CSCI3381-Cryptography

Lecture 1

January 17, 2017

1 What Does Cryptography Do?

When you visit a secure website—for example, when you go to your bank's website to pay bills—you and the other party (in this example, the bank) have to be sure about a number of things.

- Is this really the bank's website? (Can someone operate a website that you mistake for the bank's?)
- Are you really the account's owner? (Can someone impersonate *you*, gain access to your account information, and make payments from your account?)
- Is the information you and the bank exchange *private*? (Can someone eavesdrop and obtain the information?)
- Is the information you and the bank receive from one another identical to the information that was sent? (Can someone alter the information without the alteration being detected?)

An extraordinary sequence of computations takes place to provide each of these services. This course is about what is behind those computations. Cryptography studies methods for obtaining secure communication in the presence of malicious adversaries.

2 Symmetric Encryption

We start the course with methods for achieving the privacy goal listed above. This is probably what most people think of when they think of cryptography.

2.1 Terms to know

- Plaintext, Ciphertext
- Encryption Algorithm, Decryption Algorithm, Key
- Kerckhoffs' Principle
- Ciphertext-only attack; Known-plaintext attack, Chosen-plaintext attack, Chosen-ciphertext attack
- Brute-force (= exhaustive search) attack

2.2 Alice, Bob, and Eve

- Alice wants to send a message (the *plaintext* p) to Bob. Think of p as a string of text characters, or as a string of bits.
- An eavesdropper Eve is able to monitor the communication channel and read p.
- To avoid this, Alice *encrypts* p: Alice and Bob share some secret information, a string k called the *key*. The encryption algorithm E has two inputs, the plaintext p and the key k, and produces as output the *ciphertext*

  $$c = E(k, p).$$

  (Figure 1.)
- Although Eve can read c, it is pure gibberish and she is unable to learn anything about the plaintext message from it.
- When Bob receives c, he applies the *decryption algorithm* D to recover the plaintext
  $$p = D(k, c).$$

In real life, Alice might be a military commander and Bob an officer in the field. Or, in more modern and familiar terms, Alice is the customer of an online store Bob.com. Or, Alice is your computer and Bob the WiFi router in the next room. Or, Alice and Bob are the same person: Alice encrypts new data on her laptop and decrypts it to read it later; Eve is someone who steals the laptop, or hacks into it remotely.

There's an interesting question about exactly how Alice and Bob agree on the key k. This is not a problem in the scenarios concerning the WiFi router or the encryption of Alice's local data, but it is a BIG problem in the online shopping scenario. We will deal with this question in the second half of the course.

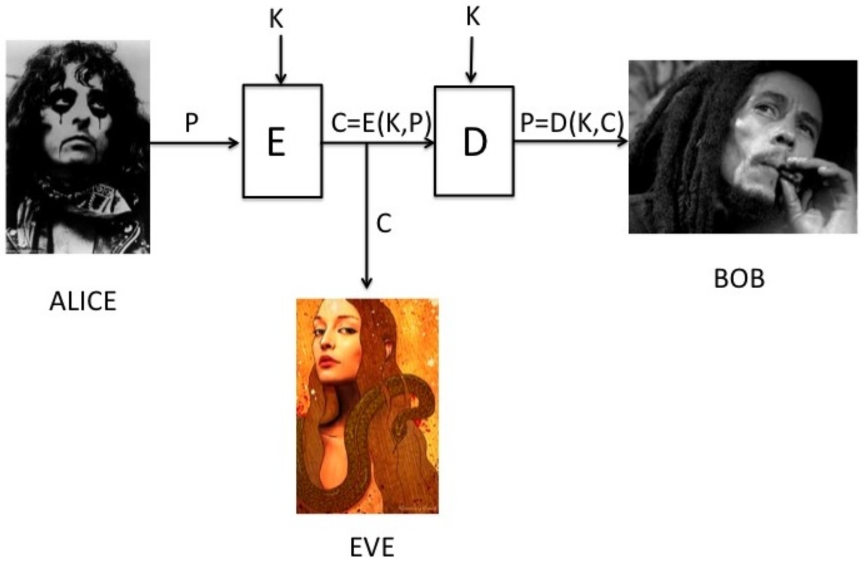

Figure 1: The basic model for symmetric encryption. Eve can read the ciphertext c and knows the encryption and decryption algorithms D and E, but without access to the key k she cannot get any information about the plaintext p.

2.3 Mathematical Formalism

We denote by $\mathcal{K}$ the *key space*–the set of possible keys. In most of our applications, $\mathcal{K}$ will be a set of bit strings of a fixed length, although our first examples will involve strings of letters of the alphabet. We denote by $\mathcal{M}$ the *message space*–the set of possible messages, including both plaintext and ciphertext messages. Thus E and D are both functions

$$E : \mathcal{K} \times \mathcal{M} \to \mathcal{M}, D : \mathcal{K} \times \mathcal{M} \to \mathcal{M}.$$

Note that while $\mathcal{K}$ is usually taken to be a finite set, $\mathcal{M}$ will at times be viewed as infinite (*e.g.,* the set of all finite bit strings), at other times as consisting of just the finite set of messages of a fixed length. We denote by $|\mathcal{K}|$ the *cardinality* (number of elements) of the set $\mathcal{K}$. We use the same absolute value notation for strings as well as sets, but now to denote the length of the string. For example, if $\mathcal{K}$is the set of bit strings of length 5, then $|\mathcal{K}| = 32$, while $k = 01101 \in \mathcal{K}$, and $|k| = 5$.

2.4 Requirements

Most of these requirements were formulated in the 1880's by Auguste Kerckhoffs, although they are here presented in modern language. The last (and surprising) one is what is usually called Kerckhoffs' Principle.

- Decryption reverses encryption: that is, for all $k \in \mathcal{K}$, $m \in \mathcal{M}$

  $$D(k, E(k, M)) = M.$$

  Put another way, for each $K \in \mathcal{K}$, the maps

  $$D_k, E_k : \mathcal{M} \to \mathcal{M},$$

  defined by

  $$D_k(m) = D(k, m), E_k(m) = E(k, m),$$

  are mutually inverse bijections.

- Encryption and decryption are easy to compute (if you know the key). What this means in practice is that the time required to compute $D(k, m)$ and $E(k, m)$ should be linear in $|m|$ (proportional to the length of the message).

- Decryption should be *for all practical purposes impossible* to carry out without the key. That is, there should be no practical algorithm for recovering m just from knowledge of the ciphertext $E(k, m)$, We will see later exactly what is meant by 'impossible for all practical purposes'. (In the lingo of computational complexity, we usually say 'infeasible' in preference to 'impossible'.)

- The system should remain secure *even if Eve knows the algorithms for computing D and E.* That is, the *only* part of the system required to be secret is the shared key k.

The last point needs some explanation. The encryption and decryption algorithms we use on a daily basis are embedded in our Web browsers, wireless cards and routers, and the like, and are therefore already available for Eve's inspection. But Kerckhoffs was talking about *military* cryptography, not about WiFi and the Internet; still, the danger of relying on secrecy of a cryptographic system was still apparent to him. If a lot of people need to know how the system works, then there is a good chance that this information will be leaked. If a system compromised in this way is required to be secret, then it would have to be replaced by a completely new system, and these are extremely difficult to design. On the other hand, if a key is compromised, one only has to choose a new key. The danger of ignoring Kerckhoffs' principle has been demonstrated many times: poorly designed cryptographic systems whose inventors hoped to protect through secrecy generally wind up being broken.

2.5 Brute-force attack and the size of the key

Since we suppose that Eve knows the encryption and decryption algorithms, she is able to launch the following attack if she intercepts a ciphertext C: For each key $k \in \mathcal{K}$, compute $P = D(k, c)$. Presumably only one of these candidate plaintexts p will make any sense, and Eve will accept this as the result.

This is a *brute-force*, or *exhaustive search* attack. In order to preserve our requirement that the recovery of the plaintext be 'for all practical purposes' impossible, the size of the key space $\mathcal{K}$ must be so large that only a tiny fraction of it can be searched in practice.

How large is this? We will look at the question more closely later on, but we can establish some back-of-the-envelope benchmarks. One billion (10^9) decryptions per second might be feasible on special-purpose fast parallel hardware. Let's be generous and allow Eve the power to perform one *trillion* (10^{12}) decryptions per second. After more than ten years of work on the problem, the plaintext will likely lose any importance to the attacker. Again, let's add a fudge factor and allow Eve to decrypt for one hundred years. The number of seconds in one hundred years is slightly more than three billion (3×10^9). Multiplied by one trillion, this gives Eve the possibility of performing 3×10^{21} decryptions. This is around 2^{72}. If the key is represented by a string of bits, we would require 72 bits for the key. You will often see 80 bits used as a kind of benchmark for security against exhaustive search attacks. (This is a moving target, and technological advances can change the story. We will see a more sophisticated general approach to this problem later.)

3 Information available to the attacker

There are several different kinds of attacks that Eve can launch, depending on the information available to her. Systems that are secure against the weakest kind of attack (ciphertext-only) may be vulnerable to the stronger attacks.

- *Ciphertext-only attack.* Eve only has the intercepted ciphertexts at her disposal, and uses these to infer the plaintexts.

- *Known-plaintext attack.* Eve knows several pairs $(p.c)$ where $C = E(p, k)$. For instance, she might have guessed correctly that the first few intercepted ciphertexts are encryptions of a boilerplate header used on all messages. She uses these pairs to help deduce the plaintexts associated with other ciphertexts.

- *Chosen-plaintext attack.* Eve is able to obtain the encryptions of plaintexts that she chooses herself.
- *Chosen-ciphertext attack.* Eve is able to obtain decryptions of ciphertexts she chooses herself.

Chosen-plaintext and chosen-ciphertext attacks may seem far-fetched, but there are real-life instances. (A chosen-plaintext attack by US on partially-broken Japanese cipher in World War II revealed the location of impending attack on Midway Island. Some systems that respond with error messages for improperly-formed ciphertexts have been shown to be vulnerable to chosen-ciphertext attacks that can actually be carried out in practice.)