6.857 Computer and Network Security	Fall Term, 1997
Lecture 2 : September 9, 1997	
Lecturer: Ron Rivest	*Scribe: Zia Ahmed*

1 Unconditional Security

This course is primarily about computer security, and involves only some cryptography. Unconditional security methods, though excellent in theory, are hard to implement or execute efficiently in practice. Unconditional security implies that no assumptions are made about the computing power and resources available to an adversary. This is often overkill, since an adversary may well be limited by time (computing time) or space (memory). However, a study of unconditional security methods is definitely a prerequisite to the study of more practical and conventional methods of network and computer security. Some methods are discussed below.

1.1 One Time Pad (OTP)

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

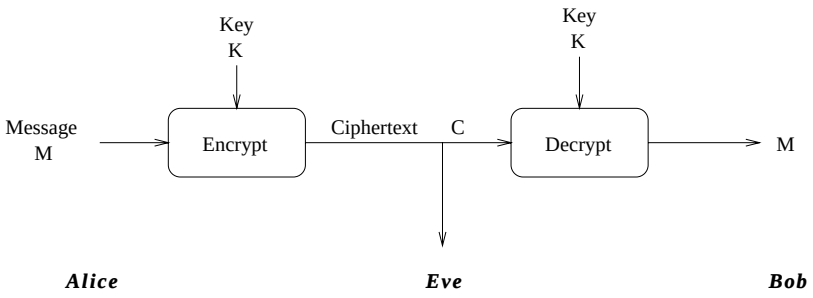

Figure 1: Sending a message over the network.

IDEA: Bob should have something or know something which distinguishes him from Eve. In classical cryptography, this is called a *shared secret key* (K). This key is

known to both Alice and Bob, but not to Eve. The secret key raises two problems:

Generation: How is the key generated? (someone makes up K at random or uses a convoluted deterministic generator).

Distribution: How is the key distributed to Alice and Bob without Eve getting it? (the key is shipped to Alice and Bob on a disk or CD-ROM prior to the message being sent).

For the encryption and decryption operations, a *one time pad* is used (Vernam, 1913):

Example: Let the message $M = 011010$. Let the key $K = 101100$ (generated at random, such that the length of the key equals the length of the message).

Now, the ciphertext, $C = M \oplus K = 110110$, where $\oplus$ is the bit-wise exclusive-or operator.

Decryption is trivial since $\oplus$ is associative, and each element is it's own inverse.

So, $C \oplus K = (M \oplus K) \oplus K = M \oplus (K \oplus K) = M \oplus 0 = M$

Theorem 1 $P(M = x|C = y) = P(M = x)$

Proof: *Assume that M and C are n bits long.*

Now, $P(M = x|C = y) = \frac{P(M=x \wedge C=y)}{P(C=y)}$

$P(M = x \wedge C = y) = P(M = x \wedge K = (x \oplus y))$

$= P(M = x).P(K = (x \oplus y))$ *(K is independent of M)*

$= P(M = x).2^{-n}$ *(K is chosen uniformly from bit strings of length n)*

Also, $P(C = y) = \sum_x P(M = x \wedge C = y)$

$= \sum_x P(M = x).2^{-n}$

$= \sum_x P(M = x)$

$= 2^{-n}$ *(that is, each C is equally likely).*

So, $P(M = x|C = y) = \frac{P(M=x).2^{-n}}{2^{-n}}$

$= P(M = x)$ ■

There are some problems with this approach. Note that each key can only be used *once*; in the case where K is reused, security may be compromised. As an example, suppose that M_1 and M_2 are encrypted with the same key K to generate ciphertexts C_1 and C_2.

Now, $C_1 = M_1 \oplus K$ and $C_2 = M_2 \oplus K$.

$C_1 \oplus C_2 = (M_1 \oplus K) \oplus (M_2 \oplus K)$

$= M_1 \oplus M_2$

Depending on the entropy of the message, an adversary can use this to figure out both messages.

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. There's no way for Bob to check if Alice sent exactly the same message that he received. Thus, OTP provides *privacy* but not *authentication*.

1.2 Generating Random Bits

As can be seen from the previous section, generating large sequences of random bits is very important. Practical cryptography calls for long keys, the generation of which may be non-trivial. 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.

Sources of randomness:

- Toss coins
- Radioactive Decay
- Typing speed of a user at a terminal
- Thermal noise
- Lava lamps
- Noisy diode
- Background radiation

- Hard disk speed variation
- Other physical chaotic systems

These are excellent sources of randomness (some better than the others) but they are difficult to realize in practice. An ordinary PC is not supplied with a radioactive sample, nor equipped with a Geiger counter to detect randomness at the quantum level.

It is also possible to remove bias from a raw sample. Suppose that a random sample has a bias towards pairs of bits (ie. 11 and 00 occur together more frequently than singly). A possible fix to this (as proposed by von Neumann) is to generate two bits at a time; return 0 if 10 is detected, 1 is 01 is detected and redo the experiment if 00 or 11 is detected. Thus, the natural bias in the sample is removed.

Other possible sources of (pseudo) randomness are deterministic sources; the so-called *pseudo-random number generators* (PRGs). A PRG generates a long n-bit sequence from a short seed, which is typically provided by the user or the system clock. The bit generation involves a long and convoluted function; however, since an adversary may gain access to this function, the use of a PRG is not entirely secure. Since the PRG uses a deterministic source, the security proof of the earlier section is no longer applicable.

1.3 Message Authentication Code (MAC)

A key problem in network security is that of *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 by Eve, the adversary?

The solution lies in the message authentication code (MAC). The MAC is a function of the message M and the secret key K. Thus, the MAC $y = f(M, K)$. The function f is public. Alice appends the MAC y at the end of the message M. Bob, upon receiving the string $< M, y >$, uses his secret key and the function f to compute $f(M, K)$. If this is the same as y, Bob concludes that the message is authentic. If on the other hand Eve interferes and changes M to $M^{'}$, Bob computes $f(M^{'}, K)$ and on finding that this is different from y, concludes that the message has been corrupted by Eve. This is similar to a CRC with the difference that the problem here is that of an actual adversary, and not accidental corruption of the message. Depending on the implementation, the MAC may or may not be encrypted; Alice may either send $< M \oplus K, y >$, or $< M, y > \oplus K$. Both methods have their pros and cons.