Applied Cryptography Spring Semester 2023 Lectures 4, 5 and 6

Kenny Paterson (@kennyog) , Felix Günther

Applied Cryptography Group

https://appliedcrypto.ethz.ch/

Overview of Lectures 4, 5 and 6

- Motivation for modes of operation
- ECB mode
- CBC mode
- CTR mode
- Other modes
- IND-CPA security for symmetric encryption
- The PRP-PRF switching lemma
- (Proof of security for CTR mode)

Motivation for modes of operation

Motivation for modes of operation

- A block cipher encrypts a message of exactly *n* bits.
    - What if the message is not a multiple of *n* bits?
    - What if the message is not of a fixed length but actually a TCP-like stream?

- Modes of operation provide different ways of using a block cipher to encrypt flexible amounts of data.
    - Different performance characteristics.
    - Different error-propagation properties.
    - Different suitability for different applications .

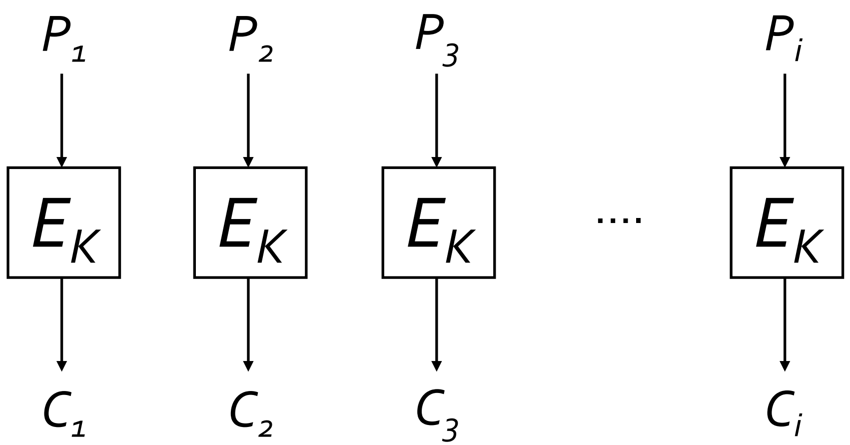
Main modes of operation

- NIST SP 800-38A specifies 5 modes
  - ECB - Electronic Code Book.
  - CBC - Cipher Block Chaining.
  - CFB - Cipher Feedback.
  - OFB - Output Feedback
  - CTR – Counter Mode.
- NIST SP 800-38C and NIST SP 800-38D
  - Authenticated Encryption modes: AES-CCM and AES-GCM.
- Most common modes now in use: CBC mode, CTR mode, AES-GCM.
- Details of all modes via: https://csrc.nist.gov/projects/block-cipher-techniques/bcm/current-modes

ECB mode

Electronic Code Book (ECB)

- ECB is the simplest way to use a block cipher to encrypt longer messages.

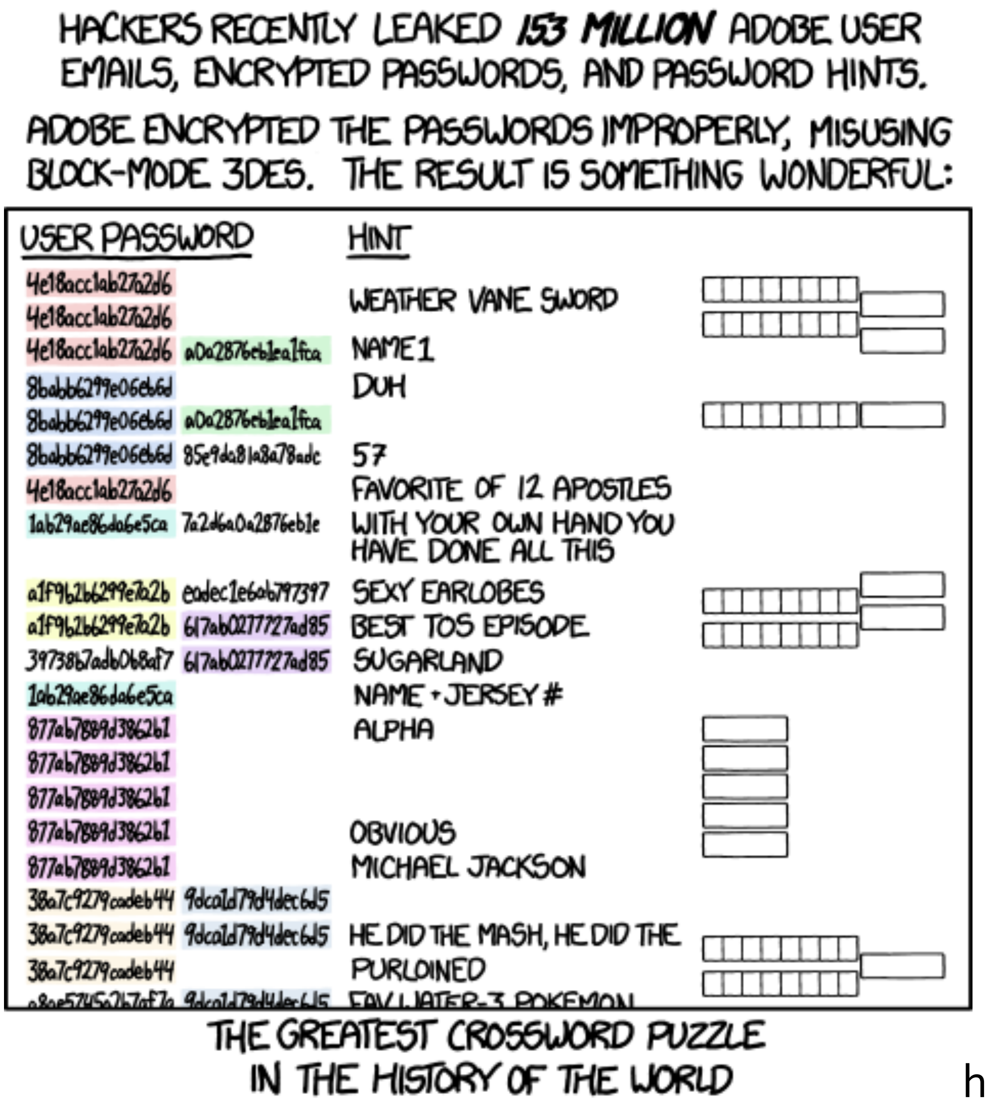


- Split message into blocks, may need to pad last block.
- Encryption can be parallelized.
- Any error in a ciphertext block affects the decryption of a single block.

ECB information leakage

- For a fixed key *K*, a given block of plaintext is always encrypted in the same way to produce the same ciphertext block.
- Encryption is *deterministic.*
- Leads to serious information leakage in many applications.
- ECB mode is *very rarely* the correct mode to use.
- Exceptions exists: e.g., searchable encryption for high entropy plaintext spaces.

ECB information leakage



https://xkcd.com/1286/

ECB information leakage

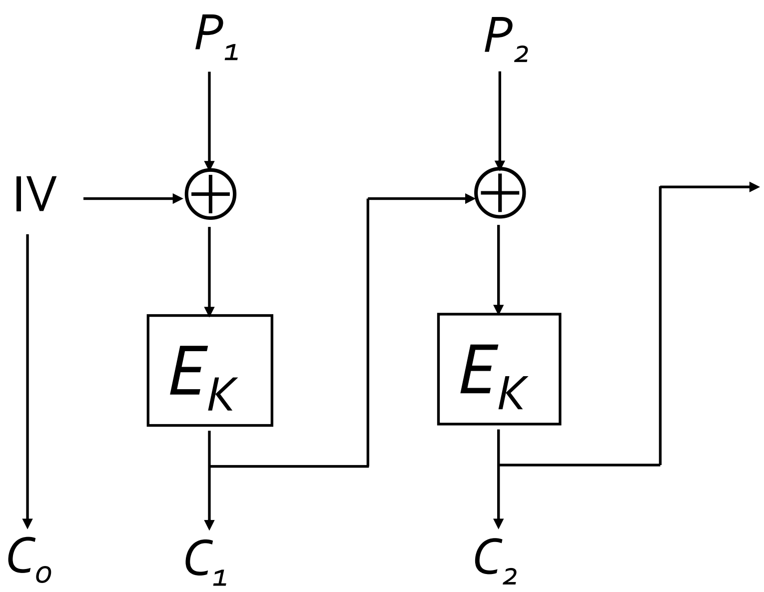
Tux the Penguin, the Linux mascot. Created in 1996 by Larry Ewing with The GIMP. lewing@isc.tamu.edu

ECB-Tux

CBC mode

Cipher Block Chaining (CBC) mode

- Randomized mode of encryption, aims to hinder information leakage of ECB mode.
- Uses previous ciphertext block (or IV for first block) to randomise the input to the block cipher at each application.
- Encryption not easily parallelisable.



Encryption equation:

$C_0 = IV$

$C_i = E_K(P_i \oplus C_{i-1})$

Decryption equation:

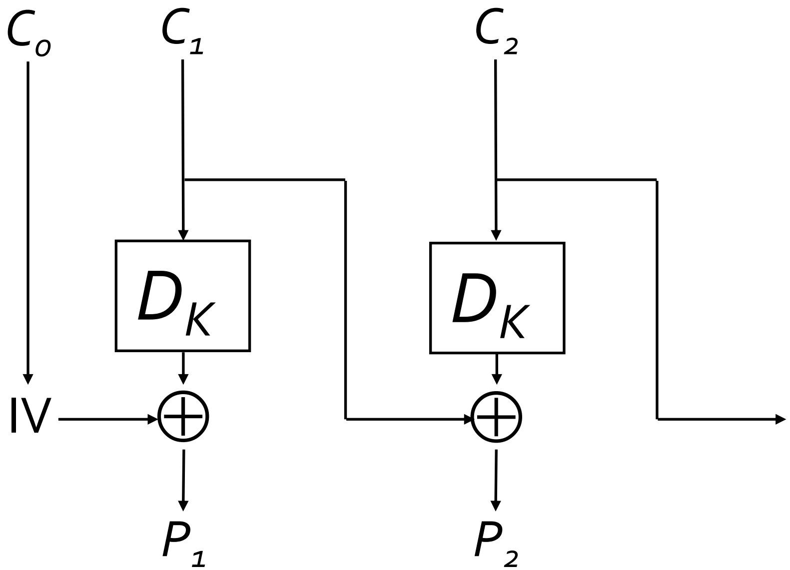
So:

$$D_K(C_i) = P_i \oplus C_{i-1}$$

and hence:

$$P_i = D_K(C_i) \oplus C_{i-1}$$

CBC mode decryption



Encryption equation:

$C_0 = \text{IV}$

$C_i = E_K(P_i \oplus C_{i-1})$

Decryption equation:

So:

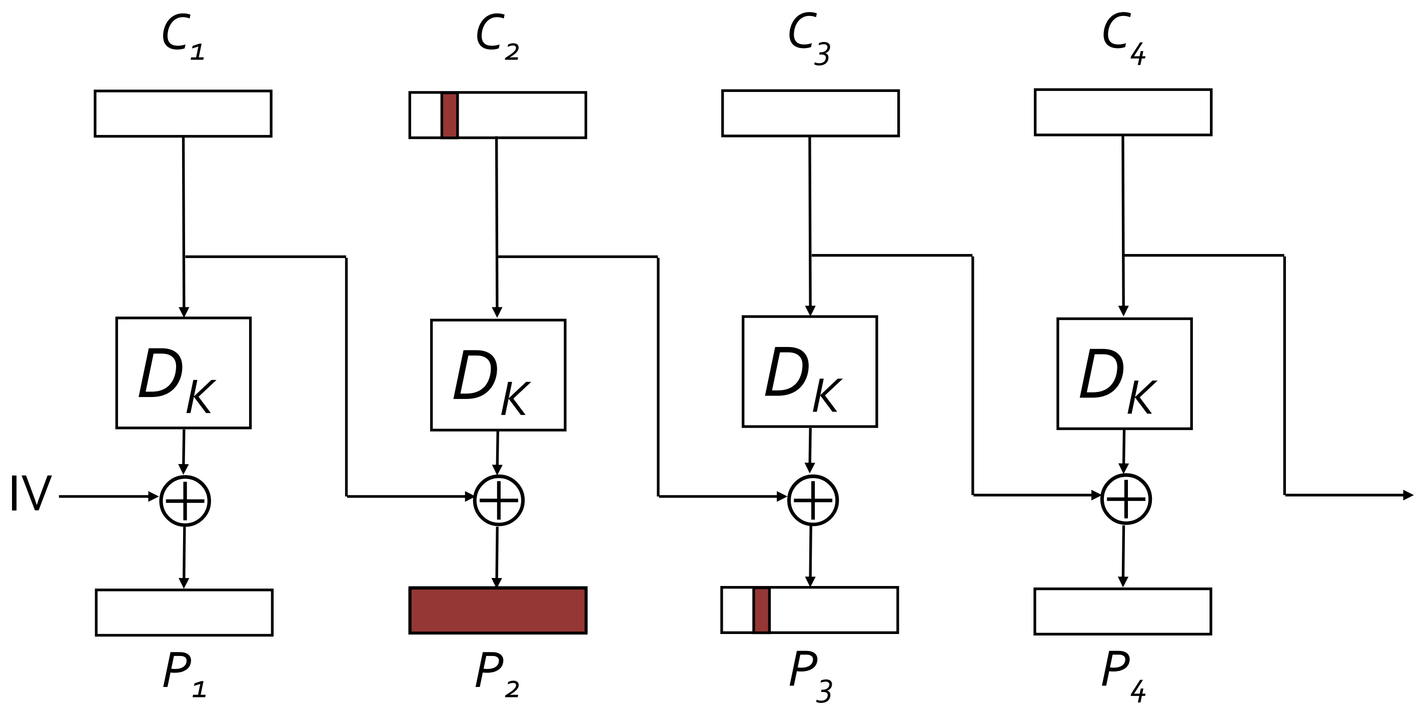
$$D_K(C_i) = P_i \oplus C_{i-1}$$

and hence:

$$P_i = D_K(C_i) \oplus C_{i-1}$$

Error propagation in CBC

- Suppose an error arises in ciphertext block C_2: $C_2 \rightarrow C_2 \oplus \Delta$.



- Then the error propagates to P_3, and P_2 becomes garbage.
- More formally: if we replace C_i with $C_i \oplus \Delta$, then after decryption, P_{i+1} is replaced by $P_{i+1} \oplus \Delta$, and P_i gets randomized.

Further remarks on CBC mode

- IV is needed for decryption so must be included as part of ciphertext.
  - Often written as C_0 to emphasize this.
  - Leading to ciphertext expansion: ciphertext will be at least one block larger than plaintext.
- IV should be uniformly random for each message encrypted (requires a good source of randomness).
  - Use of non-random IVs leads to practical attacks.
- CBC mode processes complete blocks so padding is needed to make plaintext bit-length a multiple of the block size n.
  - Improper handling of padding during decryption leads to practical attacks.

Further remarks on CBC mode

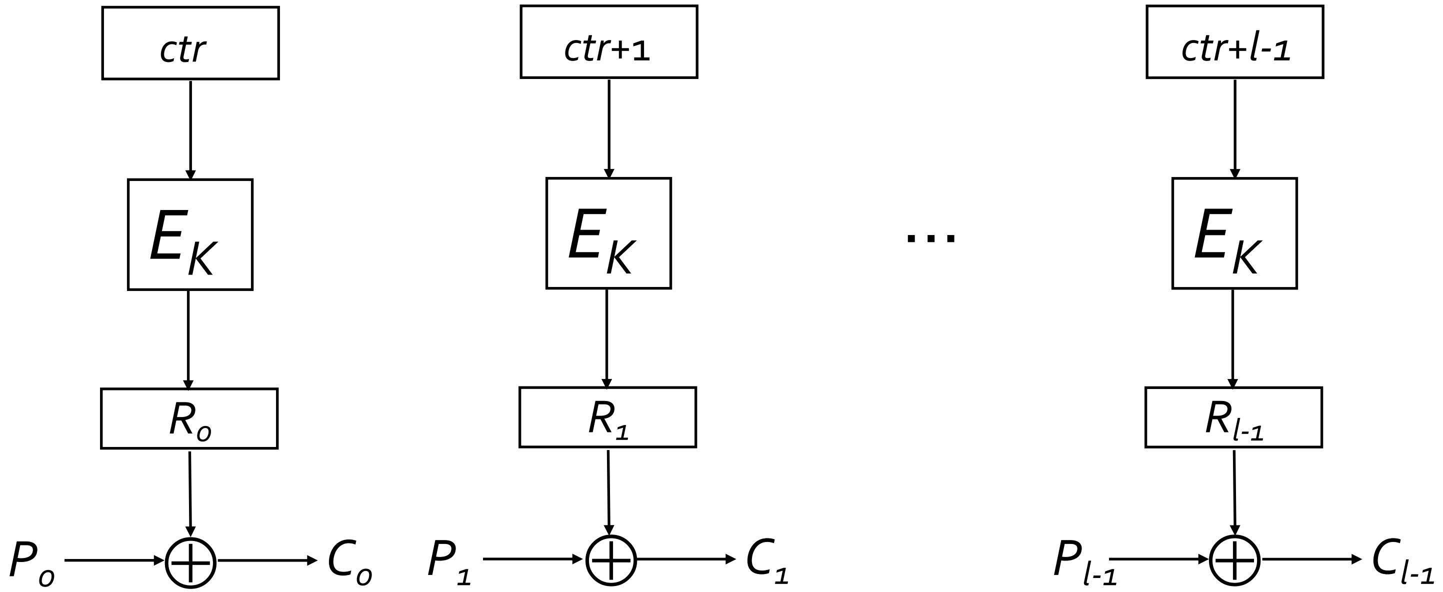
- CBC mode is problematic if block size (n) is too small.
  - Many encryptions under same key leads to ciphertext block collisions.
  - These start to occur after approx. $2^{n/2}$ blocks are encrypted under one key, by a birthday bound analysis.
  - Recall: $C_i = E_K(P_i \oplus C_{i-1})$
  - Hence, if $C_i = C_j$, then $P_i \oplus C_{i-1} = P_j \oplus C_{j-1}$, so

$$P_i = P_j \oplus C_{j-1} \oplus C_{i-1}$$

  - For example, if P_j is known, now P_i can be recovered: a partially known plaintext attack, realistic for lots of network protocols, e.g. HTTP.
  - Sweet32 attack applies this idea to DES/3DES in TLS, requiring encryption of about 2^{34} blocks under the same key.
  - More at: https://sweet32.info/

CTR mode

Counter (CTR) mode encryption



- An incrementing counter is used to generate pseudo-random blocks of output (think of bit-string *ctr* as being an integer, do the addition mod 2^n).
- Parallelisable, can also pre-compute encryption masks R_i before plaintext is known.

Counter (CTR) mode encryption

- CTR is effectively a stream cipher mode.
  - It turns a block cipher into a stream cipher.
  - Encryption is just XORing plaintext *P* with keystream obtained from output of block cipher operating on an incrementing counter.
  - Block ciphers are usually slower than dedicated stream cipher designs in general, so there is still a place for stream ciphers in applications, e.g. A5, Snow in mobile telecoms.
- Error propagation: a bit-flip in the ciphertext leads to a bit-flip in the plaintext.
  - More generally, XOR of a mask Δ with the ciphertext leads to the same mask Δ being XORed onto the plaintext.
  - So CTR mode does not provide any integrity, as with any other stream cipher.
  - Did CBC mode provide integrity?

Counter (CTR) mode encryption: Practicalities

- No padding is needed since the last block of output (R_{l-1}) can be truncated to the exact length of P_{l-1}.
- Usually transmit *ctr* as part of the ciphertext:

$$C = ctr, C_0, C_1, \ldots, C_{l-1}.$$

- We can omit *ctr* if the decryption process can recover it by other means, e.g. due to synchronization.
- Encryption uses E_K, so does decryption.
  - Hence no need to implement D_K.
  - In fact no need for E_K to even be invertible!
  - Technically, this means that CTR mode can be implemented using a pseudorandom *function* rather than a pseudorandom *permutation*.

Counter (CTR) mode encryption: Selecting counters

- Key security requirement: for a fixed key K, all counter values used (across all encryptions) must be distinct.
  - If a counter is repeated, then XOR of ciphertext blocks yields XOR of plaintext blocks.
  - Similar to reuse of one-time pad/keystream repeat issue for a stream cipher (recall from exercise sheet 1).

Counter (CTR) mode encryption: Selecting counters

- Can achieve distinctness by using one of the following methods:

1. Start with $ctr = 0$ and change key for each plaintext (often impractical, key derivation can be expensive).

2. Start with a fresh, random value for *ctr* for each plaintext (requires a good source of randomness, need to limit key use to prevent colliding counters arising by chance).

3. Keep track of the last value of *ctr* used, start from $ctr+1$ in next plaintext (requires maintenance of state).

4. Construct *ctr* by concatenating a fixed size *per plaintext nonce N* supplied by the calling application and an internal counter (starting from zero for each new plaintext), i.e.:

$$ctr = N \,||\, 0\ldots00, \quad ctr+1 = N \,||\, 0\ldots.01, \quad ctr+2 = N \,||\, 0\ldots.10, \ldots.$$

(requires per plaintext nonce to not repeat, hence some kind of state needed in application; also requires limit on plaintext size to prevent internal counter wrap).

This approach is used in GCM (see later, in AE lectures).

Counter (CTR) mode encryption: Security

- Security of CTR mode relies on pseudo-randomness of block cipher.
  - CTR mode was initially viewed with suspicion by practitioners as demanding "too much" security from the block cipher, due to "closely related inputs", i.e. counter values.
  - But pseudo-randomness is now generally accepted as a reasonable design target for a block cipher.
- Intuition for security analysis (to be formalized):
  - If the block cipher is a PRP, then we can replace its output with output from random permutation Π (no efficient adversary can tell the difference).
  - We can then replace the outputs from random permutation $\boldsymbol{\Pi}$ with outputs from a random **function** *F.*
  - Only difference here is that π has no repeating outputs, but *F* may; this will only become apparent after many calls to Π/ *F.*
  - Finally, replace outputs of *F* with truly random strings – this is fine because outputs of *F* are independent random strings, assuming all of the *ctr* inputs to *F* are distinct.
  - But now we just have one-time pad encryption!

Other modes

Other modes

- CFB turns a block cipher into a stream cipher.
  - Not as efficient as a dedicated stream cipher.
  - CFB gives a *self-synchronising* stream cipher.
- OFB also turns a block cipher into a stream cipher.
  - OFB does not automatically resynchronise.
  - External markers are needed for synchronisation.
- These modes sometimes pop up in applications, but you're unlikely to see or need them.
- We'll see *authenticated* modes (GCM) later.
- Yet more esoteric modes exist, e.g. IGE in Telegram.

IND-CPA security for symmetric encryption

Formalising Symmetric Encryption

A symmetric encryption scheme consists of a triple of efficient algorithms: SE = (KGen,Enc,Dec), with the following characteristics:

KGen: key generation, typically selects a key K uniformly at random from a set $\mathcal{K}$; we assume $\mathcal{K} = \{0,1\}^k$.

Enc: encryption, takes as input key K, plaintext $m \in \mathcal{M} \subseteq \{0, 1\}^*$ and produces output $c \in C \subseteq \{0, 1\}^*$.

Dec: decryption, takes as input key K, ciphertext $c \in C \subseteq \{0, 1\}^*$ and produces output $m \in \mathcal{M}$ or an error, denoted $\perp$.

Correctness: we require that for all keys K, and for all plaintexts m,

$$\mathrm{Dec}_K(\mathrm{Enc}_K(m)) = m.$$

Definitional notes

- $\mathcal{K}$ is called the key space.
- $\mathcal{M}$ is called the message space.
- C is called the ciphertext space.
- Enc is typically randomised (cf. CBC mode, CTR mode); later Enc will have additional inputs allowing it to be made deterministic.
- Dec is assumed to be deterministic.
- In reality, there will be a maximum plaintext length L that can be encrypted by a given scheme, we might then set $\mathcal{M}$ to $\{0, 1\}^{\leq L}$.
- Ciphertexts are typically (slightly) longer than plaintexts; we would like to minimize the expansion.

IND-CPA security for Symmetric Encryption: Intuition

- Recall the notion of perfect security: the ciphertext leaks nothing about the plaintext except what was previously known about it from the plaintext distribution.
- A computational version of this: we cannot efficiently compute anything useful about plaintexts from ciphertexts.
- This can be formalised via a simulation-based security notion called *semantic security*:

> *For any efficient adversary A given access to encryptions of plaintexts of its choice, whatever A can output can also be output by a simulator S that has access only to the code of A and the lengths of the ciphertexts (but not to the ciphertexts).*

- We will use an equivalent but easier to work with game-based definition, IND-CPA security:

> *For any efficient adversary A, given the encryption of one of two equal-length messages of its choice, A is unable to distinguish which one of the two messages was encrypted.*

- NB: Boneh-Shoup refers to both notions as *semantic security.*

IND-CPA security for Symmetric Encryption

Adversary

Challenger

$b \leftarrow_{\$} \{0,1\}$

$K \leftarrow_{\$} \text{KGen}$

<u>LoR oracle</u>

$c \leftarrow \text{Enc}_K(m_b)$

(m_0, m_1)

c

b'

$$\text{Adv}_{\text{SE}}^{\text{IND-CPA}}(A) := 2\left|\Pr[b'=b] - ½\right|.$$

IND-CPA security for Symmetric Encryption

- Challenger selects a key K by running KGen, and a random bit b.
- The adversary is given repeated access *to Left-or-Right (LoR) encryption oracle*.
- In each query, the adversary A submits pairs of equal length plaintexts (m_0, m_1) to the challenger.
    - A can set $m_0 = m_1$, so it gets an encryption oracle "for free".
- The adversary A gets back $c \leftarrow \mathrm{Enc}_K(m_b)$.
- After all queries are made, A outputs its estimate b' for bit b.
- A is successful if $b' = b$.

IND = Indistinguishability

- = under

CPA = Chosen Plaintext Attack

Formal definition of IND-CPA security

Game $\mathbf{IND\text{-}CPA}(\mathcal{A}, \mathrm{SE})$

$b \leftarrow_{\$} \{0,1\}$

$\mathsf{K} \leftarrow_{\$} \mathrm{KGen}$

$b' \leftarrow_{\$} \mathcal{A}^{\mathrm{LoR}(\cdot,\cdot)}()$

Return $(b' = b)$

Oracle $\mathrm{LoR}(m_0, m_1)$:

$c \leftarrow_{\$} \mathrm{Enc}_{\mathsf{K}}(m_b)$

Return c

Figure 1: Game formalizing IND-CPA security of symmetric encryption scheme SE $=$ (KGen, Enc, Dec) with key-space $\{0,1\}^k$. By $\mathcal{A}^{\mathrm{LoR}}(\cdot,\cdot)$ we mean that adversary $\mathcal{A}$ can query the oracle LoR on (multiple) pairs of inputs of its choice. Each pair of inputs must consist of a pair of equal-length messages from the scheme's message space.

We define the advantage of an adversary $\mathcal{A}$ in this game as follows:

$$\mathbf{Adv}_{\mathrm{SE}}^{\mathrm{IND\text{-}CPA}}(\mathcal{A}) = 2 \cdot \left| \Pr\left[\,\text{Game } \mathbf{IND\text{-}CPA}(\mathcal{A}, \mathrm{SE}) \Rightarrow \mathsf{true}\,\right] - \frac{1}{2} \right|.$$

IND-CPA security for Symmetric Encryption

Definition

A symmetric encryption scheme SE is said to be (t, q, ε)-secure if, for all adversaries A running in time at most t and making at most q encryption queries, the advantage $\mathrm{Adv}_{\mathrm{SE}}^{\mathrm{IND\text{-}CPA}}(A)$ is at most ε.

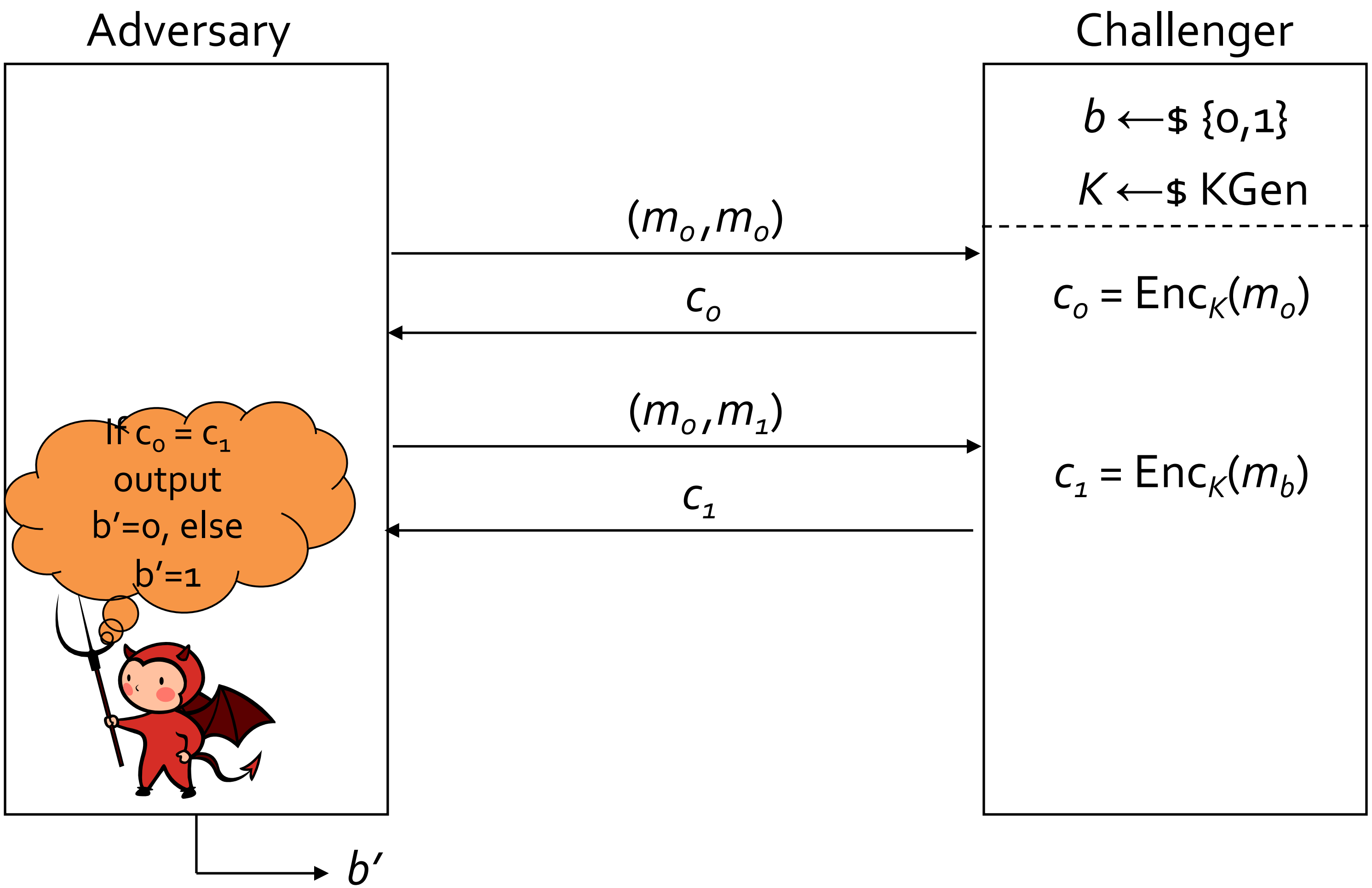
IND-CPA security for Symmetric Encryption: Properties

- IND-CPA security captures message recovery attacks: any attacker that can recover *m* from *c* can be converted into an attacker that break IND-CPA security.
- IND-CPA security captures key recovery attacks: any attacker that, given some pairs (*m*,*c*) can recover the key *K*, can be converted into an attacker that breaks IND-CPA security.
- IND-CPA security ensures that every bit of the plaintext is hidden.
- etc. …
- It can be proved that schemes like CBC mode and CTR mode meet the IND-CPA security definition *if used properly and if they are built using a good block cipher.*
  - We will prove this for CTR mode next.

IND-CPA security for Symmetric Encryption: Limitations

- The IND-CPA definition says nothing about security when the messages in a pair do *not* have equal length.
  - Ciphertext length often leaks information about plaintext length – cf. CTR mode.
  - This leakage can be fatal in real-world applications.
  - Addressed if desired by various forms of padding, traffic shaping, etc.
- The definition says nothing about integrity – and we already know that modes like CBC and CTR do not offer it.
- The definition does not give the adversary access to any kind of decryption capability: no chosen ciphertext attacks are modelled.
- The definition says nothing about attacks based on side-channel leakage, implementation vulnerabilities,…

Deterministic schemes are not IND-CPA secure



Adversary wins!

The PRP-PRF Switching Lemma

Recall: Pictorial definition of PRP security

Setup:
$b \leftarrow_\$ \{0,1\}$
$K \leftarrow_\$ \{0,1\}^k$
$\Pi \leftarrow_\$ \mathrm{Perms}[\{0,1\}^n]$

(recall: E_K: $\{0,1\}^n \longrightarrow \{0,1\}^n$)

Adversary A

x_1

y_1

x_2

y_2

b'

$b = 0$:

$y_i = E_K(x_i)$

$b = 1$:

$y_i = \Pi(x_i)$

$\mathrm{Adv}_E^{PRP}(A) := 2|\Pr[b'=b] - ½|$.

Pictorial definition of PRF security

<u>Setup:</u>

$b \leftarrow_\$ \{0,1\}$

$K \leftarrow_\$ \{0,1\}^k$

$f \leftarrow_\$ \text{Funcs}[\{0,1\}^n, \{0,1\}^n]$

(recall: $E_K: \{0,1\}^n \longrightarrow \{0,1\}^n$)

Adversary A

x_1

y_1

x_2

Upon receiving the i-th query x_i from A do:

If $x_i = x_j$ for some $j < i$

then $y_i \leftarrow y_j$

else

$y_i \leftarrow_\$ \{0,1\}^n$

Return y_i.

b

|.

<u>$b = 0$:</u>

$y_i = E_K(x_i)$

<u>$b = 1$:</u>

$y_i = f(x_i)$

PRF security and the PRP-PRF switching lemma

- Difference in definitions: sampling from random function f instead of random permutation Π.
- We have just defined PRF security for block ciphers:

$$E: \{0,1\}^k \times \{0,1\}^n \longrightarrow \{0,1\}^n$$

- But it can more generally be defined for **any** keyed function:

$$F: \{0,1\}^k \times \{0,1\}^l \longrightarrow \{0,1\}^L.$$

The PRP-PRF switching lemma essentially says that any PRP is also a PRF:

Lemma

Let E be a block cipher. Then for any adversary A making q queries:

$$|\mathrm{Adv}_E^{PRP}(A) - \mathrm{Adv}_E^{PRF}(A)| \leq q^2/2^{n+1}.$$

The Advantage Rewriting Lemma

Let b be a uniformly random bit; let b' be the output of some algorithm. Then

$$2\,|\Pr[b'=b] - ½\,| \;=\; |\Pr[b'=1 \mid b=1] - \Pr[b'=1 \mid b=0]\,|.$$

Proof:

$$\Pr[b'=b] - ½ \;=\; \Pr[b'=b \mid b=1]\cdot\Pr[b=1] + \Pr[b'=b \mid b=0]\cdot\Pr[b=0] - ½$$

$$= \Pr[b'=1 \mid b=1]\cdot½ + \Pr[b'=0 \mid b=0]\cdot½ - ½$$

$$= ½(\,\Pr[b'=1 \mid b=1] + \Pr[b'=0 \mid b=0] - 1\,)$$

$$= ½(\,\Pr[b'=1 \mid b=1] - (1 - \Pr[b'=0 \mid b=0])\,)$$

$$= ½(\,\Pr[b'=1 \mid b=1] - \Pr[b'=1 \mid b=0]\,)$$

NB1: This kind of rewriting can be done for any distinguishing-style game and advantage expression.

NB2: By a similar argument, we can also show:

$$2\,|\Pr[b'=b] - ½\,| \;=\; |\Pr[b'=0 \mid b=0] - \Pr[b'=0 \mid b=1]\,|.$$

The Difference Lemma

Let Z, W_1, W_2 be (any) events defined over some probability space. Suppose that:

$$\Pr[W_1 \wedge \neg Z] = \Pr[W_2 \wedge \neg Z].$$

(In typical uses, we have: $(W_1 \wedge \neg Z)$ occurs **if and only if** $(W_2 \wedge \neg Z)$ occurs.)

Then we have:

$$|\Pr[W_2] - \Pr[W_1]| \leq \Pr[Z].$$

Proof:

$$\begin{aligned} |\Pr[W_2] - \Pr[W_1]| &= |\Pr[W_2 \wedge Z] + \Pr[W_2 \wedge \neg Z] - \Pr[W_1 \wedge Z] - \Pr[W_1 \wedge \neg Z]| \\ &= |\Pr[W_2 \wedge Z] - \Pr[W_1 \wedge Z]| \\ &\leq \Pr[Z]. \end{aligned}$$

(The last step follows on noting that both $\Pr[W_1 \wedge Z]$ and $\Pr[W_2 \wedge Z]$ lie between 0 and Pr(Z).)

Using the Difference Lemma

The Difference Lemma (restatement)

Let Z, W_1, W_2 be (any) events defined over some probability space. Suppose that:

$$\Pr[W_1 \wedge \neg Z] = \Pr[W_2 \wedge \neg Z].$$

Then we have:

$$|\Pr[W_2] - \Pr[W_1]| \leq \Pr[Z].$$

- In security proofs, we will often face the situation where Z is some "bad" event of bounded, low probability.
- W_1 might be A's success probability in game G_1; W_2 in some modified game G_2.
- This lemma shows that if the lemma's required condition holds and Z is "rare" then A's success probabilities in the two games are close.
- Then, in a security proof, we can move from G_1 to G_2 without introducing too much difference in success probabilities.
- This lemma is one the main tools that we will use in *game hopping* proofs.

Proof of the PRP-PRF switching lemma

Lemma

Let E be a block cipher. Then for any algorithm A making q queries:

$$|\mathrm{Adv}_E^{PRP}(A) - \mathrm{Adv}_E^{PRF}(A)| \leq q^2/2^{n+1}.$$

Proof:

Let A be an (q, t, ε) adversary; we will run A in one of 3 different games:

- G_0: choose $f := E_K(\cdot)$, where $K \xleftarrow{\$} \{0,1\}^k$, and uses this f to respond to A's queries.
- G_1: choose $f \xleftarrow{\$}$ Perms[$\{0,1\}^n$].
- G_2: choose $f \xleftarrow{\$}$ Funcs[$\{0,1\}^n$, $\{0,1\}^n$].

Let W_i be the event that A outputs $b' = 1$ in game G_i; set $p_i = \Pr[W_i]$. Now:

- G_0 corresponds to the case "b=0" in both the PRP and PRF security games, hence $p_0 = \Pr[b'=1 | b=0]$.
- G_1 corresponds to the case "b=1" in the PRP security game, hence $p_1 = \Pr[b'=1 | b=1]$.
- G_2 corresponds to the case "b=1" in the PRF security game, hence $p_2 = \Pr[b'=1 | b=1]$.

Proof of the PRP-PRF switching lemma

So:

$$|p_1 - p_0| = |\Pr[b'=1 \mid b=1] - \Pr[b'=1 \mid b=0]| = \text{Adv}_E^{PRP}(A). \text{ (Probs in } G_1 \text{ and } G_0\text{)}$$

and:

$$|p_2 - p_0| = |\Pr[b'=1 \mid b=1] - \Pr[b'=1 \mid b=0]| = \text{Adv}_E^{PRF}(A). \text{ (Probs in } G_2 \text{ and } G_0\text{)}$$

Hence:

$$\begin{aligned} |\text{Adv}_E^{PRP}(A) - \text{Adv}_E^{PRF}(A)| &= \big|\, |p_1 - p_0| - |p_2 - p_0| \,\big| \\ &\leq |p_2 - p_1|. \qquad (*) \end{aligned}$$

Now consider games G_1 and G_2:

In G_1, adversary A interacts with a random permutation; in G_2 with a random function.

Games G_1 and G_2

G_1: $f \xleftarrow{\$}$ Perms[$\{0,1\}^n$]
G_2: $f \xleftarrow{\$}$ Funcs[$\{0,1\}^n$, $\{0,1\}^n$]

Upon receiving the i-th query x_i from A do:
If $x_i = x_j$ for some $j < i$
then $y_i \leftarrow y_j$
else
$y_i \xleftarrow{\$} \{0,1\}^n \setminus \{y_0, \ldots, y_{i-1}\}$
Return y_i.

Adversary A

x_2

b

Upon receiving the i-th query x_i from A do:
If $x_i = x_j$ for some $j < i$
then $y_i \leftarrow y_j$
else
$y_i \xleftarrow{\$} \{0,1\}^n$
Return y_i.

G_1

$y_i = f(x_i)$

G_2

$y_i = f(x_i)$

Proof of the PRP-PRF switching lemma

G_1 and G_2 are identical unless a repeated value occurs amongst the y_i in G_2.

Let Z denote this event.

A basic probability analysis shows that $\Pr[Z] \leq q^2/2^{n+1}$:

- Sample q values y_i uniformly at random from $\{0,1\}^n$.
- $\Pr[y_i = y_j] = 2^{-n}$ for each pair of indices (i,j).
- There are $q(q-1)/2 \leq q^2/2$ pairs of indices.
- The events are not independent, but we can use the union bound to complete the argument:

$$\Pr[Z] = \Pr[y_i = y_j \text{ for some } i \neq j] \leq q^2/2 \cdot 2^{-n} = q^2/2^{n+1}.$$

- NB this analysis depends only on the number of queries made by *A* and is independent of *A*'s running time.

Back to Proof of the PRP-PRF switching lemma

Recall:

W_1 is the event that A outputs "1" in game G_1, $\Pr[W_1] = p_1$.

W_2 is the event that A outputs "1" in game G_2, $\Pr[W_2] = p_2$.

G_1 and G_2 are identical unless event Z (a collision amongst the y_i) occurs.

So we have: event $W_1 \wedge \neg Z$ occurs if and only if event $W_2 \wedge \neg Z$ occurs.

Now apply the difference lemma to obtain:

$$|p_2 - p_1| = |\Pr[W_2] - \Pr[W_1]| \leq \Pr[Z] \leq q^2/2^{n+1}.$$

Recall from (*):

$$\begin{aligned} |\mathrm{Adv}_E^{PRP}(A) - \mathrm{Adv}_E^{PRF}(A)| &\leq |p_2 - p_1| \\ &\leq q^2/2^{n+1}. \end{aligned}$$

This completes the proof of the PRP-PRF switching lemma.

Homework

- **Action 1:** for more detail on the PRP-PRF switching lemma, read Boneh-Shoup, Theorem 4.4.
- **Action 2:** for more on the Difference Lemma (sometimes called the "Fundamental Lemma of Game Playing" read Boneh-Shoup, Theorem 4.7.
- **Action 3:** start exercise sheet 2 and prepare for lab 2.

- **Next week's lectures:**
  - Proof that CTR[*E*] is secure under the assumption that *E* is a PRP;
  - Attacks on CBC mode: padding oracles and the BEAST!