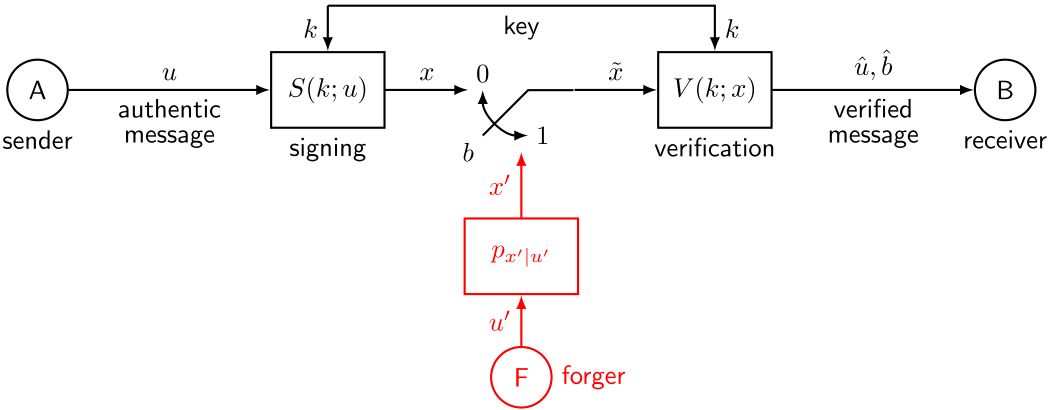
Laboratory session 3

Message authentication and integrity protection

Nicola Laurenti, Laura Crosara May 12, 2025

Laboratory session 3 — Contents

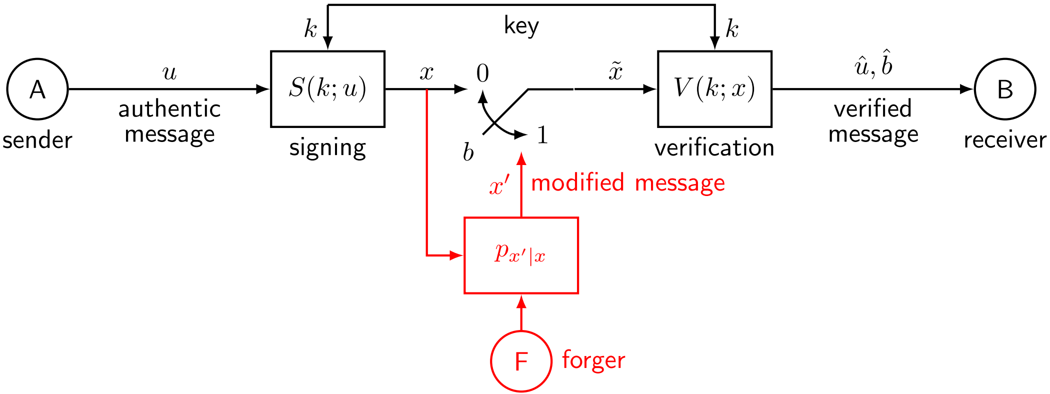
General model of the authentication problem



Forging attack

F wants to build x' so that $\hat{u} = u'$ and $\hat{b} = 0$ (i.e., u' is accepted)

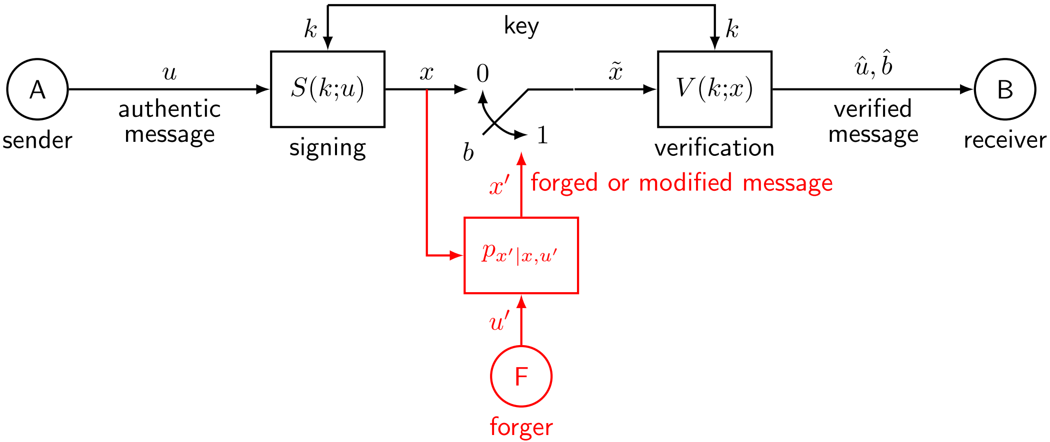
General model of the integrity protection problem



Illegitimate modification (alteration) attack

F can block x and wants to replace it with x' such that $\hat{u} \neq u$ and $\hat{b} = 0$

Authentication + integrity protection system



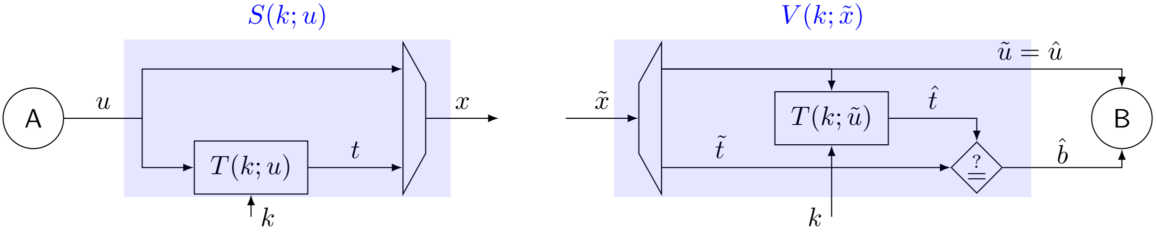
Authentication tags

A typical solution for signing is to append a tag to the message

$$x = (u, t) \quad , \quad t = T(k, u)$$

The tag depends on both the key and the message

The corresponding verification splits the received signal into message and tag, computes the correct tag on the received message and checks it against the received tag



Implement the authentication scheme

Consider the following symmetric scheme for message authentication and integrity protection:

- the message u is a sequence of M bits
- the key k is a sequence of K bits
- the authentication tag t is the binary representation of the integer product $s = s_u s_k$, where s_u and s_k are the sum of the digits in the decimal (base 10) representation of u and k, respectively.

The tag is transmitted along with the message u, and the receiver, which knows k, computes the tag from the received message and checks whether it is identical to the received tag. If so, the message is accepted as authentic.

Task 1

Using a programming language of your choice, implement the above scheme, and the corresponding verification so that the complexity of each is polynomial in both M and K.

Design and implement a substitution attack

Your aim is to evaluate the weakness of the naïve symmetric scheme for message authentication and integrity protection, implemented in Task 1.

Task 2

Using a programming language of your choice

- design and implement a substitution attack to the above scheme that, without knowing the key k, and after blocking and observing a legitimate message/tag pair $x = (u, t)$ sent by A, replaces it with a different pair $\tilde{x} = (\tilde{u}, \tilde{t})$ so that $\tilde{u}$ is accepted by B as authentic;
- evaluate the computational complexity for this attack.

What is the success probability of your attack strategy?

Design and implement a forging attack

Task 3

Using a programming language of your choice

- design and implement a forging attack that, without knowing the key k, aims at creating a message/tag pair $\tilde{x} = (\tilde{u}, \tilde{t})$, such that $\tilde{u}$ is accepted by B as authentic;
- evaluate its computational complexity;
- evaluate its success probability.

What you need to turn in

Each team must turn in, through the Moodle assignment submission procedure:

1. the source code for your implementation (either as a single file, many separate files, or a compressed folder)
2. a short report (to be submitted as a separate file from the source code file / compressed folder) in a graphics format (PDF, DJVU or PostScript are ok; Word, TeX or LaTeX source are not), including:
   2.1 a brief description of your implementations for Tasks 1-3, explaining your choices;
   2.2 a plot of the average required computation time vs M and K, for the authentication/verification scheme;
   2.3 two plots of the average required computation time vs M and K, both for the successful substitution and forging attacks;
   2.4 a plot of the success probability vs M and K, for the forging attack.

