A Course in Cryptography

RAFAEL PASS
ABHI SHELAT

Lecture Notes 2007

Notation

Algorithms

Let A denote an algorithm. We write $A(\cdot)$ to denote an algorithm with one input and $A(\cdot,\cdot)$ for two inputs. In general, the output of a (randomized) algorithm is described by a probability distribution; we let $A(x)$ denotes the probability distribution associated with the output of A on input x. An algorithm is said to be deterministic if the probability distribution is concentrated on a single element.

Experiments

We denote by $x \leftarrow S$ the experiment of sampling an element x from a probability distribution S. If F is a finite set, then $x \leftarrow F$ denotes the experiment of sampling *uniformly* from the set F. We use semicolon to describe the ordered sequences of event that make up an experiment, e.g.,

$$x \leftarrow S;\ (y, z) \leftarrow A(x)$$

Probabilities

If $p(.,.)$ denotes a predicate, then

$$\Pr[x \leftarrow S;\ (y, z) \leftarrow A(x) : p(y, z)]$$

is the probability that the predicate $p(y, z)$ is true after the ordered sequence of events $(x \leftarrow S;\ (y, z) \leftarrow A(x))$. The notation $\{x \leftarrow S;\ (y, z) \leftarrow A(x) : (y, z)\}$ denotes the probability distribution over $\{y, z\}$ generated by the experiment $x \leftarrow S;\ (y, z) \leftarrow A(x)$.

Chapter 1

Introduction

The word cryptography stems from the Greek words *kryptós*—meaning "hidden"—and *gráfein*—meaning " to write". Indeed, the classical cryptographic problem, which dates back millenia, considers the task of using "hidden writing" to secure, or conceal communication between two parties.

1.1 Classical Cryptography: Hidden Writing

Consider two parties, Alice and Bob. Alice wants to *privately* send messages (called *plaintexts*) to Bob over an *insecure channel*. By an insecure channel, we here refer to an "open" and tappable channel; in particular, Alice and Bob would like their privacy to be maintained even in face of an *adversary* Eve (for eavesdropper) who listens to all messages sent on the channel. How can this be achieved?

A possible solution Before starting their communication, Alice and Bob agree on some "secret code" that they will later use to communicate. A secret code consists of a *key*, an algorithm, Enc, to *encrypt* (scramble) plaintext messages into *ciphertexts* and an algorithm Dec to *decrypt* (or descramble) ciphertexts into plaintext messages. Both the encryption and decryption algorithms require the key to perform their task.

Alice can now use the key to encrypt a message, and the send the ciphertext to Bob. Bob, upon receiving a ciphertext, uses the key to decrypt the ciphertext and retrieve the orignal message.

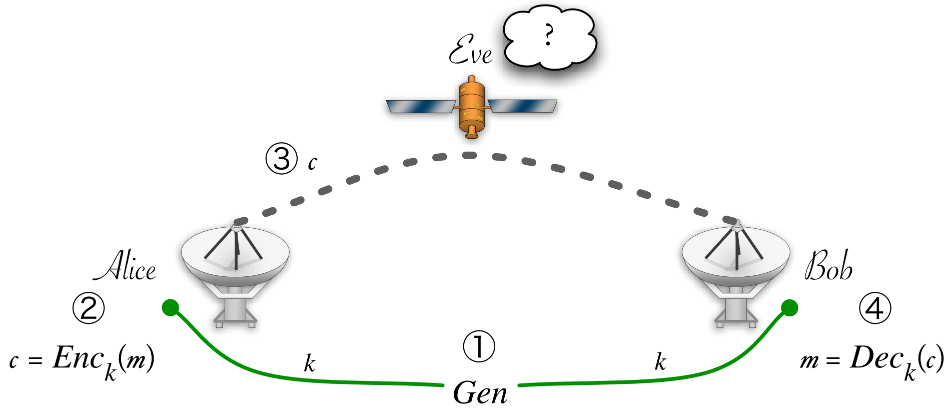

Figure 1.1: Illustration of the steps involved in private-key encryption. First, a key k must be generated by the Gen algorithm and privately given to Alice and Bob. In the picture, this is illustrated with a green "land-line." Later, Alice encodes the message m into a ciphertext c and sends it over the insecure channel—in this case, over the airwaves. Bob receives the encoded message and decodes it using the key k to recover the original message m. The eavesdropper Eve does not learn anything about m except perhaps its length.

Private-Key Encryption

To formalize the above task, we must consider an additional algorithm, Gen, called the *key-generation* algorithm; this algorithm is executed by Alice and Bob to generate the key k which they use to encrypt and decrypt messages.

A first question that needs to be addressed is what information needs to be "public"—i.e., known to everyone—and what needs to "private"—i.e., kept secret. In the classical approach—*security by obscurity*—all of the above algorithms, Gen, Enc, Dec, and the generated key k were kept private; the idea was of course that the less information we give to the adversary, the harder it is to break the scheme. A design principle formulated by Kerchoff in 1884—known as *Kerchoff's principle*—instead stipulates that the only thing that one should assume to be private is the key k; everything else (i.e., Gen, Encand Dec) should be assumed to be public! Why should we do this? Designs of encryption algorithms are often eventually leaked, and when this happens the effects to privacy could be disastreous. Suddenly the scheme might be completely broken; this might even be the case if just a part of the algorithm's description is leaked. The more conservative approach advocated by Kerchoff instead guarantees that security is preserved even if everything but the key is known to the adversary. Furthermore, if a publicly known encryption scheme still has not been broken, this gives us more confidence in its "true" security (rather than if only the few people that designed it were unable to break it). As we will see later, Kerchoff's principle will be the first step to formally defining the security of encryption schemes.

Note that an immediate consequence of Kerchoff's principle is that all of the algorithms $\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}$ can not be *deterministic*; if this were so, then Eve would be able to compute everything that Alice and Bob could compute and would thus be able to decrypt anything that Bob can decrypt. In particular, to prevent this we must require the key generation algorithm, Gen, to be randomized.

Definition 3.1. (Private-key Encryption). A triplet of algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is called a *private-key* encryption scheme over the messages space $\mathcal{M}$ and the keyspace $\mathcal{K}$ if the following holds:

1. Gen(called the *key generation algorithm*) is a randomized algorithm that returns a key k such that $kin\mathcal{K}$. We denote by $k \leftarrow \mathsf{Gen}$ the process of generating a key k.
2. Enc (called the *encryption algorithm*) is an (potentially randomized) algorithm that on input a key $k \in K$ and a message $m \in M$, outputs a ciphertext c. We denote by $c \leftarrow \mathsf{Enc}_k(m)$ the output of Enc on input key k and message m.
3. Dec (called the *decryption algorithm*) is a deterministic algorithm that on input a key k and a ciphertext c and outputs a message m.
4. For all $m \in M$ and $k \in K$,

$$Pr[Dec_k(Enc_k(m)) = m] = 1$$

To simplify notation we also say that $(\mathcal{M}, \mathcal{K}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a private-key encryption scheme if $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a *private-key* encryption scheme over the messages space $\mathcal{M}$ and the keyspace $\mathcal{K}$. To simplify further, we sometimes say that $(\mathcal{M}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a private-key encryption scheme if there exists some key space $\mathcal{K}$ such that $(\mathcal{M}, \mathcal{K}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a private-key encryption scheme.

Note that the above definition of a private-key encryption scheme does not specify any secrecy (or privacy) properties; the only non-trivial requirement is that the decryption algorithm Dec uniquely recovers the messages encrypted using Enc (if these algorithms are run on input the same key k).

Later in course we will return to the task of defining secrecy. However, first, let us provide some historical examples of private-key encryption schemes and colloquially discuss their "security" without any particular definition of secrecy in mind.

Some Historical Ciphers

The *Ceasar Cipher* (named after Julius Ceasar who used it to communicate with his generals) is one of the simplest and well-known private-key encryption schemes. The encryption method consist of replacing each letter in the message with one that is a fixed number of places down the alphabet. More precisely,

Definition 4.1. The *Ceasar Cipher* denotes the tuple $(\mathcal{M}, \mathcal{K}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ defined as follows.

$$\begin{array}{rcl} \mathcal{M} & = & \{A, B, \ldots, Z\}^* \\ \mathcal{K} & = & \{0, 1, 2, \ldots, 25\} \\ \mathsf{Gen} & = & k \text{ where } k \xleftarrow{r} \mathcal{K}. \\ Enc_k(m_1 m_2 \ldots m_n) & = & c_1 c_2 \ldots c_n \text{ where } c_i = m_i + k \bmod 26 \\ Dec_k(c_1 c_2 \ldots c_n) & = & m_1 m_2 \ldots m_n \text{ where } m_i = c_i - k \bmod 26 \end{array}$$

In other words, encryption is a cyclic shift of the same length (k) on each letter in the message and the decryption is a cyclic shift in the opposite direction. We leave to reader to verify the following proposition.

Proposition 4.1. *The* Caesar Cipher *is a private-key encryption scheme.*

At first glance, messages encrypted using the Ceasar Cipher look "scrambled" (unless k is known). However, to break the scheme we just need to try all 26 different values of k (which is easily done) and see if what we get back is something that is readable. If the message is "relatively" long, the scheme is easily broken. To prevent this simple *brute-force* attack, let us modify the scheme.

In the improved *Substitution Cipher* we replace letters in the message based on an arbitrary permutation over the alphabet (and not just cyclic shifts as in the Caesar Cipher).

Definition 4.2. The *Subsitution Cipher* denotes the tuple $\mathcal{M}, \mathcal{K}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}$ defined as follows.

$$\begin{aligned}
\mathcal{M} &= \{A, B, \ldots, Z\}^* \\
\mathcal{K} &= \text{the set of permutations over } \{A, B, \ldots, Z\} \\
\mathsf{Gen} &= k \text{ where } k \overset{r}{\leftarrow} \mathcal{K}. \\
Enc_k(m_1 m_2 \ldots m_n) &= c_1 c_2 \ldots c_n \text{ where } c_i = k(m_i) \\
Dec_k(c_1 c_2 \ldots c_n) &= m_1 m_2 \ldots m_n \text{ where } m_i = k^{-1}(c_i)
\end{aligned}$$

Proposition 5.1. *The* Subsitution Cipher *is a private-key encryption scheme.*

To attack the substitution cipher we can no longer perform the brute-force attack because there are now $26!$ possible keys. However, by performing a careful frequency analysis of the alphabet in the English language, the key can still be easily recovered (if the encrypted messages is sufficiently long)!

So what do we do next? Try to patch the scheme again? Indeed, cryptography historically progressed according to the following "crypto-cycle":

1. **A**, the "artist", invents an encryption scheme.

2. **A** claims (or even mathematically proves) that *known attacks* do not work.

3. The encryption scheme gets employed widely (often in critical situations).

4. The scheme eventually gets broken by improved attacks.

5. Restart, usually with a patch to prevent the previous attack.

Thus, historically, the main job of a cryptographer was *cryptoanalysis*—namely, trying to break encryption algorithms. Whereas cryptoanalysis i still is an important field of research, the philosophy of modern cryptography is instead "if we can do the cryptography part right, there is no need for cryptanalysis!".

1.2 Modern Cryptography: Provable Security

Modern Cryptography is the transition from cryptography as an *art* to cryptography as a principle-driven *science*. Instead of inventing ingenious ad-hoc schemes, modern cryptography relies on the following paradigms:

- Providing mathematical *definitions of security*.

- Providing *precise mathematical assumptions* (e.g. "factoring is *hard*", where *hard* is formally defined). These can be viewed as axioms.
- Providing *proofs of security*, i.e., proving that, if some particular scheme can be broken, then it contradicts our assumptions (or axioms). In other words, if the assumptions were true, the scheme cannot be broken.

This is the approach that we will consider in this course.

As we shall see, despite its conservative nature, we will be able to obtain solutions to very paradoxical problems that reach far beyond the original problem of security communication.

Beyond Secure Communication

In the original motivating problem of secure communication, we had two honest parties, Alice and Bob and a malicious eavesdropper Eve. Suppose, Alice and Bob in fact do not trust each other but wish to perform some joint computation. For instance, Alice and Bob each have a (private) list and wish to find the intersection of the two list without revealing anything else about the contents of their lists. Such a situation arises, for example, when two large financial institutions which to determine their "common risk exposure," but wish to do so without revealing anything else about their investments. One good solution would be to have a trusted center that does the computation and reveals only the answer to both parties. But, would either bank trust the "trusted" center with their sensitive information? Using techniques from modern cryptography, a solution can be provided without a trusted party. In fact, the above problem is a special case of what is known as *secure two-party computation*.

Secure two-party computation - informal definition: A secure two-party computation allows two parties A and B with private inputs a and b respectively, to compute a function $f(a, b)$ that operates on joint inputs a, b while guaranteeing the same *correctness* and *privacy* as if a trusted party had performed the computation for them, even if either A or B try to deviate in any possible malicious way.

Under certain number theoretic assumptions (such as "factoring is hard"), there exists a protocol for secure two-party computation.

The above problem can be generalized also to situations with multiple distrustful parties. For instance, consider the task of electronic elections: a set of n parties which to perform an election in which it is guaranteed that the votes are correctly counted, but these votes should at the same time remain private! Using a so called *multi-party computation* protoocol, this task can be achieved.

A toy example: The match-making game

To illustrate the notion of secure-two party computation we provide a "toy-example" of a secure computation using physical cards. Alice and Bob want to find out if they are meant for each other. Each of them have two choices: either they love the other person or they do not. Now, they wish to perform some interaction that allows them to determine whether there is a match (i.e., if they both love each other) or not—and nothing more! For instance, if Bob loves Alice, but Alice does not love him back, Bob does not want to reveal to Alice that he loves her (revealing this could change his future chances of making Alice love him). Stating it formally, if LOVE and NO-LOVE were the inputs and MATCH and NO-MATCH were the outputs, the function they want to compute is:

$$\begin{aligned} f(\text{LOVE}, \text{LOVE}) &= \text{MATCH} \\ f(\text{LOVE}, \text{NO-LOVE}) &= \text{NO-MATCH} \\ f(\text{NO-LOVE}, \text{LOVE}) &= \text{NO-MATCH} \\ f(\text{NO-LOVE}, \text{NO-LOVE}) &= \text{NO-MATCH} \end{aligned}$$

Note that the function f is simply an *and* gate.

The protocol: Assume that Alice and Bob have access to five cards, three identical hearts($\heartsuit$) and two identical clubs($\clubsuit$). Alice and Bob each get one heart and one club and the remaining heart is put on the table, turned over.

Next Alice and Bob also place their card on the table, also turned over. Alice places her two cards on the left of the heart which is already on the table, and Bob places the hard on the right of the heart. The order in which Alice and Bob place their two cards depend on their input (i.e., if the love the other person or not). If Alice loves, then Alice places her cards as $\clubsuit\heartsuit$; otherwise she places them as $\heartsuit\clubsuit$. Bob on the other hand places his card in the opposite order: if he loves, he places $\heartsuit\clubsuit$, and otherwise places $\clubsuit\heartsuit$. These orders are illustrated in Fig. 1.2.

When all cards have been placed on the table, the cards are piled up. Alice and Bob then each take turns to cut the pile of cards (once each). Finally, all cards are revealed. If there are three hearts in a row then there is a match and no-match otherwise.

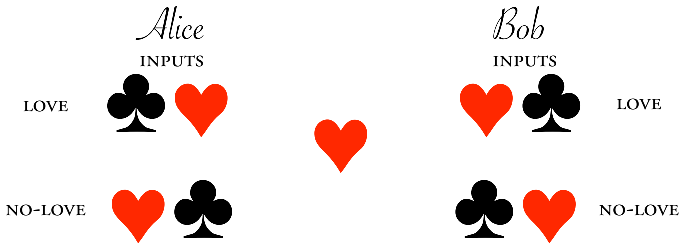

Figure 1.2: Illustration of the Match game with Cards

Analyzing the protocol: We proceed to analyze the above protocol. Given inputs for Alice and Bob, the configuration of cards on the table before the cuts is described in Fig. 1.3. Only the first case—i.e., (LOVE, LOVE)—results in three hearts in a row. Furthermore this property is not changed by the cyclic shift induced by the cuts made by Alice and Bob. We conclude that the protocols correctly computes the desired function.

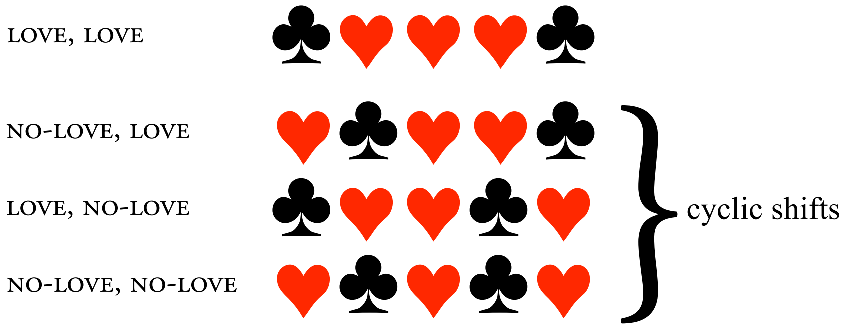

Figure 1.3: The possible outcomes of the Match Protocol. In the case of a mismatch, all three outcomes are cyclic shifts of one-another.

Next, note that in the remaining three cases (when the protocol will output NO-MATCH, all the above configurations are cyclic shifts of one another. If one of Alice and Bob is honest—and indeed performs a random cut— the final card configuration that the parties get to see is identically distributed no matter which of the three cases we were in, in the first place. Thus, even if one of Alice and Bob tries to deviate in the protcol (by not performing a random cut), the privacy of the other party is still maintained.

Zero-knowledge proofs

Zero knowledge proofs is a special case of a secure computation. Informally, in a Zero Knowledge Proof there are two parties, Alice and Bob. Alice wants to convince Bob that some statement is true; for instance, Alice wants to convince Bob that a number N is a product of two primes p, q. A trivial solution would be for Alice to send p and q to Bob. Bob can then check that p and q are primes (we will see later in the course how this can be done) and next multiply the numbers to check if their products N. But this solution reveals p and q. Is this necessary? It turns out that the answer is no. Using a zero-knowledge proof Alice can convince Bob of this statement without revealing p and q.

1.3 Shannon's Treatment of Provable Secrecy

Modern (provable) cryptography started when Claude Shannon formalized the notion of private-key encryption. Thus, let us return to our originl problem of securing communication between Alice and Bob.

Shannon Secrecy

As a first attempt, we might consider the following notion of security:

> The adversary cannot learn (all or part of) the key from the ciphertext.

The problem, however, is that such a notion does not make any guarantees about what the adversary can learn about the *plaintext* message! Another approach might be:

> The adversary cannot learn (all, part of, any letter of, any function of, or any partial information about) the plaintext.

This seems like quite a strong notion. In fact, it is too strong because the adversary may already possess some partial information about the plaintext that is acceptable to reveal. Informed by these attempts, we take as our intuitive definition of security:

> Given some *a priori* information, the adversary cannot learn any additional information about the plaintext by observing the ciphertext.

Such a notion of secrecy was formalized by Claude Shannon in 1949 [**?**] in his seminal paper constituting the birth of modern cryptography.

Definition 9.1. (Shannon secrecy). A private-key encryption scheme $(\mathcal{M}, \mathcal{K}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is said to be *Shannon-secret with respect to the distibution D* over $\mathcal{M}$ if for all $m' \in \mathcal{M}$ and for all c,

$$\mathsf{Pr}\left[k \leftarrow \mathsf{Gen}; m \leftarrow D : m = m' \mid Enc_k(m') = c\right] = \mathsf{Pr}\left[m \leftarrow D : m = m'\right].$$

An encryption scheme is said to be *Shannon secret* if it is Shannon secret with respect to all distributions D over $\mathcal{M}$.

The probability is taken with respect to the random output of Gen, the choice of m and the randomization of Enc. The quantity on the left represents the adversary's *a posteriori* distribution on plaintexts after observing a ciphertext; the quantity on the right, the *a priori* distribution. Since these distributions are required to be equal, we thus require that the adversary does not gain any additional information by observing the ciphertext.

Perfect Secrecy

To gain confidence in our definition, we also provide an alternative approach to defining security of encryption schemes. The notion of *perfect secrecy* requires that the distribution of ciphertexts for any two messages are identical. This formalizes our intuition that the ciphertexts carry no information about the plaintext.

Definition 10.1. (Perfect Secrecy). A private-key encryption scheme $(\mathcal{M}, \mathcal{K}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is *perfectly secret* if for all m_1 and m_2 in $\mathcal{M}$, and for all c,

$$\mathsf{Pr}\left[k \leftarrow \mathsf{Gen} : Enc_k(m_1) = c\right] = \mathsf{Pr}\left[k \leftarrow \mathsf{Gen} : Enc_k(m_2) = c\right].$$

Perhaps not surprisingly, the above two notions of secrecy are equivalent.

Theorem 10.1. *An encryption scheme is perfectly secrect if and only if it is Shannon secret.*

Proof. We prove each implication separately. To simplify the proof, we let the notation $\mathsf{Pr}_k\left[\cdot\right]$ denote $\mathsf{Pr}\left[k \leftarrow \mathsf{Gen}\right]$, $\mathsf{Pr}_m\left[\cdot\right]$ denote $\mathsf{Pr}\left[m \leftarrow D : \cdot\right]$, and $\mathsf{Pr}_{k,m}\left[\cdot\right]$ denote $\mathsf{Pr}\left[k \leftarrow \mathsf{Gen}; m \leftarrow D : \cdot\right]$.

Perfect secrecy implies Shannon secrecy. Assume that $(\mathcal{M}, \mathcal{K}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is perfectly secret. Consider any distribution D over $\mathcal{M}$, any message $m' \in \mathcal{M}$, and any c. We show that

$$\Pr_{k,m}\left[m = m' \mid Enc_k(m) = c\right] = \Pr_m\left[m = m'\right].$$

By the definition of conditional probabilities, the l.h.s. can be rewritten as

$$\frac{\Pr_{k,m}[m = m' \cap Enc_k(m) = c]}{\Pr_{k,m}[Enc_k(m) = c]}$$

which can be further simplified as

$$\frac{\Pr_{k,m}[m = m' \cap Enc_k(m') = c]}{\Pr_{k,m}[Enc_k(m) = c]} = \frac{\Pr_m[m = m']\Pr_k[Enc_k(m') = c]}{\Pr_{k,m}[Enc_k(m) = c]}$$

The central idea behind the proof is to show that

$$\Pr_{k,m}[Enc_k(m) = c] = \Pr_k\left[Enc_k(m') = c\right]$$

which establishes th result. Recall that,

$$\Pr_{k,m}[Enc_k(m) = c] = \sum_{m'' \in \mathcal{M}} \Pr_m\left[m = m''\right]\Pr_k\left[Enc_k(m'') = c\right]$$

The perfect secrecy condition allows us to replace the last term to get:

$$\sum_{m'' \in \mathcal{M}} \Pr_m\left[m = m''\right]\Pr_k\left[Enc_k(m') = c\right]$$

This last term can be moved out of the summation and simplified as:

$$\Pr_k\left[Enc_k(m') = c\right]\sum_{m'' \in \mathcal{M}} \Pr_m\left[m = m''\right] = \Pr_k\left[Enc_k(m') = c\right]$$

Shannon secrecy implies perfect secrecy. Assume that $(\mathcal{M}, \mathcal{K}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is Shannon secret. Consider $m_1, m_2 \in \mathcal{M}$, and any c. Let D be the uniform distribution over $\{m_1, m_2\}$. We show that

$$\Pr_k[Enc_k(m_1) = c] = \Pr_k[Enc_k(m_2) = c].$$

The definition of D implies that $\Pr_m[m = m_1] = \Pr_m[m = m_2] = \frac{1}{2}$. It therefore follows by Shannon secrecy that

$$\Pr_{k,m}[m = m_1 \mid Enc_k(m) = c] = \Pr_{k,m}[m = m_2 \mid Enc_k(m) = c]$$

By the definition of conditional probability,

$$\begin{aligned}
\mathsf{Pr}_{k,m}\left[m = m_1 \mid Enc_k(m) = c\right] &= \frac{\mathsf{Pr}_{k,m}\left[m = m_1 \cap Enc_k(m) = c\right]}{\mathsf{Pr}_{k,m}\left[Enc_k(m) = c\right]} \\
&= \frac{\mathsf{Pr}_{m}\left[m = m_1\right] \mathsf{Pr}_{k}\left[Enc_k(m_1) = c\right]}{\mathsf{Pr}_{k,m}\left[Enc_k(m) = c\right]} \\
&= \frac{\frac{1}{2} \cdot \mathsf{Pr}_{k}\left[Enc_k(m_1) = c\right]}{\mathsf{Pr}_{k,m}\left[Enc_k(m) = c\right]}
\end{aligned}$$

Analogously,

$$\mathsf{Pr}_{k,m}\left[m = m_2 \mid Enc_k(m) = c\right] = \frac{\frac{1}{2} \cdot \mathsf{Pr}_{k}\left[Enc_k(m_2) = c\right]}{\mathsf{Pr}_{k,m}\left[Enc_k(m) = c\right]} .$$

Cancelling and rearranging terms, we conclude that

$$\mathsf{Pr}_{k}\left[Enc_k(m_1) = c\right] = \mathsf{Pr}_{k}\left[Enc_k(m_2) = c\right] .$$

□

The One-Time Pad

Given our definition of security, we turn to the question of whether perfectly secure encryption schemes exists. It turns out that both the encryption schemes we have seen so far (i.e., the Caesar and Substitution ciphers) are secure as long as we only consider messages of length 1. However, when considering messages of length 2 (or more) the schemes are no longer secure—in fact, it is easy to see that encryptions of the strings AA and AB have disjoint distributions, thus violating perfect secrecy (prove this!).

Nevertheless, this suggests that we might obtain perfect secrecy by somehow adapting these schemes to operate on each element of a message independently. This is the intuition behind the *one-time pad* encryption scheme, invented by Gilbert Vernam and Joseph Mauborgne in 1919.

Definition 12.1. The One-Time Pad encryption scheme is described by the following tuple $(\mathcal{M}, \mathcal{K}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$.

$$\begin{aligned}
\mathcal{M} &= \{0,1\}^n \\
\mathcal{K} &= \{0,1\}^n \\
\mathsf{Gen} &= k = k_1 k_2 \ldots k_n \leftarrow \{0,1\}^n \\
Enc_k(m_1 m_2 \ldots m_n) &= c_1 c_2 \ldots c_n \text{ where } c_i = m_i \oplus k_i \\
Dec_k(c_1 c_2 \ldots c_n) &= m_1 m_2 \ldots m_n \text{ where } m_i = c_i \oplus k_i
\end{aligned}$$

The $\oplus$ operator represents the binary xor operation.

Proposition 12.1. *The One-Time Pad is a perfectly secure private-key encryption scheme.*

Proof. It is straight-forward to verify that the One Time Pad is a private-key encryption scheme. We turn to show that the One-Time Pad is perfectly secret and begin by showing the the following claims.

Claim 13.1. *For any* $c, m \in \{0,1\}^n$,

$$\mathsf{Pr}\left[k \leftarrow \{0,1\}^n : Enc_k(m) = c\right] = 2^{-k}$$

Claim 13.2. *For any* $c \notin \{0,1\}^n, m \in \{0,1\}^n$,

$$\mathsf{Pr}\left[k \leftarrow \{0,1\}^n : Enc_k(m) = c\right] = 0$$

Claim 13.1 follows from the fact that for any $m, c \in \{0,1\}^n$ there is only one k such that $Enc_k(m) = m \oplus k = c$, namely $k = m \oplus c$. Claim 13.2 follows from the fact that for every $k \in \{0,1\}^n$ $Enc_k(m) = m \oplus k \in \{0,1\}^n$.

From the claims we conclude that for any $m_1, m_2 \in \{0,1\}^n$ and every c it holds that

$$\mathsf{Pr}\left[k \leftarrow \{0,1\}^n : Enc_k(m_1) = c\right] = \mathsf{Pr}\left[k \leftarrow \{0,1\}^n : Enc_k(m_2) = c\right]$$

which concludes the proof. □

So perfect secrecy is obtainable. But at what cost? When Alice and Bob meet to generate a key, they must generate one that is as long as all the messages they will send until the next time they meet. Unfortunately, this is not a consequence of the design of the One-Time Pad, but rather of perfect secrecy, as demonstrated by Shannon's famous theorem.

Shannon's Theorem

Theorem 13.1 (Shannon)**.** *If an encryption scheme* $(\mathcal{M}, \mathcal{K}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is perfectly secret, then* $|\mathcal{K}| \geq |\mathcal{M}|$.

Proof. Assume there exists a perfectly secret $(\mathcal{M}, \mathcal{K}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ such that $|\mathcal{K}| < |\mathcal{M}|$. Take any $m_1 \in \mathcal{M}$, $k \in \mathcal{K}$, and let $c \leftarrow Enc_k(m_1)$. Let $\mathbf{Dec}(c)$ denote the set $\{m \mid \exists k \in \mathcal{K} \;.\; m = Dec_k(c)\}$ of all possible decryptions of c under all possible keys. As Dec is deterministic, this set has size at most $|\mathcal{K}|$. But since $|\mathcal{M}| > |\mathcal{K}|$, there exists some message m_2 not in $\mathbf{Dec}(c)$. By the definition of a private encryption scheme it follows that

$$\mathsf{Pr}\left[k \leftarrow \mathcal{K} : Enc_k(m_2) = c\right] = 0$$

But since

$$\Pr\left[k \leftarrow \mathcal{K} : Enc_k(m_1) = c\right] > 0$$

we conclude that

$$\Pr\left[k \leftarrow \mathcal{K} : Enc_k(m_1) = c\right] \neq \Pr\left[k \leftarrow \mathcal{K} : Enc_k(m_2) = c\right] > 0$$

which contradicts the perfect secrecy of $(\mathcal{M}, \mathcal{K}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$. □

Note that proof of Shannon's theorem in fact describes an *attack* on every private-key encryption scheme $(\mathcal{M}, \mathcal{K}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ such that $|\mathcal{K}| < |\mathcal{M}|$. In fact, it follows that for any such encryption scheme there exists $m_1, m_2 \in M$ and a constant $\epsilon > 0$ such that

$$\Pr\left[k \leftarrow \mathcal{K}; Enc_k(m_1) = c : m_1 \in \mathbf{Dec}(c)\right] = 1$$

but

$$\Pr\left[k \leftarrow \mathcal{K}; Enc_k(m_1) = c : m_2 \in \mathbf{Dec}(c)\right] \leq 1 - \epsilon$$

The first equation follows directly from the definition of private-key encryption, whereas the second equation follows from the fact that (by the proof of Shannon's theorem) there exists some c in the domain of $Enc.(m_1)$ such that $m_2 \notin \mathbf{Dec}(c)$. Consider, now, a scenario where Alice uniformly picks a message m from $\{m_1, m_2\}$ and sends the encryption of m to Bob. We claim that Eve, having seen the encryption c of m can guess whether $m = m_1$ or $m = m_2$ with probability higher than $\frac{1}{2}$. Eve, upon receiving c simply checks if $m_2 \in \mathbf{Dec}(c)$. If $m_2 \notin \mathbf{Dec}(c)$, Eve guesses that $m = m_1$, otherwise she makes a random guess. If Alice sent the message $m = m_2$ then $m_2 \in \mathbf{Dec}(c)$ and Eve will guess correctly with probability $\frac{1}{2}$. If, on the other hand, Alice sent $m = m_1$, then with probability ϵ, $m_2 \notin \mathbf{Dec}(c)$ and Eve will guess correctly with probability 1, whereas with probability $1 - \epsilon$ Eve will make a random guess, and thus will be correct with probability $\frac{1}{2}$. We conclude that Eve's success probability is

$$\frac{1}{2} \cdot \frac{1}{2} + \frac{1}{2} \cdot (\epsilon \cdot 1 + (1 - \epsilon) \cdot \frac{1}{2}) = \frac{1}{2} + \frac{\epsilon}{2}$$

Thus we have exhibited a very *concise* attack for Eve, which makes it possible for her to guess what message Alice sends with probability better than $\frac{1}{2}$.

A possible critique against our argument is that if ϵ is very small (e.g., 2^{-100}), then the utility of this attack is limited. However, as we shall see in the following stonger version of Shannon's theorem which states that if $\mathcal{M} = \{0,1\}^n$ and $\mathcal{K} = \{0,1\}^{n-1}$ (i.e., if the key is simply one bit shorter than the message), then $\epsilon = \frac{1}{2}$.

Theorem 14.1. *Let* $(\mathcal{M}, \mathcal{K}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be a private-key encryption scheme where* $\mathcal{M} = \{0,1\}^n$ *and* $\mathcal{K} = \{0,1\}^{n-1}$*. Then, there exist messages* $m_0, m_1 \in \mathcal{M}$ *such that*

$$\Pr\left[k \leftarrow \mathcal{K}; Enc_k(m_1) = c : m_2 \in \mathbf{Dec}(c)\right] \leq \frac{1}{2}$$

Proof. Given $c \leftarrow Enc_k(m)$ for some key $k \in \mathcal{K}$ and message $m \in \mathcal{M}$, consider the set $\mathbf{Dec}(c)$. Since Dec is deterministic it follows that $|\mathbf{Dec}(c)| \leq |\mathcal{K}| = 2^{n-1}$. Thus, for all $m_1 \in \mathcal{M}, k \leftarrow \mathcal{K}$,

$$\Pr\left[m' \leftarrow \{0,1\}^n; c \leftarrow Enc_k(m_1) : m' \in \mathbf{Dec}(c)\right] \leq \frac{2^{n-1}}{2^n} = \frac{1}{2}$$

Since the above probability is bounded by $\frac{1}{2}$ for *every* key $k \in \mathcal{K}$, this must also hold for a random $k \leftarrow Gen$.

$$\Pr\left[m' \leftarrow \{0,1\}^n; k \leftarrow \mathsf{Gen}; c \leftarrow Enc_k(m_1) : m' \in \mathbf{Dec}(c)\right] \leq \frac{1}{2} \tag{1.1}$$

Additionally, since the bound holds for a random message m', there must exists some *particular* message m_2 that minimizes the probability. In other words, for every message $m_1 \in \mathcal{M}$, there exists some message $m_2 \in \mathcal{M}$ such that

$$\Pr\left[k \leftarrow \mathsf{Gen}; c \leftarrow Enc_k(m_1) : m_2 \in \mathbf{Dec}(c)\right] \leq \frac{1}{2}$$

□

Remark 15.1. Note that the theorem is stronger than stated. In fact, we showed that for *every* $m_1 \in \mathcal{M}$, there exist some string m_2 that satisfies the desired condition. We also mention that if we content ourselves with getting a bound of $\epsilon = \frac{1}{4}$, the above proof actually shows that for *every* $m_1 \in \mathcal{M}$, it holds that for at least *one fourth* of the messages $m_2 \in \mathcal{M}$,

$$\Pr\left[k \leftarrow \mathcal{K}; Enc_k(m_1) = c : m_2 \in \mathbf{Dec}(c)\right] \leq \frac{1}{4};$$

otherwise we would contradict equation (1.1).

Thus, by relying on Theorem 14.1, we conclude that if the key length is only one bit shorter than the message message length, there exist messages m_1, m_2 such that Eve's success probability is $\frac{1}{2} + \frac{1}{4} = \frac{3}{4}$ (or alternatively, there exist *many* messages m_1, m_2 such that Eve's success probability is at least $\frac{1}{2} + \frac{1}{8} = \frac{5}{8}$). This is clearly not acceptable in most applications of an encryption scheme! So, does this mean that to get any "reasonable" amount of security Alice and Bob must share a long key?

Note that although Eve's attack only takes a few lines of code to describe, its running-time is high. In fact, to perform her attack—which amounts to checking whether $m_2 \in \mathbf{Dec}(c)$)—Eve must try all possible keys $k \in \mathcal{K}$ to check whether c possibly could decrypt to m_2. If, for instance, $\mathcal{K} = \{0,1\}^n$, this requires her to perform 2^n (i.e., exponentially many) different decryptions! Thus, although the attack can be simply described, it is not "feasible" by any *efficient* computing device. This motivates us to consider only "feasible" adversaries—namely adversaries that are *computationally bounded*. Indeed, as we shall see later in the course, with respect to such adversaries, the implications of Shannon's Theorem can be overcome.

1.4 Overview of the Course

In this course we will focus on some of the key *concepts* and *techniques* in modern cryptography. The course will be structured around the following notions:

Computational Hardness and One-way Functions. As illustrated above, to circumvent Shannon's lower bound we have to restrict our attention to computationally bounded adversaries. The first part of the course deals with notions of resource-bounded (and in particular *time-bounded*) computation, computational hardness, and the notion of one-way functions. One-way functions—i.e., functions that are "easy" to compute, but "hard" to invert (for computationally-bounded entities)—are at the heart of most modern cryptographic protocols.

Indistinguishability. The notion of (computational) indistinguishability formalizes what it means for a computationally-bounded adversary to be unable to "tell apart" two distributions. This notion is central to modern definitions of security for encryption schemes, but also for formally defining notions such as pseudo-random generation, bit commitment schemes, and much more.

Knowledge. A central desiderata in the design of cryptographic protocols is to ensure that the protocol execution *does not leak more "knowledge" than what is necessary*. In this part of the course, we provide and investigate "knowledge-based" (or rather *zero knowledge*-based) definitions of security.

Authentication. Notions such as *digital signatures* and *messages authentication codes* are digital analogues of traditional written signatures. We explore different notions of authentication and show how cryptographic techniques can be used to obtain new types of authentication mechanism not achievable by traditional written signatures.

Composability. It turns out that cryptographic schemes that are secure when executed in isolation can be completely compromised if many instances of the scheme are simultaneously executed (as is unavoidable when executing cryptographic protocols in modern networks). The question of *composability* deals with issues of this type.

Computing on Secret Inputs. Finally, we consider protocols which allow mutually distrustful parties to perform arbitrary computation on their respective (potentially secret) inputs. This includes *secret-sharing* protocols and *secure two-party (or multi-party) computation* protocols. We have described the later earlier in this chapter; secret-sharing protocols are methods which allow a set of n parties to receive "shares" of a secret with the property that any "small" subset of shares leaks no information about the secret, but once an appropriate number of shares are collected the whole secret can be recovered.

Appendix A

Basic Probability

Basic Facts

- Events A and B are said to be *independent* if

$$Pr[A \cap B] = Pr[A] \cdot Pr[B]$$

- The *conditional probability* of event A given event B, written as $\Pr[A \mid B]$ is defined as

$$\Pr[A \mid B] = \frac{\Pr[A \cap B]}{\Pr[B]}$$

- *Bayes theorem* relates the $\Pr[A \mid B]$ with $\Pr[B \mid A]$ as follows:

$$\Pr[A \mid B] = \frac{\Pr[B \mid A] \Pr[A]}{\Pr[B]}$$

- Events $A_1, A_2, \dots, A_n$ are said to be *pairwise independent* if for every i and every $j \neq i$, A_i and A_j are independent.

- *Union Bound:* Let $A_1, A_2, \dots, A_n$ be events. Then,

$$Pr[A_1 \cup A_2 \cup \ldots \cup A_n] \leq Pr[A_1] + Pr[A_2] + \ldots Pr[A_n]$$

- Let X be a random variable with range Ω. The *expectation* of X is a number defined as follows.

$$E[X] = \sum_{x \in \Omega} x Pr[X = x]$$