Cybersecurity MSc: Applied Cryptography 2020-21

Appeal Project: Secure communication through a noisy channel

February 17, 2022 Due date: February 26, 2022

Changelog

- v1.0 - Initial version.

Introduction

Alice an Bob intend to communicate with each other through a secure communication channel. For that, they will both use a communication device that ensures their authentication (meaning that they are sure to be exchanging data with each other) but their communication is not confidential (attackers may eavesdrop it).

To make life more complicated, the communication channel is noisy, meaning that it can flip bits on the transmitted data. To help communication endpoints to be aware of the transmission errors caused by the bit flips, the communication channel implements a redundancy service, described below. Such redundancy helps the receiver to become aware of the presence of bit flips, but not exactly where they occurred and how many they are. In rare circumstances it may even happen that bit flips are not revealed by the redundancy service.

The goal of this project is to implement a confidential, bidirectional communication between Alice and Bob over the previously referred noisy channel. For that, Alice and Bob need to agree upon a shared, secret key and use that key to encrypt messages. However, due to the noisy characteristics of the channel, they must as well tackle errors caused by bit flips. In general, errors in communications can dealt with in two ways: by correcting them or by giving up and requesting a retransmission. The possibility to apply correction depends on the kind and amount or redundancy, and has limits. But, when possible, it reduces retransmission overheads.

1 Noisy channel

The noisy channel is supported by a black-box mechanism that includes redundancy. The redundancy works as follows (see Figure 1). Assume that a message m, with length l, is provided for transmission. First, the message is forcefully padded, with a PKCS#7 strategy, to an N-byte alignment; we will call it m' (with a length $l' > l$). Then, m' is broken in N consecutive blocks, $B_1, \cdots, B_N$, and a new redundancy block R is added to it. Thus, the final message, m'', is the concatenation of m' with R. The value of R is given by the exclusive-or of all m' blocks:

$$R = \bigoplus_{1}^{N} B_i$$

This redundancy mechanism allows a fast detection of bit flip errors anywhere in the message m'', but not their correction by itself. Furthermore, false negatives (undetected bit flips) may occur. The

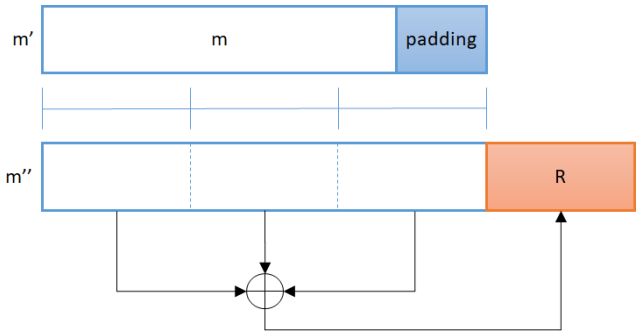

Figure 1: Padding of m, producing m', and addition of redundancy block R, producing m''. In this example $N = 3$.

fast detection can be done by recomputing the expected value of R from the N first message blocks and checking if it matches the received R. A mismatch is a clear error indication; an equality most probably means an absence of errors, but there is a small probability of having them.

The noisy channel implementation that is provided simulates bit flip errors on each block of m'' with a given probability P. The amount of bit flips is unknown, it can range from 1 to the total bits of the block. All blocks may be affected by errors, including R.

1.1 Message correction with digests

Lets assume that we have a single bit flip in the value of m'' received. In this case, the error can occur in two places:

- In the block R; or
- In any other m'' block (in other words, in the m' part that constitutes m'').

In the first case, if the original message m packed in m'' contains a digest (or a MAC) of its contents for integrity control, it is easy to check if m was affected by bit flips or not.

In the second case, we can do the same, but interactively. Using R, which is correct, we compute a corrected B_i (let it be B_i^*) using

$$B_i^* = R \oplus \bigoplus_{j=1}^{N} B_j \quad , j \neq i$$

and use it instead of B_i to check the message digest (or MAC). A successful validation is a proof that B_i is wrong and B_i^* is its correct value. If no B_i could be found to be wrong, then that means that bit flips occurred in two or more blocks, possibly in R as well, and recovery is impossible without extra means.

Finally, note that the padding area can be used for integrity verification, since it has a well-structured value. In other words, bit flips in the padding area can be detected if the padding does not match its specification. However, extensive errors in the padding may generate a wrong value that conforms with the specification. In this case, the padding area can be tentatively rewritten with different padding lengths prior to performing other validations, namely the ones previously referred.

2 Homework

As previously referred, the work consists of implementing a secure, bidirectional communication between two applications, A and B, through a noisy channel. The confidentiality in each direction must use a cipher with a continuous feedback for ensuring sequentiality. This means, for instance, that if Alice sends m_1, m_2 and m_3 in a row, these messages should be encrypted as if forming a larger message formed by their concatenation. Thus, if using CBC, for instance, the IV for m_i should be the last block of the cryptogram of m_{i-1}.

You can use a stream or a block cipher for confidentiality. Please note that block cipher modes require padding, and this padding is independent from the padding of the noisy communication channel.

You can use a digest or a MAC for integrity control. Note, however, that using a simple digest over encrypted contents is a weak form of integrity protection, since an attacker can change both to a match (i.e., an attacker can change the cryptogram and then its digest accordingly). Therefore, if a simple digest is used, it should be computed over plaintext contents, and not encrypted contents. You should take this requirement into consideration in the design of your solution.

The secret, the shared key used by Alice and Bob to interact with each other must be negotiated with Elliptic Curves' Diffie-Hellman (ECDH). You can use any of the standardized curves (NIST curves, Bernstein's curves, etc.). The first IV used in for encryption must be derived from the same ECDH negotiation. As suggested by good practices, do not use the same keys and IVs in different communication directions. Thus, use a different value for each sense of the communication when deriving the two $\langle$ key, IV $\rangle$ pairs.

2.1 Code provided

The noisy channel black-box will be provided as an executable. This executable is able to launch two independent applications that can communicate with each other through the noisy channel.

The noisy channel can be used through four file descriptors, with values 3, 4, 5 and 6:

3 – This descriptor allows an application to read bytes from the noisy channel, but not the redundancy blocks. They are N-byte aligned.

4 – This descriptor allows an application to write bytes to the noisy channel. These must be N-byte aligned.

5 – This descriptor allows an application to properly manage the redundancy added to the transmission. It is a read-only descriptor, thus an information source.

The first 2 bytes it provides is the value of N (unsigned 16-bit integer) that is being used by the noisy channel.

Thereafter, for each message received, it provides a signed 2-byte integer. If negative, that means a mismatch between the received R and the rest of the message. Its absolute value yields the length of R, which can be read next from this same descriptor. This length, multiplied by N, gives the length of m' that can then be read from descriptor 3.

6 – This descriptor allows an application to manage an acknowledge channel with the peer through the noisy channel. Upon receiving a message, if it could not be correctly received, a byte with the value zero should be written, and the noisy channel handler on the peer side will retransmit the previous message. Otherwise, a byte with any other value should be written.

This descriptor also allows an application to signal the end of a message to be transmitted through the noisy channel. In particular, a message can be provided chunk by chunk, using descriptor 4, and the end of the message provisioning is signaled by writing a byte (with any value) in this descriptor.

The noisy channel calling syntax is:

```
nchannel N-value P-value app1 [app1 arguments] | app2 [app2 arguments]
```

where `N-value` is the value of N used by the channel (an integer value, higher than 1) and `P-value` is the error probability on each block (a positive real value, between 0 and 1).

The `nchannel` application recognizes the symbol `|` as the separator between the command description of both applications. Note, however, that this symbol needs to be provided as `\|` in a Linux console, in order to prevent the console's shell to recognize `|` as an inter-command pipe.

In Elearning you can get the source code of this application, as well as the source code of two test applications: `input`, that reads data from the standard input, sends it through a noisy channel to `echo`, which echos the data received, which is received by `input` and written in the standard output. These applications implement a very basic error control mechanism: they require a retransmission if the message upon an error revealed by the received R.

3 Homework delivery

Send your code to the course teachers through Elearning (a submission link will be provided). Include a small report, with no more than 10 pages, describing the implementation (not copies of the code!) and the choices taken (with a justification!), namely:

- What is the structure chosen for the messages;
- How are messages encrypted and the criteria for your choices;
- What is the integrity control chosen for the messages, and the criteria for your choices; and
- The complete message validation algorithm.

Include in your report an experimental analysis of the percentage of messages that had to be retransmitted do to errors as a function of the error probability P.