A note on symmetric cryptography

Kristian Gjøsteen[*]

August 19, 2010

Abstract

This note describes the basic definitions and constructions in symmetric cryptography, and informally discusses security properties.

1 Introduction

The setting in this note is that Alice and Bob want to communicate securely. The only connection available to Alice and Bob is an insecure channel (attackers can listen to their messages and perhaps also modify them). Examples of channels are phones, semaphore flags or morse code transmitted as sound through a wall.

What does *communicate securely* mean? In this note, Alice and Bob are only interested in the security properties *confidentiality* and *integrity*.

Confidentiality means that anyone who listens to the insecure channel cannot extract any information from that channel, except for the fact that data is being transmitted and the length of the data transmitted.

Integrity means that any tampering with data transmitted over the insecure channel will be detected, except for injecting copies of old transmitted data (replay).

Note that we can have confidentiality without integrity, and integrity without confidentiality. Also note that in this two-party setting, integrity provides implicit authentication, since anything Bob receives either originated with himself or with Alice.

The main assumption in symmetric cryptography is that at some point in time before they need to communicate, Alice and Bob were able to agree on a shared secret that nobody else knows. Since this secret must be kept secure, it should be very short. This shared secret is called the *key*.

We shall see how we can use various cryptographic tools to enable Alice and Bob to communicate securely.

1.1 Notation

The *exclusive or* binary operation is denoted by $\oplus$ and combines 0 and 1 as follows:

	0	1
0	0	1
1	1	0

Note that $x \oplus x = 0$ and $x \oplus 0 = x$ for all x. A bit string is a sequence of 0s and 1s, and $\oplus$ operates on bit strings in the obvious bitwise fashion: $110011 \oplus 101011 = 011000$.

We denote the set of bit strings of length n by $\{0,1\}^n$, and the set of bit strings of arbitrary length by $\{0,1\}^*$. $(\{0,1\}^n)^*$ denotes the set of bit strings whose length is divisible by n. If x and y are bit strings, $x \| y$ denotes the concatenation of those bit strings (so $010101 \| 1111 = 0101011111$).

[*]With some improvements by Harald Hanche-Olsen

A function f taking as input a bit string of length l and a bit string of arbitrary length and returning a bit string of length n will be denoted by $f\colon \{0,1\}^l \times \{0,1\}^* \to \{0,1\}^n$.

An algorithm is a sequence of instructions for performing some computation, accepting some input and producing some output. A probabilistic algorithm is allowed to toss coins (get random bits) when computing its output, and therefore, invoking the algorithm twice on the same input may produce different output. A deterministic algorithm is not allowed to toss coins, and will therefore always produce the same output from a given input. We denote by $A(x)$ the output of the algorithm A when given the input x. (If A is probabilistic, $A(x)$ actually denotes a probability space, but we gloss over that fact in this note.)

2 Hash functions

Definition 1. A *hash function* with output length n is a function $h: \{0,1\}^* \to \{0,1\}^n$, and the work required to compute the hash value of a bit string is linear in the length of the bit string.

Design strategies A common design for hash functions is to iterate a *compression function* $h': \{0,1\}^n \times \{0,1\}^N \to \{0,1\}^n$. First the message m is split into blocks $m_0, m_1, \dots, m_{s-1}$ of length N (this requires padding). Then y_0 is set to a specified initialization vector, we use some iteration rule, e.g. $y_{i+1} = h'(y_i, m_i)$, to compute y_{i+1} for $0 \le i < s$, and define $h(m) = y_s$.

Applications Hash functions have numerous applications in cryptography, for example in public key encryption and digital signatures where properties such as one-way-ness and collision resistance are important. But they are included in this note primarily because of their use in message authentication codes. For this application, one-way-ness and collision resistance are irrelevant; the purpose of the hash function is to act as a "bit mixer". Many commonly used hash functions seem to be good "bit mixers".

3 Encryption

Definition 2. A *symmetric cryptosystem* consists of three sets $(\mathscr{P}, \mathscr{C}, \mathscr{K})$ and two algorithms $(\mathscr{E}, \mathscr{D})$. $\mathscr{P}$ is the set of possible plaintexts, $\mathscr{C}$ is the set of possible ciphertexts, and $\mathscr{K}$ is the set of keys.

The *encryption algorithm* $\mathscr{E}$ is a probabilistic algorithm that takes as input a key from $\mathscr{K}$ and a message from $\mathscr{P}$, and outputs a ciphertext from $\mathscr{C}$.

The *decryption algorithm* $\mathscr{D}$ is a deterministic algorithm that takes as input a key from $\mathscr{K}$ and a ciphertext from $\mathscr{C}$ and outputs either a message from $\mathscr{P}$ or an error (signifying decryption failure).

For all messages $m \in \mathscr{P}$, all keys $k \in \mathscr{K}$, and all coin tosses made by $\mathscr{E}$, we must have that

$$\mathscr{D}(k, \mathscr{E}(k, m)) = m.$$

Typically, $\mathscr{P}$ and $\mathscr{C}$ will be subsets of the set of all possible bit strings. $\mathscr{K}$ will typically be the set of all possible bit strings of a given length, say l, and in this case we say that we have a symmetric cryptosystem with key length l.

We say that a symmetric cryptosystem is secure if it provides confidentiality, that is, an adversary that sees an encryption of a message m is unable to learn anything about the message m except possibly its length. And this must hold, even if the adversary has seen encryptions of other messages, some of them possibly influenced or chosen by the adversary.

3.1 Block ciphers

A block cipher is not a symmetric cryptosystem, but a building block that can be used to construct symmetric cryptosystems.

Definition 3. A *block cipher with block length n and key length l* is a pair (g_1, g_2) of functions $\{0,1\}^l \times \{0,1\}^n \to \{0,1\}^n$ such that for all $k \in \{0,1\}^l$ and $x \in \{0,1\}^n$ we have that

$$g_1(k, g_2(k, x)) = x = g_2(k, g_1(k, x)).$$

Another way to express this is to say that $g_1(k, \cdot)$ and $g_2(k, \cdot)$ are inverse permutations on $\{0,1\}^n$. A block cipher is said to be secure if it is hard to distinguish the permutation defined by a random key from a random permutation on $\{0,1\}^n$ for someone who does not know the key.

There are many ways to construct symmetric cryptosystems from a block cipher (g_1, g_2).

Electronic Code Book (ECB) mode For this cryptosystem, we have $\mathscr{P} = (\{0,1\}^n)^* = \mathscr{C}$ and $\mathscr{K} = \{0,1\}^l$.

The encryption algorithm splits the message m of length sn into s bit strings $m_0, \dots, m_{s-1}$ of length n, computes $c_i = g_1(k, m_i)$ for $0 \le i < s$, and outputs the ciphertext $c_0 \parallel c_1 \parallel \dots \parallel c_{s-1}$.

The decryption algorithm splits the ciphertext c of length sn into s bit strings of length n, $c_0, c_1, \dots, c_{s-1}$, computes $m_i = g_2(k, c_i)$ for $0 \le i < s$, and outputs the message $m_0 \parallel \dots \parallel m_{s-1}$.

ECB mode is in general insecure. DO NOT USE IT!

Cipher Block Chaining (CBC) mode For this cryptosystem, we have $\mathscr{P} = (\{0,1\}^n)^* \supseteq \mathscr{C}$, and $\mathscr{K} = \{0,1\}^l$.

The encryption algorithm splits the message m of length sn into s bit strings $m_0, \dots, m_{s-1}$ of length n. Then it tosses n coins to get a random bit string of length n called the *initialization vector (IV)*. Set c_0 to be this bit string. For $1 \le i \le s$, compute $c_i = g_1(k, c_{i-1} \oplus m_{i-1})$. Then output the ciphertext $c_0 \parallel c_1 \parallel c_2 \parallel \dots \parallel c_s$.

The decryption algorithm splits the ciphertext c of length $(s+1)n$ into $s+1$ bit strings of length n, $c_0, c_1, \dots, c_s$, and computes $m_i = g_2(k, c_{i+1}) \oplus c_i$ for $0 \le i < s$, and outputs the message $m_0 \parallel m_1 \parallel \dots \parallel m_{s-1}$.

Note that the IV must be unpredictable for an attacker. Choosing a random string will always work, but there are other possibilities available.

Padding schemes A block cipher used in CBC mode as described above can only encrypt messages whose length is divisible by the block length. To encrypt messages of arbitrary length we need a *padding scheme*. A very simple solution is the following: first add a single 1 to the message, then add 0s until the length of the bit string is divisible by block length. To remove the padding, remove any trailing 0s and then remove the 1. (Note that this will increase the length of the message by at least one bit.)

3.2 Stream ciphers

A stream cipher is simply a pseudo-random bit generator.

Definition 4. A *pseudo-random bit generator* (*PRBG*) with key length l is a deterministic algorithm $\mathscr{G}$ taking as input a key from $\{0,1\}^l$, an initialization vector from $\{0,1\}^n$ and a length N, and outputs a infinite bit string. In practice one only ever uses a finite initial segment of the bit string. We write $\mathscr{G}(k, v; N)$ for the first N bits of $\mathscr{G}(k, v)$, and require that these can be computed in time proportional to N.

The initialization vector is often called a *nonce*. (In general, a nonce is a bit string – or number – that is used only once.)

A PRBG is secure if it is hard to distinguish the output bit strings from random bit strings for someone who does not know the key.

Stream cipher A stream cipher cryptosystem based on a PRBG $\mathscr{G}$ has $\mathscr{P} = \{0,1\}^* \supseteq \mathscr{C}$.

The encryption algorithm takes a key $k \in \{0,1\}^l$ and a message m of length N. It tosses n random coins to get a random bit string iv of length n, and outputs the ciphertext $iv \,\|\, (m \oplus \mathscr{G}(k, iv; N))$.

The decryption algorithm takes a key $k \in \{0,1\}^l$ and a ciphertext of length $n+N$. Denote the first n bits of the ciphertext by iv, and the remaining N bits by r. The algorithm outputs the message $r \oplus \mathscr{G}(k, iv; N)$.

Initialization vector reuse If initialization vectors are reused with a stream cipher (that is, two different messages are encrypted using the same IV and the same key), confidentiality is lost for both messages. If the IV length is sufficiently large (say 128 or more), random IVs will prevent reuse.

Counter mode We can construct a PRBG from a block cipher (g_1, g_2) with block length n and key length l as follows. Compute $pad_i = g_1(k, (iv+i) \bmod 2^n)$ for $i = 0,1,2,\dots$ and output the bit string $pad_0 \,\|\, pad_1 \,\|\, pad_2 \,\|\, \cdots$.

(Note that we can consider bit strings to be numbers, and vice versa. So adding the integer i to the bit string iv makes sense. And when we take the remainder when divided by 2^n, the resulting integer can be represented by exactly n bits.)

In the same fashion, we can construct a PRBG from a suitable hash function h with output length n as follows. Compute $pad_i = h\big(k \,\|\, ((iv+i) \bmod 2^n)\big)$ for $i = 0,1,2,\dots$ and output the bit string $pad_0 \,\|\, pad_1 \,\|\, pad_2 \,\|\, \cdots$.

4 Integrity

In general, encryption does not provide integrity. The best counter-example is a stream cipher, where it is trivial to "flip bits". We preserve integrity using a message authentication code (often called a keyed hash function).

Definition 5. A *message authentication code (MAC)* on a set of plaintexts $\mathscr{P}$ with key length l and tag length n is a function $f\colon \{0,1\}^l \times \mathscr{P} \to \{0,1\}^n$.

Sometimes, a MAC is defined to also take a *nonce*. The MAC value of a given message is called a *tag*.

A MAC is secure if, given many message-tag pairs $(m_i, t_i) \in \mathscr{P} \times \{0,1\}^n$ such that $f(k, m_i) = t_i$, it is hard for someone who does not know k to find a pair $(m, t) \in \mathscr{P} \times \{0,1\}^n$ such that $f(k,m) = t$ and $m \neq m_i$ for all m_i.

A MAC can be used alone to provide integrity to communications, without providing confidentiality. It can also be used with a symmetric cryptosystem to provide both confidentiality and integrity.

CBC-MAC We can construct a MAC from a block cipher (g, h) with key length l and block length n as follows. The set of plaintexts $\mathscr{P}$ is $(\{0,1\}^n)^*$. Split the message m into s blocks $m_0, m_1, \dots, m_{s-1}$ of length n. Set y_0 to be the string of n zeros, and define $y_{i+1} = g(k, m_i \oplus y_i)$. Then $f(k,m) = y_s$.

Note that this construction is not secure by itself, but needs a sophisticated padding scheme to be secure.

HMAC A hash function is a function $h : \{0,1\}^* \to \{0,1\}^n$. For some hash functions, the following construction yields a MAC $f\colon \{0,1\}^l \times \{0,1\}^* \to \{0,1\}^n$ called *Hash-MAC* or *HMAC*:

$$f(k,m) = h\big((k \oplus opad) \,\|\, h(k \oplus ipad \,\|\, m)\big).$$

ipad and *opad* are suitable constants.

5 Secure channel

The simple way for Alice and Bob to secure their communications over the insecure channel is to use a symmetric cryptosystem $(\mathscr{E},\mathscr{D})$ with key length l_1 and a MAC $f\colon \{0,1\}^{l_2} \times \{0,1\}^* \to \{0,1\}^n$ as follows.

The shared secret is a pair (k_1, k_2) of randomly and independently chosen bit strings of length l_1 and l_2.

When Alice wants to send the message m to Bob, she first encrypts it using the cryptosystem and k_1: $c \leftarrow \mathscr{E}(k_1, m)$. Then she computes a tag for the ciphertext using the MAC and k_2: $t \leftarrow f(k_2, c)$. Then Alice transmits the pair (c, t) over the insecure channel.

To read the message contained in the pair (c, t), Bob first verifies the tag t by checking that $t = f(k_2, c)$. If the tag is incorrect, Bob discards the pair (c, t). Otherwise, he decrypts the ciphertext: $m \leftarrow \mathscr{D}(k_1, c)$ and reads the message m.

Typically, there will be two different and independent shared key pairs (k_1, k_2) and (k_1', k_2'), one for messages from Alice to Bob and one for messages from Bob to Alice.

Remark Anyone who can listen to the insecure channel and insert data can now send the message m to Bob again and again, simply by sending the pair (c, t) to Bob over the channel. How best to prevent this depends on how the secure channel is used and is beyond the scope of this note.

Remark This combination of cryptosystem and MAC can be taken as a new cryptosystem. This cryptosystem will then provide confidentiality and integrity. Other cryptosystems provide both confidentiality and integrity by design, e.g. the block cipher modes GCM and CCM.