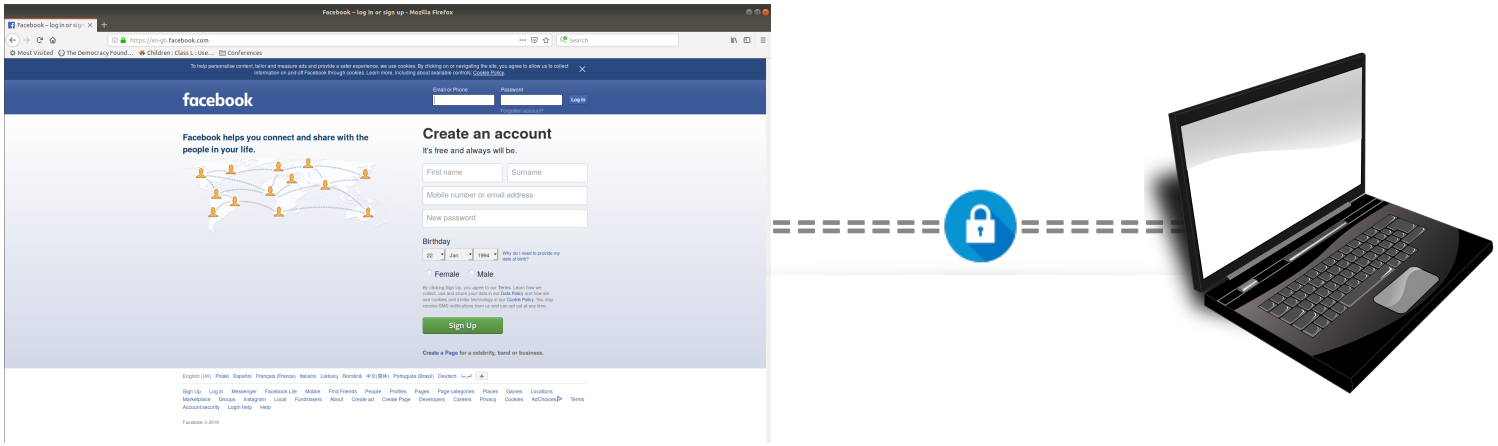
Cryptography: Symmetric encryption

Markulf Kohlweiss & Myrto Arapinis
School of Informatics
University of Edinburgh

January 13, 2021

Goal: confidentiality

- Secure communications



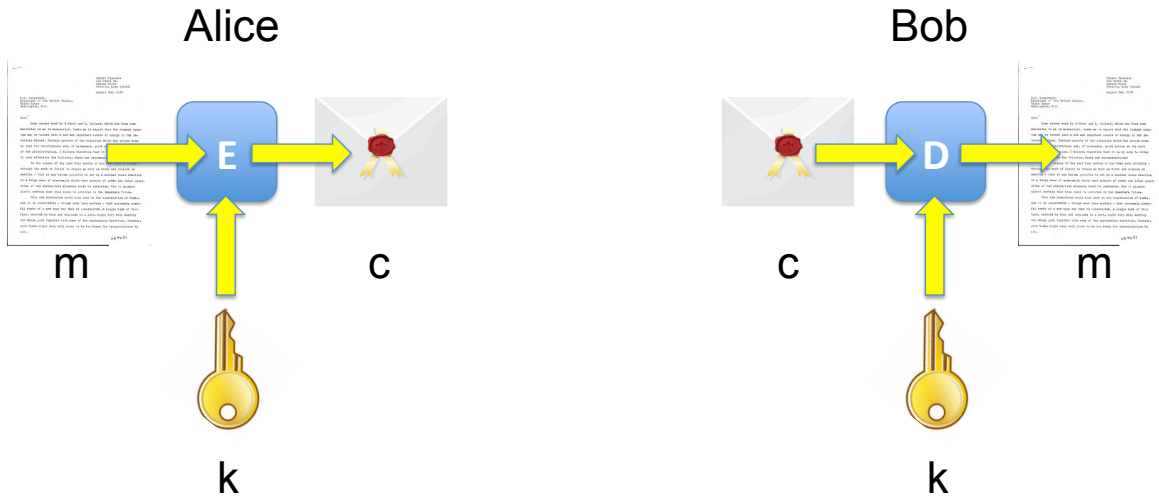
- File protection

Symmetric encryption schemes

A symmetric cipher consists of two algorithms

- encryption algorithm $E: \mathcal{K} \times \mathcal{M} \to \mathcal{C}$
- decryption algorithm $D: \mathcal{K} \times \mathcal{C} \to \mathcal{M}$

st. $\forall k \in \mathcal{K}$, and $\forall m \in \mathcal{M}$, $D(k, E(k, m)) = m$



- same key k to encrypt and decrypt
- the key k is secret: only known to Alice and Bob

What is a good encryption scheme?

An encryption scheme is secure against a given adversary, if this adversary cannot

- recover the secret key k
- recover the plaintext m underlying a ciphertext c
- recover any bits of the plaintext m underlying a ciphertext c
- ...

Kerckhoff's principle

The architecture and design of a security system/mechanism should be made public

No security through obscurity!

- The encryption (E) and decryption (D) algorithms are public
- The security relies entirely on the secrecy of the key

Open design allows for a system to be scrutinised by many users, white hat hackers, academics, *etc.*
$\rightsquigarrow$ early discovery and corrections of flaws/vulnerabilities

Adversary's capabilities

- A cryptographic scheme is secure under some assumptions, that is against a certain type of attacker
- A cryptographic scheme may be vulnerable to certain types of attacks but not others

Adversary's capabilities

- A cryptographic scheme is secure under some assumptions, that is against a certain type of attacker
- A cryptographic scheme may be vulnerable to certain types of attacks but not others

The attacker know the encryption/decryption algorithms but may have access to :

- Ciphertext only attack - some ciphertexts $c_1, \ldots, c_n$
- Known plaintext attack some plaintext/ciphertext pairs (m_1, c_1), $\ldots$, (m_n, c_n) st. $c_i = E(k, m_i)$)
- Chosen plaintext attack (CPA) - he has access to encryption oracle - can trick user to encrypt messages $m_1, \ldots, m_n$ of his choice
- Chosen ciphertext attack (CCA) - he has access to decryption oracle - can trick user to decrypt ciphertexts $c_1, \ldots, c_n$ of his choice
- unlimited, or polynomial, or realistic ($\leq 2^{80}$) computational power

Brute-force attack - attack on all schemes

- Try all possible keys $k \in \mathcal{K}$ - requires some knowledge about the structure of plaintext

- Making exhaustive search unfeasible:
  - $\mathcal{K}$ should be sufficiently large, *i.e.* keys should be sufficiently long
  - Keys should be sampled uniformly at random from $\mathcal{K}$

The One-Time Pad (OTP)

The One-Time Pad (OTP)

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

The One-Time Pad (OTP)

- $\mathcal{M} = \mathcal{C} = \mathcal{K} = \{0,1\}^n$
- Encryption: $\forall k \in \mathcal{K}.\ \forall m \in \mathcal{M}.\ E(k,m) = k \oplus m$

The One-Time Pad (OTP)

- $\mathcal{M} = \mathcal{C} = \mathcal{K} = \{0,1\}^n$
- Encryption: $\forall k \in \mathcal{K}.\ \forall m \in \mathcal{M}.\ E(k,m) = k \oplus m$

$$
\begin{array}{rcccccccc}
k & = & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\
m & = & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\
\hline
c & = & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0
\end{array}
$$

The One-Time Pad (OTP)

- $\mathcal{M} = \mathcal{C} = \mathcal{K} = \{0,1\}^n$
- Encryption: $\forall k \in \mathcal{K}.\ \forall m \in \mathcal{M}.\ E(k,m) = k \oplus m$

$$\begin{array}{rcccccccc} k & = & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ m & = & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ \hline c & = & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \end{array}$$

- Decryption: $\forall k \in \mathcal{K}.\ \forall c \in \mathcal{C}.\ D(k,c) = k \oplus c$

The One-Time Pad (OTP)

- $\mathcal{M} = \mathcal{C} = \mathcal{K} = \{0,1\}^n$
- Encryption: $\forall k \in \mathcal{K}.\ \forall m \in \mathcal{M}.\ E(k,m) = k \oplus m$

$$\begin{array}{rcccccccc} k & = & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ m & = & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ \hline \\ c & = & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \end{array}$$

- Decryption: $\forall k \in \mathcal{K}.\ \forall c \in \mathcal{C}.\ D(k,c) = k \oplus c$

$$\begin{array}{rcccccccc} k & = & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ c & = & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ \hline \\ m & = & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{array}$$

The One-Time Pad (OTP)

- $\mathcal{M} = \mathcal{C} = \mathcal{K} = \{0,1\}^n$
- Encryption: $\forall k \in \mathcal{K}.\ \forall m \in \mathcal{M}.\ E(k,m) = k \oplus m$

$$\begin{array}{rcccccccc} k & = & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ m & = & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ \hline c & = & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \end{array}$$

- Decryption: $\forall k \in \mathcal{K}.\ \forall c \in \mathcal{C}.\ D(k,c) = k \oplus c$

$$\begin{array}{rcccccccc} k & = & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ c & = & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ \hline m & = & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{array}$$

- Consistency: $D(k, E(k,m)) = k \oplus (k \oplus m) = m$

Perfect secrecy

Definition

A cipher (E, D) over $(\mathcal{M}, \mathcal{C}, \mathcal{K})$ satisfies perfect secrecy if for all messages $m_1, m_2 \in \mathcal{M}$ of same length ($|m_1| = |m_2|$), and for all ciphertexts $c \in \mathcal{C}$

$$|Pr(E(k, m_1) = c) - Pr(E(k, m_2) = c)| \leq \epsilon$$

where $k \xleftarrow{r} \mathcal{K}$ and ϵ is some "negligible quantity".

OTP satisfies perfect secrecy

Theorem (Shannon 1949)

The One-Time Pad satisfies perfect secrecy

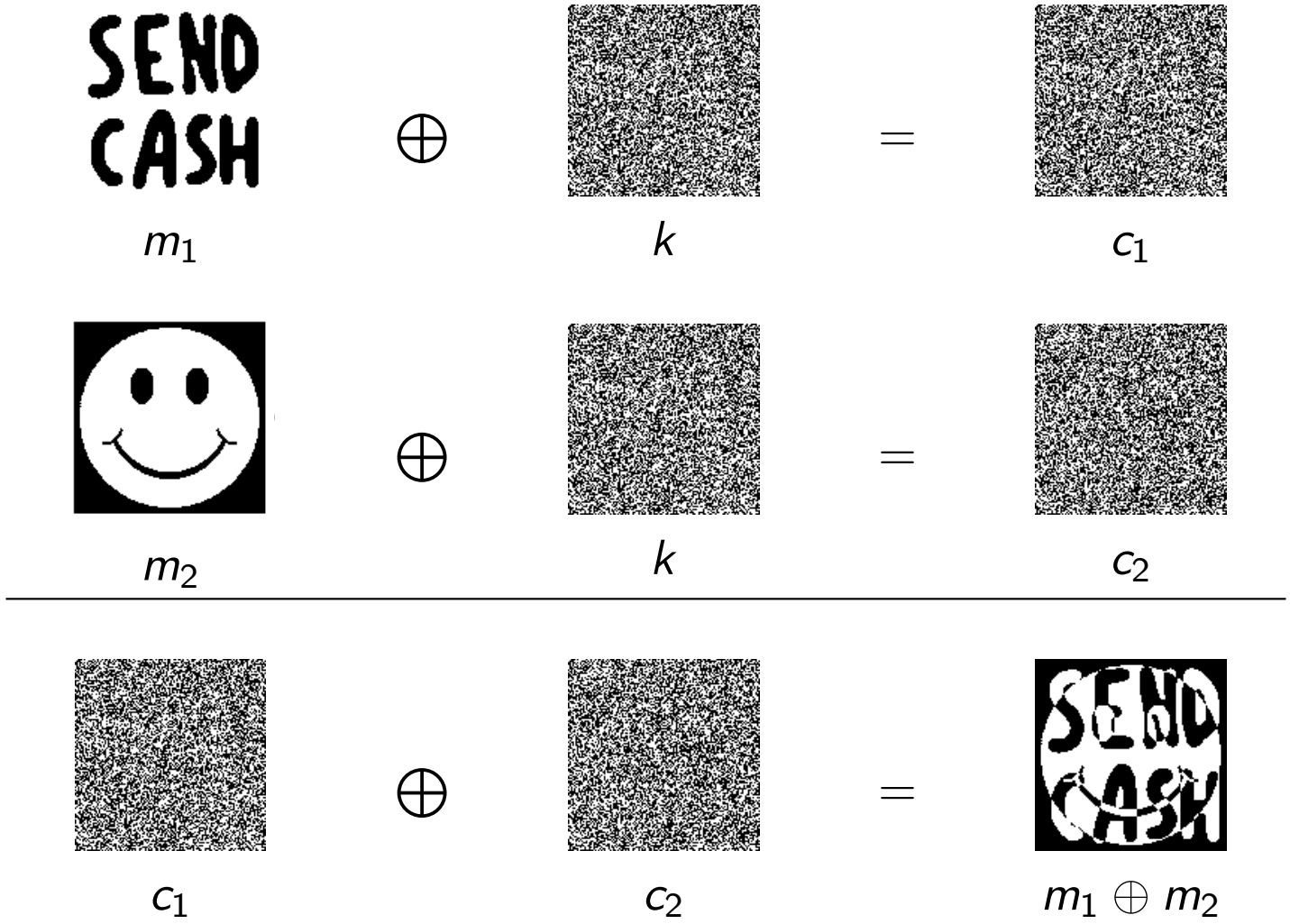
Proof: We first note that for all messages $m \in \mathcal{M}$ and all ciphertexts $c \in \mathcal{C}$

$$\begin{aligned} Pr(E(k,m) = c) &= \frac{\#\{k \in \mathcal{K}:\ k \oplus m = c\}}{\#\mathcal{K}} \\ &= \frac{\#\{k \in \mathcal{K}:\ k = m \oplus c\}}{\#\mathcal{K}} \\ &= \frac{1}{\#\mathcal{K}} \end{aligned}$$

where $k \xleftarrow{r} \mathcal{K}$.
Thus, for all messages $m_1, m_2 \in \mathcal{M}$, and for all ciphertexts $c \in \mathcal{C}$

$$|Pr(E(k,m_1) = c) - Pr(E(k,m_2) = c)| \leq \left| \frac{1}{\#\mathcal{K}} - \frac{1}{\#\mathcal{K}} \right| = 0$$

Two-time pad attacks



Source: Cryptosmith and David Lowry-Duda, `crypto.stackexchange.com/`

Limitations of OTP

Limitations of OTP

- Key-length!
  - The key should be as long as the plaintext

Limitations of OTP

- Key-length!
  - The key should be as long as the plaintext
- Getting true randomness!
  - The key should not be guessable from an attacker
  - If the key is not truly random, frequency analysis might again be possible

Limitations of OTP

- Key-length!
  - The key should be as long as the plaintext
- Getting true randomness!
  - The key should not be guessable from an attacker
  - If the key is not truly random, frequency analysis might again be possible
- Perfect secrecy does not capture all possible attacks

Limitations of OTP

- Key-length!
  - The key should be as long as the plaintext
- Getting true randomness!
  - The key should not be guessable from an attacker
  - If the key is not truly random, frequency analysis might again be possible
- Perfect secrecy does not capture all possible attacks
  - OTP is subject to two-time pad attacks
    given $m_1 \oplus k$ and $m_2 \oplus k$, we can compute
    $m_1 \oplus m_2 = (m_1 \oplus k) \oplus (m_2 \oplus k)$
    English has enough redundancy s.t. $m_1 \oplus m_2 \rightarrow m_1, m_2$

Limitations of OTP

- Key-length!
  - The key should be as long as the plaintext
- Getting true randomness!
  - The key should not be guessable from an attacker
  - If the key is not truly random, frequency analysis might again be possible
- Perfect secrecy does not capture all possible attacks
  - OTP is subject to two-time pad attacks
    given $m_1 \oplus k$ and $m_2 \oplus k$, we can compute
    $m_1 \oplus m_2 = (m_1 \oplus k) \oplus (m_2 \oplus k)$
    English has enough redundancy s.t. $m_1 \oplus m_2 \rightarrow m_1, m_2$
  - OTP is malleable
    given the ciphertext $c = E(k, m)$ with $m = \text{"}to\ bob : secret\ text\text{"}$, it is possible to compute the ciphertext $c' = E(k, m')$ with
    $m' = \text{"}to\ eve : secret\ text\text{"}$
    $c' := c \oplus \text{"}to\ bob : 00 \ldots 00\text{"} \oplus \text{"}to\ eve : 00 \ldots 00\text{"}$

Stream ciphers

Stream ciphers

- Goal: make the OTP practical

Stream ciphers

- Goal: make the OTP practical
- Idea: use a pseudorandom key rather than a really random key

Stream ciphers

- Goal: make the OTP practical
- Idea: use a pseudorandom key rather than a really random key
  - The key will not really be random, but will look random

Stream ciphers

- Goal: make the OTP practical
- Idea: use a pseudorandom key rather than a really random key
  - The key will not really be random, but will look random
  - The key will be generated from a key seed using a Pseudo-Random Generator (PRG)
    $G : \{0,1\}^s \rightarrow \{0,1\}^n$ with $s << n$

Stream ciphers

- Goal: make the OTP practical
- Idea: use a pseudorandom key rather than a really random key
  - The key will not really be random, but will look random
  - The key will be generated from a key seed using a Pseudo-Random Generator (PRG)
    $G: \{0,1\}^s \rightarrow \{0,1\}^n$ with $s << n$
- Encryption using a PRG G: $E(k,m) = G(k) \oplus m$

Stream ciphers

- Goal: make the OTP practical
- Idea: use a pseudorandom key rather than a really random key
  - The key will not really be random, but will look random
  - The key will be generated from a key seed using a Pseudo-Random Generator (PRG)
    $G: \{0,1\}^s \rightarrow \{0,1\}^n$ with $s << n$
- Encryption using a PRG G: $E(k,m) = G(k) \oplus m$
- Decryption using a PRG G: $D(k,c) = G(k) \oplus c$

Stream ciphers

- Goal: make the OTP practical
- Idea: use a pseudorandom key rather than a really random key
  - The key will not really be random, but will look random
  - The key will be generated from a key seed using a Pseudo-Random Generator (PRG)
    $G: \{0,1\}^s \rightarrow \{0,1\}^n$ with $s << n$
- Encryption using a PRG G: $E(k,m) = G(k) \oplus m$
- Decryption using a PRG G: $D(k,c) = G(k) \oplus c$
- Stream ciphers are subject to two-time pad attacks

Stream ciphers

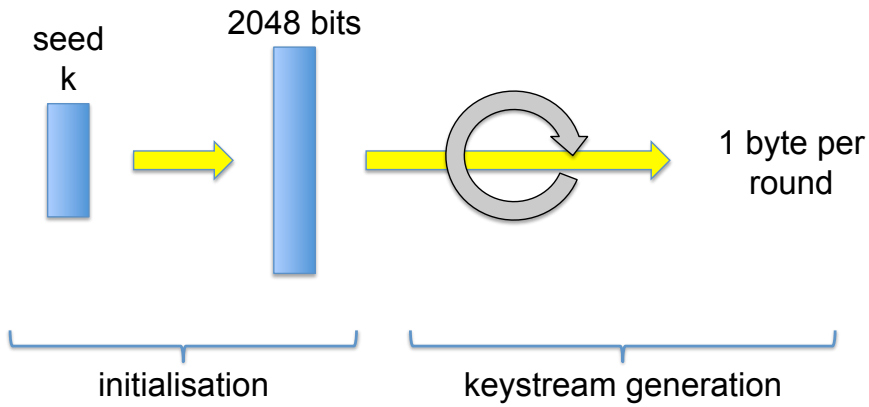
- Goal: make the OTP practical
- Idea: use a pseudorandom key rather than a really random key
  - The key will not really be random, but will look random
  - The key will be generated from a key seed using a Pseudo-Random Generator (PRG)
    $G: \{0,1\}^s \rightarrow \{0,1\}^n$ with $s << n$
- Encryption using a PRG G: $E(k,m) = G(k) \oplus m$
- Decryption using a PRG G: $D(k,c) = G(k) \oplus c$
- Stream ciphers are subject to two-time pad attacks
- Stream ciphers are malleable

RC4

- Stream cipher invented by Ron Rivest in 1987

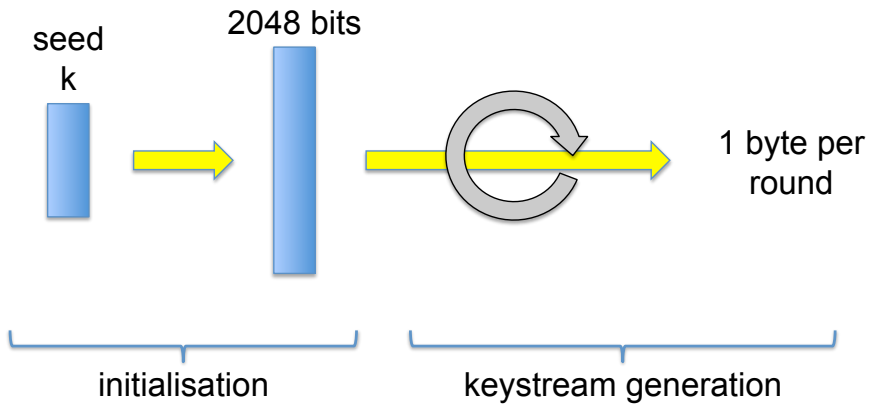
RC4

- Stream cipher invented by Ron Rivest in 1987
- Consists of 2 phases:



RC4

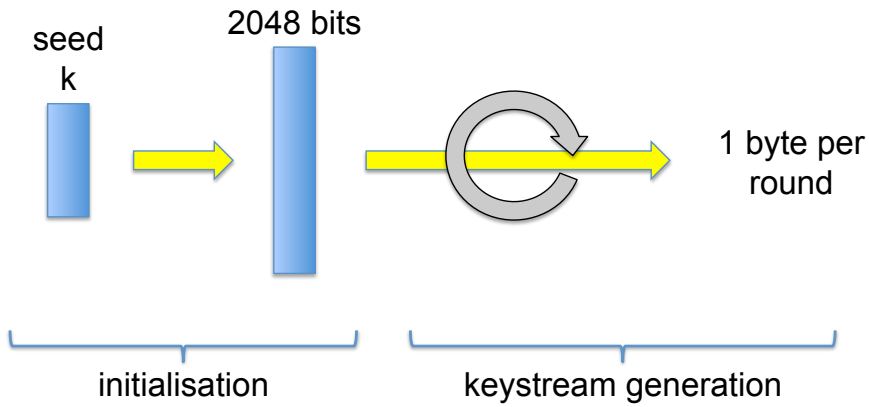
- Stream cipher invented by Ron Rivest in 1987
- Consists of 2 phases:



- Main data structure: array S of 256 bytes.

RC4

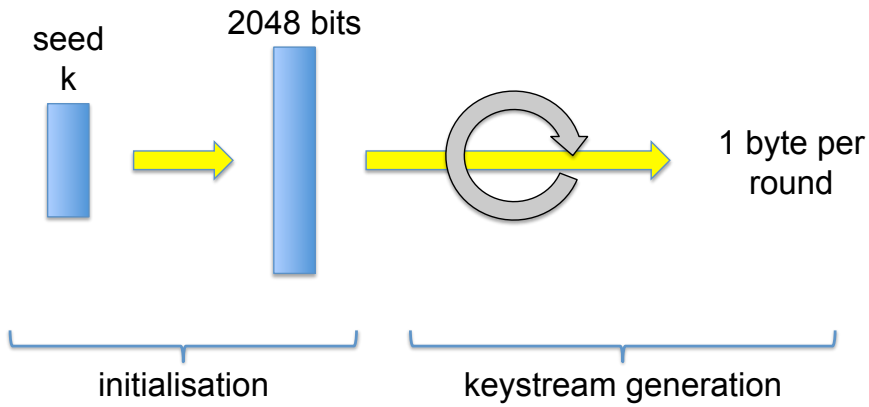
- Stream cipher invented by Ron Rivest in 1987
- Consists of 2 phases:



- Main data structure: array S of 256 bytes.
- Used in HTTPS and WEP

RC4

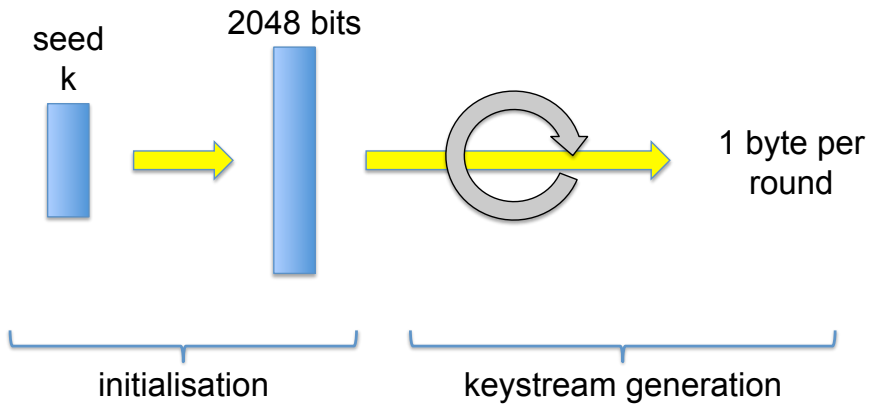
- Stream cipher invented by Ron Rivest in 1987
- Consists of 2 phases:



- Main data structure: array S of 256 bytes.
- Used in HTTPS and WEP
- Weaknesses of RC4:
  - first bytes are biased
    $\longrightarrow$ drop the first to 256 generated bytes

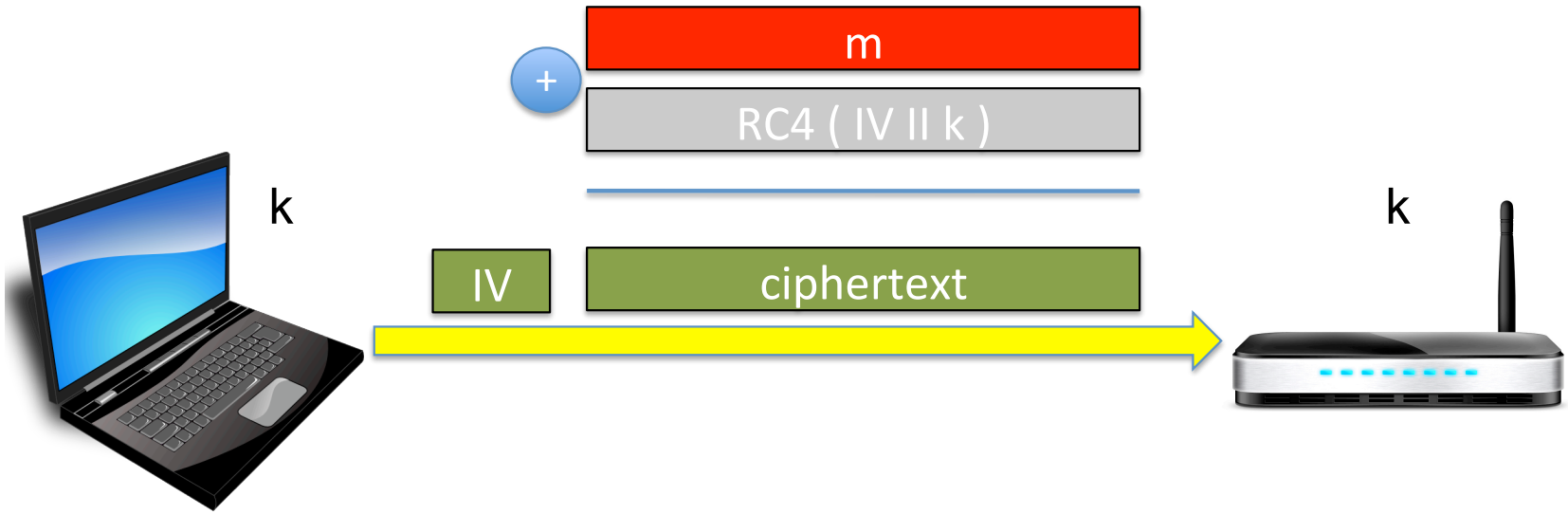
RC4

- Stream cipher invented by Ron Rivest in 1987
- Consists of 2 phases:



- Main data structure: array S of 256 bytes.
- Used in HTTPS and WEP
- Weaknesses of RC4:
  - first bytes are biased
    $\longrightarrow$ drop the first to 256 generated bytes
  - subject to related keys attacks
    $\longrightarrow$ choose randomly generated keys as seeds

WEP uses RC4



Initialisation Vector (IV): 24-bits long string

Weaknesses of WEP

Weaknesses of WEP

- two-time pad attack: IV is 24 bits long, so the key is reused after at most 2^{24} frames
  $\longrightarrow$ use longer IVs

Weaknesses of WEP

- two-time pad attack: IV is 24 bits long, so the key is reused after at most 2^{24} frames
  $\longrightarrow$ use longer IVs
- Fluhrer, Mantin and Shamir (FMS) attack (related keys attack):
  - the keys only differ in the 24 bits IV
  - first bytes of key stream known because standard headers are always sent
  - for certain IVs knowing m bytes of key and keystream means you can deduce byte $m + 1$ of key

  $\longrightarrow$ instead of using related IVs, generate IVs using a PRG

Weaknesses of TLS

MUST READ THESE TEN CITIES ARE HOME TO THE BIGGEST BOTNETS

RC4 NOMORE crypto exploit used to decrypt user cookies in mere hours

Websites using RC4 encryption need to change their protocols as exploits using design flaws are now far easier to perform.

By Charlie Osborne for Zero Day | July 20, 2015 -- 10:21 GMT (11:21 BST) | Topic: Security

RELATED STORIES

Security
Accenture acquires Defense Point Security to boost US federal defenses

Security
Facebook rolls out opt-in encryption for 'secret' Messenger chats

Security
Feds subpoena, gag encrypted chat firm Open Whisper Systems

Security
Insulin pump vulnerabilities could lead to overdose

Android BitCoin attack

ars TECHNICA BIZ & IT TECH SCIENCE POLICY CARS GAMING & CULTURE FORUMS SIGN IN

RANDOM THEFT —

All Android-created Bitcoin wallets vulnerable to theft

Android Java SecureRandom function flaw undermines security of Android wallets.

LEE HUTCHINSON - 8/12/2013, 3:15 PM

Xurxo Martínez

Bitcoin.org released a security advisory over the weekend warning the Bitcoin community that any Bitcoin wallet generated on any Android device is insecure and open to theft. The insecurity appears to stem from a flaw in the Android Java SecureRandom class, which under certain circumstances can

Modern stream ciphers

Project eStream: project to "identify new stream ciphers suitable for widespread adoption", organised by the EU ECRYPT network

$\longrightarrow$ HC-128, Rabbit, Salsa20/12, SOSEMANUK,
Grain v1, MICKEY 2.0, Trivium

Modern stream ciphers

Project eStream: project to "identify new stream ciphers suitable for widespread adoption", organised by the EU ECRYPT network

$\longrightarrow$ HC-128, Rabbit, Salsa20/12, SOSEMANUK,
Grain v1, MICKEY 2.0, Trivium

Conjecture

These eStream stream ciphers are "secure"

Concluding remarks on Stream Ciphers

Concluding remarks on Stream Ciphers

- Perfect secrecy does not capture all possible attacks.
  $\longrightarrow$ need for different security definition

Concluding remarks on Stream Ciphers

- Perfect secrecy does not capture all possible attacks.
  $\longrightarrow$ need for different security definition
- Theorem (Shannon 1949) Let (E, D) be a cipher over $(\mathcal{M}, \mathcal{C}, \mathcal{K})$. If (E, D) satisfies perfect secrecy, then the keys should be at least as long as the plaintexts ($|\mathcal{M}| \leq |\mathcal{K}|$).
  $\Rightarrow$ Stream ciphers do not satisfy perfect secrecy because the keys in $\mathcal{K}$ are smaller than the messages in $\mathcal{M}$
  $\longrightarrow$ need for different security definition

Concluding remarks on Stream Ciphers

- Perfect secrecy does not capture all possible attacks.
  $\longrightarrow$ need for different security definition
- Theorem (Shannon 1949) Let (E, D) be a cipher over $(\mathcal{M}, \mathcal{C}, \mathcal{K})$. If (E, D) satisfies perfect secrecy, then the keys should be at least as long as the plaintexts ($|\mathcal{M}| \leq |\mathcal{K}|$).
  $\Rightarrow$ Stream ciphers do not satisfy perfect secrecy because the keys in $\mathcal{K}$ are smaller than the messages in $\mathcal{M}$
  $\longrightarrow$ need for different security definition
- The design of crypto primitives is subtle and error prone.
  $\longrightarrow$ use standardised publicly know primitives

Concluding remarks on Stream Ciphers

- Perfect secrecy does not capture all possible attacks.
  $\longrightarrow$ need for different security definition
- Theorem (Shannon 1949) Let (E, D) be a cipher over $(\mathcal{M}, \mathcal{C}, \mathcal{K})$. If (E, D) satisfies perfect secrecy, then the keys should be at least as long as the plaintexts ($|\mathcal{M}| \leq |\mathcal{K}|$).
  $\Rightarrow$ Stream ciphers do not satisfy perfect secrecy because the keys in $\mathcal{K}$ are smaller than the messages in $\mathcal{M}$
  $\longrightarrow$ need for different security definition
- The design of crypto primitives is subtle and error prone.
  $\longrightarrow$ use standardised publicly know primitives
- Crypto primitives are secure under a precisely defined threat model.
  $\longrightarrow$ respect the security assumptions of the crypto primitives

Concluding remarks on Stream Ciphers

- Perfect secrecy does not capture all possible attacks.
  $\longrightarrow$ need for different security definition
- Theorem (Shannon 1949) Let (E, D) be a cipher over $(\mathcal{M}, \mathcal{C}, \mathcal{K})$. If (E, D) satisfies perfect secrecy, then the keys should be at least as long as the plaintexts ($|\mathcal{M}| \leq |\mathcal{K}|$).
  $\Rightarrow$ Stream ciphers do not satisfy perfect secrecy because the keys in $\mathcal{K}$ are smaller than the messages in $\mathcal{M}$
  $\longrightarrow$ need for different security definition
- The design of crypto primitives is subtle and error prone.
  $\longrightarrow$ use standardised publicly know primitives
- Crypto primitives are secure under a precisely defined threat model.
  $\longrightarrow$ respect the security assumptions of the crypto primitives
- Many attacks dues to poor implementations of cryptography

Cryptography: Block ciphers

Block ciphers

A block cipher with parameters k and ℓ is a pair of deterministic algorithms (E, D) such that

- Encryption $E : \{0,1\}^k \times \{0,1\}^\ell \to \{0,1\}^\ell$
- Decryption $D : \{0,1\}^k \times \{0,1\}^\ell \to \{0,1\}^\ell$

Examples:

3DES: $\ell = 64,\ k = 168$

AES: $\ell = 128,\ k = 128, 192, 256$

Data Encryption Standard (DES)

- Early 1970s: Horst Feistel designs Lucifer at IBM
  $k = 128$ bits, $\ell = 128$ bits
- 1973: NBS calls for block cipher proposals.
  $\longrightarrow$ IBM submits a variant of Lucifer.
- 1976: NBS adopts DES as a federal standard
  $k = 56$ bits, $\ell = 64$ bits
- 1997: DES broken by exhaustive search
- 2001: NIST adopts AES to replace DES
  $k = 128, 192, 256$ bits, $\ell = 128$ bits

Widely deployed in banking (ATM machines) and commerce

Attacks on DES

Attacks on DES

- **Exhaustive search:** it takes 2^{56} to do an exhaustive search over the key space
  $\longrightarrow$ COBACOBANA (120 FPGAs, $\sim$ 10K\$): 7 days

Attacks on DES

- **Exhaustive search:** it takes 2^{56} to do an exhaustive search over the key space
  $\longrightarrow$ COBACOBANA (120 FPGAs, $\sim$ 10K\$): 7 days

- **Linear cryptanalysis:** found affine approximations to DES
  $\longrightarrow$ can find 14 key bits in time 2^{42}
  brute force the remaining 56-14=42 in time 2^{42}
  $\Rightarrow$ total attack time $\approx 2^{43}$

Attacks on DES

- **Exhaustive search:** it takes 2^{56} to do an exhaustive search over the key space
  $\longrightarrow$ COBACOBANA (120 FPGAs, $\sim$ 10K\$): 7 days

- **Linear cryptanalysis:** found affine approximations to DES
  $\longrightarrow$ can find 14 key bits in time 2^{42}
  brute force the remaining 56-14=42 in time 2^{42}
  $\Rightarrow$ total attack time $\approx 2^{43}$

$\Rightarrow$ DES is badly broken! Do not use it in new projects!!

Triple DES (3DES)

- Goal: build on top of DES a block cipher resistant against exhaustive search attacks

Triple DES (3DES)

- Goal: build on top of DES a block cipher resistant against exhaustive search attacks
- Used in bank cards and RFID chips

Triple DES (3DES)

- Goal: build on top of DES a block cipher resistant against exhaustive search attacks
- Used in bank cards and RFID chips
- Let $DES = (E_{DES}, D_{DES})$. We build $3DES = (E_{3DES}, D_{3DES})$ as follows

Triple DES (3DES)

- Goal: build on top of DES a block cipher resistant against exhaustive search attacks
- Used in bank cards and RFID chips
- Let $DES = (E_{DES}, D_{DES})$. We build $3DES = (E_{3DES}, D_{3DES})$ as follows
  - $E_{3DES} : (\{0,1\}^k)^3 \times \{0,1\}^\ell \to \{0,1\}^\ell$
    $E_{3DES}((K_1, K_2, K_3), M) = E_{DES}(K_1, D_{DES}(K_2, E_{DES}(K_3, M)))$
    $\longrightarrow \; K_1 = K_2 = K_3 \Rightarrow DES$

Triple DES (3DES)

- Goal: build on top of DES a block cipher resistant against exhaustive search attacks
- Used in bank cards and RFID chips
- Let $DES = (E_{DES}, D_{DES})$. We build $3DES = (E_{3DES}, D_{3DES})$ as follows
  - $E_{3DES} : (\{0,1\}^k)^3 \times \{0,1\}^\ell \to \{0,1\}^\ell$
    $E_{3DES}((K_1, K_2, K_3), M) = E_{DES}(K_1, D_{DES}(K_2, E_{DES}(K_3, M)))$
    $\longrightarrow \; K_1 = K_2 = K_3 \Rightarrow DES$
  - $D_{3DES} : (\{0,1\}^k)^3 \times \{0,1\}^\ell \to \{0,1\}^\ell$
    $D_{3DES}((K_1, K_2, K_3), C) = D_{DES}(K_3, E_{DES}(K_2, D_{DES}(K_1, C)))$

Triple DES (3DES)

- Goal: build on top of DES a block cipher resistant against exhaustive search attacks
- Used in bank cards and RFID chips
- Let $DES = (E_{DES}, D_{DES})$. We build $3DES = (E_{3DES}, D_{3DES})$ as follows
  - $E_{3DES} : (\{0,1\}^k)^3 \times \{0,1\}^\ell \to \{0,1\}^\ell$
    $E_{3DES}((K_1, K_2, K_3), M) = E_{DES}(K_1, D_{DES}(K_2, E_{DES}(K_3, M)))$
    $\longrightarrow \; K_1 = K_2 = K_3 \Rightarrow DES$
  - $D_{3DES} : (\{0,1\}^k)^3 \times \{0,1\}^\ell \to \{0,1\}^\ell$
    $D_{3DES}((K_1, K_2, K_3), C) = D_{DES}(K_3, E_{DES}(K_2, D_{DES}(K_1, C)))$

$\longrightarrow$ 3 times as slow as DES!!

Triple DES (3DES)

- Goal: build on top of DES a block cipher resistant against exhaustive search attacks
- Used in bank cards and RFID chips
- Let $DES = (E_{DES}, D_{DES})$. We build $3DES = (E_{3DES}, D_{3DES})$ as follows
  - $E_{3DES} : (\{0,1\}^k)^3 \times \{0,1\}^\ell \to \{0,1\}^\ell$
    $E_{3DES}((K_1, K_2, K_3), M) = E_{DES}(K_1, D_{DES}(K_2, E_{DES}(K_3, M)))$
    $\longrightarrow \ K_1 = K_2 = K_3 \Rightarrow DES$
  - $D_{3DES} : (\{0,1\}^k)^3 \times \{0,1\}^\ell \to \{0,1\}^\ell$
    $D_{3DES}((K_1, K_2, K_3), C) = D_{DES}(K_3, E_{DES}(K_2, D_{DES}(K_1, C)))$

  $\longrightarrow$ 3 times as slow as DES!!
- key-size $= 3 \times 56 = 168$ bits
  $\Rightarrow$ Exhaustive search attack in 2^{168}

Triple DES (3DES)

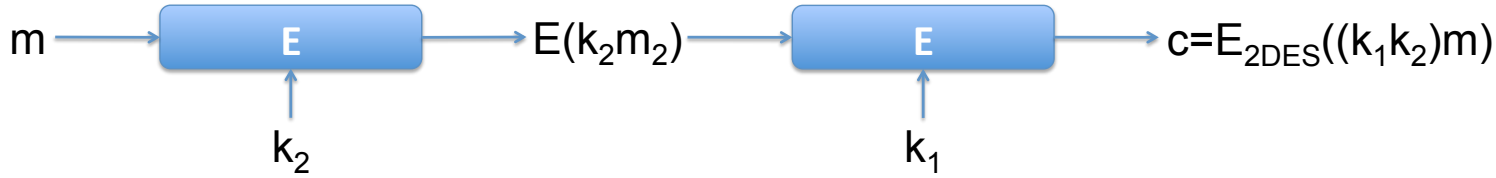
- Goal: build on top of DES a block cipher resistant against exhaustive search attacks
- Used in bank cards and RFID chips
- Let $DES = (E_{DES}, D_{DES})$. We build $3DES = (E_{3DES}, D_{3DES})$ as follows
  - $E_{3DES} : (\{0,1\}^k)^3 \times \{0,1\}^\ell \to \{0,1\}^\ell$
    $E_{3DES}((K_1, K_2, K_3), M) = E_{DES}(K_1, D_{DES}(K_2, E_{DES}(K_3, M)))$
    $\longrightarrow \ K_1 = K_2 = K_3 \Rightarrow DES$
  - $D_{3DES} : (\{0,1\}^k)^3 \times \{0,1\}^\ell \to \{0,1\}^\ell$
    $D_{3DES}((K_1, K_2, K_3), C) = D_{DES}(K_3, E_{DES}(K_2, D_{DES}(K_1, C)))$

  $\longrightarrow$ 3 times as slow as DES!!
- key-size $= 3 \times 56 = 168$ bits
  $\Rightarrow$ Exhaustive search attack in 2^{168}
- simple (meet-in-the-middle) attack in time 2^{118}

What about double DES (2DES)?

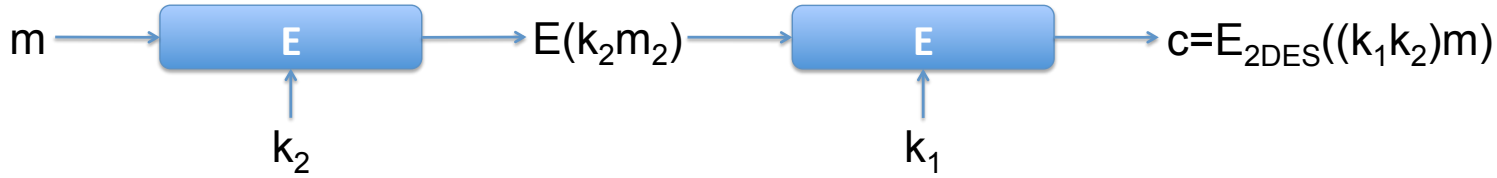
What about double DES (2DES)?

- $E_{2DES}((K_1, K_2), M) = E_{DES}(K_1, E_{DES}(K_2, M))$



What about double DES (2DES)?

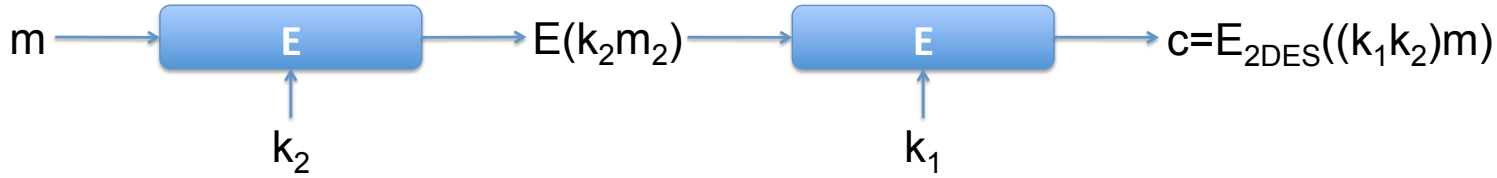
- $E_{2DES}((K_1, K_2), M) = E_{DES}(K_1, E_{DES}(K_2, M))$



$\Rightarrow$ For m and c such that $E_{2DES}((k_1, k_2), m) = c$ we have that $E_{DES}(k_2, m) = D_{DES}(k_1, c)$

What about double DES (2DES)?

- $E_{2DES}((K_1, K_2), M) = E_{DES}(K_1, E_{DES}(K_2, M))$

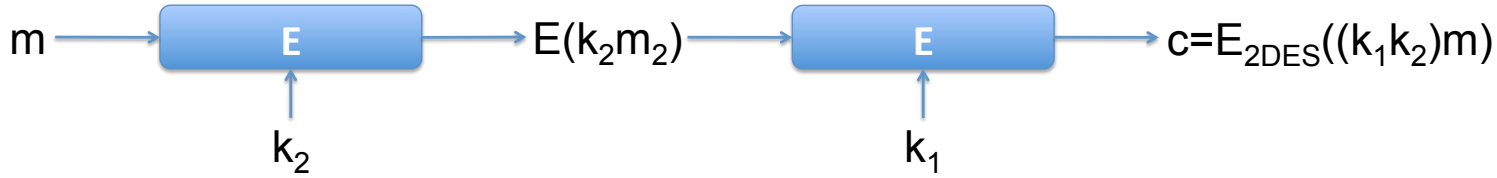


  $\Rightarrow$ For m and c such that $E_{2DES}((k_1, k_2), m) = c$ we have that $E_{DES}(k_2, m) = D_{DES}(k_1, c)$

- 2DES admits a meet-in-the-middle attack that reduces the time for key recovery from 2^{112} for an exhaustive search to 2^{56}. Given $M = (m_1, \ldots, m_{10})$ and $C = (E_{2DES}((k_1, k_2), m_1), \ldots, E_{2DES}((k_1, k_2), m_{10}))$

What about double DES (2DES)?

- $E_{2DES}((K_1, K_2), M) = E_{DES}(K_1, E_{DES}(K_2, M))$



$\Rightarrow$ For m and c such that $E_{2DES}((k_1, k_2), m) = c$ we have that $E_{DES}(k_2, m) = D_{DES}(k_1, c)$

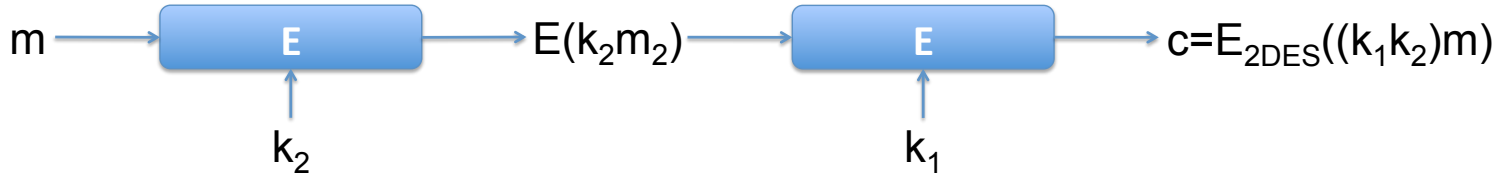
- 2DES admits a meet-in-the-middle attack that reduces the time for key recovery from 2^{112} for an exhaustive search to 2^{56}. Given $M = (m_1, \ldots, m_{10})$ and $C = (E_{2DES}((k_1, k_2), m_1), \ldots, E_{2DES}((k_1, k_2), m_{10}))$

  - For all possible k_2, $compute E_{DES}(k_2, M)$
  - Sort table according to the resulting $E_{DES}(k_2, M)$

  $\Big\}\ 2^{56}\log(2^{56})$

What about double DES (2DES)?

- $E_{2DES}((K_1, K_2), M) = E_{DES}(K_1, E_{DES}(K_2, M))$



$\Rightarrow$ For m and c such that $E_{2DES}((k_1, k_2), m) = c$ we have that $E_{DES}(k_2, m) = D_{DES}(k_1, c)$

- 2DES admits a meet-in-the-middle attack that reduces the time for key recovery from 2^{112} for an exhaustive search to 2^{56}. Given $M = (m_1, \ldots, m_{10})$ and $C = (E_{2DES}((k_1, k_2), m_1), \ldots, E_{2DES}((k_1, k_2), m_{10}))$

  - For all possible k_2, $compute E_{DES}(k_2, M)$
  - Sort table according to the resulting $E_{DES}(k_2, M)$

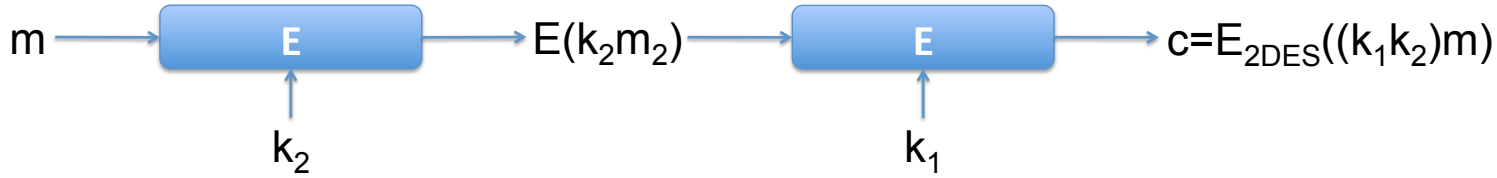
  $\Big\}\ 2^{56}\log(2^{56})$

  - For each possible k_1, $compute D_{DES}(k_1, C)$
  - Look up in the table if $D_{DES}(k_1, C) = E_{DES}(k_2, M)$

  $\Big\}\ 2^{56}\log(2^{56})$

What about double DES (2DES)?

- $E_{2DES}((K_1, K_2), M) = E_{DES}(K_1, E_{DES}(K_2, M))$



$\Rightarrow$ For m and c such that $E_{2DES}((k_1, k_2), m) = c$ we have that $E_{DES}(k_2, m) = D_{DES}(k_1, c)$

- 2DES admits a meet-in-the-middle attack that reduces the time for key recovery from 2^{112} for an exhaustive search to 2^{56}. Given $M = (m_1, \ldots, m_{10})$ and $C = (E_{2DES}((k_1, k_2), m_1), \ldots, E_{2DES}((k_1, k_2), m_{10}))$

  - For all possible k_2, $compute E_{DES}(k_2, M)$
  - Sort table according to the resulting $E_{DES}(k_2, M)$

  $\Big\}\ 2^{56}\log(2^{56})$

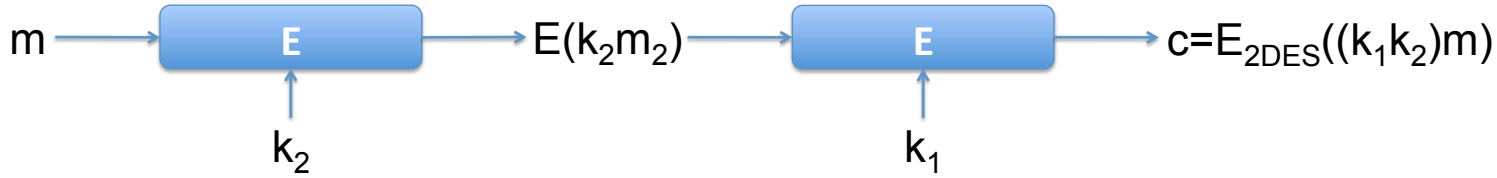
  - For each possible k_1, $compute D_{DES}(k_1, C)$
  - Look up in the table if $D_{DES}(k_1, C) = E_{DES}(k_2, M)$

  $\Big\}\ 2^{56}\log(2^{56})$

  $\Rightarrow$ time $< 2^{63}$

What about double DES (2DES)?

- $E_{2DES}((K_1, K_2), M) = E_{DES}(K_1, E_{DES}(K_2, M))$



$\Rightarrow$ For m and c such that $E_{2DES}((k_1, k_2), m) = c$ we have that $E_{DES}(k_2, m) = D_{DES}(k_1, c)$

- 2DES admits a meet-in-the-middle attack that reduces the time for key recovery from 2^{112} for an exhaustive search to 2^{56}. Given $M = (m_1, \ldots, m_{10})$ and $C = (E_{2DES}((k_1, k_2), m_1), \ldots, E_{2DES}((k_1, k_2), m_{10}))$

  - For all possible k_2, $compute E_{DES}(k_2, M)$
  - Sort table according to the resulting $E_{DES}(k_2, M)$

  $\Big\}\ 2^{56}\log(2^{56})$

  - For each possible k_1, $compute D_{DES}(k_1, C)$
  - Look up in the table if $D_{DES}(k_1, C) = E_{DES}(k_2, M)$

  $\Big\}\ 2^{56}\log(2^{56})$

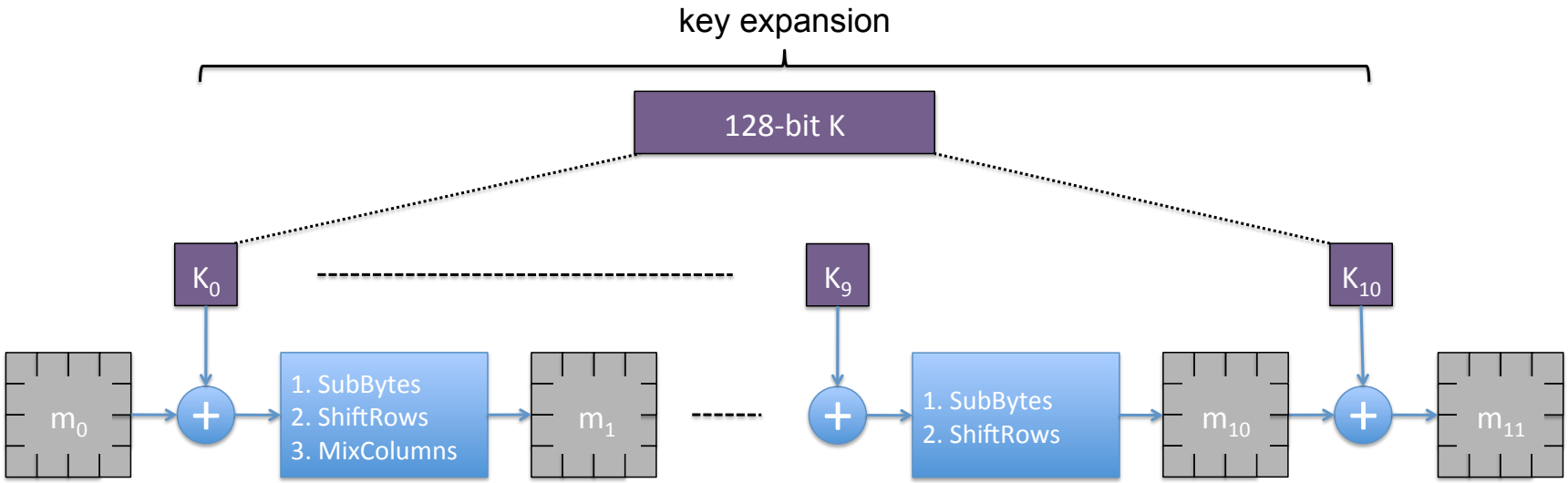
  $\Rightarrow$ time $< 2^{63}$

- Similar attack on 3DES in time 2^{118}

The Advanced Encryption Standard (AES)

- Goal: replace 3DES which is too slow (3DES is 3 times as slow as DES)
- 2001: NIST adopts Rijndael as AES
- Block size $\ell = 128$ bits, Key size $k = 128, 192, 256$ bits

AES: encryption circuit



- m_i : 4×4 byte matrix, K_i: 128-bit key
- m_0: plaintext, m_{11}: ciphertext
- at the last round MixColumns is not applied

Attacks on AES

- **Related-key attack** on the 192-bit and 256-bit versions of AES: exploits the AES key schedule [A. Biryukov, D. Khovratovich (2009)]
  $\longrightarrow$ key recovery in time $\sim 2^{99}$

- First **key-recovery attack** on full AES [A. Bogdanov, D. Khovratovich, C. Rechberger (2011)]
  $\longrightarrow$ 4 times faster than exhaustive search

Attacks on AES

- **Related-key attack** on the 192-bit and 256-bit versions of AES: exploits the AES key schedule [A. Biryukov, D. Khovratovich (2009)]
  $\longrightarrow$ key recovery in time $\sim 2^{99}$

- First **key-recovery attack** on full AES [A. Bogdanov, D. Khovratovich, C. Rechberger (2011)]
  $\longrightarrow$ 4 times faster than exhaustive search

$\Rightarrow$ Existing attacks on AES-128 are still not practical, but should use AES-192 and AES-256 in new projects!

Additional protection against quantum computers.

Using block ciphers

Goal

Encrypt M using a block cipher operating on blocks of length ℓ when $|M| \neq \ell$

Padding - $|M| \leq \ell$

- Bit padding - append a *set bit* ('1') at the end of message, and then append as many *reset bits* ('0') required
  Example: padding a 52-bits message for a 64-bits block:
  11010011 01010110 10010000 00111010 10110101 01011010 1111**1000 00000000**
  padding a 64-bits message M for 64-bits blocks requires adding a padding block:

  M|**10000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000**

Padding - $|M| \leq \ell$

- Bit padding - append a *set bit* ('1') at the end of message, and then append as many *reset bits* ('0') required
  Example: padding a 52-bits message for a 64-bits block:
  11010011 01010110 10010000 00111010 10110101 01011010 1111**1000 00000000**
  padding a 64-bits message M for 64-bits blocks requires adding a padding block:
  M|**10000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000**

- ANSI X.923 - byte padding - pad with zeros, the last byte defines the number of padded bytes
  Example: padding a 4-bytes message for 8-bytes blocks:
  DD DD DD DD **00 00 00 04**
  padding a $8k$-bytes messages for 8-bytes blocks requires adding a padding block:
  DD DD DD DD DD DD DD DD | **00 00 00 00 00 00 00 08**

Padding - $|M| \leq \ell$

- Bit padding - append a *set bit* ('1') at the end of message, and then append as many *reset bits* ('0') required
  Example: padding a 52-bits message for a 64-bits block:
  11010011 01010110 10010000 00111010 10110101 01011010 1111**1000 00000000**
  padding a 64-bits message M for 64-bits blocks requires adding a padding block:
  M|**10000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000**

- ANSI X.923 - byte padding - pad with zeros, the last byte defines the number of padded bytes
  Example: padding a 4-bytes message for 8-bytes blocks:
  DD DD DD DD **00 00 00 04**
  padding a $8k$-bytes messages for 8-bytes blocks requires adding a padding block:
  DD DD DD DD DD DD DD DD | **00 00 00 00 00 00 00 08**

- PKCS#7 - byte padding - the value of each added byte is the total number of padding bytes. The padding will be 01, or 02 02, or 03 03 03, or 04 04 04 04, *etc.*
  Example: padding a 4-bytes message for 8-bytes blocks:
  DD DD DD DD **04 04 04 04**
  padding a 8-bytes message for 8-bytes blocks requires adding a padding block:
  DD DD DD DD DD DD DD DD | **08 08 08 08 08 08 08 08**

Electronic Code Book (ECB) mode

(E, D) a block cipher.

Electronic Code Book (ECB) mode

(E, D) a block cipher.

To encrypt message M under key K using ECB mode:

Electronic Code Book (ECB) mode

(E, D) a block cipher.

To encrypt message M under key K using ECB mode:

- M is padded:

$$\Rightarrow \quad M' = M||P \text{ such that } |M'| = m \times \ell$$

Electronic Code Book (ECB) mode

(E, D) a block cipher.

To encrypt message M under key K using ECB mode:

- M is padded:
  $$\Rightarrow \; M' = M||P \text{ such that } |M'| = m \times \ell$$
- M' is broken into m blocks of length ℓ
  $$\Rightarrow \; M' = M_1 \,||\, M_2 \,||\, \ldots \,||\, M_m$$

Electronic Code Book (ECB) mode

(E, D) a block cipher.

To encrypt message M under key K using ECB mode:

- M is padded:
  $$\Rightarrow \quad M' = M||P \text{ such that } |M'| = m \times \ell$$
- M' is broken into m blocks of length ℓ
  $$\Rightarrow \quad M' = M_1 \,||\, M_2 \,||\, \ldots \,||\, M_m$$
- Each block M_i is encrypted under the key K using the block cipher
  $$\Rightarrow \quad C_i = E(K, M_i) \text{ for all } i \in \{1, \ldots, m\}$$

Electronic Code Book (ECB) mode

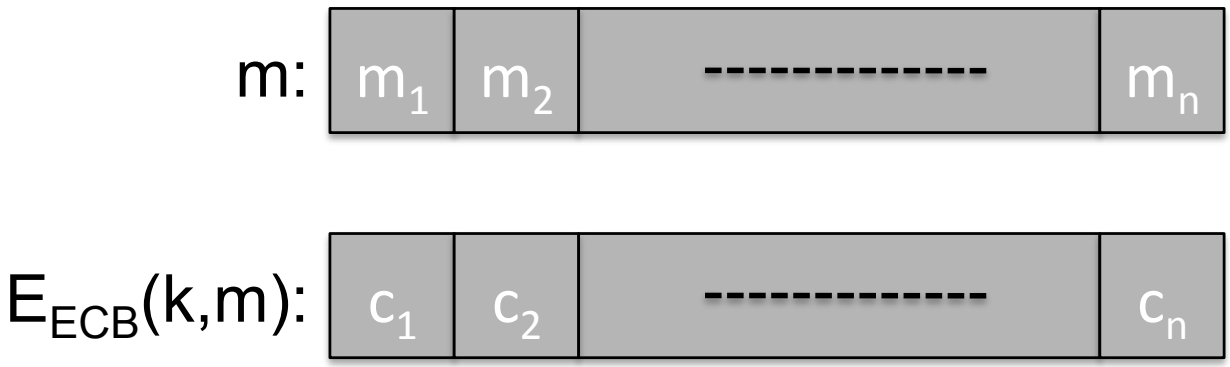
(E, D) a block cipher.

To encrypt message M under key K using ECB mode:

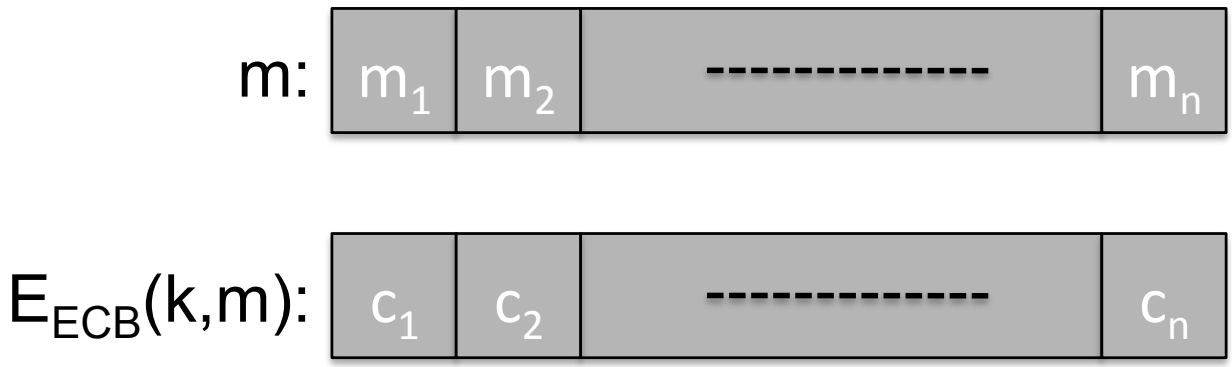
- M is padded:
  $$\Rightarrow \; M' = M||P \text{ such that } |M'| = m \times \ell$$
- M' is broken into m blocks of length ℓ
  $$\Rightarrow \; M' = M_1 \,||\, M_2 \,||\, \ldots \,||\, M_m$$
- Each block M_i is encrypted under the key K using the block cipher
  $$\Rightarrow \; C_i = E(K, M_i) \text{ for all } i \in \{1, \ldots, m\}$$
- The ciphertext corresponding to M is the concatenation of the C_is
  $$\Rightarrow \; C = C_1 \,||\, C_2 \,||\, \ldots \,||\, C_m$$

Weakness of ECB

Weakness of ECB



Weakness of ECB



Problem: $\forall i, j.\ m_i = m_j \ \Rightarrow\ c_i = E(k, m_i) = E(k, m_j) = c_j$

$\Rightarrow$ Malleable and weak to frequency analysis!

Weakness of ECB in pictures

Original image

Weakness of ECB in pictures

Original image

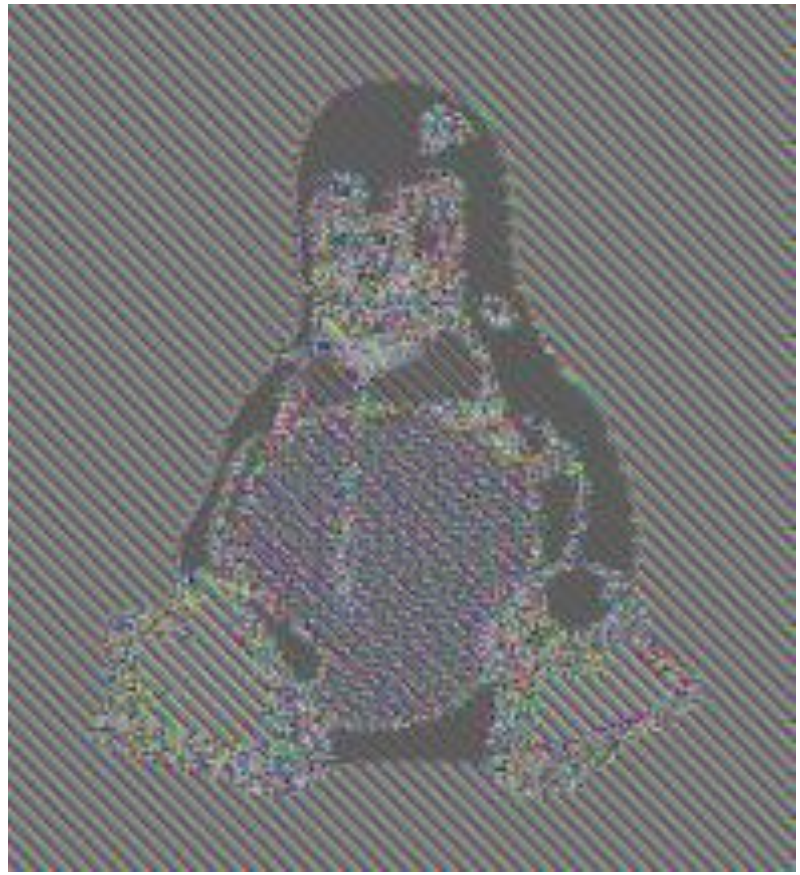
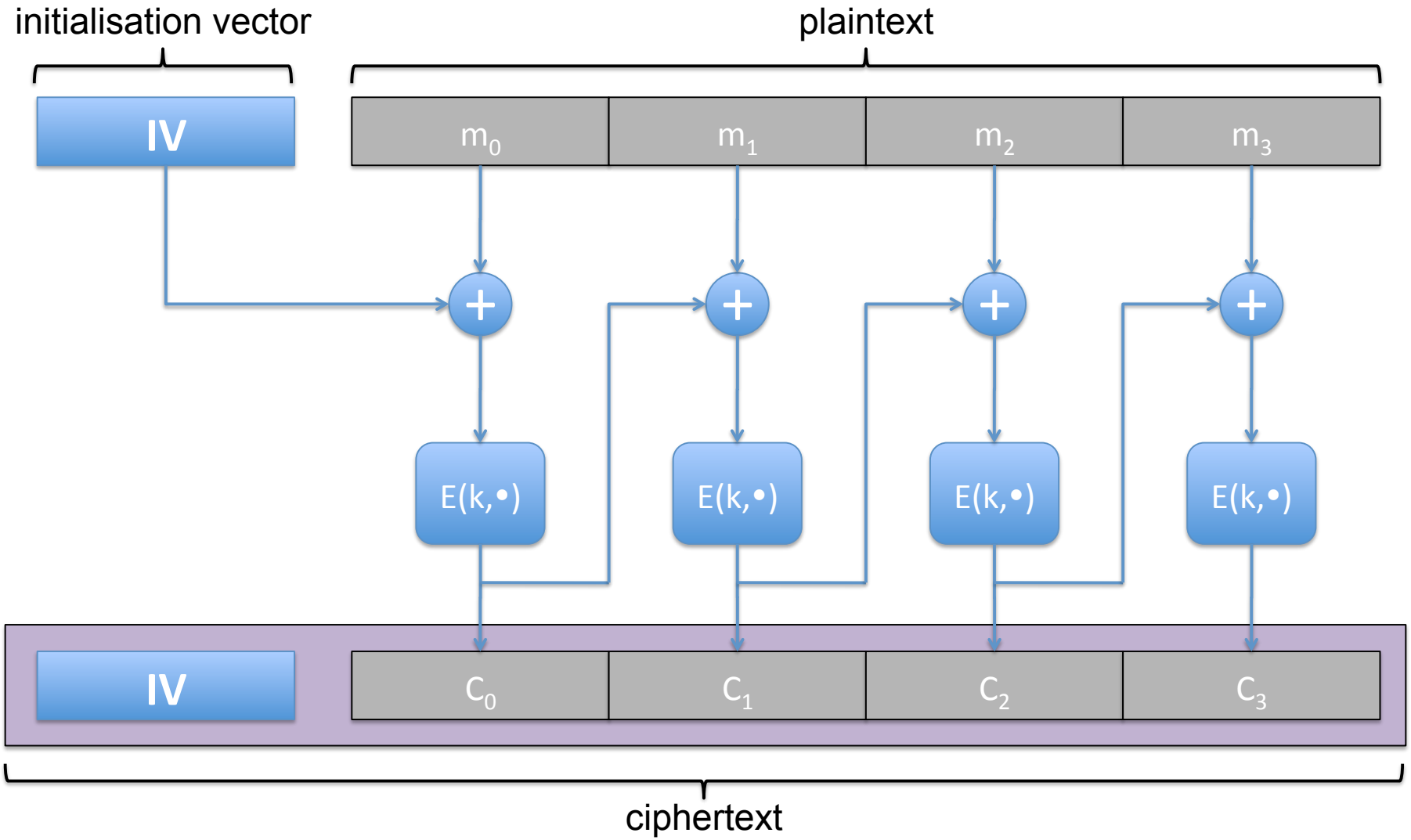


Image encrypted using ECB mode

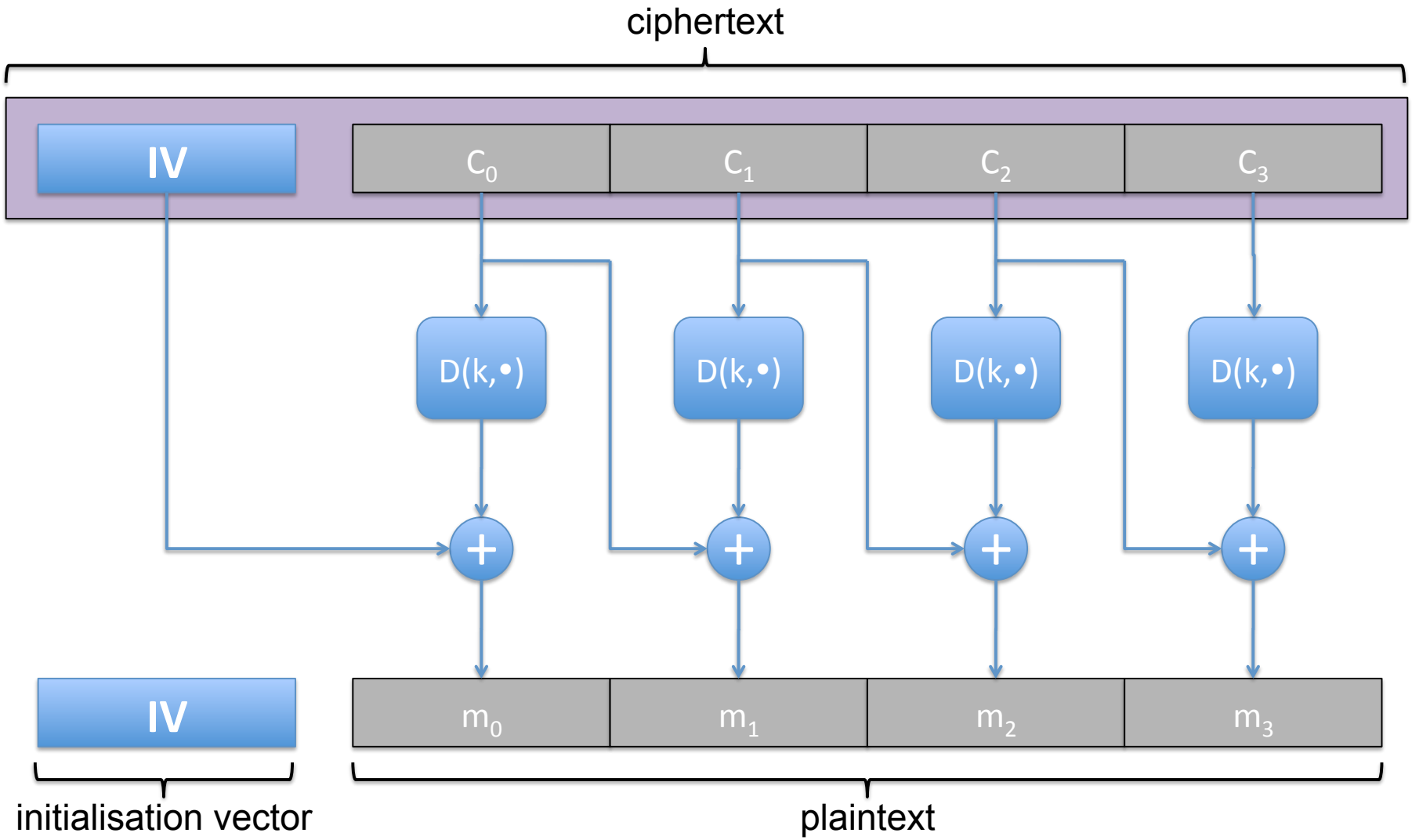
Cipher-block chaining (CBC) mode: encryption

(E, D) a block cipher that manipulates blocks of size ℓ.



IV chosen at random in $\{0,1\}^{\ell}$

Cipher-block chaining (CBC) mode: decryption



Sony PlayStation

Sony PlayStation

- Prevent games being copied

Sony PlayStation

- Prevent games being copied
- CD & full disk encryption

Sony PlayStation

- Prevent games being copied
- CD & full disk encryption
- Users can read and write on dedicated areas of disk

Sony PlayStation

- Prevent games being copied
- CD & full disk encryption
- Users can read and write on dedicated areas of disk
- Games loaded in read only area of disk

Sony PlayStation

- Prevent games being copied
- CD & full disk encryption
- Users can read and write on dedicated areas of disk
- Games loaded in read only area of disk
- With CBC encryption need to encrypt/decrypt whole disk to access a game

Sony PlayStation

- Prevent games being copied
- CD & full disk encryption
- Users can read and write on dedicated areas of disk
- Games loaded in read only area of disk
- With CBC encryption need to encrypt/decrypt whole disk to access a game
- Sony PS used ECB full-disk encryption

Sony PlayStation

- Prevent games being copied
- CD & full disk encryption
- Users can read and write on dedicated areas of disk
- Games loaded in read only area of disk
- With CBC encryption need to encrypt/decrypt whole disk to access a game
- Sony PS used ECB full-disk encryption
- Hardware controlled user access to data

Sony PlayStation disk encryption attack

- Remove disk and make copy

Sony PlayStation disk encryption attack

- Remove disk and make copy
- Put disk back in PlayStation

Sony PlayStation disk encryption attack

- Remove disk and make copy
- Put disk back in PlayStation
- Copy a file to the disk

Sony PlayStation disk encryption attack

- Remove disk and make copy
- Put disk back in PlayStation
- Copy a file to the disk
- Remove disk and find area of disk that changed (that is the user encrypted file)

Sony PlayStation disk encryption attack

- Remove disk and make copy
- Put disk back in PlayStation
- Copy a file to the disk
- Remove disk and find area of disk that changed (that is the user encrypted file)
- Copy target data to the user area

Sony PlayStation disk encryption attack

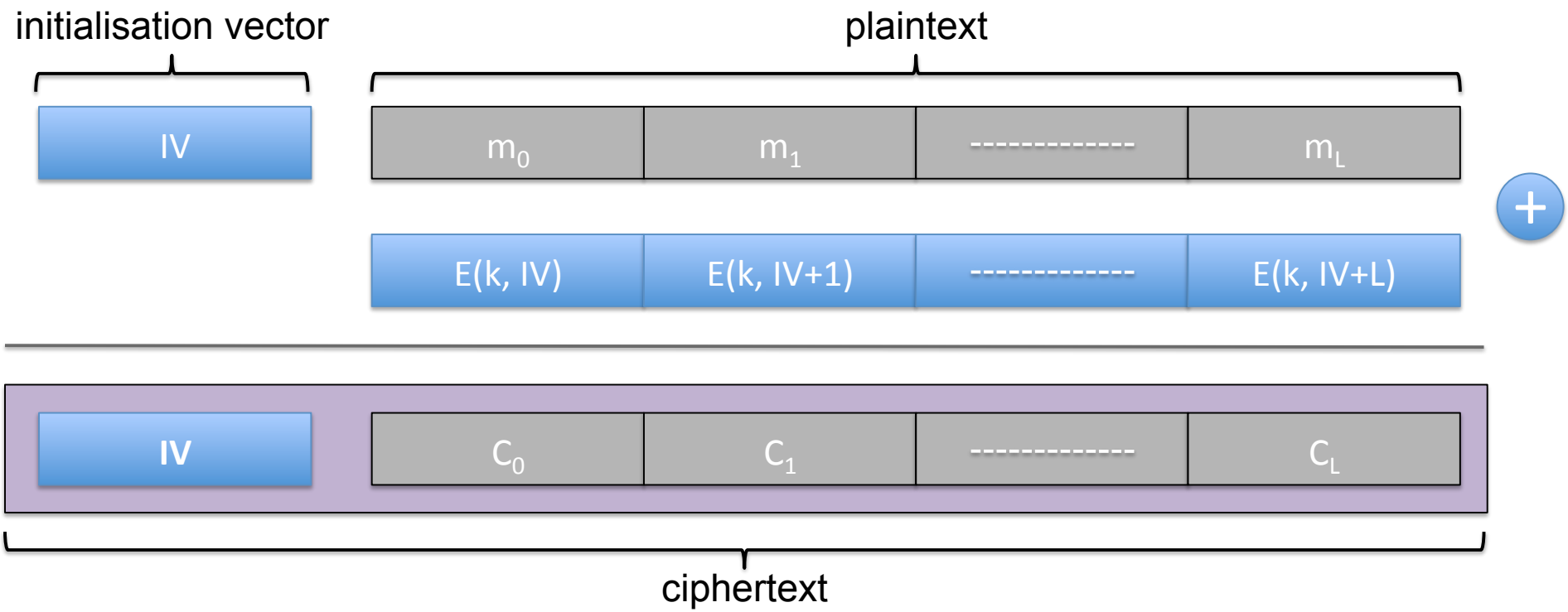
- Remove disk and make copy
- Put disk back in PlayStation
- Copy a file to the disk
- Remove disk and find area of disk that changed (that is the user encrypted file)
- Copy target data to the user area
- Put disk back in PlayStation and ask for user data

Sony PlayStation disk encryption attack

- Remove disk and make copy
- Put disk back in PlayStation
- Copy a file to the disk
- Remove disk and find area of disk that changed (that is the user encrypted file)
- Copy target data to the user area
- Put disk back in PlayStation and ask for user data
- PlayStation decrypts the file and gives it to user

Counter (CTR) mode

(E, D) a block cipher that manipulates blocks of size ℓ.



IV chosen at random in $\{0, 1\}^{\ell}$

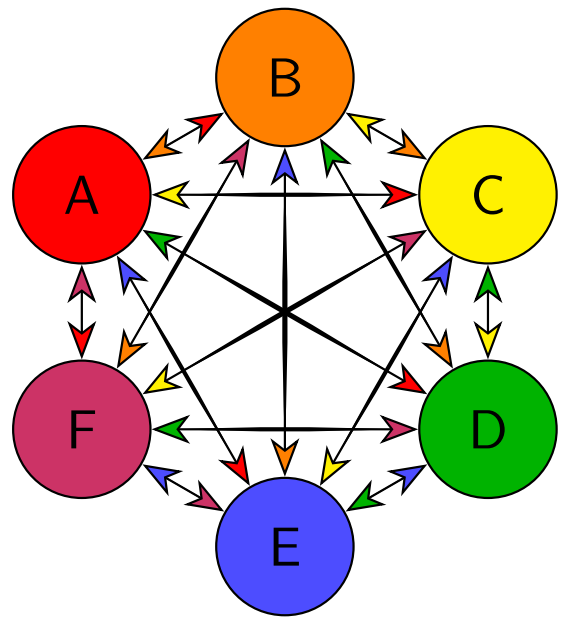
Block-size is also a problem!

- Sweet32 - birthday attacks on 64-bit block ciphers in TLS and openVPN
- Attack due to block-size being too small

The key management problem

The confidentiality problem is now reduced to a key management problem:

- Where are keys generated?
- How are keys generated?
- How are keys shared?
- Where are keys stored?
- Where are the keys actually used?
- How are key revoked and replaced?



One shared secret key per pair of users that want to communicate

What we have learned on Symmetric Encryption

- Frequency analysis as a cryptanalysis attack on classic encryption
- Importance of randomness in cryptography
- Stream ciphers
  - simple and efficient symmetric encryption schemes
  - use a random IV to thwart two-time pad attacks
  - good for random plaintexts otherwise subject to malleability attacks
- Block ciphers - use AES not DES
- CBC mode is more secure that ECB but less resilient to packets loss
- CTR mode more secure than ECB and parallelisable
- Keep up to date with cryptanalysis advances and standards
- Do not implement crypto lightly - use public reference implementations