SPAN+AVISPA for Verifying Cryptographic Protocols

Thomas Genet

genet@irisa.fr

IRISA/Université de Rennes 1
Outline

1. Cryptography and Protocols Basics
2. What are we going to formalise and verify? The Dolev-Yao model
3. Basics of Protocol Formalization in SPAN+AVISPA
4. Defining a simple protocol with SPAN+AVISPA
5. Other examples and conclusion
Outline

1. Cryptography and Protocols Basics

2. What are we going to formalise and verify? The Dolev-Yao model

3. Basics of Protocol Formalization in SPAN+AVISPA

4. Defining a simple protocol with SPAN+AVISPA

5. Other examples and conclusion
Cryptography in Three Slides

Let:
- $m, m_1, m_2$ be messages
- $K$ be a key
- $A$ and $B$ be agents
Cryptography in Three Slides

Let:
- $m, m_1, m_2$ be messages
- $K$ be a key
- $A$ and $B$ be agents

We denote by:
- $m_1, m_2$ the message composed of $m_1$ and $m_2$
Cryptography in Three Slides

Let:
- $m, m_1, m_2$ be messages
- $K$ be a key
- $A$ and $B$ be agents

We denote by:
- $m_1, m_2$ the message composed of $m_1$ and $m_2$
- $\{m\}_K$ the message $m$ encrypted with key $K$. 
Cryptography in Three Slides

Let:

- $m, m_1, m_2$ be messages
- $K$ be a key
- $A$ and $B$ be agents

We denote by:

- $m_1, m_2$ the message composed of $m_1$ and $m_2$
- $\{m\}_K$ the message $m$ encrypted with key $K$.
- $A \leftrightarrow B : m$ $A$ sends the message $m$ to $B$. 
Cryptography in Three Slides

Let:
- \( m, m_1, m_2 \) be messages
- \( K \) be a key
- \( A \) and \( B \) be agents

We denote by:
- \( m_1, m_2 \) the message composed of \( m_1 \) and \( m_2 \)
- \( \{m\}_K \) the message \( m \) encrypted with key \( K \)
- \( A \leftrightarrow B : m \) \( A \) sends the message \( m \) to \( B \)
- \( I(A) \leftrightarrow B : m \) \( I \) impersonates \( A \) to send the message \( m \) to \( B \)
Cryptography in Three Slides

Let:

- \( m, m_1, m_2 \) be messages
- \( K \) be a key
- \( A \) and \( B \) be agents

We denote by:

- \( m_1, m_2 \) the message composed of \( m_1 \) and \( m_2 \)
- \( \{m\}_K \) the message \( m \) encrypted with key \( K \).
- \( A \leftrightarrow B : m \) \( A \) sends the message \( m \) to \( B \).
- \( I(A) \leftrightarrow B : m \) \( I \) impersonates \( A \) to send the message \( m \) to \( B \)
- \( A \leftrightarrow I(B) : m \) the message \( m \) sent by \( A \) to \( B \) is intercepted by \( I \)
Cryptography in Three Slides (II)

Asymmetric encryption (e.g. RSA, PGP, . . . ):
- The public key ($K_A$) is given to all agents.
- The private key ($K_{1A}$) is only known by $A$.
- $\text{decrypt}(\{m\} \hookrightarrow K_{1A}) = m$.

Symmetric encryption (e.g. AES):
- $K_{AB} \odot K_{1AB} = \text{decrypt}(\{m\} \hookrightarrow K_{1AB}) = m$. 

T. Genet (IRISA)
Cryptography in Three Slides (II)

- Assymmetric encryption (e.g. RSA, PGP, . . . ) :
  - The public key ($K_A$) is given to all agents
  - The private key ($K_{1A}$) is only known by $A$
  - Decrypt ($\{m\} \overset{K_A}{\mapsto} K_{1A} \mapsto m$)

- Symmetric encryption (e.g. AES) :
  - $K_{AB} \overset{\text{crypt}}{\mapsto} K_{AB}$
  - Decrypt ($\{m\} \overset{K_{AB}}{\mapsto} K_{AB} \mapsto m$)
Cryptography in Three Slides (II)

- Asymmetric encryption (e.g. RSA, PGP, . . .):
  - the public key \( K_A \) is given to all agents

Symmetric encryption (e.g. AES):
- \( K_{AB} \) \( K_{1AB} \)
- decrypt \( \{m\} K_{AB} \hat{\rightarrow} K_{AB} \) = m
Cryptography in Three Slides (II)

- Assymmetric encryption (e.g. RSA, PGP, ...):
  - the public key ($K_A$) is given to all agents
  - the private key ($K_A^{-1}$) is only known by $A$

- Symmetric encryption (e.g. AES):
  - $K_{AB}$ is used for encryption
  - $K_{AB}$ is used for decryption
Cryptography in Three Slides (II)

- **Asymmetric encryption (e.g. RSA, PGP, ...)**:
  - the public key \((K_A)\) is given to all agents
  - the private key \((K_A^{-1})\) is only known by \(A\)
  - \(\text{decrypt}(\{m\}_{K_A}, K_A^{-1}) = m\)

- **Symmetric encryption (e.g. AES)**:
  - \(K_{AB} \xleftarrow{} K_{1AB}\)
  - \(\text{decrypt}(\{m\}_{K_{AB}} \xrightarrow{} K_{AB}) = m\)
Cryptography in Three Slides (II)

- Asymmetric encryption (e.g. RSA, PGP, ...):
  - the public key \( K_A \) is given to all agents
  - the private key \( K_A^{-1} \) is only known by \( A \)
  - \( decrypt(\{m\}_K_A, K_A^{-1}) = m = decrypt(\{m\}_{K_A^{-1}}, K_A) \)
Cryptography in Three Slides (II)

- Assymmetric encryption (e.g. RSA, PGP, . . . ) :
  - the public key ($K_A$) is given to all agents
  - the private key ($K_A^{-1}$) is only known by $A$
  - $decrypt(\{m\}_{K_A}, K_A^{-1}) = m = decrypt(\{m\}_{K_A^{-1}}, K_A)$

- Symmetric encryption (e.g. AES) :
Assymmetric encryption (e.g. RSA, PGP, ...) :

- the public key ($K_A$) is given to all agents
- the private key ($K^{-1}_A$) is only known by $A$
- $\text{decrypt}({m}_K, K^{-1}_A) = m = \text{decrypt}({m}_{K^{-1}_A}, K_A)$

Symmetric encryption (e.g. AES) :

- $K_{AB} \equiv K^{-1}_{AB}$
Cryptography in Three Slides (II)

- **Assymmetric encryption (e.g. RSA, PGP, ...) :**
  - the public key ($K_A$) is given to all agents
  - the private key ($K_A^{-1}$) is only known by $A$
  - $\text{decrypt}({m}_{K_A}, K_A^{-1}) = m = \text{decrypt}({m}_{K_A^{-1}}, K_A)$

- **Symmetric encryption (e.g. AES) :**
  - $K_{AB} \equiv K_{AB}^{-1}$
  - $\text{decrypt}({m}_{K_{AB}}, K_{AB}) = m$
Cryptography in Three Slides (—II)

Properties associated to cryptographic functions
Cryptography in Three Slides (—II)

Properties associated to cryptographic functions

- Confidentiality
  - \{“4976 0974 2373 7788”\}_{KB}
  - \{teurgoule_recipe.pdf\}_{K_{AB}}
Cryptography in Three Slides (—II)

Properties associated to cryptographic functions

- Confidentiality
  - \{“4976 0974 2373 7788”\}_{K_B}
  - \{teurgoule_recipe.pdf\}_{K_{AB}}

- Message authentication (Electronic signature)
  - “genet@irisa.fr”, \{ “genet@irisa.fr” \}_{K_{genet}^{-1}}
Cryptography in Three Slides (—II)

Properties associated to cryptographic functions

- Confidentiality
  
  - \( \{4976 \ 0974 \ 2373 \ 7788\}^K_B \)
  
  - \( \{\text{teurgoule_recipe.pdf}\}^K_{AB} \)

- Message authentication (Electronic signature)
  
  - “genet@irisa.fr”, \( \{\text{“genet@irisa.fr”}\}^{{K^{-1}}_{genet}} \)

- Message integrity
  
  - The content \( m \) of encrypted message \( \{m\}^K \) cannot be modified without knowing \( K \)
Properties of cryptographic protocols

Some properties:

- Secrecy
- Authentication of a message
- Authentication of an agent
- Freshness/anti-replay
- Contract signature
- Fairness
- Anonymity
- Unlinkability
- ...
Definitions

Definition (Secrecy)
A protocol ensures secrecy of messages if an intruder cannot (syntactically) deduce them. (a.k.a. "weak secrecy")

Definition (Message Authentication)
An agent $A$ authenticates a message $m$ if $A$ knows which agent has built $m$.

Definition (Agent Authentication)
An agent $A$ authenticates an agent $B$ if, after a successful session of the protocol, $A$ knows that he has run the protocol with $B$.

Definition (Freshness)
During a protocol session, a message is fresh if this message has specifically been built for this protocol session.
**Definitions**

**Définition (Secrecy)** A protocol ensures *secrecy of message* $s$ if an intruder cannot (syntactically) *deduce* $s$. (a.k.a. ”weak secrecy”)
Definitions

Définition (Secrecy) A protocol ensures secrecy of message $s$ if an intruder cannot (syntactically) deduce $s$. (a.k.a. ”weak secrecy”)

Définition (Message Authentication) An agent $A$ authenticates a message $m$ if $A$ knows which agent has built $m$. 
Definitions

Définition (Secrecy) A protocol ensures secrecy of message $s$ if an intruder cannot (syntactically) deduce $s$. (a.k.a. ”weak secrecy”)

Définition (Message Authentication) An agent $A$ authenticates a message $m$ if $A$ knows which agent has built $m$.

Définition (Agent Authentication) An agent $A$ authenticates an agent $B$ if, after a successful session of the protocol, $A$ knows that he has run the protocol with $B$. 
Définition *(Secrecy)* A protocol ensures *secrecy of message s* if an intruder cannot (syntactically) *deduce s.* (a.k.a. ”weak secrecy”)

Définition *(Message Authentication)* An agent $A$ *authenticates a message* $m$ if $A$ knows which agent has *built* $m$.

Définition *(Agent Authentication)* An agent $A$ *authenticates an agent* $B$ if, after a successful session of the protocol, $A$ knows that he has *run the protocol with* $B$.

Définition *(Freshness)* During a protocol session, a *message is fresh* if this message has *specifically been built for this protocol session*. 
Protocol Basic Principles (I)

\( B \) sends a secret \( s \) to \( A \)

### Protocol 0

1. \( B \leftrightarrow A : B, s \)
Protocol Basic Principles (I)

\( B \) sends a secret \( s \) to \( A \)

**Protocol 0**

1. \( B \leftrightarrow A : B, s \)

Easy to attack by an intruder:

1. \( B \leftrightarrow I(A) : B, s \)
Protocol Basic Principles (I)

\( B \) sends a secret \( s \) to \( A \)

---

**Protocol 0**

1. \( B \leftrightarrow A : B, s \)

   **Easy to attack by an intruder:** 1. \( B \leftrightarrow I(A) : B, s \)

---

**Protocol 1**

1. \( B \leftrightarrow A : \{B, s\}_{K_A} \)
Protocol Basic Principles (I)

B sends a secret \( s \) to A

**Protocol 0**

1. \( B \leftrightarrow A : B, s \)

Expected Properties:

- \( s \) is secret, but \( s \) cannot be authenticated!

Attacked by an intruder:

1. \( I(B), I(A): B, s \)

**Protocol 1**

1. \( B \leftrightarrow A : \{ B, s \}_K_A \)

If key \( K_A \) and \( \{ \}_K_A \) are robust, only A can read \( s \)
Protocol Basic Principles (I)

$B$ sends a secret $s$ to $A$

Protocol 0

1. $B \leftrightarrow A : B, s$

Easy to attack by an intruder:

1. $B \leftrightarrow I(A) : B, s$

Protocol 1

1. $B \leftrightarrow A : \{B, s\}_{K_A}$

If key $K_A$ and $\{\}_K$ are robust, only $A$ can read $s$

Expected Properties: $s$ is secret
Protocol Basic Principles (I)

B sends a secret $s$ to A

Protocol 0

1. $B \leftrightarrow A : B, s$

Easy to attack by an intruder: 1. $B \leftrightarrow I(A) : B, s$

Protocol 1

1. $B \leftrightarrow A : \{B, s\}_{K_A}$

If key $K_A$ and $\{\}^{K_A}$ are robust, only A can read $s$

**Expected Properties**: $s$ is secret, but $s$ cannot be authenticated!

T. Genet (IRISA)  Practice of SPAN+AVISPA
Protocol Basic Principles (I)

*B* sends a secret *s* to *A*

**Protocol 0**

1. \( B \leftrightarrow A : B, s \)

Easy to attack by an intruder : 1. \( B \leftrightarrow I(A) : B, s \)

**Protocol 1**

1. \( B \leftrightarrow A : \{B, s\}_K_A \)

If key \( K_A \) and \( \{\}_K_A \) are robust, only *A* can read *s*

**Expected Properties** : *s* is secret, but *s* cannot be authenticated!

Attacked by an intruder : 1. \( I(B) \leftrightarrow A : \{B, s_I\}_K_A \) \( (K_A \text{ is public}) \)

\( s_I \) is known by *I* and accepted by *A* as sent by *B*!
Protocol Basic Principles (II)

$B$ sends a secret $s$ to $A$

**Protocol 2**

1. $B \leftrightarrow A : B, \{s\}_{K_B^{-1}}$
Protocol Basic Principles (II)

$B$ sends a secret $s$ to $A$

**Protocol 2**

1. $B \leftrightarrow A : B, \{s\}_{K_B^{-1}}$

**Expected Properties** : $s$ is authenticated by $A$ as built by $B$
Protocol Basic Principles (II)

$B$ sends a secret $s$ to $A$

**Protocol 2**

1. $B \leftrightarrow A : B, \{s\}_{K_B^{-1}}$

**Expected Properties**: $s$ is authenticated by $A$ as built by $B$, $s$ not secret
Protocol Basic Principles (II)

$B$ sends a secret $s$ to $A$

Protocol 2

1. $B \leftrightarrow A : B, \{s\}_{K_B^{-1}}$

**Expected Properties:** $s$ is authenticated by $A$ as built by $B$, $s$ not secret

---

Attacked by an intruder:

1. $B \leftrightarrow I(A) : B, \{s\}_{K_B^{-1}}$ (for $K_B$ public)
Protocol Basic Principles (II)

*B* sends a secret *s* to *A*

### Protocol 2

1. \( B \leftrightarrow A : B, \{s\}_{K_B^{-1}} \)

**Expected Properties:** *s* is authenticated by *A* as built by *B*, *s* not secret

Attacked by an intruder:

1. \( B \leftrightarrow I(A) : B, \{s\}_{K_B^{-1}} \) \hspace{1cm} (\( K_B \) public)

### Protocol 3

1. \( B \leftrightarrow A : \{B, s, \{s\}_{K_B^{-1}}\}_K^A \)
Protocol Basic Principles (II)

*B* sends a secret *s* to *A*

### Protocol 2

1. \( B \leftrightarrow A : B, \{s\}_{K_B^{-1}} \)

**Expected Properties:** *s* is authenticated by *A* as built by *B*, *s* not secret

Attacked by an intruder: 1. \( B \leftrightarrow I(A) : B, \{s\}_{K_B^{-1}} \) (\( K_B \) public)

### Protocol 3

1. \( B \leftrightarrow A : \{B, s, \{s\}_{K_B^{-1}}\}_{K_A} \)

**Expected Properties:** *s* is secret, *s* is authenticated by *A* as built by *B*
Protocol Basic Principles (II)

$B$ sends a secret $s$ to $A$

### Protocol 2

1. $B \leftarrow A : B, \{s\}_{K_B^{-1}}$

**Expected Properties**: $s$ is authenticated by $A$ as built by $B$, $s$ not secret

---

Attacked by an intruder:

1. $B \leftarrow I(A) : B, \{s\}_{K_B^{-1}}$  \hspace{1cm} ($K_B$ public)

### Protocol 3

1. $B \leftarrow A : \{B, s, \{s\}_{K_B^{-1}}\}_{K_A}$

**Expected Properties**: $s$ is secret, $s$ is authenticated by $A$ as built by $B$

---

But I can make $A$ accept $s$ again, using message replay:

...3 years later

1. $I(B), \leftarrow A : \{B, s, \{s\}_{K_B^{-1}}\}_{K_A}$

$A$ does not know if the message is fresh! $A$ cannot authenticate $B$.
Protocol Basic Principles (II)

Protocol 2

1. \( B \leftrightarrow A : B, \{s\}_{K_B^{-1}} \)

Expected Properties: \( s \) is authenticated by \( A \) as built by \( B \), \( s \) not secret

Attacked by an intruder:

1. \( B \leftrightarrow I(A) : B, \{s\}_{K_B^{-1}} \) (\( K_B \) public)

Protocol 3

1. \( B \leftrightarrow A : \{B, s, \{s\}_{K_B^{-1}}\}_{K_A} \)

Expected Properties: \( s \) is secret, \( s \) is authenticated by \( A \) as built by \( B \)

But \( I \) can make \( A \) accept \( s \) again, using message replay:

1. \( B \leftrightarrow A : \{B, s, \{s\}_{K_B^{-1}}\}_{K_A} \)

... 3 years later

1. \( I(B) \leftrightarrow A : \{B, s, \{s\}_{K_B^{-1}}\}_{K_A} \)
Protocol Basic Principles (II)

B sends a secret $s$ to A

Protocol 2

1. $B \leftrightarrow A : B, \{s\}_{K_B^{-1}}$

Expected Properties: $s$ is authenticated by A as built by B, $s$ not secret

Attacked by an intruder: 1. $B \leftrightarrow I(A) : B, \{s\}_{K_B^{-1}}$ (for $K_B$ public)

Protocol 3

1. $B \leftrightarrow A : \{B, s, \{s\}_{K_B^{-1}}\}_{K_A}$

Expected Properties: $s$ is secret, $s$ is authenticated by A as built by B

But I can make A accept $s$ again, using message replay:

1. $B \leftrightarrow A : \{B, s, \{s\}_{K_B^{-1}}\}_{K_A}$  

... 3 years later

1. $I(B) \leftrightarrow A : \{B, s, \{s\}_{K_B^{-1}}\}_{K_A}$

A does not know if the message is fresh!  
A cannot authenticate B
Protocol Basic Principles (III)

$B$ sends a secret $s$ to $A$

Protocol 4 (secrecy, 2 authentications and freshness)
Protocol Basic Principles (III)

\( B \) sends a secret \( s \) to \( A \)

---

Protocol 4 (secrecy, 2 authentications and freshness)

\( A \) generates a fresh random number (a *nonce*) : \( N_A \)
Protocol Basic Principles (III)

B sends a secret s to A

Protocol 4 (secrecy, 2 authentications and freshness)

A generates a fresh random number (a nonce): $N_A$

1. $A \leftrightarrow B : \{A, B, N_A\}_{K_B}$

“Challenge” from A to B

Expected Properties:
- $s$ is secret
- $N_A$ is fresh, and so is $\{A \leftrightarrow B \leftrightarrow N_A \leftrightarrow s\}_{K_A}$
- $A$ authenticates $s$ as built by $B$
- $A$ authenticates $B$ during the session.
Protocol Basic Principles (III)

*B* sends a secret *s* to *A*

### Protocol 4 (secrecy, 2 authentications and freshness)

*A* generates a fresh random number (a *nonce*): \( N_A \)

1. \( A \leftrightarrow B : \{A, B, N_A\}_B \)  
   “Challenge” from *A* to *B*

2. \( B \leftrightarrow A : \{A, B, N_A, s\}_A \)  
   *B* answers to the challenge
Protocol Basic Principles (III)

*B* sends a secret *s* to *A*

---

### Protocol 4 (secrecy, 2 authentications and freshness)

*A* generates a fresh random number (a *nonce*): \( N_A \)

1. \( A \leftrightarrow B : \{A, B, N_A\}_K_B \) \hspace{1cm} \text{“Challenge” from } A \text{ to } B
2. \( B \leftrightarrow A : \{A, B, N_A, s\}_K_A \) \hspace{1cm} B answers to the challenge

---

**Expected Properties:**

- \( s \) is secret
Protocol Basic Principles (III)

$B$ sends a secret $s$ to $A$

### Protocol 4 (secrecy, 2 authentications and freshness)

$A$ generates a fresh random number (a *nonce*): $N_A$

1. $A \leftrightarrow B : \{A, B, N_A\}_{K_B}$
   “Challenge” from $A$ to $B$

2. $B \leftrightarrow A : \{A, B, N_A, s\}_{K_A}$
   $B$ answers to the challenge

#### Expected Properties:

- $s$ is secret
- $N_A$ is fresh, and so is $\{A, B, N_A, s\}_{K_A}$
Protocol Basic Principles (III)

$B$ sends a secret $s$ to $A$

Protocol 4 (secrecy, 2 authentications and freshness)

$A$ generates a fresh random number (a nonce): $N_A$

1. $A \leftrightarrow B : \{A, B, N_A\}^K_B$ \hfill \text{“Challenge” from $A$ to $B$}
2. $B \leftrightarrow A : \{A, B, N_A, s\}^K_A$ \hfill $B$ answers to the challenge

Expected Properties:

- $s$ is secret
- $N_A$ is fresh, and so is $\{A, B, N_A, s\}^K_A$
- $A$ authenticates $s$ as built by $B$
Protocol Basic Principles (III)

B sends a secret s to A

Protocol 4 (secrecy, 2 authentications and freshness)

A generates a fresh random number (a nonce): $N_A$

1. $A \leftrightarrow B : \{A, B, N_A\}_K_B$ \hspace{1cm} “Challenge” from A to B
2. $B \leftrightarrow A : \{A, B, N_A, s\}_K_A$ \hspace{1cm} B answers to the challenge

Expected Properties:

- $s$ is secret
- $N_A$ is fresh, and so is $\{A, B, N_A, s\}_K_A$
- A authenticates $s$ as built by $B$
- A authenticates $B$ during the session.
Protocol Basic Principles (III)

$B$ sends a secret $s$ to $A$

**Protocol 4 (secrecy, 2 authentications and freshness)**

$A$ generates a fresh random number (a nonce): $N_A$

1. $A \leftrightarrow B : \{A, B, N_A\}^K_B$ \quad “Challenge” from $A$ to $B$
2. $B \leftrightarrow A : \{A, B, N_A, s\}^K_A$ \quad $B$ answers to the challenge

**Expected Properties:**

- $s$ is secret
- $N_A$ is fresh, and so is $\{A, B, N_A, s\}^K_A$
- $A$ authenticates $s$ as built by $B$
- $A$ authenticates $B$ during the session.

**Protocol 4, with symmetric keys**

1. $A \leftrightarrow B : \{A, B, N_A\}^K_{AB}$ \quad “Challenge” de $A$ pour $B$
2. $B \leftrightarrow A : \{A, B, N_A, s\}^K_{AB}$ \quad Réponse de $B$ au challenge
Outline

1. Cryptography and Protocols Basics
2. What are we going to formalise and verify? The Dolev-Yao model
3. Basics of Protocol Formalization in SPAN+AVISPA
4. Defining a simple protocol with SPAN+AVISPA
5. Other examples and conclusion
Detect and Prevent Logical Attacks

Dolev-Yao model = Protocol logic (its mechanics)
Intruder capabilities

Assumption 1: Cryptography is secure
- The intruder cannot decrypt a message without the key
- The intruder cannot guess a secret key or a nonce

Assumption 2: The intruder has full control over the network
- The intruder can start any number of parallel protocol sessions
- The intruder knows all the public data of the protocol
- The intruder has all the privileges/keys of corrupted agents
- The intruder can read, store, block every sent message
- The intruder can build and send messages
- The intruder can compose/decompose messages
- The intruder can encrypt/decrypt if he has the key
Detect and Prevent Logical Attacks

**Dolev-Yao model** = Protocol logic (its mechanics)

<table>
<thead>
<tr>
<th>Intruder capabilities</th>
</tr>
</thead>
</table>

- **Assumption 1 : Cryptography is secure**
  - the intruder cannot decrypt a message without the key
  - the intruder cannot guess a secret key or a nonce

T. Genet (IRISA)  Practice of SPAN+AVISPA
Detect and Prevent Logical Attacks

Dolev-Yao model = Protocol logic (its mechanics) |
                      Intruder capabilities

- Assumption 1: Cryptography is secure
  - the intruder cannot decrypt a message without the key
  - the intruder cannot guess a secret key or a nonce

- Assumption 2: The intruder has a full control over the network
  - he can start any number of parallel protocol sessions
  - he knows all the public data of the protocol
  - he has all the privileges/keys of corrupted agents
  - he can read, store, block every sent message
  - he can build and send messages
  - he can compose/decompose messages
  - he can encrypt/decrypt if he has the key
## Logical Attacks: Limits of Manual Detection

### NSPK: Needham-Schroeder Public Key Protocol (1978)

<table>
<thead>
<tr>
<th>Step</th>
<th>Message</th>
<th>Authentication</th>
<th>Security Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>$A \leftrightarrow B : {A, N_A}_{K_B}$</td>
<td>Secrecy of $N_A$, $N_B$</td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td>$B \leftrightarrow A : {N_A, N_B}_{K_A}$</td>
<td>Messages 2 and 3 authenticated</td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>$A \leftrightarrow B : {N_B}_{K_B}$</td>
<td>Mutual Authentication of $A$ et $B$</td>
<td></td>
</tr>
</tbody>
</table>
**Logical Attacks : Limits of Manual Detection**

**NSPK : Needham-Schroeder Public Key protocol (1978)**

1. \( A \leftrightarrow B : \{A, N_A\}_{K_B} \)  
   - Secrecy of \( N_A, N_B \)

2. \( B \leftrightarrow A : \{N_A, N_B\}_{K_A} \)  
   - Messages 2 and 3 authenticated

3. \( A \leftrightarrow B : \{N_B\}_{K_B} \)  
   - Mutual Authentication of \( A \) et \( B \)

- Has resisted to manual analysis during 17 years
- Was proven safe several times (paper proofs)
- 1995 : using Model-Checking, Gawin Lowe found a Logical Attack cancelling all the security properties of the protocol
Outline

1. Cryptography and Protocols Basics
2. What are we going to formalise and verify? The Dolev-Yao model
3. Basics of Protocol Formalization in SPAN+AVISPA
4. Defining a simple protocol with SPAN+AVISPA
5. Other examples and conclusion
How to formalize a protocol in a verification tool?

« Alice&Bob » notation is too ambiguous to rigourously formalize

1. \( A \leftrightarrow B : \{ A, B, N_A \}^{K_B} \)
   
   A generates a random value \( N_A \)

2. \( B \leftrightarrow A : \{ A, B, N_A, s \}^{K_A} \)
   
   A checks that the received \( N_A \) corresponds to the one generated in 1.

All formal tools use languages overcoming those problems

- Description by role: process calculi (Proverif, AVISPA, Scyther)
- Description by rules: Horn clauses, rewriting rules (Tamarin)
Why using SPAN+AVISPA?

- AVISPA: cryptographic protocol verification tool
  European Project (2003-2005):
  Universities of Genova/Zurich/Nancy/Besançon and Siemens
  ⇒ good at finding and **explaining** Attacks
  ⇒ security proofs for finite scenarios (< Proverif, Tamarin, ...)

T. Genet (IRISA)
Why using SPAN+AVISPA?

- **AVISPA**: cryptographic protocol verification tool
  European Project (2003-2005):
  Universities of Genova/Zurich/Nancy/Besançon and Siemens
  ⇒ good at finding and **explaining** Attacks
  ⇒ security proofs for finite scenarios (Proverif, Tamarin, ...)

- **SPAN**: graphical animation layer over AVISPA
  ⇒ helps to define AVISPA protocol specifications
  ⇒ good for debugging AVISPA specifications
  ⇒ good for **graphical visualization** of attacks
Basics of HLPSL syntax for AVISPA

1. Protocol definition: roles and transition
2. Property definition:
   - secrecy : secrecy_of
   - message authentication : weak_authentication_on
   - agent authentication : authentication_on
3. Verification scenarios: session and composition
Outline

1. Cryptography and Protocols Basics

2. What are we going to formalise and verify? The Dolev-Yao model

3. Basics of Protocol Formalization in SPAN+AVISPA

4. Defining a simple protocol with SPAN+AVISPA

5. Other examples and conclusion
Defining a simple protocol with SPAN+AVISPA

Key exchange protocol

- Three agents: A, B and a Trusted Third Party T
- A and T share a symmetric key $K_{AT}$
- B and T share a symmetric key $K_{BT}$
- A wants to establish a symmetric session key $K_{AB}$ shared with B

![Diagram showing the key exchange protocol with A, T, and B connected by $K_{AT}$ and $K_{BT}$]
Defining a simple protocol with SPAN+AVISPA

Key exchange protocol

- Three agents: A, B and a Trusted Third Party T
- A and T share a symmetric key $K_{AT}$
- B and T share a symmetric key $K_{BT}$
- A wants to establish a symmetric session key $K_{AB}$ shared with B

![Diagram of the protocol]

?? $K_{AB}$
Defining a simple protocol with SPAN+AVISPA

Key exchange protocol

- Three agents: A, B and a Trusted Third Party T
- A and T share a symmetric key $K_{AT}$
- B and T share a symmetric key $K_{BT}$
- A wants to establish a symmetric session key $K_{AB}$ shared with B

Protocol development in SPAN+AVISPA

1. Specifying protocol and properties
2. Debugging specification using animation:
   - Find the blocking transition, monitor the variables
3. Attack discovery, strengthening the protocol
4. Tuning and optimizing the protocol
Outline

1. Cryptography and Protocols Basics
2. What are we going to formalise and verify? The Dolev-Yao model
3. Basics of Protocol Formalization in SPAN+AVISPA
4. Defining a simple protocol with SPAN+AVISPA
5. Other examples and conclusion