CS195-7: Introduction to Cryptography September 3, 2002

Lecture 1

Instructor:Anna Lysyanskaya

This course is about secure and authentic communication in an adversarial setting.

Let us consider two fundamental cryptographic problems: secure communication and data authenticity. In this lecture, we will focus on the first one; while the problem set will familiarize you with the second one.

Consider a toy scenario: Alice wants to send a message to Bob, but there is an eavesdropper Eve who can observe everything that Alice says. Instead of sending her message m in the clear, Alice will encrypt it and send the ciphertext c instead. We want to devise a system where Alice can send a ciphertext c that Bob will be able to correctly decrypt, and yet Eve will have no idea what was sent.

1 Defining Security

In defining security, we must first answer questions such as: Who are Alice, Bob, and Eve? What is a message? What is needed for the encryption and decryption algorithms? What does it mean to learn nothing?

Alice, Bob, and Eve are all algorithms. Our task is to specify the algorithms for Alice and for Bob, such that no matter what Eve does, our cryptosystem remains secure. (Moreover, we should assume that Eve does not know the algorithms that we specify for Alice and Bob. Such an assumption would weaken our definition, while not gaining us anything. Having Alice and Bob run proprietary obfuscated code makes it harder to deploy the cryptosystem at hand, while an attacker can actually get his hands on the algorithm by, for example, bribing the people who designed it.)

As for the message, as far as Eve is concerned, the message comes from some probability distribution. Let us write $m \leftarrow M$ to denote that message M came from probability distribution M. We can define a secure cryptosystem for a specific M, or for *all* distributions M on messages of a given length. (It is easy to see that we cannot hope that Eve learns nothing about the length of the message, or rather the number of bits needed to write down the message, since the length of the ciphertext c is an upper bound on the length of the message.)

It is clear that in order to communicate securely, Alice and/or Bob must know something that Eve does not know. Otherwise, Eve can decrypt the message just as well as Bob can — we have assumed that she knows his algorithm. Therefore, in addition to the encryption and decryption procedures, we must assume a setup procedure that generates the secrets for Alice and Bob. To be as general as possible, let us imagine that this setup procedure generates secrets for Alice and Bob (let us call them s_A and s_B), and also may give away some extra information to Eve (let us call it s_E), and also possibly generates some public parameters P.

The encryption algorithm will have to take as input: the message m that is being encrypted, perhaps also Alice's secret input s_A, the public parameters P, and maybe some random bits r as well. The encryption algorithm will produce the ciphertext c. The decryption will have to take as input the ciphertext c, Bob's secret s_B, and the public parameters P, and, in order for this system to have any meaning, must output the message m that Alice encrypted in the first place.

The hardest part to capture is, of course, what it means to learn nothing. The traditional information-theoretic approach (due to Shannon) is to say that the *a-priori* information about the message — namely that it came from distribution M — is all that Eve will know even after she sees ciphertext c, i.e., even conditioned on the fact that c was the ciphertext Alice sent to Bob.

To summarize, the definition is as follows:

Definition 1.1 (Unconditional Security). *Algorithms (G, E, D) constitute a secure cryptosystem for all messages spaces of length n if*

Setup algorithm *Algorithm G takes as input a sequence of random bits, and produces secret inputs s_A, s_B, s_E for Alice, Bob, and Eve respecitively. It also sets up some public parameters P.*

Encryption *Algorithm E takes as input the message $m \in M$, Alice's secret s_A, the public parameters P, and some randomness r, and outputs the ciphertext c.*

Decryption *The decryption algorithm D takes as input a ciphertext c, Bob's secret s_B, and public parameters P, and outputs a message m' such that for all $m \in M$, c, s_A, s_B, P, if the probability that c is a ciphertext for m, and Alice's and Bob's secrets are s_A and s_B, and the public parameters are P, is positive, then on input (c, s_B, P), Bob will always output the message m.*

Security *For all messages m, and for all possible ciphertexts c for the message m, $\Pr_M[m] = \Pr_{M,G,E}[m|c]$, where $\Pr_M$ denotes that the distribution that m is drawn from is M, while the notation $\Pr_{M,G,E}$ denotes that the joint distribution of (m, c) is determined by running algorithm G, then picking message m from the distribution M, and then running E on the appropriate inputs. (Later on, we will develop more convenient notation.)*

2 Achieving Our Definition

It turns out that it is possible to achieve this definition using a very simple encryption algorithm. However, this encryption algorithm requires a very expensive setup procedure: Alice and Bob must share a secret of the same length as the message that they are trying to communicate. Unfortunately, this is optimal, and the argument for showing it is also rather simple. Let us now elaborate.

2.1 The One-Time Pad Cryptosystem

Assume that the (binary) length of the message that will be sent is n. The generation procedure G simply picks a binary string s of length n uniformly at random, and gives it to Alice and to Bob as their secret input. The encryption and decryption are as follows:

$$E(m, s) = m \oplus s$$
$$D(c, s) = c \oplus s$$

Theorem 2.1. *The one-time pad is a secure cryptosystem for any message space M on messages of length n.*

Proof. It is obvious that the generation, encryption and decryption requirements of our definition are met. Let us now prove the security requirement.

First, recall that $\Pr[m|c] = \Pr[m \wedge c] / \Pr[c]$. Now,

$$\begin{aligned} \Pr[m \wedge c] &= \Pr[m] \Pr[c|m] \\ &= \Pr[m] \Pr[s = c \oplus m | m] \\ &= \Pr[m] 2^{-n} \end{aligned}$$

because s is chosen independently of m. On the other hand,

$$\begin{aligned} \Pr[c] &= \sum_{m' \in \{0,1\}^n} \Pr[m'] \Pr[c|m'] \\ &= \sum_{m' \in \{0,1\}^n} \Pr[m'] 2^{-n} \\ &= 2^{-n} \end{aligned}$$

Therefore,

$$\begin{aligned} \Pr[m|c] &= \frac{\Pr[m \wedge c]}{\Pr[c]} \\ &= \frac{2^{-n} \Pr[m]}{2^{-n}} \\ &= \Pr[m] \end{aligned}$$

which completes the proof. □

This cryptosystem is called a “Vernam cipher,” after Gilbert S. Vernam who proposed it in 1917 to protect telegraph communications. (Note that 1917 was before probability theory existed, so I think it must have been Claude Shannon who first proved security of the Vernam cipher using probability theory.)

So we are doing well: we have a great cryptosystem. But suppose we wanted to use it twice. Will it still be secure? Actually, no. This is because, while Eve will not be able to retrieve full information about the two messages sent, she will be able to compute their exclusive-or! Here is why: let $c_1 = m_1 \oplus s$ be the ciphertext for m_1, while $c_2 = m_2 \oplus s$ is the ciphertext for m_2. Observe that $c_1 \oplus c_2 = (m_1 \oplus s) \oplus (m_2 \oplus s) = (m_1 \oplus m_2) \oplus (s \oplus s) = m_1 \oplus m_2$. So we are not secure at all! What happened? The thing is that if we are going to encrypt two messages, it is almost like encrypting one message that is twice as long. So we are encryping a $2n$-bit message, while our proof of security was only for n-bit messages.

It turns out that this limitation is fundamental.

2.2 Shannon's Impossibility Result

If Alice and Bob only share a secret of length n, then there is no method which can enable them to securely communicate a message of length $n+1$.

Theorem 2.2. *Let (G, E, D) be given. Let M be a message space, and S be the space of keys that G produces as shared keys for Alice and Bob. $|M| > |S|$, then there exists $m \in M$ and a ciphertext c such that* $\Pr[m] \neq \Pr[m|c]$.

Proof. This proof can simply be done by counting. Fix a possible ciphertext c. Try all possible keys s to decrypt it. Since there are only $|S| < |M|$ possibilities, some message $m \in M$ cannot possibly be the correct decryption of ciphertext c. So while we have $\Pr[m] > 0$, it is the case that $\Pr[m|c] = 0$. $\square$

Actually, Shannon showed something stronger than what I showed here. He showed that even if you allow $\Pr[m] \approx \Pr[m|c]$ instead of "=", the limitation is still there.

3 Relaxing the Definition

Despite the limitation discovered by Shannon, we still would very much like to communcate securely! But the only way to do that is to settle for a weaker notion of security. One way of relaxing the definition is as follows: instead of requiring that no matter how sneaky Eve is, she does not find out anything about the message, we may only consider Eve whose computational or other resources are bounded. So we may settle for a situation when in fact $\Pr[m] \neq \Pr[m|c]$, and yet no algorithm for Eve that executes sufficiently fast will be able to notive the difference.

How to appropriately relax this definition will be the subject of the next lecture.