Cryptography

Lecture 7: Operation Modes

November 26, 2024

Overview

Contents

1. Classical cryptography
   (Shift & Vigenère cipher, one-time pad, perfect secrecy)
2. Security definitions & threat models
   (Computational security, CPA & CCA)
3. Private-key cryptography
   (Message authentication, hash functions, primitives, relevant ciphers)
4. Public-key cryptography
   (Assumptions, key management, digital signatures, relevant ciphers)

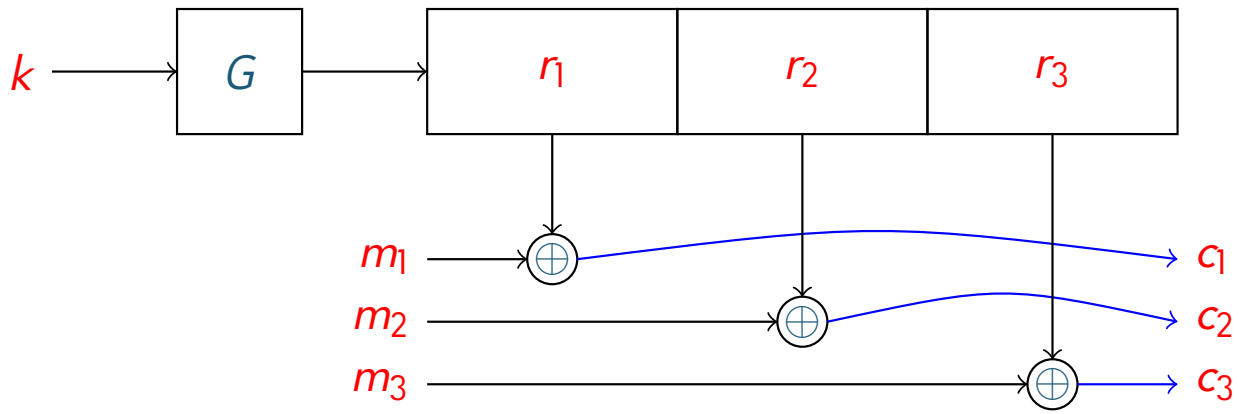
Operation Modes of Symmetric Ciphers

1. Stream ciphers
2. Block ciphers

Stream Cipher Operation Modes

Pseudorandom generator G with $G(k) = r_1 r_2 r_3$

Synchronized



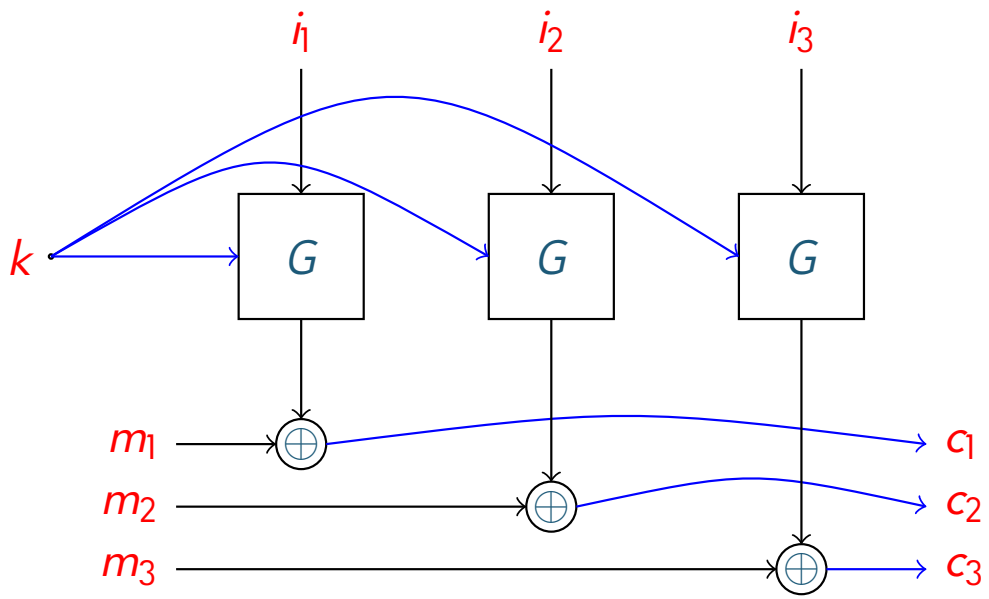
EAV-secure	CPA-secure
✓	✗

(deterministic cipher)

Stream Cipher Operation Modes

Pseudorandom function G (stream cipher with random initialization vector)

Unsynchronized

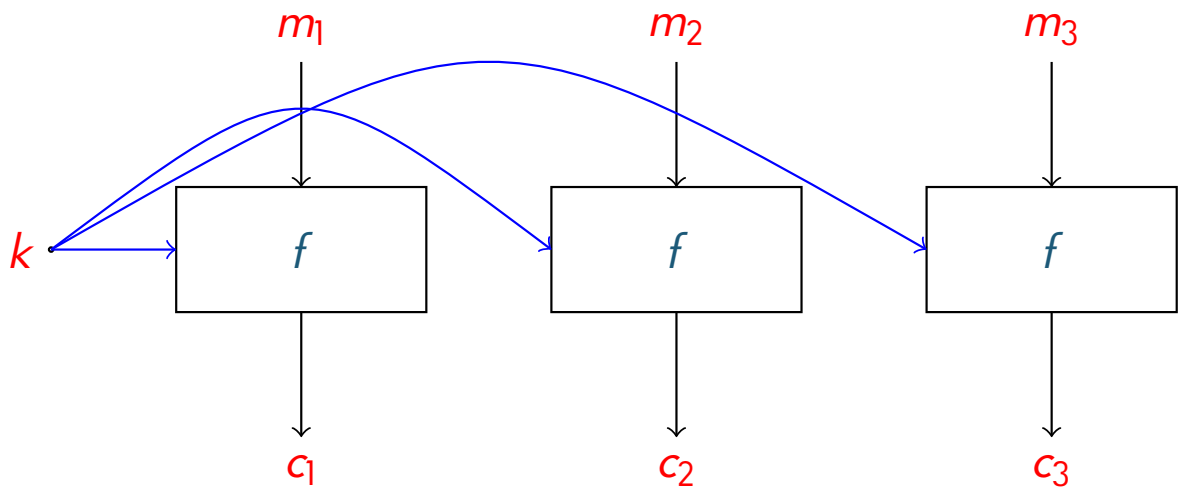


EAV-secure	CPA-secure
✓	✓

Block Cipher Operation Modes

Pseudorandom permutation f and message $m = m_1 m_2 m_3$
(= pseudorandom function with f_k bijective/permutation for each k)

Electronic Code Book (ECB)



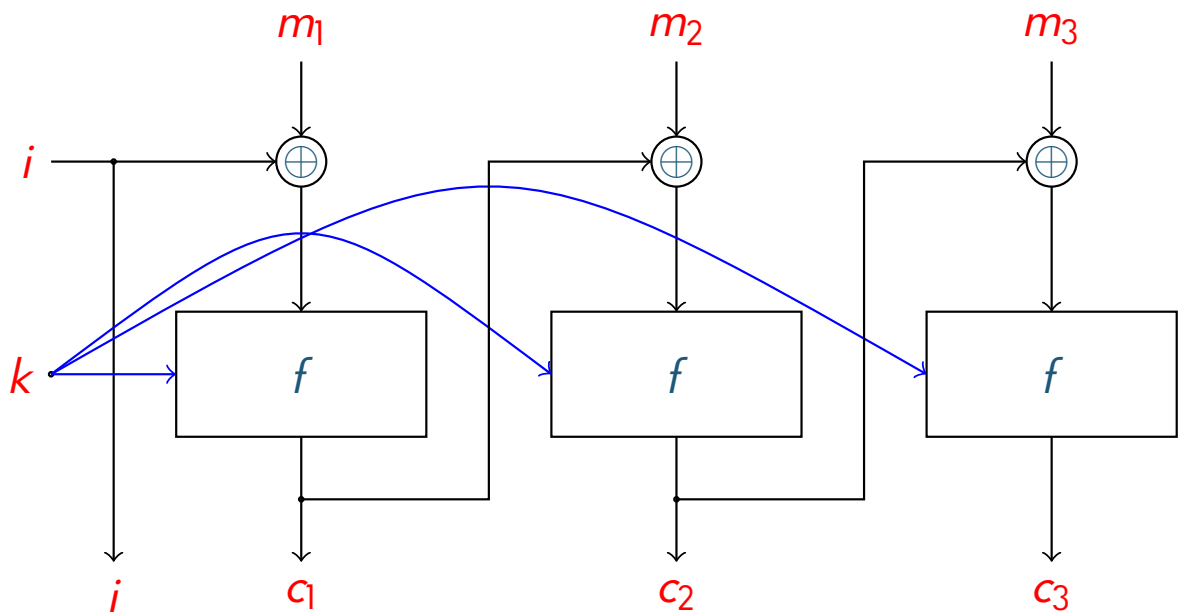
EAV-secure	CPA-secure
✗	✗

(deterministic cipher, multiple messages for same key)

Chosen-Plaintext-Attack Security

Pseudorandom permutation f, message $m = m_1 m_2 m_3$, and random i

Cipher Block Chaining (CBC)



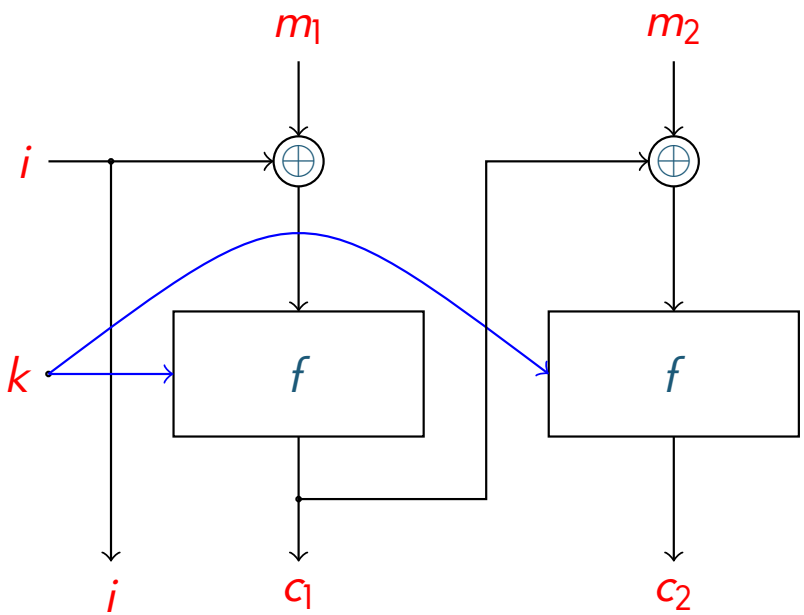
EAV-secure	CPA-secure
✓	✓

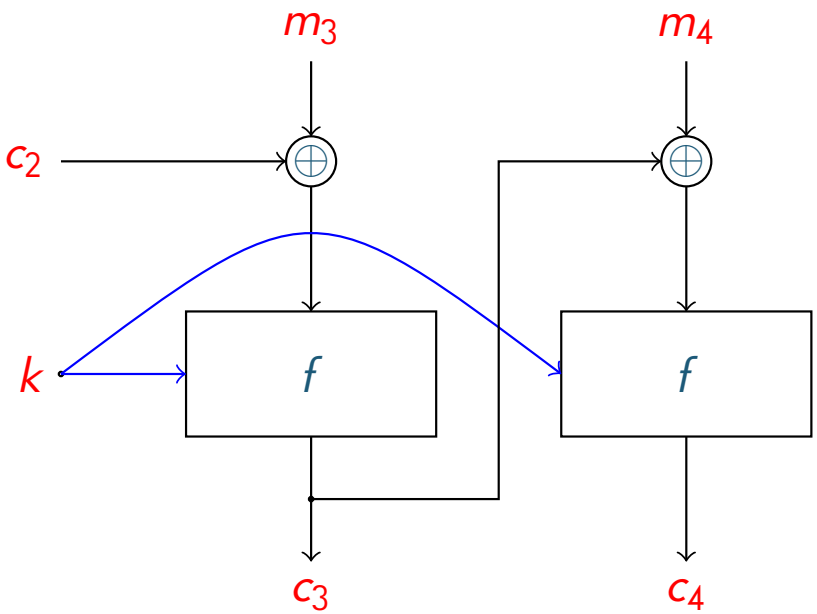
Chosen-Plaintext-Attack Security

Pseudorandom permutation f, messages $m = m_1 m_2$ and $m' = m_3 m_4$

Chained Cipher Block Chaining (cCBC)

(as used in SSL 3.0 or TLS 1.0)





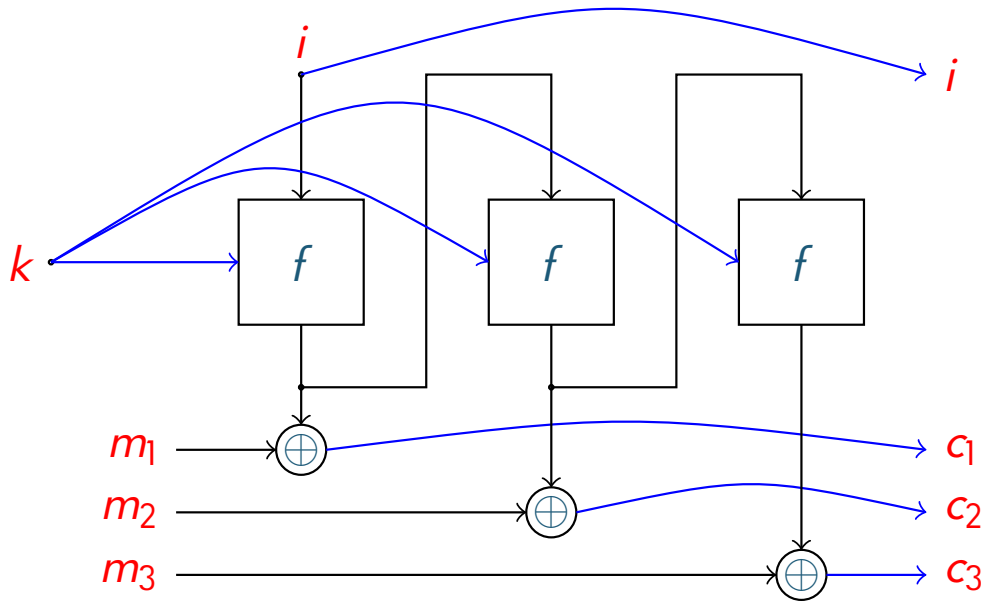
EAV-secure	CPA-secure
✓	✗

(deterministic cipher starting at 2nd message)

Chosen-Plaintext-Attack Security

Pseudorandom function f, message $m = m_1 m_2 m_3$, and random i

Output Feedback (OFB)

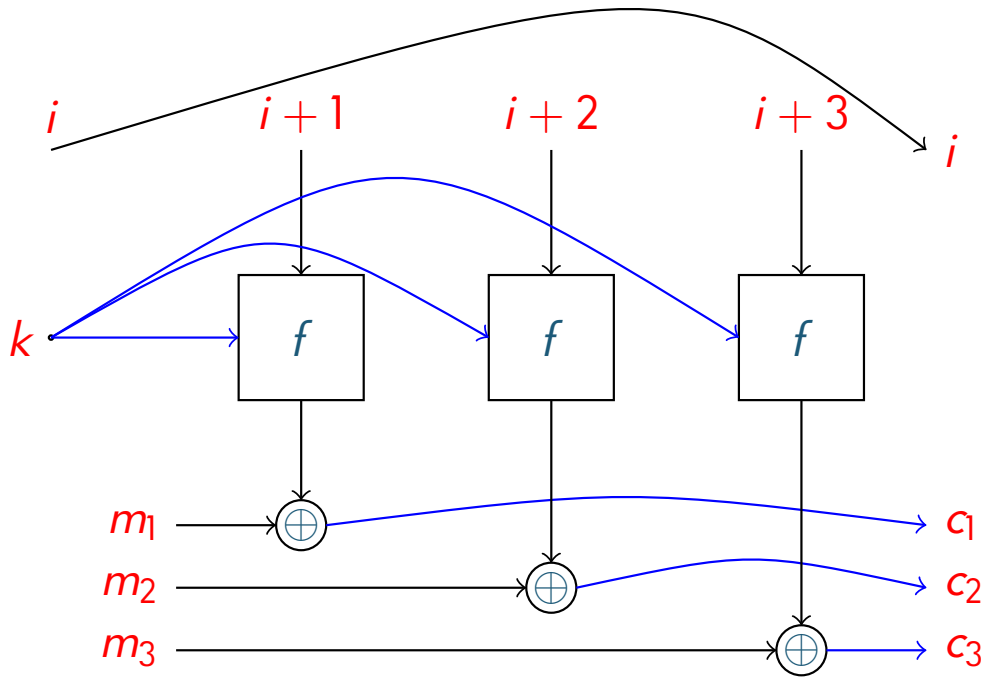


EAV-secure	CPA-secure
✓	✓

Chosen-Plaintext-Attack Security

Pseudorandom function f, message $m = m_1 m_2 m_3$, and random i

Counter (CTR)



EAV-secure	CPA-secure
✓	✓

Chosen-Ciphertext-Attack Security

Oracles (recall)

- Encryption oracle provides access to enc_k for currently used key k
  (given message m returns ciphertext c with $\mathrm{P}[c \leftarrow \mathrm{enc}_k(m)] \neq 0$)
- Decryption oracle provides access to dec_k for currently used key k
  (given ciphertext c returns message $m = \mathrm{dec}_k(c)$)
- Do not predict used random bits (uses its own random bits)

§7.1 Definition (CCA-secure)

Private-key encryption scheme $\mathcal{E} = (\mathrm{gen}, \mathrm{enc}, \mathrm{dec})$ is CCA-secure
if for every PPT algorithm $\mathcal{A}$ with access to encryption & decryption oracle

$$\mathrm{P}\left[\mathrm{PrivK}^{\mathrm{one}}_{\mathcal{E},\mathcal{A}}(n)\right] \simeq \frac{1}{2}$$

where $\mathcal{A}$ must not query decryption oracle on challenge ciphertext
after it is received by $\mathcal{A}$ (before receipt $\mathcal{A}$ may ask any query)

Chosen-Ciphertext-Attack Security

Construction (§6.3)

Let f be pseudorandom function. Scheme $\mathcal{E}_f = (\mathrm{gen}, \mathrm{enc}, \mathrm{dec})$ with

- $\mathrm{P}^n_k(k) = 2^{-n}$ for all $k \in \{0,1\}^n$
- $\mathrm{enc}_k(m)$ is $r \leftarrow U_n$; return $\langle r, f_k(r) \oplus m \rangle$ for all $k, m \in \{0,1\}^n$
- $\mathrm{dec}_k(\langle r, w \rangle) = f_k(r) \oplus w$ for all $k, r, w \in \{0,1\}^n$

Example adversary

1. Receive security parameter 1^n
2. Select messages $m_0 = 0^n$ and $m_1 = 1^n$, which fulfill $|m_0| = n = |m_1|$
3. Receive ciphertext $\langle r, s \rangle$ with $s = f_k(r) \oplus m_b$ for unknown $b \in \{0,1\}$
4. Query decryption oracle on $\langle r, \overline{s} \rangle \neq \langle r, s \rangle$, which is permitted and receive $m' = f_k(r) \oplus \overline{s} = f_k(r) \oplus s \oplus m_1 = f_k(r) \oplus f_k(r) \oplus m_b \oplus m_1$
5. Return $b' = (m' \stackrel{?}{=} m_0)$, which is always correct (i.e. $b = b'$)

$\rightarrow \mathcal{E}_f$ not CCA-secure

Chosen-Ciphertext-Attack Security

Notes

- No discussed scheme CCA-secure
- Seems outrageously unrealistic (and full power certainly is)
- Illustration of common padding attack

PKCS #5

- Block cipher block length L and $\ell = |m| \bmod L$ for message m
- Pad last block with $L - \ell$ times byte $L - \ell$ if $L - \ell \neq 0$
  (if $L - \ell = 0$ then add full block of bytes L)
- Example: $\ell = 4$ and $L = 8$

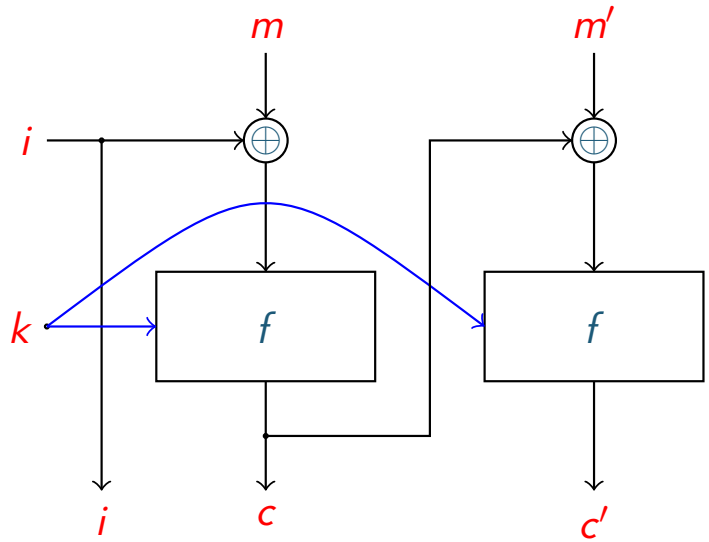
$$(134, 232, 45, 25) \rightarrow (134, 232, 45, 25, 4, 4, 4, 4)$$

- Abuse return of "padding error" of service
  (`javax.crypto.BadPaddingException`)

Chosen-Ciphertext-Attack Security

Illustration against CBC

1. Let ciphertext i, c, c' and $x = 1$
2. Modify x-th byte of c to obtain c_x
3. Send i, c_x, c' to receiver
   (x-th byte of m' will change)
4. If padding error $\rightarrow$ pad starts at x
5. Otherwise increase x and back to 2



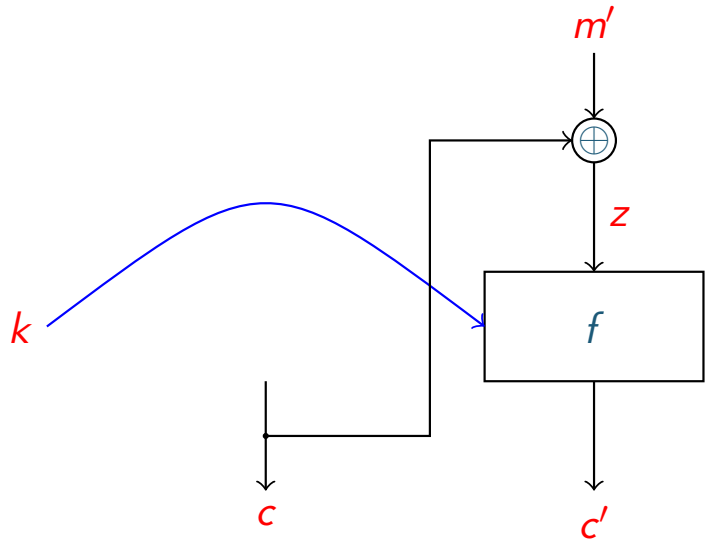
Determine message

- Know last y bytes of m' are y; determine byte b before those
- Modify last y bytes of c by adding $y \oplus (y+1)$; let byte before be b'
- Try all values z instead of b' before last y bytes of c
- If no padding error $\rightarrow$ $(y+1) \oplus z = b \oplus b'$, so $b = (y+1) \oplus z \oplus b'$

Chosen-Ciphertext-Attack Security

Example determining pad length

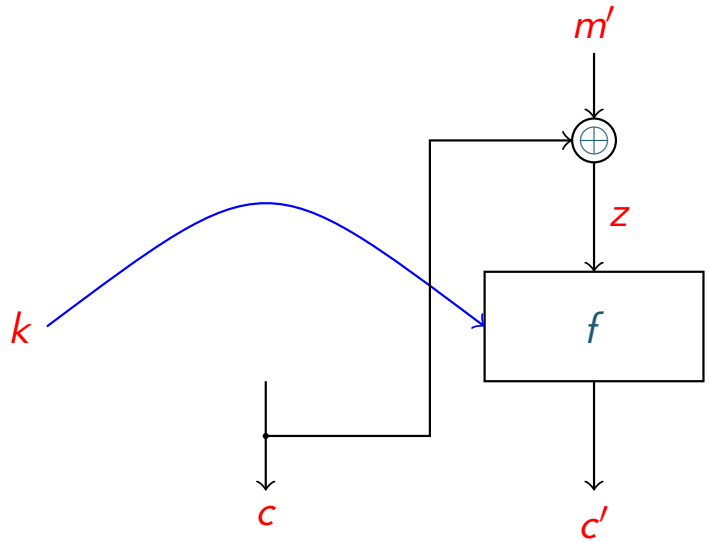
- $z = f_k^{-1}(c')$ unknown to adversary
- $m' = z \oplus c$
- Let $m' = (152, 187, 2, 2)$
  $c = (\underline{27}, \underline{202}, \underline{143}, \underline{9})$
  $z = (131, 113, 141, 11)$
- Send $c_1 = (\underline{228}, \underline{202}, \underline{143}, \underline{9})$ and $\underline{c'}$
- z remains and $m^{(1)} = (103, 187, 2, 2)$ remains correctly padded
- Send $c_3 = (\underline{27}, \underline{202}, \underline{112}, \underline{9})$ and $\underline{c'}$
- Then $m^{(3)} = (152, 187, 253, 2)$ incorrectly padded



Chosen-Ciphertext-Attack Security

Example determining message

- $z = f_k^{-1}(c')$ unknown to adversary
- $m' = z \oplus c$
- Let $m' = (152, 187, \underline{2}, \underline{2})$
  $c = (\underline{27}, \underline{202}, \underline{143}, \underline{9})$
  $z = (131, 113, \underline{141}, \underline{11})$
- Send $c^{(0)} = (\underline{27}, \underline{0}, \underline{142}, \underline{8})$ and $\underline{c'}$
- z remains and $m^{(0)} = (103, 113, \underline{3}, \underline{3})$ incorrectly padded
- Send $c^{(114)} = (\underline{27}, \underline{114}, \underline{142}, \underline{8})$ and $\underline{c'}$
- $m^{(114)} = (103, \underline{3}, \underline{3}, \underline{3})$ correct pad
  so $m'_2 = 202 \oplus 114 \oplus 3 = 187$



Overview

Contents

1. Classical cryptography
   (Shift & Vigenère cipher, one-time pad, perfect secrecy)
2. Security definitions & threat models
   (Computational security, CPA & CCA)
3. Private-key cryptography
   (Message authentication, hash functions, primitives, relevant ciphers)
4. Public-key cryptography
   (Assumptions, key management, digital signatures, relevant ciphers)

Proposals for Pseudorandom Generators — RC4

RC4

- Stream cipher “Rivest Cipher 4” aka “Ron’s Code 4”
- Was extremely popular due to its speed & simplicity
- IETF (Internet Engineering Task Force) mandates that RC4 must not be used in TLS since 2015
- Utilizes standard construction, so we only describe generator

Ron Rivest (* 1947)

- US cryptographer
- Professor at MIT & Turing award laureate
- Co-developed RSA, RC1–6 & MD(2,4,6)

© Ron Rivest

Proposals for Pseudorandom Generators — RC4

Characteristics of RC4

- Initial vector is 256 bytes $= 2,048$ bits
  (default initial vector is array $(0, 1, \ldots, 255)$ of bytes)
- States are 258 bytes $= 2,064$ bits
- Seed length $\ell \in \{1, \ldots, 256\}$ in bytes
- Generates 1 byte $= 8$ bits per round

Initialization with seed s and initial vector v

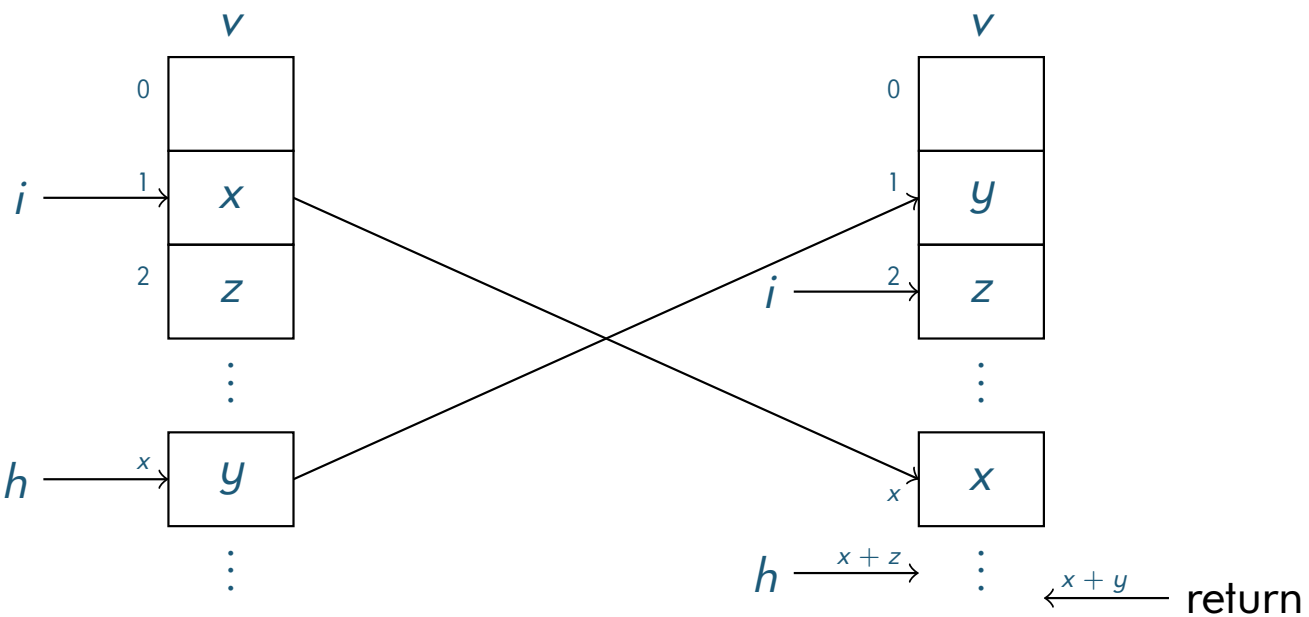
1. $h \leftarrow 0$
2. For all $i \in \{0, 1, \ldots, 255\}$ in ascending order
   1. $h \leftarrow (h + v[i] + s[i \bmod \ell]) \bmod 256$
   2. Swap values $v[i]$ and $v[h]$
3. Return $(v, 0, 0)$

Proposals for Pseudorandom Generators — RC4

Next byte generation with state (v, i, h)

1. $i \leftarrow i + 1 \bmod 256$
2. $h \leftarrow h + v[i] \bmod 256$
3. Swap values $v[i]$ and $v[h]$
4. Return output bits $v\big[v[i] + v[h] \bmod 256\big]$ and state (v, i, h)

Illustration for $i = 0$ and $h = 0$

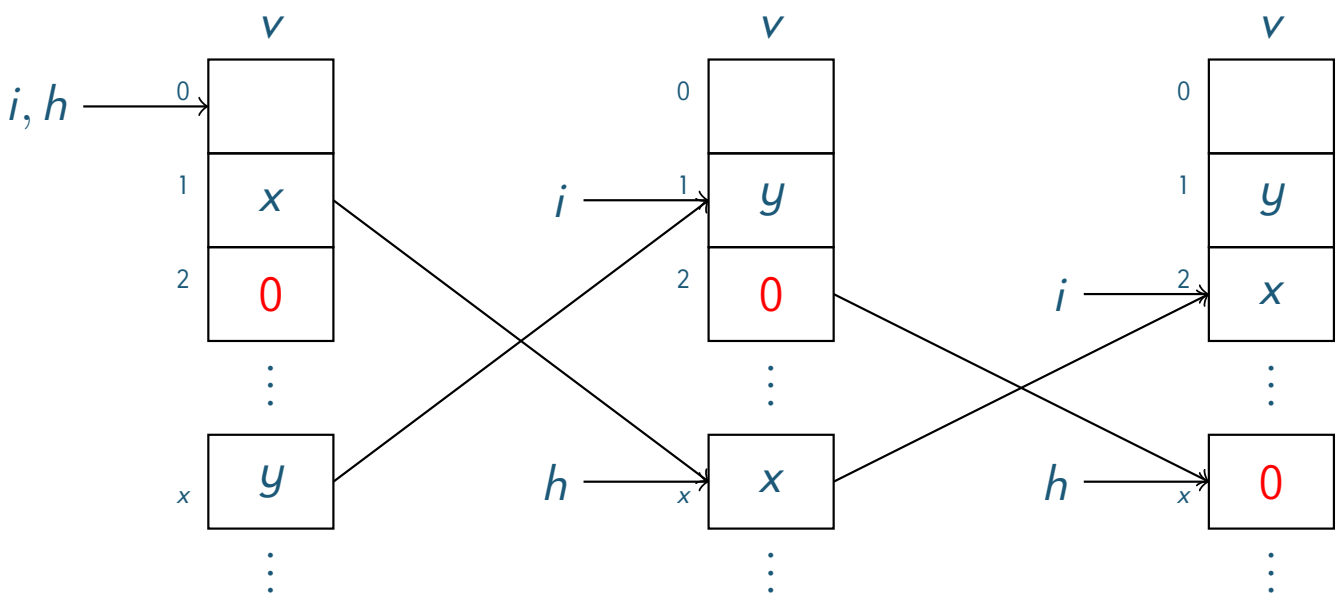


Proposals for Pseudorandom Generators — RC4

Bias [Mantin, Shamir 2002]

- Suppose $v[2] = 0$ and $v[1] = x \neq 2$
- Assuming initial vector v is uniformly distributed this occurs with probability $\frac{1}{256} \cdot \frac{254}{255}$
- Second output byte guaranteed to be 0
- In remaining cases 0 output with basically uniform probability

$$p \approx \frac{254}{256 \cdot 255} + \frac{\cancel{256} \cdot 255 - \cancel{254}}{256 \cdot 255} \cdot \frac{1}{\cancel{256}} \approx \frac{\cancel{254} + \cancel{254}}{256 \cdot \cancel{255}} \approx \frac{1+1}{256} > \frac{1}{256}$$



Summary

- Operation modes of private key ciphers
- Practical pseudorandom generators