Chapter 2

An Introductory Example

We start with the well-known example of the so-called "Needham-Schroeder public-key protocol" [212], that has been designed in 1978 and for which an attack was found in 1996 by G. Lowe [193], using formal methods.

2.1 An Informal Description

The protocol is a so-called "mutual authentication protocol". Two parties A and B wish to agree on some value, *e.g.* they wish to establish a shared secret that they will use later for fast confidential communication. The parties A and B only use a public communication channel (for instance a postal service, Internet or a mobile phone). The transport of the messages on such channels is insecure. Indeed, a malicious agent might intercept the letter (resp. message) look at its content and possibly replace it with another message or even simply destroy it.

In order to secure their communication, the agents use lockers (or encryption). We consider here public-key encryption: the lockers can be reproduced and distributed, but the key to open them is owned by a single person. Encrypting a message m with the public key of A is written $\{m\}_{\mathsf{pk}(A)}$ whereas concatenating two messages m_1 and m_2 is written $\langle m_1, m_2\rangle$. An informal description of the protocol in the so-called Alice-Bob notation is given in Figure 1.1.

$$\begin{array}{ll}
1.\ A \to B: & \{\langle A, N_A\rangle\}_{\mathsf{pk}(B)} \\
2.\ B \to A: & \{\langle N_A, N_B\rangle\}_{\mathsf{pk}(A)} \\
3.\ A \to B: & \{N_B\}_{\mathsf{pk}(B)}
\end{array}$$

Figure 2.1: Informal description of the Needham-Schroeder public key protocol

Description. First the agent A encrypts a nonce N_A, *i.e.* a random number freshly generated, and her identity with the public key of B and sends it on the public channel (message 1). Only the agent B, who owes the corresponding private key can open this message. Upon reception, he gets N_A, generates his own nonce N_B and sends back the pair encrypted with the public key of A (message 2). Only the agent A is able to open this message. Furthermore, since only B was able to get N_A, inserting N_A in the plaintext is a witness that it comes from the agent B. Finally, A, after decrypting, checks that the first component is N_A and retrieves the second component N_B. As an acknowledgement, she sends back N_B encrypted by the public key of B (message 3). When B receives this message, he checks that the content is N_B. If this succeeds, it is claimed that, if the agents A and B are honest, then both parties agreed on the nonces N_A and N_B (they share these values). Moreover, these values are secret: they are only known by the agents A and B.

$$\begin{array}{llll}
1.\ A \to D: & \{\langle A, N_A\rangle\}_{\mathsf{pk}(D)} & & \\
 & & 1'.\ D(A) \to B: & \{\langle A, N_A\rangle\}_{\mathsf{pk}(B)} \\
 & & 2'.\ B \to A: & \{\langle N_A, N_B\rangle\}_{\mathsf{pk}(A)} \\
2.\ B \to A: & \{\langle N_A, N_B\rangle\}_{\mathsf{pk}(A)} & & \\
3.\ A \to D: & \{N_B\}_{\mathsf{pk}(D)} & & \\
 & & 3'.\ D(A) \to B: & \{N_B\}_{\mathsf{pk}(B)}
\end{array}$$

Figure 2.2: Attack on the Needham-Schroeder public key protocol

Attack. Actually, an attack was found in 1996 by G. Lowe [193] on the Needham-Schroeder public-key protocol. The attack described in Figure 1.2 relies on the fact that the protocol can be used by several parties. Moreover, we have to assume that an honest agent A starts a session of the protocol with a dishonest agent D (message 1). Then D, impersonating A, sends a message to B, starting another instance of the protocol (message 1′). When B receives this message, supposedly coming from A, he answers (messages 2′ & 2). The agent A believes this reply comes from C, hence she continues the protocol (message 3). Now, the dishonest agent D decrypts the ciphertext and learn the nonce N_B. Finally, D is able to send the expected reply to B (message 3′). At this stage, two instances of the protocol have been completed with success. In the second instance B believes that he is communicating with A: contrarily to what is expected, A and B do not agree on N_B. Moreover, N_B is not a secret shared only between A and B.

Fixed version of the protocol. It has been proposed to fix the protocol by including the respondent's identity in the response (see Figure 1.3).

$$\begin{array}{ll}
1.\ A \to B: & \{\langle A, N_A\rangle\}_{\mathsf{pk}(B)} \\
2.\ B \to A: & \{\langle\langle N_A, N_B\rangle, B\rangle\}_{\mathsf{pk}(A)} \\
3.\ A \to B: & \{N_B\}_{\mathsf{pk}(B)}
\end{array}$$

Figure 2.3: Description of the Needham-Schroeder-Lowe protocol

The attack described above cannot be mounted in the corrected version of the protocol. Actually, it is reported in [193] that the technique that permitted to find the Lowe attack on the Needham-Schroeder public key protocol found no attack on this protocol.

2.2 A More Formal Analysis

The Alice-Bob notation is a semi-formal notation that specifies the conversation between the agents. We have to make more precise the view of each agent. This amounts specifying the concurrent programs that are executed by each party. One has also to be precise when specifying how a message is processed by an agent. In particular, what parts of a received message are checked by the agent? What are the actions performed by the agent to compute the answer?

A classical way to model protocols is to use a process algebra. However, in order to model the messages that are exchanged, we need a process algebra that allows processes to send first-order terms build over a signature, names and variables. These terms model messages that are exchanged during a protocol.

Example 2.1 *Consider for example the signature* $\Sigma = \{\{\_\}_{\_}, \mathsf{pk}\_, \mathsf{sk}(\_), \mathsf{dec}, \langle\_,\_\rangle, \mathsf{proj}_1, \mathsf{proj}_2\}$ *which contains three binary function symbols modelling asymmetric encryption, decryption, and pairing, and four unary function symbols modelling projections, public key and private key. The* *signature is equipped with an equational theory and we interpret equality up to this theory. For instance the theory*

$$\mathsf{dec}(\{x\}_{\mathsf{pk}(y)}, \mathsf{sk}(y)) = x, \ \mathsf{proj}_1(\langle x_1, x_2\rangle) = x_1, \ \textit{and}\ \mathsf{proj}_2(\langle x_1, x_2\rangle) = x_2.$$

models that decryption and encryption cancel out whenever suitable keys are used. One can also retrieves the first (resp. second) component of a pair.

Processes $P, Q, R, \ldots$ are constructed as follows. The process $\mathsf{new}\ N.P$ restricts the name N in P and can for instance be used to model that N is a fresh random number. $\mathsf{in}(c, x).P$ models the input of a term on a channel c, which is then substituted for x in process P. $\mathsf{out}(c, t)$ outputs a term t on a channel c. The conditional $\mathsf{if}\ M = N\ \mathsf{then}\ P\ \mathsf{else}\ Q$ behaves as P when M and N are equal modulo the equational theory and behaves as Q otherwise.

The program (or process) that is executed by an agent, say a, who wants to initiate a session of the Needham-Schroeder protocol with another agent b is as follows:

$A(a,b)$	$\hat{=}$	$\mathsf{new}\ N_a.$	a `generates a fresh message` N_a
		$\mathsf{out}(c, \{a, N_a\}_{\mathsf{pk}(b)}).$	`the message is sent on the channel` c
		$\mathsf{in}(c, x).$	a `is waiting for an input on` c
		$\mathsf{let}\ x_0 = \mathsf{dec}(x, \mathsf{sk}(a))\ \mathsf{in}$	a `tries to decrypt the message`
		$\mathsf{if}\ \mathsf{proj}_1(x_0) = N_a\ \mathsf{then}$	a `checks that the first component is` N_a
		$\mathsf{let}\ x_1 = \mathsf{proj}_2(x_0)\ \mathsf{in}$	a `retrieves the second component`
		$\mathsf{out}(c, \{x_1\}_{\mathsf{pk}(b)})$	a `sends her answer on` c

Note that we use variables for the unknown components of messages. These variables can be (a priori) replaced by any message, provided that the attacker can build it and that it is accepted by the agent. In the program described above, if the decryption fails or if the first component of the message received by a is not equal to N_a, then a will abort the protocol.

Similarly, we have to write the program that is executed by an agent, say b, who has to answer to the messages sent by the initiator of the protocol. This program may look like this:

$B(a,b)$	$\hat{=}$	$\mathsf{in}(c, y).$	b `is waiting for an input on` c
		$\mathsf{let}\ (a, y_0) = \mathsf{dec}(y, \mathsf{sk}(b))\ \mathsf{in}$	b `tries to decrypt it and then retrieves the second component of the plaintext`
		$\mathsf{new}\ N_b.$	b `generated a fresh random number` N_b
		$\mathsf{out}(c, \{y_0, N_b\}_{\mathsf{pk}(a)}).$	b `sends his answer on the channel` c
		$\mathsf{in}(c, y').$	b `is waiting for an input on` c
		$\mathsf{if}\ \mathsf{dec}(y', \mathsf{sk}(b)) = N_b\ \mathsf{then}\ \mathsf{Ok}.$	b `tries to decrypt the message and he checks whether its content is` N_b `or not`

The (weak) secrecy property states for instance that, if a, b are honest (their secret keys are unknown to the environment), then, when the process $B(a,b)$ reaches the Ok state, N_b is unknown to the environment. We will also see later how to formalise agreement properties. The "environment knowledge" is actually a component of the description of the global state of the network. Basically, all messages that can be built from the public data and the messages that have been sent are in the knowledge of the environment.

Any number of copies of A and B (with any parameter values) are running concurrently in a hostile environment. Such a hostile environment is modelled by any process that may receive and emit on public channels. We also assume that such an environment owes as many public/private key pairs as it wishes (compromised agents), an agent may also generate new values when needed. The only restrictions on the environment is on the way it may construct new messages: the encryption and decryption functions, as well as public keys are assumed to be known from the environment. However no private key (besides those that it generates) is known. We exhibit now a process that will yield the attack, assuming that the agent d is a dishonest (or compromised) agent who leaked his secret key:

$P \mathrel{\hat{=}}$	$\mathsf{in}(c, z_1).$	d `receives a message (from` a`)`
	$\mathsf{let}\ \langle a, z_1' \rangle = \mathsf{dec}(z_1, \mathsf{sk}(d))\ \mathsf{in}$	d `decrypts it`
	$\mathsf{out}(c, \{\langle a, z_1' \rangle\}_{\mathsf{pk}(b)}).$	d `sends the plaintext encrypted with` $\mathsf{pk}(b)$
	$\mathsf{in}(c, z_2).\mathsf{out}(c, z_2).$	d `forwards to` a `the answer he obtained from` b
	$\mathsf{in}(c, z_3).$	d `receives the answer from` a
	$\mathsf{let}\ z_3' = \mathsf{dec}(z_3, \mathsf{sk}(d))\ \mathsf{in}$	d `decrypts it and learn` N_b
	$\mathsf{out}(c, \{z_3'\}_{\mathsf{pk}(b)}).$	d `sends the expected message` $\{N_b\}_{\mathsf{pk}(b)}$ `to` b.

The Needham-Schroeder-Lowe protocol has been proved secure in several formal models close to the one we have sketched in this section [72, 24].

2.3 An Attack on the Fixed Version of the Protocol

Up to now, the encryption is a black-box: nothing can be learnt on a plaintext from a ciphertext and two ciphertexts are unrelated.

Consider however a simple El-Gamal encryption scheme. Roughly (we skip here the group choice for instance), the encryption scheme is given by a cyclic group G of order q and generator g; these parameters are public. Each agent a may choose randomly a secret key $\mathsf{sk}(a)$ and publish the corresponding public key $\mathsf{pk}(a) = g^{\mathsf{sk}(a)}$. Given a message m (assume for simplicity that it is an element $g^{m'}$ of the group), encrypting m with the public key $\mathsf{pk}(a)$ consists in drawing a random number r and letting $\{m\}_{\mathsf{pk}(a)} = (\mathsf{pk}(a)^r \times g^{m'}, g^r)$. Decrypting the message consists in raising g^r to the power $\mathsf{sk}(a)$ and dividing the first component of the pair by $g^{r\times\mathsf{sk}(a)}$. We have that:

$$[\mathsf{pk}(a)^r \times g^{m'}]/(g^r)^{\mathsf{sk}(a)} = [(g^{\mathsf{sk}(a)})^r \times g^{m'}]/(g^r)^{\mathsf{sk}(a)} = g^{m'} = m.$$

This means that this encryption scheme satisfies the equation $\mathsf{dec}(\{x\}_{\mathsf{pk}(y)}, \mathsf{sk}(y)) = x$. However, as we will see, this encryption scheme also satisfies some other properties that are not taken into account in our previous formal analysis.

Attack. Assume now that we are using such an encryption scheme in the Needham-Schroeder-Lowe protocol and that pairing two group elements $m_1 = g^{m_1'}$ and $m_2 = g^{m_2'}$ is performed in a naive way: $\langle m_1, m_2 \rangle$ is mapped to $g^{m_1' + 2^{|m_1'|} \times m_2'}$ (*i.e.* concatenating the binary representations of the messages m_1' and m_2'). In such a case, an attack can be mounted on the protocol (see Figure 1.4).

Actually, the attack starts as before. We assume that the honest agent a is starting a session with a dishonest party d. Then d decrypts the message and re-encrypt it with the public key of b. The honest party b replies sending the expected message $\{\langle\langle N_a, N_b\rangle, b\rangle\}_{\mathsf{pk}(a)}$. The attacker intercepts this message. Note that the attacker can not simply forward it to a since it does not have the expected form. The attacker intercepts $\{\langle\langle N_a, N_b\rangle, b\rangle\}_{\mathsf{pk}(a)}$, *i.e.* $(\mathsf{pk}(a)^r \times g^{N_a + 2^\alpha \times N_b + 2^{2\alpha} \times b}, g^r)$ where α is the length of a nonce. The attacker knows g, α, b, hence he can compute $g^{-2^{2\alpha} \times b} \times g^{2^{2\alpha} \times d}$ and multiply the first component, yielding $\{\langle\langle N_a, N_b\rangle, d\rangle\}_{\mathsf{pk}(a)}$. Then the attack can go on as before: a replies by sending $\{N_b\}_{\mathsf{pk}(d)}$ and the attacker sends $\{N_b\}_{\mathsf{pk}(b)}$ to b, impersonating a.

This example is however a toy example since pairing could be implemented in another way. In [251] there is a real attack that is only based on weaknesses of the El Gamal encryption scheme. In particular, the attack does not dependent on how pairing is implemented.

1. $a \to d : \{\langle a, N_a \rangle\}_{\mathsf{pk}(d)}$
   - 1′. $d(a) \to b : \{\langle a, N_a \rangle\}_{\mathsf{pk}(b)}$
   - 2′. $b \to a : \{\langle\langle N_a, N_b\rangle, b\rangle\}_{\mathsf{pk}(a)} = (g^{N_a + 2^\alpha \times N_b + 2^{2\alpha} \times b} \times \mathsf{pk}(a)^r, g^r)$

d intercepts this message, and computes

$$[g^{N_a + 2^\alpha \times N_b + 2^{2\alpha} \times b} \times \mathsf{pk}(a)^r] \times g^{-2^{2\alpha} \times b} \times g^{2^{2\alpha} \times d} = g^{N_a + 2^\alpha \times N_b + 2^{2\alpha} \times d} \times \mathsf{pk}(a)^r$$

2. $d \to a : \{\langle\langle N_a, N_b\rangle, d\rangle\}_{\mathsf{pk}(a)} = (g^{N_a + 2^\alpha \times N_b + 2^{2\alpha} \times d} \times \mathsf{pk}(a)^r, g^r)$
3. $a \to d : \{N_b\}_{\mathsf{pk}(d)}$
   - 3′. $d \to b : \{N_b\}_{\mathsf{pk}(d)}$

Figure 2.4: Attack on the Needham-Schroeder-Lowe protocol with El-Gamal encryption.

This shows that the formal analysis only proves the security in a formal model, that might not be faithful. Here, the formal analysis assumed a model in which it is not possible to forge a ciphertext from another ciphertext, without decrypting/encrypting. This property is known as *non-malleability*, which is not satisfied by the El Gamal encryption scheme.

2.4 Further Readings

A survey by Clark and Jacob [106] describes several authentication protocols and outlines also the methods that have been used to analyse them. In addition, it provides a summary of the ways in which protocols have been found to fail. The purpose of the SPORE web page [1] is to continue on-line the seminal work of Clark and Jacob, updating their base of security protocols.

As you have seen, some protocols (or some attacks) rely on some algebraic properties of cryptographic primitives. In [122], a list of some relevant algebraic properties of cryptographic operators is given, and for each of them, some examples of protocols or attacks using these properties are provided. The survey also gives an overview of the existing methods in formal approaches for analysing cryptographic protocols.

2.5 Exercises

Exercice 1 ($\star$)
Consider the following protocol:

$$\begin{array}{ll} A \to B : & \langle A, \{K\}_{\mathsf{pk}(B)} \rangle \\ B \to A : & \langle B, \{K\}_{\mathsf{pk}(A)} \rangle \end{array}$$

First, A generates a fresh key K and sends it encrypted with the public key of B. Only B will be able to decrypt this message. In this way, B learns K and B also knows that this message comes from A as indicated in the first part of the message he received. Hence, B answers to A by sending again the key, this time encrypted with the public key of A.

Show that an attacker can learn the key K generated by an honest agent A to another honest agent B.

Exercice 2 ($\star$)
The previous protocol is corrected as in the Needham-Schroeder protocol, *i.e.* we add the identity of the agent inside each encryption.

$$\begin{array}{ll} A \to B : & \{\langle A, K \rangle\}_{\mathsf{pk}(B)} \\ B \to A : & \{\langle B, K \rangle\}_{\mathsf{pk}(A)} \end{array}$$

1. Check that the previous attack does not exist anymore. Do you think that the secrecy property stated in Exercise 1 holds?

2. Two agents want to use this protocol to establish a session key. Show that there is an attack.

Exercice 3 ($\star\star$)
For double security, all messages in the previous protocol are encrypted twice:

$$\begin{aligned} A \to B: &\quad \{\langle A, \{K\}_{\mathsf{pk}(B)}\rangle\}_{\mathsf{pk}(B)} \\ B \to A: &\quad \{\langle B, \{K\}_{\mathsf{pk}(A)}\rangle\}_{\mathsf{pk}(A)} \end{aligned}$$

Show that the protocol then becomes insecure in the sense that an attacker can learn the key K generated by an honest agent A to another honest agent B.

Exercice 4 ($\star\star\star$)
We consider a variant of the Needham-Schroeder-Lowe protocol. This protocol is as follows:

$$\begin{aligned} 1.\ A \to B: &\ \{\langle A, N_A\rangle\}_{\mathsf{pk}(B)} \\ 2.\ B \to A: &\ \{\langle N_A, \langle N_B, B\rangle\rangle\}_{\mathsf{pk}(A)} \\ 3.\ A \to B: &\ \{N_B\}_{\mathsf{pk}(B)} \end{aligned}$$

1. Check that the 'man-in-the-middle' attack described in Figure 1.2 does not exist.

2. Show that there is an attack on the secrecy of the nonce N_b.
   hint: type confusion

3. Do you think that this attack is realistic? Why?