CS6222 Cryptography

Topic: Introduction
Lecturer: Wei-Kai Lin (TA: Arup Sarker)

Date: Aug 27, 2024
Scriber: Wei-Kai Lin

1 Match-Making and Overview

We begin with a toy example that can be useful in real life.

1.1 A toy example: match-making

Consider Alice and Bob (that is, two parties) in a blind-date scenario. They have met for the first or second time, and now they want to decide if they are mutually matched. Alice has two options: Yes if she likes Bob, or No if she does not. Bob also has two options symmetrically. Bob and Alice are matched if and only if both are saying Yes. It is essentially computing the logical AND on input (x, y), where x is Alice's input and y is Bob's, while they are matched if and only if the output $\mathsf{AND}(x, y) = 1$. It is easy to compute: Alice and Bob send (and thus open) to each other x and y.

However, if Alice said Yes but Bob said No, Alice could be embarrassed because she is rejected, or Bob could be embarrassing because he rejected Alice. Similarly for the symmetric case. We don't want anyone be embarrassed in any case. Especially, if $\mathsf{AND}(x, y) = 0$, even if Bob said No (with $y = 0$), we want that *Alice is able to pretend* she said No (no matter what her x is). Is that possible to achieve?

Suppose that both Alice and Bob trust another person, Charlie (who is a trusted third party). Then, Charlie solves the problem: just let Charlie know (x, y) and compute and send $z := \mathsf{AND}(x, y)$ to Alice and Bob. Alice and Bob learn only the output z, but nothing more. Notice that by "nothing more," Alice can always learn anything that can be inferred from her input x and z. For example, if $x = 1$, then it must be $y = z$ and thus Alice learns y (by AND). The point is, when $x = 0$, Alice can not know y even given $z = 0$.

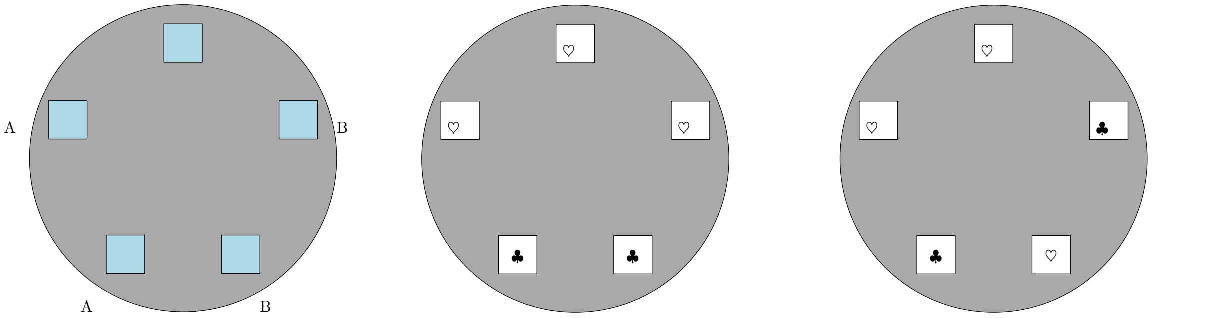
The question is, can Alice and Bob compute AND without a trusted third party?

Card-based AND protocol. Assume that Alice and Bob have cards: three identical hearts ($\heartsuit$) and two identical clubs ($\clubsuit$). Also assume they have a round dish. They perform the following protocol. We denote Alice and Bob as A and B correspondingly for short.

1. A and B each takes 1 heart $\heartsuit$ and 1 club $\clubsuit$ cards.
2. A and B jointly put the remaining heart $\heartsuit$ face down on the top of the dish.
3. For both A and B, encode Yes or No using the two cards as follows:

$$\text{Yes: } \begin{bmatrix} \heartsuit \\ \clubsuit \end{bmatrix} \qquad \text{No: } \begin{bmatrix} \clubsuit \\ \heartsuit \end{bmatrix}$$

4. A puts her two cards on the left of the dish. B puts his two cards on the right of the dish. The five cards are perfectly and symmetrically put on the dish (Figure 1a).
5. A secretly and randomly rotate the cards with dish.
6. B secretly and randomly rotate the cards with dish.
7. Open all cards, A and B are matched if and only if there are 3 hearts in a row (Figure 1b). Otherwise, they are not matched (Figure 1c).



(a) Covered cards (b) Opened cards, matched (c) Opened cards, unmatched

Figure 1: Five cards on the dish. Figure 1a shows the cards shall be covered circularly and symmetrically, where A and B denotes the cards of Alice and Bob correspondingly. Figure 1b shows an opened state that 3 hearts in a row. Figure 1c shows an opened state that 3 hearts are separated by a club.

Analysis. We need to show both correctness and privacy. The correctness is easy: we just check all 4 possible rows of the truth table of AND, and it follows that $z = 1$ if and only if all 3 hearts in a row.

The privacy is also argued by enumerating all 4 rows of truth table. Observe that if $z = 0$, then any input yields the identical circular ordering. That is, after the random rotation, any of the 3 cases (No-No, Yes-No, No-Yes) yields an identical opening. Hence, Alice can not know Bob's input (in addition to z), and similarly for Bob.

Discuss. This match-making problem considers only 2 parties (Alice and Bob), each with 1-bit input, and compute only AND. We can generalize this problem into multiple parties, each with a long input, and all parties jointly computes a complicated function f on all input. This is called secure Multi-Party Computation (MPC), a broad area in cryptography.

Some extension or variants of match-making:

- Use 2 hearts and 3 clubs to compute AND.
- Compute logical OR.
- Compute logical XOR.

Although we have simple card-based protocols for AND, OR, and XOR, the protocols typically do not compose efficiently (unlike composing boolean gates).

1.2 Course Outline

This course consists of two parts.

Part 1 covers foundational primitives, including one-way functions (OWF), pseudorandom generators (PRG), pseudorandom functions (PRF), symmetric key encryption, authentication (message authentication codes (MAC) and signatures). We will discuss:

- Are they different or related? At first glance, they serve different purposes. We want to study them systematically.
- Construction of the primitives (using basic number theory and assumptions)
- Why do we believe cryptography exists?

Part 2 covers some modern cryptography (that is, cool stuff): zero-knowledge proofs (ZKP), secure two-party computation (2PC), secure multiparty computation (MPC), fully homomorphic encryption (FHE), my research (oblivious RAM (ORAM), doubly efficient private information retrieval (DEPIR), RAM-FHE). Many of them rely on stronger or specific mathematical hardness assumptions that we will use directly.

Related topics but we will NOT cover:

- System security, cybersecurity, implementation (such as CS 3710: Introduction to Cybersecurity[1], CS 4630: Defense Against the Dark Arts[2]), blockchain or Cryptocurrency[3]
- Really math) Number theory, algebra
- Quantum computing
- Secure or private AI/ML

2 Classical Cryptography and Encryption

In ancient Greek, the word "cryptography" came from "hidden writing," which is called *encryption* today. Historically, human considered the scenario of encryption in communication. Consider the scenario that Alice wants to send a message m to Bob through a medium of communication. However, the medium may be intercepted by an eavesdropper, Eve. An example is to send a message through radio, which is electromagnetic wave and received by Eve. Hence, Alice and Bob want to *encrypt* the message.

To do so, they share two algorithms $\mathsf{Enc}, \mathsf{Dec}$ secretly and prior to their communication. Later, when they need to communicate, they perform the following steps:

1. Alice computes the ciphertext $c \leftarrow \mathsf{Enc}(m)$, where m is the plaintext message.
2. Bob recovers the plaintext by performing $m := \mathsf{Dec}(c)$. (We intuitively suppose the decryption Dec always recovers m.)

Eve eavesdrops and gets c, but Eve does not have Dec and thus may not obtain m. Notice that it is essential to denote which information is public (known to all) and which is private (known to only Alice and Bob but not Eve). Here, Enc and Dec are private. What if Eve knows Enc or Dec? This is addressed by Shannon as follows.

Kerckhoffs's principle.

> The enemy knows the system [Sha49].

This is a lesson learned from history: the algorithms are eventually leaked to Eve. Hence, we shall be conservative and assume all the algorithms are given to adversaries. Notice that Alice and Bob must share some secret (otherwise Eve can always do whatever Bob does). As a direct consequence (of giving public Enc and Dec), we move the secret shared by Alice and Bob into a secret key k.

To ensure Eve can not recover m, key k must be randomized. To be general, we denote by Gen as the (randomized) key generation algorithm. An encryption scheme is a tuple of algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ with some properties to be discussed next.

Question: Clearly, $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ can not be all deterministic algorithms. Can we have randomized Enc? Or randomized Dec?

[1] https://aaronbloomfield.github.io/ics/uva/index.html

[2] https://cs4630-uva.github.io/

[3] https://aaronbloomfield.github.io/ccc/

Limits of classical encryptions. There is a long list of encryption schemes in history. Caesar, Vigenère, Enigma, ... In Virginia, Jefferson also built his own encryption.[4] They are built by smart people who came up with the cool algorithm. Also, they are argued secure by a test of time: letting many people to try and thus showing the encryption stands against all attempts.

Eventually, they are mostly broken. In particular, modern computers can solve them in a few seconds (if not faster).[5] More recently, DES is introduced in 1970s and broken 1990s. AES is introduced in 1990s, and it is still the workhorse today, say encrypting our bank accounts. However, given the history, it could be worrying how long will AES stand.

The security of a specific encryption is a big question. A broader question could be how to construct a secure encryption. Still, there is a more fundamental question: why do we believe the existence of encryption (and cryptography)?

3 Modern Cryptography: Provable Security

Looking back at the history, the previous approaches to cryptography follows a template: we relied on some smart people to develop an encryption, and then we relied on more smart people to attempt 1000 different unsuccessful attacks so we believed the security. No matter it is broken or not, the defense and attack are depending on each case, but we want a *systematic* approach. Some people say that the difference between science and art is the systematic approach.

Modern cryptography relies on the following paradigm:

- Define the security we want mathematically.
- State the precise mathematical assumptions we need (e.g. "factoring is hard," where hard is formally defined).
- Construct a scheme, and then prove the scheme satisfy the security definition (based on the assumption). That is often to argue: if the constructed scheme does not satisfy the security definition, then we can refute the assumption.

In addition to systematic, many cryptographers view this as a "win-win" situation, and the proof essentially shows there are two worlds: we are either in the world that the scheme is secure (e.g., existing a secure public-key encryption) or in the world that we can solve the mathematical hard problem (e.g., factoring is fast). Equivalently, the hard problem is harnessed to build useful cryptography.

3.1 Definition of Secure Encryption

We begin with defining a secure encryption. A good definition shall capture the read-world security yet be achievable. Consider the secret Alice-Bob communication (Section 2). We draft a couple of potential informal definitions.

Definition, attempt 1.

> In a secure encryption, the adversarial eavesdropper Eve cannot learn (all or part of) the key from the ciphertext.

Consider a (stupid) encryption algorithm that never uses the key and always outputs directly the message. Because the key is never used, Eve can not learn anything about the key. However, our message is totally unprotected. Hence, this definition is not good: it allows undesirable (intuitively insecure) schemes.

[4] Jefferson disk, https://en.wikipedia.org/wiki/Jefferson_disk.

[5] Some still resists researchers and modern computers. See Breaking German Army Ciphers: https://cryptocellar.org/bgac/.

Definition, attempt 2.

> In a secure encryption, the adversarial eavesdropper cannot learn the plaintext from the ciphertext.

This is better, but leaking some function (or part) of the plaintext can be fatal. For example, in a (top secret) message describing the course performance of a student, leaking the grade is pretty much leaking the whole message.

Definition, attempt 3.

> In a secure encryption, the adversarial eavesdropper cannot learn *any function of* the plaintext.

This is great! Actually, it is too-good-to-be-true unfortunately. The adversary may already know something *in advance as prior knowledge* even not looking at the ciphertext. Since no encryption achieve this definition, it is not useful.

Intuitive definition.

> In a secure encryption, given any *a priori* information about the message, the adversary cannot learn any *additional* information about the plaintext by observing the ciphertext.

This is the desired definition, and we next formalize it using probability.

Formal definition. Let $\mathcal{M}$ be the space of all messages we want to encrypt. We model the a priori information as (conditional) probability distributions.

Definition 1 (Private-key encryption)**.** A tuple of algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is said to be a *private-key encryption scheme* over the messages space $\mathcal{M}$ and the keyspace $\mathcal{K}$ if the following syntax, correctness, and security holds.

- Gen is a randomized algorithm that returns a key $k \in \mathcal{K}$. We denote by $k \leftarrow \mathsf{Gen}$ the process of generating k.
- Enc is a potentially randomized algorithm that on input a key $k \in \mathcal{K}$ and a message $m \in \mathcal{M}$, outputs a ciphertext c. We denote by $c \leftarrow \mathsf{Enc}_k(m)$ the computation of Enc on k and m.
- Dec is a deterministic algorithm that on input input a key k and a ciphertext c outputs a message $m \in \mathcal{M}$. We denote by $m' := \mathsf{Dec}_k(c)$ the computation of Dec.

For correctness, we require that for all $m \in \mathcal{M}$, it holds that

$$\Pr[k \leftarrow \mathsf{Gen} : \mathsf{Dec}_k(\mathsf{Enc}_k(m)) = m] = 1,$$

where the probability is taken over the randomness of $\mathsf{Gen}, \mathsf{Enc}$. The security will be defined below in Shannon or perfect secrecy.

Notation: throughout this course, the left arrow $x \leftarrow D$ denotes the following depending on the context:

- When D is a randomized algorithm, we mean that the random variable x is the randomized output of D.
- When D is a distribution, we mean that the random variable x is sampled from D.
- When D is a set, we mean that the random variable x is sampled uniformly at random from D.

On the other hand, we use the symbol $:=$ to denote the assignment of a deterministic output.

Definition 2 (Shannon secrecy)**.** The private-key encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ with respect to $(\mathcal{M}, \mathcal{K})$ is *Shannon-secret* if for all distribution D over $\mathcal{M}$, for all $m' \in \mathcal{M}$ and for all c, it holds that

$$\Pr[k \leftarrow \mathsf{Gen}; m \leftarrow D : \quad m = m' \mid \mathsf{Enc}_k(m) = c] \quad = \quad \Pr[m \leftarrow D : m = m'].$$

Notation: we sometimes put the random experiment in the probability notation and use semicolon to separate the experiment (e.g., sampling k and m) and the event (e.g., $\mathsf{Enc}_k(m) = c$ in the above).

References

[Sha49] C. E. Shannon. Communication theory of secrecy systems. *The Bell System Technical Journal*, 28(4):656–715, 1949.