A brief introduction to cryptography

Meryem Benammar

3 février 2022

Résumé

In the present textbook, we briefly touch on basic notions on cryptography, and more specifically, on stream ciphers. The aim is to give an intuitive understanding of the complexity of designing practical security scheme, and to motivate the tremendous undertaken by the cryptography community in order to develop feasible and practical algorithms. A nice introduction on cryptography can be found in *Understanding Cryptography by Christof Paar* (link on the course page)

Table des matières

1 Security in a communication chain

At this point of the course, it is time for us to gather up all three components of the information engineering chain.

In the previous lectures, we introduced the principle of data compression and forward error correction, and showed how these two functions allow to save resources (through compression) while achieving reliability (through error correction). In this course, we will add the third constraint in information engineering problems, namely, information security, through the design of encryption and decryption schemes.

The first question we should ask is where exactly does information security take place in a communication chain ? Figure 1 answers this question.

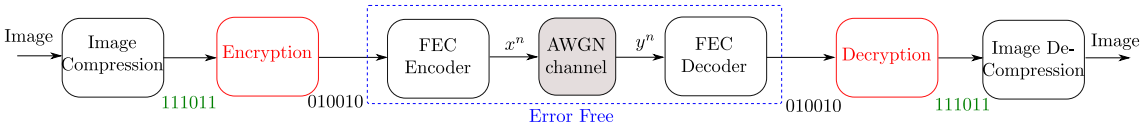


FIGURE 1 – A secure communication chain

The rationale behind this order of blocks is as follows. If we perform security prior to image compression, then the compression scheme would be useless since the information security algorithms in general destroy any apparent structure of the data, which is elementary for compression. If on the contrary encryption is performed after error correction, then any loss of a single bit of the encrypted message would be detrimental for the decryption process (as will become clearer later).

Hence, naturally, information security is handled after information compression and before error correction.

2 The one-time pad

2.1 The secure communication problem

Let us consider a setting with a legitimate transmitter, a legitimate receiver and an eavesdropper which has access to all the information sent on the noise-free channel. Let us assume also that the transmitter and receiver share a secret key which they can use to communicate securely.

Taking into account the characteristics of the image compression, and the FEC, the channel for a secure communication chain can be established as in figure 2.

The secure communication problem consists in :

- A message u^k of k bits which needs to be transmitted to a trusted user (these bits are supposed uniformly distributed)
- An encryption (ciphering) function which produces the following encrypted message c^k given by

$$c^k = f(u^k, v^k) \tag{1}$$

  where v^k is a possible source of randomness.
- An error free binary communication link, i.e., u^k is received without errors, on which an eavesdropper receiver all encrypted messages

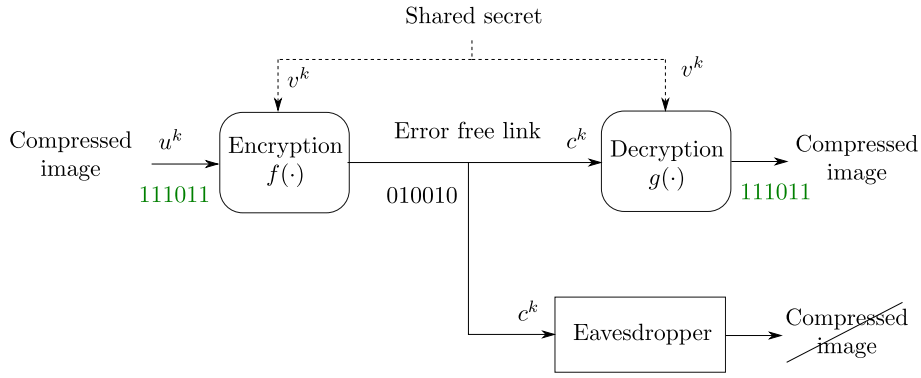


FIGURE 2 – A secure communication chain

- A decryption (deciphering) which produces the following decrypted message

$$\hat{u}^k = g(c^k, v^k) \tag{2}$$

where v^k is the same source of randomness as the one of the encryption.

The encryption function $f(\cdot)$, source of randomness v^k and decryption function $g(\cdot)$, need to be chosen so that

$$\mathbb{P}(\hat{U}^k \neq U^k) = 0 \text{ and } I(U^k; C^n) = 0, \tag{3}$$

where $\mathbb{P}(\hat{U}^k \neq U^k)$ measures the reliability of the communication, and $I(U^k; C^n)$ measures the so-called information leakage to an eavesdropper which observes only the encrypted message c^n.

Any scheme which satisfies these conditions is termed a perfectly secure communication scheme. Perfect secrecy implies that an eavesdropper who observes only the encrypted message does have a full uncertainty about the transmitted message, as if it needed to guess it at random.

$$I(U^k; C^k) = 0 \implies H(U^k|C^k) = H(U^k) = k. \tag{4}$$

2.2 The one-time pad cipher

The one-time pad, also known as Vernam's cipher, is the most elementary, and yet perfect, secure communication scheme. It is depicted in figure 3.

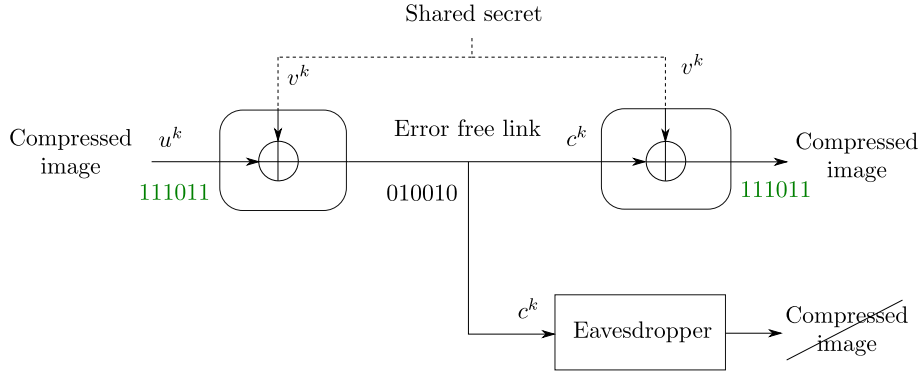


FIGURE 3 – The one time pad

The one-time pad principle is the most perfect source of secrecy that one can achieve from a mathematical point of view. It consists in the following :

- A key (or cipher) v^k of k bits which is know perfectly to both communication parties
- An encryption (ciphering) function which produces the following message c^k given by
$$c^k = u^k \oplus v^k \tag{5}$$
- A decryption (deciphering) which combines the key v^k and the received message c^k to produce the following
$$u^k = c^k \oplus v^k \tag{6}$$

Shannon proved in 1949 that this scheme is perfectly secure as follows. If you measure the *information leakage* which occurs between the original message and the encrypted message, which can be measured by the mutual information $I(U^k; C^k)$, then you can show that this leakage is zero if and only if a few conditions are satisfied :

- The key must be at least as long (in bits) as the message (if it is longer, we pad the message with zero bits)
- The key must be a perfectly uniformly distributed binary stream (it cannot stem from a pseudo random generator)
- The key must never be reused in whole or in part
- The key must be kept completely secret from any undesired node

In this course, our aim is to highlight the importance of these assumptions and show that, if one of them is violated, the scheme is no longer secure.

2.3 One-time pad vulnerabilities

In the following, we discuss the implication of violating each of the one-time pad principles.

- A shorter key than the message : it is clear why using a shorter key than the message, leaves out unprotected part of the message.
- A non-uniform key : if the key is uniform then the only option an intruder can have is an exhaustive search over all possible keys, which amounts to an exhaustive search over all possible messages (since the key is at least as long as the message). If the key used is non-uniform, then it means that some values are more probable than others, which might allow the eavesdropper to perform fewer operations to make the guess.
- Re-using the key : Assume that two messages u_1^k and u_2^k were encrypted with the same key v^k. Then, when receiving both encrypted messages c_1^k and c_2^k, one can perform a xor operation on the result and obtain

Watch a video about this

$$c_1^k \oplus c_2^k = u_1^k \oplus v^k \oplus u_2^k \oplus v^k = u_1^k \oplus u_2^k. \tag{7}$$

Although one cannot recover u_1^k and u_2^k separately, one can have information on the difference of both messages $u_1^k \oplus u_2^k$, which might leak some useful information and reduce the search space for both u_1^k and u_2^k.

- Prior information on the key : if the key was generated using a certain predictable process (as for instance the output of a pseudo-random number generator) then, the eavesdropper can reverse engineer or even try to guess the key and hence, decipher part of the transmitted message.

In this course, we show how using the same key twice, or leaking the seed, is detrimental to the one-time pad security.

3 Cryptography and stream ciphers

In the following, we will investigate practical implementations of the one-time pad, but prior to that, let us give a brief presentation of cryptographic schemes.

3.1 Symmetric and asymmetric cryptography

In cryptography, there exist two types of security mechanisms : symmetric cryptography in which the nodes share a secret prior to communication, and asymmetric cryptography which relies on public and private keys.

Symmetric cryptography has been around for quite a few centuries now (example : Caesar's cipher, Enigma machine), to ensure mostly data encryption. However, asymmetric cryptography dates back to 1976, is mostly used for key agreement and digital signatures (example : RSA).

In this course, we will be mostly interested in symmetric cryptography, where a secret key is shared between the two legitimate nodes in a secure communication system.

3.2 Stream ciphers

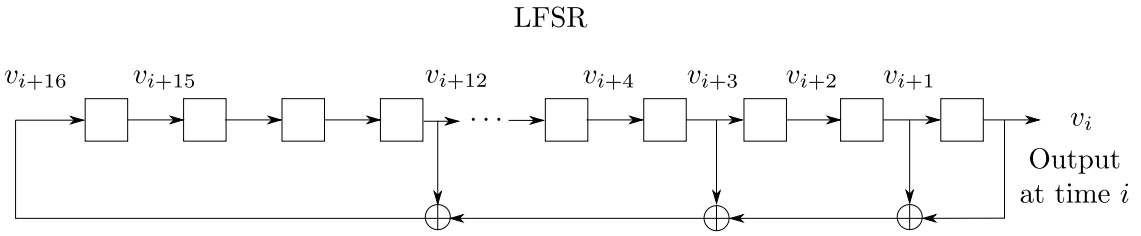
Among symmetric cryptography schemes lie the families of *block ciphers* and *stream ciphers*. Both types of ciphers differ in the definition of the encryption function $f(\cdot)$ and the decryption function $g(\cdot)$.

- In a block cipher, alike the Enigma machine, the message is transformed into an encrypted message by means of an often non-closed form and iterative transformation. The encryption and decryption operations are often different, however, they require the same initial seed.
- In a stream cipher, the encryption and decryption operations are those of the one time pad, and hence, stream ciphers are a practical implementation of the one-time pad. The common random key used for these encryption/decryption operations is often generated from a Pseudo-Random Number Generator (PRNG) which is often initialized with a so-called seed.

In this course we are interested in the family of *stream ciphers* which represent a practical implementation of the one-time pad principle.

3.3 PRNG and CS-PRNG

Stream ciphers require a source of randomness, which would be as perfect as possible. However, having a common source of pure randomness available at both the transmitter and receiver is almost impossible (since it would require two random physical processes



Initial seed: $(v_{16}, ..., v_1)$

State equation: $v_{i+16} = v_{i+12} \oplus v_{i+3} \oplus v_{i+1} \oplus v_i$

Polynomial description: $1 + v^4 + v^{13} + v^{15} + v^{16} = 0$

FIGURE 4 – The LFSR selected for the mission

exactly equal). Hence, in practical problems, we resort to a Pseudo-Random Number Generator is (PRNG), which although apparently random, are in fact a perfectly deterministic transformation, hence, reproducible at the destination.

Once initialized with a given seed, the PRNG will produce a relatively long binary stream whose length will depend on the characteristics of the PRNG (period, memory, internal state, ...). Powerful PRNGs have the property of producing arbitrarily long, non-periodic and uniform series of bits, which makes them a good candidate to construct stream ciphers. However, not all powerful PRNG are good stream ciphers, since some of the PRNG known in literature are easily breakable with a little prior on the message transmitted (header, pattern, ...). The class of PRNG which are hard to break and deemed secured are called cryptographically secure PRNG (CS-PRNG).

In this course, we will be interested in a specific PRNG called the Lagged Fibonacci Shift Register (LFSR), which is not a CS-PRNG, and thus, presents some security issues which will be highlighted in this course.

4 LFSR as a stream cipher

4.1 LFSR structure

The LFSR random number generator is a function which, given a seed s, produces a long stream of seemingly uniform and independent bits by means of a bench of shift registers. An example of an LFSR which was selected for this study is given in figure 4. An LFSR is characterized by the following :

- A state equation, here given by

$$v_{i+16} = v_{i+12} \oplus v_{i+3} \oplus v_{i+1} \oplus v_i \tag{8}$$

- A memory m which is equal in our case to 16
- An initial seed which is given by the initial $m = 16$ outputs of the LFSR $(v_{16}, \ldots, v_1)$, and which cannot be the all 0 seed (otherwise the LFSR does not produce anything).

An example of the outputs of the LFSR at times from $i = 17$ upwards are given by

$$v_{17} = v_{13} \oplus v_4 \oplus v_2 \oplus v_1 \quad (9)$$
$$v_{18} = v_{14} \oplus v_5 \oplus v_3 \oplus v_2 \quad (10)$$
$$v_{19} = v_{15} \oplus v_6 \oplus v_4 \oplus v_3 \quad (11)$$
$$v_{20} = v_{16} \oplus v_7 \oplus v_5 \oplus v_4 ... \quad (12)$$

The LFSR being a PRNG, the values output by it will eventually loop again to an initial value, after a certain number of bits called the period P. For an LFSR of memory m, it can be shown that the maximum value of the period $P = 2^m$, however, this value is not achieved unless the LFSR is a so-called maximal LFSR, which is allowed by the state equation. A maximal LFSR presents also the advantage of having the same period whatever the initial seed. A list of all maximal LFSR can be found in Wikipedia for a given memory m, and the LFSR selected here is one of those best LFSR.

4.2 LFSR and security

Although LFSR have been quite common in pseudo random number generation (for Monte Carlo simulations or for stochastic optimization, ...) it has long been showed that an LFSR is not cryptographically secure, and hence should not be used as a stream cipher. We detail in the following these vulnerabilities.

4.2.1 Known plaintext attack

Assume that part of the encrypted message is known to the eavesdropper. This is in general the case for any packet (header of the packet, structure, periodicity of beacons, ...), and assume that we have a LFSR with $m = 16$ bits of internal state (memory).

- If the LFSR state equation is known, then the knowledge of any consecutive 16 bits of the output of the LFSR leads to breaking it. These bits could be the seed (initial state) or any later state intercepted during the communication by knowledge of a certain structure in a message or prior information on the type of message ...
- If the LFSR state equation is not known, then knowing $2 * 16 = 32$ consecutive bits is enough to break the LFSR using the Berlekamp Massey algorithm.

In this course, we will care mostly about the first scenario, and assume that the LFSR equation is known, and that the only variable unknown is the seed, and by making the students recover the seed, allow them to break the LFSR and reconstruct the images.

4.2.2 Brute force attack

Another reason why the LFSR should not be used is due to the fact that, if the memory of the LFSR is not too long, then a brute force attack can be implemented to recover the seed and hence, get the binary stream at the input. Recovering the seed would require $2^m - 1$ operations where m is the memory of the LFSR (the -1 stems from the fact that the all zero seed is impossible). Hence, brute force attacks on simple LFSR is one of the biggest vulnerabilities for this PRNG.