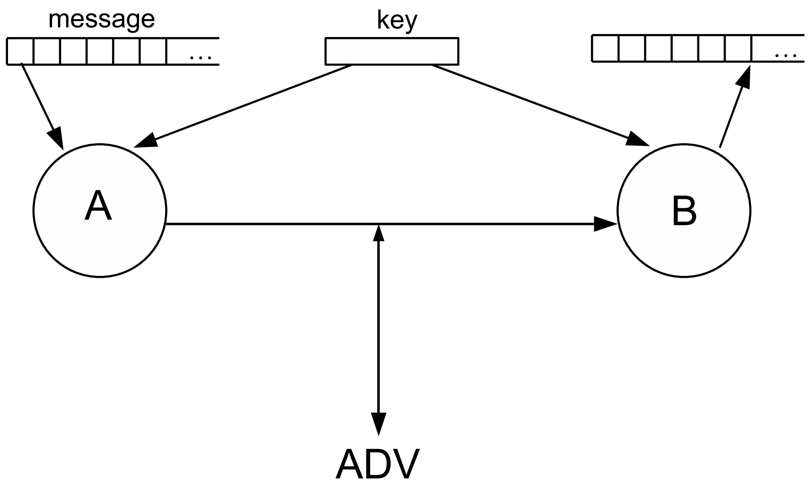
Computer Science 2426F — Fall, 2020
St. George Campus — University of Toronto

Notes #0

Introduction

We begin by describing – very informally – what the typical "man in the street" thinks of as the quintessential application of cryptography: *secure sessions (using a shared secret key).* Two people A and B have gotten together and chosen a random n bit key K. They then separate, and can only communicate over a very insecure internet. We have the following picture:



A now, from time to time, wishes to send stuff – we call it "plain text" – to B. We refer to the entirety of what A will ever want to send to B as the "message", although A will only be sending the message a "piece" at a time. For now, we can think of each piece as being a single bit. Unfortunately, there is an adversary ADV who has complete control of the internet. ADV not only listens to everything that A says, but also completely controls what is sent to B. To defend against ADV, A will be in some sense "encrypting" each piece using the shared key K.

We will always assume that the adversary knows the algorithms that the good guys (A and B in this case) are using; the only thing the adversary doesn't know are the randomly chosen keys. In this case we hope that ADV will not be able to learn anything "significant" about the message – we will call this *privacy* – and that ADV will not have a significant chance of making B output something wrong – we will call this *integrity.* We will define this all very carefully later in the course. For now, to be a bit less vague, the first condition roughly means that even if ADV is able to choose part of the message himself, he should be no good at figuring out any other part of it; the second condition roughly means that even if ADV is able to choose the whole message himself, he shouldn't be able to cause B to output an incorrect piece. Of course, ADV can choose to stop sending stuff to B causing B to output nothing, and A can send garbage to B causing B to "$fail$".

Note that this session may go on for many years, and that our security conditions are with respect to the entire message, not the individual pieces. It is not sufficient that each piece be somehow sent securely. We will see later that it easy to come up with a system where each piece is sent securely but the message (that is, all the pieces) is completely insecure.

We will see later that if ADV is allowed to use unlimited computing time, then there is no way to do secure sessions if the message is longer than the key. Therefore, we will insist that ADV be polynomial-time. There is still a big problem, however. We are unable to prove P≠NP, and we will see later that if P=NP, then even with this time constraint on the adversary there is no way to do secure sessions. We will therefore have to base security on certain complexity theory assumptions, the goal being to use as plausible and as few assumptions as possible.

We will later prove (at least part of) the following very fundamental theorem of cryptography.

Theorem: The following are all equivalent.

- It is possible to do secure sessions.
- There exist pseudo-random generators
- There exist "one-way functions". (Informally, a one-way function is a function that is easy to compute but hard, on the average, to invert.)
- There exist secure digital signature schemes.

There are a number of cryptographic primitives that require stronger assumptions than the above. A very important one is "public-key cryptography". One very important application of public-key cryptography is the following: Say that we have a "public-key infrastructure", and A wishes to have a secure conversation with B but they do not have a shared private key. Then A and B can use public-key encryption and engage in a protocol that will allow them to agree on a private key for that session. This notion of "session-key exchange" is very complicated, and we discuss it carefully much later.

The reader may note that the notion of a "session" often is used to mean A and B talking to each other, whereas we have used it in the more narrow sense of A talking to B. Let's say we have a method to do this "uni-directional" session securely. How can we have A and B talk securely to each other? Simple. A talks securely to B using our assumed method, and B talks to A using the same method. *It is unbelievably important to note that the "method" involves the choice of a random key. Since we are using this method twice, we must choose an independently random key each time.* More generally it is important to realize that whenever a key is generated in cryptography, it is intended to be used in one and only one way (although the way may be quite complex and last for years). *Never* reuse an old key for a new purpose. This can be deadly, and we will shortly see an example.

A Simple Setting and One-Time Pads

One of the central topics in this course is "pseudo-random generators". In order to motivate this topic, we now consider a simpler version of secure sessions. For one thing, we only permit ADV to listen in (or eavesdrop), not to change anything on the line. (Keep this in mind when we define "perfect security" below; our notion of security here relates only to what we will later call "privacy", not to integrity.) Also, we place no restriction on the computing time of ADV.

We say that the key K consists of n bits $K = K_1K_2 \dots K_n$; we say that the message M consists of m bits $M = M_1M_2 \dots M_m$. We have an encryption function $Enc : \{0,1\}^m \times \{0,1\}^n \to \{0,1\}^*$, where we view $Enc(M,K)$ as the encryption of M under key K that A sends over the channel to B. We also have a decryption function $Dec : \{0,1\}^* \times \{0,1\}^n \to \{0,1\}^m$, and we insist on the *correctness* condition: $Dec(Enc(M,K),K) = M$, for all M,K.

If $m \leq n$, that is $|M| \leq |K|$, then it is possible to do this perfectly securely: we do not need to make any assumptions about the run time of the adversary, or any assumptions from computational complexity.[1] The most famous way is called the "one-time pad", presumably named after the pad (containing the key) that a spy would carry in his pocket. We define $Enc(M, K) = E = E_1 E_2 \dots E_m$ where $E_i = M_i \oplus K_i$ for each $1 \leq i \leq m$; $\oplus$ means exclusive-or, that is, the sum mod 2. B decrypts by computing $M_i = E_i \oplus K_i$; that is, $Dec(E, K) = E_1 \oplus K_1, E_2 \oplus K_2, \dots, E_m \oplus K_m$. We can prove that this is perfectly secure, but first we need a definition of perfect security. The following theorem gives three definitions of perfect security and states that they are all equivalent; this is the notion of security that is usually used in this setting. The theorem after that states that one-time pad is perfectly secure. The last theorem states that if $|M| > |K|$ (that is, $m > n$) then *no* pair Enc, Dec is perfectly secure. We leave the proofs of these theorems as an exercise.

Theorem: Let Enc, Dec be correct. Then the following definitions of perfect security for Enc are equivalent.

1. For $M \in \{0,1\}^m$, define the distribution D_M on strings as follows: to choose a random member of D_M, choose a random $K \in \{0,1\}^n$ and output $Enc(M, K)$. Then Enc, Dec is *perfectly secure* if D_M is exactly the same for every M. That is, for every $\alpha \in \{0,1\}^*$, the probability of α according to D_M is independent of M.

2. For every two messages, no function can tell which one has been encrypted. That is, Enc, Dec is *perfectly secure* if for every $M_0, M_1 \in \{0,1\}^m$ and for every $f : \{0,1\}^* \rightarrow \{0,1\}$, the following holds: consider the experiment where b is randomly chosen from $\{0,1\}$ and K is randomly chosen from $\{0,1\}^n$; then the probability that $f(Enc(M_b, K)) = b$ is exactly equal to $1/2$.

3. Enc, Dec is *perfectly secure* if for every $f : \{0,1\}^* \rightarrow \{0,1\}^m$, the following holds: consider the experiment where M is randomly chosen from $\{0,1\}^m$ and K is randomly chosen from $\{0,1\}^n$; then the probability that $f(Enc(M, K)) = M$ is equal to $1/2^m$. That is, an Adversary seeing the encryption of a random M cannot guess the value of M better than he could by just outputting an arbitrary answer.

Theorem: The one-time pad Enc, Dec is perfectly secure as defined in the previous theorem.

Theorem: If $m > n$, then *no* correct Enc, Dec is perfectly secure.

In particular, the last theorem tells us that a "two-time pad" is not secure. In a two-time pad, $m = 2n$; both the first half of the message, and the second half of the message, are sent using the key as a one-time pad. Intuitively this is insecure since anyone who knows (or can guess) part of the first half of the message can learn part of the second half of the message. This is insecure even against a computationally limited adversary.

In practice, we will want to have secure sessions with short keys (say a few hundred bits) and enormously long messages that come in a piece at a time. A crucial tool for doing this will be pseudo-random generators, our first main topic. To motivate this, let us stay for now with the setting of an adversary that only eavesdrops. We will now assume that our adversary is restricted to run in time polynomial in n. A natural idea is to take the key K, and use a pseudo-random number generator G to extend it to a much longer key K', and then use K' as a one-time pad. We will later come back to the question of what "secure sessions" should mean.

[1]If X is a string, then $|X|$ represents the *length* of X.

What “pseudo-random” property should G have? The idea is that a computationally limited adversary should not be able to significantly distinguish between the situation where he sees a pseudo-randomly generated string, and the situation where he sees a truly randomly generated string.