Winter 2018 CS 485/585 Introduction to Cryptography

Lecture 5

Portland State University
Lecturer: Fang Song

Jan. 23, 2018

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

Agenda

- (Last time) OTP-PRG: Proof by Reduction; PRF/PRP (Block cipher)
- Block cipher practical constructions: DES and AES
- CPA security

Constructions of Block ciphers

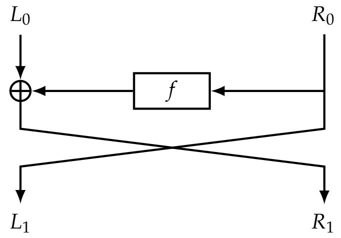
- In theory: beautiful constructions from (OWF to) PRG to PRF to PRP.
- In practice: below.

Case study: DES. The Data Encryption Standard (DES) was developed at IBM in response to a solicitation for proposals from the National Bureau of Standards (NIST National institutes of standards now). Published in 1975, and adopted for "unclassified" applications in 1977.

It consists of initial permutation (IP), 16 rounds of a simple round function (Feistel Network), and a final permutation ($FP = IP^{-1}$).

Draw DES diagram [Boneh-Shoup Fig. 4.9]

- key length: 56 bits. *Key scheduling* derives $k_i, i = 1, \dots, 16$ each of 48-bit long.
- block size: 64 bits.
- Each round: a Feistel permutation, which constructs a permutation out of a function on a smaller domain.
- Round function: 32-bit input first gets expanded to 48-bits and XORed with round key k_i. The outcome goes into a *Substitution-Permutation Network* (SPN) inspired the Shannon's *Confusion-Diffusion* paradigm. Usually called the *DES mangler function*.



- S-box: highly non-linear functions.

Draw round function with k_i, S-boxes, ... Note that everything is public (key-scheduling, S-boxes, IP/FP, ...) except the key.

Case study: AES Due to the short key of DES, it does not provide enough security even though no vulnerability is found. Beginning in 1997 and completed in 2000, NIST picked Rijndael designed by Belgian cryptogrpahers (Joan Daemen and Vincent Rijmen) as the AES (Advanced Encryption Standard) to substitute DES.

cipher name	key-size (bits)	block size (bits)	number of rounds
AES-128	128	128	10
AES-192	192	128	12
AES-256	256	128	14

Table 1: AES family

Draw diagram

Block-cipher mode of operations

As we will see, practical block ciphers work on a small data block (e.g. 128 bits). How to encipher long messages of multiple blocks? This is called modes of operation of block ciphers.

> Block cipher modes of operation
> Assume PRP/PRF $\{F_k(\cdot) : \mathcal{M} \to \mathcal{C}\}$ with key space $\mathcal{K}$ (e.g., $\mathcal{M} = \mathcal{C} = \mathcal{K} = \{0,1\}^n, n = 128$), how to encrypt messages in $\mathcal{M}^t$?

Notation: for $m \in \mathcal{M}^t$, usually divide it into blocks $m[i] \in \mathcal{M}, i = 1, \ldots, t$.

Draw ECB diagram

1. **Electronic Code Book (ECB) mode**. What's the simplest idea you may think of?

   Unfortunately, ECB is not secure. Consider $m = m[1] \| m[1]$ and $m' = m[1] \| m[2]$ with $m[1] \neq m[2]$. It is trivial to distinguish the ciphertexts corresponding to m and m'.

2. **Deterministic counter mode (CTR)** Recall construction of a stream cipher from a block cipher.

$$E_k(m) := (F_k(1) \oplus m[1], \ldots, F_k(t) \oplus m[t]) .$$
$$D_k(c) := (F_k(1) \oplus c[1], \ldots, F_k(t) \oplus c[t]) .$$

   Note that Counter mode works with PRF as well, since decryption does not need F_k^{-1}.

3. **Randomized Counter (RCTR)**

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

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

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

   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. RCTR actually achieves a stronger security notion we will see soon (IND-CPA). [2]

4. **Cipher-Block-Chaining CBC** Reading and HW. Need inverse permutation, inherently sequential.

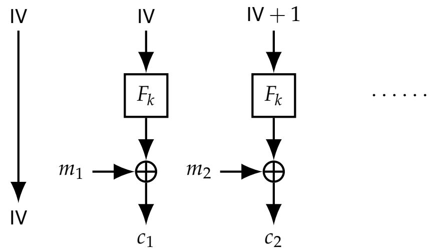


Figure 1: Randomized Counter Mode

Encryption against Chosen-plaintext-attacks

Computational secrecy successfully resolves one limitation of perfect secrecy: the need of long keys. We can now encrypt long messages using short keys (e.g., using PRG-OTP), as far as only efficient (PPT) adversaries with negligible advantage of breaking a scheme are concerned. The other limitation "one-time" hasn't been addressed yet. We would like to be able encrypt multiple messages under the same key. Intuitively, observing the ciphertexts of multiple plaintexts should not give further information about any of these underlying plaintexts. Clearly, the schemes we've seen are not secure in this sense. [3]

The fundamental problem here is that the encryption algorithms are *deterministic*, therefore if you see two identical ciphertexts, you will know for sure the underlying plaintexts are *identical*. Namely, randomized encryption is necessary for multi-message secrecy. This can be made formal.

Proposition 1 (KL-Prop. 3.20, Thm. 3.21. Necessity of random encryption for multi-message secrecy)**.** *There is a private-key encryption scheme that is computationally secret (in the presence of an eavesdropper) but is* ***NOT*** *multi-message secret. In fact, any scheme that has a* deterministic *encryption algorithm cannot be multi-message secret.*

Instead of formally defining and constructing *multi-message secrecy* (Cf. [KL: Def. 3.19]), we will study a stronger security definition that automatically implies multi-message security.

[1] identify with integers in $\{0, 1, \dots, 2^n - 1\}$

[2] 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.

[3] Example: OTP and PRG-OTP

Defining CPA security

Both perfect secrecy and computational secrecy consider rather mild attackers-they can only eavesdrop (one ciphertext). We consider a stronger type of attacks: *chosen-plaintext attacks* (CPA).

Here we assume an adversary has control, at least partially, over what messages the users encrypt. Consider two honest parties, sharing a key and applies some encryption scheme to protect their communication. We assume that an adversary could somehow issue a bunch of messages m_i of its choice and then get the corresponding ciphertexts c_i by listening into the channel. We hope that it remains difficult for the adversary to decipher any other ciphertext, despite the additional capability.

Draw communication channel and CPAttacks

Is CPA feasible? You'd wonder, are we too nice to the adversary by giving it too much power? Well, it is actually realistic or even common. The most popular example of CPA you will often hear about is probably the stories during WW II. For instance, The British placed mines at certain locations and (intentionally) managed to let the Germans discover them. Sounds dumb? Don't judge too quickly. They knew that the Germans would encrypt the locations and send back to the headquarters. These ciphertexts were exploited at Bletchley Park to break the Enigma machine. A classic CPA indeed!

Let's look at a modern example. We use Google everyday, and your search traffic is all encrypted. This includes in particular one important object on a search page! The one that Google makes great revenue from – *ads*.[4] Imagine a wealthy attacker with some fund at hand, it could place any ads on Google and obtain the ciphertexts from Google.

You should be convinced that CPA both powerful and realistic. Security against CPA is considred the de facto requirement for modern cryptosystems. Without further delay, let's discuss formally secure encryption against CPA.

Recal two components of a security definiton. We'd use the template of indist. game to specify our security goal. How do we model CPA more formally, i.e., the ability of getting encryptions of messages chosen by the adversary? We've seen a similar scenario: hand the encryption algorithm with a secret key $E_k(\cdot)$ as an oracle to the adversary. Let $\Pi = (G, E, D)$ be an encryption scheme.

FS NOTE: Draw cpa diagram

Definition 2 ([KL: Def. 3.22])**.** Π is *CPA-secure* or has *indistinguishable encryptions under a chosen-plaintext attack*, if for any PPT adversary $\mathcal{A}$ it holds that

$$\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1] \leq 1/2 + \mathrm{negl}(n)\,.$$

[4] used to be yellowish shadow; currently a small mark AD, hard to identify

1. CH generates key $k \leftarrow G(1^n)$.
2. Adversary $\mathcal{A}$ is given 1^n and oracle access to $E_k(\cdot)$ (i.e., $\mathcal{A}$ can make queries m_i and obtain $E_k(m_i)$).
3. $\mathcal{A}$ outputs a pair of messages $m_0, m_1 \in \mathcal{M}$ with $|m_0| = |m_1|$.
4. CH picks uniform $b \leftarrow \{0,1\}$, and computes *challenge ciphertext* $c \leftarrow E_k(m_b)$. c is given to $\mathcal{A}$.
5. $\mathcal{A}$ continues to access $E_k(\cdot)$ and outputs a bit b' in the end.
6. Define the output of the game

$$\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1, \quad \text{iff.} \quad b' = b\,,$$

Figure 2: CPA indistinguishability game $\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n)$

Remark 1. $\mathcal{A}$ can make queries adaptively, i.e., ask new queries based on what it has seen so far. After receiving the challenge cipher, it still has access on $E_k(\cdot)$. So the queries can depend on c and any prior knowledge. Now you should see there is no way that a deterministic encryption can achieve CPA-security!

We introduce CPA-security for the purpose of encrypting multiple messages. You could generalize the CPA-security definition to multiple messages, which is left to reading. As we promised, CPA security is robust and multi-message security comes at no extra cost.

Let's be more precise about what we want. We change the CPA-indistinguishability game and define multi-message CPA indist. game $\mathsf{PrivK}^{\mathsf{mult\text{-}cpa}}_{\mathcal{A},\Pi}(n)$:

- $\mathcal{A}$ generates two tuples of messages $\vec{m}_0 = (m_0^1, \ldots, m_0^t)$ and $\vec{m}_1 = (m_1^1, \ldots, m_1^t)$ for some $t \geq 1$. We require that $|m_0^j| = |m_1^j|$ for all j.
- CH picks random $b \leftarrow \{0,1\}$, and computes $\vec{c} := (E_k(m_b^1), \ldots, E_k(m_b^t))$.

Similarly, we define the experiment output $\mathsf{PrivK}^{\mathsf{mult\text{-}cpa}}_{\mathcal{A},\Pi}(n) = 1$ if $b' = b$, and $\mathsf{PrivK}^{\mathsf{mult\text{-}cpa}}_{\mathcal{A},\Pi}(n) = 0$ otherwise.

Definition 3 ([KL: Hybrid of Def. 3.19 & 3.23])**.** Π is *multi-message CPA-secure* or has *indistinguishable multiple encryptions under a chosen-plaintext attack*, if for any PPT adversary $\mathcal{A}$

$$\Pr[\mathsf{PrivK}^{\mathsf{mult\text{-}cpa}}_{\mathcal{A},\Pi}(n) = 1] \leq 1/2 + \mathrm{negl}(n)\,.$$

Theorem 4 ([KL: Thm. 3.24])**.** *CPA-security* $\equiv$ *multi-message CPA-security.* [5]

[5] 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.

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.

Constructing CPA-secure ciphers

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 DCTR modes are not CPA-secure either. RCTR and CBC are indeed CPA secure. 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:

> **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 3: CPA-secure encryption from PRF

Now if you stare at the diagram, I hope you can recognize it! It is nothing but the first iteration of Randomzed Counter Mode. Let's prove that it is CPA-secure, which will warm you up to read/prove RCTR as well as CBC modes.

Theorem 5 (KL-Thm. 3.31)**.** *Σ in Fig. 3 is CPA-secure (for message of length n).*

Idea: 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.