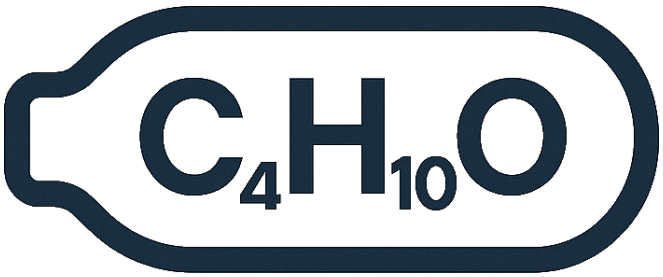
LICHTNER Marc
SIBELLAS Baptiste
VANRULLEN Lancelot

Protocol

PROTOxyde d'alCO(o)L



03/10/2025

Description of the PROTOxyde d'alCO(o)L Protocol

Membres du groupe

- LICHTNER Marc
- SIBELLAS Baptiste
- VANRULLEN Lancelot

The Protocol

The Protocol is described as follows :

$$A \to B : A, \{|N_A, B|\}_{K\{AS\}}$$

$$S \to B : S, \{|N_A, A|\}_{K\{BS\}}$$

$$B \to A : B, \{N_A, N_B\}_{pub(A)}$$

$$A \to B : A, \{K, N_B\}_{pub(B)}$$

$$B \to A : B, N_B$$

Initial Knowledge

We assume that at the beginning of the Protocol agents **A and B know the public key** pub(C) of any agent C. A **pre-shared key** used for asymmetric encryption exists between every agent and the server S. This server S is honest.

Values Generated During the Protocol Execution

N_A and N_B are Nonces generated respectively by A and B. Reminder : those Nonces are associated to **one and only one session** with a unique other agent. E.g. : When A wants to talk with B, a Nonce N_A will be issued by A, and only a message apparently coming from B should contain N_A – any message coming from apparently any other agent than B with N_A will be discarded and the **Protocol will fail**.

Assumption on the Protocol

Here are the assumptions made about how the Protocol handles specific points :

- The key ***K* is unique and freshly generated** at each session
- Nonces are **unique** to one session with one other identity (see previous part)
- If an agent A is having a session with B, A or B **must not start another session** with the other agent it is having a session with already.

- If Protocol fails on any side (A or B) because of : **incorrect content** of message (pattern matching fails), **incorrect key** used, **incorrect Nonces**, etc. (as seen in classes) - then the Protocol "state" goes back to the **first state** it can be in.
- If no response is received but was expected, a **timeout** will declare whether or not the exchange failed
- An **agent will abort the Protocol** run if it receives a message that uses symmetric encryption, but also uses the wrong key (E.g : if B receives $\{|N_A, B|\}_{K\{IS\}}$ instead of $\{|N_A, B|\}_{K\{AS\}}$, the Protocol will fail).
- An **agent will abort the Protocol** run if it receives a message that uses asymmetric encryption, but also uses the wrong public key (E.g : if A receives $\{N_A, N_B\}_{pub(I)}$, the Protocol will fail).
- The **randomness** used in this environment is perfect and collisions between Nonces is impossible.

Protocol Description

- In the first step, A sends to the server **who** it wants to talk to as well as a **Nonce** used to verify the right delivery of this message. This is protected using the pre-shared key.
- Then, the server transmits to the recipient the **Nonce** as well as **who** contacted them. This is also protected using the pre-shared key.
- From now on, we move to **asymmetric encryption** and messages are protected using **Nonces**. In the third step, B acknowledges the fact that **A** is trying to contact them - also transmitting a **Nonce** of their own.
- A verifies that it received both **"B"** and the **Nonce** it issued for this specific identity. When the check passes, it is sure that it is talking with the good recipient and transmits the key with B's Nonce.
- Upon receiving the **Key K** and the **Nonce**, B also checks the fact that the identity A announced was the one associated to the Nonce received. If this check passes we are sure that the key K was sent by A to B - and without tampering.
- Finally, since both Nonces were completely secret and are not going to be used again, B **releases one of them** to put an end to the session.

Security Properties

- If B finishes a Protocol run and thinks that he received a key K from A, then A effectively sent K to B. This is guaranteed thanks to the usage of the Nonces at each step of the Protocol and the identity verifications happening in the honest agents (that are the only ones to know the secret Nonces).
- If A finishes a Protocol run during which he sent K to B, then B has effectively received K from A. We are sure of this thanks to the last step that guarantees that if one of the Nonces is released then the key K must have been sent along the way.
- The key K stays secret among A and B. This is thanks to the asymmetric encryption between the two agents that agreed to talk to one another.

- In each session, a new fresh key K is exchanged. Not only is this an assumption made because we need the issuer to create a new fresh key for each session, but then the freshness is guaranteed thanks to the Nonces unique for each session.

Cost of the Protocol

Step 1: $64 = 1 + 10 + 50 + 2 + 1$
Step 2: $64 = 1 + 10 + 50 + 2 + 1$
Step 3: $55 = 1 + 1 + 50 + 2 + 1$
Step 4: $55 = 1 + 1 + 50 + 2 + 1$
Step 5: $2 = 1 + 1$

Total: 240 $= 2 * 64 + 2 * 55 + 2$