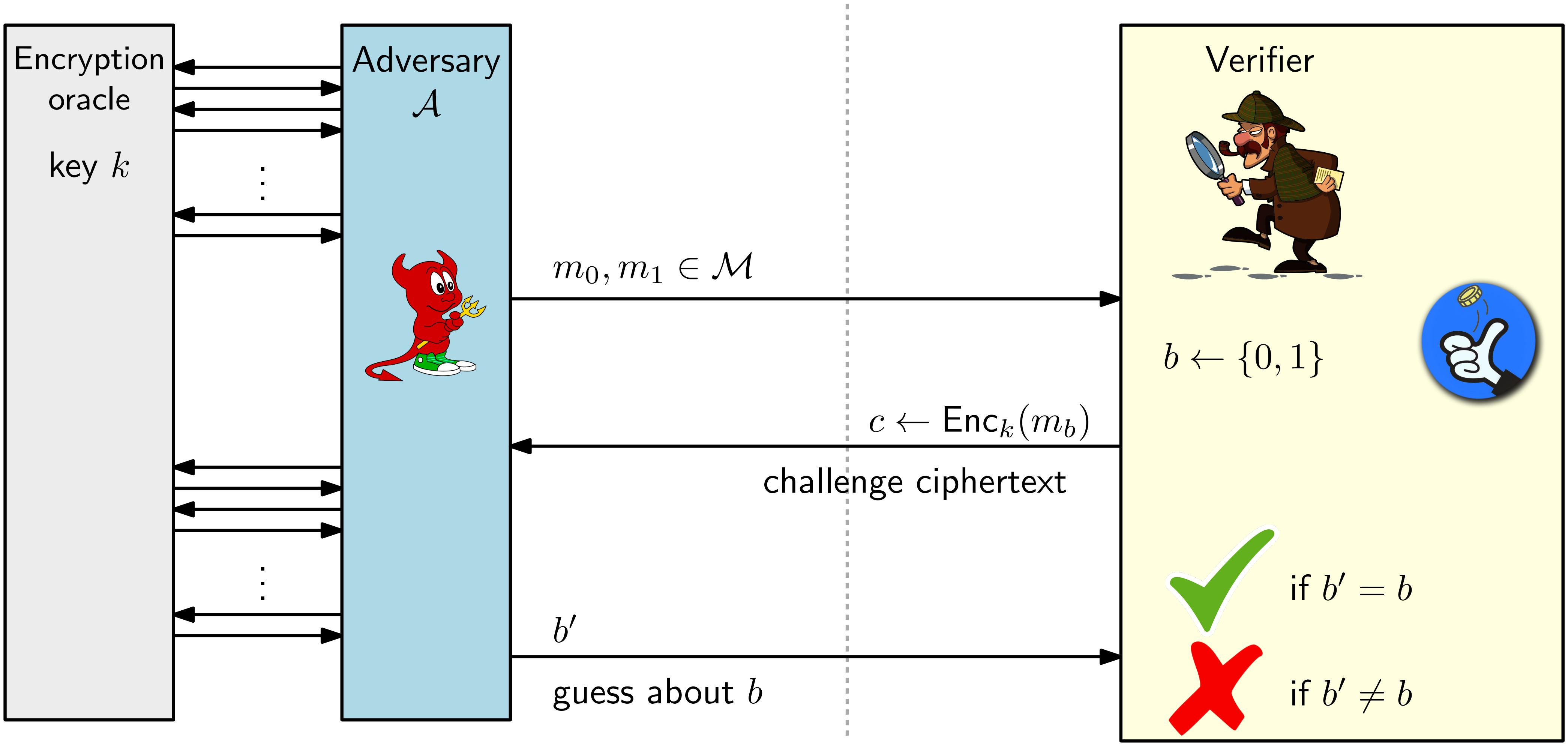
Recap: Modeling CPA security

A key $k \leftarrow \mathsf{Gen}(1^n)$ is generated



Definition of CPA-security

Definition: *A private key encryption scheme Π has indistinguishable encryptions under a chosen-plaintext attack (is* **CPA-secure***) if, for every probabilistic polynomial-time adversary $\mathcal{A}$, there is a negligible function ε such that:*

$$\Pr[\mathit{PrivK}^{cpa}_{\mathcal{A},\Pi}(n) = 1] \leq \frac{1}{2} + \varepsilon(n)$$

Any private-key encryption scheme that is CPA-secure is also CPA-secure for multiple encryptions

$\Downarrow$

If Π is CPA-secure then Π has indistinguishable multiple encryptions in the presence of an eavesdropper (and hence it is also EAV-secure)

Achieving CPA security (a first attempt)

Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a (length-preserving) pseudorandom permutation

Achieving CPA security (a first attempt)

Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a (length-preserving) pseudorandom permutation

Consider the following scheme for messages $m \in \{0,1\}^n$:

$$\mathsf{Enc}_k(m) = F_k(m) \qquad\qquad \mathsf{Dec}_k(c) = F_k^{-1}(c)$$

Achieving CPA security (a first attempt)

Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a (length-preserving) pseudorandom permutation

Consider the following scheme for messages $m \in \{0,1\}^n$:

$$\mathsf{Enc}_k(m) = F_k(m) \qquad\qquad \mathsf{Dec}_k(c) = F_k^{-1}(c)$$

Is this CPA-secure?

- We expect the output of Enc to be indistinguishable from a random binary string...

Achieving CPA security (a first attempt)

Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a (length-preserving) pseudorandom permutation

Consider the following scheme for messages $m \in \{0,1\}^n$:

$$\mathsf{Enc}_k(m) = F_k(m) \qquad\qquad \mathsf{Dec}_k(c) = F_k^{-1}(c)$$

Is this CPA-secure?

- We expect the output of Enc to be indistinguishable from a random binary string...

Notice that Enc depends only on k and m... **. . . the scheme is stateless and deterministic!**

Achieving CPA security (a first attempt)

Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a (length-preserving) pseudorandom permutation

Consider the following scheme for messages $m \in \{0,1\}^n$:

$$\mathsf{Enc}_k(m) = F_k(m) \qquad\qquad \mathsf{Dec}_k(c) = F_k^{-1}(c)$$

Is this CPA-secure?

- We expect the output of Enc to be indistinguishable from a random binary string...

Notice that Enc depends only on k and m... **...the scheme is stateless and deterministic!**

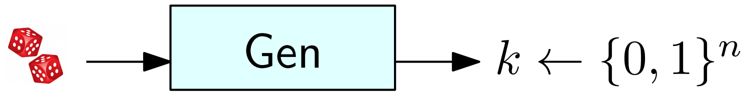
No stateless, deterministic encryption scheme can be CPA-secure

Actually achieving CPA security

Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a (length-preserving) pseudorandom function

Consider the following encryption scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$:

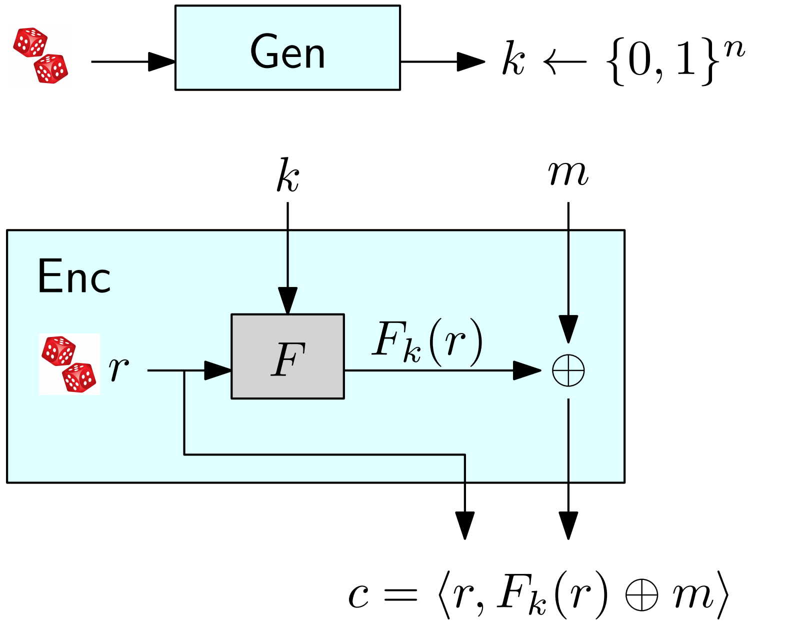
- $\mathsf{Gen}(1^n)$: return a key k chosen u.a.r. from $\{0,1\}^n$



Actually achieving CPA security

Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a (length-preserving) pseudorandom function

Consider the following encryption scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$:

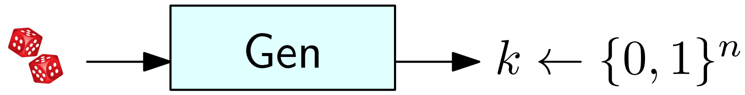


- Gen(1^n) : return a key k chosen u.a.r. from $\{0,1\}^n$
- Enc$_k(m)$ (where $|m| = n$):
  - Choose r uniformly at random from $\{0,1\}^n$
  - Return $c = \langle r, F_k(r) \oplus m \rangle$

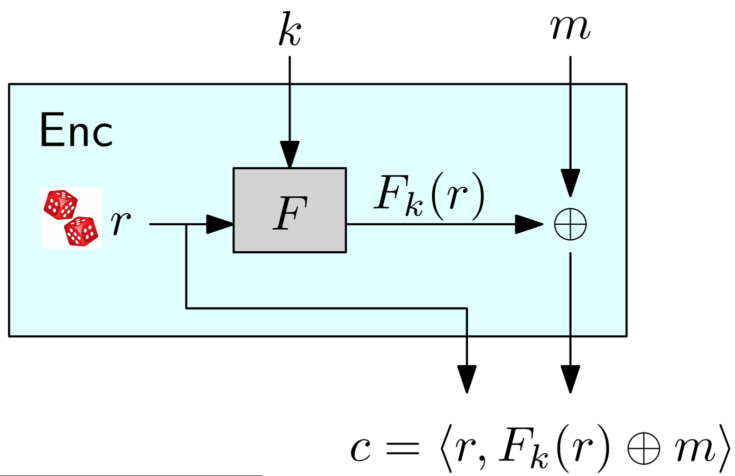
Actually achieving CPA security

Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a (length-preserving) pseudorandom function

Consider the following encryption scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$:

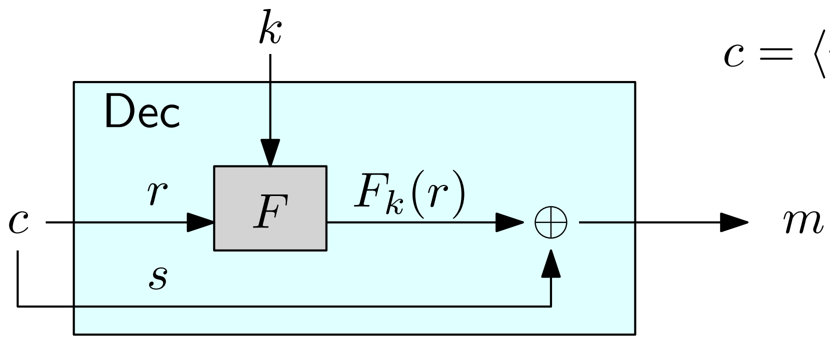


- $\mathsf{Gen}(1^n)$: return a key k chosen u.a.r. from $\{0,1\}^n$



- $\mathsf{Enc}_k(m)$ (where $|m| = n$):
  - Choose r uniformly at random from $\{0,1\}^n$
  - Return $c = \langle r, F_k(r) \oplus m \rangle$

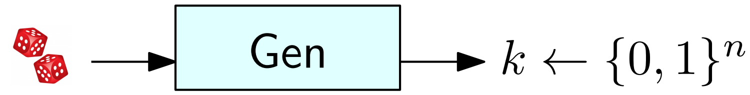
- $\mathsf{Dec}_k(c)$:
  - Split the ciphertext c into $\langle r, s \rangle$
  - Return $F_k(r) \oplus s$



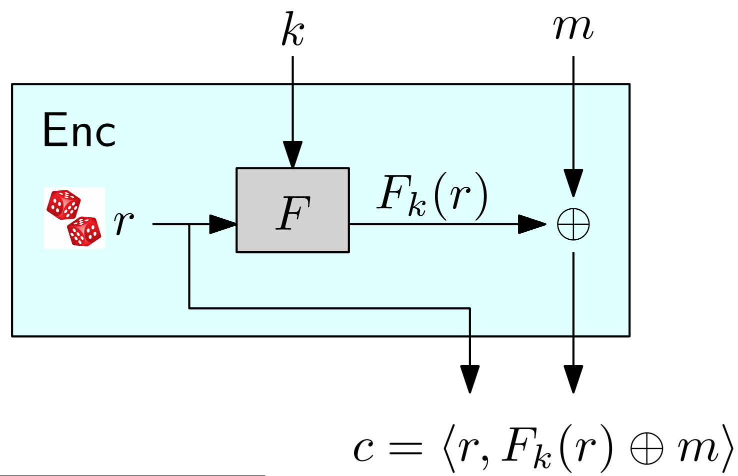
Actually achieving CPA security

Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a (length-preserving) pseudorandom function

Consider the following encryption scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$:

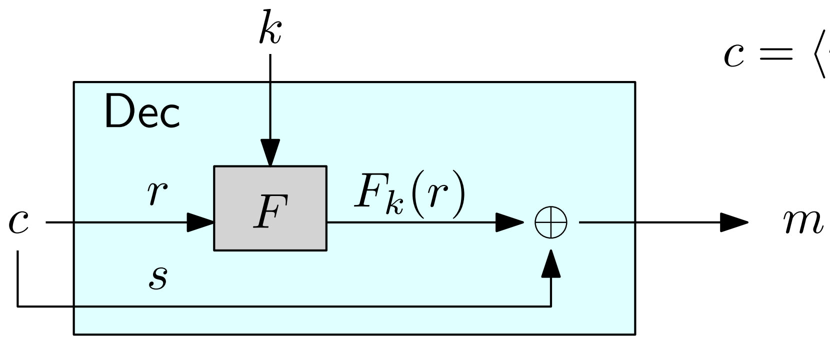


- $\mathsf{Gen}(1^n)$: return a key k chosen u.a.r. from $\{0,1\}^n$



- $\mathsf{Enc}_k(m)$ (where $|m| = n$):
  - Choose r uniformly at random from $\{0,1\}^n$
  - Return $c = \langle r, F_k(r) \oplus m \rangle$

- $\mathsf{Dec}_k(c)$:
  - Split the ciphertext c into $\langle r, s \rangle$
  - Return $F_k(r) \oplus s$

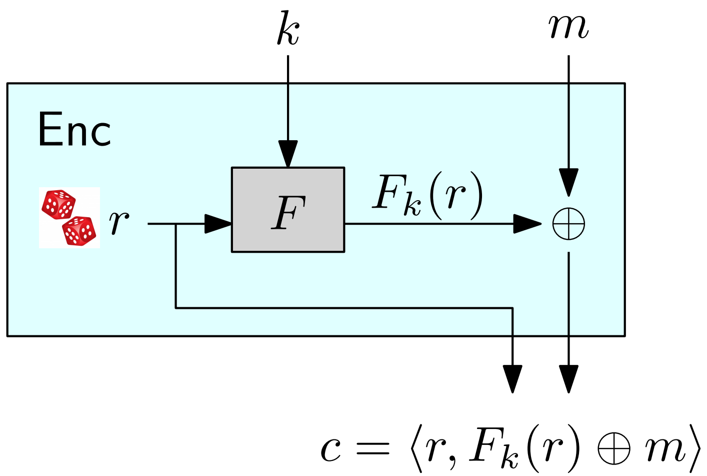


$\mathcal{M} = \{0,1\}^n \quad \mathcal{K} = \{0,1\}^n \quad \mathcal{C} = \{0,1\}^{2n}$

Actually achieving CPA security

Intuition:

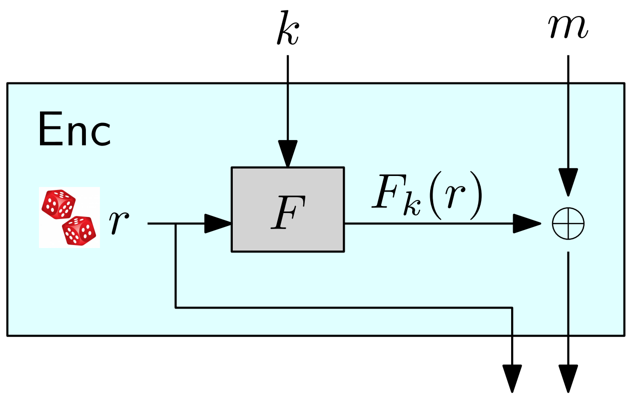
- When a message needs to be encrypted, a **fresh** random string r is generated
- r is used to "index into" a pseudorandom function F



Actually achieving CPA security

Intuition:

- When a message needs to be encrypted, a **fresh** random string r is generated
- r is used to "index into" a pseudorandom function F
- $F_k(r)$ is essentially a random string, and can only be computed if **both** k **and** r are known

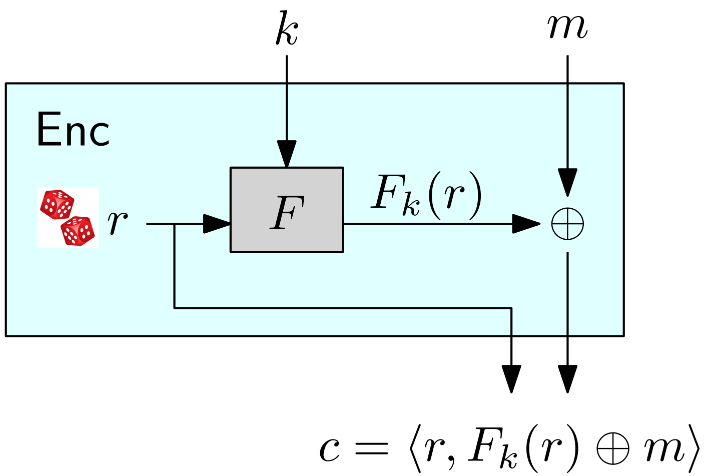


$c = \langle r, F_k(r) \oplus m \rangle$

Actually achieving CPA security

Intuition:

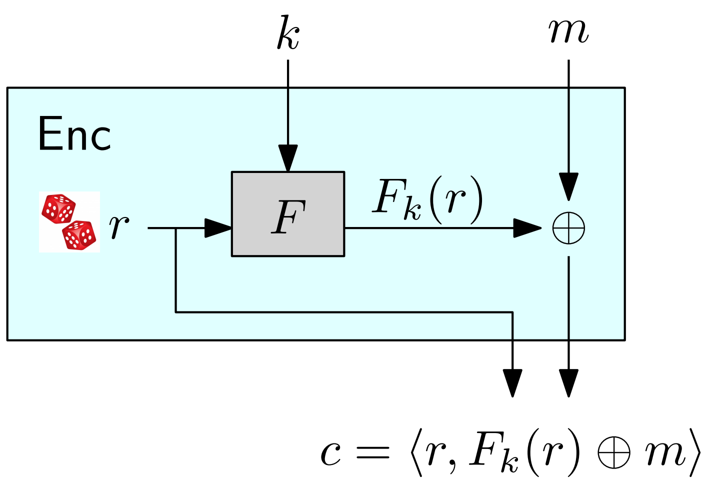
- When a message needs to be encrypted, a **fresh** random string r is generated
- r is used to "index into" a pseudorandom function F
- $F_k(r)$ is essentially a random string, and can only be computed if **both** k **and** r are known
- k is the **secret**, while r can be sent in the clear



Actually achieving CPA security

Intuition:

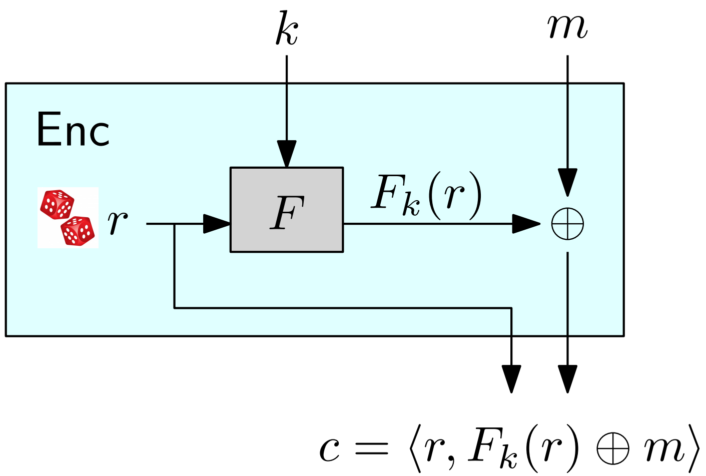
- When a message needs to be encrypted, a **fresh** random string r is generated
- r is used to "index into" a pseudorandom function F
- $F_k(r)$ is essentially a random string, and can only be computed if **both** k **and** r are known
- k is the **secret**, while r can be sent in the clear
- Encryption proceeds like in one-time pad, where the random string comes from $F_k(r)$



Actually achieving CPA security

Intuition:

- When a message needs to be encrypted, a **fresh** random string r is generated
- r is used to "index into" a pseudorandom function F
- $F_k(r)$ is essentially a random string, and can only be computed if **both** k **and** r are known
- k is the **secret**, while r can be sent in the clear
- Encryption proceeds like in one-time pad, where the random string comes from $F_k(r)$
- The process behaves similarly to the "real" OTP if the parties were to"agree on a new key" after each message



Proof of security

Can we **prove** that this encryption scheme is secure?

High-level proof strategy:

Proof of security

Can we **prove** that this encryption scheme is secure?

High-level proof strategy:

- Consider a variant $\widetilde{\Pi}$ of Π in which a truly random function f is used instead of F

Proof of security

Can we **prove** that this encryption scheme is secure?

High-level proof strategy:

- Consider a variant $\widetilde{\Pi}$ of Π in which a truly random function f is used instead of F
- Prove that if $\widetilde{\Pi}$ is CPA-secure then Π is CPA-secure

Proof of security

Can we **prove** that this encryption scheme is secure?

High-level proof strategy:

- Consider a variant $\widetilde{\Pi}$ of Π in which a truly random function f is used instead of F
- Prove that if $\widetilde{\Pi}$ is CPA-secure then Π is CPA-secure
- Prove that $\widetilde{\Pi}$ is CPA-secure

Proof of security

Can we **prove** that this encryption scheme is secure?

High-level proof strategy:

- Consider a variant $\widetilde{\Pi}$ of Π in which a truly random function f is used instead of F
- Prove that if $\widetilde{\Pi}$ is CPA-secure then Π is CPA-secure
- Prove that $\widetilde{\Pi}$ is CPA-secure

Proof of security

Can we **prove** that this encryption scheme is secure?

High-level proof strategy:

- Consider a variant $\widetilde{\Pi}$ of Π in which a truly random function f is used instead of F
- Prove that if $\widetilde{\Pi}$ is CPA-secure then Π is CPA-secure
- Prove that $\widetilde{\Pi}$ is CPA-secure

$\widetilde{\mathsf{Enc}}_k(m)$ (where $|m| = n$):

- Choose r uniformly at random from $\{0,1\}^n$
- Return $c = \langle r, f(r) \oplus m \rangle$

Proof of security

Can we **prove** that this encryption scheme is secure?

High-level proof strategy:

- Consider a variant $\widetilde{\Pi}$ of Π in which a truly random function f is used instead of F
- Prove that if $\widetilde{\Pi}$ is CPA-secure then Π is CPA-secure
- Prove that $\widetilde{\Pi}$ is CPA-secure

$\widetilde{\mathsf{Enc}}_k(m)$ (where $|m| = n$):

- Choose r uniformly at random from $\{0,1\}^n$
- Return $c = \langle r, f(r) \oplus m \rangle$

If $\widetilde{\Pi}$ is CPA-secure then Π is CPA-secure

We will show that, regardless of $\mathcal{A}$, $\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1]$ and $\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1]$ are almost the same (i.e., they differ by a negligible function)

$$\left| \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1] - \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1] \right| \leq \varepsilon(n) \quad \text{for some negligible function } \varepsilon(n)$$

If $\widetilde{\Pi}$ is CPA-secure then Π is CPA-secure

We will show that, regardless of $\mathcal{A}$, $\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1]$ and $\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1]$ are almost the same (i.e., they differ by a negligible function)

$$\left| \ \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1] - \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1] \ \right| \leq \varepsilon(n) \quad \text{for some negligible function } \varepsilon(n)$$

Security reduction:

- Suppose that the above inequality does not hold, i.e., there is some polynomial-time adversary $\mathcal{A}$ s.t.:

$$\left| \ \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1] - \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1] \ \right| = \varepsilon(n) \quad \text{for some non-negligible } \varepsilon(n)$$

If $\widetilde{\Pi}$ is CPA-secure then Π is CPA-secure

We will show that, regardless of $\mathcal{A}$, $\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1]$ and $\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1]$ are almost the same (i.e., they differ by a negligible function)

$$\left|\ \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1] - \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1]\ \right| \leq \varepsilon(n) \quad \text{for some negligible function } \varepsilon(n)$$

Security reduction:

- Suppose that the above inequality does not hold, i.e., there is some polynomial-time adversary $\mathcal{A}$ s.t.:

$$\left|\ \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1] - \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1]\ \right| = \varepsilon(n) \quad \text{for some non-negligible } \varepsilon(n)$$

- We use $\mathcal{A}$ to build a distinguisher D with non-negligible gap for the PRF F used in Π

If $\widetilde{\Pi}$ is CPA-secure then Π is CPA-secure

We will show that, regardless of $\mathcal{A}$, $\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1]$ and $\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1]$ are almost the same (i.e., they differ by a negligible function)

$$\left| \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1] - \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1] \right| \leq \varepsilon(n) \quad \text{for some negligible function } \varepsilon(n)$$

Security reduction:

- Suppose that the above inequality does not hold, i.e., there is some polynomial-time adversary $\mathcal{A}$ s.t.:

$$\left| \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1] - \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1] \right| = \varepsilon(n) \quad \text{for some non-negligible } \varepsilon(n)$$

- We use $\mathcal{A}$ to build a distinguisher D with non-negligible gap for the PRF F used in Π

- Contradiction

If $\widetilde{\Pi}$ is CPA-secure then Π is CPA-secure

We will show that, regardless of $\mathcal{A}$, $\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1]$ and $\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1]$ are almost the same (i.e., they differ by a negligible function)

$$\Big| \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1] - \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1] \Big| \leq \varepsilon(n) \quad \text{for some negligible function } \varepsilon(n)$$

Security reduction:

- Suppose that the above inequality does not hold, i.e., there is some polynomial-time adversary $\mathcal{A}$ s.t.:

$$\Big| \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1] - \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1] \Big| = \varepsilon(n) \quad \text{for some non-negligible } \varepsilon(n)$$

- We use $\mathcal{A}$ to build a distinguisher D with non-negligible gap for the PRF F used in Π

- Contradiction $\implies$ there is an adversary that "wins" $\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n)$ iff it"wins" $\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n)$

If $\widetilde{\Pi}$ is CPA-secure then Π is CPA-secure

We will show that, regardless of $\mathcal{A}$, $\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1]$ and $\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1]$ are almost the same (i.e., they differ by a negligible function)

$$\left| \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1] - \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1] \right| \leq \varepsilon(n) \quad \text{for some negligible function } \varepsilon(n)$$

Security reduction:

- Suppose that the above inequality does not hold, i.e., there is some polynomial-time adversary $\mathcal{A}$ s.t.:

$$\left| \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1] - \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1] \right| = \varepsilon(n) \quad \text{for some non-negligible } \varepsilon(n)$$

- We use $\mathcal{A}$ to build a distinguisher D with non-negligible gap for the PRF F used in Π

- Contradiction $\implies$ there is an adversary that "wins" $\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n)$ iff it "wins" $\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n)$

If $\widetilde{\Pi}$ is CPA-secure then Π is CPA-secure

Distinguisher $D^{\Phi}(1^n)$: (for the PRF F)

- Run $\mathcal{A}$.
- Whenever $\mathcal{A}$ queries the encryption oracle with m:

If $\widetilde{\Pi}$ is CPA-secure then Π is CPA-secure

Distinguisher $D^{\Phi}(1^n)$: (for the PRF F)

- Run $\mathcal{A}$.
- Whenever $\mathcal{A}$ queries the encryption oracle with m:
  - Query Φ with a $r \in \{0,1\}^n$ chosen u.a.r. and obtain a response y
  - Answer $\langle r, y \oplus m \rangle$

If $\widetilde{\Pi}$ is CPA-secure then Π is CPA-secure

Distinguisher $D^{\Phi}(1^n)$: (for the PRF F)

- Run $\mathcal{A}$.
- Whenever $\mathcal{A}$ queries the encryption oracle with m:
  - Query Φ with a $r \in \{0,1\}^n$ chosen u.a.r. and obtain a response y
  - Answer $\langle r, y \oplus m \rangle$
- When $\mathcal{A}$ outputs m_0, m_1:

If $\widetilde{\Pi}$ is CPA-secure then Π is CPA-secure

Distinguisher $D^{\Phi}(1^n)$: (for the PRF F)

- Run $\mathcal{A}$.
- Whenever $\mathcal{A}$ queries the encryption oracle with m:
  - Query Φ with a $r \in \{0,1\}^n$ chosen u.a.r. and obtain a response y
  - Answer $\langle r, y \oplus m \rangle$
- When $\mathcal{A}$ outputs m_0, m_1:
  - Choose b u.a.r. from $\{0,1\}$
  - Query Φ with a $r^* \in \{0,1\}^n$ chosen u.a.r. and obtain a response y
  - Send the ciphertext $\langle r^*, y \oplus m_b \rangle$ to $\mathcal{A}$

If $\widetilde{\Pi}$ is CPA-secure then Π is CPA-secure

Distinguisher $D^{\Phi}(1^n)$: (for the PRF F)

- Run $\mathcal{A}$.
- Whenever $\mathcal{A}$ queries the encryption oracle with m:
  - Query Φ with a $r \in \{0,1\}^n$ chosen u.a.r. and obtain a response y
  - Answer $\langle r, y \oplus m \rangle$
- When $\mathcal{A}$ outputs m_0, m_1:
  - Choose b u.a.r. from $\{0,1\}$
  - Query Φ with a $r^* \in \{0,1\}^n$ chosen u.a.r. and obtain a response y
  - Send the ciphertext $\langle r^*, y \oplus m_b \rangle$ to $\mathcal{A}$
- When $\mathcal{A}$ outputs its guess b':

If $\widetilde{\Pi}$ is CPA-secure then Π is CPA-secure

Distinguisher $D^{\Phi}(1^n)$: (for the PRF F)

- Run $\mathcal{A}$.
- Whenever $\mathcal{A}$ queries the encryption oracle with m:
  - Query Φ with a $r \in \{0,1\}^n$ chosen u.a.r. and obtain a response y
  - Answer $\langle r, y \oplus m \rangle$
- When $\mathcal{A}$ outputs m_0, m_1:
  - Choose b u.a.r. from $\{0,1\}$
  - Query Φ with a $r^* \in \{0,1\}^n$ chosen u.a.r. and obtain a response y
  - Send the ciphertext $\langle r^*, y \oplus m_b \rangle$ to $\mathcal{A}$
- When $\mathcal{A}$ outputs its guess b':
  - Return 1 if $b' = b$ and 0 otherwise

If $\widetilde{\Pi}$ is CPA-secure then Π is CPA-secure

When Φ is the PRF F_k:

If $\widetilde{\Pi}$ is CPA-secure then Π is CPA-secure

When Φ is the PRF F_k:

- The behaviour of D^{Φ} is exactly that of the experiment $\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n)$

$$\Pr[D^{F_k(\cdot)}(1^n) = 1] = \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1]$$

If $\widetilde{\Pi}$ is CPA-secure then Π is CPA-secure

When Φ is the PRF F_k:

- The behaviour of D^{Φ} is exactly that of the experiment $\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n)$

$$\Pr[D^{F_k(\cdot)}(1^n) = 1] = \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1]$$

When Φ is a random function f:

If $\widetilde{\Pi}$ is CPA-secure then Π is CPA-secure

When Φ is the PRF F_k:

- The behaviour of D^{Φ} is exactly that of the experiment $\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n)$

$$\Pr[D^{F_k(\cdot)}(1^n) = 1] = \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1]$$

When Φ is a random function f:

- The behaviour of D^{Φ} is exactly that of the experiment $\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n)$

$$\Pr[D^{f(\cdot)}(1^n) = 1] = \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1]$$

If $\widetilde{\Pi}$ is CPA-secure then Π is CPA-secure

When Φ is the PRF F_k:

- The behaviour of D^{Φ} is exactly that of the experiment $\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n)$

$$\Pr[D^{F_k(\cdot)}(1^n) = 1] = \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1]$$

When Φ is a random function f:

- The behaviour of D^{Φ} is exactly that of the experiment $\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n)$

$$\Pr[D^{f(\cdot)}(1^n) = 1] = \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1]$$

$$\Big|\ \Pr[D^{F_k(\cdot)}(1^n) = 1] - \Pr[D^{f(\cdot)}(1^n) = 1]\ \Big| = \Big|\ \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1] - \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1]\ \Big| = \varepsilon(n)$$

If $\widetilde{\Pi}$ is CPA-secure then Π is CPA-secure

When Φ is the PRF F_k:

- The behaviour of D^{Φ} is exactly that of the experiment $\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n)$

$$\Pr[D^{F_k(\cdot)}(1^n) = 1] = \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1]$$

When Φ is a random function f:

- The behaviour of D^{Φ} is exactly that of the experiment $\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n)$

$$\Pr[D^{f(\cdot)}(1^n) = 1] = \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1]$$

$$\Big| \Pr[D^{F_k(\cdot)}(1^n) = 1] - \Pr[D^{f(\cdot)}(1^n) = 1] \Big| = \Big| \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n) = 1] - \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1] \Big| = \varepsilon(n)$$

Not negligible!

Proof of security

Can we **prove** that this encryption scheme is secure?

High-level proof strategy:

- Consider a variant $\widetilde{\Pi}$ of Π in which a truly random function f is used instead of F
- Prove that if $\widetilde{\Pi}$ is CPA-secure then Π is CPA-secure ✔
- Prove that $\widetilde{\Pi}$ is CPA-secure

$\widetilde{\mathsf{Enc}}_k(m)$ (where $|m| = n$):

- Choose r uniformly at random from $\{0,1\}^n$
- Return $c = \langle r, f(r) \oplus m \rangle$

Proof of security

Can we **prove** that this encryption scheme is secure?

High-level proof strategy:

- Consider a variant $\widetilde{\Pi}$ of Π in which a truly random function f is used instead of F
- Prove that if $\widetilde{\Pi}$ is CPA-secure then Π is CPA-secure
- Prove that $\widetilde{\Pi}$ is CPA-secure

$\widetilde{\mathsf{Enc}}_k(m)$ (where $|m| = n$):

- Choose r uniformly at random from $\{0,1\}^n$
- Return $c = \langle r, f(r) \oplus m \rangle$

Showing that $\widetilde{\Pi}$ is CPA-secure

Let $\mathcal{A}$ be any polynomial-time adversary and consider the experiment $\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n)$

Showing that $\widetilde{\Pi}$ is CPA-secure

Let $\mathcal{A}$ be any polynomial-time adversary and consider the experiment $\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n)$

- The challenge ciphertext is computed as follows:
  - $b \in \{0,1\}$ and $r^* \in \{0,1\}^n$ are chosen u.a.r.
  - The challenge ciphertext is $\langle r^*, f(r^*) \oplus m_b \rangle$

Showing that $\widetilde{\Pi}$ is CPA-secure

Let $\mathcal{A}$ be any polynomial-time adversary and consider the experiment $\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n)$

- The challenge ciphertext is computed as follows:
  - $b \in \{0,1\}$ and $r^* \in \{0,1\}^n$ are chosen u.a.r.
  - The challenge ciphertext is $\langle r^*, f(r^*) \oplus m_b \rangle$
- Whenever $\mathcal{A}$ queries the encryption oracle for the i-th time with a message m_i:
  - A value r_i is chosen u.a.r. from $\{0,1\}^n$
  - The oracle answers with $\langle r_i, f(r_i) \oplus m_i \rangle$

Showing that $\widetilde{\Pi}$ is CPA-secure

Let $\mathcal{A}$ be any polynomial-time adversary and consider the experiment $\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n)$

- The challenge ciphertext is computed as follows:
  - $b \in \{0,1\}$ and $r^* \in \{0,1\}^n$ are chosen u.a.r.
  - The challenge ciphertext is $\langle r^*, f(r^*) \oplus m_b \rangle$
- Whenever $\mathcal{A}$ queries the encryption oracle for the i-th time with a message m_i:
  - A value r_i is chosen u.a.r. from $\{0,1\}^n$
  - The oracle answers with $\langle r_i, f(r_i) \oplus m_i \rangle$

We consider two cases:

- There is at least one r_i s.t. $r_i = r^*$
- The value r^* is different from all r_is

Showing that $\widetilde{\Pi}$ is CPA-secure

Let $\mathcal{A}$ be any polynomial-time adversary and consider the experiment $\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n)$

- The challenge ciphertext is computed as follows:
  - $b \in \{0,1\}$ and $r^* \in \{0,1\}^n$ are chosen u.a.r.
  - The challenge ciphertext is $\langle r^*, f(r^*) \oplus m_b \rangle$
- Whenever $\mathcal{A}$ queries the encryption oracle for the i-th time with a message m_i:
  - A value r_i is chosen u.a.r. from $\{0,1\}^n$
  - The oracle answers with $\langle r_i, f(r_i) \oplus m_i \rangle$

We consider two cases:

- There is at least one r_i s.t. $r_i = r^*$ We call this event "repeat"
- The value r^* is different from all r_is

Showing that $\widetilde{\Pi}$ is CPA-secure

Let $\mathcal{A}$ be any polynomial-time adversary and consider the experiment $\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n)$

- The challenge ciphertext is computed as follows:
  - $b \in \{0,1\}$ and $r^* \in \{0,1\}^n$ are chosen u.a.r.
  - The challenge ciphertext is $\langle r^*, f(r^*) \oplus m_b \rangle$
- Whenever $\mathcal{A}$ queries the encryption oracle for the i-th time with a message m_i:
  - A value r_i is chosen u.a.r. from $\{0,1\}^n$
  - The oracle answers with $\langle r_i, f(r_i) \oplus m_i \rangle$

We consider two cases:

- There is at least one r_i s.t. $r_i = r^*$ — We call this event "repeat"
- The value r^* is different from all r_is — This event is the complement of "repeat", i.e. $\overline{\text{repeat}}$

Showing that $\widetilde{\Pi}$ is CPA-secure

$$\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1] \quad = \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \wedge \mathsf{repeat}] + \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \wedge \overline{\mathsf{repeat}}\,]$$

Showing that $\widetilde{\Pi}$ is CPA-secure

$$\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1] \quad = \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \wedge \mathsf{repeat}] + \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \wedge \overline{\mathsf{repeat}}\,]$$

$$\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \wedge \mathsf{repeat}]$$

Showing that $\widetilde{\Pi}$ is CPA-secure

$$\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1] \quad = \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \wedge \mathsf{repeat}] + \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \wedge \overline{\mathsf{repeat}}\,]$$

$$\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \wedge \mathsf{repeat}] \leq \Pr[\mathsf{repeat}]$$

Showing that $\widetilde{\Pi}$ is CPA-secure

$$\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1] \quad = \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \wedge \mathsf{repeat}] + \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \wedge \overline{\mathsf{repeat}}\,]$$

$$\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \wedge \mathsf{repeat}] \leq \Pr[\mathsf{repeat}] \ \leq \sum_i \Pr[r_i = r^*]$$

Showing that $\widetilde{\Pi}$ is CPA-secure

$$\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1] \quad = \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \wedge \mathsf{repeat}] + \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \wedge \overline{\mathsf{repeat}}\,]$$

$$\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \wedge \mathsf{repeat}] \leq \Pr[\mathsf{repeat}] \ \leq \textstyle\sum_i \Pr[r_i = r^*] \leq \sum_i \frac{1}{2^n}$$

Showing that $\widetilde{\Pi}$ is CPA-secure

$$\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n)=1] \quad = \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n)=1 \wedge \mathsf{repeat}] + \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n)=1 \wedge \overline{\mathsf{repeat}}\,]$$

$$\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n)=1 \wedge \mathsf{repeat}] \le \Pr[\mathsf{repeat}] \ \le \sum_i \Pr[r_i = r^*] \le \sum_i \tfrac{1}{2^n} \quad = \tfrac{q(n)}{2^n}$$

where $q(n)$ is an upper bound on the number of queries performed by $\mathcal{A}$.

Showing that $\widetilde{\Pi}$ is CPA-secure

$$\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n)=1] \quad = \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n)=1 \wedge \mathsf{repeat}] + \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n)=1 \wedge \overline{\mathsf{repeat}}\,]$$

$$\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n)=1 \wedge \mathsf{repeat}] \leq \Pr[\mathsf{repeat}] \ \leq \sum_i \Pr[r_i = r^*] \leq \sum_i \frac{1}{2^n} \quad = \frac{\overset{\text{polynomially bounded}}{q(n)}}{2^n}$$

where $q(n)$ is an upper bound on the number of queries performed by $\mathcal{A}$.

Showing that $\widetilde{\Pi}$ is CPA-secure

$$\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1] \quad = \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \wedge \mathsf{repeat}] + \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \wedge \overline{\mathsf{repeat}}\,]$$

polynomially bounded

$$\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \wedge \mathsf{repeat}] \le \Pr[\mathsf{repeat}] \le \sum_i \Pr[r_i = r^*] \le \sum_i \frac{1}{2^n} \quad = \boxed{\frac{q(n)}{2^n}}$$

where $q(n)$ is an upper bound on the number of queries performed by $\mathcal{A}$.

negligible

Showing that $\widetilde{\Pi}$ is CPA-secure

$$\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1] \quad = \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \wedge \mathsf{repeat}] + \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \wedge \overline{\mathsf{repeat}}\,]$$

$$\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \wedge \mathsf{repeat}] \leq \Pr[\mathsf{repeat}] \; \leq \sum_i \Pr[r_i = r^*] \leq \sum_i \frac{1}{2^n} \quad = \frac{\overbrace{q(n)}^{\text{polynomially bounded}}}{2^n}$$

negligible

where $q(n)$ is an upper bound on the number of queries performed by $\mathcal{A}$.

$$\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \wedge \overline{\mathsf{repeat}}]$$

Showing that $\widetilde{\Pi}$ is CPA-secure

$$\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1] \quad = \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \wedge \mathsf{repeat}] + \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \wedge \overline{\mathsf{repeat}}\,]$$

polynomially bounded

$$\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \wedge \mathsf{repeat}] \leq \Pr[\mathsf{repeat}] \ \leq \sum_i \Pr[r_i = r^*] \leq \sum_i \frac{1}{2^n} \quad = \boxed{\frac{\boxed{q(n)}}{2^n}}$$

where $q(n)$ is an upper bound on the number of queries performed by $\mathcal{A}$.

negligible

$$\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \wedge \overline{\mathsf{repeat}}] \ \ = \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \mid \overline{\mathsf{repeat}}] \ \cdot \ \Pr[\overline{\mathsf{repeat}}]$$

Showing that $\widetilde{\Pi}$ is CPA-secure

$$\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1] \quad = \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \wedge \mathsf{repeat}] + \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \wedge \overline{\mathsf{repeat}}\,]$$

polynomially bounded

$$\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \wedge \mathsf{repeat}] \leq \Pr[\mathsf{repeat}] \leq \sum_i \Pr[r_i = r^*] \leq \sum_i \frac{1}{2^n} \quad = \boxed{\frac{q(n)}{2^n}}$$

where $q(n)$ is an upper bound on the number of queries performed by $\mathcal{A}$.

negligible

$$\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \wedge \overline{\mathsf{repeat}}] \quad = \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \mid \overline{\mathsf{repeat}}] \cdot \Pr[\overline{\mathsf{repeat}}]$$

$$\leq \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \mid \overline{\mathsf{repeat}}]$$

Showing that $\widetilde{\Pi}$ is CPA-secure

$$\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1] \quad = \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \wedge \mathsf{repeat}] + \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \wedge \overline{\mathsf{repeat}}\,]$$

polynomially bounded

$$\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \wedge \mathsf{repeat}] \leq \Pr[\mathsf{repeat}] \;\leq \textstyle\sum_i \Pr[r_i = r^*] \leq \sum_i \frac{1}{2^n} \quad = \boxed{\frac{q(n)}{2^n}}$$

where $q(n)$ is an upper bound on the number of queries performed by $\mathcal{A}$.

negligible

$$\begin{aligned}\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \wedge \overline{\mathsf{repeat}}] \;&= \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \mid \overline{\mathsf{repeat}}] \cdot \Pr[\overline{\mathsf{repeat}}] \\ &\leq \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \mid \overline{\mathsf{repeat}}]\end{aligned}$$

- Since $r_i \neq r^* \, \forall i$, and f is a random function, all $f(r_i)$ are chosen independently from $f(r^*)$

Showing that $\widetilde{\Pi}$ is CPA-secure

$$\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1] \quad = \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \land \mathsf{repeat}] + \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \land \overline{\mathsf{repeat}}\,]$$

polynomially bounded

$$\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \land \mathsf{repeat}] \leq \Pr[\mathsf{repeat}] \;\leq \textstyle\sum_i \Pr[r_i = r^*] \leq \sum_i \frac{1}{2^n} \quad = \boxed{\frac{q(n)}{2^n}}$$

where $q(n)$ is an upper bound on the number of queries performed by $\mathcal{A}$.

negligible

$$\begin{aligned}\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \land \overline{\mathsf{repeat}}] \quad &= \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \mid \overline{\mathsf{repeat}}] \cdot \Pr[\overline{\mathsf{repeat}}] \\ &\leq \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \mid \overline{\mathsf{repeat}}]\end{aligned}$$

- Since $r_i \neq r^* \, \forall i$, and f is a random function, all $f(r_i)$ are chosen independently from $f(r^*)$
- $\mathcal{A}$ learns nothing about $f(r^*)$ while interacting with the encryption oracle

Showing that $\widetilde{\Pi}$ is CPA-secure

$$\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1] \quad = \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \wedge \mathsf{repeat}] + \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \wedge \overline{\mathsf{repeat}}\,]$$

polynomially bounded

$$\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \wedge \mathsf{repeat}] \leq \Pr[\mathsf{repeat}] \;\leq \textstyle\sum_i \Pr[r_i = r^*] \leq \sum_i \frac{1}{2^n} \quad = \boxed{\frac{q(n)}{2^n}}$$

where $q(n)$ is an upper bound on the number of queries performed by $\mathcal{A}$.

negligible

$$\begin{aligned}\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \wedge \overline{\mathsf{repeat}}] \;& = \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \mid \overline{\mathsf{repeat}}] \cdot \Pr[\overline{\mathsf{repeat}}] \\ & \leq \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \mid \overline{\mathsf{repeat}}]\end{aligned}$$

- Since $r_i \neq r^* \, \forall i$, and f is a random function, all $f(r_i)$ are chosen independently from $f(r^*)$
- $\mathcal{A}$ learns nothing about $f(r^*)$ while interacting with the encryption oracle $\qquad \Pr[b' = b] = \frac{1}{2}$

Showing that $\widetilde{\Pi}$ is CPA-secure

$$\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n)=1] \quad = \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n)=1 \wedge \mathsf{repeat}] + \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n)=1 \wedge \overline{\mathsf{repeat}}\,]$$

polynomially bounded

$$\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n)=1 \wedge \mathsf{repeat}] \le \Pr[\mathsf{repeat}] \;\le \textstyle\sum_i \Pr[r_i = r^*] \le \sum_i \frac{1}{2^n} \quad = \boxed{\frac{q(n)}{2^n}}$$

where $q(n)$ is an upper bound on the number of queries performed by $\mathcal{A}$.

negligible

$$\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n)=1 \wedge \overline{\mathsf{repeat}}] \;\; = \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n)=1 \mid \overline{\mathsf{repeat}}] \cdot \Pr[\overline{\mathsf{repeat}}]$$

$$\le \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n)=1 \mid \overline{\mathsf{repeat}}] \;\; = \frac{1}{2}$$

- Since $r_i \neq r^* \; \forall i$, and f is a random function, all $f(r_i)$ are chosen independently from $f(r^*)$
- $\mathcal{A}$ learns nothing about $f(r^*)$ while interacting with the encryption oracle $\qquad \Pr[b' = b] = \frac{1}{2}$

Showing that $\widetilde{\Pi}$ is CPA-secure

negligible

$$\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1] \quad = \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \wedge \mathsf{repeat}] + \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \wedge \overline{\mathsf{repeat}}\,] \quad \le \tfrac{1}{2} + \boxed{\tfrac{q(n)}{2^n}}$$

polynomially bounded

$$\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \wedge \mathsf{repeat}] \le \Pr[\mathsf{repeat}] \ \le \textstyle\sum_i \Pr[r_i = r^*] \le \sum_i \frac{1}{2^n} \quad = \boxed{\frac{\boxed{q(n)}}{2^n}}$$

where $q(n)$ is an upper bound on the number of queries performed by $\mathcal{A}$.

negligible

$$\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \wedge \overline{\mathsf{repeat}}] \quad = \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \mid \overline{\mathsf{repeat}}] \cdot \Pr[\overline{\mathsf{repeat}}]$$

$$\le \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\widetilde{\Pi}}(n) = 1 \mid \overline{\mathsf{repeat}}] \ = \tfrac{1}{2}$$

- Since $r_i \neq r^* \, \forall i$, and f is a random function, all $f(r_i)$ are chosen independently from $f(r^*)$
- $\mathcal{A}$ learns nothing about $f(r^*)$ while interacting with the encryption oracle $\qquad \Pr[b' = b] = \frac{1}{2}$

Proof of security

Can we **prove** that this encryption scheme is secure?

High-level proof strategy:

- Consider a variant $\widetilde{\Pi}$ of Π in which a truly random function f is used instead of F
- Prove that if $\widetilde{\Pi}$ is CPA-secure then Π is CPA-secure
- Prove that $\widetilde{\Pi}$ is CPA-secure

$\widetilde{\mathsf{Enc}}_k(m)$ (where $|m| = n$):

- Choose r uniformly at random from $\{0,1\}^n$
- Return $c = \langle r, f(r) \oplus m \rangle$

Proof of security

Can we **prove** that this encryption scheme is secure?

High-level proof strategy:

- Consider a variant $\widetilde{\Pi}$ of Π in which a truly random function f is used instead of F
- Prove that if $\widetilde{\Pi}$ is CPA-secure then Π is CPA-secure
- Prove that $\widetilde{\Pi}$ is CPA-secure

$\widetilde{\mathsf{Enc}}_k(m)$ (where $|m| = n$):

- Choose r uniformly at random from $\{0, 1\}^n$
- Return $c = \langle r, f(r) \oplus m \rangle$

$\square$

Caveats

- The values r used for encryption should never be reused

Caveats

- The values r used for encryption should never be reused
  - If a value r is reused, then there are two ciphertexts $\langle r, F_k(r) \oplus m \rangle$ and $\langle r, F_k(r) \oplus m' \rangle$

Caveats

- The values r used for encryption should never be reused
  - If a value r is reused, then there are two ciphertexts $\langle r, F_k(r) \oplus m \rangle$ and $\langle r, F_k(r) \oplus m' \rangle$
  - The adversary can detect this, and compute $(F_k(r) \oplus m) \oplus (F_k(r) \oplus m') = m \oplus m'$

Caveats

- The values r used for encryption should never be reused
  - If a value r is reused, then there are two ciphertexts $\langle r, F_k(r) \oplus m \rangle$ and $\langle r, F_k(r) \oplus m' \rangle$
  - The adversary can detect this, and compute $(F_k(r) \oplus m) \oplus (F_k(r) \oplus m') = m \oplus m'$
  - Same problem as in OTP

Caveats

- The values r used for encryption should never be reused
  - If a value r is reused, then there are two ciphertexts $\langle r, F_k(r) \oplus m \rangle$ and $\langle r, F_k(r) \oplus m' \rangle$
  - The adversary can detect this, and compute $(F_k(r) \oplus m) \oplus (F_k(r) \oplus m') = m \oplus m'$
  - Same problem as in OTP

- However, when n is sufficiently large and r is chosen uniformly at random, the probability of a repeat is negligible
- This is exactly what we used in our security proof!

Caveats

- The values r used for encryption should never be reused
  - If a value r is reused, then there are two ciphertexts $\langle r, F_k(r) \oplus m \rangle$ and $\langle r, F_k(r) \oplus m' \rangle$
  - The adversary can detect this, and compute $(F_k(r) \oplus m) \oplus (F_k(r) \oplus m') = m \oplus m'$
  - Same problem as in OTP

- However, when n is sufficiently large and r is chosen uniformly at random, the probability of a repeat is negligible
- This is exactly what we used in our security proof!

- If n is too short, or it is not chosen from a uniform distribution then repeats might happen!

Encrypting long messages

What if we want to send a message that is longer than the security parameter (i.e., $|m| > n$) ?

Encrypting long messages

What if we want to send a message that is longer than the security parameter (i.e., $|m| > n$) ?

Idea:

- We have shown that any CPA-secure encryption scheme is also secure for multiple encryptions

Encrypting long messages

What if we want to send a message that is longer than the security parameter (i.e., $|m| > n$) ?

Idea:

- We have shown that any CPA-secure encryption scheme is also secure for multiple encryptions
- Think of m as the concatenation of many messages of length n, i.e., $m = m_1 \,\|\, m_2 \,\|\, m_3 \,\|\, \ldots$
- Encrypt each m_i separately

Encrypting long messages

What if we want to send a message that is longer than the security parameter (i.e., $|m| > n$) ?

Idea:

- We have shown that any CPA-secure encryption scheme is also secure for multiple encryptions
- Think of m as the concatenation of many messages of length n, i.e., $m = m_1 \,\|\, m_2 \,\|\, m_3 \,\|\, \ldots$
- Encrypt each m_i separately

Does it work?

Encrypting long messages

What if we want to send a message that is longer than the security parameter (i.e., $|m| > n$) ?

Idea:

- We have shown that any CPA-secure encryption scheme is also secure for multiple encryptions
- Think of m as the concatenation of many messages of length n, i.e., $m = m_1 \| m_2 \| m_3 \| \ldots$
- Encrypt each m_i separately

Does it work? Yes, but...

Encrypting long messages

What if we want to send a message that is longer than the security parameter (i.e., $|m| > n$) ?

Idea:

- We have shown that any CPA-secure encryption scheme is also secure for multiple encryptions
- Think of m as the concatenation of many messages of length n, i.e., $m = m_1 \,\|\, m_2 \,\|\, m_3 \,\|\, \dots$
- Encrypt each m_i separately

Does it work? Yes, but...

- The ciphertext is (at least) twice as long as the plaintext

Encrypting long messages

What if we want to send a message that is longer than the security parameter (i.e., $|m| > n$) ?

Idea:

- We have shown that any CPA-secure encryption scheme is also secure for multiple encryptions
- Think of m as the concatenation of many messages of length n, i.e., $m = m_1 \,\|\, m_2 \,\|\, m_3 \,\|\, \dots$
- Encrypt each m_i separately

Does it work? Yes, but...

- The ciphertext is (at least) twice as long as the plaintext
- The last chunk needs special handling (padding, truncating the output, ...)

Encrypting long messages

What if we want to send a message that is longer than the security parameter (i.e., $|m| > n$) ?

Idea:

- We have shown that any CPA-secure encryption scheme is also secure for multiple encryptions
- Think of m as the concatenation of many messages of length n, i.e., $m = m_1 \,\|\, m_2 \,\|\, m_3 \,\|\, \ldots$
- Encrypt each m_i separately

Does it work? Yes, but...

- The ciphertext is (at least) twice as long as the plaintext
- The last chunk needs special handling (padding, truncating the output, ...)

Can we do better?

Stream ciphers

Pseudorandom generators are not very convenient to work with:

Stream ciphers

Pseudorandom generators are not very convenient to work with:

- A PRG has a fixed output length
  (altough we can build a PRG with a bigger expansion factor from a PRG with a smaller expansion factor)

Stream ciphers

Pseudorandom generators are not very convenient to work with:

- A PRG has a fixed output length
  (altough we can build a PRG with a bigger expansion factor from a PRG with a smaller expansion factor)

- The whole output is produced in "one shot"

Stream ciphers

Pseudorandom generators are not very convenient to work with:

- A PRG has a fixed output length
  (altough we can build a PRG with a bigger expansion factor from a PRG with a smaller expansion factor)

- The whole output is produced in “one shot”

Practical construction of (candidate) PRGs are called **stream ciphers**

Stream ciphers

Pseudorandom generators are not very convenient to work with:

- A PRG has a fixed output length
  (altough we can build a PRG with a bigger expansion factor from a PRG with a smaller expansion factor)

- The whole output is produced in “one shot”

Practical construction of (candidate) PRGs are called **stream ciphers**

- They are able to produce an infinite stream of random bits, one at a time, on demand

Stream ciphers

Pseudorandom generators are not very convenient to work with:

- A PRG has a fixed output length
  (altough we can build a PRG with a bigger expansion factor from a PRG with a smaller expansion factor)

- The whole output is produced in "one shot"

Practical construction of (candidate) PRGs are called **stream ciphers**

- They are able to produce an infinite stream of random bits, one at a time, on demand
- They are easier to use and more flexible (e.g., they explicitly deal with IVs)

Stream ciphers

Pseudorandom generators are not very convenient to work with:

- A PRG has a fixed output length
  (altough we can build a PRG with a bigger expansion factor from a PRG with a smaller expansion factor)

- The whole output is produced in "one shot"

Practical construction of (candidate) PRGs are called **stream ciphers**

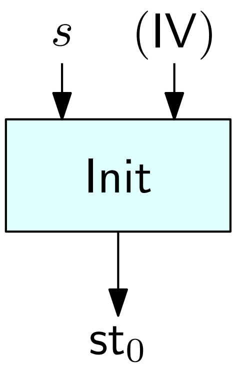
- They are able to produce an infinite stream of random bits, one at a time, on demand
- They are easier to use and more flexible (e.g., they explicitly deal with IVs)

Warning: Sometimes the term "stream cipher" is used to refer to the encryption scheme built from the actual stream cipher (as defined here)

Stream ciphers

A stream cipher is a pair of deterministic polynomial-time algorithms

- **Init:** takes a n-bit seed s, and possibly a n-bit *initialization vector* (IV), and outputs a *state* st_0

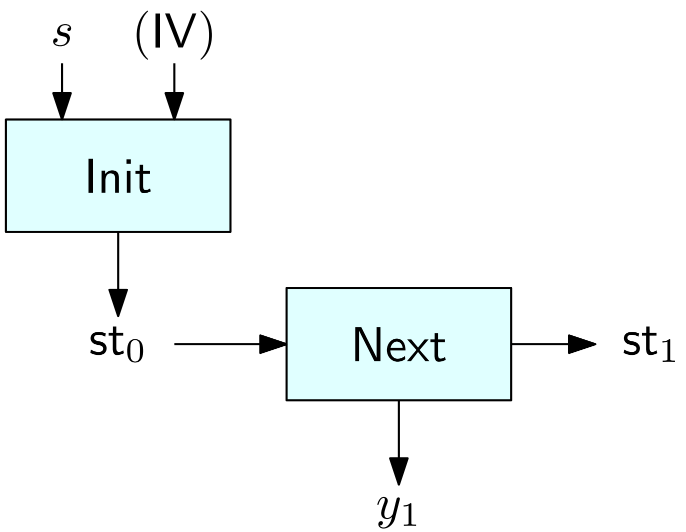


Stream ciphers

A stream cipher is a pair of deterministic polynomial-time algorithms

- **Init:** takes a n-bit seed s, and possibly a n-bit *initialization vector* (IV), and outputs a *state* st_0

- **Next:** takes a generic state st and outputs a bit y and a new (updated) state st'

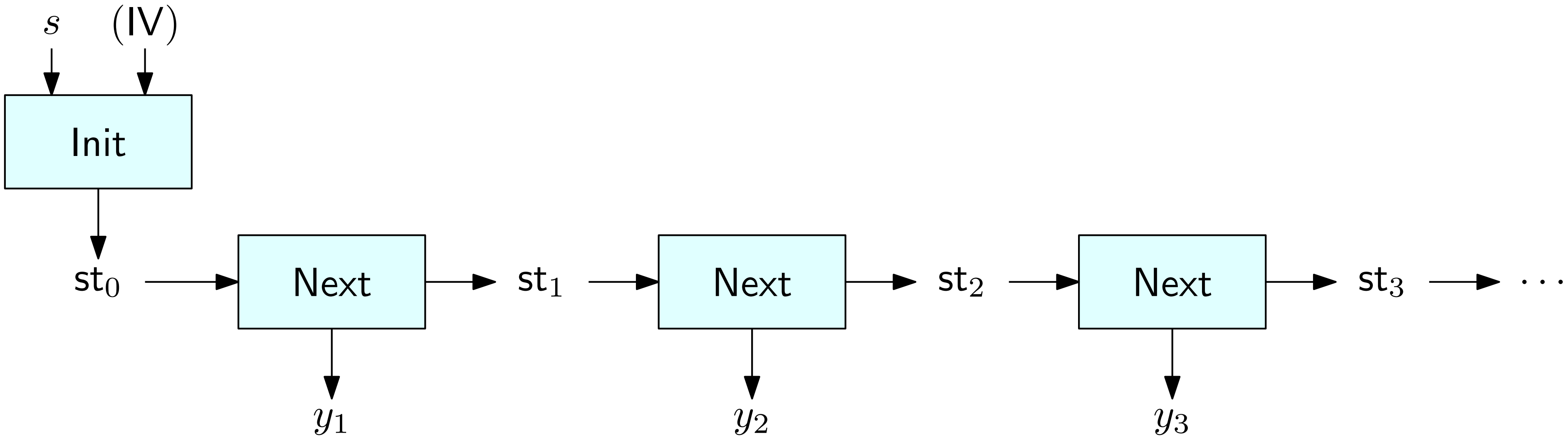


Stream ciphers

A stream cipher is a pair of deterministic polynomial-time algorithms

- **Init:** takes a n-bit seed s, and possibly a n-bit *initialization vector* (IV), and outputs a *state* st_0
- **Next:** takes a generic state st and outputs a bit y and a new (updated) state st$'$

Idea: we can generate as many random bits as we need, by repeatedly calling Next



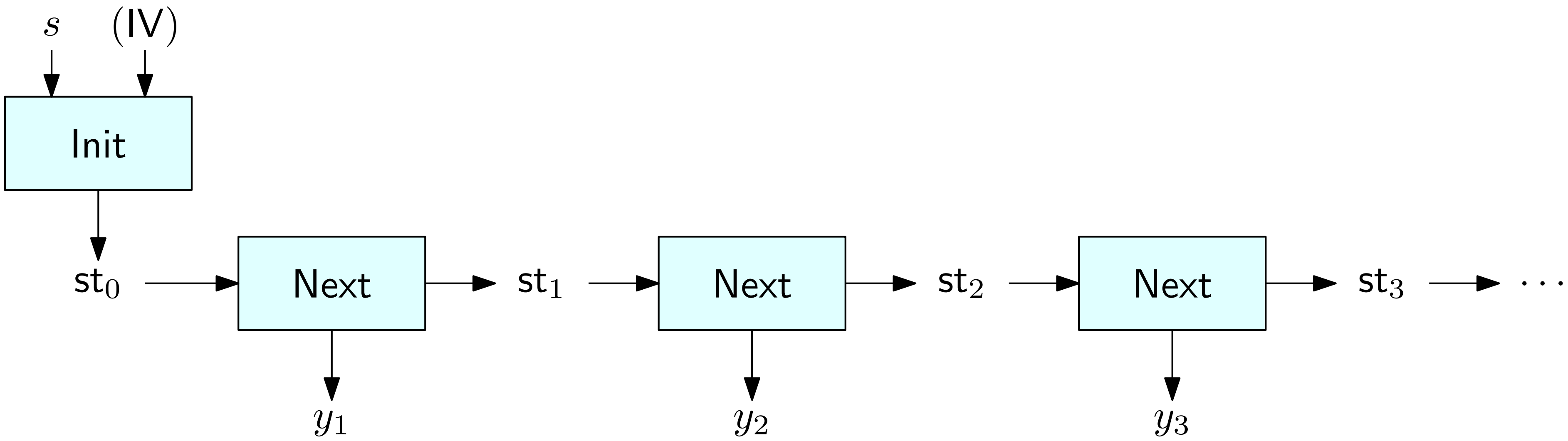
Stream ciphers

A stream cipher is a pair of deterministic polynomial-time algorithms

- **Init:** takes a n-bit seed s, and possibly a n-bit *initialization vector* (IV), and outputs a *state* st_0

- **Next:** takes a generic state st and outputs a bit y and a new (updated) state st'

Idea: we can generate as many random bits as we need, by repeatedly calling Next



* In practice, **Next** can output multiple bits at once (e.g., a byte)

Secure Stream Ciphers

A stream cipher is secure if the output stream generated by starting from a seed chosen u.a.r. is pseudorandom

- Any polynomial-length output stream is indistinguishable from a stream in which each bit is chosen u.a.r. in $\{0, 1\}$

Secure Stream Ciphers

A stream cipher is secure if the output stream generated by starting from a seed chosen u.a.r. is pseudorandom

- Any polynomial-length output stream is indistinguishable from a stream in which each bit is chosen u.a.r. in $\{0, 1\}$

Formally:

- Given a stream cipher $(\mathsf{Init}, \mathsf{Next})$, define the function $\mathsf{GetBits}(\mathsf{st}, 1^\ell)$ as the function that returns the pair (y, st_ℓ), where

Secure Stream Ciphers

A stream cipher is secure if the output stream generated by starting from a seed chosen u.a.r. is pseudorandom

- Any polynomial-length output stream is indistinguishable from a stream in which each bit is chosen u.a.r. in $\{0, 1\}$

Formally:

- Given a stream cipher $(\mathsf{Init}, \mathsf{Next})$, define the function $\mathsf{GetBits}(\mathsf{st}, 1^\ell)$ as the function that returns the pair (y, st_ℓ), where
  - $y = y_1 y_2 \dots y_\ell$ is the string of the random bits output by n successive calls of Next starting from state st

Secure Stream Ciphers

A stream cipher is secure if the output stream generated by starting from a seed chosen u.a.r. is pseudorandom

- Any polynomial-length output stream is indistinguishable from a stream in which each bit is chosen u.a.r. in $\{0,1\}$

Formally:

- Given a stream cipher $(\mathsf{Init}, \mathsf{Next})$, define the function $\mathsf{GetBits}(\mathsf{st}, 1^\ell)$ as the function that returns the pair (y, st_ℓ), where
  - $y = y_1 y_2 \ldots y_\ell$ is the string of the random bits output by n successive calls of Next starting from state st
  - st_ℓ is the state output by the final (i.e., ℓ-th) call to Next

Secure Stream Ciphers

A stream cipher is secure if the output stream generated by starting from a seed chosen u.a.r. is pseudorandom

- Any polynomial-length output stream is indistinguishable from a stream in which each bit is chosen u.a.r. in $\{0, 1\}$

Formally:

- Given a stream cipher $(\mathsf{Init}, \mathsf{Next})$, define the function $\mathsf{GetBits}(\mathsf{st}, 1^\ell)$ as the function that returns the pair (y, st_ℓ), where
  - $y = y_1 y_2 \dots y_\ell$ is the string of the random bits output by n successive calls of Next starting from state st
  - st_ℓ is the state output by the final (i.e., ℓ-th) call to Next

If the stream cipher does not use IVs:

- Define the function $G^\ell(s)$ (from $\{0, 1\}^n$ to $\{0, 1\}^\ell$) as the string y of $\mathsf{GetBits}(\mathsf{Init}(s), 1^\ell)$
- The stream cipher is **secure** if $G^\ell(s)$ is a **pseudorandom generator** for any polynomial ℓ

Secure Stream Ciphers

A stream cipher is secure if the output stream generated by starting from a seed chosen u.a.r. is pseudorandom

- Any polynomial-length output stream is indistinguishable from a stream in which each bit is chosen u.a.r. in $\{0, 1\}$

Formally:

- Given a stream cipher $(\mathsf{Init}, \mathsf{Next})$, define the function $\mathsf{GetBits}(\mathsf{st}, 1^\ell)$ as the function that returns the pair (y, st_ℓ), where
  - $y = y_1 y_2 \dots y_\ell$ is the string of the random bits output by n successive calls of Next starting from state st
  - st_ℓ is the state output by the final (i.e., ℓ-th) call to Next

If the stream cipher uses IVs:

- Define the function $F_s^\ell(\mathsf{IV})$ (from $\{0, 1\}^n \times \{0, 1\}^n$ to $\{0, 1\}^\ell$) as the string y of $\mathsf{GetBits}(\mathsf{Init}(s, \mathsf{IV}), 1^\ell)$

- The stream cipher is **secure** if $F_s^\ell(\mathsf{IV})$ is a **pseudorandom function** for any polynomial ℓ

Secure Stream ciphers with IVs from Pseudorandom Functions

If we have a pseudorandom function $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$, we can use it to build a stream cipher that takes an initialization vector

Secure Stream ciphers with IVs from Pseudorandom Functions

If we have a pseudorandom function $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$, we can use it to build a stream cipher that takes an initialization vector

The idea is similar to that used to construct a PRG from a PRF

Secure Stream ciphers with IVs from Pseudorandom Functions

If we have a pseudorandom function $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$, we can use it to build a stream cipher that takes an initialization vector

The idea is similar to that used to construct a PRG from a PRF

The resulting stream cipher will use a n-bit seed, a $3n/4$-bits IV, and will output n bits per call to Next (this is an example, these parameters can be tuned)

Secure Stream ciphers with IVs from Pseudorandom Functions

If we have a pseudorandom function $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$, we can use it to build a stream cipher that takes an initialization vector

The idea is similar to that used to construct a PRG from a PRF

The resulting stream cipher will use a n-bit seed, a $3n/4$-bits IV, and will output n bits per call to Next (this is an example, these parameters can be tuned)

Init(s, IV):

- Output $(s, \mathsf{IV}, 0)$

Secure Stream ciphers with IVs from Pseudorandom Functions

If we have a pseudorandom function $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$, we can use it to build a stream cipher that takes an initialization vector

The idea is similar to that used to construct a PRG from a PRF

The resulting stream cipher will use a n-bit seed, a $3n/4$-bits IV, and will output n bits per call to Next (this is an example, these parameters can be tuned)

Init(s, IV):

- Output $(s, \mathsf{IV}, 0)$

Next(st):

- Unpack the state st in $(s, \mathsf{IV}, \langle i \rangle)$
- Output the n bits $F_s(\mathsf{IV} \,\|\, \langle i \rangle)$ and the new state $(s, \mathsf{IV}, \langle i+1 \rangle)$

Secure Stream ciphers with IVs from Pseudorandom Functions

If we have a pseudorandom function $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$, we can use it to build a stream cipher that takes an initialization vector

The idea is similar to that used to construct a PRG from a PRF

The resulting stream cipher will use a n-bit seed, a $3n/4$-bits IV, and will output n bits per call to Next (this is an example, these parameters can be tuned)

Init(s, IV):

- Output $(s, \mathsf{IV}, 0)$

Next(st):

- Unpack the state st in $(s, \mathsf{IV}, \langle i \rangle)$
- Output the n bits $F_s(\mathsf{IV} \,\|\, \langle i \rangle)$ and the new state $(s, \mathsf{IV}, \langle i+1 \rangle)$

$\langle i \rangle$ = Binary encoding of i using $n/4$ bits

Modes of operation of Stream Ciphers

We can use stream ciphers in two different **modes of operation**

Modes of operation of Stream Ciphers

We can use stream ciphers in two different **modes of operation**

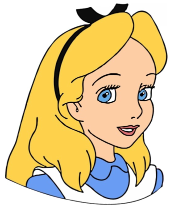
- **Synchronized mode**: The sender and receiver each maintain a state, which must be kept synchronized between messages
  - Useful for short communication sessions. Each message must be delivered exactly once and all messages must be received in order
  - Example: data exchanged over a TCP connection
  - Does not need to use IVs, Ciphertext length = message length

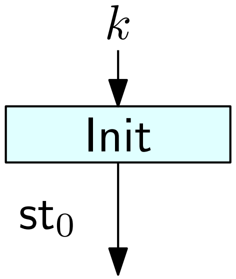
Modes of operation of Stream Ciphers

We can use stream ciphers in two different **modes of operation**

- **Synchronized mode**: The sender and receiver each maintain a state, which must be kept synchronized between messages
  - Useful for short communication sessions. Each message must be delivered exactly once and all messages must be received in order
  - Example: data exchanged over a TCP connection
  - Does not need to use IVs, Ciphertext length = message length
- **Unsynchronized mode**: The sender and receiver do not need to store any information during the communication session (i.e., they are stateless)
  - Useful for long messages, and communication over a long period of time. Does not require messages to be delivered in order
  - Each message uses its own IV
  - Needs IVs, Ciphertext length = message length + IV length ($\approx$ message length for long messages)

Synchronized mode

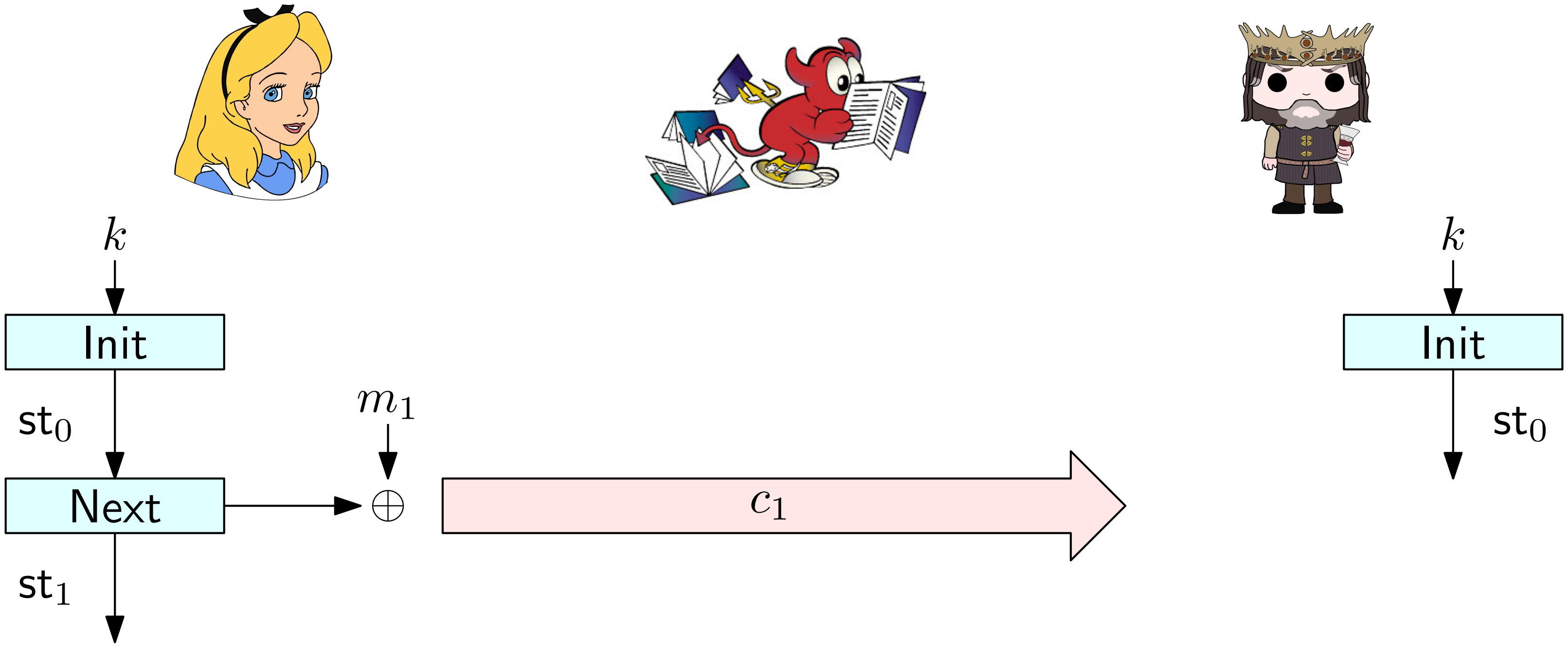




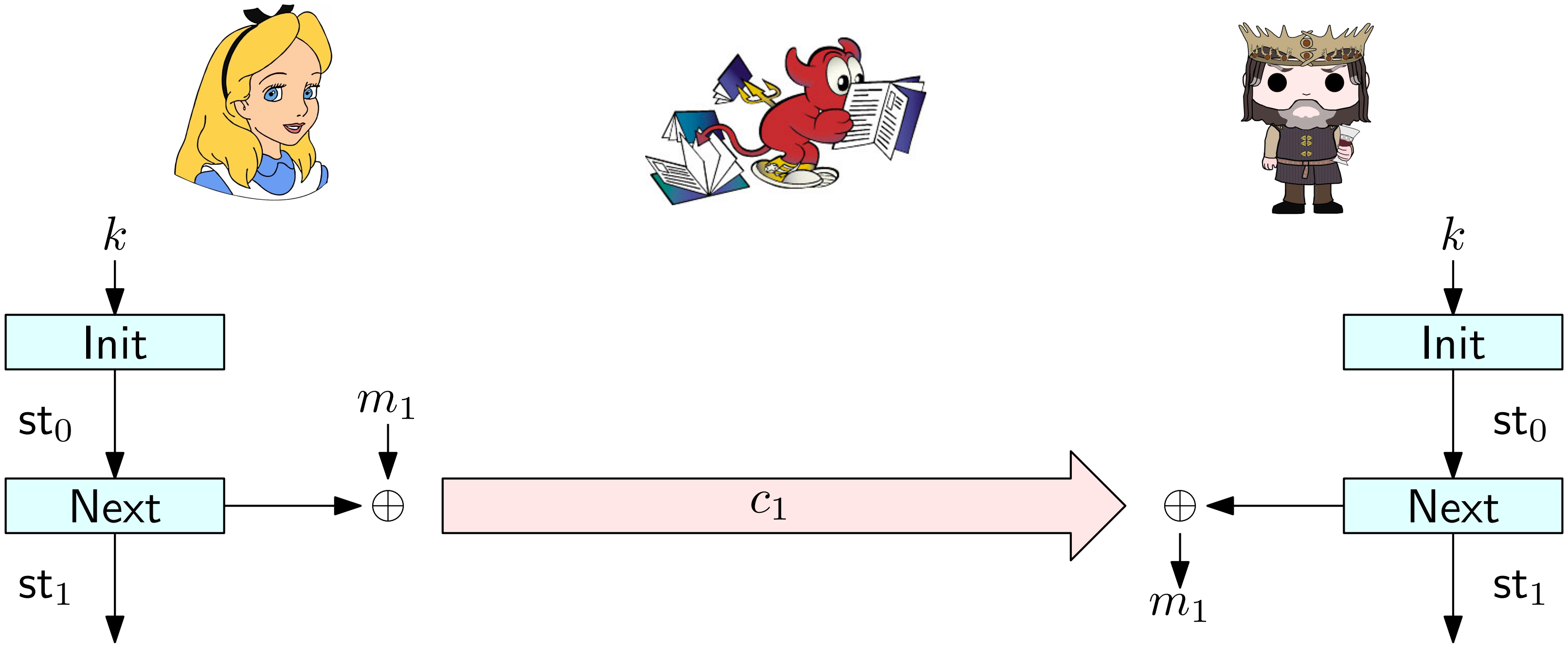
Synchronized mode

Synchronized mode

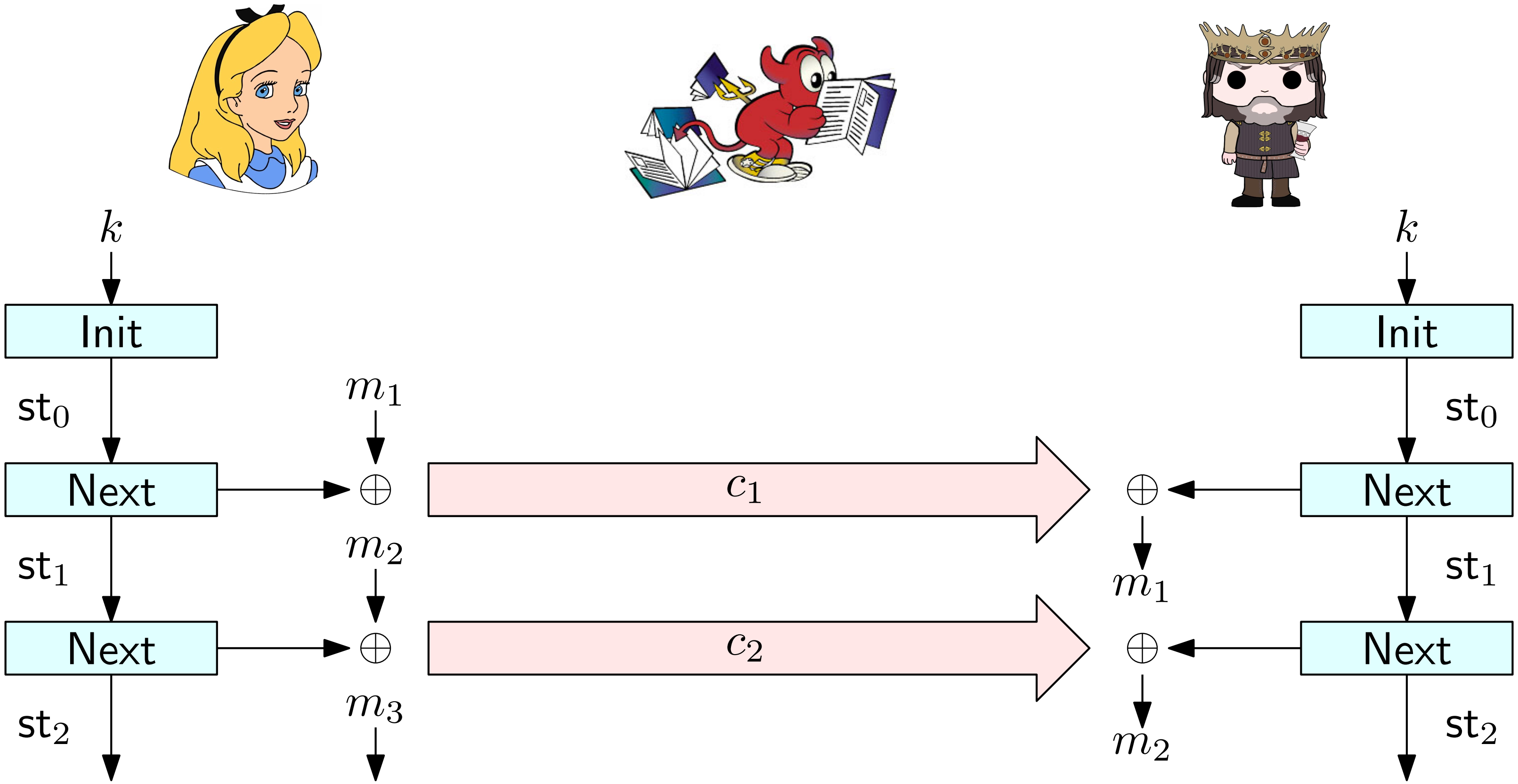
Synchronized mode



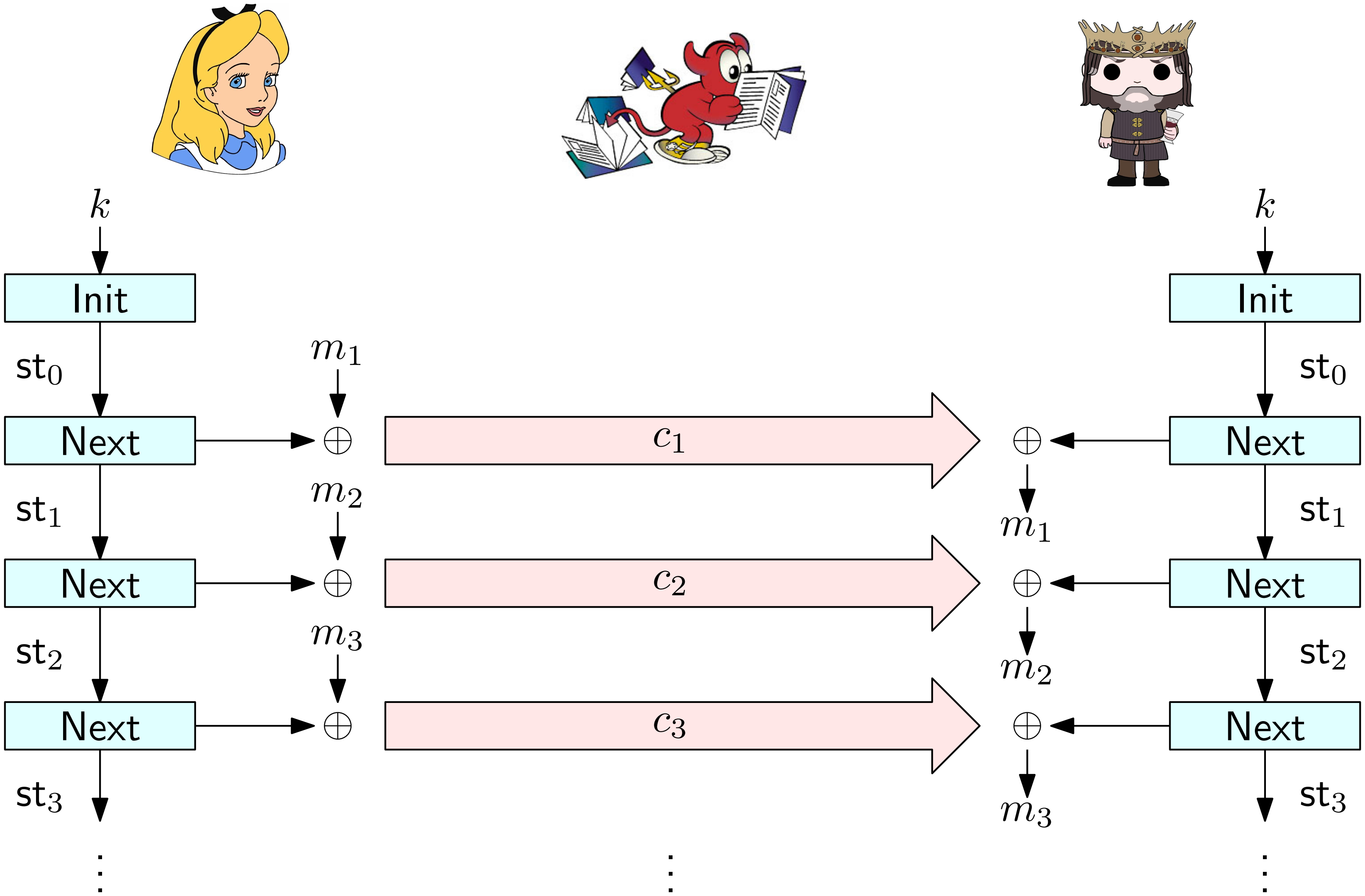
Synchronized mode



Synchronized mode



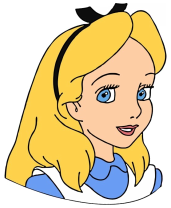
Synchronized mode

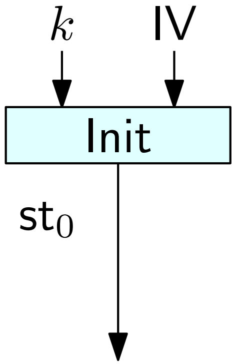


Alice & Bob need to keep track of the last state for as long as they wish to communicate

Unsynchronized mode

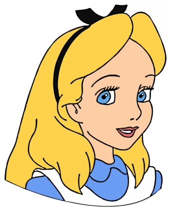
Alice picks a random IV

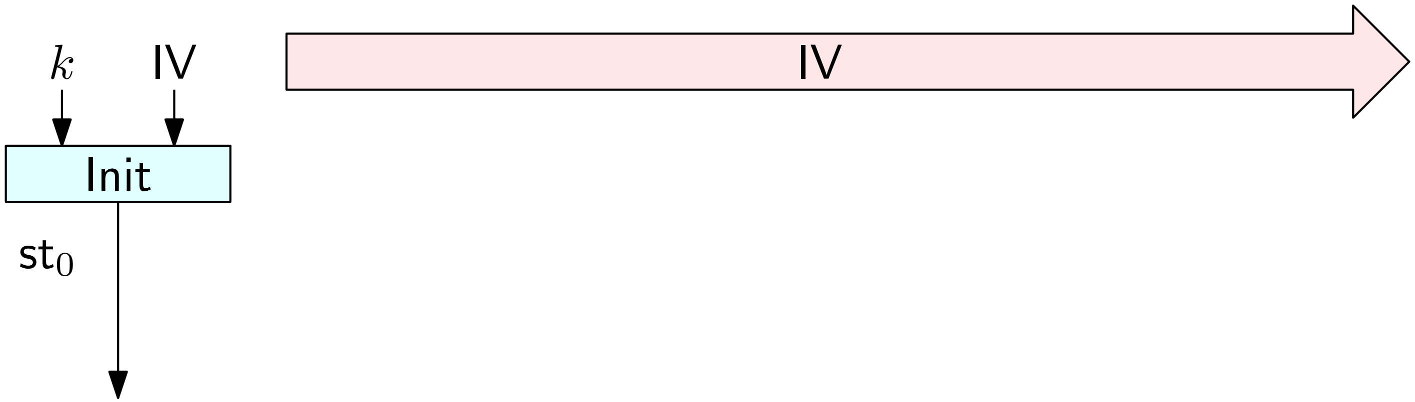




Unsynchronized mode

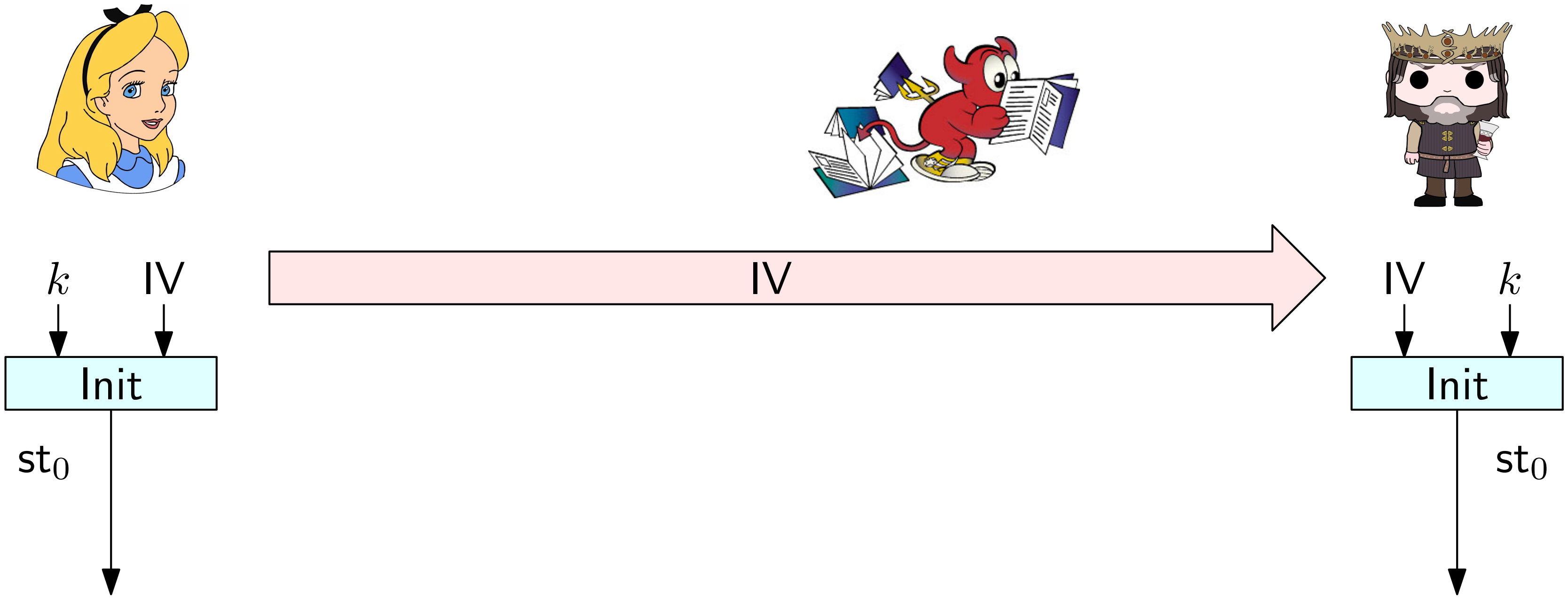
Alice picks a random IV





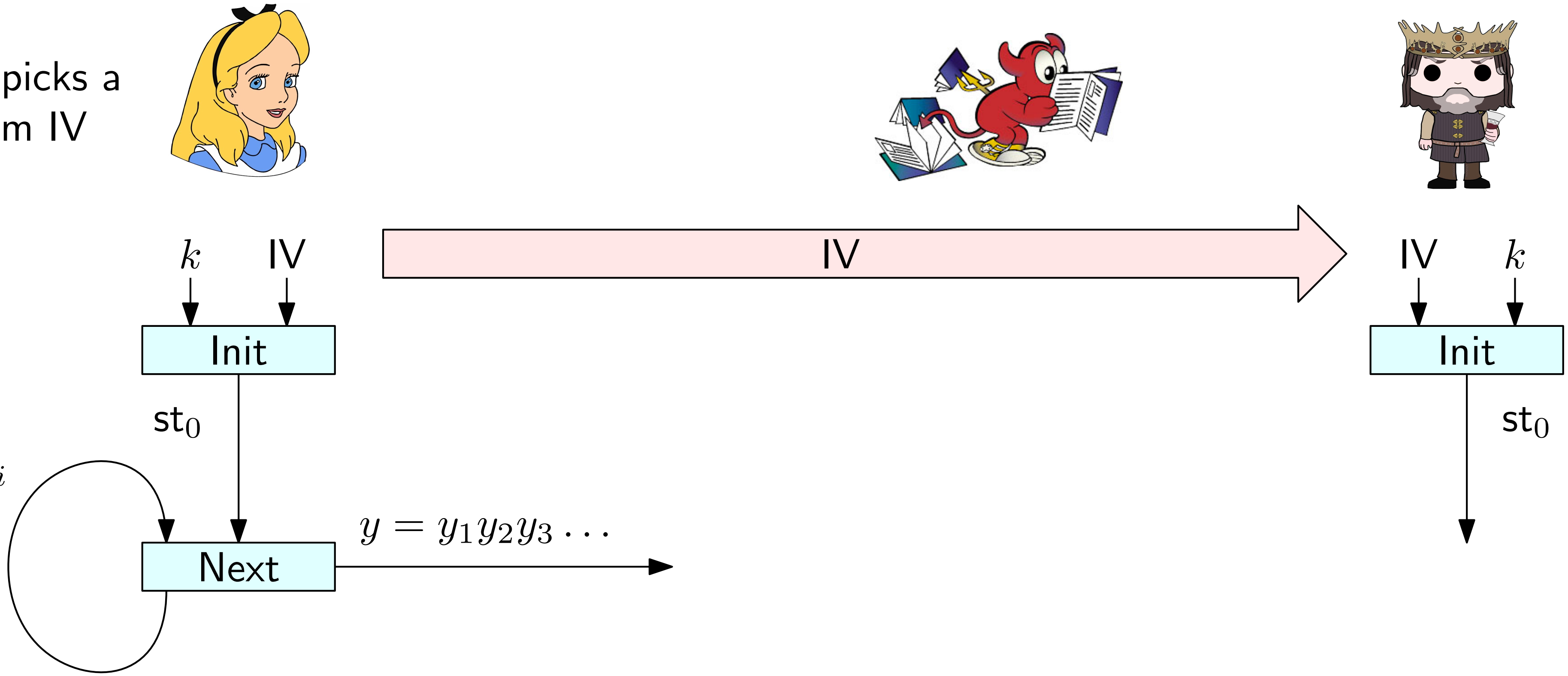
Unsynchronized mode

Alice picks a random IV



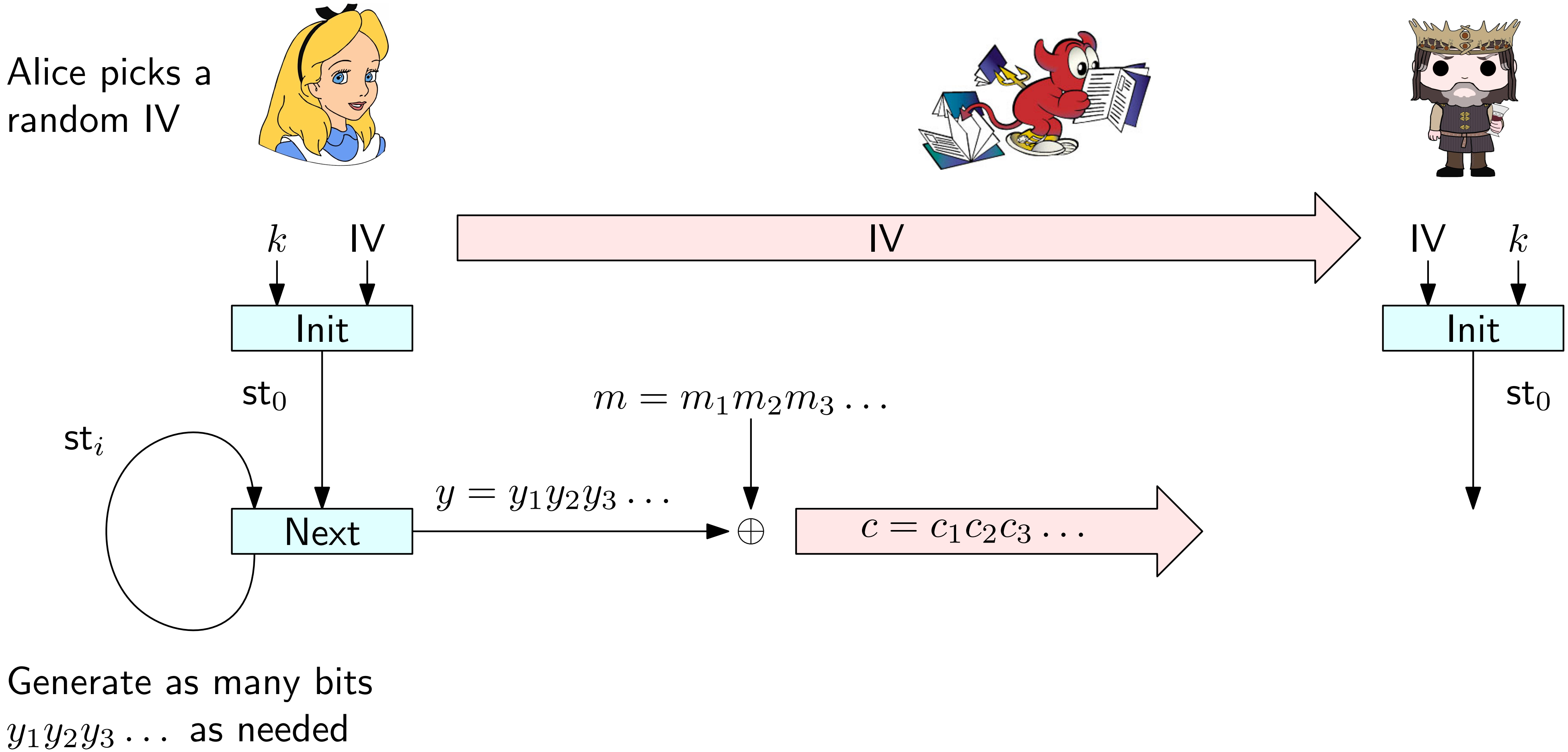
Unsynchronized mode

Alice picks a random IV

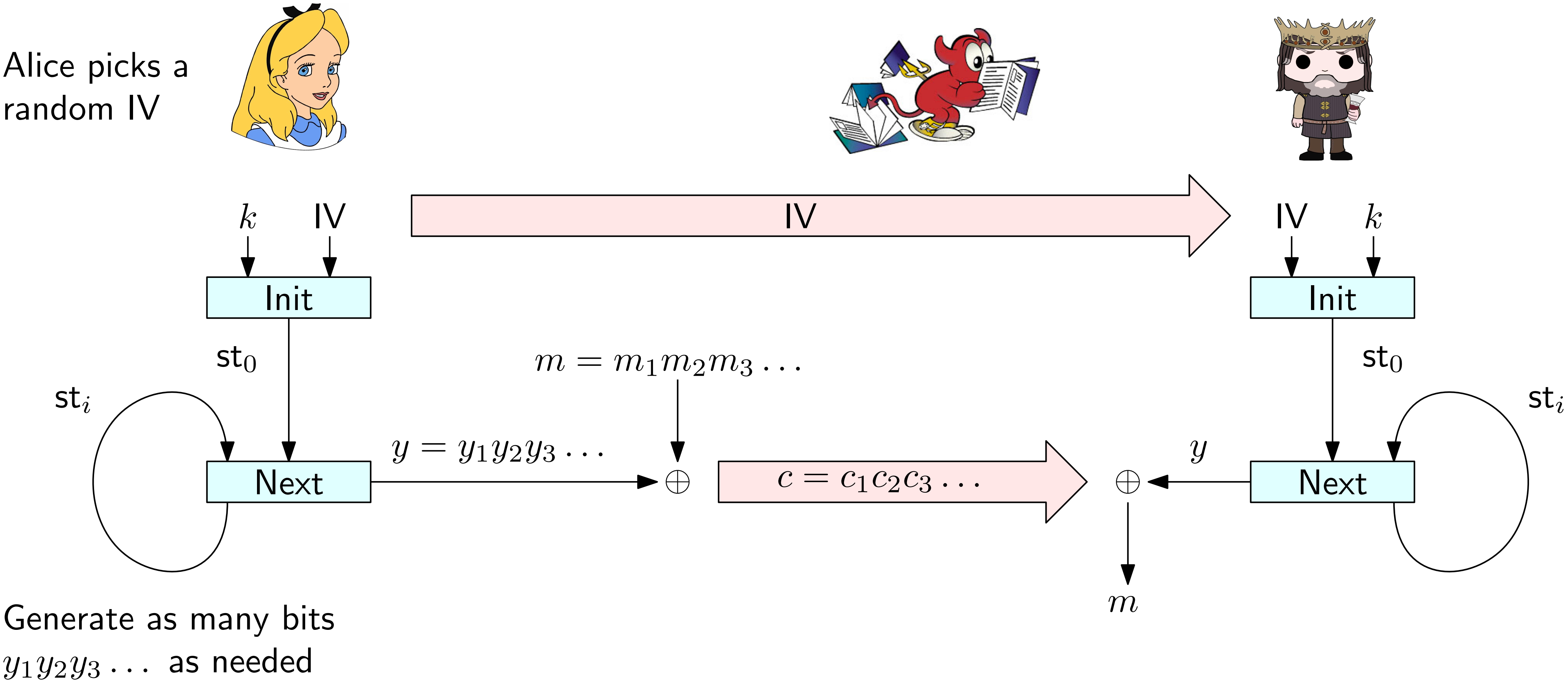


Generate as many bits
$y_1 y_2 y_3 \ldots$ as needed

Unsynchronized mode



Unsynchronized mode



Stream ciphers

- A stream cipher without IV can be thought of as a more convenient interface to a PRG
- A stream cipher with IV can be thought of as a more convenient interface to a PRF

Stream ciphers

- A stream cipher without IV can be thought of as a more convenient interface to a PRG
- A stream cipher with IV can be thought of as a more convenient interface to a PRF
- We don't know if (secure) stream ciphers exist (we don't know if PRGs / PRFs exist)

Stream ciphers

- A stream cipher without IV can be thought of as a more convenient interface to a PRG
- A stream cipher with IV can be thought of as a more convenient interface to a PRF
- We don't know if (secure) stream ciphers exist (we don't know if PRGs / PRFs exist)
- In practice we have some candidate stream cipher constructions that are conjectured to be secure
- These construction have withstood years of public scrutiny and attempted cryptanalysis

Stream ciphers

- A stream cipher without IV can be thought of as a more convenient interface to a PRG
- A stream cipher with IV can be thought of as a more convenient interface to a PRF
- We don't know if (secure) stream ciphers exist (we don't know if PRGs / PRFs exist)
- In practice we have some candidate stream cipher constructions that are conjectured to be secure
- These construction have withstood years of public scrutiny and attempted cryptanalysis

- Some popular practical constructions of stream ciphers:
  - Trivium: optimized for hardware
  - RC4 (insecure): optimized for software
  - ChaCha20: replacement of RC4