Massachusetts Institute of Technology
6.857: Network and Computer Security
Professors Ronald L. Rivest and Yael Tauman Kalai

Handout R2
February 26, 2020
Andrés Fábrega

Recitation 2: Encryption

1 Preliminaries

In class, we saw the definition of an encryption scheme:

Definition 1.1. An *encryption scheme* is a triplet of PPT algorithms $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ with an associated (finite) message space $\mathcal{M}$, key space $\mathcal{K}$, and ciphertext space $\mathcal{C}$ such that:

- $\mathsf{Gen}(1^n)$: outputs a secret key $k \in \mathcal{K}$.
- $\mathsf{Enc}(k, m)$: on input $k \in \mathcal{K}, m \in \mathcal{M}$, outputs an associated ciphertext $c \in \mathcal{C}$.
- $\mathsf{Dec}(k, c)$: on input $k \in \mathcal{K}, c \in \mathcal{C}$, outputs the associated message $m \in \mathcal{M}$.

Gen must be randomized, but Enc need not be, for now (looking ahead, some security definitions will require this). Also, note that $\mathcal{M}$ and $\mathcal{K}$ are specified a priori, but $\mathcal{C}$ is a function of the latter two sets (i.e., all possible outputs of Enc, when evaluated at all possible combinations of messages and keys). Further, Dec is deterministic, and we assume perfect correctness: $\forall n \in \mathcal{N}, m \in \mathcal{M}$ it holds that

$$\Pr_{k \leftarrow \mathsf{Gen}(1^n)}[\mathsf{Dec}(k, \mathsf{Enc}(k, m)) = m] = 1 \tag{1}$$

Even though this is the "textbook" definition of an encryption scheme, let's analyze more closely the general context in which it is used. The setup is that we have two parties, Alice and Bob, communicating via an insecure channel in the presence of a passive adversary Eve. Our threat model, i.e., our assumptions about Eve, is that she has full read-access to the wire (so, she can read all messages being sent). Further, she knows *everything* about the system (including $\mathcal{M}$ and $\mathcal{K}$), except the secret key (Kerchoff's Principle). Thus, Alice and Bob use an encryption scheme in order to communicate secretly, and keep plaintexts hidden from Eve. A few remarks:

(i) The key generation algorithm (and all other randomized algorithms we will see in this class), assume the fact that parties have access to an unlimited source of randomness (independent, random bits). The way this happens in practice is that we use high-entropy data (keystrokes, mouse movement, radioactive decay, etc) to get uniform bits (a complicated process, outside the scope of this recitation). Note that high-entropy is not guaranteed to be uniform, so it needs to be processed.

(ii) The definition of an encryption scheme implicitly assumes that the encryptor and decryptor hold the same key, but the adversary doesn't. In practice, *key exchange*, the process by which two parties (secretly) agree on a key, is a crucial and non-trivial step that must precede the use of any (symmetric) encryption scheme. If our threat model assumes that Eve can listen into the wire, how/why could she not know the secret key? This sounds like a "chicken-or-the-egg problem": we need to share a key in order to to communicate secretly, but we need to communicate secretly to share a key! We will see later in class how this can be done, using beautiful mathematics.

(iii) The security definitions we saw in class (and our threat model) make no mention of integrity nor authenticity of the message/ciphertext, and were just concerned with confidentiality (i.e., Eve is passsive). After seeing enough messages, can Eve "craft" a valid ciphertext (i.e., such that decrypting it with k yields an element of the message space)? Perhaps she can flip a few bits, and change Alice's message to a closely related one? For example, if Alice encrypt's just a single bit b using the OTP to

get $c \in \{0,1\}$, Eve can trivially find an encryption of $1-b$, even without knowledge of the secret key! So, we say that **an encryption scheme constructs a confidential (albeit unauthenticated) channel from an insecure channel**.

Putting it all together, Alice and Bob communicate via a two-step protocol: (i) generate and agree on a secret key (key exchange), and (ii) use a secure encryption scheme with said key. In the next few lectures, we will expand our threat model to consider active adversaries that can tamper/malleate the ciphertexts. To do this, we will see how integrity and authenticity are defined, and how we can construct primitives with these properties, in order to build schemes that are both confidential and authenticated. With this, we will achieve our goal of constructing a **secure** channel from an insecure channel in the presence of an active adversary. Stay tuned!

2 Information-Theoretic Security

On Wednesday, Prof. Kalai introduce the first security definition (of many we will see this semester): *perfect secrecy*. Intuitively, this means that an all-powerful adversary (unbounded runtime and space complexity) that receives a single ciphertext does not learn anything about the underlying message. In other words, we want all ciphertexts to be completely independent of the messages.

Definition 2.1. An encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is *perfectly secret* if for all $m_1, m_2 \in \mathcal{M}$ and every $c \in \mathcal{C}$ it holds that

$$\Pr[\mathsf{Enc}(k, m_1) = c] = \Pr[\mathsf{Enc}(k, m_2) = c] \tag{2}$$

Where the probabilities are over the random sampling of the key k, and potentially the randomness of Enc (albeit it could be determinstic).

Note that this formula holds even if the adversary knows m_0 or m_1, and our only assumption is that she doesn't know the key. We can rephrase this by saying that $(m_0, \mathsf{Enc}(k, m_0))$ is indistinguishable from $(m_0, \mathsf{Enc}(k, m_1))$.

This formula may be a bit unintuitive to understand at first. However, we can rephrase it in a game-based manner (games are an important part of security definitions, so you should become familiar with them). We will consider a game between the (all-powerful) adversary A, and a challenger C, who holds a secret key k. A and C will then play as follows:

1. A chooses two messages m_0 and m_1 from $\mathcal{M}$, and sends them to C.
2. C flips a coin to choose a bit $b \in 0, 1$ (for example, if it lands on heads pick 0, and if lands on tails pick 1). Based on this, C computes $c \coloneqq \mathsf{Enc}(k, m_b)$. That is, C randomly chooses one of A's messages, and sends her the encryption of it.
3. A, upon receiving c, will try to guess which of her two messages is the one encrypted. Finally, A outputs her guess b'. If $b' = b$, i.e., she guessed the message correctly, A wins. Otherwise, C wins.

We say that a scheme is *perfectly indistinguishable* if and only if, for every possible adversary A, the probability that A wins this game is $\frac{1}{2}$. That is, A has no better strategy than just guessing the bit at random! As mentioned earlier, this is completely analogous to (2), and is just a way to rephrase the equation.

What perfect secrecy tells us is that seeing a ciphertext does not reveal anything about the message. We can write this sentence in the form of an equation: $\forall m \in \mathcal{M}, c \in \mathcal{C}$ s.t. $Pr[C = c] > 0$ it holds that

$$\Pr[M = m | C = c] = \Pr[M = m] \tag{3}$$

Here, M and C are random variables denoting the message being encrypted and the resulting ciphertext, respectively (recall that we have probability distributions over $\mathcal{M}$ and $\mathcal{C}$).

So, seeing a ciphertext does not tell Eve anything that she didn't know already. What is the relationship between (2) and (3)? Are there schemes that satisfy one definition but not the other? No!

Claim. *An encryption scheme is perfectly secure if and only if it satisfies (3). That is, (2) and (3) are equivalent.*

We now have two formulas that can be used interchangeably! When trying to prove the security of a scheme, either one is sufficient. In summary, we saw three analogous formulations of information-theoretic security: perfect secrecy (eq. (2)), perfect indistinguishability (game-based analogue of the former), and eq. (3).

2.1 One-Time Pad

In class we also saw the canonical example of a perfectly-secure scheme: the one-time pad (OTP). Loosely, recall that we encrypt a message m with a key k by computing $k \oplus m$, and we decrypt a ciphertext c by computing $k \oplus c$. Correctness is easy to see: $\mathsf{Dec}(k, \mathsf{Enc}(k, m)) \coloneqq k \oplus (m \oplus k) = m \oplus (k \oplus k) = m$, as desired (x-or is commutative and associative).

Prof. Kalai already proved the fact that OTP is information-theoretic secure by showing that it satisfies eq. (2). Today, we will prove it again, using our alternative formula.

Theorem 2.1. *The OTP is perfectly secret.*

Proof. We will show that OTP satisfies eq. (3), i.e., $\Pr[M = m | C = c] = \Pr[M = m]$ for any $m, c \in \{0,1\}^n$.

By Bayes' Theorem, $\Pr[M = m | C = c] = \frac{\Pr[C=c|M=m] \cdot \Pr[M=m]}{\Pr[C=c]}$. Let's compute each part individually:

First, note the following: $\Pr[C = c | M = m] = \Pr[\mathsf{Enc}(k, m) = c] = \Pr[k \oplus m = c] = \Pr[k = c \oplus m] = \frac{1}{2^n}$. The last equality is true because k was chosen uniformly at random from $\{0,1\}^n$.

Now we need to compute $Pr[C = c]$. By the law of total probability, $Pr[C = c] = \sum_{m' \in \mathcal{M}} (\Pr[C = c | M = m'] \cdot \Pr[M = m'])$. From above, we know $\Pr[C = c | M = m'] = \frac{1}{2^n}$ for all messsages, so we can factor this out to get $Pr[C = c] = \frac{1}{2^n} \sum_{m' \in \mathcal{M}} \Pr[M = m']$. Since the second term is just a sum over all possible values of the random variable, it must be equal to 1. Thus, $Pr[C = c] = \frac{1}{2^n}$.

Putting it all together:

$$\Pr[M = m | C = c] = \frac{\Pr[C = c | M = m] \cdot \Pr[M = m]}{\Pr[C = c]} = \frac{\frac{1}{2^n} \cdot \Pr[M = m]}{\frac{1}{2^n}} = \Pr[M = m]$$

as desired.

3 Perfect Secrecy Is... Not Perfect

We will now cover some limitations of perfect secrecy. As great as it sounds, it is has some serious drawbacks, which is why textbook OTP is not used in practice. In the following sections, we will highlight a few problems of OTP, and generalize them to all perflecty secret schemes.

3.1 Key Reuse

The one-time pad suffers from a big problem: it is one-time secure. Namely, the formulas for perfect secrecy hold for one (and only one) message. As soon as a second message is encrypted with the same key, security

breaks apart: an avdersary can compute $c_1 \oplus c_2 = (k \oplus m_1) \oplus (k \oplus m_2) = m_1 \oplus m_2$. Is this a problem? Yes! We learn information about the messages (for example, we can take any two messages whose x-or is not equal to the above, and know that these two can not be the encrypted messages). This is a problem in practice, and we can use statistical techniques to efficiently find the entire messages (for instance, if the messages are English words, we can exploit letter patterns and frequency). Further, if the messages are a single bit, we can learn if they are the same bit or not.

In general, later in the class we will see that an encryption scheme that is many-time secure *must* be randomized. Intuitively, note that for a many-time perfectly secure scheme, $(\mathsf{Enc}(k, m_1), \mathsf{Enc}(k, m_1))$ and $(\mathsf{Enc}(k, m_1), \mathsf{Enc}(k, m_2))$ must be indistinguishable, for any $m_1, m_2 \in \mathcal{M}$. However, if Enc is deterministic, these are evidently different: the former contains two copies of a bit-string, and the latter (with high probability) two different bit strings.

The condition above (nondeterminism) is necessary but not sufficient for many-time perfect security. As a matter of fact, no set of conditions are sufficient: many-time perfect security is impossible! Namely, if the adversary has various ciphertexts encrypted with the same key, she can enumerate over all keys, and find the key which properly decrypts all ciphertexts. If she has enough of these, with high probability, she has found the correct key.

3.2 Key Length

It is easy to see that the OTP requires the key to be as long as the message: if $|m| > |k|$, then $k \oplus m$ reveals the higher order $|m| - |k|$ bits of m! So, if our message is very long, requiring such a long key is not practical.

Sadly, a more general problem is true for any perfectly secure scheme.

Claim. *For any perfectly secure scheme,* $|\mathcal{K}| \geq |\mathcal{M}|$

Proof. At a high-level, the idea is that, for any ciphertext and any message, there must exist some key that could be used to encrypt the message and get the ciphertext. If the message space is larger than the key space, by exhaustion, there will be some messsage(s) with no associated key. So, an all-powerful adversary can trivially break the security: try all message-key pairs, and eventually will stumble upon a message m for which no key yields the ciphertext. So, she has learned something about the ciphertext: it can't be the encryption of m.

More formally, we prove this by contradiction, i.e., assume $|\mathcal{M}| > |\mathcal{K}|$. Let c be an arbitrary ciphertext in $\mathcal{C}$, and let S be the set of all messages for which there exists a key that decrypts c to them. In "Pythonic" syntax: $S = \{\mathsf{Dec}(k, c)$ `for` $k \in \mathcal{K}\}$. Note that $|S| \leq |\mathcal{K}|$, since there is at most one message per key (Dec is deterministic, so no key-ciphertext pair can output more than one message). By assumption, there must be some message m not in S: $|S| \leq |\mathcal{K}| < |\mathcal{M}|$. I.e., $\Pr[M = m | C = c] = 0$. Recall, however, that perfect secrecy requires $\Pr[M = m | C = c] = \Pr[M = m]$! We thus reach a contradiction.

Note that, for OTP, the condition that $|k| > |m|$ implies that $|\mathcal{K}| > |\mathcal{M}|$, as the lemma above claims. I.e., the claim is a generalization of OTP's problem.

4 Looking Ahead

We just argued that perfect security is impractical, and impossible for more than one message. Is this the end of the road? Should we all just retire as cryptographers? Thankfully, if we relax our threat model and security definitions, we **can** construct pratical, many-time secure schemes. To do this, we will assume "efficient" adversaries, which are only allowed to run in polynomial time, and we will allow some negligible

probability of failure, instead of requiring security to always hold. This is the start of the wonderful world of *computational security*, which we will explore in the next few lectures.