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

Handout R3
February 18, 2022
Andrés Fábrega

Recitation 3: Recap

Today we will do a recapt of the story so far. We are soon going to start switching out of the "secret-key encryption" module of the class and into other topics (hash functions, key exchange, public-key crypto, etc). So, today we will review some of the core concepts we have seen so far, particularly those that have public-key analogues.

We started the class with one motivating question: how can I send a message to another person in the presence of an adversary? This led us to our first primitive: encryption schemes.

1 Encryption Schemes

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

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

In addition to correctness, the other property we care about is, of course, security. There are many definitions of security, and different schemes satisfy some but not others. In general, we think of a security definition as having a *threat model* (what can the adversary see or do) and an *adversarial goal* (what is the adversary trying to do).

A few remarks about encryption:

(i) 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.

(ii) We assume that the adversary knows *everything* about the system (including $\mathcal{M}$ and $\mathcal{K}$), except the secret key (Kerchoff's Principle). For example, the adversary knows the explicit algorithms being used (and their inner workings/implementations).

1.1 Security

On Wednesday, Prof. Kalai introduce the first security definition (of many we will see this semester): *(one-time) perfect security*. 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 1.2. 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

$$\mathsf{Enc}(k, m) \equiv \mathsf{Enc}(k, m') \tag{2}$$

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.

Again, intuitively, 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).

1.2 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).

In class, we showed how the one-time pad satisfies eq. (2). Since eq. (2) is equivalent to eq. (3), we will verify that OTP also satisfies eq. (3).

Theorem 1.1. *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.

The intuition behind this proof (and OTP in general), is that for every ciphertext and every message, I can find a pad that "proves" that this message was the plaintext. So, the adversary has no way to tell which message is actually the encrypted one.

Even though OTP is perfectly secure it has some important limitations!

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.

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: $S = \{\mathsf{Dec}(k, c)\ \texttt{for}\ k \in \mathcal{K}\}$. 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.

1.3 Many-Time Security

We just saw that the one-time pad is not many time secure. What does many-time security even mean?

Definition 1.3. An encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is *many-time secure* if for all $m_1, ..., m_t, m'_1, ..., m'_t \in \mathcal{M}$ (where $|m_i| = |m'_i|$) it holds that

$$(\mathsf{Enc}(k, m_1), ..., \mathsf{Enc}(k, m_t)) \equiv (\mathsf{Enc}(k, m'_1), ..., \mathsf{Enc}(k, m'_t)) \tag{4}$$

The randomness is again over the random sampling of the key k, and the randomness of Enc.

Just like in one-time security, we can rephrase this in the form of a game, analogous to the prior one, except that in step 1 the adversary sends all of $m_1, ..., m_t, m'_1, ..., m'_t$ to the challenger, who encrypts either all of $m_1, ..., m_t$ or $m'_1, ..., m'_t$ in step 2.

As mentioned earlier, note that a many-time secure encryption scheme *must* be probabilistic. Otherwise, it would be very easy to distinguish between the encryptions of (m, m) and (m, m'), since the former would produce to equal ciphertexts and the former two distinct ones, so the adversary can easily tell which pair of ciphertexts got encrypted.

Recall from lecture that there are no many-time secure schemes if the adversary is all powerful! So, we relax our threat model, and assume A is polynomially bounded ("efficient").

Going back to the security game, we can strengthen the threat model even more, to get what we call *CPA security*:

1. The adversary picks any message m and sends this to the challenger. The challenger computes $c \coloneqq \mathsf{Enc}(k, m)$ and sends this back to the adversary.
2. Repeat step 1 as many times as the adversary wants, with any messages.
3. Eventually, A chooses two messages m_0 and m_1 from $\mathcal{M}$, and sends them to C.
4. 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.
5. 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 think of steps 1 and 2 in the game as giving the adversary access to an "encryption oracle". Also, we think of steps 3 and 4 as the "challenge phase". We say that the scheme is *CPA secure* if the adversary wins this game with probability $\frac{1}{2} + \texttt{negl}$.

CPA security is one of the "gold standards" for non-authenticated security. One important question is how realistic is CPA security? What do the encryption oracles even represent? What is the point of distinguishing two messages?

The idea behind the encryption oracle is that, in practice, the adversary may influence the behavior of the communicating parties, and have some impact on what gets encrypted (the Wikipedia article for "Chosen-plaintext" attack has some real world examples). So, the "extreme" version of influencing what parties encrypt is to decide the entire message that they encrypt, which is precisely what the encryption oracle allows! As for distinguishing between two chosen messages, this is strictly easier than finding information about a random (unknown) message that gets encrypted (which may be what we intuitively ask of an encryption scheme). So, CPA security basically takes a very strong threat model and a very weak adversarial goal. If a scheme is secure in this "overkill" setting, then it most definitely will be secure in practice, where the adversary has less power but is trying to achieve an even more complex goal.

2 Authentication

Even though CPA security is quite strong in terms of secrecy guarantees, it says nothing about integrity! That is, it says nothing about an adversaries ability to malleate messages or send messages on the sender's behalf. What we need is, basically, CPA security + authentication, which we formalize in terms of *CCA security*. This new variant of security is exactly like the CPA security game, but with one important difference: the adversary now has access to a decryption oracle. That is, in addition to being able to get encryptions for arbitrary messages, the adversary can now request the decryption of arbitrary ciphertexts. So, in a sense, the decryption oracle captures the adversary's ability to craft messages that decrypt to something useful (by, say, impersonating the sender).

2.1 Message Authentication Codes

The basic primitive that we use to get integrity (and, thus, CCA security) is called a message authentication code (MAC):

Definition 2.1. A *message authentication code (MAC)* is a triplet of PPT algorithms $\Pi = (\mathsf{Gen}, \mathsf{MAC}, \mathsf{Verify})$ such that:

- $\mathsf{Gen}(1^n)$: outputs a secret key k.
- $\mathsf{MAC}(k, m)$: outputs a tag t for the message m
- $\mathsf{Verify}(k, m, t)$: outputs 1 if t is a valid tag for m, and 0 otherwise.

Note that, generally, the MAC algorithm is deterministic, in which case the verification algorithm consists of recomputing the tag on m (using the MAC algorithm), and verifying that this matches the claimed t.

Intuitively, we want the MAC to be such that an adversary can't compute tags for new messages, i.e., the adversary cannot pretend to be the sender and cannot manipulate the messages from the sender. We encapsulate this in the security definition of a MAC:

Definition 2.2. A MAC scheme $(\mathsf{Gen}, \mathsf{MAC}, \mathsf{Verify})$ is secure against *chosen-message attacks* if any efficient adversary, with access to a tagging oracle that outputs tags for any messages of the adversaries choosing, cannot (i.e., with negligible probability) generate a valid tag for a new message (i.e., not sent to the tagging oracle).

We can, once again, think of this definition in terms of a game:

1. The adversary picks a message m and sends this to the challenger. The challenger computes $t \coloneqq \mathsf{MAC}(k, m)$ and sends this back to the adversary.
2. Repeat step 1 as many times as the adversary wants, with any messages.
3. Eventually, the adversary must output a message-tag pair (m', t') (where m was not one of the messages sent in steps 1 and 2). If $\mathsf{Verify}(k, m', t') = 1$, the adversary wins the game. Otherwise, C wins.

Like in CPA security, the intuition of this security definition is that an adversary may influence the messages that get sent (and thus, tagged), so we consider the strongest version of this threat model (an adversary that can *decide* the messages that get sent).

Two important notes: (1) a MAC provides no protection against *replay attacks*. So, a secure MAC is indeed vulnerable to this type of attack. Replay attacks are not a goal of a MAC, and we fix this in practice by using, e.g., a counter or timestamps; (2) a MAC provides no secrecy guarantees. That is, a MAC is only concerned about integrity, and says nothing about an adversaries ability to learn the underlying message. Hence, when used as part of a larger system, we generally want to use the MAC alongside an encryption scheme, to achieve both secrecy and integrity. We thus reach a simple but important theorem:

Theorem 2.1. *Combining a CPA-secure encryption scheme and MAC scheme that's secure against chosen-message attacks yields a CCA secure encryption scheme.*

Note that the order in which we combine the encryption scheme and MAC is crucial, and we must use the *encrypt-then-mac* approach: we first encrypt the message, then compute a tag (using a different key!) on this ciphertext, and send the ciphertext and the tag together. To decrypt, we first check that the tag is valid for that ciphertext (otherwise, return the "invalid" symbol $\perp$), and then we decrypt the ciphertext.

Proof. We will provide a sketch of the proof for the theorem. The high-level idea is that the secure MAC makes the decryption oracle useless, i.e., only the encryption oracle is useful, so the security of the scheme reduces to CPA security. Since the decryption oracle returns $\perp$ when the tag is invalid, the oracle is only useful if the inputs to it contain valid tags. The only way for the adversary to get valid tags is via the encryption oracle (which returns items of the form $c|t$, where t is a tag for c). By the security of the MAC, just seeing these t that are returned from the encryption oracle does not allow an adversary to generate a new tag t' for a new ciphertext, however. So, the adversary is limited to only using the decryption oracle with tags that she got from the encryption oracle (and, thus, ciphertexts that she got from the encryption

oracle)! Clearly, this is not useful to the adversary, as she already knows which messages the decryption oracle will return for these ciphertexts (since she fed the messsages to the encryption oracle to begin with). So, the decryption oracle provides no new information, and the only possible attack vector is the encryption oracle, which boils down to the CPA security of the scheme.

As a technical note: CCA security is a secrecy notion, and it doesn't say anything about integrity per se / directly. That is, in the security game, the adversay's only goal is to figure out which message got encrypted. However, we use CCA security as a proxy for integrity too, since all natural schemes that are CCA secure also provide integrity (since, e.g., they may make the decryption oracle useless). However, there are indeed CCA secure encryption schemes that are malleable! These are very contrived examples, however, and all common/natural/practical CCA secure schemes are unforgeable, so we think of CCA security as secrecy+integrity.