CSCI3381-Cryptography

Lecture 1

August 26, 2014

1 What Does Cryptography Do?

Every time you visit a secure website—for example, when you go to your bank's website to pay bills—you and 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?)

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.

- Cryptography is not the be-all and end-all of Computer Security: "Using encryption on the Internet is the equivalent of arranging an armored car to deliver credit card information from someone living in a cardboard box to someone living on a park bench." (Gene Spafford).
- Scientific study of cryptography goes back hundreds of years—an instance of Computer Science before computers.
- Modern cryptography is a beautiful bridge between theory and practice; it uses ideas from probability theory, number theory and computational complexity to create practical tools that people use every day.

Cryptographic ideas can be applied to other problems:

- Can elections be conducted fairly so that everyone can verify the result, but all voters' ballots are secret?
- Can a system of 'digital cash' that is both trustworthy and untraceable be devised?
- Can you prove that you know some piece of information without giving away any of the information?
- Can several people play poker by e-mail, with the assurance that the game is being conducted fairly, and without a trusted neutral party to monitor the game? Can you toss a coin over the telephone?

2 Symmetric Encryption

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 attack

2.2 Alice, Bob, and Eve

- Alice wants to send to Bob the *plaintext* message P. 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
$$P = D(K, C).$$

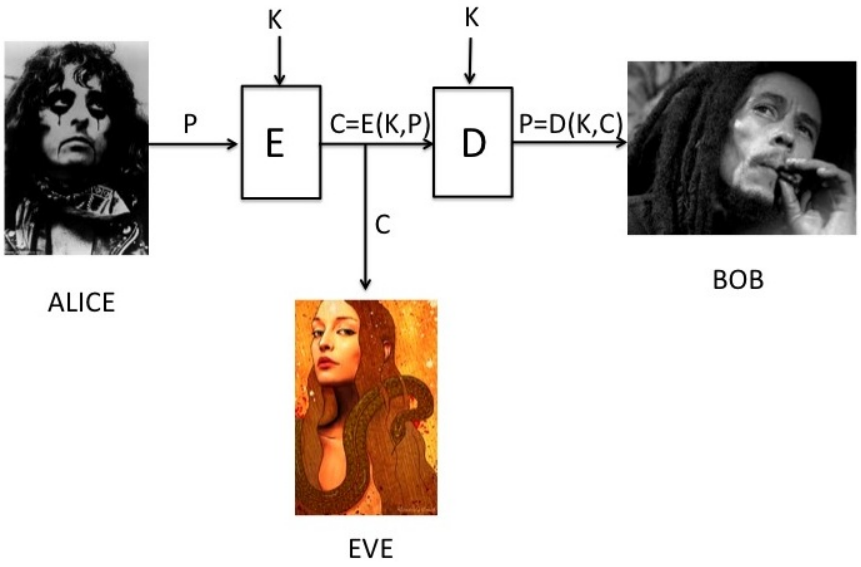

Figure 1: The basic model for symmetric encryption. Eve can read the ciphertext C and knows the encryption and decryption algorithms D and E, but without access to the key K she cannot get any information about the plaintext P.

2.3 Mathematical notation

You should be (or prepare to become) comfortable with standard mathematical notation for these things. Underlying our scenario are:

- A set $\mathcal{M}$ of possible messages, and a set $\mathcal{M}'$ of possible encryptions of messages. In practice, as we proceed in the course, we will have $\mathcal{M} = \mathcal{M}'$ be the sets of all finite strings of bits (0's and 1's), although at the outset we will treat $\mathcal{M}$ as the set of all strings of lower case letters that are meaningful English.

- A set $\mathcal{K}$ of *keys*.
- A function $E : \mathcal{K} \times \mathcal{M} \to \mathcal{M}'$ and a function $D : \mathcal{K} \times \mathcal{M}' \to \mathcal{M}$ that satisfy the following property: for all $P \in \mathcal{M}$ and $K \in \mathcal{K}$,

$$D(K, E(K, P)) = P.$$

Another way to say this is that for each fixed $K \in \mathcal{K}$, the function $E_K : \mathcal{M} \to \mathcal{M}'$ defined by

$$E_K(P) = E(K, P),$$

is one-to-one and onto, and that the function $D_K : \mathcal{M}' \to \mathcal{M}$, defined analogously, is the inverse of E_K.

2.4 Assumptions about the model

- The encryption and decryption algorithms are fast (linear in the length $|P|$ of plaintext).
- The encryption algorithm might be *probabilistic*—that is, it might use some randomly generated information as inputs. This will have the effect of the same plaintext-key combination leading to different ciphertexts on different occasions. On the other hand, the decryption algorithm is *deterministic*—the same ciphertext-key pair will always decrypt to the same plaintext.
- **Eve knows what the encryption and decryption algorithms are!** The only secret is the key K. This is called *Kerckhoffs' Principle,* and dates back to the 19th century. It seems counter-intuitive, but there are good reasons for it.
  - We don't have a large assortment of different encryption methods at our disposal, so a lot of people will have to know how the encryption and decryption algorithms work, making it vulnerable to leaks. When all the security of the system lies in the secrecy of the key, as Kerckhoffs' Principle requires, we can change the key when its security is compromised; it is not so easy to change cryptosystems.
  - If lots of people know how the system works, it will receive more scrutiny, and weaknesses will be discovered more rapidly.
  - When cryptography is used in the Internet, Kerckhoffs' Principle is inescapable: Web browsers implement a standard cryptographic system described in openly available documents, used by all designers of secure web services.

Kerckhoffs' Principle, and the reasoning behind it, has several important consequences:

- Proprietary encryption algorithms should not be trusted. If the designers of software try to achieve 'security through obscurity' this way, there is *less* confidence in the security of the system: Few people know how it works or why it is supposed to work, and there is a risk that someone who finds out how it works will be able to break it and read intercepted communication.

- A cryptographic system that you dream up in a few days is almost certainly insecure. Of course some people *do* design encryption and decryption algorithms, and we will study how some of these work. But the practical focus in this course is about the correct deployment of existing algorithms rather than the design of new ones.

- *Key length.* Since Eve knows D she can *in principle* launch an attack by *brute force*: Given the intercepted ciphertext C, for each possible key K', compute $D(K', C)$ until you see something that makes sense. The system must therefore be designed so that this algorithm does not work *in practice*, and that means that $\mathcal{K}$ must be too large to search in this way. With a 40-bit key there are $2^{40} \approx 10^{12}$ (one trillion) different keys, testing them all is feasible, especially if one is able to distribute the search over many processors. Keys more than 128 bits long are likely to be secure against this kind of brute-force attack.

- *Key re-use.* A single key should not be used indefinitely. In secure Web communications, the encryption key is used only once in each session. This raises the very important question of how Alice and Bob are supposed to agree on a key in the first place!

2.5 Different modes of attack

There are several different kinds of attacks that Eve can launch, depending on the kind of resources 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.)