6.857 Computer and Network Security: Lecture Notes (Draft)

Part 3 : Unconditional Security

Version: September 17, 1998

We begin our discussion of cryptographic techniques by looking at techniques that achieve *unconditional security*—security that does not depend upon an assumption that the attacker has limited computing power. Techniques for unconditional security are sometimes called *information-theoretic* techniques, since they are based on the fact that the attacker doesn't get enough *information* to break the system.

Unconditional security methods, though excellent in theory, are hard to implement or execute efficiently in practice, since they require the generation, distribution, and management of large amounts of key material, which must be kept secret. As we shall see, the key material can be used only once, otherwise a loss of security results. The more you communicate, the more key material you need. Hence the name "*one-time* pad." The term "pad," referring to the key itself, may have originated from the way Russian spies kept their key material on preprinted pads of paper; when they were done with the digits on one sheet they would destroy that sheet and move on to the next sheet in the pad.

Later on, we shall examine schemes that are conditionally secure—they are secure on the assumption that the attacker has limited computing power. Such schemes are also called *computationally secure.* Most practical cryptographic techniques, such as DES or RSA, are only computationally secure; they are preferred in practice because they greatly simply key generation and management, compared to unconditionally secure schemes. Nonetheless, it is instructive to look at uncondtionally secure cryptosystems first, as a model for ideal cryptography.

1 The One-Time Pad (OTP) Encryption Scheme

Our problem is to design a secure method to deliver a message $M \in \{0,1\}^n$ (an n bit string) from Alice (A) to Bob (B) *securely.* An eavesdropper Eve (E), should not learn anything more about M.

Basic question: What distinguishes Bob from Eve? What allows him to understand a ciphertext that Eve can't?

Answer: Bob should possess something (an object) or know something (by definition, a *key*) which Eve doesn't.

It is usually easier to arrange for Bob to know a key than to arrange for him to have a object (e.g. a decoding box) that Eve can't have. For one thing, it is easier to change the key than to manufacture and re-distribute new decoding boxes. In classical cryptography, the key (K) is known to both (and only) Bob and Alice, and is thus called a *shared secret key.*

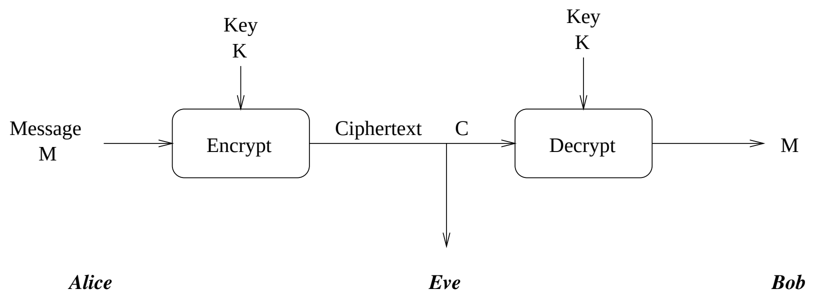

Figure 1: Sending a message over the network.

This raises several problems:

Encryption: How is the key used to provide secure confidential transmission of a message from Alice to Bob?

Key Generation: How is the key generated?

Key Distribution: How is the key distributed to Alice and Bob without Eve getting it?

1.1 Encryption with the One-Time Pad

For the encryption and decryption operations, a *one time pad* is used (Vernam, 1917).

Example: Suppose Alice wishes to send Bob the message

$$M = 011010 \ ,$$

and supposed they have previously established a shared secret (and randomly chosen) key

$$K = 101100 \ .$$

Note that the key has the same length as the message; this is required. The ciphertext C is now defined as the bit-wise exclusive-or (xor, or $\oplus$) of the message and the key:

$$C = M \oplus K = 110110 \ .$$

Decryption is trivial since $\oplus$ is associative, and each element is its own inverse; exclusive-oring the key back into the ciphertext yields the message again:

$$C \oplus K = (M \oplus K) \oplus K = M \oplus (K \oplus K) = M \oplus 0 = M \ .$$

The basic fact about the one-time pad is the following. We shall give an example illustrating its meaning, and then prove it as a theorem.

Claim 1 *Seeing the ciphertext C does not increase the adversary's knowledge about the underlying message M (assuming that the pad is randomly chosen and only used once).*

Eve may know something about M beforehand, such as that M is in English. It is worth noting (but not really sufficient to prove the claim) that a given ciphertext C can be the encryption of *any* message M—just take the key to be $K = C \oplus M$, so that $C = M \oplus K$.

To illustrate the claim, suppose that the message M is either 0 or 1, and suppose Eve knows *a priori* that the chance that Alice will pick $M = 0$ is $2/3$, and the chance that Alice will pick $M = 1$ is $1/3$. More formally, we have unconditional a priori probabilities $P(M = 0) = 2/3$, $P(M = 1) = 1/3$.

Suppose now that Eve overhears the ciphertext $C = 1$. For Eve the conditional probability of $M = 0$, having now seen that $C = 1$, is

$$\begin{aligned} P(M = 0 | C = 1) &= \frac{P(M = 0 \wedge C = 1)}{P(C = 1)} \\ &= \frac{1/3}{1/2} = 2/3 \; . \end{aligned}$$

(See the circled portion in figure 2.) Since the conditional probability of $M = 0$, having seen the ciphertext $C = 1$, is the same as the unconditional (a priori) probability that $M = 0$, Eve has not learning anything about M by seeing the ciphertext C. In other words, Eve's probability distribution on M (reflecting her state of knowledge about M) has not changed, and so seeing C has taught Eve nothing at all. Anything Eve knows after seeing C it already knew before seeing C; such knowledge was represented in her a priori probability distribution.

We now give a formal proof of the claim.

Theorem 1

$$P(M = x | C = y) = P(M = x)$$

Proof: Assume that M and C are n bits long. For any n-bit values x and y

$$P(M = x | C = y) = \frac{P(M = x \wedge C = y)}{P(C = y)} \; .$$

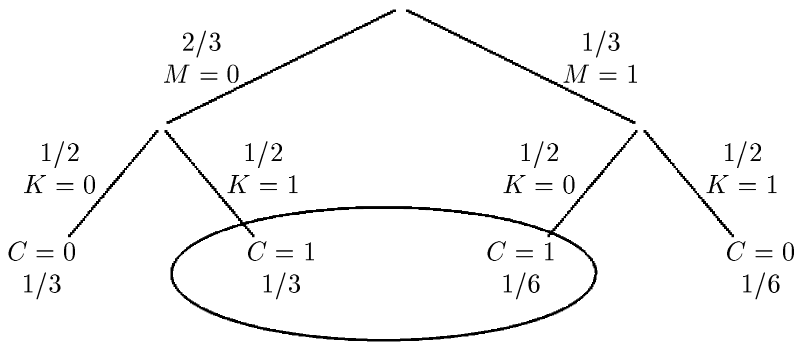

Figure 2: Probability tree for ciphertexts

Now

$$\begin{aligned} P(M = x \wedge C = y) &= P(M = x \wedge K = (x \oplus y)) \\ &= P(M = x) \cdot P(K = (x \oplus y)) \quad \text{(K is independent of M)} \\ &= P(M = x) \cdot 2^{-n} \quad \text{(K is chosen uniformly from bit strings of length } n\text{)} \end{aligned}$$

and

$$\begin{aligned} P(C = y) &= \sum_x P(M = x \wedge C = y) \\ &= \sum_x P(M = x) \cdot 2^{-n} \\ &= 2^{-n} \sum_x P(M = x) \\ &= 2^{-n} \end{aligned}$$

That is, each C is equally likely, independent of the message. So,

$$\begin{aligned} P(M = x | C = y) &= \frac{P(M = x) \cdot 2^{-n}}{2^{-n}} \\ &= P(M = x) \end{aligned}$$

which was to be shown. Note that the proof depends critically on the assumption that the key K is generated according to the uniform distribution; the theorem is otherwise false. ■

Thus, the one-time pad provides perfect secrecy (unconditional confidentiality).

1.2 The Two-Time Pad?

There are some problems with the one-time pad. Note that each key can only be used *once*; if K is reused, security may be compromised. For example, suppose that M_1 and M_2 are encrypted with the same key K to generate ciphertexts C_1 and C_2:

$$\begin{aligned} C_1 &= M_1 \oplus K \\ C_2 &= M_2 \oplus K \\ C_1 \oplus C_2 &= (M_1 \oplus K) \oplus (M_2 \oplus K) \\ &= (M_1 \oplus M_2) \oplus (K \oplus K) \\ &= M_1 \oplus M_2 \end{aligned}$$

An adversary can thus take the xor of two ciphertexts to obtain the xor of the two original messages, since the key cancels! When the messages have low entropy (lots of redundancy) as do English messages, it is often possible to figure out *both* messages from their xor. It is worth noting, however, that if M_1 and M_2 are *compressed* before encryption, it is much harder to figure out the original messages from the xor of the two compressed messages.

If you think that using a one-time pad more than once is too dumb to happen in real life, you may be interested in the recently declassified story of project Venona (see reference). During the cold war apparently the Russians were reusing pads (!) for some of their encrypted communications. The National Security Agency somehow (?) found out and started a massive effort to decrypt the messages. This wasn't easy, since the messages were already encoded once (with a codebook scheme) before being encrypted with a one(two)-time pad. By collecting a huge number of ciphertexts and looking for dependencies they were able to decrypt a good portion of the messages. It is still not known how the Russians made such a fundamental error.

Another problem is that OTP is *malleable*; an adversary can change C so that Bob's decrypted M is different from the message that Alice sent in a predictable way: by changing the i-th bit of C Eve causes the i-th bit of Bob's decrypted M to change. There's no way, using just a OTP, for Bob to check that he received the message exactly as Alice sent it. The OTP provides *confidentiality* but not *authentication*.

Finally, note that the key is both large (as large as the message) and used only once. For high data-rate application (e.g. video) you would have to pre-generate and distribute *huge* amounts of key!

1.3 Key Generation—Generating Random Bits

As seen above, generating long sequences of random bits is important for use of the one-time pad. Indeed, any sort of cryptography depends on the generation of random bits—that is the traditional way of generating a secret that you know but the attacker doesn't. But doing this well can be tricky or difficult. Only recently, a security bug in Netscape was found by graduate students at Berkeley; apparently, the keys used by Netscape were not long enough, and their source of "randomness" left much to be desired. (See `http://www.ddj.com/ddj/1996/1996_01/wagner.htm`)

Sources of randomness:

- Coin tosses
- Radioactive decay
- Typing speed of a user at a terminal
- Thermal noise
- Digitized images of Lava lamps (See `http://lavarand.sgi.com/`; this is cool!)
- Noisy diode (perhaps the most traditional approach)
- Background radiation
- Hard disk speed variation
- Input from a microphone or an antenna
- Other physical chaotic systems

These are excellent sources of randomness (some better than the others) but they may be difficult to realize in practice. A typical PC is not supplied with a radioactive sample, nor equipped with a Geiger counter.

If your source of randomness is biased or otherwise not quite perfect, processing of the sampled bits can improve the source. For example, suppose the source you have provides independent bits, but that they are not evenly biased (e.g. 0 is more frequent than 1). A fix proposed by von Neumann is to generate *two* bits at a time; return 0 if 10 is detected, return 1 if 01 is detected and resample if 00 or 11 is detected (throw these away). Even if 0 is more frequent than 1, 01 is just as frequent as 10, so the bias in the sample is removed, at a small cost in data rate.

See `http://www.cs.berkeley.edu/~daw/netscape-randomness.html` for many other links to documents on generating random bits.

Another common approach to generating "random" bits is to use a *pseudo-random number generators* (PRNGs). A PRNG generates a long n-bit sequence from a shorter (random!) seed, using a suitably complicated process. However, the result is certainly not more secure than the short seed (e.g. a 56-bit seed might be determined by a search), and may be weaker if the PRNG has other weaknesses. (It is usually assumed that the adversary has access to the function, but not the seed.) You should convince yourself that the theorem above on the security of the one-time pad doesn't apply when the pad is generated deterministically from a smaller seed.

1.4 Length of One-Time Key k

Is it possible to shorten the length of the one-time key K, $|K|$, needed for confidentiality? To do so, we must start making assumptions about computational difficulty.

Alice encrypts a message M that is n bits long with a key K that is t bits long, resulting in ciphertext C, which is also n bits long. Imagine that Eve hears C and has an infinite amount of computing power. Eve tries all 2^t keys and examines the resulting plaintexts M; she then gets 2^t candidate messages. Assume that the number of reasonable messages of length M is α^n $(1 \leq \alpha < 2)$ (e.g. for English $\alpha \approx 1.1$). The expected number of keys giving a reasonable result is thus:

$$2^t(\alpha/2)^n$$

Thus, when $t + n(\lg(\alpha) - 1) < 0$ then the number of keys giving reasonable results ≤ 1. When

$$n > t/(1 - \lg(\alpha))$$

then there is probably only one key that works ($n > 7t$ for English). This argues that unconditional security requires growing the length of the keys with the length of the message. Thus, unconditional security requires *long* one-time keys.

2 Unconditionally Secure Authentication

A central problem in network security is *authentication*: how can Bob be sure that the message he has received was actually sent by Alice, and if so, was not corrupted en route (either randomly by communication bit errors, or maliciously by Eve)?

(In this section we are not concerned with confidentiality, just with authentication.)

The answer lies in use of a *message authentication code*, (aka *MAC* or *authentication tag*). A MAC y is a function of both the message M and a shared secret key K that is known to only Alice and Bob:

$$y = f(M, K) \ .$$

The function f is public.

Alice computes the tag y and transmits the pair (M, y) to Bob. Bob uses his knowledge of the secret key K to recompute $y = f(M, K)$. If his recomputed y is the same as the received y, Bob concludes that the message is authentic. If, on the other hand, Eve had interfered and changed M to M', Bob would compute $y' = f(M', K)$, and, finding it different than y, conclude that the message has been forged or corrupted. Similarly, if Eve tries to modify the message and also make corresponding modifications to the MAC, she should have little chance of success.

A MAC is similar to a CRC (cyclic redundancy check) except that the CRC is only useful for detecting random errors; an adversary can easily find an M' that has the same CRC has a given message M. MACs are designed to defeat active adversaries as well as random errors.

The message and its MAC may be encrypted (using another key) if confidentiality as well as authentication is desired; this is discussed later.

To achieve unconditional authentication it is important that a new key K be used every time as we shall see. This also helps prevent replay attacks: Eve cannot resend the message the next day; Bob will not accept it.

GOAL: Eve should not be able to generate a valid pair $(M', f(M', K))$ even after hearing one valid pair $(M, f(M, K))$.

IDEA: The one-time key K is a pair (a, b) of coefficients: $K = (a, b)$ and then we define

$$f(M, K) = y = aM + b$$

Having seen one (M, y) pair is of no help in figuring out the appropriate y' for a different message M'. *One point does not determine a line.*

More formally:

We use some number theory; see (Cormen, 1990) for an algorithmic treatment of basic number theory.

Let p be a prime. Then $Z_p = \{0, 1, ..., p-1\}$ forms a *finite field* modulo p with respect to the four standard operations $+, -, *, /$, so that (almost) all the usual properties of the real numbers hold:

- $+$ is associative and commutative with identity 0
- $*$ is associative and commutative with identity 1
- all elements have additive inverses: $-5 \equiv 12 \pmod{17}$
- all nonzero elements have multiplicative inverses: $5^{-1} \equiv 7 \pmod{17}$ (multiplicative inverses can be computed using an extended version of Euclid's GCD algorithm; see (Cormen, 1990))
- division: $2/5 \equiv 2 * 7 \equiv 14 \pmod{17}$ since $5^{-1} \equiv 7 \pmod{17}$

- $*$ distributes over $+$: $a * (b + c) \equiv ab + ac \pmod p$.

(But not all properties of the reals carry over to a finite field; for example in a finite field you can add 1 to itself some nonnegative number of times and get 0: for example, $1 + 1 + 1 \equiv 0 \pmod 3$.)

We suppose that p is a large publicly-known prime. Suppose $|M| = n$ (that is, M has n bits), and suppose further that $|p| > n$.

Now choose a and b uniformly at random from Z_p, and define

$$\begin{aligned} K &= (a, b) \\ f(M, K) &\equiv aM + b \pmod p \ . \end{aligned}$$

$|f(M, k)| = |p| \geq |M|$; the MAC is at least as long as the message.

Suppose Eve hears (M, y), where $y \equiv aM + b \pmod p$. What has she learned? She knows M, y, and p, but not a and b. There are p pairs of coefficients (a, b) that are consistent with the equation $y \equiv aM + b \pmod p$. Suppose Eve picks some $a_i \in Z_p$. Then choosing $(a, b) = (a_i, b_i)$ satisfies the above equation, where $b_i \equiv y - a_i M \pmod p$. All such choices (a_i, b_i) are equally likely, as far as Eve knows.

Suppose Eve intercepts M and replaces it with another message M' that she wishes Bob to accept. She must compute a new MAC that Bob will accept for M'. She guesses a_i and computes $b_i \equiv y - a_i M$ to compute a new MAC using the guessed key $K_i = (a_i, b_i)$:

$$\begin{aligned} f(M', K_i) &\equiv a_i M + (y - a_i M) \pmod p \\ &\equiv a_i(M' - M) + y \pmod p \end{aligned}$$

But each possible choice of an a_i gives a different MAC, since

$$a_i(M' - M) + y \equiv a_j(M' - M) + y \pmod p$$

implies

$$a_i \equiv a_j \pmod p \ .$$

Thus, $f(M', K_i)$ is equally likely to be any value in Z_p. (While we have argued this for only one strategy for Eve—guessing a and computing b—it holds in general.) Therefore, Eve's chance of getting the MAC right is only $1/p$.

A Two-Time MAC?

What happens if Alice and Bob use the same secret key K to compute the MAC's for *two* distinct messages M and M'? Thus:

$$\begin{aligned} y &\equiv aM + b \pmod p \\ y' &\equiv aM' + b \pmod p \end{aligned}$$

But now Eve has two equations in two unknowns, and she can solve for a and b:

$$\begin{aligned} a &\equiv (y - y')/(M - M') \pmod p \\ b &\equiv y - aM \pmod p \ . \end{aligned}$$

Then Eve knows both a and b, and could replace (M', y') with a valid (M'', y''). Therefore, *do not* reuse a key.

Length of the MAC

Both the key and the MAC have length proportional to $|M|$. Having $|K| = |M|$ is unavoidable, but it is possible to improve the length of the MAC, with only a small loss in security.

Ideas to shorten MAC:

- only send low-order 64 bits of $y \equiv aM + b \pmod p$
- choose small p (64, 65 bits)
  divide M into chunks $M = M_1, M_2, ..., M_t$ where $0 \leq M_i < p$
  $K = (a_1, a_2, ..., a_t, b)$ where $0 \leq a_i < p$, and $0 \leq b < p$
  $f(M, K) \equiv \sum a_i M_i + b \pmod p$

3 Simultaneous Privacy and Authentication

Previously, we were only concerned with authentication. Alice was sending her message M cleartext over the channel, and Eve could read it. Now we want to consider the case where Alice wants her message to be private so that only Bob can read it, *and* she wants to authenticate the message. Now we combine the One-Time Pad (OTP) and the One-Time MAC (OTM). It is important to use one key K for confidentiality, and a second independent key K' for authentication. We have two choices:

- encrypt M with OTP, then append MAC of ciphertext:
  $$(M \oplus K, f(M \oplus K, K'))$$
- append MAC to M, use OTP to encrypt (M, y) pair:
  $$(M, f(M, K')) \oplus K$$

There are two advantages of the first choice, where we encrypted the message M and then appended the MAC of the ciphertext. First, Bob can check the MAC before decrypting. If it fails, then he does not have to waste time decrypting $M \oplus K$. Second, encrypting the message first uses fewer random bits (assume that the message has length n and that a MAC has length m):

- $M \oplus K, f(M \oplus K, K')$
  $|K| = n$
  $|K'| = 2n$
  Total bits of key used $= n + 2n = 3n$.

- $(M, f(M, K')) \oplus K$
  $|K| = 2n$
  $|K'| = 2n$
  Total bits of key used $= 2n + 2n = 4n$

4 References

Cormen, Thomas H., Charles E. Leiserson, and Ronald L. Rivest. *Number Theoretic Algorithms*, Chapter 33 of *Introduction to Algorithms* (MIT Press/McGraw-Hill, 1990).

NSA. *The VENONA Project.* `http://www.nsa.gov:8080/docs/venona/venona.html`.

Vernam, Gilbert S. *Secret Signaling System.* U.S. Patent 1,310,719. (Issued July 22, 1919).