Winter 2018 CS 485/585 Introduction to Cryptography
Lecture 6

Portland State University
Lecturer: Fang Song

Jan. 25, 2018

Draft note. Version: February 4, 2018. Email fang.song@pdx.edu for comments and corrections.

Agenda

- (Last time) DES and AES, modes of operations, CPA security
- PRF-OTP, randomized counter mode,
- Message authentication

Logistics

- HW 2 typo corrected; updated PDF on webpage.
- Lec 2 - 5 notes posted

Constructing CPA-secure ciphers

Theorem 1 ([`KL`: Thm. 3.24]). *CPA-security ≡ multi-message CPA-security.* [1]

This immediately gives some advantages of CPA-security: we don't have to explicitly worry about multi-message security, since it follows from single-message CPA-security. We can also get an encrytion for messages of arbitrary length from an encryption that can only encrypt messages of a fixed length, say just one bit. $E'_k(m) := E_k(m_1)\| \ldots \|E_k(m_\ell)$ is CPA-secure.

Take-away message
1-bit encryption is complete for CPA security.

How to get CPA-secure schemes? By the simple rule of randomized encryption, we know that none of stream ciphers can be CPA-secure. ECB and (Deterministic) Counter modes are not CPA-secure either. Here we give a simple construction from any PRF with a proof of CPA-security. Again, think about our good old friend OTP and its adaption to stream cipher PRG-OTP. We know the output from a PRF would look random, can we just use its output as our pad? But what input we give to PRF? Remember we have to introduce some randomness. How about we evaluate PRF on a random input? Sounds good. Let's write down what we have:

[1] Proof idea. Intuitively, if one can tell apart $E_k(\vec{m}_0)$ and $E_k(\vec{m}_1)$, you can imagine they only differ at one entry (m_0^j, m_1^j). Then one can break (single-message) CPA using m_0^j and m_1^j, and the remaining ciphertexts can be obtained from the Enc oracle.

FS NOTE: Draw diagram of encryption alg. Let $F_k(\cdot) : \{0,1\}^n \to \{0,1\}^n$ be a PRF. Construct $\Sigma := (G, E, D)$:

- $G(1^n)$: pick uniform $k \leftarrow \{0,1\}^n$.
- $E_k(m)$: pick uniform $r \leftarrow \{0,1\}^n$ and output $c := (r, F_k(r) \oplus m)$
- $D_k(c)$: parse $c = (r, s)$ and output $m := F_k(r) \oplus s$.

Figure 1: CPA-secure encryption from PRF

Theorem 2 (KL-Thm. 3.31)**.** *Σ in Fig. 1 is CPA-secure (for message of length n).*

Idea: recall the proof of PRG-OTP. Imagine F_k being a truly random function, then it will be one-time-pad with independently random key for each message, except when the random string r^* used in generating the challenge ciphertext coincides with some r used to answer $\mathcal{A}$'s queries to $E_k(\cdot)$ (assume $\mathcal{A}$ makes $q(n)$ queries for some polynomial $q(n)$). But since r^* is chosen at random, this occurs with tiny probability $q(n)/2^n$. Therefore the only possibility that an adversary breaks CPA would be by distinguishing F from truly random, but this contradicts F being a PRF.

Proof. Consider a variant of Σ, where we substitute a truly random function for the PRF. Call it $\tilde{\Sigma} = (\tilde{G}, \tilde{E}, \tilde{D})$ [2]. Then for any adversary $\mathcal{A}$ who makes at most $q(n)$ encryption queries ($q(n)$ must be bounded by some polynomial, why?). We complete the proof in two lemmas:

[2] imaginary scheme just for the sake of proof. We cannot implement it efficiently.

- Lemma 3 says that $\mathcal{A}$ will succeed in the CPA-indist. game with the same probability whether it's Σ or $\tilde{\Sigma}$, except with negligible discrepency, assuming F is a PRF.
- Lemma 4 shows that when it is $\tilde{\Sigma}$, $\mathcal{A}$ succeeds with probability $1/2 + q/2^n$, only negligibly better than a random guess.

Combining them, we have that $\left|\Pr\left[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Sigma}(n) = 1\right]\right| \le 1/2 + \mathrm{negl}(n)$. □

Lemma 3. $\left|\Pr\left[\mathit{PrivK}^{\mathit{cpa}}_{\mathcal{A},\Sigma}(n) = 1\right] - \Pr\left[\mathit{PrivK}^{\mathit{cpa}}_{\mathcal{A},\tilde{\Sigma}}(n) = 1\right]\right| \le \mathrm{negl}(n)$.

Proof. Notation:

$$p_{\mathcal{A},\Sigma} := \Pr\left[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Sigma}(n) = 1\right], \qquad p_{\mathcal{A},\tilde{\Sigma}} \Pr\left[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\tilde{\Sigma}}(n) = 1\right]$$
$$p_{D,F_k} := \Pr[D^{F_k}(1^n) = 1 : k \leftarrow \{0,1\}^n], \qquad p_{D,f\leftarrow\mathcal{F}} := \Pr[D^f(1^n) = 1]\,.$$

This is done by a reduction. We construct a distinguisher D, and show that $|p_{\mathcal{A},\Sigma} - p_{\mathcal{A},\tilde{\Sigma}}| \le |p_{D,F_k} - p_{D,f\leftarrow\mathcal{F}}|$ which is negligible since we assume that F_k is a PRF.

FS NOTE: Draw reduction diagram.
Distinguisher D: given 1^n and oracle $\mathcal{O} : \{0,1\}^n \to \{0,1\}^n$

1. Run $\mathcal{A}(1^n)$. Whenever $\mathcal{A}$ makes encryption query m, answer as follows
   (a) Choose uniform $r \leftarrow \{0,1\}^n$
   (b) Query $\mathcal{O}(\cdot)$ and obtain response $y := \mathcal{O}(r)$.
   (c) Returen ciphertext $c = (r, y \oplus m)$.
2. When $\mathcal{A}$ outputs (m_0, m_1), pick random bit $b \leftarrow \{0,1\}$, and generate c on m_b as above.
3. Continue answering encryption queries as above till $\mathcal{A}$ outputs b'. Output 1 if $b' = b$ and 0 otherwise.

Observe that D runs in polynomial time as $\mathcal{A}$ does.

- If $\mathcal{O}$ is PRF F_k with a random key k. Then $\mathcal{A}$'s view is identical to its view in $\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Sigma}(n)$. Therefore $p_{\mathcal{A},\Sigma} = p_{D,F_k}$.
- If $\mathcal{O}$ is a random function. Then $\mathcal{A}$'s view is identical to its view in $\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\tilde{\Sigma}}(n)$. Therefore $p_{\mathcal{A},\tilde{\Sigma}} = p_{D,f\leftarrow\mathcal{F}}$.

Therefore $\left|\Pr\left[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Sigma}(n) = 1\right] - \Pr\left[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\tilde{\Sigma}}(n) = 1\right]\right| \leq \mathrm{negl}(n)$.

□

Lemma 4. $\left|\Pr\left[\mathit{PrivK}^{\mathit{cpa}}_{\mathcal{A},\tilde{\Sigma}}(n) = 1\right]\right| \leq 1/2 + q(n)/2^n$.

Proof. Skipped. See KL. □

This construction inspires another mode of operation for block ciphers: **Randomized Counter (RCTR)**

- $E_k(m)$: pick $\mathsf{IV} \leftarrow \mathcal{M}$, for $j = 1, \ldots t$,

$$c[j] := F_k(\mathsf{IV} + j - 1) \bigoplus m[j] \quad (\text{addition mod } N = 2^n)\,.$$

  $c := (\mathsf{IV}, c[1], \ldots, c[t])$.
- $D_k(c)$: $m[j] := F_k(\mathsf{IV} + j - 1) \bigoplus c[j]$.

IV is called *initial value* or *counter*.

Distinction from CTR. We pick a random *counter* as starting point, instead of a fixed sequence of inputs to derive key stream. [3] Note CTR mode is not CPA-secure (why?).

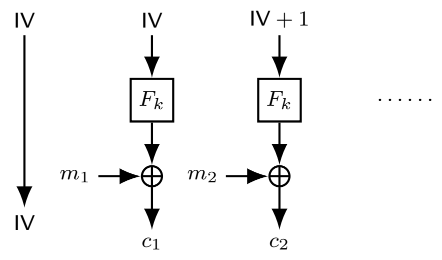

Figure 2: Randomized Counter Mode

[3] Application. A variant of AES-RCTR is used in IPsec protocol, specified in RFC 3686`https://www.rfc-editor.org/rfc/rfc3686.txt`. The main change is part of IV is randomly chosen and then fixed for all encryptions under a key. Our description assumes independent IV for each message.

Data Integrity

Secrecy vs. integrity Secrecy/confidentiality is not the only concern. Sometimes we care less about whether the message is heard by someone else, but rather, we want to make sure the message was not modified intentially by an adversary. Two examples

- Software update. You update your computer or phone applications from e.g. Apple Store from time to time. Here secrecy is irrelevant, but how do you know the patch program is indeed coming from Apple and the copy you downloaded hasn't been modified ?(In practice, this is achieved by public-key authentication, i.e., digital signatures.)
- Bank transfer. Suppose the Bank receives a money transfer request from user A to user M of amount X. Is this a valid request? Is X the right amount? (what if M is greedy and malicious?)

Rigorously precisely, there are two problems we are facing: *identity* authentication (identification) and *message* authentication. But in our setting identification is usually implied by message authentication assuming certain secret information (i.e., key) for achieving message authentication is only known to honest users and safely protected. Therefore we will focus on message authentication hereafter.

(Standard) encryption does not provide integrity. A common misconception is that since encryption "hides" the message, an adversary cannot modify the underlying message and hence data integrity is achieved. This is wrong!

- OTP or stream cipher. Suppose we encrypt one-bit by OTP as our vote (for P? 1 $\rightarrow$C and 0 $\rightarrow$ T). Then it's easy to just flip the ciphertext which will completely change the winner.
- (R)CTR: $c[j] := F_k(\mathsf{IV}+j) \oplus m[j]$ suffers from the bit-flipping attack for the same reason.
- CBC: $c[0] = \mathsf{IV}, c[1] = F_k(\mathsf{IV} \oplus m[1])$. Therefore if I just flip the ℓth bit of IV, then the decrypted message $\hat{m}[1]$ will have the ℓth bit flipped too. Read KL

Note that in these examples secrecy is not compromised since the adversary didn't gain any useful information about the underlying plaintext. Nonetheless, it is able to change what the honest user gets to see after decryption.

Message authentication codes (MAC)

The right tool cryptographers designed for protecting integrity is called *message authentication codes* (MAC).

Definition 5 (KL-Def. 4.1)**.** A message authentication code (MAC) consists of three poly-time algorithms (G, T, V):

- (Key-Gen, randomized) G: $k \leftarrow G(1^n)$ with $|k| \geq n$.
- (Tag-Gen, possibly randomized) S: takes k and message $m \in \{0,1\}^*$, generates a tag $t \leftarrow S_k(m)$.
- (Verification, deterministic) V: takes key k, a message $m \in \{0,1\}^*$ and a tag t. Output $b := V_k(m, t)$ where $b = 1$ indicates valid, and $b = 0$ indicates invalid.

Canonical verification for deterministic MAC. Suppose S is deterministic. $V_k(m, t)$ often goes as follows: compute $\tilde{t} \leftarrow S_k(m)$ and compare $\tilde{t} \overset{?}{=} t$. Such a scheme is usually called a canonical MAC.

Defining secure MAC

What do we want from a MAC?

- Security goal (i.e., what kind of "break" we want to avoid): intuitively we want that $\mathcal{A}$, without knowing the secret key, should not be able to produce a valid pair (m, t) such that $V_k(m, t) = 1$. We call it a forgery.
- Attack model: we assume the communication channel between honest users is public (such as the Internet), so $\mathcal{A}$ could intercept many valid (m_i, t_i) pairs. It's also feasible for $\mathcal{A}$ to control over the messages sometimes. Consider the bank transfer example. Suppose user M is an ebay seller, and carries out a few legit transactions with user A. The content of the messages, i.e. the prices, is under user M's control. Therefore, to be safe, we allow an adversary to request MAC tags on *any* messages of its choice. This is called an adaptive chosen-message-attack.
- Security goal **refined**: we consider it a "break" of MAC if $\mathcal{A}$ could come up with a forgery (m, t) and it never asked a tag of m before.

To make it formal, we consider the following game. Let $\Pi = (G, T, V)$ be a Mac, $\mathcal{A}$ an adversary,

FS NOTE: Draw eu-cma diagram

1. CH generates key $k \leftarrow G(1^n)$.
2. Adversary $\mathcal{A}$ is given 1^n and oracle access to $S_k(\cdot)$ (i.e., $\mathcal{A}$ can make queries m_i and obtain $t_i \leftarrow S_k(m_i)$). Let $\mathcal{L}$ be the set of all queries that $\mathcal{A}$ asked.
3. $\mathcal{A}$ outputs (m^*, t^*) in the end. The output of the game $\mathsf{Mac\text{-}forge}_{\mathcal{A},\Pi}(n) = 1$ iff. (1) $V_k(m^*, t^*) = 1$ and (2) $m^* \notin \mathcal{L}$; in this case we say that $\mathcal{A}$ wins.

Figure 3: The message authentication game $\mathsf{Mac\text{-}forge}_{\mathcal{A},\Pi}(n)$

Definition 6. MAC Π is existentially unforgeable under a chosen-message attack or eu-cma-secure, if for all PPT $\mathcal{A}$,

$$\Pr[\mathsf{Mac\text{-}forge}_{\mathcal{A},\Pi}(n) = 1] \leq \mathrm{negl}(n)\,.$$

Remarks and discussion.

- Did you wonder, since secrecy is not a concern here, is a secret key really necessary? It is!
- "Existentially unforgeable" refers to the fact that the adversary cannot forge a valid tag on *any* message, i.e., there doesn't exist a message efficient $\mathcal{A}$ can forge on.
- Is the definition "too" strong? After all, we probably only care about no-forgery on "meaningful" messages. But "meaningful" is not an absolute term and is *application dependent.* Be conservative and save further worries in applications.
- **Replay attacks.** The security definition itself does not prevent a simple attack in practice: copy a previous message-tag pair and resend it to an honest user at a later point. Again, consider the greedy ebay seller Mr. M, what if he keeps forwarding the money transfer request?

  Two common techniques for thwarting such attacks:

  1. *Sequence number (counter)*
  2. *time-stamps* $t = S_k(\mathsf{TIME}\|m)$ and verify $V_k(\mathsf{TIME}\|m, t) = 1$, and TIME is "recent".

  Both need synchronization to some extend.