Advanced Cryptography

（Provable Security）

Yi LIU

Block Ciphers (Pseudorandom Permutations)

- Even in the case of $in = out$, A function from $\{0,1\}^{in}$ to $\{0,1\}^{out}$ chosen at random is unlikely to have an inverse, therefore a PRF instantiated with a random key is unlikely to have an inverse.
- A pseudorandom permutation (PRP) — also called a block cipher — is essentially a PRF that is guaranteed to be invertible for every choice of seed.

Block Ciphers (Pseudorandom Permutations)

Let $F\colon \{0,1\}^{\lambda} \times \{0,1\}^{blen} \to \{0,1\}^{blen}$ be a deterministic function. We refer to $blen$ as the blocklength of F and any element of $\{0,1\}^{blen}$ as a block. We call F a secure pseudorandom permutation (PRP) (block cipher) if the following two conditions hold:

- **Invertible given k**: There is a function $F^{-1} : \{0,1\}^{\lambda} \times \{0,1\}^{blen} \to \{0,1\}^{blen}$ satisfying $F^{-1}(k, F(k,x)) = x$, for all $k \in \{0,1\}^{\lambda}$ and all $x \in \{0,1\}^{blen}$.
- **Security**: $\mathcal{L}^{F}_{\text{prp-real}} \approx \mathcal{L}^{F}_{\text{prp-rand}}$, where:

$\mathcal{L}^{F}_{\text{prp-real}}$
$k \leftarrow \{0,1\}^{\lambda}$
$\underline{\text{LOOKUP}(x \in \{0,1\}^{blen})}$:
return $F(k,x)$

$\mathcal{L}^{F}_{\text{prp-rand}}$
$T :=$ empty assoc. array
$\underline{\text{LOOKUP}(x \in \{0,1\}^{blen})}$:
if $T[x]$ undefined:
$T[x] \leftarrow \{0,1\}^{blen} \setminus T.\text{values}$
return $T[x]$

"T .values" refers to $\{v \mid \exists x : T[x] = v\}$

Switching Lemma (PRPs are PRFs, Too!)

Lemma If $in = out = blen = \lambda$, $\mathcal{L}_{\text{prp-rand}} \approx \mathcal{L}_{\text{prf-rand}}$.

$$\mathcal{L}^F_{\text{prf-rand}}$$

$T :=$ empty assoc. array

$\underline{\textsc{lookup}(x \in \{0,1\}^{in}):}$
  if $T[x]$ undefined:
    $T[x] \leftarrow \{0,1\}^{out}$
  return $T[x]$

$$\mathcal{L}^F_{\text{prp-rand}}$$

$T :=$ empty assoc. array

$\underline{\textsc{lookup}(x \in \{0,1\}^{blen}):}$
  if $T[x]$ undefined:
    $T[x] \leftarrow \{0,1\}^{blen} \setminus T.\text{values}$
  return $T[x]$

Switching Lemma (PRPs are PRFs, Too!)

Lemma If $in = out = blen = \lambda$, $\mathcal{L}_{\text{prp-rand}} \approx \mathcal{L}_{\text{prf-rand}}$.

proof Recall the replacement-sampling lemma

$\mathcal{L}_{\text{samp-L}}$

$\underline{\text{SAMP}():}$

$\quad r \leftarrow \{0,1\}^\lambda$

$\quad \text{return } r$

$\mathcal{L}_{\text{samp-R}}$

$R := \emptyset$

$\underline{\text{SAMP}():}$

$\quad r \leftarrow \{0,1\}^\lambda \setminus R$

$\quad R := R \cup \{r\}$

$\quad \text{return } r$

Switching Lemma (PRPs are PRFs, Too!)

Lemma If $in = out = blen = \lambda$, $\mathcal{L}_{\text{prp-rand}} \approx \mathcal{L}_{\text{prf-rand}}$.

proof Recall the replacement-sampling lemma

Now consider the following library

$$\begin{array}{|l|}\hline \mathcal{L}^* \\ \hline T := \text{empty assoc. array} \\ \\ \underline{\text{LOOKUP}(x \in \{0,1\}^\lambda):} \\ \quad \text{if } T[x] \text{ undefined:} \\ \quad\quad T[x] \leftarrow \text{SAMP}() \\ \quad \text{return } T[x] \\ \hline \end{array} \diamond \begin{array}{|l|}\hline \mathcal{L}_{\text{samp-L}} \\ \hline \underline{\text{SAMP}():} \\ \quad r \leftarrow \{0,1\}^\lambda \\ \quad \text{return } r \\ \hline \end{array} \qquad \begin{array}{|l|}\hline \mathcal{L}^F_{\text{prf-rand}} \\ \hline T := \text{empty assoc. array} \\ \\ \underline{\text{LOOKUP}(x \in \{0,1\}^{in}):} \\ \quad \text{if } T[x] \text{ undefined:} \\ \quad\quad T[x] \leftarrow \{0,1\}^{out} \\ \quad \text{return } T[x] \\ \hline \end{array}$$

Switching Lemma (PRPs are PRFs, Too!)

Lemma If $in = out = blen = \lambda$, $\mathcal{L}_{\text{prp-rand}} \approx \mathcal{L}_{\text{prf-rand}}$.

proof Recall the replacement-sampling lemma

Now consider the following library

$\mathcal{L}^*$

$T :=$ empty assoc. array

$\underline{\text{LOOKUP}(x \in \{0,1\}^\lambda):}$
  if $T[x]$ undefined:
    $T[x] \leftarrow \text{SAMP}()$
  return $T[x]$

$\diamond$

$\mathcal{L}_{\text{samp-R}}$

$R := \emptyset$

$\underline{\text{SAMP}():}$
  $r \leftarrow \{0,1\}^\lambda \setminus R$
  $R := R \cup \{r\}$
  return r

$\mathcal{L}^F_{\text{prp-rand}}$

$T :=$ empty assoc. array

$\underline{\text{LOOKUP}(x \in \{0,1\}^{blen}):}$
  if $T[x]$ undefined:
    $T[x] \leftarrow \{0,1\}^{blen} \setminus T.\text{values}$
  return $T[x]$

Switching Lemma (PRPs are PRFs, Too!)

Lemma If $in = out = blen = \lambda$, $\mathcal{L}_{\text{prp-rand}} \approx \mathcal{L}_{\text{prf-rand}}$.

proof Recall the replacement-sampling lemma

$$\mathcal{L}_{\text{prf-rand}} \equiv \mathcal{L}^* \diamond \mathcal{L}_{\text{samp-L}} \approx \mathcal{L}^* \diamond \mathcal{L}_{\text{samp-R}} \equiv \mathcal{L}_{\text{prp-rand}}$$

Corollary Let $F\colon \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^\lambda$ be a secure PRP (with $blen = \lambda$). Then F is also a secure PRF.

proof $\mathcal{L}^F_{\text{prf-real}} \equiv \mathcal{L}^F_{\text{prp-real}} \approx \mathcal{L}^F_{\text{prp-rand}} \approx \mathcal{L}^F_{\text{prf-rand}}$

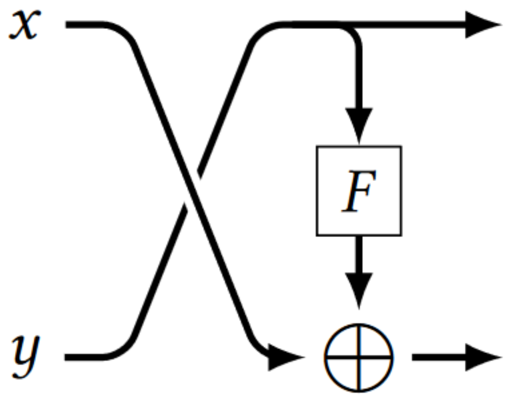
Constructing a PRP from a PRF

- Feistel construction: convert a not-necessarily-invertible function $F : \{0,1\}^n \to \{0,1\}^n$ into an invertible function $F^* : \{0,1\}^{2n} \to \{0,1\}^{2n}$. The function F^* is called the Feistel round with round function F

$\underline{F^*(x \| y)}$:

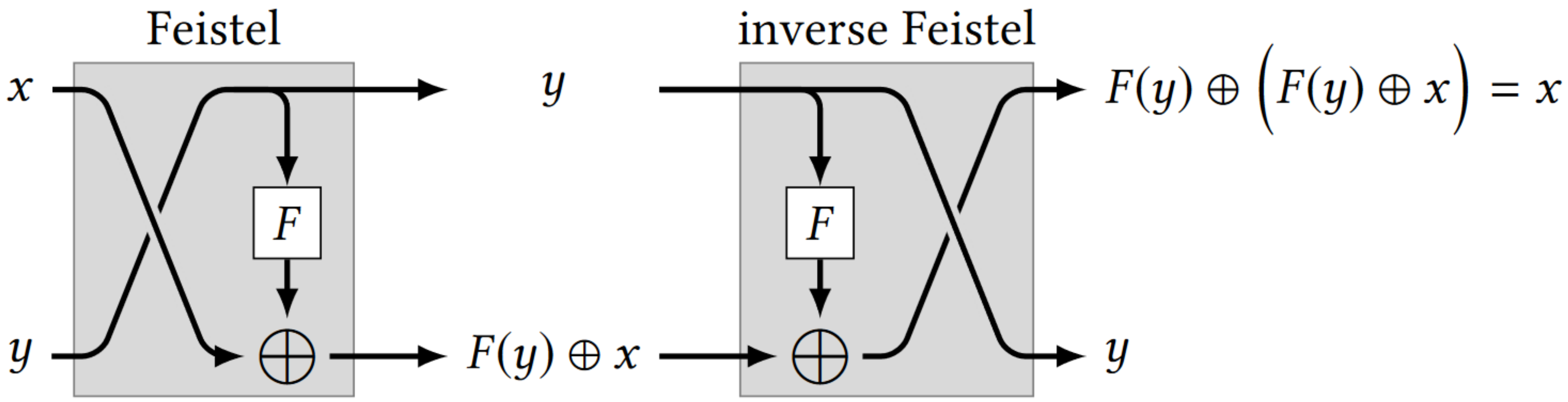
  // each of x, y are n bits

  return $y \| (F(y) \oplus x)$



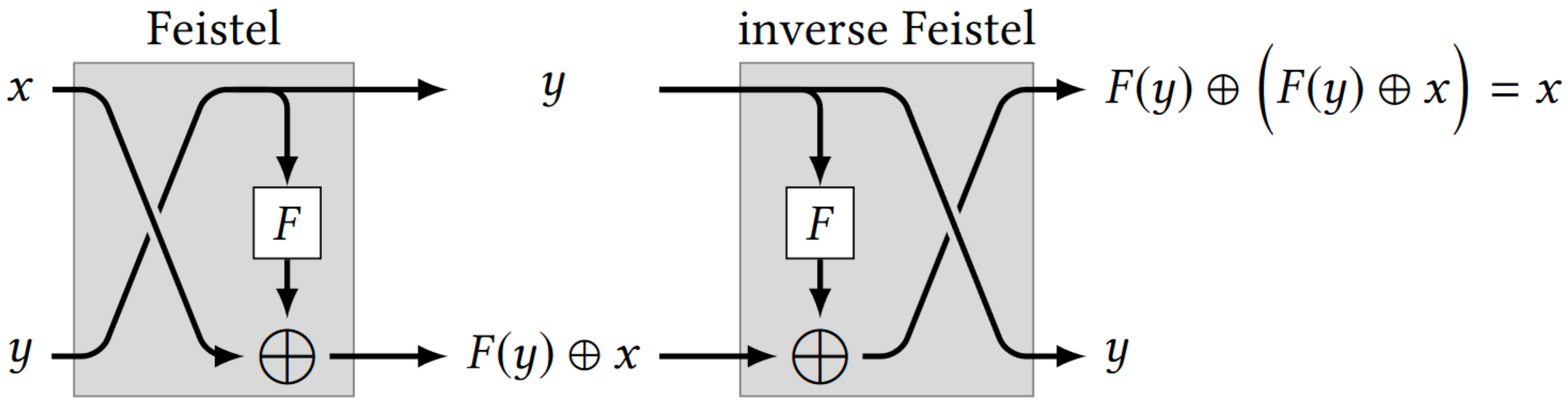
Constructing a PRP from a PRF

- Feistel construction: convert a not-necessarily-invertible function $F : \{0,1\}^n \to \{0,1\}^n$ into an invertible function $F^* : \{0,1\}^{2n} \to \{0,1\}^{2n}$. The function F^* is called the Feistel round with round function F



Feistel Construction

- If $F(y) = 0^n$, then $F^*(x\|y) \;=\; y\|(F\,(y) \oplus x) = y\|(0^n \oplus x) = y\|x$
- If $F(y) = y$, then $F^*(x\|y) = y\|(F(y) \oplus x) = y\|(y \oplus x)$, also invertible since we can solve for x.

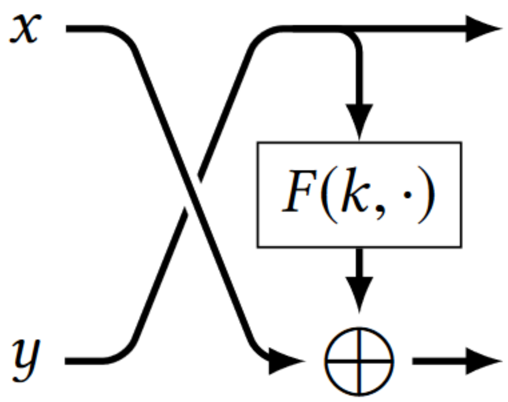


Feistel Cipher

- The output of F^* contains half of its input, making it quite trivial to break the PRP-security of F^*.
- We can avoid this trivial attack by performing several Feistel rounds in succession, resulting in a construction called a Feistel cipher.
- At each round, we can even use a different key to the round function. (key schedule)

$\underline{F^*(k, x \| y)\text{:}}$

return $y \| (F(k, y) \oplus x)$



Feistel Cipher

- We can avoid this trivial attack by performing several Feistel rounds in succession, resulting in a construction called a Feistel cipher.
- At each round, we can even use a different key to the round function. (key schedule)

$\underline{\mathbb{F}_r((k_1,\ldots,k_r), v_0 \| v_1)}$:

for $i = 1$ to r:

  $v_{i+1} := F(k_i, v_i) \oplus v_{i-1}$

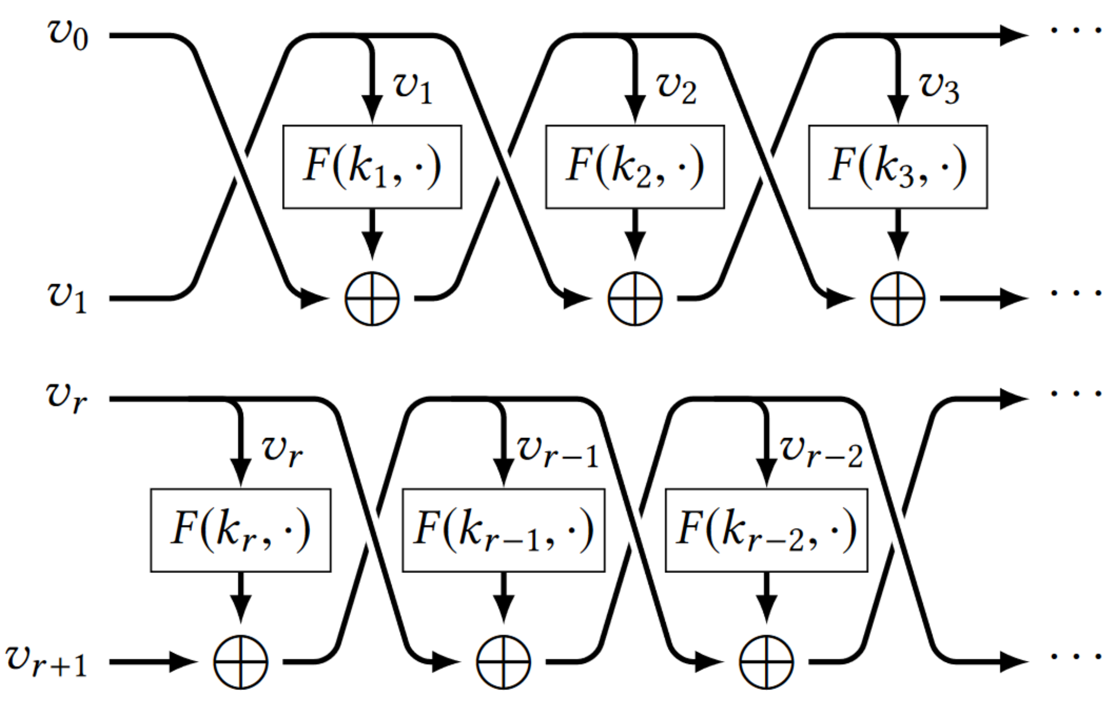
return $v_r \| v_{r+1}$

$\underline{\mathbb{F}_r^{-1}((k_1,\ldots,k_r), v_r \| v_{r+1})}$:

for $i = r$ downto 1:

  $v_{i-1} := F(k_i, v_i) \oplus v_{i+1}$

return $v_0 \| v_1$

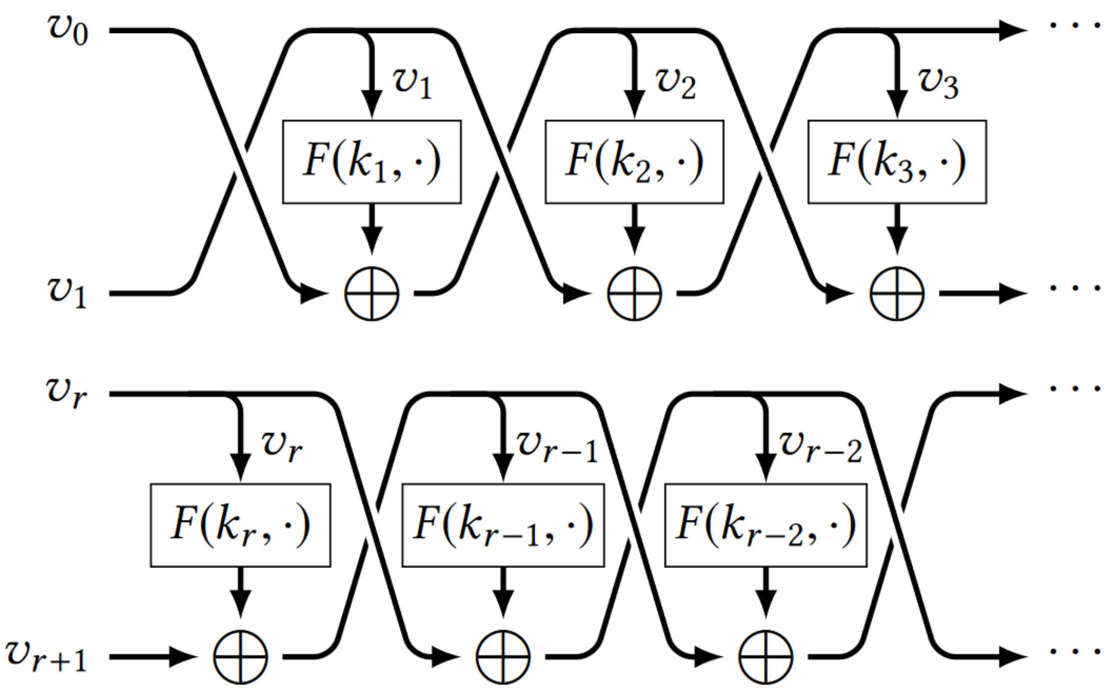


Feistel Cipher

- **Theorem** (Luby-Rackoff) If $F\colon \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^\lambda$ is a secure PRF, then the 3-round Feistel cipher is a secure PRP.

$\mathbb{F}_r((k_1,\ldots,k_r), v_0 \| v_1)$:
  for $i = 1$ to r:
    $v_{i+1} := F(k_i, v_i) \oplus v_{i-1}$
  return $v_r \| v_{r+1}$

$\mathbb{F}_r^{-1}((k_1,\ldots,k_r), v_r \| v_{r+1})$:
  for $i = r$ downto 1:
    $v_{i-1} := F(k_i, v_i) \oplus v_{i+1}$
  return $v_0 \| v_1$



PRFs and Block Ciphers in Practice

- We currently have no proof that any secure PRP exists.
- Advanced Encryption Standard (AES) block cipher demonstrates the cryptographic community's best efforts at instilling such confidence.

- As we have seen, once you have access to a good block cipher, it can be used directly also as a secure PRF, and it can be used to construct a simple PRG.
  - The PRF-security and PRG-security of these constructions is guaranteed to be as good as the PRP-security of AES.

Strong Pseudorandom Permutations

- The PRP security definition only guarantees a security property for F and not its inverse.
- It is possible to construct F which is a secure PRP, whose inverse F^{-1} is not a secure PRP!

$$E_k(x) = \begin{cases} 0 & \text{if } x = k \\ \mathrm{AES}_k(k) & \text{if } x = \mathrm{AES}_k^{-1}(0) \\ \mathrm{AES}_k(x) & \text{otherwise.} \end{cases}$$

Strong Pseudorandom Permutations

Definition Let $F: \{0,1\}^\lambda \times \{0,1\}^{blen} \to \{0,1\}^{blen}$ be a deterministic function. We say that F is a secure strong pseudorandom permutation (SPRP) if $\mathcal{L}^F_{\text{sprp-real}} \approx \mathcal{L}^F_{\text{sprp-rand}}$.

$\mathcal{L}^F_{\text{sprp-real}}$

$k \leftarrow \{0,1\}^\lambda$

$\underline{\textsc{lookup}(x \in \{0,1\}^{blen}):}$
  return $F(k, x)$

$\underline{\textsc{invlookup}(y \in \{0,1\}^{blen}):}$
  return $F^{-1}(k, y)$

$\mathcal{L}^F_{\text{sprp-rand}}$

T, T_{inv} := empty assoc. arrays

$\underline{\textsc{lookup}(x \in \{0,1\}^{blen}):}$
  if $T[x]$ undefined:
    $y \leftarrow \{0,1\}^{blen} \setminus T$.values
    $T[x] := y; \quad T_{inv}[y] := x$
  return $T[x]$

$\underline{\textsc{invlookup}(y \in \{0,1\}^{blen}):}$
  if $T_{inv}[y]$ undefined:
    $x \leftarrow \{0,1\}^{blen} \setminus T_{inv}$.values
    $T_{inv}[y] := x; \quad T[x] := y$
  return $T_{inv}[y]$

Strong Pseudorandom Permutations

Definition Let $F: \{0,1\}^\lambda \times \{0,1\}^{blen} \to \{0,1\}^{blen}$ be a deterministic function. We say that F is a secure strong pseudorandom permutation (SPRP) if $\mathcal{L}^F_{\text{sprp-real}} \approx \mathcal{L}^F_{\text{sprp-rand}}$.

Theorem (Luby-Rackoff) If $F: \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^\lambda$ is a secure PRF, then the 4-round Feistel cipher is a secure SPRP.

Security Against Chosen Plaintext Attacks

CPA security

- Our previous security definitions for encryption capture the case where a key is used to encrypt only one plaintext. Clearly it would be more useful to have an encryption scheme that allows many plaintexts to be encrypted under the same key.

Definition Let Σ be an encryption scheme. We say that Σ has security against chosen-plaintext attacks (CPA security) if $\mathcal{L}^{\Sigma}_{\text{cpa-L}} \approx \mathcal{L}^{\Sigma}_{\text{cpa-R}}$, where:

$\mathcal{L}^{\Sigma}_{\text{cpa-L}}$	$\mathcal{L}^{\Sigma}_{\text{cpa-R}}$
$k \leftarrow \Sigma.\mathsf{KeyGen}$	$k \leftarrow \Sigma.\mathsf{KeyGen}$
$\underline{\textsc{eavesdrop}(m_L, m_R \in \Sigma.\mathcal{M})\text{:}}$	$\underline{\textsc{eavesdrop}(m_L, m_R \in \Sigma.\mathcal{M})\text{:}}$
$\quad c := \Sigma.\mathsf{Enc}(k, m_L)$	$\quad c := \Sigma.\mathsf{Enc}(k, m_R)$
$\quad$ return c	$\quad$ return c

CPA security is often called “IND-CPA” security: indistinguishability of ciphertexts under chosen-plaintext attack

Limits of Deterministic Encryption

- For a block cipher, F corresponds to encryption, F^{-1} corresponds to decryption, and all outputs of F look pseudorandom. Use block cipher "as-is", is it satisfy CPA security?
- Consider the following adversary A, that tries to distinguish the $L_{\text{cpa}-*}$ libraries:

$$\mathcal{A}$$

$$c_1 := \text{EAVESDROP}(0^{blen}, 0^{blen})$$
$$c_2 := \text{EAVESDROP}(0^{blen}, 1^{blen})$$
$$\text{return } c_1 \stackrel{?}{=} c_2$$

Limits of Deterministic Encryption

- Consider the following adversary $\mathcal{A}$, that tries to distinguish the $L_{\text{cpa}-*}$ libraries:

$\mathcal{A}$
$c_1 := \text{EAVESDROP}(\mathtt{0}^{blen}, \mathtt{0}^{blen})$
$c_2 := \text{EAVESDROP}(\mathtt{0}^{blen}, \mathtt{1}^{blen})$
return $c_1 \stackrel{?}{=} c_2$

$\diamond$

$\mathcal{L}^{\Sigma}_{\text{cpa-L}}$
$k \leftarrow \{\mathtt{0},\mathtt{1}\}^{\lambda}$
$\underline{\text{EAVESDROP}(m_L, m_R)\text{:}}$
$c := F(k, m_L)$
return c

$\mathcal{A} \diamond \mathcal{L}_{\text{cpa}-\text{L}}$ always outputs 1

Limits of Deterministic Encryption

- Consider the following adversary $\mathcal{A}$, that tries to distinguish the $L_{\text{cpa}-*}$ libraries:

$$\boxed{\begin{array}{l} \hline \mathcal{A} \\ \hline c_1 := \text{EAVESDROP}(\mathtt{0}^{blen}, \mathtt{0}^{blen}) \\ c_2 := \text{EAVESDROP}(\mathtt{0}^{blen}, \mathtt{1}^{blen}) \\ \text{return } c_1 \stackrel{?}{=} c_2 \end{array}} \diamond \boxed{\begin{array}{l} \hline \mathcal{L}^{\Sigma}_{\text{cpa-R}} \\ \hline k \leftarrow \{\mathtt{0},\mathtt{1}\}^{\lambda} \\ \\ \underline{\text{EAVESDROP}(m_L, m_R):} \\ \quad c := F(k, m_R) \\ \quad \text{return } c \end{array}}$$

$\mathcal{A} \diamond \mathcal{L}_{\text{cpa}-\text{L}}$ always outputs 1

$\mathcal{A} \diamond \mathcal{L}_{\text{cpa}-\text{R}}$ never outputs 1

This adversary has advantage 1 in distinguishing the libraries, so the bare block cipher F is not a CPA-secure encryption scheme.

Limits of Deterministic Encryption

- The reason a bare block cipher does not provide CPA security is that it is deterministic. Calling $\mathrm{Enc}(k, m)$ twice — with the same key and same plaintext — leads to the same ciphertext.

- Deterministic encryption can never be CPA-secure!
- It leaks whether two ciphertexts encode the same plaintext.

Avoiding Deterministic Encryption

We must design an Enc algorithm such that calling it twice with the same plaintext and key results in different ciphertexts.

- Encryption/decryption can be stateful, meaning that every call to Enc or Dec will actually modify the value of k. (symmetric ratchet construction)
- Encryption can be randomized. Each time a plaintext is encrypted, the Enc algorithm chooses fresh, independent randomness specific to that encryption.
- Encryption can be nonce-based. A "nonce" stands for "number used only once".
  - A nonce does not need to be chosen randomly; it does not need to be secret; it only needs to be distinct among all calls made to Enc.

Avoiding Deterministic Encryption

- Encryption can be nonce-based. A “nonce” stands for “number used only once”.
  - A nonce only needs to be distinct among all calls made to Enc.
  - Nonce-based encryption requires a change to the interface of encryption, and therefore a change to the correctness & security definitions as well. The encryption/decryption algorithms syntax is updated to $\mathrm{Enc}(k, v, m)$ and $\mathrm{Dec}(k, v, c)$, where v is a nonce.
  - The correctness property is that $\mathrm{Dec}(k, v, \mathrm{Enc}(k, v, m)) = m$ for all k, v, m, so both encryption & decryption algorithms should use the same nonce.

$k \leftarrow \Sigma.\mathrm{KeyGen}$ $V := \emptyset$ $\textsc{eavesdrop}(v, m_L, m_R \in \Sigma.\mathcal{M})$: if $v \in V$: return err $V := V \cup \{v\}$ $c := \Sigma.\mathrm{Enc}(k, v, m_L)$ return c	$\approx$	$k \leftarrow \Sigma.\mathrm{KeyGen}$ $V := \emptyset$ $\textsc{eavesdrop}(v, m_L, m_R \in \Sigma.\mathcal{M})$: if $v \in V$: return err $V := V \cup \{v\}$ $c := \Sigma.\mathrm{Enc}(k, v, m_R)$ return c

Real-vs-Random Version of CPA

Definition Let Σ be an encryption scheme. We say that Σ has pseudorandom ciphertexts in the presence of chosen-plaintext attacks (CPA$ security) if $\mathcal{L}^{\Sigma}_{\text{cpa\$-real}} \approx \mathcal{L}^{\Sigma}_{\text{cpa\$-rand}}$, where:

$\mathcal{L}^{\Sigma}_{\text{cpa\$-real}}$

$k \leftarrow \Sigma.\text{KeyGen}$

$\underline{\textsc{ctxt}(m \in \Sigma.\mathcal{M}):}$

$\quad c := \Sigma.\text{Enc}(k, m)$

$\quad$ return c

$\mathcal{L}^{\Sigma}_{\text{cpa\$-rand}}$

$\underline{\textsc{ctxt}(m \in \Sigma.\mathcal{M}):}$

$\quad c \leftarrow \Sigma.\mathcal{C}$

$\quad$ return c

This definition is also called "IND$-CPA", meaning "indistinguishable from random under chosen plaintext attacks."

Real-vs-Random Version of CPA

Definition Let Σ be an encryption scheme. We say that Σ has pseudorandom ciphertexts in the presence of chosen-plaintext attacks (CPA$ security) if $\mathcal{L}^{\Sigma}_{\text{cpa\$-real}} \approx \mathcal{L}^{\Sigma}_{\text{cpa\$-rand}}$.

$\mathcal{L}^{\Sigma}_{\text{cpa\$-real}}$
$k \leftarrow \Sigma.\text{KeyGen}$
$\underline{\textsc{ctxt}(m \in \Sigma.\mathcal{M})\text{:}}$
$\quad c := \Sigma.\text{Enc}(k, m)$
$\quad$return c

$\mathcal{L}^{\Sigma}_{\text{cpa\$-rand}}$
$\underline{\textsc{ctxt}(m \in \Sigma.\mathcal{M})\text{:}}$
$\quad c \leftarrow \Sigma.\mathcal{C}$
$\quad$return c

- It is easier to prove CPA$ security than to prove CPA security.
- CPA$ security implies CPA security.
- Most of the schemes we will consider achieve CPA$ anyway.

CPA$ security implies CPA security

Claim If an encryption scheme has CPA$ security, then it also has CPA security.

proof

We want to prove that $\mathcal{L}^{\Sigma}_{\text{cpa}-\text{L}} \approx \mathcal{L}^{\Sigma}_{\text{cpa}-\text{R}}$, using the assumption that $\mathcal{L}^{\Sigma}_{\text{cpa\$}-\text{real}} \approx \mathcal{L}^{\Sigma}_{\text{cpa\$}-\text{rand}}$.

$\mathcal{L}^{\Sigma}_{\text{cpa-L}}$

$k \leftarrow \Sigma.\text{KeyGen}$

$\underline{\text{EAVESDROP}(m_L, m_R)\text{:}}$
$\quad c := \Sigma.\text{Enc}(k, m_L)$
$\quad$ return c

$\Rightarrow$

$\underline{\text{EAVESDROP}(m_L, m_R)\text{:}}$
$\quad c := \text{CTXT}(m_L)$
$\quad$ return c

$\diamond$

$\mathcal{L}^{\Sigma}_{\text{cpa\$-real}}$

$k \leftarrow \Sigma.\text{KeyGen}$

$\underline{\text{CTXT}(m)\text{:}}$
$\quad c := \Sigma.\text{Enc}(k, m)$
$\quad$ return c

CPA\$ security implies CPA security

Claim If an encryption scheme has CPA\$ security, then it also has CPA security.

proof

We want to prove that $\mathcal{L}^{\Sigma}_{\text{cpa-L}} \approx \mathcal{L}^{\Sigma}_{\text{cpa-R}}$, using the assumption that $\mathcal{L}^{\Sigma}_{\text{cpa\$-real}} \approx \mathcal{L}^{\Sigma}_{\text{cpa\$-rand}}$.

$$
\begin{array}{|l|}
\hline
\underline{\textsc{eavesdrop}(m_L, m_R):} \\
\quad c := \textsc{ctxt}(m_L) \\
\quad \text{return } c \\
\hline
\end{array}
\diamond
\begin{array}{|l|}
\hline
\qquad \mathcal{L}^{\Sigma}_{\text{cpa\$-real}} \\
\hline
k \leftarrow \Sigma.\mathsf{KeyGen} \\
\\
\underline{\textsc{ctxt}(m):} \\
\quad c := \Sigma.\mathsf{Enc}(k, m) \\
\quad \text{return } c \\
\hline
\end{array}
\Rightarrow
\begin{array}{|l|}
\hline
\underline{\textsc{eavesdrop}(m_L, m_R):} \\
\quad c := \textsc{ctxt}(m_L) \\
\quad \text{return } c \\
\hline
\end{array}
\diamond
\begin{array}{|l|}
\hline
\qquad \mathcal{L}^{\Sigma}_{\text{cpa\$-rand}} \\
\hline
\underline{\textsc{ctxt}(m):} \\
\quad c \leftarrow \Sigma.\mathcal{C} \\
\quad \text{return } c \\
\hline
\end{array}
$$

CPA$ security implies CPA security

Claim If an encryption scheme has CPA$ security, then it also has CPA security.

proof

We want to prove that $\mathcal{L}^{\Sigma}_{\text{cpa}-\text{L}} \approx \mathcal{L}^{\Sigma}_{\text{cpa}-\text{R}}$, using the assumption that $\mathcal{L}^{\Sigma}_{\text{cpa\$}-\text{real}} \approx \mathcal{L}^{\Sigma}_{\text{cpa\$}-\text{rand}}$.

$\text{EAVESDROP}(m_L, m_R)$:
$c := \text{CTXT}(m_L)$
return c

$\diamond$

$\mathcal{L}^{\Sigma}_{\text{cpa\$-rand}}$
$\text{CTXT}(m)$:
$c \leftarrow \Sigma.C$
return c

$\Rightarrow$

$\text{EAVESDROP}(m_L, m_R)$:
$c := \text{CTXT}(m_R)$
return c

$\diamond$

$\mathcal{L}^{\Sigma}_{\text{cpa\$-rand}}$
$\text{CTXT}(m)$:
$c \leftarrow \Sigma.C$
return c

CPA$ security implies CPA security

Claim If an encryption scheme has CPA$ security, then it also has CPA security.

proof

We want to prove that $\mathcal{L}^{\Sigma}_{\text{cpa-L}} \approx \mathcal{L}^{\Sigma}_{\text{cpa-R}}$, using the assumption that $\mathcal{L}^{\Sigma}_{\text{cpa\$-real}} \approx \mathcal{L}^{\Sigma}_{\text{cpa\$-rand}}$.

EAVESDROP(m_L, m_R):
- $c := \text{CTXT}(m_R)$
- return c

$\diamond$

$\mathcal{L}^{\Sigma}_{\text{cpa\$-rand}}$

CTXT(m):
- $c \leftarrow \Sigma.\mathcal{C}$
- return c

$\Rightarrow$

EAVESDROP(m_L, m_R):
- $c := \text{CTXT}(m_R)$
- return c

$\diamond$

$\mathcal{L}^{\Sigma}_{\text{cpa\$-real}}$

$k \leftarrow \Sigma.\text{KeyGen}$

CTXT(m):
- $c := \Sigma.\text{Enc}(k, m)$
- return c

CPA$ security implies CPA security

Claim If an encryption scheme has CPA$ security, then it also has CPA security.

proof

We want to prove that $\mathcal{L}^{\Sigma}_{\text{cpa−L}} \approx \mathcal{L}^{\Sigma}_{\text{cpa−R}}$, using the assumption that $\mathcal{L}^{\Sigma}_{\text{cpa\$−real}} \approx \mathcal{L}^{\Sigma}_{\text{cpa\$−rand}}$.

$\underline{\text{EAVESDROP}(m_L, m_R)\text{:}}$
$c := \text{CTXT}(m_R)$
return c

$\diamond$

$\mathcal{L}^{\Sigma}_{\text{cpa\$-real}}$
$k \leftarrow \Sigma.\text{KeyGen}$
$\underline{\text{CTXT}(m)\text{:}}$
$c := \Sigma.\text{Enc}(k, m)$
return c

$\mathcal{L}^{\Sigma}_{\text{cpa-R}}$
$k \leftarrow \Sigma.\text{KeyGen}$
$\underline{\text{EAVESDROP}(m_L, m_R)\text{:}}$
$c := \Sigma.\text{Enc}(k, m_R)$
return c

CPA-Secure Encryption Based On PRFs

- On the one hand, we need an encryption method that is randomized, so that each plaintext m is mapped to a large number of potential ciphertexts.
- On the other hand, the decryption method must be able to recognize all of these various ciphertexts as being encryptions of m.
- If Alice and Bob share a huge table T initialized with uniform data, then Alice can encrypt a plaintext m to Bob by saying something like “this is encrypted with one-time pad, using key #674696273” and sending $T[674696273] \oplus m$. Seeing the number 674696273 doesn’t help the eavesdropper know what $T[674696273]$ is.
- Use PRF!

CPA-Secure Encryption Based On PRFs

Let F be a secure PRF with $in = \lambda$. Define the following encryption scheme based on F

$$\mathcal{K} = \{0,1\}^\lambda \qquad \mathcal{M} = \{0,1\}^{out} \qquad \mathcal{C} = \{0,1\}^\lambda \times \{0,1\}^{out}$$

$\underline{\mathsf{KeyGen}:}$

$k \leftarrow \{0,1\}^\lambda$
return k

$\underline{\mathsf{Enc}(k, m):}$

$r \leftarrow \{0,1\}^\lambda$
$x := F(k, r) \oplus m$
return (r, x)

$\underline{\mathsf{Dec}(k, (r, x)):}$

$m := F(k, r) \oplus x$
return m

Claim This scheme has CPA$ security (and therefore CPA security) if F is a secure PRF.

CPA-Secure Encryption Based On PRFs

Claim This scheme has CPA\$ security (and therefore CPA security) if F is a secure PRF.

Proof We prove that $\mathcal{L}^{\Sigma}_{\text{cpa\$-real}} \approx \mathcal{L}^{\Sigma}_{\text{cpa\$-rand}}$ using the hybrid technique

$\mathcal{L}^{\Sigma}_{\text{cpa\$-real}}$

$k \leftarrow \{0,1\}^{\lambda}$

$\underline{\textsc{ctxt}(m):}$

$\quad r \leftarrow \{0,1\}^{\lambda}$

$\quad x := F(k,r) \oplus m$

$\quad \text{return } (r,x)$

$\Rightarrow$

$\underline{\textsc{ctxt}(m):}$

$\quad r \leftarrow \{0,1\}^{\lambda}$

$\quad z := \textsc{lookup}(r)$

$\quad x := z \oplus m$

$\quad \text{return } (r,x)$

$\diamond$

$\mathcal{L}^{F}_{\text{prf-real}}$

$k \leftarrow \{0,1\}^{\lambda}$

$\underline{\textsc{lookup}(r):}$

$\quad \text{return } F(k,r)$

CPA-Secure Encryption Based On PRFs

Claim This scheme has CPA$ security (and therefore CPA security) if F is a secure PRF.

Proof We prove that $\mathcal{L}^{\Sigma}_{\text{cpa\$-real}} \approx \mathcal{L}^{\Sigma}_{\text{cpa\$-rand}}$ using the hybrid technique

$\underline{\textsc{ctxt}(m):}$
$r \leftarrow \{0,1\}^{\lambda}$
$z := \textsc{lookup}(r)$
$x := z \oplus m$
return (r, x)

$\diamond$

$\mathcal{L}^{F}_{\text{prf-real}}$
$k \leftarrow \{0,1\}^{\lambda}$
$\underline{\textsc{lookup}(r):}$
return $F(k, r)$

$\Rightarrow$

$\underline{\textsc{ctxt}(m):}$
$r \leftarrow \{0,1\}^{\lambda}$
$z := \textsc{lookup}(r)$
$x := z \oplus m$
return (r, x)

$\diamond$

$\mathcal{L}^{F}_{\text{prf-rand}}$
$T :=$ empty
$\underline{\textsc{lookup}(r):}$
if $T[r]$ undefined:
$T[r] \leftarrow \{0,1\}^{out}$
return $T[r]$

Q: Is the proof over?

r may be repeated! Our proof must explicitly contain reasoning about why PRF inputs are unlikely to be repeated.

CPA-Secure Encryption Based On PRFs

Claim This scheme has CPA$ security (and therefore CPA security) if F is a secure PRF.

Proof We prove that $\mathcal{L}^{\Sigma}_{\text{cpa\$-real}} \approx \mathcal{L}^{\Sigma}_{\text{cpa\$-rand}}$ using the hybrid technique

$\textsc{ctxt}(m)$:
- $r \leftarrow \{0,1\}^{\lambda}$
- $z := \textsc{lookup}(r)$
- $x := z \oplus m$
- return (r, x)

$\diamond$

$\mathcal{L}^{F}_{\text{prf-rand}}$

$T :=$ empty

$\textsc{lookup}(r)$:
- if $T[r]$ undefined:
  - $T[r] \leftarrow \{0,1\}^{out}$
- return $T[r]$

$\textsc{ctxt}(m)$:
- $r \leftarrow \textsc{samp}()$
- $z := \textsc{lookup}(r)$
- $x := z \oplus m$
- return (r, x)

$\diamond$

$\mathcal{L}^{F}_{\text{prf-rand}}$

$T :=$ empty

$\textsc{lookup}(r)$:
- if $T[r]$ undefined:
  - $T[r] \leftarrow \{0,1\}^{out}$
- return $T[r]$

$\diamond$

$\mathcal{L}_{\text{samp-L}}$

$\textsc{samp}()$:
- $r \leftarrow \{0,1\}^{\lambda}$
- return r

CPA-Secure Encryption Based On PRFs

Claim This scheme has CPA$ security (and therefore CPA security) if F is a secure PRF.

Proof We prove that $\mathcal{L}^{\Sigma}_{\text{cpa\$-real}} \approx \mathcal{L}^{\Sigma}_{\text{cpa\$-rand}}$ using the hybrid technique

```
CTXT(m):
  r ← SAMP()
  z := LOOKUP(r)
  x := z ⊕ m
  return (r, x)
```

◇

$\mathcal{L}^{F}_{\text{prf-rand}}$

```
T := empty

LOOKUP(r):
  if T[r] undefined:
    T[r] ← {0,1}^out
  return T[r]
```

◇

$\mathcal{L}_{\text{samp-L}}$

```
SAMP():
  r ← {0,1}^λ
  return r
```

$\Rightarrow$

```
CTXT(m):
  r ← SAMP()
  z := LOOKUP(r)
  x := z ⊕ m
  return (r, x)
```

◇

$\mathcal{L}^{F}_{\text{prf-rand}}$

```
T := empty

LOOKUP(r):
  if T[r] undefined:
    T[r] ← {0,1}^out
  return T[r]
```

◇

$\mathcal{L}_{\text{samp-R}}$

```
R := ∅

SAMP():
  r ← {0,1}^λ \ R
  R := R ∪ {r}
  return r
```

CPA-Secure Encryption Based On PRFs

Claim This scheme has CPA$ security (and therefore CPA security) if F is a secure PRF.

Proof We prove that $\mathcal{L}^{\Sigma}_{\text{cpa\$-real}} \approx \mathcal{L}^{\Sigma}_{\text{cpa\$-rand}}$ using the hybrid technique

$\underline{\textsc{ctxt}(m)\text{:}}$
$r \leftarrow \boxed{\textsc{samp}()}$
$z := \textsc{lookup}(r)$
$x := z \oplus m$
return (r, x)

$\diamond$

$\mathcal{L}^{F}_{\text{prf-rand}}$
$T :=$ empty
$\underline{\textsc{lookup}(r)\text{:}}$
if $T[r]$ undefined:
$T[r] \leftarrow \{0,1\}^{out}$
return $T[r]$

$\diamond$

$\mathcal{L}_{\text{samp-R}}$
$R := \emptyset$
$\underline{\textsc{samp}()\text{:}}$
$r \leftarrow \{0,1\}^{\lambda} \setminus R$
$R := R \cup \{r\}$
return r

$\underline{\textsc{ctxt}(m)\text{:}}$
$r \leftarrow \textsc{samp}()$
$z := \textsc{lookup}(r)$
$x := z \oplus m$
return (r, x)

$\diamond$

$\underline{\textsc{lookup}(r)\text{:}}$
$t \leftarrow \{0,1\}^{out}$
return t

$\diamond$

$\mathcal{L}_{\text{samp-R}}$
$R := \emptyset$
$\underline{\textsc{samp}()\text{:}}$
$r \leftarrow \{0,1\}^{\lambda} \setminus R$
$R := R \cup \{r\}$
return r

Recall that our goal is to arrive at a hybrid in which the outputs of CTXT are chosen uniformly. These outputs include the value r , but now r is no longer being chosen uniformly!

CPA-Secure Encryption Based On PRFs

Claim This scheme has CPA\$ security (and therefore CPA security) if F is a secure PRF.

Proof We prove that $\mathcal{L}^{\Sigma}_{\text{cpa\$-real}} \approx \mathcal{L}^{\Sigma}_{\text{cpa\$-rand}}$ using the hybrid technique

$$
\boxed{\begin{array}{l}
\underline{\textsc{ctxt}(m):} \\
\quad r \leftarrow \textsc{samp}() \\
\quad z := \textsc{lookup}(r) \\
\quad x := z \oplus m \\
\quad \text{return } (r, x)
\end{array}}
\diamond
\boxed{\begin{array}{l}
\underline{\textsc{lookup}(r):} \\
\quad t \leftarrow \{0,1\}^{out} \\
\quad \text{return } t
\end{array}}
\diamond
\boxed{\begin{array}{l}
\mathcal{L}_{\text{samp-R}} \\
\hline
R := \emptyset \\
\\
\underline{\textsc{samp}():} \\
\quad r \leftarrow \{0,1\}^{\lambda} \setminus R \\
\quad R := R \cup \{r\} \\
\quad \text{return } r
\end{array}}
\Rightarrow
\boxed{\begin{array}{l}
\underline{\textsc{ctxt}(m):} \\
\quad r \leftarrow \textsc{samp}() \\
\quad z := \textsc{lookup}(r) \\
\quad x := z \oplus m \\
\quad \text{return } (r, x)
\end{array}}
\diamond
\boxed{\begin{array}{l}
\underline{\textsc{lookup}(r):} \\
\quad t \leftarrow \{0,1\}^{out} \\
\quad \text{return } t
\end{array}}
\diamond
\boxed{\begin{array}{l}
\mathcal{L}_{\text{samp-L}} \\
\hline
\underline{\textsc{samp}():} \\
\quad r \leftarrow \{0,1\}^{\lambda} \\
\quad \text{return } r
\end{array}}
$$

CPA-Secure Encryption Based On PRFs

Claim This scheme has CPA$ security (and therefore CPA security) if F is a secure PRF.

Proof We prove that $\mathcal{L}^{\Sigma}_{\text{cpa\$-real}} \approx \mathcal{L}^{\Sigma}_{\text{cpa\$-rand}}$ using the hybrid technique

$$\boxed{\begin{array}{l} \underline{\textsc{ctxt}(m):} \\ \quad r \leftarrow \textsc{samp}() \\ \quad z := \textsc{lookup}(r) \\ \quad x := z \oplus m \\ \quad \text{return } (r, x) \end{array}} \diamond \boxed{\begin{array}{l} \underline{\textsc{lookup}(r):} \\ \quad t \leftarrow \{0,1\}^{out} \\ \quad \text{return } t \end{array}} \diamond \boxed{\begin{array}{l} \mathcal{L}_{\text{samp-L}} \\ \hline \underline{\textsc{samp}():} \\ \quad r \leftarrow \{0,1\}^{\lambda} \\ \quad \text{return } r \end{array}} \Rightarrow \boxed{\begin{array}{l} \underline{\textsc{ctxt}(m):} \\ \quad r \leftarrow \{0,1\}^{\lambda} \\ \quad z \leftarrow \{0,1\}^{out} \\ \quad x := z \oplus m \\ \quad \text{return } (r, x) \end{array}}$$

CPA-Secure Encryption Based On PRFs

Claim This scheme has CPA$ security (and therefore CPA security) if F is a secure PRF.

Proof We prove that $\mathcal{L}^{\Sigma}_{\text{cpa\$-real}} \approx \mathcal{L}^{\Sigma}_{\text{cpa\$-rand}}$ using the hybrid technique

$\text{CTXT}(m)$:
- $r \leftarrow \{0,1\}^{\lambda}$
- $z \leftarrow \{0,1\}^{out}$
- $x := z \oplus m$
- return (r, x)

$\Rightarrow$

$\mathcal{L}^{\Sigma}_{\text{cpa\$-rand}}$

$\text{CTXT}(m)$:
- $r \leftarrow \{0,1\}^{\lambda}$
- $x \leftarrow \{0,1\}^{out}$
- return (r, x)

Block Cipher Modes of Operation

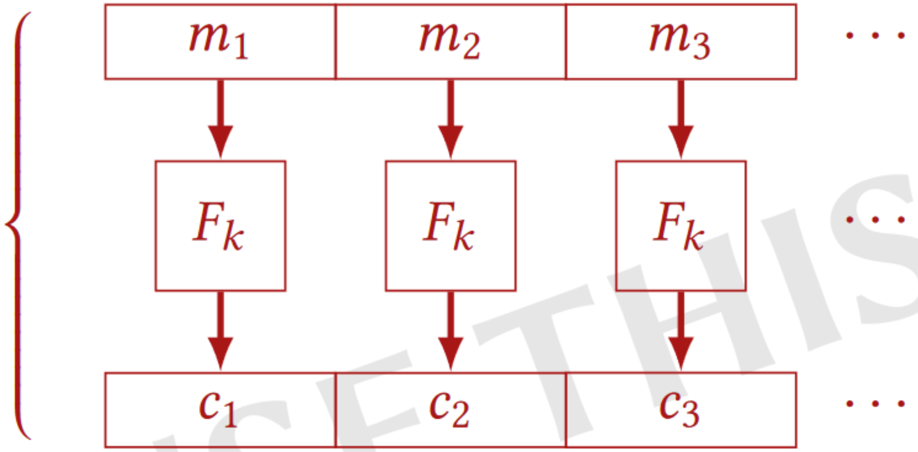
Block Cipher Modes of Operation

- One of the drawbacks of the previous CPA-secure encryption scheme is that its ciphertexts are λ bits longer than its plaintexts.
- Is there any way to encrypt data (especially lots of it) without requiring such a significant overhead?

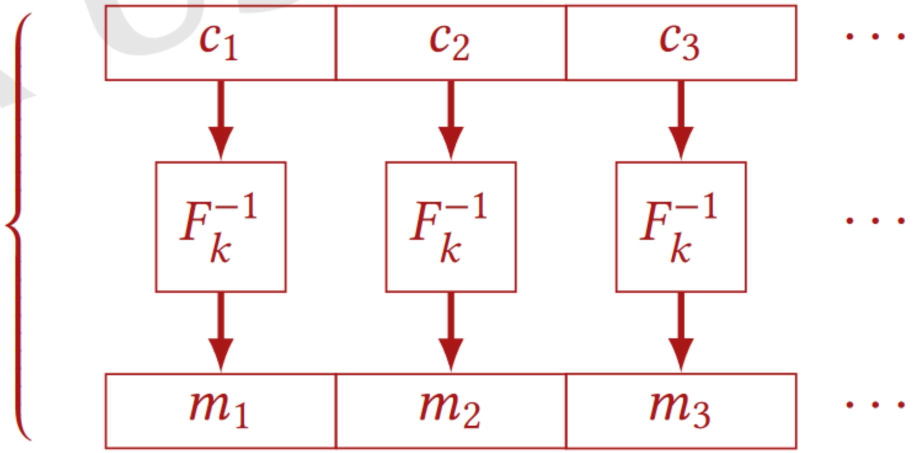
- $blen$ will denote the blocklength of a block cipher F. We'll write F_k as shorthand for $F(k,\cdot)$.
- When m is the plaintext, we will write $m = m_1 \| m_2 \| \cdots \| m_\ell$, where each m_i is a single block

ECB: Electronic Codebook (never never use this! Never!!)

$\mathrm{Enc}(k, m_1 \| \cdots \| m_\ell)$:
- for $i = 1$ to ℓ:
  - $c_i := F(k, m_i)$
- return $c_1 \| \cdots \| c_\ell$



$\mathrm{Dec}(k, c_1 \| \cdots \| c_\ell)$:
- for $i = 1$ to ℓ:
  - $m_i := F^{-1}(k, c_i)$
- return $m_1 \| \cdots \| m_\ell$



NEVER USE THIS!

CBC: Cipher Block Chaining

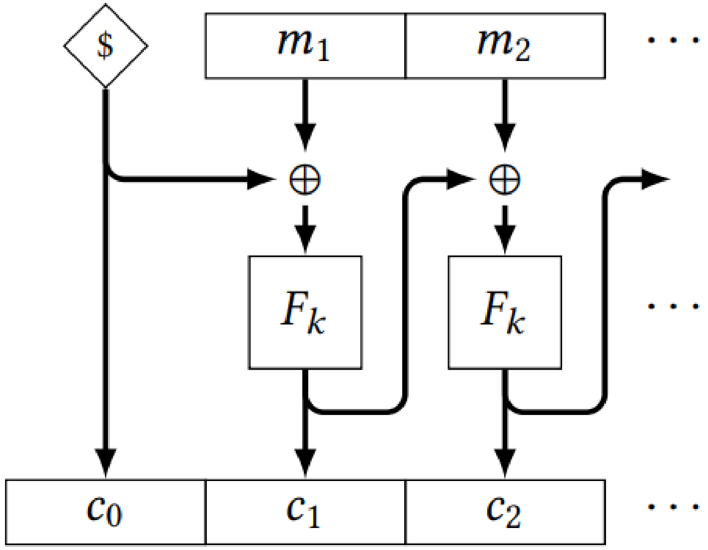
$\underline{\mathsf{Enc}(k, m_1 \| \cdots \| m_\ell):}$

$c_0 \leftarrow \{\texttt{0},\texttt{1}\}^{blen}$:

for $i = 1$ to ℓ:

  $c_i := F(k, m_i \oplus c_{i-1})$

return $c_0 \| c_1 \| \cdots \| c_\ell$

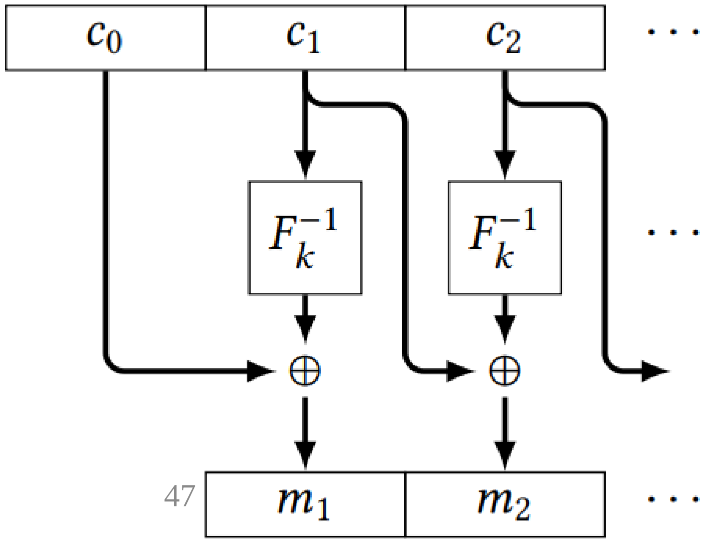


$\underline{\mathsf{Dec}(k, c_0 \| \cdots \| c_\ell):}$

for $i = 1$ to ℓ:

  $m_i := F^{-1}(k, c_i) \oplus c_{i-1}$

return $m_1 \| \cdots \| m_\ell$



CBC: Cipher Block Chaining

- The most common mode in practice.
- The CBC encryption of an ℓ-block plaintext is $\ell + 1$ blocks long.
- The first ciphertext block is called an initialization vector (IV).
- Here we have described CBC mode as a randomized encryption, with the IV of each ciphertext being chosen uniformly.
- CBC mode provides CPA security.

CTR: Counter

- The next most common mode in practice is counter mode.
- Just like CBC mode, it involves an additional IV block r that is chosen uniformly. The idea is to then use the sequence
$$F(k, r);\ F(k, r+1);\ F(k, r+2); \cdots$$

CTR: Counter

$\underline{\mathsf{Enc}(k, m_1 \| \cdots \| m_\ell):}$

$r \leftarrow \{0,1\}^{blen}$

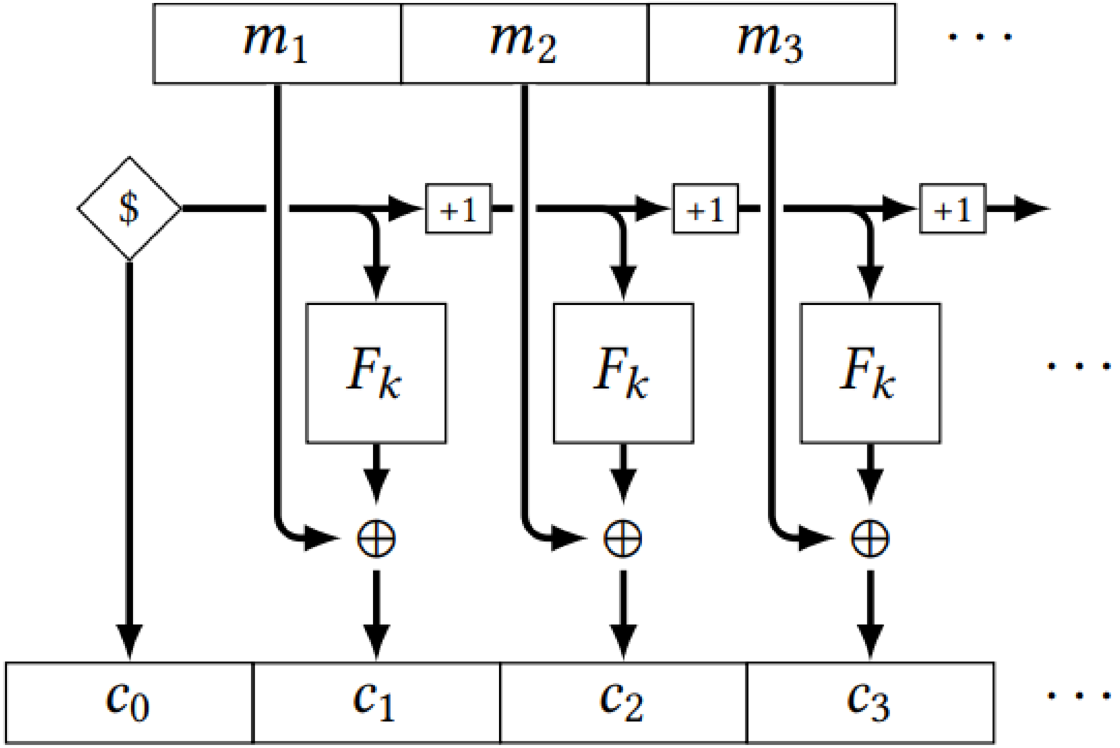
$c_0 := r$

for $i = 1$ to ℓ:

  $c_i := F(k, r) \oplus m_i$

  $r := r + 1 \% 2^{blen}$

return $c_0 \| \cdots \| c_\ell$



OFB: Output Feedback

- OFB (output feedback) mode is rarely used in practice. But it has the easiest security proof.

$\underline{\mathsf{Enc}(k, m_1 \| \cdots \| m_\ell):}$

$r \leftarrow \{0,1\}^{blen}$

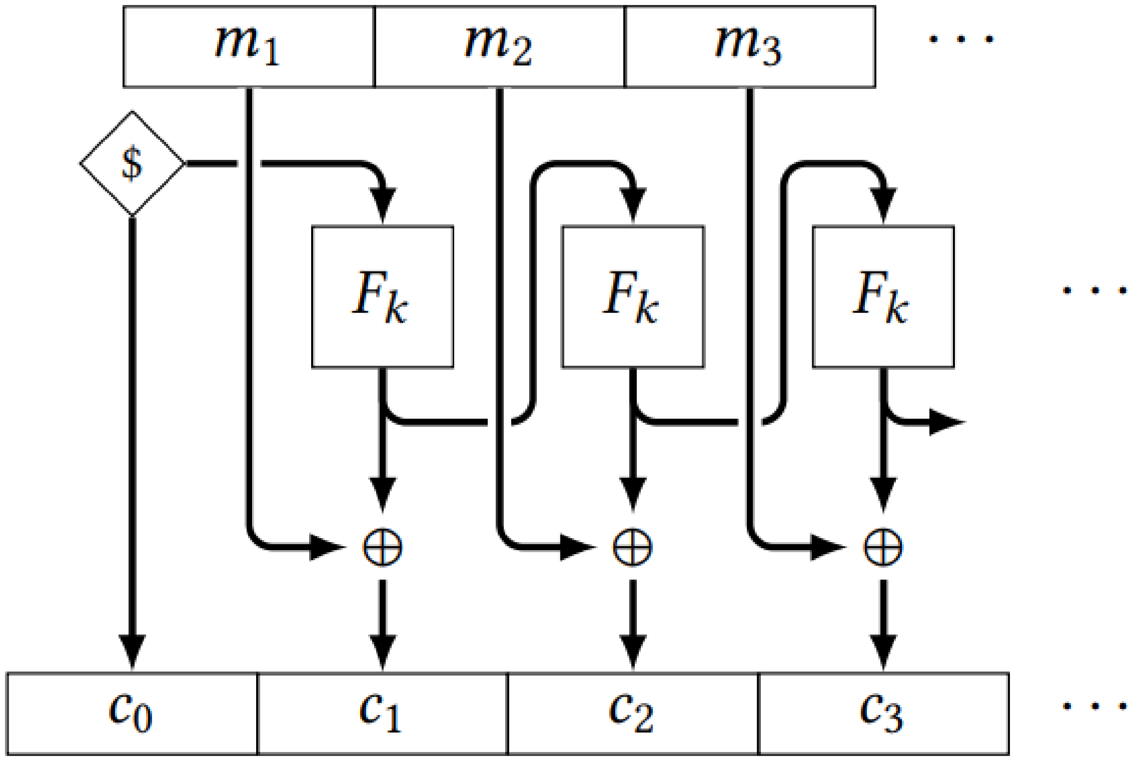
$c_0 := r$

for $i = 1$ to ℓ:

  $r := F(k, r)$

  $c_i := r \oplus m_i$

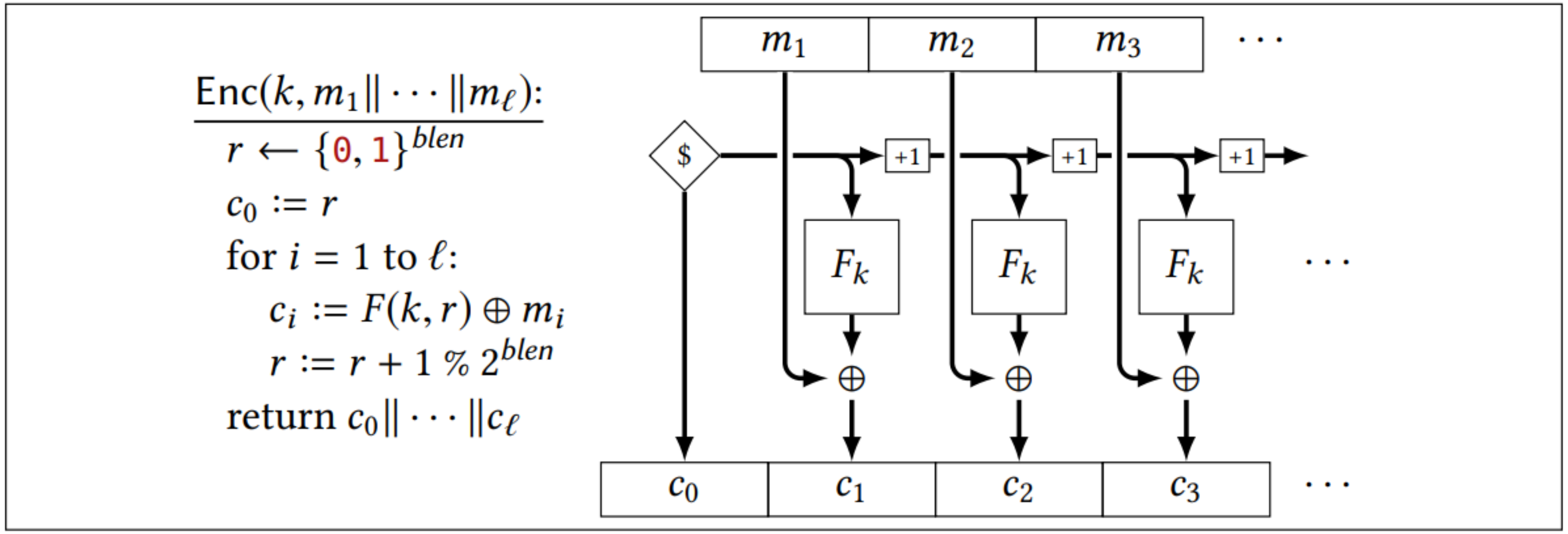
return $c_0 \| \cdots \| c_\ell$



Compare & Contrast

CBC and CTR modes are essentially the only two modes that are ever considered in practice for CPA security. Both provide the same security guarantees.

- CTR mode does not even use the block cipher's inverse F^{-1}. CTR mode can be instantiated from a PRF; it doesn't need a PRP. However, in practice it is rare to encounter an efficient PRF that is not a PRP.



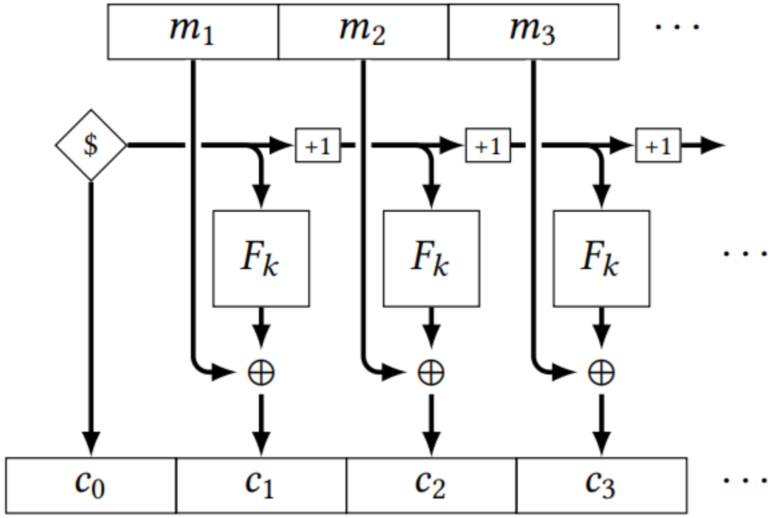
Compare & Contrast

CBC and CTR modes are essentially the only two modes that are ever considered in practice for CPA security. Both provide the same security guarantees.

- CTR mode encryption can be parallelized. CBC mode cannot.
- If calls to the block cipher are expensive, it might be desirable to pre-compute and store them before the plaintext is known. CTR mode can, CBC mode cannot.

$\mathsf{Enc}(k, m_1 \| \cdots \| m_\ell)$:

$r \leftarrow \{0,1\}^{blen}$
$c_0 := r$
for $i = 1$ to ℓ:
  $c_i := F(k, r) \oplus m_i$
  $r := r + 1 \,\%\, 2^{blen}$
return $c_0 \| \cdots \| c_\ell$

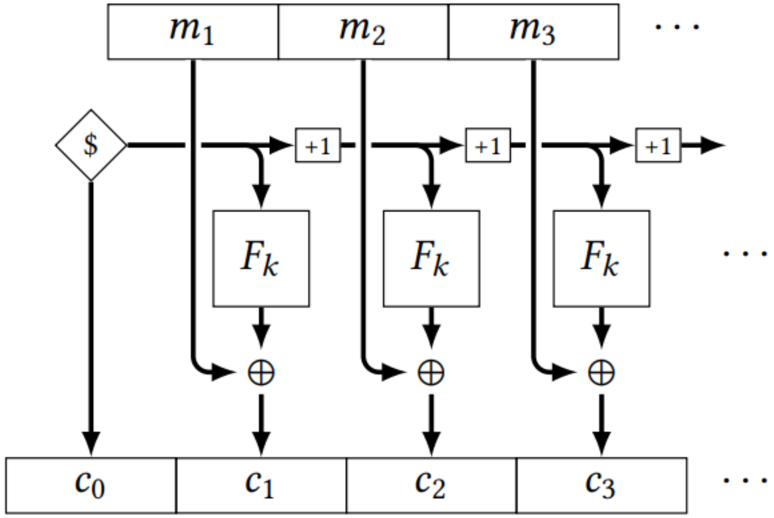


Compare & Contrast

CBC and CTR modes are essentially the only two modes that are ever considered in practice for CPA security. Both provide the same security guarantees.

- It is relatively easy to modify CTR to support plaintexts that are not an exact multiple of the blocklength.

$\underline{\mathsf{Enc}(k, m_1 \| \cdots \| m_\ell):}$
$r \leftarrow \{0,1\}^{blen}$
$c_0 := r$
for $i = 1$ to ℓ:
  $c_i := F(k, r) \oplus m_i$
  $r := r + 1 \,\%\, 2^{blen}$
return $c_0 \| \cdots \| c_\ell$



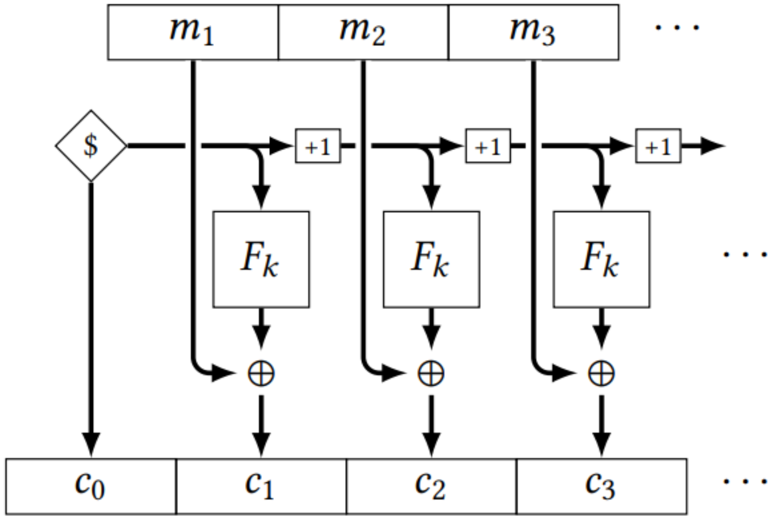
Compare & Contrast

CBC and CTR modes are essentially the only two modes that are ever considered in practice for CPA security. Both provide the same security guarantees.

- It is common for implementers to misunderstand the security implications of the IV in these modes. Many careless implementations allow an IV to be reused. The effects of IV-reuse in CTR mode are quite devastating to message privacy.

$\underline{\text{Enc}(k, m_1 \| \cdots \| m_\ell)\text{:}}$

$r \leftarrow \{0,1\}^{blen}$
$c_0 := r$
for $i = 1$ to ℓ:
  $c_i := F(k, r) \oplus m_i$
  $r := r + 1 \,\%\, 2^{blen}$
return $c_0 \| \cdots \| c_\ell$



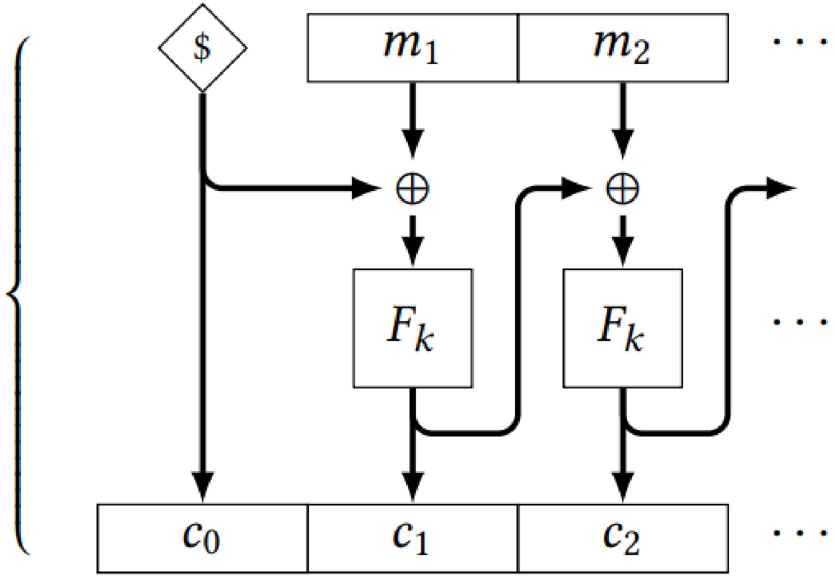
Compare & Contrast

CBC and CTR modes are essentially the only two modes that are ever considered in practice for CPA security. Both provide the same security guarantees.

- In CBC mode, reusing an IV can actually be safe, if the two plaintexts have different first blocks!

$\underline{\mathsf{Enc}(k, m_1 \| \cdots \| m_\ell):}$

$c_0 \leftarrow \{0,1\}^{blen}$;
for $i = 1$ to ℓ:
  $c_i := F(k, m_i \oplus c_{i-1})$
return $c_0 \| c_1 \| \cdots \| c_\ell$



CPA Security and Variable-Length Plaintexts

- In CBC mode, a plaintext consisting of ℓ blocks is encrypted into a ciphertext of $\ell + 1$ blocks. In other words, the ciphertext leaks the number of blocks in the plaintext.
- Cannot achieve the CPA security we have defined.

$\mathcal{L}^{\Sigma}_{\text{cpa-L}}$	$\mathcal{L}^{\Sigma}_{\text{cpa-R}}$
$k \leftarrow \Sigma.\text{KeyGen}$	$k \leftarrow \Sigma.\text{KeyGen}$
$\text{EAVESDROP}(m_L, m_R \in \Sigma.\mathcal{M})$:	$\text{EAVESDROP}(m_L, m_R \in \Sigma.\mathcal{M})$:
$c := \Sigma.\text{Enc}(k, m_L)$	$c := \Sigma.\text{Enc}(k, m_R)$
return c	return c

CPA Security and Variable-Length Plaintexts

- Suppose we don't really care about hiding the length of plaintexts. Is there a way to make a security definition that says: *ciphertexts hide everything about the plaintext, except their length*?
- When discussing encryption schemes that support variable-length plaintexts, CPA security will refer to the following updated libraries.

$\mathcal{L}^{\Sigma}_{\text{cpa-L}}$
$k \leftarrow \Sigma.\text{KeyGen}$
$\underline{\text{CTXT}(m_L, m_R \in \Sigma.\mathcal{M}):}$
if $\lvert m_L \rvert \neq \lvert m_R \rvert$ return err
$c := \Sigma.\text{Enc}(k, m_L)$
return c

$\mathcal{L}^{\Sigma}_{\text{cpa-R}}$
$k \leftarrow \Sigma.\text{KeyGen}$
$\underline{\text{CTXT}(m_L, m_R \in \Sigma.\mathcal{M}):}$
if $\lvert m_L \rvert \neq \lvert m_R \rvert$ return err
$c := \Sigma.\text{Enc}(k, m_R)$
return c

CPA Security and Variable-Length Plaintexts

- Then when discussing encryption schemes supporting variable-length plaintexts, CPA$ security will refer to the following libraries:

$\mathcal{L}^{\Sigma}_{\text{cpa\$-real}}$

$k \leftarrow \Sigma.\text{KeyGen}$

$\textsc{challenge}(m \in \Sigma.\mathcal{M})$:

$\quad c := \Sigma.\text{Enc}(k, m)$

$\quad \text{return } c$

$\mathcal{L}^{\Sigma}_{\text{cpa\$-rand}}$

$\textsc{challenge}(m \in \Sigma.\mathcal{M})$:

$\quad c \leftarrow \Sigma.\mathcal{C}(|m|)$

$\quad \text{return } c$

- With respect to these updated security definitions, CPA$ security implies CPA security as before.

Don’t Take Length-Leaking for Granted!

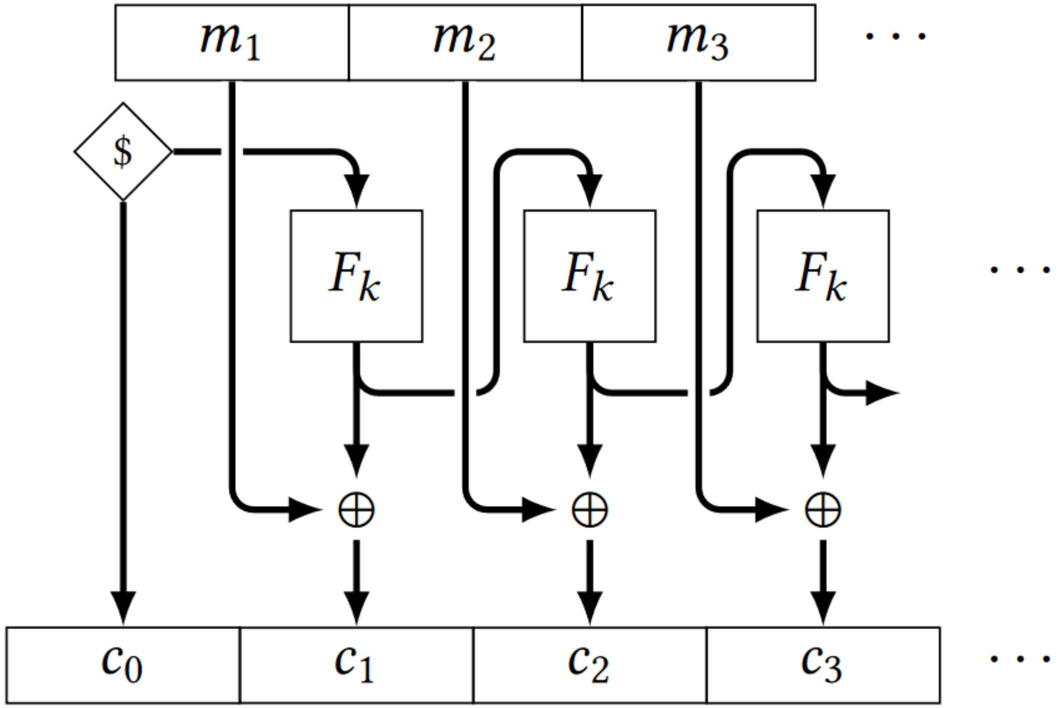
- We have just gone from requiring encryption to leak no partial information to casually allowing some specific information to leak.
- If we want to truly support plaintexts of arbitrary length, then leaking the length is in fact unavoidable.
- By observing only the length of encrypted network traffic, many serious attacks are possible.
    - Google maps
    - Variable-bit-rate (VBR) :the changes in bit rate are reflected as changes in packet length
        - Netflix, Youtube…
        - Voice chat programs (who was speaking, the language being spoken)

Security of OFB Mode

Claim OFB mode has CPA$ security, if its underlying block cipher F is a secure PRF (no need for PRP) with parameters $in = out = \lambda$.

$\underline{\text{Enc}(k, m_1 \| \cdots \| m_\ell):}$

$r \leftarrow \{0,1\}^{blen}$
$c_0 := r$
for $i = 1$ to ℓ:
  $r := F(k, r)$
  $c_i := r \oplus m_i$
return $c_0 \| \cdots \| c_\ell$



Security of OFB Mode

Claim OFB mode has CPA$ security, if its underlying block cipher F is a secure PRF (no need for PRP) with parameters $in = out = \lambda$.

proof

- Each ciphertext block (apart from the IV) is computed as $c_i := r \oplus m_i$. By the one-time pad rule, it suffices to show that the r values are independently pseudorandom.
- Each r value is the result of a call to the PRF. These PRF outputs will be independently pseudorandom only if all of the inputs to the PRF are distinct.

Security of OFB Mode

Claim OFB mode has CPA$ security, if its underlying block cipher F is a secure PRF (no need for PRP) with parameters $in = out = \lambda$.

proof

$\mathcal{L}^{\text{OFB}}_{\text{cpa\$-real}}$

$k \leftarrow \{0,1\}^\lambda$

$\underline{\text{CTXT}(m_1 \| \cdots \| m_\ell):}$

$\quad r \leftarrow \{0,1\}^\lambda$

$\quad c_0 := r$

$\quad \text{for } i = 1 \text{ to } \ell:$

$\quad\quad r := F(k, r)$

$\quad\quad c_i := r \oplus m_i$

$\quad \text{return } c_0 \| c_1 \| \cdots \| c_\ell$

$\Rightarrow$

$\underline{\text{CTXT}(m_1 \| \cdots \| m_\ell):}$

$\quad r \leftarrow \{0,1\}^\lambda$

$\quad c_0 := r$

$\quad \text{for } i = 1 \text{ to } \ell:$

$\quad\quad r := \text{LOOKUP}(r)$

$\quad\quad c_i := r \oplus m_i$

$\quad \text{return } c_0 \| c_1 \| \cdots \| c_\ell$

$\diamond$

$\mathcal{L}^{F}_{\text{prf-real}}$

$k \leftarrow \{0,1\}^\lambda$

$\underline{\text{LOOKUP}(r):}$

$\quad \text{return } F(k, r)$

Security of OFB Mode

Claim OFB mode has CPA$ security, if its underlying block cipher F is a secure PRF (no need for PRP) with parameters $in = out = \lambda$.

proof

$\textsc{ctxt}(m_1 \| \cdots \| m_\ell)$:
- $r \leftarrow \{0,1\}^\lambda$
- $c_0 := r$
- for $i = 1$ to ℓ:
  - $r := \textsc{lookup}(r)$
  - $c_i := r \oplus m_i$
- return $c_0 \| c_1 \| \cdots \| c_\ell$

$\diamond$

$\mathcal{L}^F_{\text{prf-real}}$
- $k \leftarrow \{0,1\}^\lambda$

$\textsc{lookup}(r)$:
- return $F(k, r)$

$\Rightarrow$

$\textsc{ctxt}(m_1 \| \cdots \| m_\ell)$:
- $r \leftarrow \{0,1\}^\lambda$
- $c_0 := r$
- for $i = 1$ to ℓ:
  - $r := \textsc{lookup}(r)$
  - $c_i := r \oplus m_i$
- return $c_0 \| c_1 \| \cdots \| c_\ell$

$\diamond$

$\mathcal{L}^F_{\text{prf-rand}}$
- $T :=$ empty

$\textsc{lookup}(x)$:
- if $T[x]$ undefined:
  - $T[x] \leftarrow \{0,1\}^\lambda$
- return $T[x]$

Security of OFB Mode

Claim OFB mode has CPA$ security, if its underlying block cipher F is a secure PRF (no need for PRP) with parameters $in = out = \lambda$.

proof

$\textsc{ctxt}(m_1 \| \cdots \| m_\ell)$:
- $r \leftarrow \{0,1\}^\lambda$
- $c_0 := r$
- for $i = 1$ to ℓ:
  - $r := \textsc{lookup}(r)$
  - $c_i := r \oplus m_i$
- return $c_0 \| c_1 \| \cdots \| c_\ell$

$\diamond$

$\mathcal{L}^F_{\text{prf-rand}}$

$T :=$ empty

$\textsc{lookup}(x)$:
- if $T[x]$ undefined:
  - $T[x] \leftarrow \{0,1\}^\lambda$
- return $T[x]$

$\textsc{challenge}(m_1 \| \cdots \| m_\ell)$:
- $r := \textsc{samp}()$
- $c_0 := r$
- for $i = 1$ to ℓ:
  - $r := \textsc{lookup}(r)$
  - $c_i := r \oplus m_i$
- return $c_0 \| c_1 \| \cdots \| c_\ell$

$\diamond$

$T :=$ empty

$\textsc{lookup}(x)$:
- if $T[x]$ undefined:
  - $T[x] := \textsc{samp}()$
- return $T[x]$

$\diamond$

$\mathcal{L}_{\text{samp-L}}$

$\textsc{samp}()$:
- $r \leftarrow \{0,1\}^\lambda$
- return r

Security of OFB Mode

Claim OFB mode has CPA$ security, if its underlying block cipher F is a secure PRF (no need for PRP) with parameters $in = out = \lambda$.

```
CHALLENGE(m_1||···||m_ℓ):
  r := SAMP()
  c_0 := r
  for i = 1 to ℓ:
    r := LOOKUP(r)
    c_i := r ⊕ m_i
  return c_0||c_1||···||c_ℓ
```

$\diamond$

```
T := empty

LOOKUP(x):
  if T[x] undefined:
    T[x] := SAMP()
  return T[x]
```

$\diamond$

```
L_samp-L

SAMP():
  r ← {0,1}^λ
  return r
```

```
CTXT(m_1||···||m_ℓ):
  r := SAMP()
  c_0 := r
  for i = 1 to ℓ:
    r := LOOKUP(r)
    c_i := r ⊕ m_i
  return c_0||c_1||···||c_ℓ
```

$\diamond$

```
T := empty

LOOKUP(x):
  if T[x] undefined:
    T[x] := SAMP()
  return T[x]
```

$\diamond$

```
L_samp-R

R := ∅

SAMP():
  r ← {0,1}^λ \ R
  R := R ∪ {r}
  return r
```

Security of OFB Mode

Claim OFB mode has CPA$ security, if its underlying block cipher F is a secure PRF (no need for PRP) with parameters $in = out = \lambda$.

$\textsc{ctxt}(m_1\|\cdots\|m_\ell)$:
- $r := \textsc{samp}()$
- $c_0 := r$
- for $i = 1$ to ℓ:
  - $r := \textsc{lookup}(r)$
  - $c_i := r \oplus m_i$
- return $c_0\|c_1\|\cdots\|c_\ell$

$\diamond$

$T :=$ empty

$\textsc{lookup}(x)$:
- if $T[x]$ undefined:
  - $T[x] := \textsc{samp}()$
- return $T[x]$

$\diamond$

$\mathcal{L}_{\text{samp-R}}$

$R := \emptyset$

$\textsc{samp}()$:
- $r \leftarrow \{0,1\}^\lambda \setminus R$
- $R := R \cup \{r\}$
- return r

$\Rightarrow$

$\textsc{ctxt}(m_1\|\cdots\|m_\ell)$:
- $r := \textsc{samp}()$
- $c_0 := r$
- for $i = 1$ to ℓ:
  - $r := \textsc{lookup}(r)$
  - $c_i := r \oplus m_i$
- return $c_0\|c_1\|\cdots\|c_\ell$

$\diamond$

$\textsc{lookup}(x)$:
- $t := \textsc{samp}()$
- return t

$\diamond$

$\mathcal{L}_{\text{samp-R}}$

$R := \emptyset$

$\textsc{samp}()$:
- $r \leftarrow \{0,1\}^\lambda \setminus R$
- $R := R \cup \{r\}$
- return r

Security of OFB Mode

Claim OFB mode has CPA$ security, if its underlying block cipher F is a secure PRF (no need for PRP) with parameters $in = out = \lambda$.

$\textsc{ctxt}(m_1 \| \cdots \| m_\ell)$:
$\quad r := \textsc{samp}()$
$\quad c_0 := r$
$\quad$ for $i = 1$ to ℓ:
$\quad\quad r := \textsc{lookup}(r)$
$\quad\quad c_i := r \oplus m_i$
$\quad$ return $c_0 \| c_1 \| \cdots \| c_\ell$

$\diamond$

$\textsc{lookup}(x)$:
$\quad t := \textsc{samp}()$
$\quad$ return t

$\diamond$

$\mathcal{L}_{\text{samp-R}}$

$R := \emptyset$

$\textsc{samp}()$:
$\quad r \leftarrow \{0,1\}^\lambda \setminus R$
$\quad R := R \cup \{r\}$
$\quad$ return r

$\textsc{ctxt}(m_1 \| \cdots \| m_\ell)$:
$\quad r := \textsc{samp}()$
$\quad c_0 := r$
$\quad$ for $i = 1$ to ℓ:
$\quad\quad r := \textsc{lookup}(r)$
$\quad\quad c_i := r \oplus m_i$
$\quad$ return $c_0 \| c_1 \| \cdots \| c_\ell$

$\diamond$

$\textsc{lookup}(x)$:
$\quad t := \textsc{samp}()$
$\quad$ return t

$\diamond$

$\mathcal{L}_{\text{samp-L}}$

$\textsc{samp}()$:
$\quad r \leftarrow \{0,1\}^\lambda$
$\quad$ return r

Security of OFB Mode

Claim OFB mode has CPA$ security, if its underlying block cipher F is a secure PRF (no need for PRP) with parameters $in = out = \lambda$.

proof

$\textsc{ctxt}(m_1 \| \cdots \| m_\ell)$:

- $r := \textsc{samp}()$
- $c_0 := r$
- for $i = 1$ to ℓ:
  - $r := \textsc{lookup}(r)$
  - $c_i := r \oplus m_i$
- return $c_0 \| c_1 \| \cdots \| c_\ell$

$\diamond$

$\textsc{lookup}(x)$:

- $t := \textsc{samp}()$
- return t

$\diamond$

$\mathcal{L}_{\text{samp-L}}$

$\textsc{samp}()$:

- $r \leftarrow \{0,1\}^\lambda$
- return r

$\Rightarrow$

$\textsc{ctxt}(m_1 \| \cdots \| m_\ell)$:

- $r \leftarrow \{0,1\}^\lambda$
- $c_0 := r$
- for $i = 1$ to ℓ:
  - $r \leftarrow \{0,1\}^\lambda$
  - $c_i := r \oplus m_i$
- return $c_0 \| c_1 \| \cdots \| c_\ell$

Security of OFB Mode

Claim OFB mode has CPA$ security, if its underlying block cipher F is a secure PRF (no need for PRP) with parameters $in = out = \lambda$.

proof

$\underline{\textsc{ctxt}(m_1 \| \cdots \| m_\ell):}$

$\quad r \leftarrow \{0,1\}^\lambda$
$\quad c_0 := r$
$\quad$for $i = 1$ to ℓ:
$\qquad r \leftarrow \{0,1\}^\lambda$
$\qquad c_i := r \oplus m_i$
$\quad$return $c_0 \| c_1 \| \cdots \| c_\ell$

$\mathcal{L}^{\text{OFB}}_{\text{cpa\$-rand}}$

$\underline{\textsc{ctxt}(m_1 \| \cdots \| m_\ell):}$

$\quad$for $i = 0$ to ℓ:
$\qquad c_i \leftarrow \{0,1\}^\lambda$
$\quad$return $c_0 \| c_1 \| \cdots \| c_\ell$

Padding & Ciphertext Stealing

- So far we have assumed that all plaintexts are exact multiples of the blocklength.
- How are block ciphers used in practice with data that has arbitrary length?

- Padding
- Ciphertext Stealing

Padding

- Encode arbitrary-length data into data that is a multiple of the blocklength.
- A padding scheme should consist of two algorithms:
  - pad: takes as input a string of any length, and outputs a string whose length is a multiple of the blocklength
  - unpad: the inverse of pad. We require that $\text{unpad}(\text{pad}(x)) = x$ for all strings x.

- In the real world, data almost always comes in bytes and not bits
- Padding schemes are not a security feature!

Padding

- Null padding
  - Fill the final block with null bytes (00)
  - It is not always reversible
  - pad(31 41 59) and pad(31 41 59 00) will give the same result

Padding

- ANSI X.923 standard
  - Data is padded with null bytes, except for the last byte of padding which indicates how many padding bytes there are. (tell the receiver how many bytes to remove to recover the original message)
  - If the original unpadded data is already a multiple of the block length, then an entire extra block of padding must be added.

01	34	11	d9	81	88	05	57	1d	73	c3	00	00	00	00	05	⇒ *valid*
95	51	05	4a	d6	5a	a3	44	af	b3	85	00	00	00	00	03	⇒ *valid*
71	da	77	5a	5e	77	eb	a8	73	c5	50	b5	81	d5	96	01	⇒ *valid*
5b	1c	01	41	5d	53	86	4e	e4	94	13	e8	7a	89	c4	71	⇒ *invalid*
d4	0d	d8	7b	53	24	c6	d1	af	5f	d6	f6	00	c0	00	04	⇒ *invalid*

Padding

- PKCS#7 standard
  - If *b* bytes of padding are needed, then the data is padded not with null bytes but with b bytes.

01	34	11	d9	81	88	05	57	1d	73	c3	**05**	**05**	**05**	**05**	**05**	$\Rightarrow$	*valid*
95	51	05	4a	d6	5a	a3	44	af	b3	85	03	03	**03**	**03**	**03**	$\Rightarrow$	*valid*
71	da	77	5a	5e	77	eb	a8	73	c5	50	b5	81	d5	96	**01**	$\Rightarrow$	*valid*
5b	1c	01	41	5d	53	86	4e	e4	94	13	e8	7a	89	c4	**71**	$\Rightarrow$	*invalid*
d4	0d	d8	7b	53	24	c6	d1	af	5f	d6	f6	**04**	**c0**	**04**	**04**	$\Rightarrow$	*invalid*

Padding

- ISO/IEC 7816-4 standard
  - The data is padded with a 80 byte followed by null bytes. To remove the padding, remove all trailing null bytes and ensure that the last byte is 80 (and then remove it too).
  - The significance of 80 is clearer when you write it in binary as 10000000

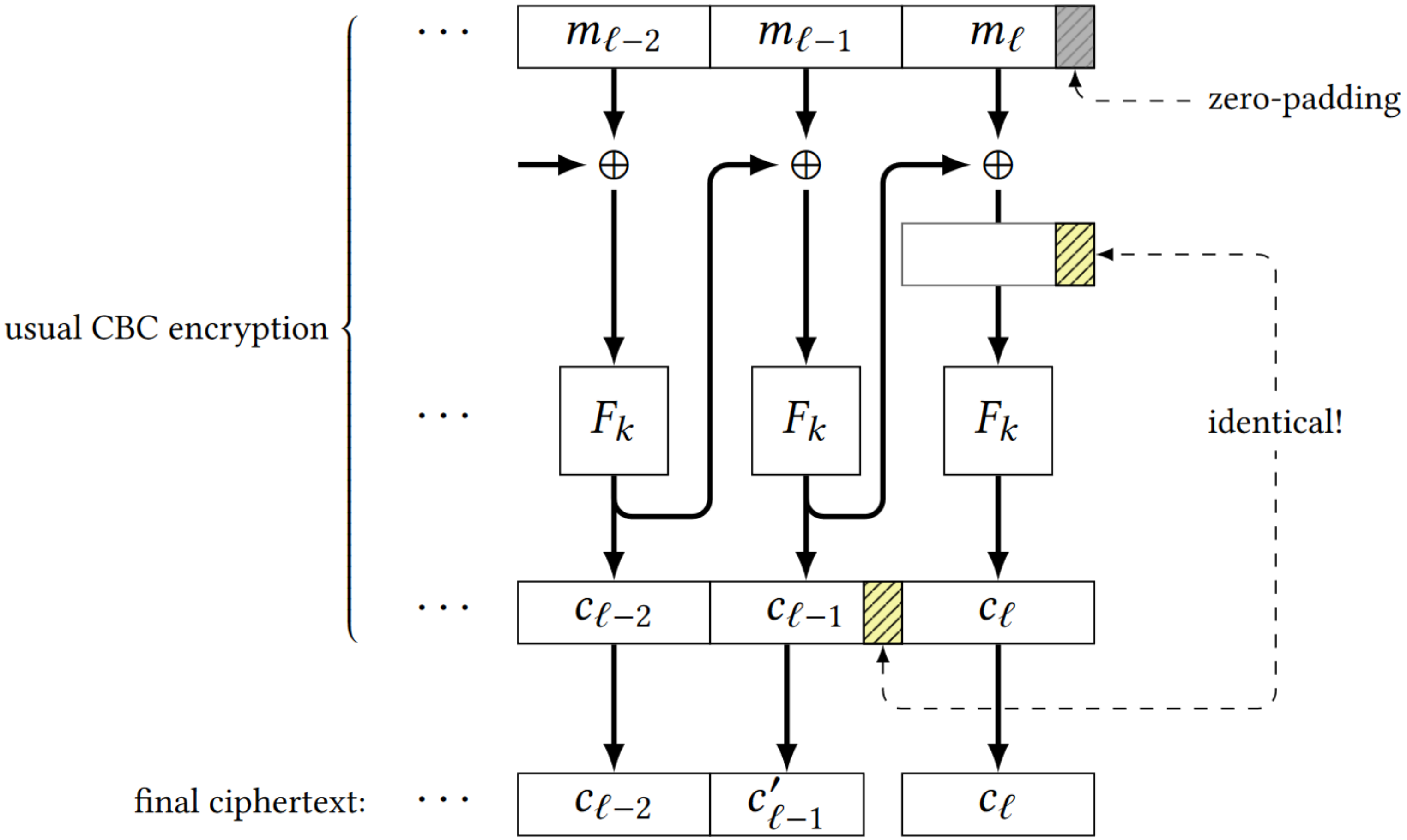
01	34	11	d9	81	88	05	57	1d	73	c3	**80**	**00**	**00**	**00**	**00**	$\Rightarrow$	*valid*
95	51	05	4a	d6	5a	a3	44	af	b3	85	03	03	**80**	**00**	**00**	$\Rightarrow$	*valid*
71	da	77	5a	5e	77	eb	a8	73	c5	50	b5	81	d5	96	**80**	$\Rightarrow$	*valid*
5b	1c	01	41	5d	53	86	4e	e4	94	13	e8	7a	89	c4	**71**	$\Rightarrow$	*invalid*
d4	0d	d8	7b	53	24	c6	d1	af	5f	d6	f6	**c4**	**00**	**00**	**00**	$\Rightarrow$	*invalid*

Ciphertext Stealing

- The main idea behind ciphertext stealing is to use a standard block-cipher mode that only supports full blocks (e.g., CBC mode), and then simply throw away some bits of the ciphertext, in such a way that decryption is still possible.

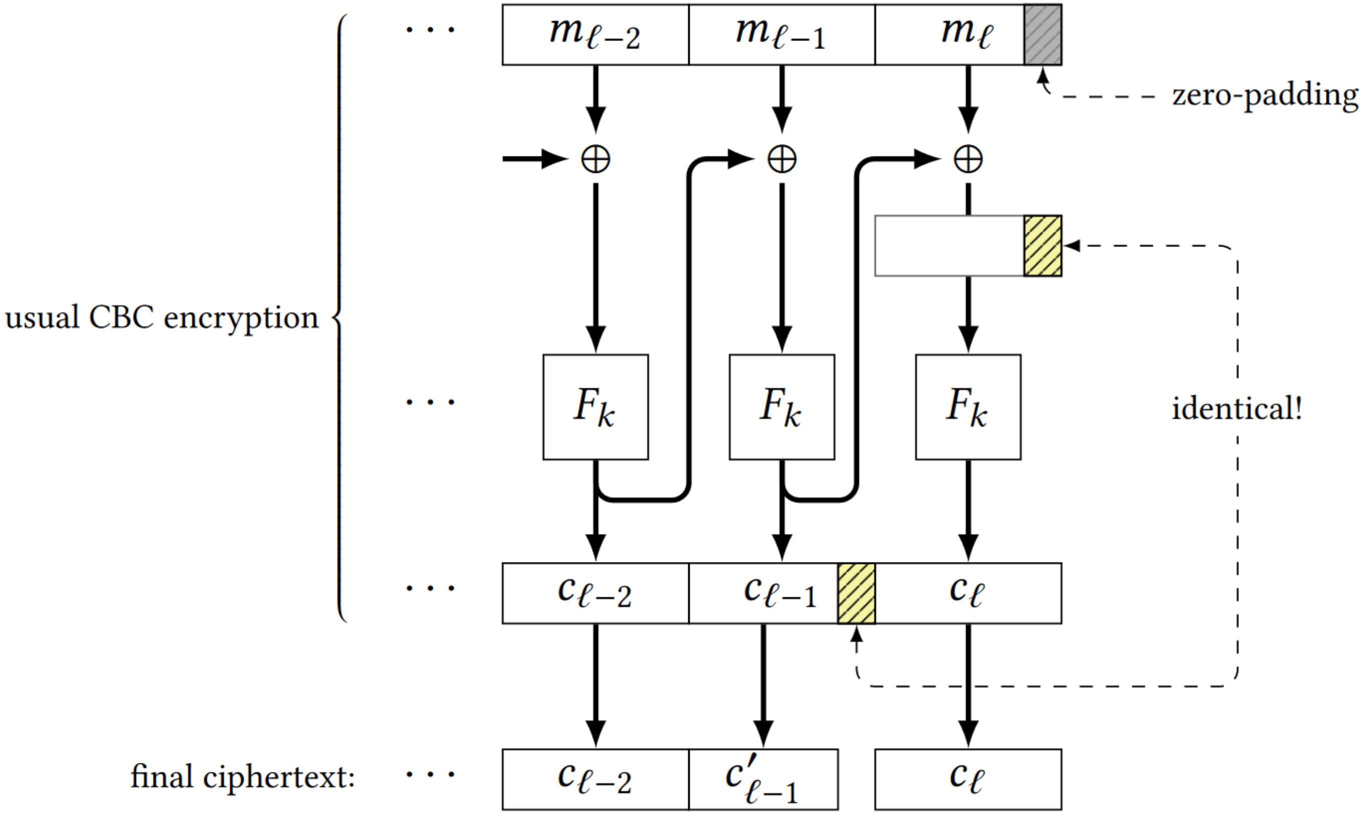
Ciphertext Stealing (for CBC mode)

- We start by extending m_ℓ with j zeroes and performing CBC as usual. Then throw away are the last j bits of $c_{\ell-1}$ (the next-to-last block of the CBC ciphertext).



Ciphertext Stealing (for CBC mode)

- For decryption: Those missing j bits were redundant, because there is another way to compute them. Let $x^* := c_{\ell-1} \oplus m_\ell$. Since the last j bits of m_ℓ are 0's, the last j bits of x^* are the last j bits of $c_{\ell-1}$ — the missing bits.



Ciphertext Stealing (for CBC mode)

- In practice, the last two blocks of the ciphertext are often interchanged.

$\underline{\text{Enc}(k, m_1 \| \cdots \| m_\ell)}$:

$\quad$ *// each m_i is blen bits,*

$\quad$ *// except possibly m_ℓ*

$\quad j := blen - |m_\ell|$

$\quad m_\ell := m_\ell \| \mathtt{0}^j$

$\quad c_0 \leftarrow \{\mathtt{0}, \mathtt{1}\}^{blen}$;

$\quad$ for $i = 1$ to ℓ:

$\quad\quad c_i := F(k, m_i \oplus c_{i-1})$

$\quad$ if $j \neq 0$:

$\quad\quad$ remove final j bits of $c_{\ell-1}$

$\quad\quad$ swap $c_{\ell-1}$ and c_ℓ

$\quad$ return $c_0 \| c_1 \| \cdots \| c_\ell$

$\underline{\text{Dec}(k, c_0 \| \cdots \| c_\ell)}$:

$\quad$ *// each c_i is blen bits,*

$\quad$ *// except possibly c_ℓ*

$\quad j := blen - |c_\ell|$

$\quad$ if $j \neq 0$:

$\quad\quad$ swap $c_{\ell-1}$ and c_ℓ

$\quad\quad x :=$ last j bits of $F^{-1}(k, c_\ell)$

$\quad\quad c_{\ell-1} := c_{\ell-1} \| x$

$\quad$ for $i = 1$ to ℓ:

$\quad\quad m_i := F^{-1}(k, c_i) \oplus c_{i-1}$

$\quad$ remove final j bits of m_ℓ

$\quad$ return $m_1 \| \cdots \| m_\ell$

The marked lines correspond to plain CBC mode.