E}(B) & : \{K_{sess}\}_{\mathbf{PK_E}} \\
4. & \mathbf{E}(B) & \rightarrow & A & : \{\mathbf{M'}\}_{K_{sess}}
\end{array}$$

E intercepts this

event $\overline{rec}\, M'$

Entity Authentication

- Informally, entity authentication should allow the verification of an entity's claimed identity, by another entity
- several attempts in the literature to formalize this notion
- Here, we follow the ones based on *correspondence* between actions of the participants (e.g., see [Woo and Lam 1993, Lowe 1997]).
- in our protocol we would like that whenever A receives the last message then B has indeed executed the protocol

  $\Rightarrow$ A is guaranteed that "B" is the real identity of the other party

Defining Entity Authentication

- two events $\overline{commit}\,(A, B)$ and $\overline{run}\,(B, A)$ representing the fact that A has successfully terminated the protocol apparently with B and B has at least started the protocol (i.e., he has received a session key) with A.

$\Rightarrow$ Idea: Require that event $\overline{commit}\,(A, B)$ is always preceded by event $\overline{run}\,(B, A)$. In other words $\overline{commit}\,(A, B)$ should not happen if B has not started the protocol.

> "P guarantees entity authentication of B with respect to A if whatever hostile environment is considered, it can never occur an event $\overline{commit}\,(A, B)$ when $\overline{run}\,(B, A)$ has not occurred previously".

Does the protocol guarantee entity authentication?

$$\begin{array}{lrcll}
1. & A & \rightarrow & KDC & : A, B \\
2. & KDC & \rightarrow & A & : PK_B \\
3. & A & \rightarrow & B & : \{K_{sess}\}_{PK_B} \quad \text{event } \overline{run}(B, A) \\
4. & B & \rightarrow & A & : \{M\}_{K_{sess}} \quad \text{event } \overline{commit}(A, B)
\end{array}$$

The same attack sequence!

$$\begin{array}{llcll}
1. & A & \rightarrow & KDC & : A, B \\
2. & KDC & \rightarrow & \mathbf{E}(A) & : PK_B & \text{E intercepts this} \\
2'. & \mathbf{E}(KDC) & \rightarrow & A & : \mathbf{PK_E} \\
3. & A & \rightarrow & \mathbf{E}(B) & : \{K_{sess}\}_{\mathbf{PK_E}} \\
4. & \mathbf{E}(B) & \rightarrow & A & : \{\mathbf{M'}\}_{K_{sess}} & \overline{commit}\,(A, B)
\end{array}$$

Note that B is doing nothing!

$\Rightarrow A$ cannot be sure about the identity of the other party, i.e., no entity authentication is guaranteed.

entity authentication is not message authenticity

Example: consider a protocol in which there is no message "to be sent" by the entimty that we want to authenticate,

$\Rightarrow$ message authenticity becomes useless.

Consider the following (faulty) authentication protocol:

$$\begin{array}{lcccl} \text{Message } 1 & A & \rightarrow & B & : \{N_A\}_{K_{AB}} \\ \text{Message } 2 & B & \rightarrow & A & : N_A \end{array}$$

N_A is a nonce (a random challenge) and K_{AB} is a key shared between A and B.

Exercise: Show an entity authentication attack on this protocol which is not a message authenticity attack.

Secrecy

- Informally, it requires that messages declared to be secret should not be learnt by unauthorized users.
- consider a new event $\overline{learnt}\, M$ that represents the fact that a certain (secret) message M has been learnt by the external environment (i.e., by the enemy).

“P guarantees secrecy of m if whatever hostile environment is considered, the event $\overline{learnt}\, m$ can never occur”.

Secrecy attack on K_{sess}

Again we consider the same attack sequence:

$$\begin{array}{llcllll}
1. & A & \rightarrow & KDC & : & A, B & \\
2. & KDC & \rightarrow & \mathbf{E}(A) & : & PK_B & \text{E intercepts this} \\
2'. & \mathbf{E}(KDC) & \rightarrow & A & : & \mathbf{PK_E} & \\
3. & A & \rightarrow & \mathbf{E}(B) & : & \{K_{sess}\}_{\mathbf{PK_E}} & \overline{learnt}\ (\mathbf{K_{sess}}) \\
4. & \mathbf{E}(B) & \rightarrow & A & : & \{\mathbf{M'}\}_{\mathbf{K_{sess}}} &
\end{array}$$

The example (partially) repaired

- The attacks work since it is possible to fake Message 2 in order to provide a wrong public key, i.e., not the one associated to B.
- In actual implementations, the KDC sends the public key together with the name of the user, all signed with its own key.

Message 1	A	$\rightarrow$	KDC	: A, B
Message 2	KDC	$\rightarrow$	A	: $\{PK_B, B\}_{SK_{KDC}}$
Message 3	A	$\rightarrow$	B	: $\{K_{sess}\}_{PK_B}$
Message 4	B	$\rightarrow$	A	: $\{M\}_{K_{sess}}$

- Exercise: Show that a message authenticity attack is still possible if B sends more than one message with the same session key.

Does the repaired protocol guarantee secrecy of M?

This was one of the “desired” properties

Let us check it out!

$$\begin{array}{llcll} 3. & E(A) & \rightarrow & B & : \{K'_{sess}\}_{PK_B} \\ 4. & B & \rightarrow & E(A) & : \{M\}_{K'_{sess}} \qquad \overline{learnt}\, M \end{array}$$

Even if the public key is now authenticated the protocol does not guarantee to B that A is the initiator of the run (no mutual authentication).

Exercise: Show that, in the repaired protocol, there is no entity authentication of A with respect to B and no message authenticity of K_{sess}.

A general scheme for security properties

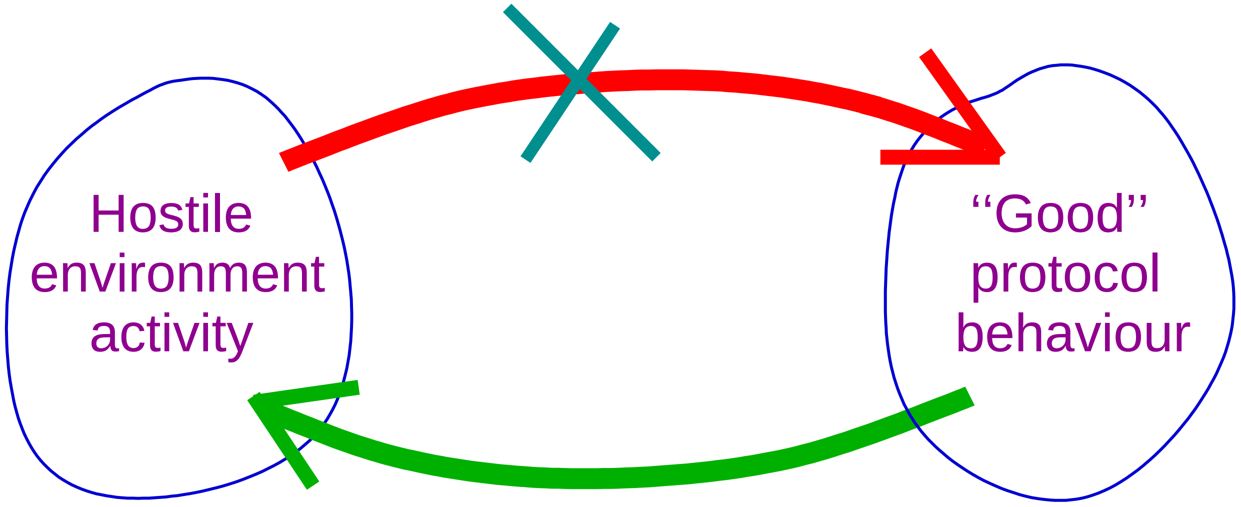
We have seen that several different properties (message authenticity, entity authentication, secrecy) can be (informally) written in a similar style which sounds like:

"P guarantees a security property S if, whatever hostile environment is considered, P never shows some particular *bad behaviour*".

In general, this set of bad behaviours depends on the particular property and sometimes may also depend on the protocol P.

Example: in message authenticity we need the parameter m of P in order to define what is a bad behaviour

How can we see that as Non-Interference?



Idea: The hostile environment should not be able to induce bad behviours of the protocol.

$\Rightarrow$ Interference $\equiv$ induced bad behaviour

Generalized NDC

We denote by $\alpha_S(P)$ the set of all possible good behaviour of P with respect to property S:

> "P guarantees a security property S if, whatever hostile environment is considered, P always shows behaviours in $\alpha_S(P)$".

In a process algebraic style (Generalized NDC):

$$P \text{ is } GNDC^{\alpha}_{\leq} \text{ iff } \forall X \in \mathcal{E}_C : P \underset{C}{\parallel} X \leq \alpha(P)$$

where $P \parallel_C X$ stands for $(P \parallel X) \setminus C$, $\leq$ is a behavioural preorder and α is a function from processes to processes.

Instantiating GNDC

We can just define a specific property by suitably instantiating the function $\alpha(P)$ and the preorder $\leq$.

We reconsider the properties presented so far, showing informally their corresponding $\alpha(P)$ functions.

For all of the three, $\leq$ is the trace preorder:

- *Message authenticity*: $\alpha_{MA}(P(M))$ is the process where all the visible actions (events) of $P(M)$ may occur but $rec(m)$ is only present with $m = M$.

Instantiating GNDC (2)

- *Entity authentication*: $\alpha_{EA}(P)$ is the process where all the visible actions (events) of P may occur but $\overline{commit}\,(A,B)$ is always preceded by $\overline{run}\,(B,A)$.
- *Secrecy of* m: $\alpha_{Sec}(P(m))$ is the set of processes where all the visible actions (events) of $P(m)$ may occur except for $\overline{learnt}\,m$.

Moreover:

- *NDC*: $\alpha_{NDC}(P(M)) = P(M) \setminus C$.

A specification language

for crypto protocols

Extending SPA with cryptography

CryptoSPA is an extension of SPA where the processes are provided with some primitives for manipulating messages and performing encryption and decryption.

It is based on the following elements:

- A set $I = \{c_1, c_2, \dots\}$ of *input* channels, a set $O = \{\bar{c_1}, \bar{c_2}, \dots\}$ of *output* ones;
- A set M of basic messages and a set K of encryption keys with a function $\cdot^{-1} : K \to K$ such that $(k^{-1})^{-1} = k$. The set $\mathcal{M}$ of all messages is defined as the least set such that $M \cup K \in \mathcal{M}$ and $\forall m \in \mathcal{M}$, $\forall k \in K$ we have that (m, m') and $\{m\}_k$ also belong to $\mathcal{M}$;

- A set $C \subseteq I \cup O$ (s.t. $c \in C$ iff $\bar{c} \in C$) of *public* channels where the enemy can intercept and fake messages;
- A function $Msg(c) : I \cup O \longrightarrow \mathcal{P}(\mathcal{M})$ which maps every channel c into the set of possible messages that can be sent and received along such a channel. Msg is such that $Msg(c) = Msg(\bar{c})$.
- A set $Act = \{c(m) \mid c \in I, m \in Msg(c)\} \cup \{\bar{c}m \mid \bar{c} \in O, m \in Msg(c)\} \cup \{\tau\}$ of actions;
- A set $Const$ of constants, ranged over by A.

The CryptoSPA Language

E	$::=$	$\underline{0}$	*empty process*
	$\mid$	$c(x).E$	*input*
	$\mid$	$\overline{c}\, e.E$	*output*
	$\mid$	$\tau.E$	*internal action*
	$\mid$	$E + E$	*non-det. choice*
	$\mid$	$E \parallel E$	*parallel composition*
	$\mid$	$E \setminus L$	*restriction*
	$\mid$	$E[f]$	*relabelling*
	$\mid$	$A(m_1, \ldots, m_n)$	*constant*
	$\mid$	$[e = e']E; E$	*matching*
	$\mid$	$[\langle e_1 \ldots e_r \rangle \vdash_{rule} x]E; E$	*message handling and cryptography*

Message manipulation and cryptography

we define an inference system which formalizes the way messages may be manipulated by processes.

$$\frac{m \quad m'}{(m, m')}(\vdash_{pair}) \qquad \frac{(m, m')}{m}(\vdash_{fst}) \qquad \frac{(m, m')}{m'}(\vdash_{snd})$$

$$\frac{m \quad k}{\{m\}_k}(\vdash_{enc}) \qquad \frac{\{m\}_k \quad k^{-1}}{m}(\vdash_{dec})$$

where $m, m' \in \mathcal{M}$ and $k, k^{-1} \in K$.

Semantics (the SPA part)

$$(input)\ \frac{m \in Msg(c)}{c(x).E \xrightarrow{c(m)} E[m/x]} \qquad (output)\ \frac{m \in Msg(c)}{\overline{c}\,m.E \xrightarrow{\overline{c}\,m} E}$$

$$(internal)\ \frac{}{\tau.E \xrightarrow{\tau} E} \qquad (+_1)\ \frac{E \xrightarrow{a} E'}{E + E_1 \xrightarrow{a} E'}$$

$$(\|_1)\ \frac{E \xrightarrow{a} E'}{E \parallel E_1 \xrightarrow{a} E' \parallel E_1} \qquad (\|_2)\ \frac{E \xrightarrow{c(m)} E' \quad E_1 \xrightarrow{\overline{c}\,m} E_1'}{E \parallel E_1 \xrightarrow{\tau} E' \parallel E_1'}$$

$$(=_1)\ \frac{m \neq m' \quad E_2 \xrightarrow{a} E_2'}{[m = m']E_1; E_2 \xrightarrow{a} E_2'} \qquad (=_2)\ \frac{m = m' \quad E_1 \xrightarrow{a} E_1'}{[m = m']E_1; E_2 \xrightarrow{a} E_1'}$$

$$([f])\ \frac{E \xrightarrow{a} E'}{E[f] \xrightarrow{f(a)} E'[f]} \qquad (\backslash L)\ \frac{E \xrightarrow{a} E' \quad chan(a) \notin L}{E\backslash L \xrightarrow{a} E'\backslash L}$$

$$(def)\ \frac{E[m_1/x_1, \ldots, m_n/x_n] \xrightarrow{a} E' \quad A(x_1, \ldots, x_n) \stackrel{\text{def}}{=} E}{A(m_1, \ldots, m_n) \xrightarrow{a} E'}$$

Semantics (modelling cryptography)

- Informally, the $[\langle m_1 \ldots m_r \rangle \vdash_{rule} x]E_1; E_2$ process tries to deduce an information z from the tuple of messages $\langle m_1 \ldots m_r \rangle$ through one application of rule $\vdash_{rule}$; if it succeeds then it behaves like $E_1[z/x]$, otherwise like E_2;

$$(\mathcal{D}_1)\,\frac{\langle m_1 \ldots m_r \rangle \vdash_{rule} m \qquad E_1[m/x] \xrightarrow{a} E_1'}{[\langle m_1 \ldots m_r \rangle \vdash_{rule} x]E_1; E_2 \xrightarrow{a} E_1'}$$

$$(\mathcal{D}_2)\,\frac{\nexists m : \langle m_1 \ldots m_r \rangle \vdash_{rule} m \qquad E_2 \xrightarrow{a} E_2'}{[\langle m_1 \ldots m_r \rangle \vdash_{rule} x]E_1; E_2 \xrightarrow{a} E_2'}$$

Modelling Cryptography (Examples)

- Example 1: Consider process

$$[\langle \{m\}_k, k^{-1}\rangle \vdash_{dec} x]E_1; E_2$$

It decrypts message $\{m\}_k$ through key k^{-1} and behaves like $E_1[m/x]$.

- Example 2: Process

$$[\langle \{m\}_k, k'\rangle \vdash_{dec} x]E_1; E_2$$

(with $k' \neq k^{-1}$) tries to decrypt the message with the wrong inverse key k' and (since it is not permitted by $\vdash_{dec}$) it behaves like E_2.

A very simple protocol

We present a very simple example of a protocol where A sends a message m_A to B encrypted with a key k_{AB} shared between A and B.

$$\begin{array}{rcl} A(m,k) & \stackrel{\text{def}}{=} & [\langle m,k\rangle \vdash_{enc} x]\overline{c}\,x \\ B(k) & \stackrel{\text{def}}{=} & c(y).[\langle y,k\rangle \vdash_{dec} z]\overline{rec}\,z \\ P & \stackrel{\text{def}}{=} & A(m_A,k_{AB}) \parallel B(k_{AB}) \end{array}$$

where $k_{AB}^{-1} = k_{AB}$ (symmetric encryption), and $Msg(c) = \{\{m\}_k \mid m \in M, k \in K\}$.

(We usually adopt the simpler notation: $A(m,k) \stackrel{\text{def}}{=} \overline{c}\,\{m\}_k$)

A very simple protocol (execution)

- We want to analyze the execution of P with no intrusions,

$\Rightarrow$ consider $P \setminus \{c\}$, as the restriction guarantees that c is a completely secure channel.

- We obtain a system which can only execute action $\overline{rec}\, m_A$ (correct transmission of m_A from A to B).

- In particular, $A(m_A, k_{AB}) \xrightarrow{\overline{c}\,\{m_A\}_{k_{AB}}} \underline{0}$

- $B(k_{AB}) \xrightarrow{c(\{m_A\}_{k_{AB}})} [\langle\{m_A\}_{k_{AB}}, k_{AB}\rangle \vdash_{dec} z]\overline{rec}\, z$. So

$$\begin{aligned} P \setminus \{c\} & \xrightarrow{\tau} & (\underline{0} \parallel [\langle\{m_A\}_{k_{AB}}, k_{AB}\rangle \vdash_{dec} z]\overline{rec}\, z) \setminus \{c\} \\ & \xrightarrow{\overline{rec}\, m_A} & (\underline{0} \parallel \underline{0}) \setminus \{c\} \end{aligned}$$

Trace equivalence

The properties we have seen so far are based on the notion of traces:

- The expression $E \overset{\alpha}{\Longrightarrow} E'$ is a shorthand for $E(\xrightarrow{\tau})^* E_1 \xrightarrow{\alpha} E_2(\xrightarrow{\tau})^* E'$,

  Moreover, $E \overset{\gamma}{\Longrightarrow} E'$ if and only if $E \overset{\alpha_1}{\Longrightarrow} E_1 \overset{\alpha_2}{\Longrightarrow} \cdots \overset{\alpha_{n-1}}{\Longrightarrow} E_{n-1} \overset{\alpha_n}{\Longrightarrow} E'$.

- $T(E) = \{\gamma \in (Act \setminus \{\tau\})^* \mid \exists E' : E \overset{\gamma}{\Longrightarrow} E'\}$.

- $E \leq_{trace} F$ iff $T(E) \subseteq T(F)$.

- $E \approx_{trace} F$ iff $E \leq_{trace} F$ and $F \leq_{trace} E$, i.e., iff $T(E) = T(F)$.

Very simple protocol (analysis)

Consider again the very simple protocol P.

Message 1 $\quad A \quad \rightarrow \quad B \quad : \{m_A\}_{k_{AB}} \quad \overline{rec}\, m_A$

$$\begin{aligned} A(m,k) &\overset{\text{def}}{=} \bar{c}\,\{m\}_k \\ B(k) &\overset{\text{def}}{=} c(y).[\langle y, k\rangle \vdash_{dec} z]\overline{rec}\, z \\ P(m,k) &\overset{\text{def}}{=} A(m,k) \parallel B(k) \end{aligned}$$

where $c \in C$ (public channel).

Very simple protocol (analysis)

$\Rightarrow$ Since only A and B knows k_{AB}, protocol $P(m_A, K_{AB})$ should guarantee the authenticity of m_A, i.e., $GNDC^{\alpha_{MA}(P(m_A,K_{AB}))}_{\leq_{trace}}$:

$$\forall X \in \mathcal{E}_C : P(m_A, K_{AB}) \underset{C}{\|} X \leq_{trace} \alpha(P(m_A, K_{AB}))$$

where

- $\alpha(P(m_A, K_{AB})) \stackrel{\text{def}}{=} \overline{rec}\, m_A.\alpha(P(m_A, K_{AB}))$
- $\mathcal{E}_C \stackrel{\text{def}}{=} \{E \in \mathcal{E} \mid sort(E) \subseteq C\}$

what do you think?

Too powerful enemies!

Consider an enemy defined as Alice

$$X(m,k) \stackrel{\mathrm{def}}{=} \overline{c}\,\{m\}_k$$

It belongs to $\mathcal{E}_C$ since $Sort(X(m,k)) = \{c\}$.

$\Rightarrow$ $X(m_X, k_{AB})$ is a process that knows k_{AB} and thus can send a faked message $\{m_X\}_{k_{AB}}$ to B!

Indeed

$$(P \,\|\, X(m_X, k_{AB})) \setminus \{c\}$$

(after one τ step) can give as output $\overline{out}\, m_X$ which represents the fact that B as received m_X instead of m_A.

$X(m_X, k_{AB})$ is "guessing" k_{AB} $\Rightarrow$ breaking cryptography!

Limiting enemy's knowledge

We solve this problem by imposing some constraints on the initial data that are known by the enemies:

- let $ID(E)$ the set of messages that syntactically appear in E (i.e., all the messages that are initially known by E)
- let $\phi_I \subseteq \mathcal{M}$ be the initial knowledge that we would like to give to the enemy (i.e., public information plus enemy private keys or nonces . . .).

$$\mathcal{E}_C^{\phi_I} = \{X \mid X \in \mathcal{E}_C \text{ and } ID(X) \subseteq \mathcal{D}(\phi_I)\}$$

- We consider as hostile processes only the ones belonging to this particular set.

Fixing GNDC

In the example above, if we require that k_{AB} is not deducible from ϕ_I we obtain that the behavior of $X(m_X, k_{AB})$ cannot be simulated by any process in $\mathcal{E}_C^{\phi_I}$ (which can never execute $\overline{c}\,\{m_X\}_{k_{AB}}$)

We finally obtain the following:

E is $GNDC^{\alpha}_{\leq}$ iff

$$\forall X \in \mathcal{E}_C^{\phi_I} : E \underset{C}{\|} X \leq \alpha(E)$$

Modelling crypto protocols

We have emphasized two important (strictly related) aspects that have to be considered when modelling cryptographic protocols:

1. We need to model cryptography
2. We need to keep some information secret

These can be approached in different ways (see, e.g., spi-calculus [Abadi, Gordon 1999]) . . . but are both necessary!

The most powerful enemy!

The quantification over all the possible enemies is problematic for the verification of GNDC (especially for automatic verification)!

- $\leq$ is a *pre-congruence* w.r.t. the operator $\|_C$ if it is a preorder and for every $E, F, F' \in \mathcal{E}$ if $F \leq F'$ then $E \|_C F \leq E \|_C F'$.
- **Proposition:** If $\leq$ is a pre-congruence w.r.t. $\|_C$ and if there exists a process $Top \in \mathcal{E}_C^{\phi_I}$ such that for every process $X \in \mathcal{E}_C^{\phi_I}$ we have $X \leq Top$, then:

$$E \in GNDC_{\leq}^{\alpha} \quad \text{iff} \quad E \underset{C}{\|} Top \leq \alpha(E)$$

The most powerful enemy for trace-equivalence

- The properties we have seen so far are based on trace preorder which is a pre-congruence with respect to $\|_C$ operator.
- Consider a family of processes $(Top^C_{trace})_\phi$ each representing the instance of the enemy with knowledge ϕ.

$$(Top^C_{trace})_\phi \quad = \quad \sum_{c \in C} c(x).(Top^C_{trace})_{\phi \cup \{x\}} + \sum_{\substack{c \in C \\ m \in \mathcal{D}(\phi) \cap Msg(c)}} \overline{c}\, m.(Top^C_{trace})_\phi$$

Theorem: if $X \in \mathcal{E}_C^{\phi_I}$ then $X \leq_{trace} (Top^C_{trace})_{\phi_I}$. ■

A general result for trace-based GNDC

As a direct consequence of the previous theorem, we obtain the following result for the family properties $GNDC^{\alpha}_{\leq_{trace}}$:

Corollary: For every function $\alpha : \mathcal{E} \rightarrow \mathcal{E}$

$$E \in GNDC^{\alpha}_{\leq_{trace}} \quad \text{iff} \quad E \underset{C}{\|} (Top^{C}_{trace})_{\phi_I} \leq_{trace} \alpha(E) \quad \blacksquare$$

which is applicable to all the properties we have seen so far.

Does this reminds something to you?

Formalization of α function (an example)

Function $\alpha_{EA}(P)$ can be defined as follows:

$$\begin{aligned} P' &= Top_{trace}^{Sort(P)\setminus\{run,commit\}} \\ P''(x,y) &= \overline{run}\,(x,y)\ .\ \overline{commit}\,(y,x) \\ \alpha_{EA}(P) &= P' \parallel P''(A,B) \end{aligned}$$

- It is possible to define in a similar way the α function for the other properties (with a little trick for secrecy).

Trace-semantics is not always enough

Non repudiation: is related to the possibility of considering a certain message as a signed contract, i.e., the sender cannot repudiate it.

$\Rightarrow$ crucial property for e-commerce!

Example: imagine M is some form of electronic payment:

$$\begin{array}{llllll} \text{Message } 1 & A & \rightarrow & B & : & M, \{M, B\}_{SK_A} \\ \text{Message } 2 & B & \rightarrow & A & : & \{M, A\}_{SK_B} \end{array}$$

- B wants the payment from A;
- A wants a proof that B has received the money (a receipt)

A and B cannot repudiate the messages! Is it enough?

fairness

$\Rightarrow$ Bob could get the money and never send the receipt!

Fairness (fair message exchange): no one of the parties involved in the exchange should get an advantage by obtaining the information before the other party is also able to obtain it.

$\Rightarrow$ It is often the case that non-repudiation with fairness is desirable!

Informally:

> "P guarantees non-repudiation with fairness to A on a message M if, whatever malicious B is considered, if B gets evidence that A has originated M than also A will eventually obtain the evidence that B has received M".

How can we encode it in our scheme?

Non-Repudiation with fairness as GNDC

We observe two important points:

- one of the parties (B) takes the role of the hostile environment.

  $\Rightarrow$ we want to see if B is able to cheat A by keeping the payment without releasing the receipt!

- this property cannot be expressed as a safety property (i.e., nothing bad happens).

  We are requiring that something good should happen:

  $\Rightarrow$ if B gets his evidence then also A should soon or later get her evidence.

Non-Repudiation with fairness as GNDC (2)

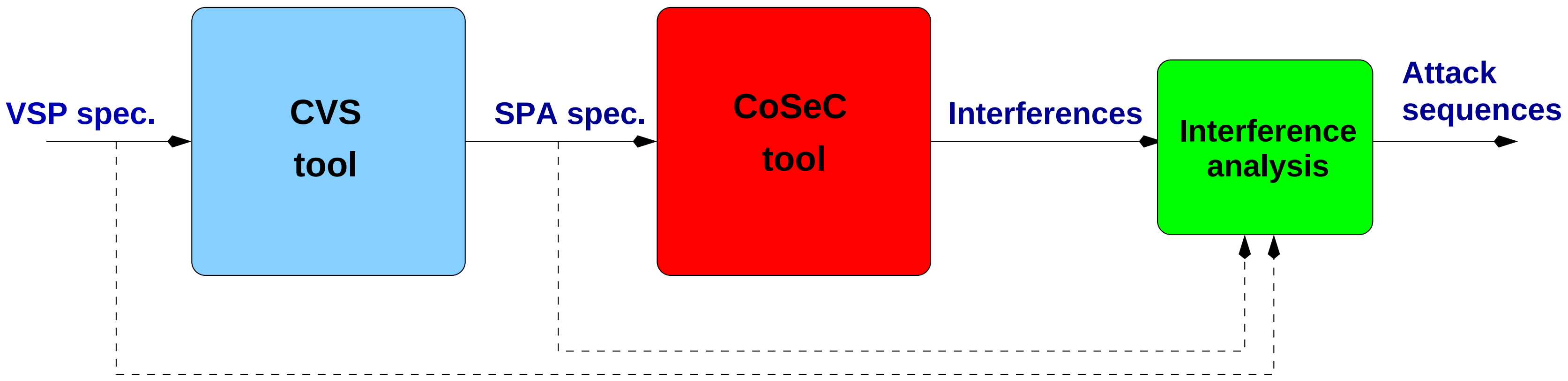
- P only contains A specification (Bob is the enemy!);
- We add two events ev_A and ev_B representing the facts that A and B get their own evidence, respectively;
- $\alpha_{NRF}(P)$ is the process where every time ev_B happens then ev_A must eventually happen.
- $\leq$ must be a deadlock-sensitive preorder (e.g., testing, bisimulation) denoted with $\leq_{deadlock}$.

  As, no deadlock should be possible after B get his evidence!

P guarantees non-repudiation with fairness to A iff P is $GNDC^{\alpha_{NRF}(P)}_{\leq_{deadlock}}$

Automatic verification of Cryptographic Protocols

Analysis scheme



CVS compiler and VSP language

CVS/VSP provides:

1. a notation more abstract than SPA was defined, called VSP notation;

2. a set of rules was defined to automatically build the enemy component;

3. a compiler, called CVS, was implemented to translate a VSP script into SPA code [A. Durante, 1998].

Attacks as interferences

After the VSP specification is compiled just one NDC check is performed.

Why does this work?

- NDC may be seen as a sufficient condition for every property which is based on trace-preorder.
- The result holds for what we will call *good candidates* for a function α, i.e., processes E such that $E \setminus C \leq_{trace} \alpha(E)$.

$\Rightarrow$ reasonable since we certainly want that at least the protocol under no attacks (i.e., $E \setminus C$) “satisfies” $\alpha(E)$.

Attacks as interferences (2)

Proposition: Let $\alpha(E)$ be a function between processes and let E be a good candidate for α. Then, E is $GNDC^{E\backslash C}_{\approx_{trace}}$ implies that E is $GNDC^{\alpha(E)}_{\leq_{trace}}$. ■

Intuition: $GNDC^{E\backslash C}_{\approx_{trace}}$ requires that the protocol behaviour is completely preserved even under attacks.

So if E satisfies a certain property under no attacks (i.e., it is a good candidate), then $GNDC^{E\backslash C}_{\approx_{trace}}$ will preserve such a property also under every possible attack.

The Woo and Lam one-way Authentication protocol

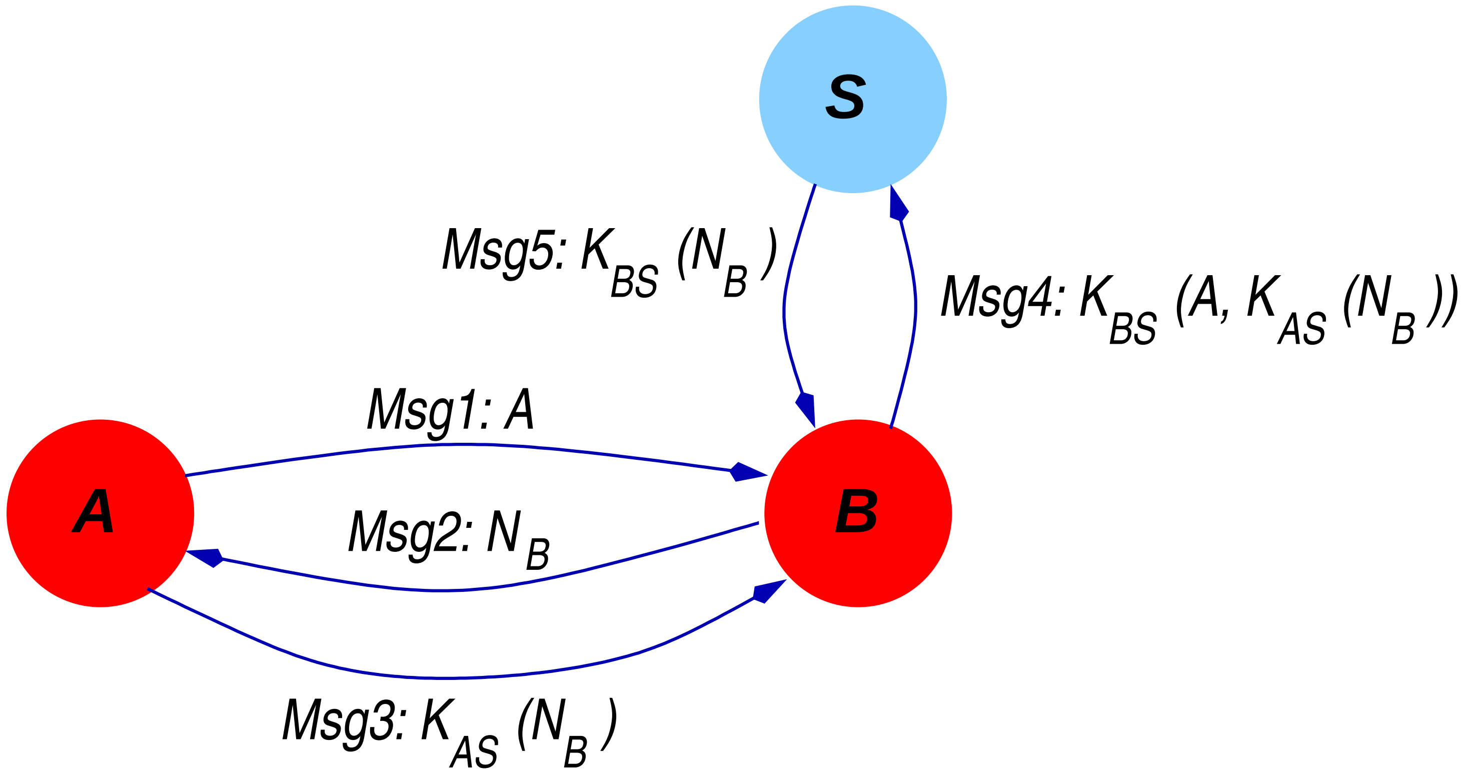
We consider the Woo and Lam one-way authentication protocol [Woo and Lam, 1992].

Aim: establish one-way authentication between A and B, using an authentication Server (S).

Woo and Lam state that the protocol is correct if:

> “whenever a responder finishes an execution of the protocol, the initiator of the protocol is in fact the principal claimed in the initial message”.

The protocol



The protocol

Message 1	A	$\rightarrow$	B	$: A$
Message 2	B	$\rightarrow$	A	$: N_B$
Message 3	A	$\rightarrow$	B	$: \{N_B\}_{K_{AS}}$
Message 4	B	$\rightarrow$	S	$: \{A, \{N_B\}_{K_{AS}}\}_{K_{BS}}$
Message 5	S	$\rightarrow$	B	$: \{N_B\}_{K_{BS}}$

The VSP specification

```
#Names
Agent:  A,B,E,S;
Nonce:  NB,NE;

#ActionDec
msg1(Agent,Agent)
msg2(Nonce)
msg3(crypt(SymKey,Nonce))
msg4(crypt(SymKey,Agent,crypt(SymKey,Nonce)))
msg5(crypt(SymKey,Nonce))
i_run(Agent,Agent)       cond($0!=$1)
r_run(Agent,Agent)
i_commit(Agent,Agent)
r_commit(Agent,Agent)

#RolesDec
Initiator
Responder
Server
Enemy
```

The VSP specification (Alice)

```
#ProcessesDef
Initiator(src :  Agent)
Var
Agent:  dst1;        cond(Responder,Enemy)
Nonce:  nonr1;

Begin
      i_run(src,dst1).
      'msg1(src,dst1).
      msg2(nonr1).
      'msg3(enc(KA(src),nonr1)).
      'i_commit(src,dst1)
End
```

The VSP specification (Bob)

```
Responder(dst:  Agent, nonr:  Nonce)
Var
Agent:  src1; cond(Initiator,Enemy)
Agent:  src4; cond(Initiator,Enemy)
Nonce:  nonr2,nonr1;

Begin
      msg1(src1,dst).
      'r_run(dst,src1).
      'msg2(nonr).
      msg3(enc(KA(src4),nonr2)).
      'msg4(enc(KA(dst),src1,enc(KA(src4),nonr2))).
      msg5(dec(KA(dst),nonr1)).
      if (nonr1=nonr) then
            'r_commit(dst,src1)
      endif
End
```

The VSP specification (Server)

```
Server(ser:  Agent)
Var
Agent:  dst2, src2, src3;
Nonce:  nons;

Begin
      msg4(dec(KA(dst2),src2,dec(KA(src3),nons))).
      if (src2=src3) then
            'msg5(enc(KA(dst2),nons))
      endif
End
```

The VSP specification (analysis parameters)

```
#Processes
Initiator(A)
Responder(B,NB)
Server(S)
Enemy(E,NE)

#RestrictionOn
channel(msg1,msg2,msg3,msg4,msg5)

#EnemyKnowledge
Agent:  All;
Memory:False;

#Observe
i_run(Initiator,Responder)
r_commit(Responder,Initiator)
```

The analysis

By analyzing the VSP specification of the Woo-Lam protocol we find out that it is not NDC.

The interference trace is the following:

```
'r_commit_B_A
```

Entity authentication attack: B is correctly terminating the protocol convinced to communicate with A but A has never started the protocol with B.

The analysis

If we extend the set of observable actions

```
#Observe
r_run(Responder,Agent)
i_run(Initiator,Agent)
r_commit(Responder,Agent)
i_commit(Initiator,Agentg)
```

we better identify the attack sequence:

```
i_run_A_E,'r_run_B_A,'i_commit_A_E,'r_commit_B_A
```

The attack sequence

Message $1.a$	A	$\rightarrow$	E	:	A
Message $1.b$	$E(A)$	$\rightarrow$	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	:	$\{N_B\}_{K_{AS}}$
Message $3.b$	$E(A)$	$\rightarrow$	B	:	$\{N_B\}_{K_{AS}}$
Message $4.b$	B	$\rightarrow$	S	:	$\{A, \{N_B\}_{K_{AS}}\}_{K_{BS}}$
Message $5.b$	S	$\rightarrow$	B	:	$\{N_B\}_{K_{BS}}$

We did not know the attack!

- This particular attack sequence is undocumented, even if it is not the first one found in the literature on this protocol.
- Moreover, it is the only one which can be carried out in a *single* session, where only one initiator and one responder are considered;
- all the other attacks reported in the literature, occurring in multiple sessions, were successfully revealed through CVS by choosing an appropriate session definition;
- In [Durente, Focardi, Gorrieri, ACM TOSEM'00] you find an example where we check different properties with just one NDC check.

Analyzed protocols

Failures upon Unflawed protocols:

Protocol	Our result	Lowe's result	Clark Jacobs
Woo Lam Public Key one-way Authentication protocol	A	U	U
ISO Public Key Two-Pass Parallel Authentication protocol	A	N	N

New failures upon flawed protocols:

Protocol	Our result	Lowe's result	Clark Jacobs
Encrypted Key Exchange-EKE	A	A	A
Station to Station protocol	A	A	?
Woo Lam Symmetric Key one-way Authentication protocol	A	A	A

Old Failures upon flawed protocols

Protocol	Our result	Lowe's result	Clark Jacobs
ISO Sym Key One-Pass Unilateral Authentication Protocol	A	A	N
ISO Sym Key Two-Pass Unilateral Authentication Protocol	N	N	N
ISO Sym Key Two-Pass Mutual Authentication Protocol	A	A	N
ISO Sym Key Three-Pass Mutual Authentication Protocol	N	N	N
ISO Public Key One-Pass Unilateral Authentication Protocol	A	A	N
ISO Public Key Two-Pass Unilateral Authentication Protocol	N	N	N
ISO Public Key Two-Pass Mutual Authentication Protocol	A	A	N
ISO Three-Pass Mutual Authentication Protocol	N	N	N
Andrew Secure RPC protocol	A	A	A
Denning-Sacco protocol	N	N	N
Otway-Rees protocol	A	A	A(?)
Amended Needham-Schroeder Protocol	N	N	N
Needham-Schroeder Signature Protocol	A	N	A

Protocol	Our result	Lowe's result	Clark Jacobs
Needham-Schroeder Public Key Protocol	A	A	A
Wide-Mouthed Frog protocol	A	A	A
Yahalom protocol	N	N	N
Carlsen's Secret Key Initiator protocol	N	N	N
Woo-Lam Symmetric Key one-way Authentication protocol	A	A	A
Woo-Lam Symmetric Key mutual Authentication protocol	A	A	A(?)
Kenhe Langerndorfer Schoenwalder	A	A	A
Kao Chow Repeated Authentication Protocol (1)	A	A	A
Kao Chow Repeated Authentication Protocol (2)	N	N	N
Shamir Rivest Adelman Three Pass protocol	A	A	A
Davis Switch Private Key Certificates Protocol (1)	A	A	A
TMN protocol (1) original version	A	A	?
TMN protocol (2) Lowe version	A	A	?
TMN protocol (3) Lowe version	A	A	?

Conclusion

- NDC as a formalization of Non-Interference and Information Flow; GNDC as a general scheme for security properties; CVS/VSP: a tool for the verification of crypto protocols (many protocols analyzed!).

present work:

- Extension to (s)pi-calculus and Ambient-calculus (mobility!).
- complete classification of properties through GNDC (as done for Non-Interference...) $\Rightarrow$ just one NDC check!
- New verification tools exploiting, e.g., partial model checking, symbolic representations, space state reduction . . .
- Extension with probabilities (stochastic process-algebra)

References

- Classification of NI properties for systems, NDC and CoSeC tool [Focardi and Gorrieri, JCS 1995], [Focardi, Gorrieri, IEEE TSE 1997] and [Focardi, Gorrieri, FOSAD'00 LNCS]
- GNDC [Focardi, Martinelli, FM'99], [Focardi, Gorrieri, Martinelli, ICALP'00] [Focardi, Gorrieri, Martinelli AMAST'00] and other.
- CVS/VSP [A. Durante 1998], [Durante, Focardi, Gorrieri, IEEE CSFW'99], [Durante, Focardi, Gorrieri, ACM TOSEM'00]
- My homepage: (you can download most of the papers) `http://www.dsi.unive.it/~focardi`
- e-mail: `focardi@dsi.unive.it`

Thank you for

your attention!