Lab - Designing security protocols:
Understanding the challenges and benefits of validation tools through practice[‡]

Saïd Ider, Maryline Laurent, Maktoum Gaba, Alan Van Trigt
SAMOVAR, Télécom SudParis, Institut Polytechnique de Paris
91120 Palaiseau, France

2026

Abstract

In an increasingly connected world that is vulnerable to cyber attacks, communications security has become a major issue, and security protocols are an essential component. The very first step toward using a secure security protocol on the Internet is to design such a protocol and ensure that it meets the expected security properties. This step is crucial, because a poorly designed protocol will inevitably lead to the deployment of a vulnerable protocol on the Internet and attacks such as identity theft and man-in-the-middle attacks.

In this practical lab, you will discover the challenges and difficulties involved in designing a cryptographic protocol (or security protocol) to protect digital exchanges, ensuring the confidentiality of messages, authentication of message origin, and mutual authentication of entities. It will enable you to independently consolidate your knowledge of applied cryptography, understand in particular the importance of nonces and their integration with encryption mechanisms to build a protocol, and learn how to detect the flaws in a security protocol at a basic level. You will use the SPAN & AVISPA ("a Security Protocol ANimator for AVISPA") tool for formal verification of security protocols. Using the SPAN graphical interface, you will be able to visualize the flaws in a security protocol that could be exploited by an attacker.

This lab is intended for bachelor's or master's level students with some knowledge of computer science and basic security concepts. It requires downloading a 740MB virtual machine (VM). It is designed to be completed in 3 hours.

1 Pedagogical objectives

The objectives are the following:

- Identify basic attack and defense methods in cryptographic protocols,
- Distinguish between symmetric and asymmetric cryptography and understand their contribution to the construction of a security protocol,
- Understand the importance of nonces and encryption mechanisms in the design of security protocols, as well as their combined integration into a protocol,
- Know how to detect, at a basic level, the flaws in a security protocol,
- Understand the crucial issues involved in designing a security protocol,
- Understand the benefits and limitations of formal verification tools such as SPAN & AVISPA.

2 Organization of the practical work and your mission

The lab includes the following steps:

- Risks associated with transmitting a secret in plain text over a communication channel,
- Symmetric encryption for confidential transmission of the secret,

[*]Lab Practical training conducted as part of the TCE (Train-Cyber-Expert) project under the Cybersecurity PEPR, with financial support from France2030

- Asymmetric encryption for confidential transmission of the secret,
- Asymmetric encryption to authenticate the origin of a message, i.e., to ensure that a message comes from the declared sender,
- Adding nonces to messages to prevent replay attacks and ensure, through asymmetric encryption, one-way authentication (of a single interlocutor) and mutual authentication (of both interlocutors),
- Analysis of the Needham-Schroeder protocol (NSPK), which enables mutual authentication and is based on the confidential exchange of nonces,
- Analysis of the Man in the Middle attack on the NSPK protocol and its solution (Lowe),
- Compilation of the methods seen previously into a formal protocol to ensure mutual authentication.

Your mission is to help the protagonists of the lab - Alice and Bob - design the most secure cryptographic protocol possible, i.e., one that satisfies one or more classic security properties: confidentiality of a secret and mutual authentication.

3 A few words about the SPAN & AVISPA tools used in the lab and their limitations

SPAN & AVISPA integrates several formal verification tools to verify that a cryptographic protocol satisfies the expected security properties. In this case, we will only use the ATSE tool in the practical assignment. If you ever have to perform this verification yourself, it is good practice to first model your cryptographic protocol in a formal language. The lab assignment uses the hlpsl ("High-Level Protocol Specification Language") language. In the formal specification obtained, you must mention the security properties you wish to verify, for example, the confidentiality of the transmitted secret. The tool then checks whether the protocol as modeled satisfies the specified properties. If it detects a possible attack by an attacker called "Intruder" in the tool, the SPAN tool will allow you to visualize the attack. The attacker in the tool has full control over the communication channel. They can eavesdrop on the channel, hijack messages, modify them, replay them, etc. This is a Dolev-Yao type attacker.

Please note! Just because the tool does not detect any attacks does not necessarily mean that your protocol satisfies the specified properties. It simply means that the tool has not detected any attacks.

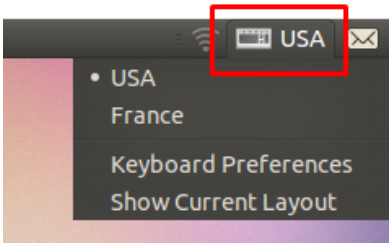
For more comprehensive documentation on using SPAN, please refer to [1]. More in-depth documentation on the concepts is available here [2].

4 Launching the lab

This lab uses a virtual machine (VM) in OVA format.
Steps if you do not have a virtualization tool:

1. Install the latest version of VirtualBox that is compatible with your computer `https://www.virtualbox.org/wiki/Downloads`
2. Launch VirtualBox and click Import.
3. Select the previously downloaded VM.
4. A screen allowing you to modify the VM configuration will appear. Click Next, then Start.
5. If the installation is successful (ignore the error message), you should see the login screen and be able to access the desktop, which includes, among other things, a SPAN executable and a TP folder.

To change the keyboard layout: At the top right of the VM, you can choose the keyboard layout, which defaults to QWERTY (US) or AZERTY (France). See the image below.



5 Notations

The notations used in the lab to describe security protocols are shown in Table 1. The AVISPA notation refers to the notations used in the hspsl files provided in the VM. The formal notation refers to the notations used in the lab instructions.

Table 1: Summary table of concepts and notations

Notion	**Formal notation**	**AVISPA notation**
A participant identity, Alice	A or a	A or a
B participant identity, Bob	B or b	B or b
Attacker's (or intruder's) identity	I or i	I or i
Secret	S	S, S' or $S1$
Sending message M from A to B	$A \rightarrow B : M$	$SND(M)$
Receiving message M from A by B	$A \rightarrow B : M$	$RCV(M)$
Concatenation of a word W to message M	$M.W$	$M.W$
Encrypting S with key K	$\{S\}_K$	$\{S\}\_K$
Encrypting S with public key of A	$\{S\}_{Ka}$	$\{S\}\_Ka$
Encrypting S with private key of A	$\{S\}_{Ka^{-1}}$	$\{S\}\_inv(Ka)$
Nonce generated by A	Na	Na or Na'
Nonce generated by B	Nb	Nb or Nb'

6 What a clear-sighted idea to send a secret in plain text!

Alice and Bob want to exchange secrets over the Internet, and they choose a communication channel that allows them to exchange messages in a super simple, fast, and free way! It's so simple, fast, and advanced that it's called NSA (for "Network Super Advanced"). We don't really know who (or which government agency) is behind this channel, and that's the problem.

Alice and Bob, wanting to brag about having been able to test this brand new channel before anyone else, decide that this is the perfect opportunity to exchange little secrets without anyone knowing!

The protocol implemented in this NSA channel is as follows:

$$A \rightarrow B : S.A$$

which means that A sends B a secret S concatenated with its identity A.

Sending the sender's identity A in the message allows the recipient to know who sent the message and to be able to reply to the right person.

Question 1) What is the risk of sending an unencrypted message in a communication channel? [1 correct answer]

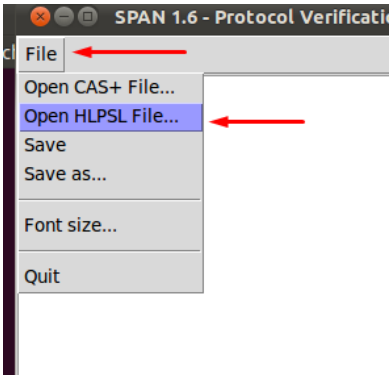
(a) People for whom the message is not intended will be able to read its content.

(b) The message may arrive too quickly because no unnecessary processing has been performed.

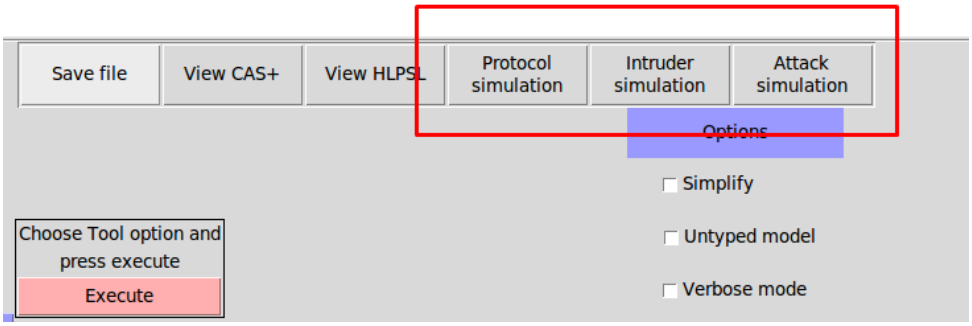
(c) None, it is well known that any system claiming to be FAIL-SAFE has never experienced any failures...

In the virtual machine (VM), go to the desktop and double-click on the "TP" folder. There you will see the 10 hlpsl files that will be used during the lab and that you will analyze with the SPAN software.

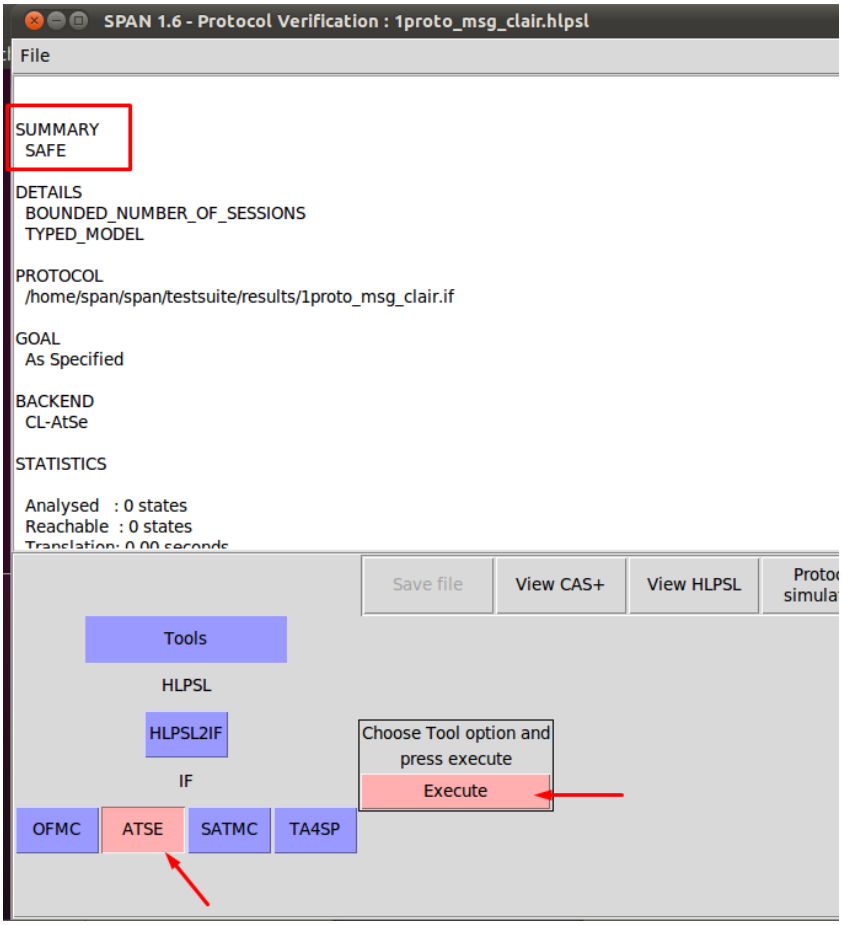
Launch SPAN by double-clicking on it from the desktop. In File→Open HLPSL File...→Desktop→TP, double-click on $1proto\_msg\_clair.hlpsl$ to open the file from SPAN.



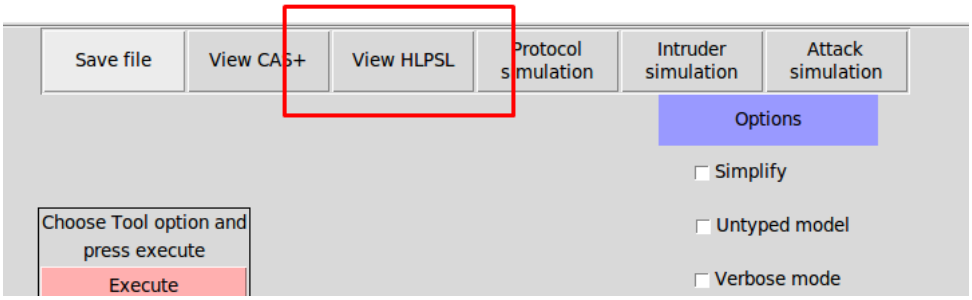
By clicking on Protocol Simulation, Intruder Simulation, or Attack Simulation, you can see the frames of the protocol sequence between legitimate participants, legitimate participants and the intruder, and the sequence of the attack detected by SPAN, respectively.



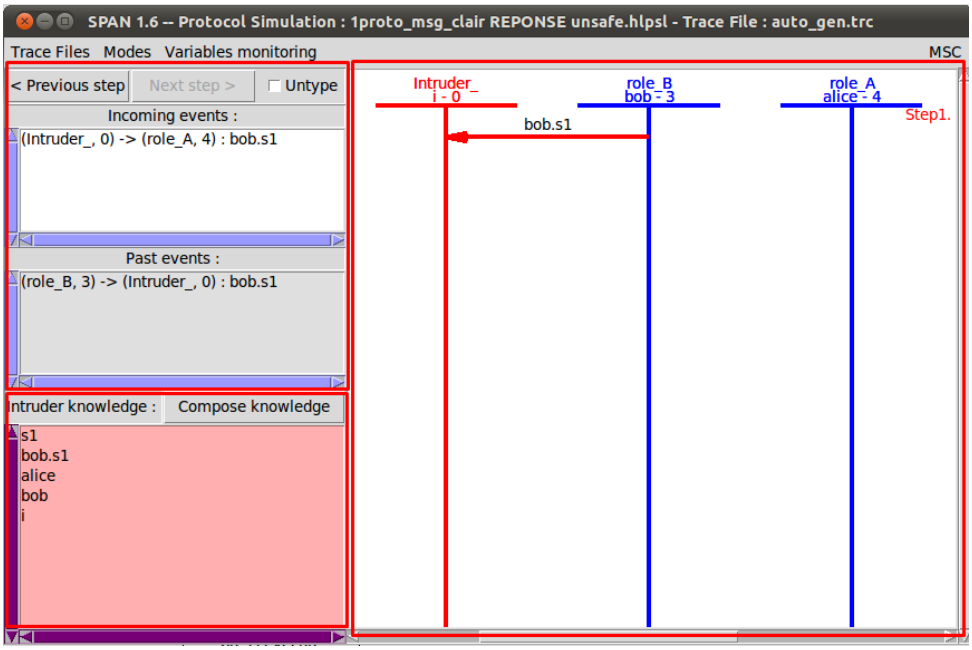
When analyzing the "msg clair" protocol with SPAN, click on ATSE in the lower left corner, then EXECUTE. You should see the result "SAFE" in SUMMARY in the upper frame, which corresponds to the verification result... This seems counterintuitive... The reason is simple and you will find it below: we need to introduce the properties that the protocol is supposed to satisfy and define the secret to be protected in the hlpsl file.



Return to the Desktop, then to TP, double-click on the file 1*proto_msg_clair.hlpsl* to open it with a text editor. In the file, uncomment by deleting the "%" in front of lines 29 "/\secret(S,sec_1,A,B)" to define what is secret in the protocol, and at the end of the file, lines 59, 60, and 61 to specify the secret to be protected and thus verify the confidentiality properties. The current line number is indicated at the bottom right of the text editor. Save and return to SPAN.
In SPAN, click View HLPSL to refresh the changes.



Click on ATSE then EXECUTE to analyze the protocol. The expected result in SUMMARY is UNSAFE. Click on Attack Simulation to display the attack frame identified by SPAN in a new window. Feel free to enlarge the window to clearly view all exchanges between all actors.



On the left, you will see the actions that can be performed and those that have already been performed in the protocol. You can click on a line in the "Incoming events" section or on the "< Previous step" button. On the right side of the window, you will see the attack frame identified by SPAN. This section is completed at the bottom left by the Intruder Knowledge section, which indicates all the elements known to the intruder at time t (here, the secret $s1$, the message *bob*.$s1$ sent, the identities of A and B *alice* and *bob*, as well as their own identity i).

Question 2) Describe the attack identified by SPAN. [1 correct answer]

(a) The intruder pretends to be Bob to Alice and can thus read the secret $s1$ sent by Alice.

(b) The intruder performs a replay attack.

(c) Despite the attack, the intruder does not have access to the content of the message, so they cannot read the secret.

(d) The intruder intercepts the secret $s1$ and can directly read its content.

To exchange secret information that only legitimate individuals can interpret, messages must be encrypted to ensure their confidentiality.

7 Symmetric encryption is wonderful (in theory)

After this harsh reality check, Alice no longer feels like she's in Wonderland... She never imagined that there could be malicious people trying to interfere with her super communication channel, "Network Super Advanced".

After regaining her composure, she remembers that there is a well-known family of encryption algorithms that is considered UNFAILING (in theory): symmetric encryption! She places a lot of hope in this technique.

Alice explains to Bob the simple principle of symmetric encryption, which also has the advantage of being computationally inexpensive: "To encrypt a secret message, you just need to randomly generate a cryptographic key that will be used to both encrypt and decrypt the message. This means we only need to communicate with people who have the symmetric key, and only those people will be able to decrypt the messages!"

Protocol summary: Participants hold symmetric keys before the communication channel is used. The sender sends their identity and the secret, encrypted using the symmetric key, to the recipient. Upon receipt, the recipient decrypts everything with the same symmetric key to read the content. Here is the result in scientific notation:

$$B \rightarrow A : \{B.S\}_K$$

From SPAN, open the file *2proto_symetrique.hlpsl*: in File→Open HLPSL File...→Desktop→TP and double-click on *2proto_symetrique.hlpsl*.
The protocol uses a symmetric key that Bob will use to encrypt the message and Alice will use to decrypt the message.
Click ATSE then EXECUTE to analyze the protocol. The expected result in SUMMARY is SAFE.

Despite the undeniable theoretical advantages of symmetric encryption, you explain to Alice that very few modern security protocols rely exclusively on symmetric cryptography, mainly because of practical considerations.

Question 3) Which of the following statements are true for the protocol defined so far with symmetric encryption? [2 correct answers]

(a) If a participant wishes to be added to the network, the other participants can send them their symmetric keys directly on the channel.

(b) If a participant loses their keys, it is no longer possible to communicate with them without publicly revealing the symmetric keys on the channel.

(c) It is impossible to add new participants without using other channels to share symmetric keys confidentially.

(d) If a participant loses their symmetric keys, they can retrieve them using mnemonic means.

(e) If a participant loses their symmetric keys, they simply need to decrypt the messages sent by brute force.

A good practice when using symmetric encryption keys is to avoid using the same key multiple times with different participants in order to limit the risk of the key being compromised and to limit the damage. In particular, if a key is stolen without the knowledge of both participants, the intruder will be able to decrypt messages that have already been exchanged (past messages) as well as future messages, leading to a loss of communication confidentiality. It is therefore a good idea to generate unique keys for each interlocutor, which can quickly become costly in terms of memory resources.

Question 4) In a system with n participants, if a single symmetric key must be stored for each participant by all participants, how many keys must be stored? [2 correct answers]

(a) Each participant stores only their symmetric key.

(b) Each participant must store $(n-1)$ keys.

(c) Among all participants, the total number of saved keys amounts to $n \times (n-1)$.

(d) Among all participants, the total number of saved keys amounts to n.

Since 1963, to limit tensions during the Cold War, the presidents of the United States and the USSR had a secret direct telephone line between Moscow and Washington: the famous red telephone. At the time, the encryption used was symmetric and the keys were single-use during conversations. Each call consumed keys

that had to be renewed, synchronized, and shared via diplomatic bags transported between the two nations.
The main communication channel, which was symmetrically encrypted, was therefore the red telephone line, but in order for it to function and guarantee security, an auxiliary channel had to be used for the exchange of keys. This auxiliary channel took the form of the physical transfer of top-secret diplomatic bags.
It should be noted that no encryption keys were exchanged on the main communication channel (the red telephone line). Eavesdropping on this line therefore did not allow the keys to be retrieved and the confidentiality of communications to be compromised. However, the risk has been shifted to the auxiliary channel, which complicates the attack, as it is necessary not only to control the main communication channel (by eavesdropping), but also the auxiliary channel (by circumventing the vigilance of the bag carriers).

Question 5) What happens if a symmetric encryption key is stolen? [1 correct answer]

(a) Messages encrypted with this key and sent over the network are compromised. Confidentiality is no longer guaranteed until all compromised keys are updated with new symmetric keys for all participants concerned.

(b) Participants will be able to systematically identify an intruder who uses a stolen key belonging to a legitimate user.

(c) Messages encrypted with other symmetric keys are also compromised, and their confidentiality will no longer be guaranteed until all symmetric keys in the network are updated.

To overcome the problem of key sharing, one initial idea is to send the keys in plain text over the communication channel, which amounts to assuming that the "first message is secure," which is OBVIOUSLY a huge risk!

NEVER transmit a key in plain text over a communication network! Always ensure that its transfer is confidential!

Security protocols based exclusively on symmetric cryptography should therefore not be favored, not for security reasons, but for reasons of deployment and scalability.

8 Asymmetric encryption, the Swiss Army knife of cryptography

Bob is disappointed by the successive failures in designing a robust security protocol and is angry with Alice. Bob decides to let it go, however, and remembers a cybersecurity class on asymmetric encryption. This is a widely used multifunctional approach that corrects many practical flaws in symmetric encryption!
Bob explains to Alice: A pair of keys is generated locally. One of these keys is designated as the **public key** and will be visible to all participants on the network. The second key is the **private key**, which will only be known to its owner and will never be revealed to anyone. The private key is the only key that can decrypt a message encrypted with the public key, and vice versa.

Principle of public key cryptography: Bob encrypts his identity and the secret using Alice's PUBLIC key, Alice then decrypts the message (sent by Bob) using her own PRIVATE key. Since Alice is the only one who knows her private key, the confidentiality of the secret included in the message will be ensured, and there is less risk of disclosure of a private key, compared to a shared symmetric key!

Question 6) Which of these statements about symmetric encryption compared to asymmetric encryption are true?

(a) There are fewer keys to back up because each participant only needs to back up their asymmetric keys and request the public keys of other participants as needed.

(b) The total number of keys to back up is doubled because both the public key AND the private key need to be backed up.

(c) If a private key is corrupted, the confidentiality of messages is not compromised because, without the public key, messages cannot be decrypted.

(d) Just like symmetric encryption, the first message must be in plaintext to initialize the encryption keys if you want to use only the main communication channel, and therefore confidentiality is only partial.

Summary of the protocol in question: Bob sends a secret and his identity encrypted with an asymmetric key to Alice.
In SPAN, open the file $3proto\_asymmetric.hlpsl$. In File→Open HLPSL File...→Desktop→TP, double-click on $3proto\_asymmetric.hlpsl$.
Return to the Desktop, then in TP, double-click on the file to open it with a text editor. Replace the word KEY in $RCV(\{B.S'\}\_KEY)$ line 9 and $SND(\{B.S\}\_KEY)$ line 26 with one of the following keys:

- Public key of A: Ka
- Public key of B: Kb
- Private key of A: $inv(Ka)$
- Private key of B: $inv(Kb)$

Please note: for this to work, the key must be identical in both places.

Save and return to SPAN.
In SPAN, click View HLPSL to update the changes. Click ATSE then EXECUTE to analyze the protocol. Analyze the frames of the various simulations Protocol simulation and Attack simulation offered by SPAN.

Question 7) Based on the results of AVISPA's attack analysis and your analysis of the situation, which of these statements are true?

(a) Messages encrypted with public keys are analyzed as "SAFE" because the intruder cannot guess the private keys of A or B.

(b) Messages encrypted with private keys are UNSAFE because the intruder can decrypt them with the public keys of A or B.

(c) Even if the message encrypted with Alice's private key appears to be SAFE, the scenario is not possible because Bob does not know this key.

(d) To ensure that the protocol is SAFE, messages must always be encrypted with the recipient's private key.

Now you have mastered the intricacies of asymmetric encryption and the use of its keys. But that's not the only benefit of asymmetric encryption; it also guarantees a second fundamental security property: **authentication**.

9 Authenticating the origin of messages with the asymmetric Swiss Army knife

The confidentiality of messages is now guaranteed from the very first message between two participants. Only the legitimate recipient is therefore able to read the content of the encrypted message. However, anyone can send an encrypted message and include the identity they want to impersonate a participant... Bob explains that he doesn't want to send messages to just any Alice, but only to HIS Alice! Alice, her cheeks flushed, replies, "B...Baka! ><". Yay! They're reconciled!

We therefore need a reliable way to prove the identity of participants to others—this is **authentication**, which is our second essential security property to verify, and in particular **authentication of the origin of messages**, which aims to ensure that a message comes from the declared sender.

Launch the $4proto\_auth\_UNSAFE.hlpsl$ protocol in SPAN. In File→Open HLPSL File...→Desktop→TP, double-click on $4proto\_auth\_UNSAFE.hlpsl$ to open the file from SPAN. This is the previous protocol that encrypts the message for Alice, to which we have added in the "goal" section (line 64) the need for authentication, as well as instructions in the roles (lines 14 and 32) for AVISPA to return "SAFE" if and only if the confidentiality and authentication properties of the message origin are both satisfied.

In SPAN, click ATSE then EXECUTE to analyze the protocol. The expected result in SUMMARY is UNSAFE. Click Attack Simulation to display the attack scenario identified by SPAN. The attack identified by AVISPA is called a spoofing attack.

Question 8) Using the attack frame returned by AVISPA, what can you say about this spoofing attack? [2 correct answers]

(a) The recipient can verify the sender's identity by decrypting the message.

(b) The result is UNSAFE because the attacker can read the secret.

(c) The attack is possible because anyone can generate an encrypted message with a public key by including any identity.

(d) The attacker decrypted the secret to generate his attack and succeeded in opening the connection with Alice.

(e) The attack would not have been possible with symmetric encryption.

To prevent this spoofing attack, it is necessary to ensure that the message does indeed come from the stated sender: Bob. The sender can, for example, encrypt their message with their private key, so that Alice can be sure that the message comes from Bob (the only one who knows their private key)[1].

Question 9) What changes can be made to lines 9 and 27 of the protocol to verify that Bob is indeed the sender of the message, and thus correct the vulnerability? [1 correct answer]

(a) $RCV(\{B.S'\}\_Ka) \rightarrow RCV(\{B.\{S'\}\_Kb\}\_Ka)$ (line 9)
$SND(\{B.S\}\_Ka) \rightarrow SND(\{B.\{S\}\_Kb\}\_Ka)$ (line 27)

(b) $RCV(\{B.S'\}\_Ka) \rightarrow RCV(\{B.\{S'\}\_inv(Kb)\}\_Ka)$ (line 9)
$SND(\{B.S\}\_Ka) \rightarrow SND(\{B.\{S\}\_inv(Kb)\}\_Ka)$ (line 27)

(c) $RCV(\{B.S'\}\_Ka) \rightarrow RCV(\{B.\{S'\}\_Ka\}\_Ka)$ (line 9)
$SND(\{B.S\}\_Ka) \rightarrow SND(\{B.\{S\}\_Ka\}\_Ka)$ (line 27)

(d) $RCV(\{B.S'\}\_Ka) \rightarrow RCV(\{B.\{S'\}\_inv(Ka)\}\_Ka)$ (line 9)
$SND(\{B.S\}\_Ka) \rightarrow SND(\{B.\{S\}\_inv(Ka)\}\_Ka)$ (line 27)

To help you: Return to the Desktop, then to TP and double-click on the $4proto\_auth\_UNSAFE.hlpsl$ file to open it with a text editor. Modify the file with the desired response and then save. In SPAN, click on View HLPSL to update the changes. Click ATSE, then EXECUTE to analyze the protocol and verify that you get the SAFE result.

The $5proto\_auth\_replay.hlpsl$ protocol adds a session to the protocol, meaning that two parallel sessions allow Alice and Bob to communicate. This addition of a session corresponds to a more realistic context on a network, as Alice and Bob need to communicate with each other several times.
Launch the $5proto\_auth\_rejeu.hlpsl$ protocol in SPAN. In File→Open HLPSL File...→Desktop→TP, double-click on $5proto\_auth\_rejeu.hlpsl$.
In SPAN, click ATSE then EXECUTE to analyze the protocol. SPAN returns UNSAFE and describes **a replay attack**.
Click Attack Simulation to display the replay attack scenario identified by SPAN. By analyzing the attack, we see that the attacker intercepted a legitimate message from Bob and sent back the exact same message later in order to mislead Alice and make her believe that the messages came from Bob when in fact they came from the attacker. This is therefore identity theft achieved through message replay. If this protocol were deployed in a real network, with Alice acting as a service or server, and authentication based on sending a password as a secret, then an attacker could log in by posing as employee Bob to the server, without Bob realizing it.

Question 10) Using the information generated in Attack Simulation by SPAN, can the attacker read the secret? [1 correct answer]

(a) Yes, because we have specified that the secret must remain confidential.

(b) No, because the intruder only has access to the encrypted message.

(c) Yes, otherwise the attacker would not have been able to authenticate.

(d) No, because the real identity of the participant who generated the message is not changed and therefore the recipient does not send the secret back to the attacker.

[1] Be careful, though! For Alice to be sure that Bob sent the message, she must verify that the message was correctly decrypted with Bob's public key. To do this, she must ensure that the content of the message is consistent, for example, that it contains Bob's identity.

The vulnerability of this protocol is simple: An identical message will have identical encryption, or at least, an encrypted message will remain cryptographically valid for the recipient. Anyone can therefore intercept the message and use it for their own purposes as many times as they want.

Another more concrete example is as follows: imagine a service that requires authentication with a server (Alice). If this authentication consists of simply sending your login/password, or your login/password encrypted with the server's public key, this information does not differ from one authentication session to another and can therefore be identified and then replayed illegitimately to the server to steal your identity.

10 Nonce exchange - How to ensure that each session is unique in order to thwart replay attacks?

After analyzing the protocol and the replay attack, Alice thinks that a way is needed to make messages unique and therefore unusable again.

This is where the **NONCES** come into play.

Nonces are, by definition, **UNIQUE** random numbers. When a protocol uses nonces, it is implied that each generation of a random number results in a different draw. This is an assumption on which the security protocol is based on and which must then be respected in the implementations of the protocol.
The use of nonces in a protocol therefore makes it possible to **make each encrypted message—provided that the message includes the nonce—unique** in its content and completely random in appearance, even if the initial message is identical. Nonces are commonly used in security protocols to ensure message freshness, i.e., to guarantee to the parties involved that the messages are part of their direct interaction and have not been replayed or injected into a communication.
Nonces are often used to interleave messages. For example, if message 1 $M1$ includes the nonce $N1$, then message 2 $M2$ in return will contain the same nonce $N1$. This will allow the sender of $M1$ to be sure that $M2$ is provided in response to $M1$ and therefore that $M2$ is fresh. We can imagine several successive messages $M1$, $M2$, $M3$ that use two nonces, $M1$ and $M2$ carrying the nonce $N1$ and $M2$ and $M3$ carrying the nonce $N2$. This notion of interleaving is very important for designing replay-resistant protocols.

There are two versions of the *proto_nonces_placement.hlpsl* protocol: one version analyzed by SPAN as SAFE ($6proto_nonces_placement_SAFE.hlpsl$) and the other as UNSAFE ($7proto_nonces_placement_UNSAFE.hlpsl$).

Launch the two protocols *proto_nonces_placement.hlpsl* in SPAN one after the other. In File→Open HLPSL File...→Desktop→TP, double-click on the desired *proto_nonces_placement.hlpsl*.

In SPAN, click ATSE then EXECUTE to analyze the protocol. The expected result in SUMMARY corresponds to the file name. Click Attack Simulation to display the frame of the attack identified by SPAN or Protocol Simulation to analyze the protocol sequence.

Question 11) Which statements about the SAFE and UNSAFE *proto_nonces_placement* protocols are true? [4 correct answers]

(a) One protocol encrypts identities and adds the nonce to the message, while the other encrypts the nonce.

(b) An attack on the UNSAFE protocol is possible because the nonces are not encrypted.

(c) An attack on the UNSAFE protocol is possible because identities are not protected.

(d) The SAFE protocol is secure because Bob and Alice are able to authenticate each other, i.e., verify each other's identity cryptographically.

(e) To correct the UNSAFE protocol, the encryption of Alice and Bob's identities must be removed.

(f) The SAFE protocol allows Alice to authenticate Bob because she has proof that Bob knows her private key. Bob had to decrypt the first message to find out the nonce, which he communicated to Alice in the second message.

After looking at the SAFE protocol **the unidirectional authentication**, where Bob authenticates himself to Alice with anti-replay provided by a nonce, let's move on to mutual authentication, where Alice and Bob authenticate each other, again with anti-replay, and where a secret is transmitted confidentially.

Alice suggests using the famous **Needham-Schroeder protocol**. The Needham-Schroeder public key protocol (NSPK) was designed in 1978. It meets this need for mutual authentication and has been widely used. It is one of the first protocols for mutual authentication between two parties over an unsecured channel.

> The following properties can be distinguished. **Unidirectional or mutual authentication** requires several message exchanges (two messages in the 6*proto_nonces_placement_SAFE.hlpsl* file), unlike **message origin authentication**, which consists of encrypting one or more elements of the message with the sender's private key. **Unidirectional authentication**, as in 6*proto_nonces_placement_SAFE.hlpsl*, allows only one party, in this case Alice, to authenticate Bob. **Mutual authentication** allows both parties to authenticate each other. This is the objective of the NSPK protocol.

How the Needham-Schroeder public key protocol (NSPK) works: Alice sends a nonce encrypted with Bob's public key. Bob decrypts the nonce sent by Alice (with his private key), adds his own nonce, and sends it back to Alice, encrypted with Alice's public key. Alice then decrypts the message sent by Bob (with her private key), retrieves Bob's nonce, and returns this nonce to Bob, encrypted with Bob's public key. Here is a formal description:

$$A \rightarrow B : \{Na.A\}_{Kb}$$
$$B \rightarrow A : \{Na.Nb\}_{Ka}$$
$$A \rightarrow B : \{Nb\}_{Kb}$$

After exchanging these nonces, both parties authenticated each other and have the guarantee that their session is unique and cannot be replayed.

Run the 8*proto_NSPK.hlpsl* protocol in SPAN. In File→Open HLPSL File...→Desktop→TP, double-click on 8*proto_NSPK.hlpsl*.
Click on ATSE then EXECUTE to analyze the protocol. SPAN analyzes the protocol as SAFE. Return to the Desktop, then to TP, double-click on the file to open it with a text editor. In the file, on lines 87 to 90, you can add, one by one, a parallel session to the one between Alice and Bob by uncommenting the desired line by removing the "%" at the beginning of the line. Only activate one new session at a time.

Save after each change. In SPAN, click View HLPSL to refresh the changes. Click ATSE then EXECUTE to analyze the protocol.

Question 12) In which cases is the result returned by SPAN of the NSPK protocol SAFE?

(a) A session between Alice and Bob.

(b) Two parallel sessions between Alice and Bob.

(c) Two parallel sessions, one between Alice and Bob and the other between Bob and Alice.

(d) Two parallel sessions, one between Alice and Bob and the other between the intruder and Bob.

(e) Two parallel sessions, one between Alice and Bob and the other between Alice and the intruder.

11 17 years of security - the Man in The Middle attack discovered by G. Lowe on the NSPK protocol and its solution

It was in 1995, 17 years after the creation of the Needham-Schroeder NSPK protocol, that an attack was identified by G. Lowe! NSPK was widely used at the time, and the discovery of the flaw caused quite a stir. This is a logical flaw in the protocol that was identified using formal methods and which no one had suspected existed until then. This demonstrates the value of using formal verification tools when designing security protocols.

The attack used to circumvent the NSPK protocol proposed in 1978 is a well-known type of attack in cybersecurity called "Man in the Middle".

A **Man in the Middle (MiTM)** attack aims to intercept communications between two parties without them suspecting that the communication channel has been hijacked.

Launch the $9proto\_NSPK\_MITM.hlpsl$ protocol in SPAN. In File→Open HLPSL File...→Desktop→TP, double-click on $9proto\_NSPK\_MITM.hlpsl$.
In SPAN, click ATSE then EXECUTE to analyze the protocol. The expected result in SUMMARY is UNSAFE. Click Attack Simulation to display the frame of the attack identified by SPAN. The attack generated by SPAN is a Man in the Middle (MitM) attack.

Question 13) Which statements are true regarding the Man in the Middle attack on the NSPK protocol as presented by SPAN. [4 correct answers]

(a) Alice sends her nonce and identity encrypted with the Intruder's public key to the Intruder. The Intruder decrypts the content and sends it encrypted with Bob's public key to Bob. Bob adds his nonce to Alice's nonce and sends it encrypted with Alice's public key to the Intruder. The Intruder forwards Bob's message containing the nonces to Alice. Alice decrypts the message, accepts the nonce, and sends Bob's nonce back to the Intruder encrypted with the Intruder's public key.

(b) Alice sends her nonce and identity encrypted with Bob's public key to the Intruder. The Intruder decrypts the content and sends it encrypted with Bob's public key to Bob. Bob adds his nonce to Alice's nonce and sends it encrypted with the Intruder's public key to the Intruder. The Intruder transmits Bob's message containing the nonces to Alice. Alice decrypts the message, accepts the nonce, and sends Bob's nonce back to the Intruder encrypted with the Intruder's public key.

(c) The intruder impersonates Alice to Bob and replays certain encrypted messages in order to compromise the mutual authentication between Alice and Bob.

(d) Alice and the Intruder are partners in crime! Without Alice's help, the Intruder wouldn't be able to get Bob's nonce. They're a bit like Bonnie and Clyde of the internet, except that Bonnie doesn't know what she's gotten herself into...

(e) The Intruder is able to decrypt messages encrypted with Alice's and Bob's public keys between Alice and Bob.

(f) Messages must be encrypted with the public key of the intermediary and not that of the recipient, which does not guarantee confidentiality if the message has to pass through several intermediaries.

(g) For the MiTM attack to be possible, Alice must initiate communication with the intruder.

The $10proto\_NSPK\_SAFE\_lowe.hlpsl$ protocol is the hlpsl specification of the NSPK protocol correction proposed by Lowe in 1995.

Launch the $10proto_NSPK_SAFE_lowe.hlpsl$ protocol in SPAN. In File→Open HLPSL File...→Desktop→TP, double-click on $10proto_NSPK_SAFE_lowe.hlpsl$.

In SPAN, click ATSE then EXECUTE to analyze the protocol. The expected result in SUMMARY is SAFE. Click Protocol Simulation then on all the actions that appear in the "Incoming events" section to display and analyze Lowe's solution frame.

Question 14) How does Lowe's solution address the vulnerability of the NSPK protocol? [2 correct answers]

(a) By adding Bob's identity to the nonce transmission, this prevents the intruder from using Bob's message to Alice to authenticate themselves.

(b) By terminating the session between Alice and the Intruder that compromised the protocol.

(c) By allowing Alice to verify who generated the nonce and thus reject the connection.

(d) By correcting encryption errors between A and B, which ensures the confidentiality of exchanges.

(e) By adding nonces to exchanges, which prevents the Intruder from decrypting them all.

Alice and Bob finally understand that the popularity of a protocol does not guarantee that it is secure in terms of its design, let alone the security of its implementation. They also understand that it is important for the specifications of a security protocol to be made public. This allows the skills and vigilance of the cyber community to be leveraged to detect potential logical or software vulnerabilities. In other words, a confidential proprietary security protocol will never be as secure as a protocol that is made public. Both also understand the importance of using formal protocol verification tools such as SPAN & AVISPA to identify potential vulnerabilities during design.

12 Final protocol: it's your turn!

After this long adventure, Alice and Bob learned that protocol security cannot be improvised.
They want to compile everything they have learned with you into a single formal protocol before implementing it.

Question 15) Which protocol described below in formal notation allows us to guarantee mutual authentication and the confidentiality and authenticity of two secrets Sa and Sb issued by Alice and Bob? [2 correct answers]

(a) $A \rightarrow B : \{A.B.\{Na.Sa\}_{Ka^{-1}}\}_{Kb}$
$B \rightarrow A : \{A.B.\{Na.Nb.Sb\}_{Kb^{-1}}\}_{Ka}$
$A \rightarrow B : \{A.B.\{Nb\}_{Ka^{-1}}\}_{Kb}$

(b) $A \rightarrow B : \{A.Na.Sa\}_{Kb}$
$B \rightarrow A : \{Na.Nb.Sb\}_{Ka}$
$A \rightarrow B : \{Nb\}_{Kb}$

(c) $A \rightarrow B : \{A.B.\{Na.Sa\}_{Kb^{-1}}\}_{Kb}$
$B \rightarrow A : \{A.B.\{Na.Nb.Sb\}_{Ka^{-1}}\}_{Ka}$
$A \rightarrow B : \{A.B.\{Nb\}_{Kb^{-1}}\}_{Kb}$

(d) $A \rightarrow B : \{\{Na.Sa\}_{Ka^{-1}}\}_{Kb}$
$B \rightarrow A : \{\{Na.Nb.Sb\}_{Kb^{-1}}\}_{Ka}$
$A \rightarrow B : \{\{Nb\}_{Ka^{-1}}\}_{Kb}$

(e) $A \rightarrow B : \{A\{Na.Sa\}_{Ka^{-1}}\}_{Kb}$
$B \rightarrow A : \{B\{Na.Nb.Sb\}_{Kb^{-1}}\}_{Ka}$
$A \rightarrow B : \{\{Nb\}_{Ka^{-1}}\}_{Kb}$

(f) $A \rightarrow B : \{A.B.\{Na.Sa\}_{Ka}\}_{Kb}$
$B \rightarrow A : \{A.B.\{Na.Nb.Sb\}_{Kb}\}_{Ka}$
$A \rightarrow B : \{A.B.\{Nb\}_{Ka}\}_{Kb}$

13 Conclusion

To design a cryptographic protocol, you must first list the security properties you expect from that protocol, then verify that the designed protocol does indeed satisfy those properties. With the help of Alice and Bob (not forgetting the attacker), you have learned how to use symmetric and asymmetric cryptography, as well as nonces, to thwart message replay attacks. You have also been able to identify the limitations and weaknesses of these elements in ensuring the confidentiality of secrets, the authentication of message origin, and one-way or mutual authentication. We have seen how difficult it is to define a secure protocol and how important it is to use formal verification tools to identify protocol flaws that could be exploited.

But remember that you have only scratched the surface of protocol security! You must be vigilant and rigorous from the design stage through to the implementation and deployment of the protocol on a network to avoid introducing vulnerabilities, such as buffer overflows, poor configuration with weak cryptographic algorithms, or a public key that does not correspond to the correct entity. In fact, a single flaw at any of these stages is enough to compromise the security of communications.

References

[1] A Short SPAN+AVISPA Tutorial, Thomas Genet, INRIA, https://inria.hal.science/hal-01213074

[2] SPAN: a Security Protocol ANimator for AVISPA USER Manual, Thomas Genet, INRIA, https://people.irisa.fr/Thomas.Genet/Crypt/manual.pdf#:~:text=Avispa%20%5BABB%2B05%2C%20avi%5D%20is%20now%20a%20commonly%20used,cation%20techniques%20on%20the%20same%20protocol%20speci%20cation

[3] Cryptographie et sécurité, Franck Pommereau, cours donné aux Master 1 MIAGE et informatique CNS SA/SR, Université Paris-Saclay/UEVE, 2023.