Winter 2018 CS 485/585 Introduction to Cryptography

Lecture 2

Portland State University *Jan. 11, 2018*
Lecturer: Fang Song

Draft note. Version: January 25, 2018. Email fang.song@pdx.edu for comments and corrections.

Agenda

- (Last time) History; Principles of modern Crypto; Defining secure private-key encryption: brainstorming
- Perfect secrecy
- One-time Pad
- Limitations

Logistics

- HW1: elec. copy on webpage; start now, and come for help
- D2L: forward d2l email to your email account. Solutions will also be posted on d2l
- Highly helpful: do the reading before class, and again (multiple times) afterwards

Perfect secrecy of private-key encryption

Syntax of private-key encryption.

We introduce the following notations to speak of a general private-key encryption scheme.

- Message/Plaintext space: $\mathcal{M}$, set of all possible messages;
- Key space: $\mathcal{K} = \{k\}$, set of all possible keys;
- Ciphertext space: $\mathcal{C} = \{c\}$, set of all possible ciphertexts;

A private-key encryption scheme $\Pi = (G, E, D)$ consists of three algorithms[1]

- G (Key-Gen Alg.): generate $k \in \mathcal{K}$, must be *randomized*. Why?
- E (Encryption Alg.): $\mathcal{K} \times \mathcal{M} \to \mathcal{C}$, randomized or deterministic.
- D (Decryption Alg.): $\mathcal{K} \times \mathcal{C} \to \mathcal{M}$. WLOG, D is deterministic.

[1] All algorithms can be (and often have to be) *randomized* (i.e., *probabilistic*). A randomized algorithm is an ordinary algorithm but can flip uniform coins at each step. You can also think of a sufficiently long string of random bits as an additional input to the algorithm other than the ordinary input. The reason for allowing randomized algorithms will become clear.

The informal security definition we arrived at from last lecture says that

> *Regardless of any information an attacker **already** has, a ciphertext should leak no **additional** information about the underlying plaintext.*

Probability theory is the right language to formalize it.

- M: random variable denoting the message, i.e., $\Pr[M = m]$ defines the a priori distribution on $\mathcal{M}$.
- K: random variable denoting the key generated by G.
- $C = E_K(M)$ the random variable denoting the resulting ciphertext of encrypting a message under a random key.

	Space	Distribution/r.v.
Plaintext	$\mathcal{M} = \{m\}$	M
Ciphertext	$\mathcal{C} = \{c\}$	C
Key	$\mathcal{K} = \{k\}$	K

Notations:

- $k \leftarrow G$, $c \leftarrow E_k(m)$: getting output of a randomized algorithm.
- $x \leftarrow X$: drawing an element x from set X uniformly at random.

Definition 1 ([KL: Def.2.3])**.** Π is perfectly secret if for every distribution over $\mathcal{M}$, and every[2] $c \in \mathcal{C}$,
[3]

$$\Pr(M = m | C = c) = \Pr(M = m)\,. \tag{1}$$

[2] we need $\Pr[C = c] > 0$ to avoid conditioning on a zero-probability event.

[3] What's the probability taken over, i.e., what random experiment are we considering, and what are the resulting sample space and distribution? Here the random experiment is pick a message according to the a prioi distribution on $\mathcal{M}$, and generate a random key according to G, and produce a ciphertext $C = E_K(M)$. Therefore the probability is taken over the random choices of both the message and the key.

This precisely formulates the security goal we wanted. But there is another component of a security definition we have not been explicit about so far. That is the *attack/threat* model, i.e., what the attacker is capable of: what resources are available, what are permitted and what are not. The threat model implicit in the above definition is that the attacker *eavesdrops* and observes one and only one ciphertext c, and tries to figure out the underlying message. [4]

Any security definition should specify precisely both *security goal* and *attack model*, if not more. To reflect, *perfect secrecy*

- Security goal: a priori knowledge = a posteriori knowldge.
- Attack model: eavesdropping attack (once only). We will encounter other attack models in the future. Simple extension: ciphertext-only attack, i.e., multiple ciphertexts; known-plaintext attack; chosen-plaintext-attack; chosen-ciphertext-attack.

[4] It is important to keep in mind that the attack model only specifies the power that is available or allowed, but does not make any assumption on how an adversary uses the power. For instance, in perfect secrecy once the adversary gets the ciphertext, we don't care how it processes it; be it exploiting a supercomputer or praying to god for an answer.

Our security goal in 1 can be interpreted in another way. Roughly, if a ciphertext reveals no information about the plaintext, isn't that mean that encrypting two messages m and m' should produce the same distribution. In other words, fixed a ciphertext c, it m and m' should be equally likely to get encrypted to c.

Definition 2. Π is perfectly secret if for every $m, m' \in \mathcal{M}$ and every $c \in \mathcal{C}$
[5]

$$\Pr(E_K(m) = c) = \Pr(E_K(m') = c)\,, \tag{2}$$

[5] What's the probability taken over? In contrast to Def. 1, (m, m', c) are all fixed. The random choice of the key is the only source of randomness.

Are the two definitions indeed equivalent? Namely is a scheme secure according to one definition *if and only if* it is secure according to the other definition?

Lemma 3. *Definition 1 is* equivalent *to Definition 2 .*

Proof. We need to show **both** directions: Def. 1 $\Rightarrow$ Def. 2 (KL-EX.2.4) and Def. 2 $\Rightarrow$ Def. 1 (KL book).

- (Def. 1 $\Rightarrow$ Def. 2) Def. 1 states that $\Pr[M = m|C = c] = \Pr[M = m]$.

  Consider any $c \in \mathcal{C}$ and $m \in \mathcal{M}$, by Bayes' theorem we have

  $$\Pr[C = c|M = m] = \Pr[M = m|C = c] \cdot \Pr[C = c]/\Pr[M = m] .$$

  Therefore, $\Pr[E_K(m)] := \Pr[C = c|M = m] = \Pr[C = c] =: \delta_c$ holds for any m and c. Therefore for any $m, m' \in \mathcal{M}$ and $c \in \mathcal{C}$, $\Pr[E_K(m) = c] = \delta_c = \Pr[E_K(m') = c]$.

□

Perfect indistiguishability These two equivalent formulations are abstract mathematical expressions. Cryptographers however usually prefer thinking *operationally*. They like actively playing a (mental) game with an adversary.

Def. 2 can be read as the distributions (over $\mathcal{C}$) resulting from encrypting one message m and encrypting another message m' are *identical*. From the attackers point of view, these two distributions are *indistinguishable*. We can reformulate this by a *experiment* or *game* between an attacker/adversary $\mathcal{A}$ and the so-called *challenger CH*. This gives another equivalent definition of PS. [6]

[6] This will also serve as the template for many of our definitions in the future.

Draw a game diagram

1. Adversary $\mathcal{A}$ outputs a pair of messages $m_0, m_1 \in \mathcal{M}$.
2. CH generates a key $k \leftarrow G$, and a uniform $b \leftarrow \{0,1\}$. Compute *challenge ciphertext* $c \leftarrow E_k(m_b)$ and give to $\mathcal{A}$.
3. $\mathcal{A}$ outputs a bit b' as the guess of b.
4. Define the output of the experiment to be 1 if $b' = b$, and 0 otherwise. Write $\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi} = 1$ if the experiment output is 1, in which case we call $\mathcal{A}$ succeeds.

Figure 1: Adversarial indistinguishability experiment $\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}$.

Definition 4 ([KL: Definition 2.5]). $\Pi = (G, E, D)$ is perfectly indistinguishable if for every attacker $\mathcal{A}$, it holds that

$$\Pr[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}] = \frac{1}{2} . \tag{3}$$

Lemma 5. *[KL: Lemma 2.6] Perfect secrecy is equivalent to perfect indistinguishability.* [7]

[7] Proof in HW.

Remark 1. why care about so many equivalent definitions? Different characterizations give a more comprehensive understanding. In different situations, some are easier to work with. Don't constrain yourself to one!

One-time Pad

We have a nice definition. But can we achieve it? A simple scheme, One-time-pad (OTP) does it. *Vernam*, patented in 1917, but existed earlier. However, the formal analysis, i.e., definition and proof of security, had to wait for about 25 years till Claude Shannon's seminal work. Shannon was a mathematician, electrical engineering and cryptographer. He was the founding person of information theory, and he was also one of the early researcher in artificial intelligence. We are studying his main contribution in cryptography.

[KL: Construction 2.8] Fix integer $\ell > 0$. $\mathcal{K} = \mathcal{M} = \mathcal{C} = \{0,1\}^\ell$.

- G: sample a uniformly random key $k \leftarrow \mathcal{K}$.
- E: on input $m \in \mathcal{M}$ and $k \in \mathcal{K}$, output $c := m \oplus k$.
- D: on input $c \in \mathcal{C}$ and $k \in \mathcal{K}$, output $m := c \oplus k$.

Figure 2: Construction of one-time-pad

[8]

[8] $\oplus$ is XOR bit-wise. OTP is correct, simple and actually got used quite a lot by national intelligence angencies in mid-20th, e.g., "red-phone link" between the White House and Kremlin during cold war.

Theorem 6 ([KL: Theorem 2.9])**.** *The one-time-pad encryption scheme is perfectly secret.*

Idea. Ciphertext is a uniformly random string regardless of the plaintext.

Proof. Use Def. 2. For any $m, m' \in \mathcal{M}$ and $c \in \mathcal{C}$.

$$\Pr[E_K(m) = c] = \Pr_{k \leftarrow \mathcal{K}}[k \oplus m = c] = \Pr_{k \leftarrow \mathcal{K}}[k = m \oplus c] = 1/2^\ell .$$

Similarly $\Pr[E_K(m') = c] = \Pr_{k \leftarrow \mathcal{K}}[k \oplus m' = c] = 1/2^\ell$. Hence $\Pr[E_K(m) = c] = \Pr[E_K(m') = c]$ for arbitrary m, m' and c. This concludes the proof. □

Crucial observations on one-time-pad.

- Key has equal length as a message. $|k| = |m|$.
- "One-time" means "one-time", seriously!

Limitations of perfect secrecy

It turns out these are not just shortcomings of a particular encryption scheme, OTP. But actually both are inherent limitations of *perfect secrecy*.

Limitation 1: Perfect secrecy cannot avoid long keys.

Theorem 7 (KL Theorem 2.10)**.** *Suppose* $\Pi = (G, E, D)$ *is perfectly secret, then* $|\mathcal{K}| \geq |\mathcal{M}|$.

Proof. Proof by contradiction. Suppose $|\mathcal{K}| < |calM|$. Consider a $c \in \mathcal{C}$, and we run the decryption algorithm on c under each $k \in \mathcal{K}$. Let

$$\mathcal{M}(c) := \{m : m = D_k(c) \text{ for some } k \in \mathcal{K}\}.$$

Clearly $\mathcal{M}(c) \leq |\mathcal{K}| < |\mathcal{M}|$. This is already saying that given c, adversary can rule out some messages, which must not be possible by perfect secrecy. More precisely, consider the uniform distribution on $\mathcal{M}$, and some $m* \notin \mathcal{M}(c)$. Then

$$\Pr[M = m * |C = c] = 0 \neq \Pr[M = m] = \frac{1}{|\mathcal{M}|}.$$

□

Attacking the system with prob. $1/2^{\ell}$ may not seem too bad, but this violation can be boosted. HW bonus problem has a strengthen of the proof here.

Limitation 2: "one-time" only. It is inherent too, but more general (applies to computational secrecy too). It has to do with the threat/attack model, i.e., secrecy against eavesdropping is not strong enough to ensure securely encrypting *multiple* messages under the same key.

Computational secrecy

Resolving limitation 1 To get away with *long* keys, have to appeal to relaxations in perfect secrecy.

Recall the perfect indist. formulation: distinguishing prob. must be exactly $1/2$ (i.e. attacker always fails completely) against *any* attacker (i.e. can be computationally unbounded) .

- consider "efficient" attackers only
- accept "small" break of scheme

  Both are necessary.
- Known-plaintext attack: exp. time, succ prob. almost 1.

- Key-guessing attack: constant-time, $1/|\mathcal{K}|$ success probability.

How to interpret the constraints? Two general approaches: **concrete** and **asymptotic**.

Concrete approach

> Template for **concrete** security
> A scheme is (t, ϵ)-*secure* if for every adversary running for time at most t succeeds in breaking the scheme with probability at most ϵ.

Example 8. No adversary running at most 2^{80} CPU cycles can break the scheme with prob. better that 2^{-60}. [9]

Important in practice, but difficult to provide and often misleading: what type of computing power, what algorithm was implemented?.

*Asymptotic approach (*We will take this!*)* [10]

Security parameter n – think of it as the key length, and everything is viewed as a function of n rather than concrete numbers: running time of all algorithms (including the adversary) and success probability.

"Efficient" = PPT, and "Small"= *negligible*.

> Template for **asympototic** security
> A scheme is *secure* if for every *probabilisitic polynomial-time* adversary $\mathcal{A}$ carrying out an formally specified attack, the probability that $\mathcal{A}$ *succeeds* in the attack is *negligible*.

More about asymptotics

Definition 9. A *probabilistic polynomial time* (PPT) algorithm is an algorithm with access to an infintely long random tape, which for all inputs $x \in \{0,1\}^*$ and random tapes halts within $p(|x|)$ steps for some polynomial p. A PPT algorithm A is said to compute $f : \{0,1\}^* \to \{0,1\}^*$ with probability q if for all inputs $x \in \{0,1\}^*$, we have $\Pr[A(x) = f(x)] \geq q$.

A *negligible* function is one that decreases asymptotically faster that any inverse polynomial function.

Definition 10. A function $f : \mathbb{N} \to \mathbb{R}$ is *negligible* if for every positive polynomial p, there is an N such that for all integers $n \geq N$ it holds that $f(n) < \frac{1}{p(n)}$. [11]

Example 11. $2^{-n}, 2^{-\sqrt{n}}, n^{-\log n}$ are all negligible.

Rule of thumb: n^5 adversary succeeds with prob. at most $2^{-0.5n}$.

[9] Modern private-key encryption often assumes **optimal** security: when the key has n bits (i.e. key space size 2^n), any adversary running for time t can break the scheme with prob. at most $ct/2^n$ for some constant c.

[10] In practice need concrete security. Asymptotic does not guarantee security for small n, but usually can be translated into concrete security.

[11] We usually denote an arbitrary negligible function by negl.