This lecture introduces the concepts that will lead to formal definitions of secure “secret” communication. A basic model for two communicating parties is introduced, and several important observations arise from the requirement for secret communication between them. Several classical attempts to solve this problem are shown to fail, and finally a working system known as the “One Time Pad” that achieves perfect secrecy is introduced.

1 DESCRIPTION OF PROBLEM

This section taken from the lecture notes for the graduate version of the course.

Assume there are three agents, Bob, Alice, and Eve. Bob wants to send Alice a private letter that only Alice can read. Eve is an adversary who may intercept the letter, but reading it should not enable her to reconstruct Bob’s message to Alice. This is the essence of the problem of secure communication. A more formal definition of “secure” will be given later.

To meet these requirements, Bob must somehow alter his message to Alice so it cannot be understood by anyone else. This alteration is called encryption. If \( m \) is the message to be encrypted, (also known as the “plaintext” or the “cleartext”) then \( c \), the encrypted message, (also known as the “ciphertext”) is produced by an encryption function \( E(m, ???) \rightarrow c \), which Bob uses. Alice uses a symmetric decryption function \( D(c, ???) \rightarrow m \).

The question marks in the function definitions stand for other pieces of information that may be provided to the encryption/decryption functions, depending on the specific method being used. Some possibilities are:

- \( S_b \): A piece of secret information known to Bob but not to Eve.
- \( S_a \): A piece of secret information known to Alice but not to Eve.
- \( PK \): A piece of public information, available to everyone.
- \( R \): A random factor, presumably drawn from some pseudo-random distribution. This variable is only applicable on Bob’s end; Alice deterministically decrypts the ciphertext (since we want errorless communication).

Notice, Eve does not know \( S_a \) or \( S_b \), even though \( S_a \) and \( S_b \) could have a common piece of shared secret \( s \). This \( s \) is not known to Eve, but is known to both Alice and Bob.

2 INITIAL ASSUMPTIONS

In order to start to formalize notions of security for \( E \) and \( D \), we will start by making the following assumptions:
A) Eve has unbounded computational resources (i.e. unlimited computing time and memory).

B) Kerckhoffs Principle: Relying on the secrecy of an algorithm is unacceptable. This means
details of all the algorithms should be publicly known (in particular, $E$ and $D$ are public, so
Eve can use them).

C) Correctness: When Alice decrypts the ciphertext from Bob, the message she obtains is always
the one Bob sent her, i.e. $m = D(E(m, ????), ????)$.

3 OBSERVATIONS

This section taken from the lecture notes for the graduate version of the course.

Some observations follow from these assumptions:

1. **Alice must have a secret from Eve.**
   
   *Rationale:* Alice must be able to decrypt the message from Bob. If Eve knows everything
   Alice knows, then Eve is functionally equivalent to Alice. Namely, if Alice can read the
   message then so can Eve. This contradicts the problem definition.

2. **Bob must have a secret from Eve. In fact, Alice and Bob must share a secret $s$ not known
to Eve.**
   
   *Rationale:* Let Bob’s message be $m$, and let $X$ be all the other information he uses to produce
   $c$ (includes his randomness, possible secret key, etc.). By Assumption A, Eve has the computa-
tional power to test all candidate values $m'$ for $m$ and $X'$ for $X$, until $E(m', X') = c$. When
   this happens, Eve is sure that $m = m'$ unless Alice knows a part of $X$. Indeed, otherwise and
   if $m$ is different from $m'$, from Alice’s point of view the message could be both $m$ and $m'$,
   which contradicts unique decodability (correctness from Assumption C). Thus, the only way
   Eve should still be unsure if $m = m'$ is if Alice knows a part of $X$, i.e. Alice and Bob share
   a non-empty secret $s$ not known to Eve.

4 EARLY ATTEMPTS

The need for cryptographic techniques to disguise sensitive communications existed long before the
advent of computers. It is fabled that Julius Caeser used the following scheme, known as the Caeser
cipher, to send secret orders to his generals:

**Definition 1 (Caeser Cipher)** Number all the letters in the alphabet sequentially, 0 to 25. Alice
and Bob choose a random secret key $k \in \{0, \ldots, 25\}$ which will their shared secret $s$. Use the
following $E$ on the message one letter ($l$) at a time:

$$E(l, k) = l + k \mod 26$$
Decryption is the same as encryption, only using subtraction $\text{mod}26$ instead of addition. Intuitively, this cipher takes the letters in the message and shifts them some number of positions through the alphabet, wrapping around the end if necessary. For example, with $k = 3$, encrypting the letter A produces ciphertext letter D, a B would produce an E, and so on. Encrypting the letter X would produce an A, since the shift would wrap around. Decryption is just reversing the shift.

The problem with this scheme is that there are only 26 possible values for $k$. Eve can simply try all 26. If the original message was $n$ letters long, there were $26^n$ possible messages that Bob could have sent to Alice. Without seeing the ciphertext, Eve would have had to guess which of the $26^n$ possible messages was actually sent. Now, even only has to guess which of 26 messages was sent. Clearly, Eve has a learned a great deal of information about the message, provided that $n > 1$. It is interesting to note that for $n = 1$, this scheme works, and Eve has learned nothing about the message... but Bob was only able to send one character.

Another failed attempt to achieve secret communication is known as the substitution cipher.

**Definition 2 (Substitution Cipher)** Choose a shared secret $s$ to be a random permutation $\Pi \in S_{26}$ (where $S_{26}$ denotes the space of all possible permutation of the 26 letters in the alphabet). Use the following $E$ on the message one letter ($l$) at a time:

$$E(l, \Pi) = \Pi(l)$$

Decryption is performed by applying the inverse permutation $(\Pi^{-1})$ to the message one letter at a time. Intuitively, the original alphabet gets mapped into a scrambled alphabet which serves as the secret key. Each letter in the message will be encrypted simply by replacing the letter with its corresponding letter in the scrambled alphabet.

This scheme also has a serious problem, but of slightly different nature. Now, the number of keys is $26! \approx 2^{80}$, which is fairly large to go through by brute force. However, a trivial statistical attack allows one to break the scheme. For example, if encrypting English text, the letter $e$ occurs most frequently. Thus, if Bob encrypted a large section of “random” English text, Eve can simply look at the most frequent ciphertext character and deduce that this is most likely corresponds to $e$. Utilizing these and other such statistics (i.e., which letter is most likes to follow $e$; perhaps it is $s$), it is very easy in practice to get the whole table.\footnote{This method was brilliantly illustrated by Sir Arthur Conan Doyle in “The Adventure of the Dancing Men.”} Thus, even though brute force is not applicable, a very simple method breaks the scheme.

We already start to see some problems with these heuristic approaches:

- How do we know if what we constructed is the right thing? Maybe it resists brute force key search and statistical attack, but there is another smarter attack? When do we stop?
- In fact, what are the legal attacks of the adversary?
- More generally, what is the formal definition of “secure encryption”?

Indeed, once we have a definition and a candidate scheme, we can prove once and for all that the scheme satisfies the definition. Then, we know that the only way the attacker can “break” the scheme is to do something not allowed by the definition, so we might need to revisit the definition, but not the particular scheme. In general, once we agree on the definition and prove some scheme satisfies it, we can sleep well knowing that the scheme is “secure”!
This is what we will do, but first we define the specific scheme which allows provides perfect secrecy for messages of arbitrary length, but at the expense of practicality, since the shared secret must be at least as long as the total message length.

5 One Time Pad

This section taken from the lecture notes for the graduate version of the course.

Recall, we have a pattern for encryption where Bob and Alice must have a shared secret \( s \). We next describe a specific implementation of such an encryption system, called the One Time Pad (OTP). (Note that OTP is a specific scheme for achieving our goal but not the only such scheme possible.) In One Time Pad, we assume that the message \( m \) is somehow chosen from the domain \( \mathcal{M} = \{0,1\}^k \), while a shared secret key \( s \) is chosen at random from the domain \( \mathcal{S} = \{0,1\}^k \). Notice, in OTP Alice and Bob do not need any other secret information (beside the shared \( s \) which is secret from Eve).

**Definition 3** The One Time Pad (OTP) encryption function is easily described; simply take the exclusive OR of the message string and the key \( s \). This produces the cipher \( c \), which can be decrypted by XORing it with the key \( s \):

\[
E(m, s) = m \oplus s \\
D(c, s) = c \oplus s
\]

We see that the One Time Pad is much like the Caesar Cipher when the message contains exactly one letter. The secret key for the Caesar Cipher is a number from 0 to 25, which actually corresponds to a letter of the alphabet. Intuitively, OTP is good because every message \( m \in \mathcal{M} \) and ciphertext \( c \in \mathcal{C} \) correspond to the unique key \( s = m \oplus c \). So without the key, all messages are indistinguishable to Eve.

Before verifying this, though, we remark why the scheme is called “one-time”. So what if we encrypt another message with the same key? The adversary then gets \( c_1 = m_1 \oplus s \) and \( c_2 = m_2 \oplus s \) which allows him/her to get \( c_1 \oplus c_2 = m_1 \oplus m_2 \). And this is partial information about the messages! Thus, while it seems (and we will soon prove it) one-time pad can be used once, it cannot be used twice with the same key. This means that to encrypt multiple messages different one-time pads are needed! Moreover, each one-time pad is as long as the message, so the total secret is as long as the total communication!

Still, we show that except for this (very important) practical issue of the key length being the same as the message length, the One Time Pad to be a “good” one-time scheme.

6 Defining Perfect Security

This section taken from the lecture notes for the graduate version of the course.

As we argued, but we’d like stronger and formal verification that OTP is secure. So what do we mean when we say that a system is secure (at least for encrypting the message once)? By “system” we don’t just mean the OTP system, but any encryption where Alice and Bob (necessarily) share
some secret key. Since in our assumptions the adversary (Eve) can access the encrypted text, an intuitive definition of perfect security is this: A method is secure iff the “odds” of the adversary to figure out \(m\) are the same whether or not he has \(c\), i.e. seeing \(c\) does not increase these “odds”. Of course, the problem is in formalizing the above informal definition. To do this, assume a message \(M\) was chosen by Bob from some probability distribution over \(\mathcal{M}\) and Bob publicly announced this distribution. For simplicity (it will turn out to suffice and actually not matter for our definition), assume \(M\) is chosen by Bob uniformly at random. At this stage, from Eve’s point of view each possibility \(M = m\) is equally likely. Now assume Bob computes the ciphertext \(C = E_s(M)\) and gives the particular outcome (which happens to be some \(c \in \mathcal{C}\)) to Eve. We now estimate again the probability that \(M = m\) after Eve has learned that \(C = c\). If the system is ideally secure, this probability should not change (i.e. remain uniform over \(\mathcal{M}\)), irrespective of specific \(m\) and \(c\). Notice, the above probability is also taken over the random choice of the shared key \(s\). Formally:

**Definition 4** Let \(M \in \mathcal{M}\) be a random message and \(C \in \mathcal{C}\) be the ciphertext of \(M\), that is, \(C = E_s(M)\). For any \(m \in \mathcal{M}\) and \(c \in \mathcal{C}\), an encryption system is called **perfectly secure** if from the perspective of the attacker, \(Pr(M = m \mid C = c) = Pr(M = m)\). This means that Eve’s probability of guessing \(M\) remains unchanged after seeing any particular outcome \(C = c\).

We can now formally show that OTP is secure:

**Proof:** Take any \(m \in \mathcal{M}\) and \(c \in \mathcal{C}\).

\[
Pr(M = m \mid C = c) = \frac{Pr(M = m \land C = c)}{Pr(C = c)} = \frac{Pr(M = m) \cdot Pr(C = c \mid M = m)}{Pr(C = c)} = \frac{Pr(M = m) \cdot Pr(C = c \mid M = m)}{\sum_{\tilde{m} \in \mathcal{M}} (Pr(M = \tilde{m}) \cdot Pr(C = c \mid M = \tilde{m}))}
\]

We obtain step 1 from Bayes law. Step 2 comes from the definition of conditional probability. In step 3, we expand out \(Pr(C = c)\), which is the sum of all the conditional probabilities that \(Pr(C = c)\), weighted by the probability of the condition. Namely, we sum over all possible \(\tilde{m}\)’s.

We then note that \(Pr(C = c \mid M = \tilde{m}) = Pr(s = c \oplus \tilde{m})\), as \(\tilde{m} \oplus s = c\) iff message \(\tilde{m}\) and unique key \(s\) produce ciphertext \(c\). Since every \(s \in \{0,1\}^k\) is just as likely, the probability is uniform over the key space: \(Pr(s = c \oplus \tilde{m}) = \frac{1}{2^k}\). Hence,

\[
Pr(M = m \mid C = c) = \frac{Pr(M = m) \cdot \frac{1}{2^k}}{\sum_{\tilde{m} \in \mathcal{M}} (Pr(M = \tilde{m}) \cdot \frac{1}{2^k})} = \frac{\sum_{\tilde{m} \in \mathcal{M}} Pr(M = \tilde{m})}{Pr(M = m)} = 1
\]

Thus, \(Pr(M = m \mid C = c) = Pr(M = m)\), so OTP’s security has been proven.\(^2\)

\(^2\)Notice, we never actually used that the message \(M\) is chosen uniformly at random. The proof works for any distribution on \(M\), as long as the one-time pad \(s\) is random.