Cryptography and Security

Lecture 1: a bit of history

About the course

Requirements

- Listen to the lectures
- Consultation the week after
- Exam in canvas (probably)
- Option to take a pre-exam
- Brief oral „defense” of the exam

Topics

- Mathematical foundations (algebra, number theory, probability)
- Cryptographic protocols (encryption, hash, signatures)
- Attacks (algorithmic, physical, . . .)

About the course

References

- Katz–Lindell: Introduction to Modern Cryptography
- Schneier: Applied cryptography

Already the ancient greeks . . .

Motivational questions

- Who sends what? (model of communication)
- What is an attack? (threat model)

Simple classical examples

- A message to be sent between two people
- Attack: the enemy intercepts the message

Already the ancient greeks . . .

Atbash

- alphabet: replace kth letter from the front with kth from the back
- first with hebrew alphabet
- Simple „substitution”: $\aleph \leftrightarrow tav, \beth \leftrightarrow shin, \ldots$
- works just as well with any alphabet, e.g. latin

```
Original:  abcdefghijklmnopqrstuvwxyz
Cipher:    zyxwvutsrqponmlkjihgfedcba
```

Properties

- Used in several places in the Bible
- Quite easy to „break”
- fixed permutation: considered weak today

Already the ancient greeks . . .

Caesar cipher

- Replace every letter with the letter that comes 3 positions later

Properties

- linked to Julius Ceasar (1st century B.C.E.)
- easy to break (Al-Kindi, 9th century)
- Still used as recently as 1915 by the Russian army
- A variant today: ROT-13 (e.g. for anti-spoiler forum posts)

Classical cryptography

19th century onwards

- military applications: protect communication
- ad-hoc constructions $\Rightarrow$ breaking is a matter of time only

Classical cryptography in modern terms

Symmetric key cryptography (informally)

- goal: protect communication between two parties
- setup: generate and share *common* secret key
- communication: on an open channel (except for key sharing)
- m : plain text („message”), c : encrypted text („ciphertext”), k : key
- Consists of 3 separate algorithms
  1. Gen – generation of key k. Choose randomly from a predefined set according to a predefined distribution
  2. Enc – encryption. Given the key k and message m, compute $c = Enc_k(m)$
  3. Dec – decryption. Given the key k and ciphertext c, compute $m = Dec_k(c)$

Kerchkoffs's principles

Design principles of encryption methods (Kerckhoffs, 1883)

1. A system must be practically, if not mathematically indecipherable.
2. It should not require secrecy. Should be no problem if enemy intercepts.
3. Key: easy to remember and change
4. Should be communicated via telegraph (low bandwidth:)
5. Should be portable and not require several people to handle, operate.
6. Should be easy to use.

„The” Kerchkoffs’s principle

Design principles of encryption methods (Kerckhoffs, 1883)

2. It should not require secrecy. Should be no problem if enemy intercepts.
3. (VS: security through obscurity)

Protect key vs. protect method

- secure distribution and storage
- replace compromised components
- communication with several peers

Conclusion

- goal: without the key, attacker cannot decrypt (or only with enormous effort)
- good encryption algorithms should be public
- public domain vs. security by obscurity

The attacker's toolkit

Traditional methods

- brute-force (try every possible key)
- frequency analysis (entropy of language)
- collision detection
- anything we (the good guys) have not even thought of

Other methods

- reverse-engineering (sec-by-obsc)
- software/hardware attacks (DOS/fault attacks)
- attacks using human weaknesses (social engineering, phishing)
- anything we have not even thought of

The attacker's toolkit

Passive attacks

- ciphertext-only attack: obtain m knowing one or more ciphertexts obtained by using the same k
- known plaintext attack: obtain further pairs (m, c) knowing one or more message-ciphertext pairs obtained by using the same k.

The attacker's toolkit

Active attacks

- chosen plaintext attack: the attacker has access to some message-ciphertext pairs where the messages are chosen by them (k fixed).
- chosen ciphertext attack: the attacker has access to some message-ciphertext pairs where the ciphertexts are chosen by them (k fixed).

Classical cryptographic protocols

Shift cipher

- principle: shift every letter by some (secret) number
- map alphabet to $0, 1, 2, \ldots, 25$, „wrap around": $25 + 1 = 0$.

Shift cipher

1. $Gen : k \in \{0, 1, \ldots, 25\}$ random.
2. Enc : for each character of message:
   $c_i = Enc_k(m_i) \equiv m_i + k \pmod{26}$
3. Dec : for each character of ciphertext:
   $m_i = Dec_k(c_i) \equiv c_i - k \pmod{26}$

Example. $m = \texttt{EXAMPLE} \to (4, 23, 0, 12, 15, 11, 4)$

1. $Gen : k = 11$
2. $Enc : c = (15, 8, 11, 23, 0, 22, 15) \to \texttt{PILXAWP}$
3. $Dec : m = (4, 23, 0, 12, 15, 11, 4) \to \texttt{EXAMPLE}$

Classical cryptographic protocols

Shift cipher

1. $Gen : k \in \{0, 1, \dots, 25\}$ random.
2. Enc : for each character of message:
   $c_i = Enc_k(m_i) \equiv m_i + k \pmod{26}$
3. Dec : for each character of ciphertext:
   $m_i = Dec_k(c_i) \equiv c_i - k \pmod{26}$

Properties

- trivial to break: try every possible key (brute-force)
- consequence: key space needs to be large
- a *necessary* condition for security (not sufficient though $\dots$)
- E.g. space of 128-bit long 0-1 sequences($= 2^{128}$ possibilities)

Classical cryptographic protocols

Mono-alphabetic replacement

- principle: replace each letter according to a randomly chosen permutation
- key space: all permutations (bijective mappings) on the alphabet

Mono-alphabetic replacement

1. $Gen : k$ a random permutation of the 26 elements
2. Enc, Dec : apply permutation/inverse caharecter by character

Example. $m = \texttt{PERMUTATION}$

1. $Gen : k = \begin{array}{l}\texttt{ABCDEFGHIJKLMNOPQRSTUVWXYZ} \\ \texttt{xeuadnbkvmrocqfsyhwglzijpt}\end{array}$
2. $Enc : c = \texttt{sdhclgxgvfq}, Dec : m = \texttt{PERMUTATION}$

Classical cryptographic protocols

Mono-alphabetic replacement

1. Gen : k a random permutation of the 26 elements
2. Enc, Dec : apply permutation/inverse caharecter by character

Properties

- key space: $26! \approx 2^{88}$ large enough, but ...
- Fixed equivalent of each letter $\Rightarrow$ statistics-based attack using frequencies
- Even short texts (10 words!) usually show statistics close to the global language stats
- Some letters easy to spot, brute force on the rest (hard to automize)
- Only on text that „makes sense”

Classical cryptographic protocols

Vigenère cipher

- principle: hide letter frequencies
- same letter shifted by variable amounts based on the position

Poly-alphabetic shift

1. $Gen: k = (k_1, \ldots, k_t)$ random (t not fixed)
2. $Enc: c_{i+jt} \equiv m_{i+jt} + k_i \pmod{26} : i = 1, \ldots, t, j = 0, \ldots$
3. $Dec: m_{i+jt} \equiv c_{i+jt} - k_i \pmod{26} : i = 1, \ldots, t, j = 0, \ldots$

Example $m = \texttt{THISISIMPOSSIBLE}$

1. $Gen: k = false$ (means a number sequence!)
2. $Enc:$
$$\begin{aligned} m &= \texttt{THISISIMPOSSIBLE} \\ k &= \texttt{falsefalsefalsef} \\ c &= \texttt{YHTIMXIXHSXSTTPJ} \end{aligned}$$

Classical cryptographic protocols

Poly-alphabetic shift

1. $Gen : k = (k_1, \ldots, k_t)$ random (t not fixed)
2. $Enc : c_{i+jt} \equiv m_{i+jt} + k_i \pmod{26} : i = 1, \ldots, t, j = 0, \ldots$
3. $Dec : m_{i+jt} \equiv c_{i+jt} - k_i \pmod{26} : i = 1, \ldots, t, j = 0, \ldots$

Properties

- If t is known $\Rightarrow$ broken (statistics on t-separated subsequences)
- Kasiski's method look for repetitions of 2,3: distances are multiple of $t \Rightarrow t =$ gcd
- count identicals: stats for $c_1, c_{1+k}, c_{2+k}, \ldots \Rightarrow$ ok for long text, short key
- Broken e.g. in American civil war ($k = \texttt{completevictory} \Rightarrow$ easy)

Wrap-up

Summary

- Kerchkhoffs's principle: make system public
- Make practical brute force attack infeasible with large key space
- It's hard to build a secure protocol
- Need science rather than ad-hoc ideas

Modern vs. traditional cryptography

Recap: traditional crypto

- An „art” rather than science
- Ad-hoc constructions
- Easy(ish) to break

Basic principles of modern crypto

1. formal and precise definitions
2. build on precisely stated assumptions
3. strictly proven security

Principle 1: precise definitions

Precise definitions

- design (What's the purpose / goal? Rather than „ex post facto")
- usage (appropriate for goal?)
- analysis (comparison)
- intuition not enough

„Define" secure encryption

An encryption method is secure if ???

Principle 1: precise definitions

Definition of secure encryption

An encryption method is considered secure if

1. no attacker can recover the key from the ciphertext itself.
2. no attacker can recover the plaintext message from the ciphertext itself (even if only a small part of the message is missing)
3. no attacker can recover a single character of the message from the ciphertext (probabilities / order of magnitude of computation needed)
4. no attacker can learn anything imprtant about the message knowing only the ciphertext (what counts as important?)
5. no attacker can recover any function (e.g. length, letter statistics, etc.) from the ciphertext

Principle 1: precise definitions

Definition of secure encryption

An encryption method is considered secure if

1. no attacker can recover the key from the ciphertext itself.
2. no attacker can recover the plaintext message from the ciphertext itself (even if only a small part of the message is missing)
3. no attacker can recover a single character of the message from the ciphertext (probabilities / order of magnitude of computation needed)
4. no attacker can learn anything imprtant about the message knowing only the ciphertext (what counts as important?)
5. no attacker can recover any function (e.g. length, letter statistics, etc.) from the ciphertext

Principle 1: precise definitions

Definition of secure encryption

An encryption method is considered secure if

1. no attacker can recover the key from the ciphertext itself.
2. no attacker can recover the plaintext message from the ciphertext itself (even if only a small part of the message is missing)
3. no attacker can recover a single character of the message from the ciphertext (probabilities / order of magnitude of computation needed)
4. no attacker can learn anything imprtant about the message knowing only the ciphertext (what counts as important?)
5. no attacker can recover any function (e.g. length, letter statistics, etc.) from the ciphertext

Principle 1: precise definitions

Definition of secure encryption

An encryption method is considered secure if

1. no attacker can recover the key from the ciphertext itself.
2. no attacker can recover the plaintext message from the ciphertext itself (even if only a small part of the message is missing)
3. no attacker can recover a single character of the message from the ciphertext (probabilities / order of magnitude of computation needed)
4. no attacker can learn anything imprtant about the message knowing only the ciphertext (what counts as important?)
5. no attacker can recover any function (e.g. length, letter statistics, etc.) from the ciphertext

Principle 1: precise definitions

Definition of secure encryption

An encryption method is considered secure if

1. no attacker can recover the key from the ciphertext itself.
2. no attacker can recover the plaintext message from the ciphertext itself (even if only a small part of the message is missing)
3. no attacker can recover a single character of the message from the ciphertext (probabilities / order of magnitude of computation needed)
4. no attacker can learn anything imprtant about the message knowing only the ciphertext (what counts as important?)
5. no attacker can recover any function (e.g. length, letter statistics, etc.) from the ciphertext

Principle 1: precise definitions

Definition of secure encryption

An encryption method is considered secure if

1. no attacker can recover the key from the ciphertext itself.
2. no attacker can recover the plaintext message from the ciphertext itself (even if only a small part of the message is missing)
3. no attacker can recover a single character of the message from the ciphertext (probabilities / order of magnitude of computation needed)
4. no attacker can learn anything imprtant about the message knowing only the ciphertext (what counts as important?)
5. no attacker can recover any function (e.g. length, letter statistics, etc.) from the ciphertext

Principle 1: precise definitions

How to make the definition formal?

- What does „break a cipher” / „recover the message” mean?
- What does „no attacker” mean (what powers do they posess)?

Example

A cryptographic protocol meant for a specific purpose is secure if no attacker with the specified (computational) power can perform a specified form of attack.

Principle 1: precise definitions

Math vs. practice

- hardware-based attacks
- human factors

Good definition of security/secrecy should

- support the intuitive view
- be supported by examples
- be backed up by ongoing analysis over time

Principle 2: precise assumptions

Two variants

- unconditional secrecy
- computational secrecy

Why?

- validation of assumptions
- comparison of methods
- facilitate security proofs

Principle 3: formal proofs of secrecy

Why?

- established difficulty vs. naive intuition
- works $\not\Rightarrow$ unbreakable
- risks of poor cryptosystem or poor software product

Proof of secrecy by reduction

Protocol X is considered secret (by a certain definition) if assumption Y is correct.

Perfect secrecy

Informálisan

What do we need to specify a scheme?

- three algorithms: Gen, Enc, Dec
- message space $\mathcal{M}$

Parameters

- *key space*: $\mathcal{K}$ set of possible keys ($k \in \mathcal{K}$)
- *message space*: $\mathcal{M}$ set of possible messages ($m \in \mathcal{M}$)
- *ciphertext space*: $\mathcal{C}$ set of possible ciphertexts
- Usually finite (esp. keyspace - „large" but finite)

Perfectly secret scheme

Parameters

- *Probability distributions* over $\mathcal{K}, \mathcal{M}, \mathcal{C}$
- $k \in \mathcal{K} : Pr(K = k)$ denotes the probability that key k is chosen.

e.g.: $\mathcal{K}$: bit sequences of length128,
$k \in_R \mathcal{K} \Rightarrow Pr(K = k) = 1/2^{128}$

- Similarly for $\mathcal{M}, \mathcal{C}$

e.g.: $|\mathcal{M}| = 2, Pr(\text{Attack tomorrow}) = 0.7, Pr(\text{No attack}) = 0.3$

- Distribution over $\mathcal{K}$ and $\mathcal{M}$ independent and arbitrary
- Distribution over $\mathcal{C}$ determined by the other two.
- *conditional probability*: $Pr(A \mid B)$: Probability of A, provided that we know B is true.

Perfectly secret scheme

Parameters

- *Probability distributions* over $\mathcal{K}, \mathcal{M}, \mathcal{C}$
- $k \in \mathcal{K} : Pr(K = k)$ denotes the probability that key k is chosen.

e.g.: $\mathcal{K}$: bit sequences of length128,
$k \in_R \mathcal{K} \Rightarrow Pr(K = k) = 1/2^{128}$

- Similarly for $\mathcal{M}, \mathcal{C}$

e.g.: $|\mathcal{M}| = 2, Pr(\text{Attack tomorrow}) = 0.7, Pr(\text{No attack}) = 0.3$

- Distribution over $\mathcal{K}$ and $\mathcal{M}$ independent and arbitrary
- Distribution over $\mathcal{C}$ determined by the other two.
- *conditional probability*: $Pr(A \mid B)$: Probability of A, provided that we know B is true.

Perfectly secret scheme

Definition

A scheme is a triple $\Pi = (Gen, Enc, Dec)$ *where :*

- Gen *is key generation, a probabilistic algorithm that returns a key* $k \in_R \mathcal{K}$ *(maybe using an input called the security parameter)*
- Enc *is encryption, a probabilistic algorithm that returns a ciphertext* $c \in \mathcal{C}$ *on inputs* $k \in \mathcal{K}$ *and* $m \in \mathcal{M}$*, i.e.* $c := Enc_k(m)$*.*
- Dec *is decryption a deterministic algorithm that returns a plaintext upon inputs* k *and* $c \in \mathcal{C}$*: the return value is* $Dec_k(c)\ in \mathcal{M}$*.*

Secrecy definitions

Intuiton

- we know the distribution of messages
- knowing the ciphertext, no information about the message should be learnt
- attacker's computational power: infinite

Definition

A scheme over $\mathcal{M}$ is perfectly secret if for any distribution over $\mathcal{M}$ and $\forall m \in \mathcal{M}, \forall c \in \mathcal{C}$:

$$Pr(M = m) = Pr(M = m | C = c).$$

Secrecy definitions

Equivalent formulation

- We cannot distinguish the ciphertexts corresponding to two different messages.

Lemma (Perfect indistinguishability)

A scheme provides perfect secrecy if for any distribution over $\mathcal{M}$, and $\forall m_1, m_2 \in \mathcal{M}, \forall c \in \mathcal{C}$:

$$Pr(C = c|M = m_1) = Pr(C = c|M = m_2).$$

Equivalent formulation

- indistinguishability game: two players: adversary and tester

Indistinguishability experiment with eavesdropper $PrivK^{eav}_{\mathcal{A},\Pi}$

1. Adversary $\mathcal{A}$ issues messages m_0, m_1 with $|m_0| = |m_1|$
2. Tester randomly chooses jey k and bit $b \in_R \{0,1\} : c = Enc_k(m_b)$. Send c to $\mathcal{A}$.
3. $\mathcal{A}$ answers by outputting $b' \in \{0,1\}$
4. $PrivK^{eav}_{\mathcal{A},\Pi} = 1$ if $b = b'$, otherwise 0. (Wins the game if guesses correctly.)

Secrecy definitions

Indistinguishability experiment with eavesdropper $PrivK^{eav}_{\mathcal{A},\Pi}$

1. Adversary $\mathcal{A}$ issues messages m_0, m_1 with $|m_0| = |m_1|$
2. Tester randomly chooses jey k and bit $b \in_R \{0,1\} : c = Enc_k(m_b)$. Send c to $\mathcal{A}$.
3. $\mathcal{A}$ answers by outputting $b' \in \{0,1\}$
4. $PrivK^{eav}_{\mathcal{A},\Pi} = 1$ if $b = b'$, otherwise 0. (Wins the game if guesses correctly.)

Definition

A scheme Π is perfectly secret over $\mathcal{M}$ if $\forall \mathcal{A}$:

$$Pr(PrivK^{eav}_{\mathcal{A},\Pi} = 1) = \frac{1}{2}.$$

Lemma

These definitions are equivalent.

One-time pad

One-time pad (OTP)

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

Gen let $k \in_R \{0,1\}^n$ uniformly random

Enc for $k \in \{0,1\}^n$ and $m \in \{0,1\}^n$, let $c = Enc_k(m) = k \oplus m$.

Dec for k and $c \in \{0,1\}^n$ let $Dec_k(c) = c \oplus k$.

Theorem

One-time pad is perfectly secret.

Drawbacks of perfect secrecy

One-time pad properties

- $|k| = |m| \Rightarrow$ too long keys / short messages only
- "one-time" (really, never reuse!)
- these are not unique to OTP, but inherent to perfect secrecy

Theorem

Let Π be a perfectly secret shceme over $\mathcal{M}$ and let $\mathcal{K}$ be the key space determined by Gen. Then $|\mathcal{K}| \geq |\mathcal{M}|$.

Recap: perfect secrecy

Informal description

An encryption method is perfectly secret if it is unbreakable

- no matter how much time you have
- no matter how powerful resources you have

One-time pad

- message and key: equal length
- throw away key after use
- the key is unknown to the attacker
- **key: true random bits**

Random sequences

What to expect from a random sequence?

- distribution of 0 and 1 (or other alphabet): uniform
- inability to predict future values based on the past/present
- no correlation between characters at different positions

Question

How do we define the randomness of a seqeunce?

Do random sequences actually exist?

- s_1=111111111111111111111111111111111111111111
- s_2=111111111111111111111000000000000000000000
- s_3=101010101010101010101010101010101010101010
- s_4=100011101011000011101010101110001110111110

Do random sequences actually exist?

- s_1=111111111111111111111111111111111111111111
- s_2=111111111111111111111000000000000000000000
- s_3=101010101010101010101010101010101010101010
- s_4=100011101011000011101010101110001110111110

Do random sequences actually exist?

- s_1=111111111111111111111111111111111111111111
- s_2=111111111111111111111000000000000000000000
- s_3=101010101010101010101010101010101010101010
- s_4=100011101011000011101010101110001110111110

Do random sequences actually exist?

- s_1=111111111111111111111111111111111111111111
- s_2=111111111111111111111000000000000000000000
- s_3=101010101010101010101010101010101010101010
- s_4=100011101011000011101010101110001110111110

Random sequence

Definition

Denote by $d(s)$ the minimal description of sequence s using some universal language $\mathcal{L}$. The length of this description is called the Kolmogorov complexity of s: $K(s) = |d(s)|$.

Definition

Sequence s is ***algorithmically random*** *if $|s| < K(s)$.*

Interpretation

Think of Kolmogorov complexity as the length of the shortest (python/C++/whatever) program that generates s as output. Randomness: no „easy" description/compression/program available.

Do random sequences actually exist?

- s_1=111111111111111111111111111111111111111111

  Short description of s_1: e.g. $[1] * 42$

- s_4=100011101011000011101010101110001110111110

  s_4 Looks random, *looks* uncompressible.

Kolmogorov complexity: the caveat

Theorem

No program exists that would compute $K(s)$ for any input s.

Proof by contradiction (sketch) Suppose a program KolmogorovComplexity(s) does the job. Suppose it has length 100000. Now, what's the shortest program that generates the output of the following algorithm?

---

for $i = 1 \to \infty$ **do**
  for each sequence s with length $\forall i$ **do**
    if KolmogorovComplexity(s)≥ 200000 **then**
      return s
    end if
  end for
end for

Generating random sequences

Quote

„Anyone who attempts to generate random numbers by deterministic means is, of course, living in a state of sin." – John von Neumann

Corollary of impossibility theorem

No mathematical algorithm to be expected that generates provably random sequences.

Properties of randomness needed

- physical source
- poor randomness implies software vulnerailities

Computational security

Kerckhoffs's principles

1. Make the cryptosystem secure in *practice* if not mathematically.
2. ...

Idea

Relax the conditions on perfect secrecy. No breaking of cipher

- in „reasonable” time
- with „reasonable” probability of success

Computational secrecy

Reasonable time = ???

- efficient adversary
- efficient algorithms

Definition

*An algorithm A has **polynomial running time** if $\exists\, p(.)$, a polynomial s.t. $\forall x \in \{0,1\}^*$ computation of $A(x)$ terminates in $\leq p(|x|)$ steps.*

Definition

*An algorithm A is **probabilistic** if it has access to a source generating uniformly random and independent bits.*

Efficient adversary

Using only probabilistic polynomial time (**PPT**) algorithms

Computational secrecy

Reasonable time = ???

- efficient adversary
- efficient algorithms

Definition

*An algorithm A has **polynomial running time** if $\exists\, p(.)$, a polynomial s.t. $\forall x \in \{0,1\}^*$ computation of $A(x)$ terminates in $\leq p(|x|)$ steps.*

Definition

*An algorithm A is **probabilistic** if it has access to a source generating uniformly random and independent bits.*

Efficient adversary

Using only probabilistic polynomial time (**PPT**) algorithms

Computational secrecy

Reasonable success

- very small probability
- negligible probability

Definition

*Function f is **negligible** if $\forall p(.)$, a polynomial $\exists N \in \mathbb{R}^+$ with* $\forall n \in \mathbb{N}, n > N : f(n) < \frac{1}{p(n)}$.

Reasonable probability of success

Negligible function as the probability of successful break.

Computational secrecy: two approaches

Concrete approach

A scheme is (t, ε)-secure if $\forall$ adversaries with at most t time, the prob. of success is at most ε.

Asymptotic approach

A scheme is secure if $\forall$ PPT adversaries have only negligible probability of successfully breaking the scheme.

Security proofs by reduction

- Unconditional security: has its limits
- we need some assumption for computational security
- a „basic” problem being hard
- based on that, we can argument for the difficulty of breaking the scheme

Reduction pattern

- If we suppose a PPT adversary: $\exists \mathcal{A}$ breaks the scheme with non-negl. prob.
- then there's an algorithm $\exists \mathcal{A}'$ solving the (by assumption) hard problem

(Computationally) secure encryption scheme

Definition

An encryption scheme is a triple $\Pi = (Gen, Enc, Dec)$ *where :*

- Gen *is key generation, a probabilistic algorithm that returns a key* $k \in_R \mathcal{K}$ *(maybe using an input called the security parameter)*
- Enc *is encryption, a probabilistic algorithm that returns a ciphertext* $c \in \mathcal{C}$ *on inputs* $k \in \mathcal{K}$ *and* $m \in \mathcal{M}$*, i.e.* $c := Enc_k(m)$*.*
- Dec *is decryption a deterministic algorithm that returns a plaintext upon inputs* k *and* $c \in \mathcal{C}$*: the return value is* $Dec_k(c)\ in \mathcal{M}$*.*

Threat model

Passive attacker: eavesdropping, has acces to a single encrypted text.

(Computationally) secure encryption scheme

Attack

- $\mathcal{A}$: eavesdropping
- a single instance of an encrypted message
- passive attack
- goal: $\mathcal{A}$ learns nothing about the plaintext m
  - *semantic security*
  - hard to handle
- instead: **indistinguishability**

(Computationally) secure encryption scheme

Definition (Indistinguishablity experiment with eavesdropper $PrivK^{eav}_{\mathcal{A},\Pi}(n)$)

1. *Adversary $\mathcal{A}$ returns two messages m_0, m_1 with $|m_0| = |m_1|$ upon input 1^n.*
2. $k = Gen(1^n), b \in_R \{0,1\} : c = Enc_k(m_b)$. *The ciphertext c is sent to $\mathcal{A}$*
3. *$\mathcal{A}$ outputs $b' \in \{0,1\}$*
4. $PrivK^{eav}_{\mathcal{A},\Pi}(n) = 1$, *if $b = b'$, otherwise 0.*

Definition

The scheme $\Pi = (Gen, Enc, Dec)$ has the indistinguishability property aginst one eavesdropping if all PPT adversaries $\mathcal{A}$, a negligible function $e(.)$ exists for which

$$P(PrivK^{eav}_{\mathcal{A},\Pi}(n) = 1) \leq \frac{1}{2} + e(n).$$

Pseudorandomness

Idea

- one-time pad idea
- replace perfect security by computational security
- replace random key by ???

Pseudorandom sequence

- PR for short
- relaxed, computational version of randomness
- *looks* random to a PPT observer
- generated from a short truly random sequence (seed)

Pseudorandomness

Idea

- one-time pad idea
- replace perfect security by computational security
- replace random key by ???

Pseudorandom sequence

- PR for short
- relaxed, computational version of randomness
- *looks* random to a PPT observer
- generated from a short truly random sequence (seed)

Pseudorandomness

Intuition

- has some physical randomness in it, but the sequence is much longer
- in reasonable time: indistinguishable from true randomness

Definition

Let $l(.)$ be a polynomial (called expansion factor) and $G : \{0,1\}^n \to \{0,1\}^{l(n)}$ a DPT (deterministic PT) algorithm. Then G is a pseudorandom generator if

1. $\forall n : l(n) > n$
2. $\forall D$ *PPT distinguisher,* $\exists e(.)$*, a negligible function with* $\forall s \in_R \{0,1\}^n, \forall r \in_R \{0,1\}^{l(n)}$*:*

$$|Pr(D(r) = 1) - Pr(D(G(s)) = 1)| \leq e(n)$$

Pseudorandomness

Intuition

- has some physical randomness in it, but the sequence is much longer
- in reasonable time: indistinguishable from true randomness

Definition

Let $l(.)$ be a polynomial (called expansion factor) and $G : \{0,1\}^n \rightarrow \{0,1\}^{l(n)}$ a DPT (deterministic PT) algorithm. Then G is a pseudorandom generator if

1. $\forall n : l(n) > n$
2. $\forall D$ *PPT distinguisher,* $\exists e(.)$*, a negligible function with* $\forall s \in_R \{0,1\}^n, \forall r \in_R \{0,1\}^{l(n)}$*:*

$$|Pr(D(r) = 1) - Pr(D(G(s)) = 1)| \leq e(n)$$

Pseudorandomness

Properties

- PRG no „real” randomness
- Brute force always works in principle
- Seed:
  - true randomness
  - secret
  - not TOO short
- PRG exists, assuming . . . is hard
- statistical tests

Example: next-bit test

Next-bit test

A sequnece passes the next-bit test if forall positions i, the attacker

- knowing the first i bits
- can NOT guess the bit at position $(i+1)$ with probability higher than $50\% + \epsilon$.

Statistical tests

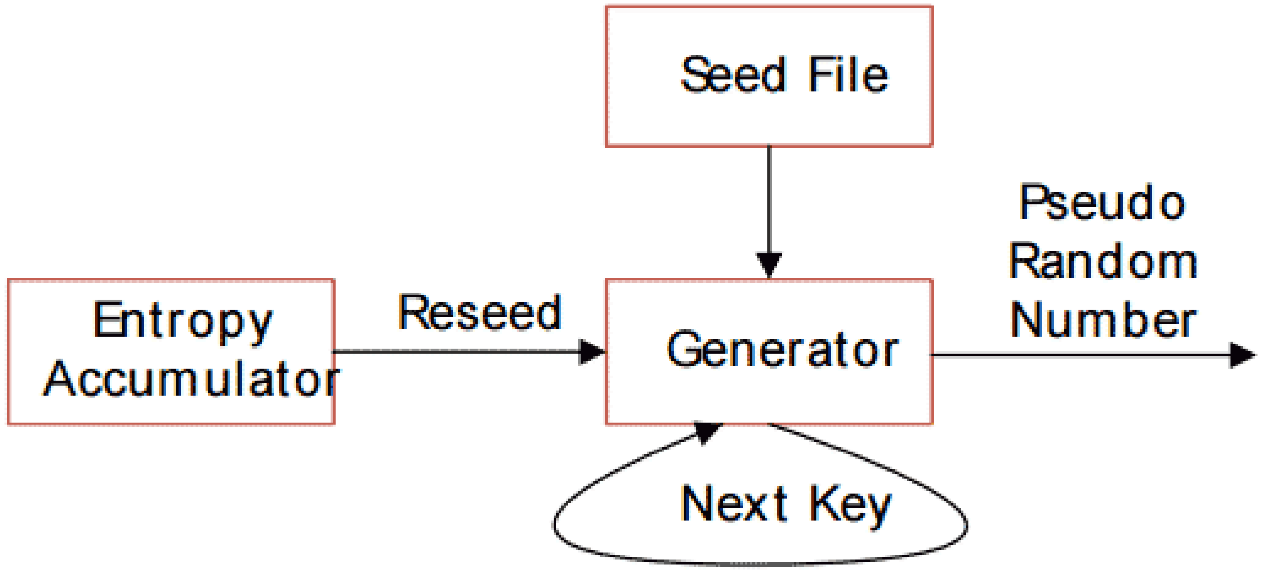
- NIST test (National standards intitute of US)
- DIEHARD test (academic origins – Marsaglia ’95)

PRG example: Fortuna

Fortuna

- Schneier, Fergusson 2003
- PRG family with 3 main components:

1. **Generator**: generate PR stream after seeding
2. **Entropy accumulator**: collect randomness
3. **Seeding**: ensure randomness in a bootstrapping phase



A secure scheme (against 1 eavsedropping)

Scheme using PRG

Let G be a PRG with expansion factor l, and

- **Gen** $k \in_R \{0,1\}^n$
- **Enc** For $k \in \{0,1\}^n$ and $m \in \{0,1\}^{l(n)}$, let $c = Enc_k(m) = G(k) \oplus m$.
- **Dec** For k, $c \in \{0,1\}^{l(n)}$, let $Dec_k(c) = c \oplus G(k)$.

Theorem

If G has the PRG properties, the $\Pi = (Gen, Enc, Dec)$ is secure in the presence of an eavesdropper (with one intercepted message).

A secure scheme (against multiple eavesdropping)

Definition (Indistinguishability experiment $PrivK^{meav}_{\mathcal{A},\Pi}(n)$)

Same as above, except:

1. *Adversary $\mathcal{A}$ issues a sequence* $M_0 = (m_{01}, \ldots, m_{0t}), M_1 = (m_{11}, \ldots, m_{1t})$ *with* $\forall i : |m_{0i}| = |m_{1i}|$
2. $k = Gen(1^n), b \in_R \{0, 1\} : C = (c_1, \ldots, c_t) : c_i = Enc_k(m_{bi})$ *received by* $\mathcal{A}$

- $\exists$ scheme secure for exactly 1 eavesdropping attempt
- deterministic algo never good $\Rightarrow$ randomization needed (IV)
- synchronization issues
- $Enc_k(m) = (IV, G(k, IV) \oplus m)$

A secure scheme (against multiple eavesdropping)

Definition (Indistinguishability experiment $PrivK^{meav}_{\mathcal{A},\Pi}(n)$)

Same as above, except:

1. *Adversary $\mathcal{A}$ issues a sequence* $M_0 = (m_{01}, \ldots, m_{0t}), M_1 = (m_{11}, \ldots, m_{1t})$ *with* $\forall i : |m_{0i}| = |m_{1i}|$
2. $k = Gen(1^n), b \in_R \{0, 1\} : C = (c_1, \ldots, c_t) : c_i = Enc_k(m_{bi})$ *received by* $\mathcal{A}$

- $\exists$ scheme secure for exactly 1 eavesdropping attempt
- deterministic algo never good $\Rightarrow$ randomization needed (IV)
- synchronization issues
- $Enc_k(m) = (IV, G(k, IV) \oplus m)$

Chosen plaintext attack

Attack

Active adversary: has access to arbitraty pairs (c, m)

Definition (CPA indistinguishability experiment $PrivK^{cpa}_{\mathcal{A},\Pi}(n)$)

1. $k = Gen(1^n)$
2. *Adversary $\mathcal{A}$ has oracle (black box) access to $Enc_k(.)$-hez, issues two plaintexts m_0, m_1 with $|m_0| = |m_1|$*
3. $b \in_R \{0, 1\} : c = Enc_k(m_b)$ *given to $\mathcal{A}$*
4. *$\mathcal{A}$ has further oracle access $Enc_k(.)$, outputs $b' \in \{0, 1\}$*
5. $PrivK^{cpa}_{\mathcal{A},\Pi}(n) = 1$, *if $b = b'$, 0 otherwise.*

Chosen plaintext attack

Definition

A scheme $\Pi = (Gen, Enc, Dec)$ *is CPA-secure if for any PPT adversary* $\mathcal{A}$ $\exists e(.)$, *a negligible function with*

$$P(PrivK^{cpa}_{\mathcal{A},\Pi}(n) = 1) \leq \frac{1}{2} + e(n).$$

- Cannot be deterministic
- Secure against one eavesdropping $\Rightarrow$ secure against multiple eavesdropping
- Can be fulfilled by PRG

Wrap-up

Summary

- Sequence is random if hard to describe/compress
- Relax perfect secrecy: computational security
- PPT adversary + negligible success probability
- Pseudorandom sequence: computational difficulty formulation

Stream cipher

- **Goal**: encrypt stream of arbitrary length
- **Idea**: One-time pad simulation with PRG + randomness

Stream cipher

Let G be a PRG and

Gen $k \in_R \{0,1\}^n$

Enc $k \in \{0,1\}^n$ key, IV random initialization vector, m message $c = Enc_k(m) = (IV, G(k, IV) \oplus m)$.

Dec If $c = (IV, s)$, then $Dec_k(c) = s \oplus G(k, IV)$.

- If G satisfies PRG properties, then this is (computationally) secure cipher against multiple eavesdropping

Reminder: chosen plaintext attack

Attack

Active adversary: arbitrary plaintext messages and corresponding ciphertexts avaiable

Definition (CPA indisitinguishability experiment $PrivK^{cpa}_{\mathcal{A},\Pi}(n)$)

1. $k = Gen(1^n)$
2. *Adversary $\mathcal{A}$ has oracle access to $Enc_k(.)$. Issues m_0, m_1, $|m_0| = |m_1|$*
3. $b \in_R \{0,1\} : c = Enc_k(m_b)$ *given to $\mathcal{A}$*
4. *$\mathcal{A}$ has oracle access to $Enc_k(.)$. Issues $b' \in \{0,1\}$*
5. $PrivK^{cpa}_{\mathcal{A},\Pi}(n) = 1$, *if $b = b'$, 0 otherwise.*

CPA

Definition

A cipher $\Pi = (Gen, Enc, Dec)$ *is CPA-secure if* $\forall \mathcal{A}$ *PPT adversary* $\exists e(.)$ *negligible with*

$$P(PrivK^{cpa}_{\mathcal{A},\Pi}(n) = 1) \leq \frac{1}{2} + e(n).$$

Tool: pseudorandom function family

- keeps input length (n-bit strings mapped to n-bit strings)
- for fixed $k \in \{0,1\}^n$ $F_k(.)$ is a member of the family
- a.k.a. keyed function
- a random element from the family is indistinguishable from a truly random function

CPA-secure cipher

CPA-security from PRF

Let F be a PRF, furthermore

Gen $k \in_R \{0,1\}^n$

Enc $c = Enc_k(m) = (r, F_k(r) \oplus m)$.

Dec For $c = (r, s)$, let $Dec_k(r, s) = F_k(r) \oplus s$.

Theorem

If F has the PRF property, then this $\Pi = (Gen, Enc, Dec)$ is CPA-secure.

Block ciphers

- **Goal**: encryption of fixed length message (block)
- **Idea**: strengthen PRF construction

Tool: strong pseudorandom permutation family (strong PRP)

- similar to PRF, but additionally:
- keeps length (n-bit strings mapped to n-bit strings), **BIJECTION**
- for fixed $k \in \{0,1\}^n$, $F_k(.)$ is a member of the family
- any random element of the family is indistinguishable from a truly random function, knowing the functions and **inverses**.

Block ciphers

Block cipher

Let F be a strong PRP and

- **Gen** $k \in_R \{0,1\}^n$
- **Enc** for key $k \in \{0,1\}^n$, message $m \in \{0,1\}^n$ and random value $r \in_R \{0,1\}^n$, let $c = Enc_k(m) = (r, F_k(r) \oplus m)$.
- **Dec** For $c = (r, s)$, let $Dec_k(r, s) = F_k(r) \oplus s$.

- Remaining challenges: encrypt long messages with block ciphers
- $m = (m_1, \ldots, m_l)$ where $\forall i : |m_i| = n$ (padding if needed)
- Let's encrypt block m_i according to some *mode of operation* one by one

Block ciphers

Block cipher

Let F be a strong PRP and

Gen $k \in_R \{0,1\}^n$

Enc for key $k \in \{0,1\}^n$, message $m \in \{0,1\}^n$ and random value $r \in_R \{0,1\}^n$, let $c = Enc_k(m) = (r, F_k(r) \oplus m)$.

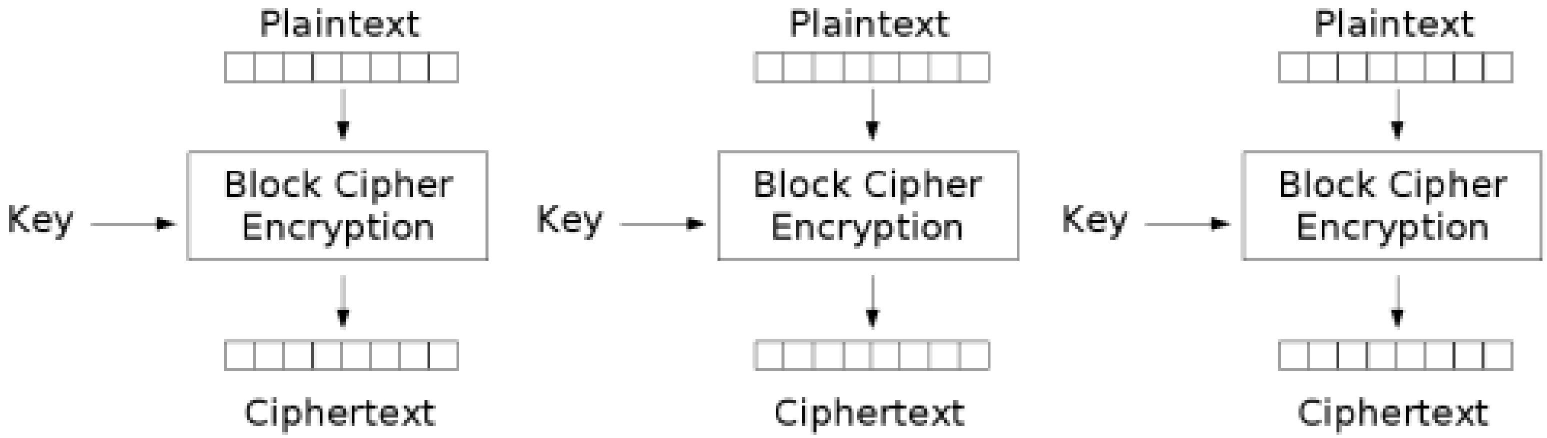
Dec For $c = (r, s)$, let $Dec_k(r, s) = F_k(r) \oplus s$.

- Remaining challenges: encrypt long messages with block ciphers
- $m = (m_1, \ldots, m_l)$ where $\forall i : |m_i| = n$ (padding if needed)
- Let's encrypt block m_i according to some *mode of operation* one by one

Modes of operation: ECB

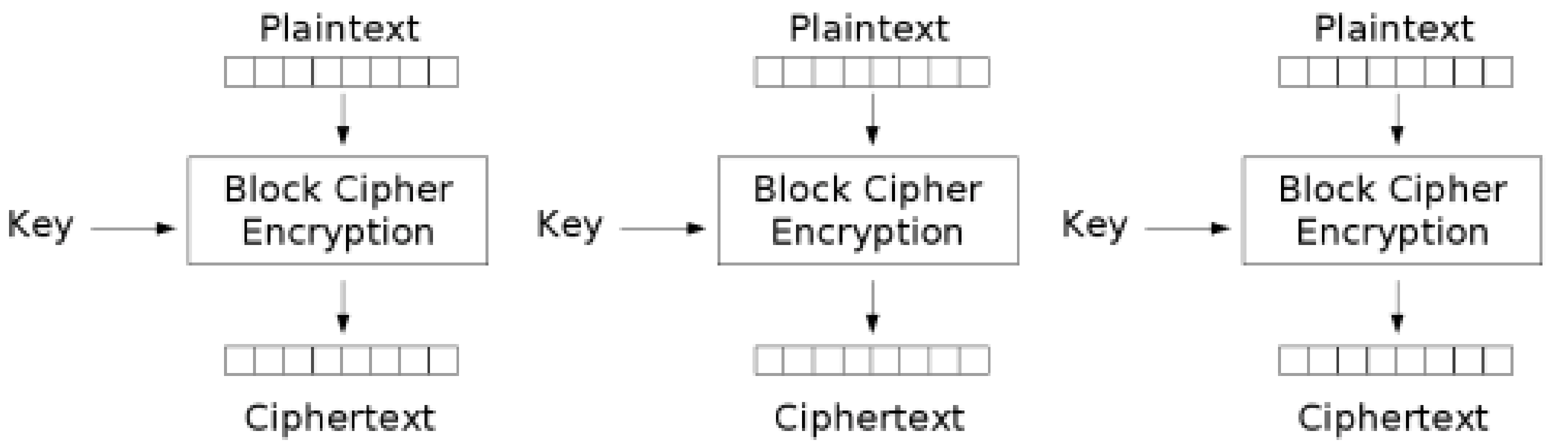
Electronic Code Book mode (ECB)

Let $F_k(.)$ be a strong PRP,
$Enc_k(m) = c = (F_k(m_1), \ldots, F_k(m_l))$.



Electronic Codebook (ECB) mode encryption

Modes of operation: ECB



Electronic Codebook (ECB) mode encryption

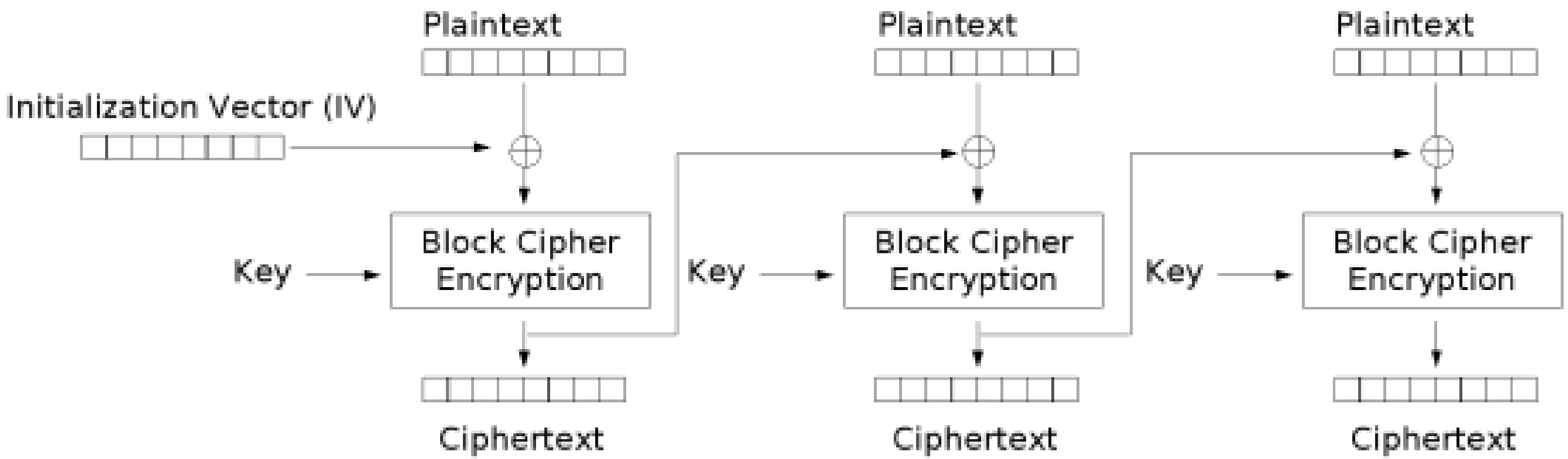
Properties

- Dec: $F_k(.)^{-1}$ should be efficiently computable
- deterministic $\Rightarrow$ no CPA-security
- eavesdropping: no security (repeated blocks)
- DON'T USE

Modes of operation: CBC

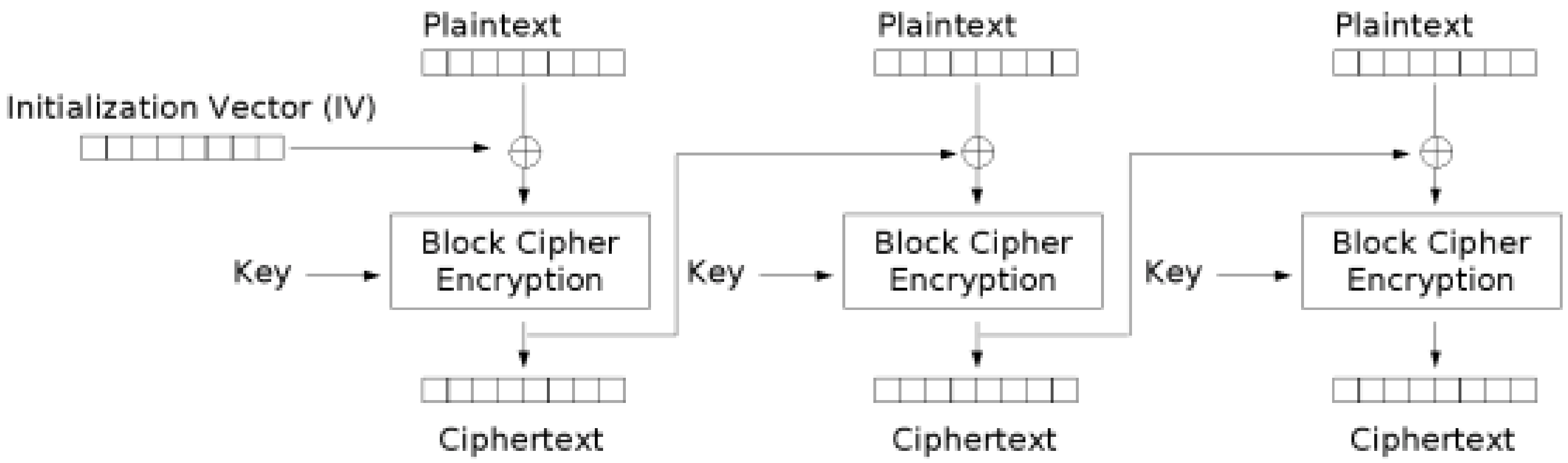
Cipher Block Chaining mode (CBC)

1. $IV \in_R \{0,1\}^n$ initialization vector
2. $c_0 = IV, c_i = F_k(m_i \oplus c_{i-1})$
3. $c = (c_0, c_1, \ldots, c_l)$



Cipher Block Chaining (CBC) mode encryption

CBC



Cipher Block Chaining (CBC) mode encryption

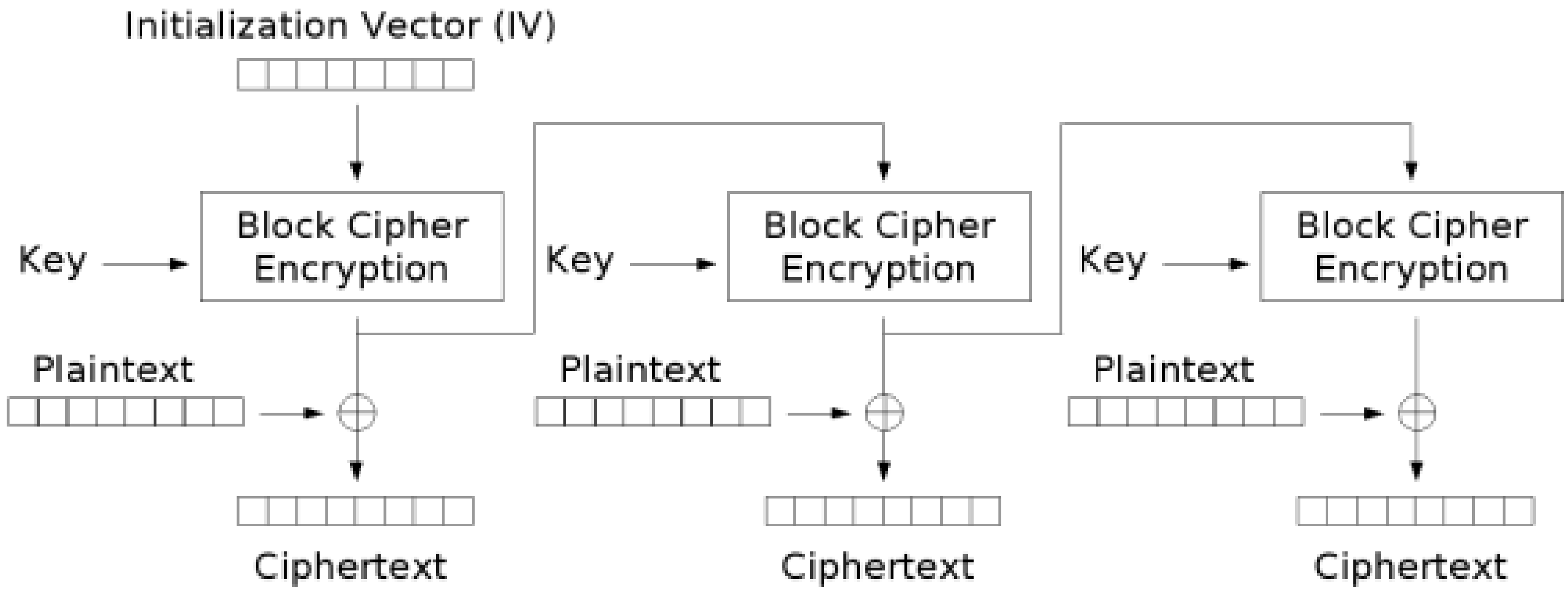
Properties

- probabilistic
- If $F_k(.)$ is strong PRP $\Rightarrow$ CPA-security
- sequential
- IV should be truly random (NO counter to be used here)

Modes of operation: OFB

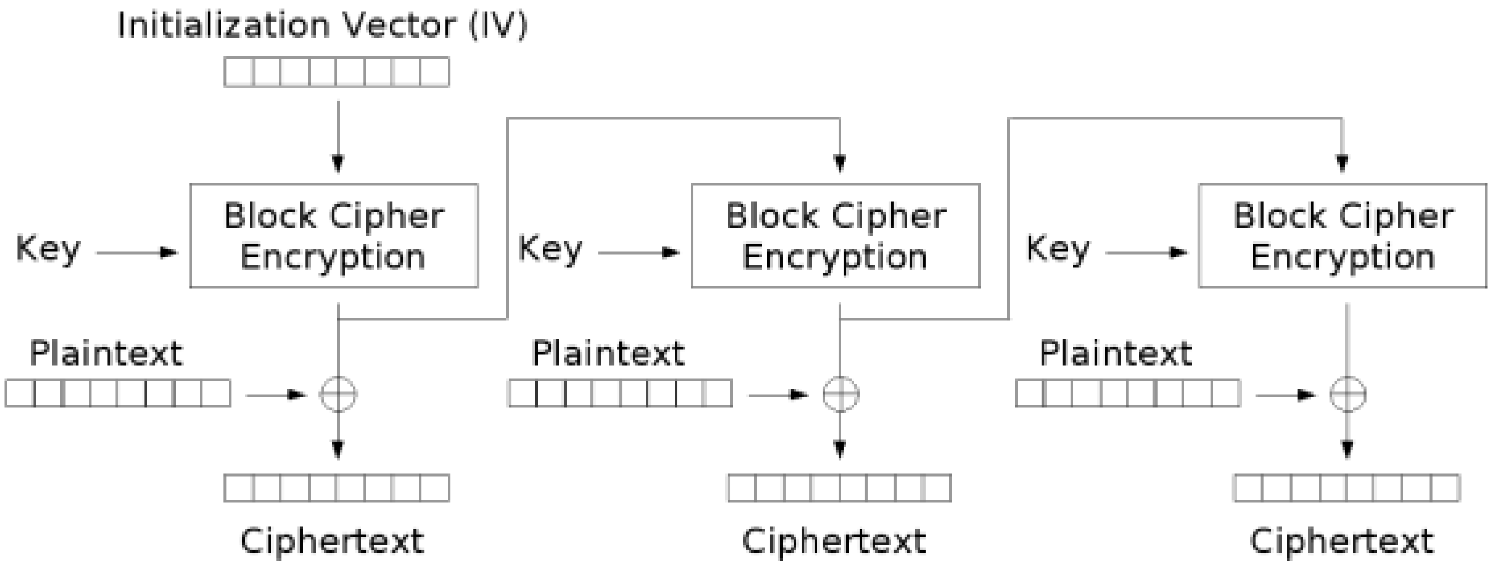
Output Feedback mode (OFB)

1. $IV \in_R \{0,1\}^n$ initialization vector
2. $r_0 = IV, r_i = F_k(r_{i-1}), c_i = m_i \oplus r_i$
3. $c = (c_0, c_1, \ldots, c_l)$



Output Feedback (OFB) mode encryption

OFB



Output Feedback (OFB) mode encryption

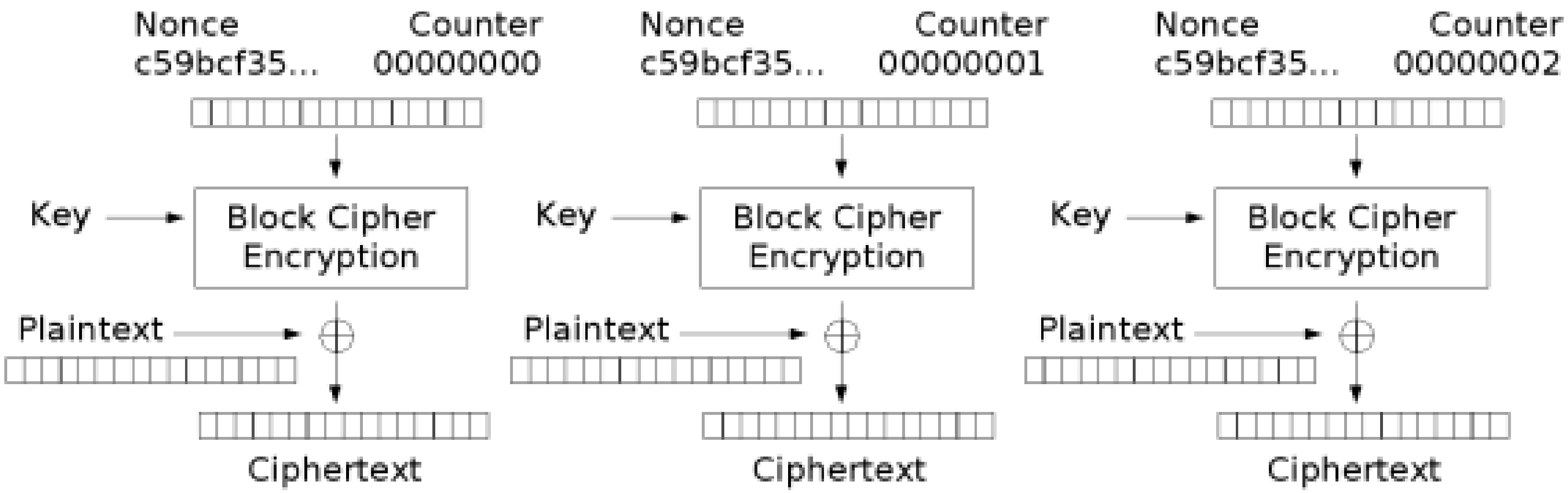
Properties

- probabilistic
- If $F_k(.)$ PRF $\Rightarrow$ CPA-security $\Rightarrow$ $F_k(.)$ has weaker assumption
- sequential
- preprocessing possible

Modes of operation: CTR

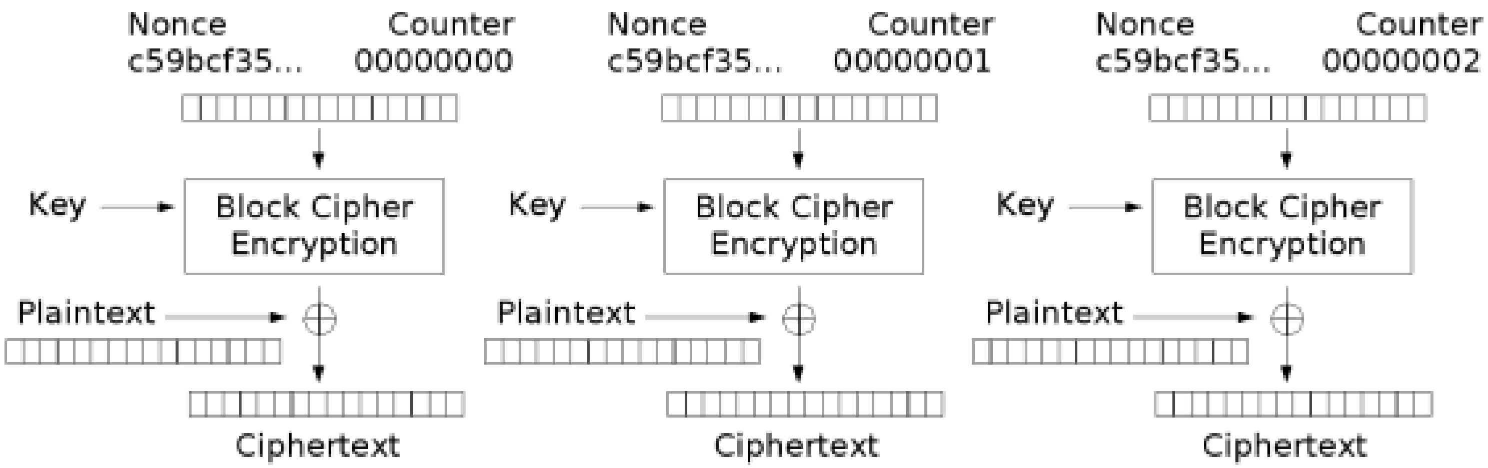
(Randomized) Counter mode (CTR)

1. $ctr = IV \in_R \{0,1\}^n$ initialization vector
2. $r_0 = IV, r_i = F_k(ctr + i), c_i = m_i \oplus r_i$
3. $c = (c_0, c_1, \ldots, c_l)$



Counter (CTR) mode encryption

CTR



Counter (CTR) mode encryption

Properties

- probabilistic
- if $F_k(.)$ is a PRF $\Rightarrow$ CPA-security $\Rightarrow$ $F_k(.)$ has weaker assumption
- parallelization
- preprocessing
- random access (decrypt ith block only)

Block or stream?

- block-size and security
- further modes
- modification of ciphertext: possible attacks

Block vs stream

- OFB and CTR mode makes stream cipher operation possible
- generates PR sequences
- better understood, more attack attempts, well-established
- stream cipher: can be faster

Block or stream?

- block-size and security
- further modes
- modification of ciphertext: possible attacks

Block vs stream

- OFB and CTR mode makes stream cipher operation possible
- generates PR sequences
- better understood, more attack attempts, well-established
- stream cipher: can be faster

Recap: block ciphers

- **Goal**: encryption of a fixed length message (block)

Tool: strong pseudorandom permutation family (PRP)

- similar to PRF
- maps n-bit strings to n-bit strings **bijectively**
- for $k \in \{0,1\}^n$, $F_k(.)$ is a member of the family
- a random element of the family is PPT-indistinguishable from a random function using the functions and the inverses.

Recap: block cipher

Block cipher

Let F be a strong PRP

Gen $k \in_R \{0,1\}^n$

Enc for key $k \in \{0,1\}^n$ and message $m \in \{0,1\}^n$ and $r \in_R \{0,1\}^n$: let $c = Enc_k(m) = (r, F_k(r) \oplus m)$.

Dec for key k and ciphertext $c = (r, s)$, we have $Dec_k(r, s) = F_k(r) \oplus s$.

Feistel network

Basic idea

- Goal: to encrypt $m \in \{0,1\}^n$, n: block length
- $t+1$ rounds of iterative computation for encryption
- subkeys obtained from k for each round
- F: round function
- mix and change subblocks of the message

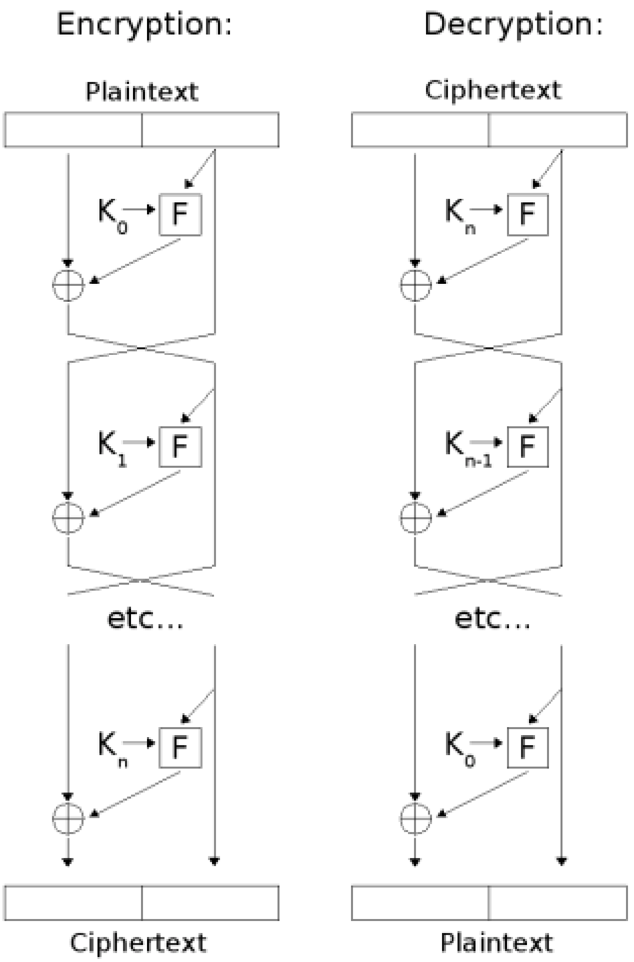
Feistel network

Feistel network

Gen $F : \{0,1\}^n \times \{0,1\}^n \Rightarrow \{0,1\}^n$ round-function, $k_0, k_1, \ldots, k_t$ subkeys

Enc $m = (L_0, R_0), i = 0, .., t+1 :$
$L_{i+1} = R_i, R_{i+1} = L_i \oplus (F_{k_i}(R_i))$
$c = (L_{t+1}, R_{t+1})$

Dec $i = n, n-1.., 0 : R_i = L_{i+1}, L_i = R_{i+1}(F_{k_i}(L_{i+1}))$
$m = (L_0, R_0)$

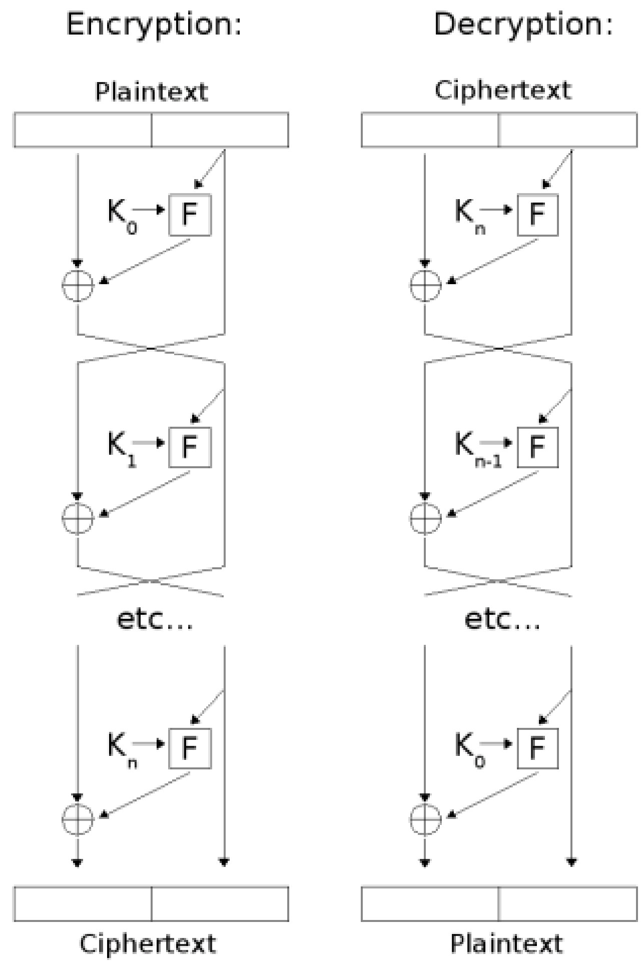


Feistel Cipher

Feistel networks

Properties

- if $F(.)$ is PRF $\Rightarrow$ PRP after 3 rounds, strong PRP after 4 rounds
- $F(.)$ any function (not necessarily invertible)
- unbalanced versions
- randomized ciphertext
- format preserving encryption
- GOST, DES, RC5, Blowfish, ...



Feistel Cipher

Feistel network

GOST

- block length: 64 bit, key size: 256 bit, 32 rounds
- round function: $F_{k_i}(m_i) : m_i \boxplus k_i \longrightarrow$ S-box $\longrightarrow <<< 11$
- 8 S-boxes of size 4x4

Properties

- soviet origins
- theoretical break
- practical break: 2^{192} time for 2^{64} pieces of data

Substitution-permutation networks

Basic idea

- encrypt $m \in \{0,1\}^n$, where n is the block lentgh
- several rounds
- subkeys generated from k
- S(ubst)-box and P(ermut)-box structure (changes with each round)
- S-box: substitution function
- P-box: permutation

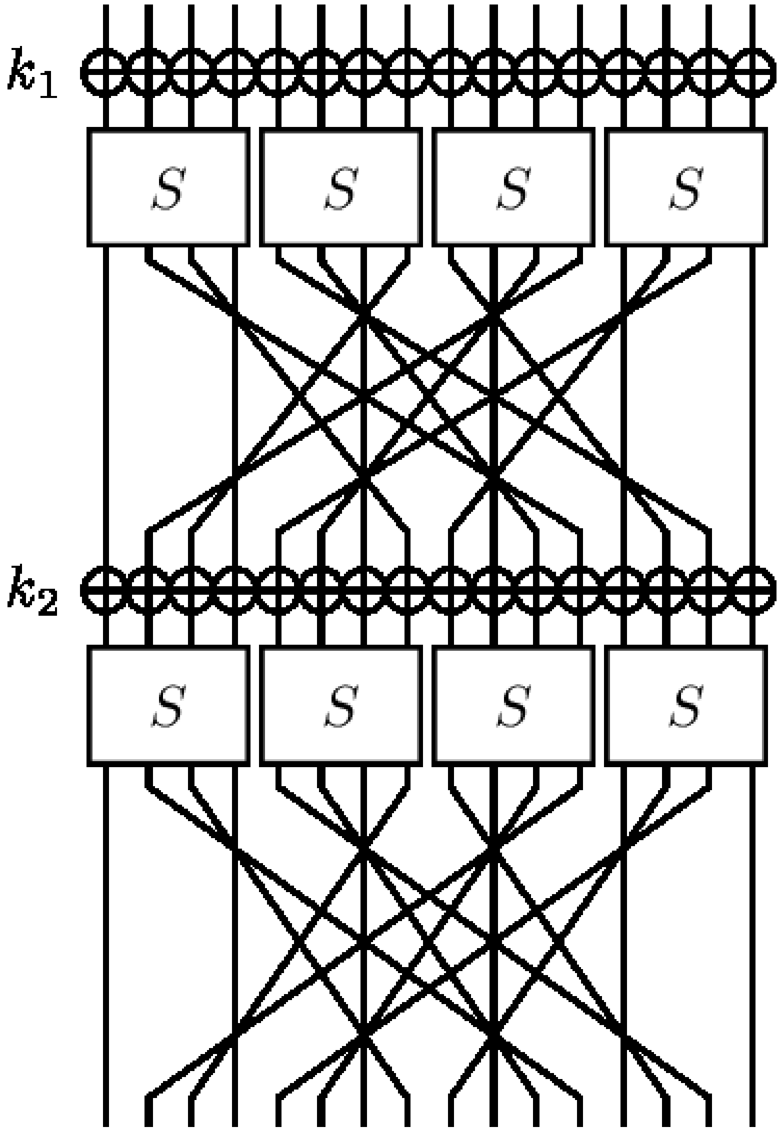
Substitution-permutation networks

S-P network

Gen $S_i : \{0,1\}^{n/l} \Rightarrow \{0,1\}^{n/l}, i = 1, \ldots, l; P : \{0,1\}^n \Rightarrow \{0,1\}^n$, $k_0, k_1, \ldots, k_t$ subkeys

Enc $m_0 = m, c_i = (c_i^1, \ldots, c_i^l) = m_i \oplus k_i, m_{i+1} = P(S_1(c_i^1), S_2(c_i^2), \ldots, S_l(c_i^l))$
$c = m_{t+1}$

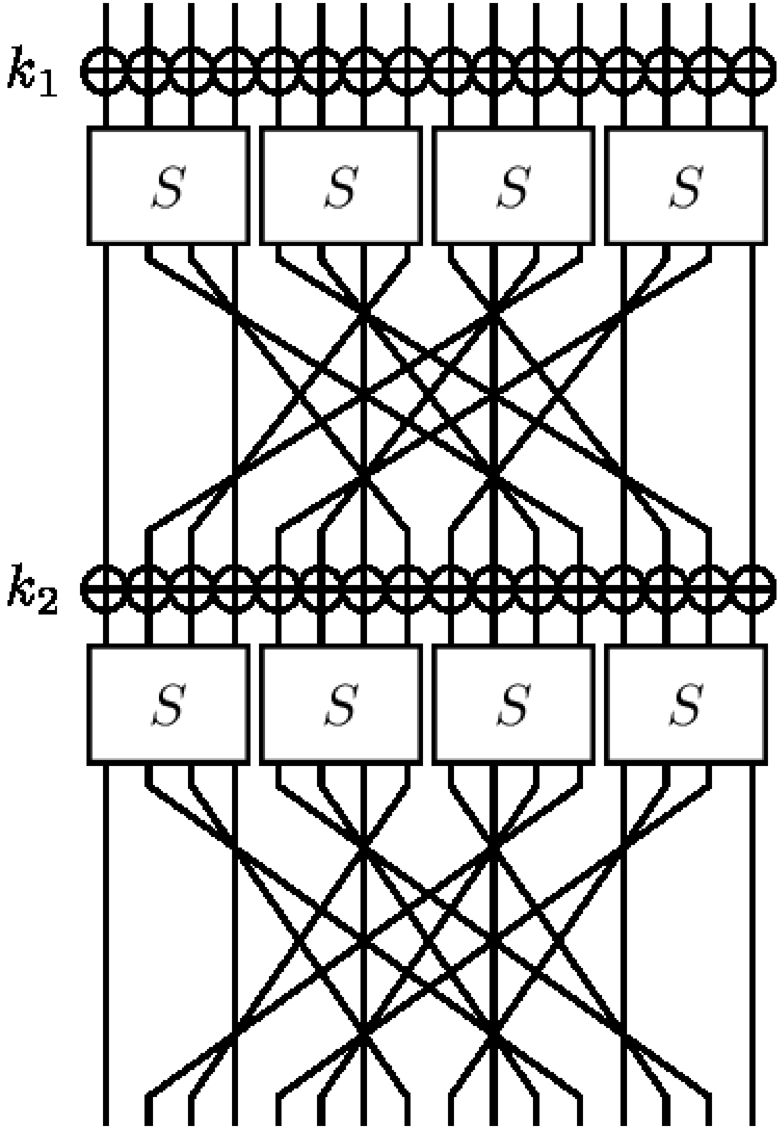
Dec Using $S_i^{-1}, i = 1, \ldots, l, P^{-1}$



Substitution-permutation networks

Properties

- S-box: bijective, avalanche effect, strong output-dependence
- P-box: permutation (shiffling)
- simple, hardware-friendly operations
- Goal: to fulfill the confusion-diffusion paradigm by Shannon
- AES (Rijndael), PRESENT, ...



Feistel vs. S-P

- Feistel: no need for invertible function
- S-P: parallel execution, might be faster on some hardware (not always, e.g. smart cards)
- $\exists$ Feistel combined with S-boxes

AES

- Rijndael block cipher's special case (Daemen, Rijmen)
- 2001: USA standard (AES competition)
- software - hardware efficiency
- no practical break known
- best known improvement over brute force: AES-128: reduced to 2^{126} operations
- side-channel attacks (protect hardware!)

Substitution-permutation networks

AES

- block length: 128 bits
- key size: 128 (10 rounds), 192 (12 rounds) or 256 bits (14 rounds)
- 4 x 4 byte-matrices (state)
- KeyExpansion: generate subkeys
- AddRoundKey: ...$\oplus$ subkey
- 9,11 v 13 rounds:
  1. SubBytes (S-box)
  2. ShiftRows (P-box)
  3. MixColumns (P-box)
  4. AddRoundKey
- final round: same without MixColumns

Substitution-permutation networks

AES

- block length: 128 bits
- key size: 128 (10 rounds), 192 (12 rounds) or 256 bits (14 rounds)
- 4 x 4 byte-matrices (state)
- KeyExpansion: generate subkeys
- AddRoundKey: ...$\oplus$ subkey
- 9,11 v 13 rounds:
  1. SubBytes (S-box)
  2. ShiftRows (P-box)
  3. MixColumns (P-box)
  4. AddRoundKey
- final round: same without MixColumns

AES background (theory)

Rijndael-test

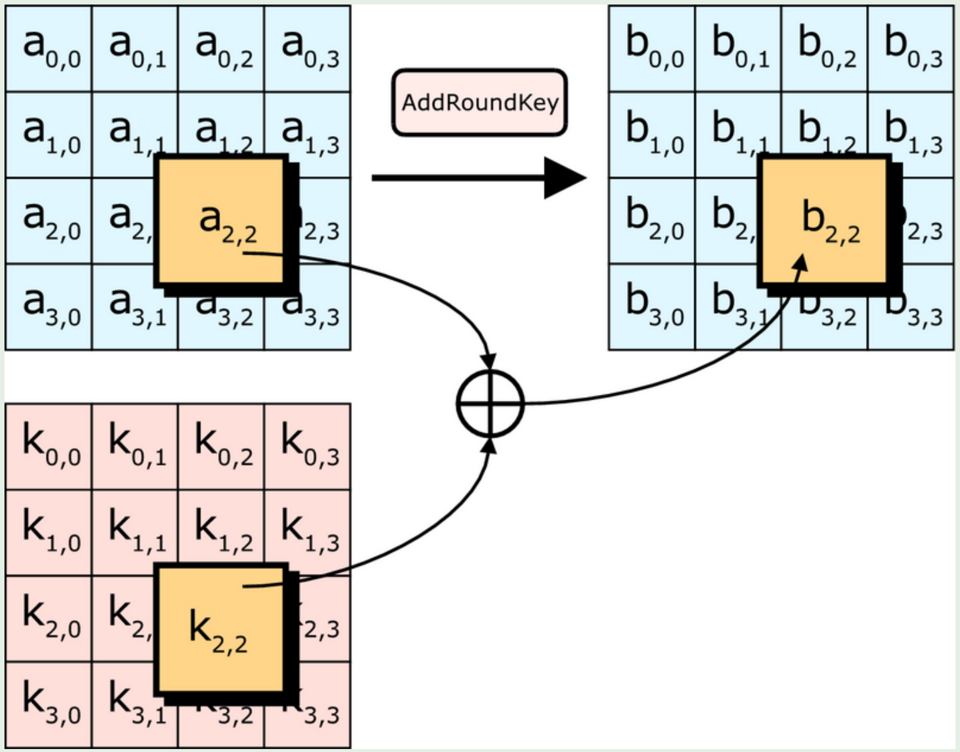
- $x^8 + x^4 + x^3 + x + 1$ polynomial over $\mathbb{Z}_2$ (irreducible)
- $\mathbb{F} = \mathbb{Z}_2[x]/\langle x^8 + x^4 + x^3 + x + 1\rangle$: 256-element field (4 basic operations on bytes)
- string to polynomial:
  $b = (b_0, b_1, \ldots, b_7) \longleftrightarrow b_0 + b_1 x + \cdots + b_7 x^7 \in \mathbb{F}$

AES key generation and expansion

KeyExpansion

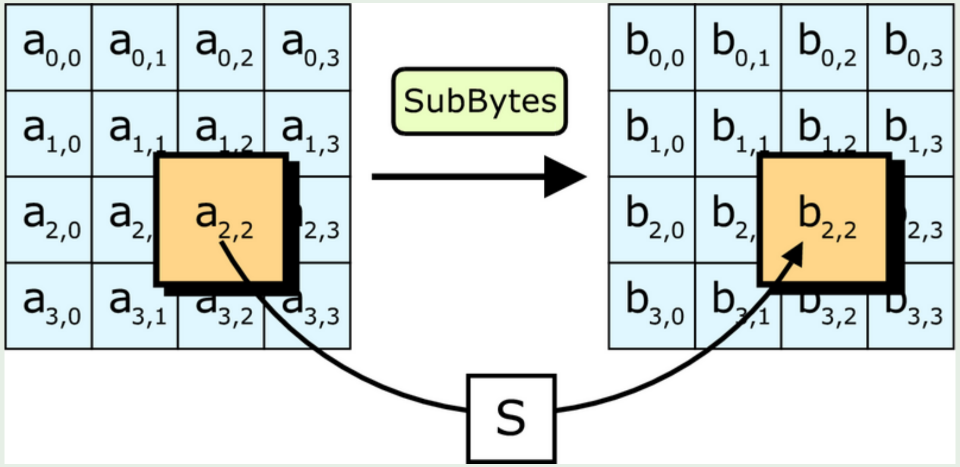
- subkeys for all rounds
- $k_{0,0}, \ldots, k_{0,3}, k_{1,0}, \ldots, k_{3,3}$
- skipping details

AddRoundKey



AES S-box

SubBytes

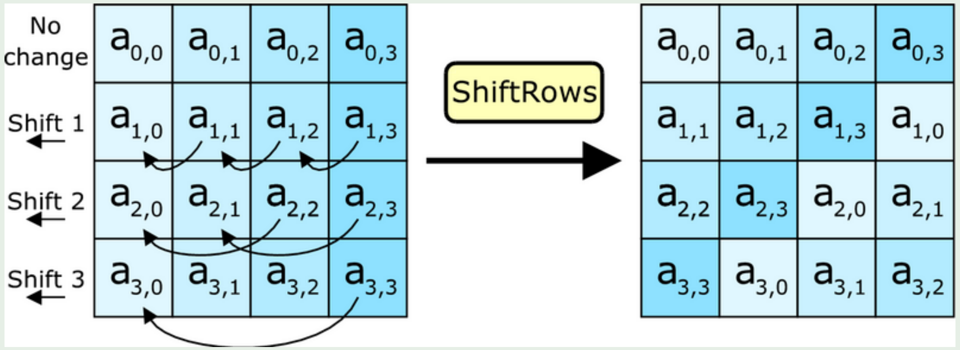


SubBytes

- $x \in \{0,1\}^8 : S(x) = Cx^{-1} + c : x^{-1} \in \mathbb{F}$
- $C \in \{0,1\}^{8x8}$ fixed invertible matrix
- $c \in \{0,1\}^8$ fixed vector
- $S(.)$ affine transformation without fixed point (linear + constant)
- inverse: $x = S^{-1}(y) = (C^{-1}(y-c))^{-1}$

AES P-box (1st half)

ShiftRows

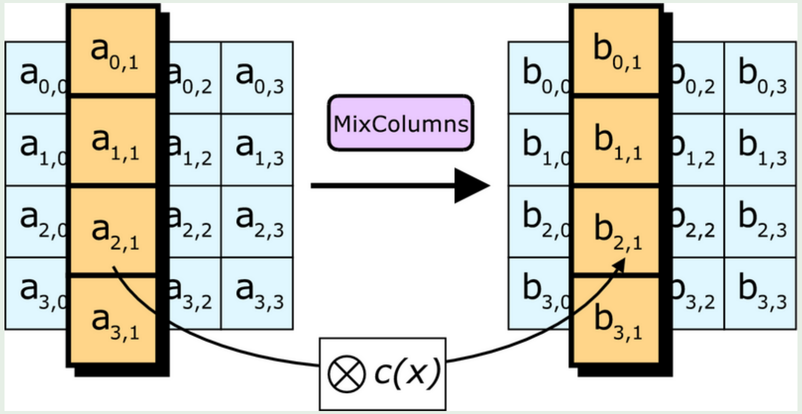


ShiftRows

- $<<<\ i$ in row i
- inverse easy

AES P-box (2nd half)

MixColumns



ShiftRows

- $a(x) = a_{3,1}x^3 + a_{2,1}x^2 + a_{1,1}x + a_{0,1}$
- $c(x) = 3x^3 + x^2 + x + 2$
- $a \otimes c \equiv a \cdot b \mod (x^4 + 1)$
- inverse: $c^{-1}(x) = 11x^3 + 13x^2 + 9x + 14 \mod (x^4 + 1)$

Hash functions: in data structures

- hash function is a compression function
- arbitrary input length
- $H : \{0,1\}^* \mapsto \{0,1\}^n$, typically $n = 128, 160, 256, etc$
- used in data structures and algorithms (set, map, etc.)
  - elements stored in a table of size k.
  - find and add operations in $O(1)$ time (amortized)
  - Key x stored in cell at index $H(x)$
  - Find x by computing $H(x)$ and looking at corresponding cell
- *collision*: $x \neq x' : H(x) = H(x')$
- collisions should not be too frequent
- good: distributes elements „evenly" accross the table

Hash functions: cryptography

- compress
- few collisions
- avoiding collisions
  - algo: good for running times, no strict avoidance, rather minimization
  - crypto: it's a must
- No special interest in values of x and $H(.)$ in algo
- Attacker may choose x in crypto
- Cryptographic hash: more of a challenge

Cryptographic hash functions

Definition

A collision of function $H(.)$ is a pair $x \neq x'$ with $H(x) = H(x')$.
A function $H(.)$ is collision-free, if any PPT attacker has only negligible probability of finding a collision.
A function $H(.)$ is a hash function if $H : \{0,1\}^* \mapsto \{0,1\}^n$.

Weaker security assumptions

1. Collision-free
2. *Second-preimage resistance*: for a given x, no PPT attacker can find another $x' \neq x : H(x') = H(x)$
3. *Preimage resistance*: for a given $y = H(x)$ which is obtained from a random (unknown) x, no PPT adversary can find $x' : H(x') = y$ („one-way function")

Cryptographic hash functions

Definition

A collision of function $H(.)$ is a pair $x \neq x'$ with $H(x) = H(x')$.
A function $H(.)$ is collision-free, if any PPT attacker has only negligible probability of finding a collision.
A function $H(.)$ is a hash function if $H : \{0,1\}^* \mapsto \{0,1\}^n$.

Weaker security assumptions

1. Collision-free
2. *Second-preimage resistance*: for a given x, no PPT attacker can find another $x' \neq x : H(x') = H(x)$
3. *Preimage resistance*: for a given $y = H(x)$ which is obtained from a random (unknown) x, no PPT adversary can find $x' : H(x') = y$ („one-way function")

Cryptographic hash functions

Design principles

- Collision-free
- Second-preimage resistance
- Preimage resistance
- *Avalanche effect*: small change in input $\Rightarrow$ large change in output
  - *Strict avalanche criterion:* changing an input bit $\Rightarrow$ changes all output bits with prob. 1/2
  - *Bit independence criterion:* $\forall i, j, k$: changing input bit $i \Rightarrow$ change in output bits j, k independent

Birthday attack

Let $H : \{0,1\}^* \mapsto \{0,1\}^n$ be a hash function. By computing (roughly) $2^{n/2}$ hash values, we will find a collision with prob. 1/2.

- Faster than brute force
- $\implies n \geq 160$
- „Breaking" a hash function usually means an attack which beats the birthday attack

Merkle-Damgård construction

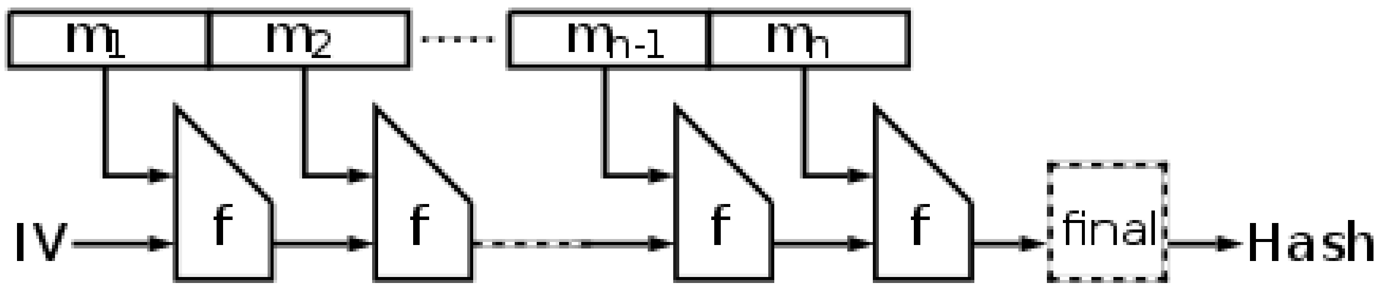
Basic idea

- In practice, input length is usually fixed
- this construction enables the use of arbitrary inputs
- Let $h : \{0,1\}^{2n} \mapsto \{0,1\}^n$ a hash function with fixed length inputs, $m \in \{0,1\}^*$, ahol $|m| = \ell < 2^n$
- construction uses chaining
- $\Rightarrow H(.)$ obtained with arbitrary inputs

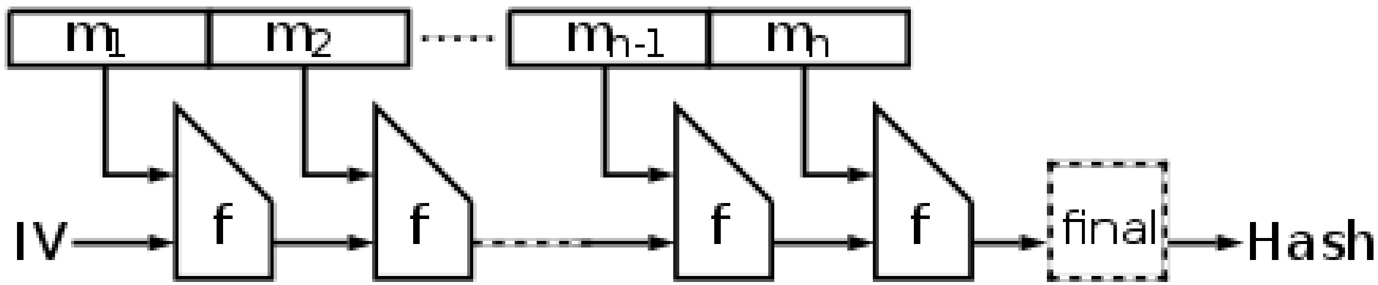
Merkle-Damgård construction

Merkle-Damgård transform

1. Split m into blocks of length n: $b := \lceil \frac{\ell}{n} \rceil$ és $m = (m_1|m_2|\ldots|m_b)$
2. Let $m_{b+1} := \ell \in \{0,1\}^k$, $z_0 := IV$
3. For $i = 1, \ldots, b+1$, compute $z_i := h(z_{i-1}|m_i)$
4. $H(m) := z_{b+1}$



Merkle-Damgård construction

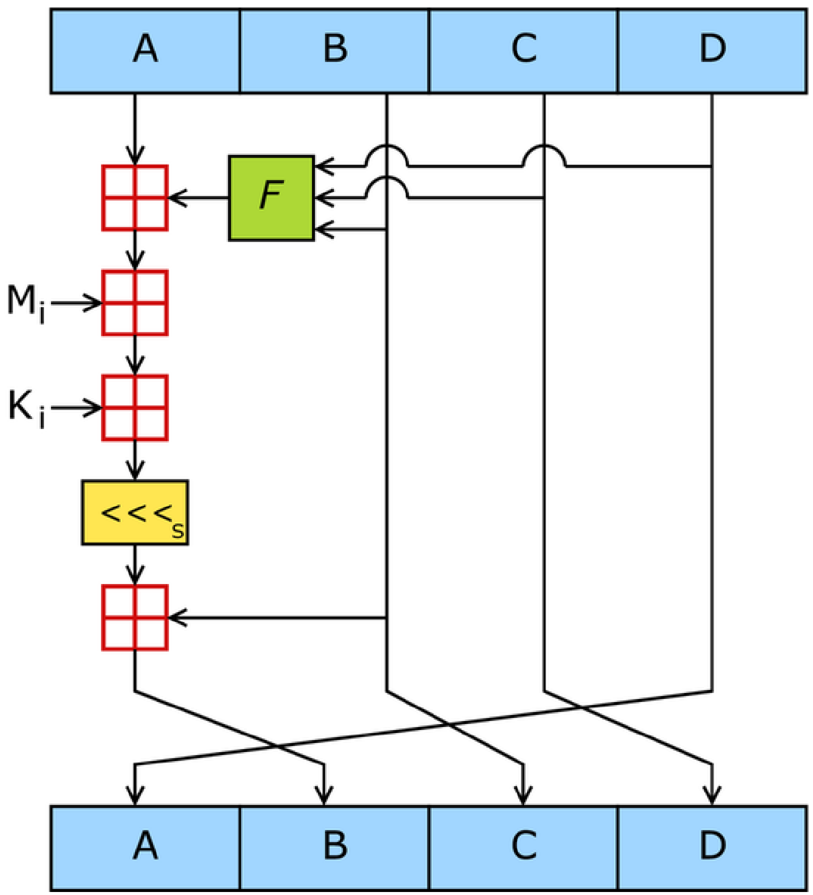


Properties

- in practice: suffices to have a hash for fixed input length
- theoretically: any compression ratio is fine
- $IV : z_0$ free to choose
- $h(.)$ collision-free $\Rightarrow H(.)$ collision-free

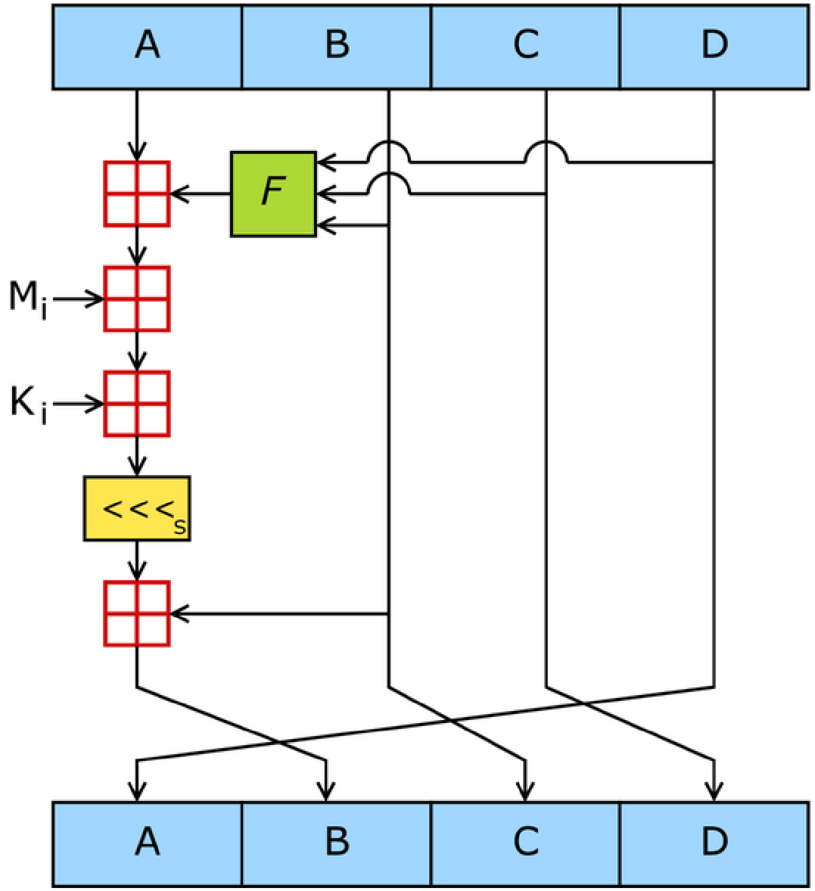
MD5 - description

- 512-to-128 bit compression and Merkle-Damgård
- Works on 32-bit words
- m broken up onto 512(=16*32)-bit blocks
- Operates on 128(=4*32)-bit „states”
- A, B, C, D fixed
- 4 rounds, 16 operations per round
- 4 non-linear F:
  1. $F(B,C,D) = (B \wedge C) \vee (\neg B \wedge D)$
  2. $G(B,C,D) = (B \wedge D) \vee (C \wedge \neg D)$
  3. $H(B,C,D) = B \oplus C \oplus D$
  4. $I(B,C,D) = C \oplus (B \vee \neg D)$
- M_i message block
- K_i costant, s shift parameter (varies for each operation)



MD5 – analysis

- Historical importance, collisions can be found!
- 128 bit output $\Longrightarrow$ birthday attack
- 1992 - MD5 published
- 1993 - „pseudo-collision” in compression function (IV -based attack)
- 1996 - collision in compression function
- 2004 - MD5CRK, distributed birthday attack
- 2004 - hash collision in under an hour (analytic attack)
- 2005 - collision in two X.509 certificates, different key, same MD5 hash
- 2010 - first published one/block collision



SHA family of hash functions

- SHA – Secure Hash Algorithm
- U.S. NSA, U.S. NIST

SHA-0 (1993)

- 160-bit output, 32-bit words, 80 rounds
- operations: $\oplus, \boxplus, \wedge, \vee, \lll$
- collision...

SHA-1 (1995)

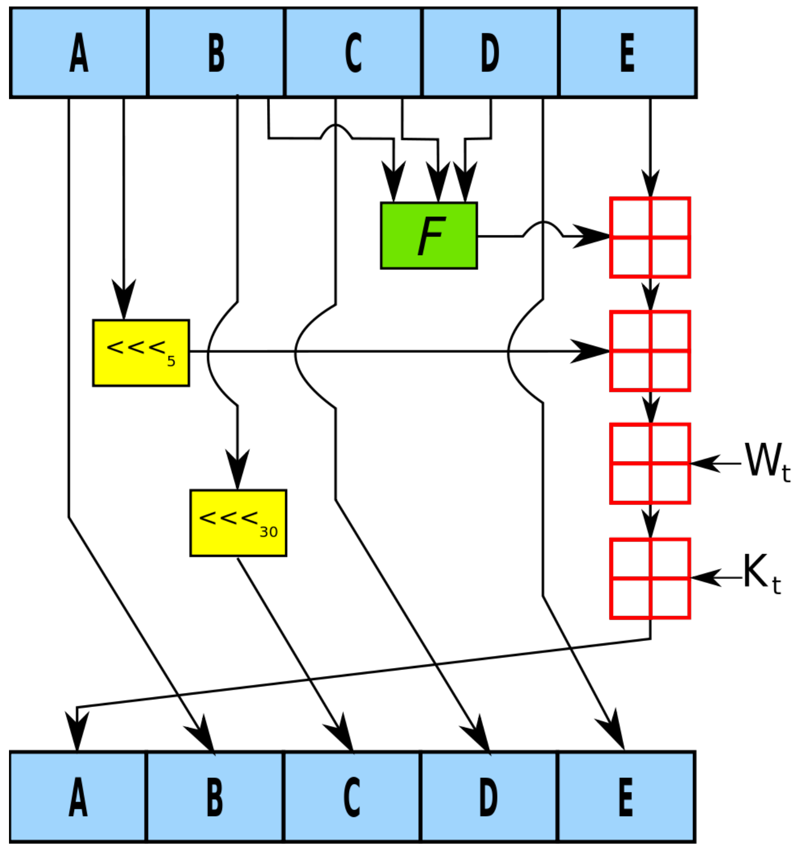
- 160-bit output, 32-bit words, 80 rounds
- more resistant, theoretical attack in 2^{61} time (2011)

SHA-2 (2001) = SHA-256/SHA-512

- 256/512-bit output, 32/64-bit words, 64/80 rounds
- no (known) collisions

SHA-3 (2014-)

- different design
- alternative to SHA-2



SHA-1, original diagram for Wikipedia created by User:Matt Crypto

NIST hash competition (2007 – 2012)

Similar to AES competition

Oct. 2008	deadline for submissions
Dec. 2008	Round 1: 51 candidates remain
Feb. 2009	NIST conference: submitted candidates
Jul. 2009	Round 2: 14 candidates
Aug. 2010	CRYPTO 2010: analyze round 2 candidates
Dec. 2010	Finalists announced

- performance: modest hardware requirements
- security: crypto/design weaknesses
- analysis: cryptanalysis by the entire crypto community
- diversity: various modes of operations and internal states

Dec. 2012	winner: Keccak
Aug. 2013	NIST announces changes compared to the standardized hash for „better security/performance”
Aug. 2015	Keccak is new SHA-3 hash standard

SHA-3/Keccak

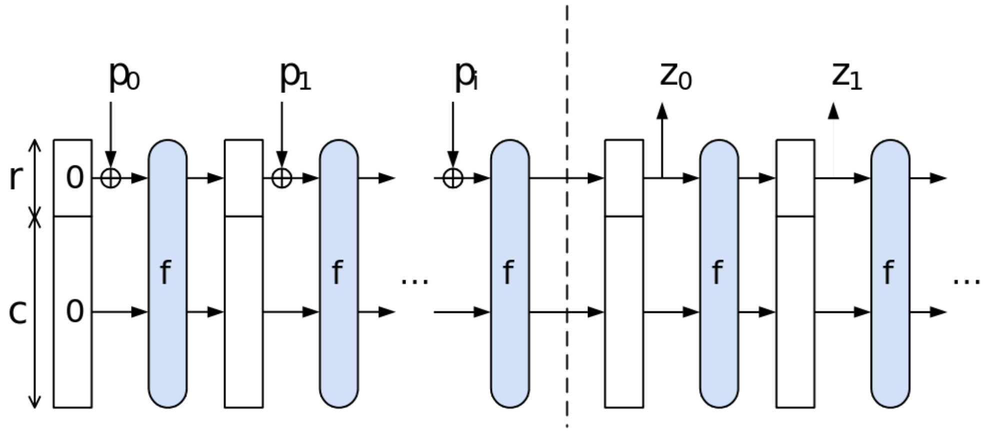


Diagram of a sponge construction from http://sponge.noekeon.org/

- Bertoni, Daemen, Peteers , Van Assche
- *Sponge construction* – permutation f of fixed length + padding rule:

1. m padded and broken up into r-bit blocks p_i
2. absorption: iteration of f: XOR of p_i with output of f from the previous block. All blocks are „absorbed" into internal state
3. squeeze out: extract output blocks z_i from (continuously updated) internal state.

Security vs. Integrity

- Secure communication
  - Alice sends message to Bob
  - in an open communication channel
  - security
  - **tools:** encryption
- Integrity
  - Alice sends message to Bob
  - in an open communication channel
  - authenticity (ID of caller, email address)
  - integrity
  - notice change in the message
  - preventing change not a crypto challenge (physical countermeasures)
  - **tool:** ???

Security vs. Integrity

- Secure communication
  - Alice sends message to Bob
  - in an open communication channel
  - security
  - **tools:** encryption
- Integrity
  - Alice sends message to Bob
  - in an open communication channel
  - authenticity (ID of caller, email address)
  - integrity
  - notice change in the message
  - preventing change not a crypto challenge (physical countermeasures)
  - **tool:** ???

Security vs. message authentication

- Stream cipher
  - Let $c := E_k(m) = G(k) \oplus m$ a ciphertext with $G(.)$ a PRG
  - Changing a single bit in $c \Longrightarrow$ changes same bit in m
  - Still secure
  - Similar with one-time pad
- Block cipher
  - OTR and CTR: same modification possible
  - more sophisticated for ECB, CBC
- encryption alone does not provide integrity
  - c hides m-et
  - BUT attacker can still mess around and modify c, thereby also m.
  - Any c corresponds to an m...
- need a new „layer”

Security vs. message authentication

- Stream cipher
  - Let $c := E_k(m) = G(k) \oplus m$ a ciphertext with $G(.)$ a PRG
  - Changing a single bit in $c \Longrightarrow$ changes same bit in m
  - Still secure
  - Similar with one-time pad
- Block cipher
  - OTR and CTR: same modification possible
  - more sophisticated for ECB, CBC
- encryption alone does not provide integrity
  - c hides m-et
  - BUT attacker can still mess around and modify c, thereby also m.
  - Any c corresponds to an m...
- need a new „layer”

Message Authentication Code (MAC): definition

Problem

- secret key shared between communicating parties
- authenticated message sent
- Has it been modified?
- Message Authentication Code (MAC)

Message Authentication Code (MAC): definition

Definition

A Message Authentication Code is a triple $(Gen, Mac, Vrfy)$ *with:*

- Gen *key generation: for security parameter* 1^n*, returns a key* k *with* $|k| \geq n$
- Mac *tag generation: for key* k *and message* $m \in \{0,1\}^*$ *returns a MAC-tag* $t := Mac_k(m)$
- $Vrfy$ *verification: for key* k*, tag* t *and message* m*, returns a bit* $b := Vrfy_k(m,t)$ *(*$b = 1$*, if* t *is a valid MAC-tag for* m*, otherwise 0.)*

The system fullfils the following correctnes definition:

$$Vrfy_k(m, Mac_k(m)) = 1.$$

MAC – security definition

What is an attack like?

- The attacker can:
  1. query MACs from Alice for various messages (examine how the message affects the tag)
  2. do some computation
  3. *forge* a MAC: a valid tag for a for some new message m (never queried before)
- security means that the attacker cannot perform the above attack efficiently

Definition

An authentication method is secure against adaptve chosen plaintext attack if any PPT adversary can only genarte a valid tag t for a message $m$with negligable probability even after querying several tags t' for messages $m' \neq m$.

MAC – security definition

Definition

An authentication method is secure against adaptive chosen plaintext attack if any PPT adversary can only genarte a valid tag t for a message $m$with negligable probability even after querying several tags t' for messages $m' \neq m$.

- too strong?
  - can query anything
  - any valid tag is a successful „break”
- in practice, only meaningful messages are of interest
- What’s „meaningful”
- *Replay attack*
  - solutions: timestamps or counter with m
  - drawback: synchronization or storage issues

MAC construction for fixes length meassages

Fixed length MAC

Let $PRF : \{0,1\}^n \mapsto \{0,1\}^n$ PRF. Then the following is a secure MAC.

Gen: $k \in_R \{0,1\}^n$

Mac: For key k and message $m \in \{0,1\}^n$, let the tag $t := PRF_k(m)$

$Vrfy$: Output 1 $\Leftrightarrow t = PRF_k(m)$

- if PRF secure, $\Rightarrow$ MAC secure
- drawback: fixed length m
- randomization (+ a few additional tricks): arbitrary length

MAC from hash: HMAC

HMAC

Let $h : \{0,1\}^{2n} \mapsto \{0,1\}^n$ and $H : \{0,1\}^* \mapsto \{0,1\}^n$ the Merkle-Damgård constructed hash. Let $IV, ipad, opad \in \{0,1\}^n$ fixed.

$$\begin{aligned} Gen&: \; k \in_R \{0,1\}^n \\ Mac&: \; t := h(h(IV|k \oplus opad)|H_{IV}(k \oplus ipad|m)) \\ Vrfy&: \; \text{output } 1 \Leftrightarrow t = Mac_k(m) \end{aligned}$$

- if h collision-free, then HMAC secure
- Merkle-Damgard not secure against so-called length extension attack

Symmetric vs. public-key crypto

Symmetric keys

- Common key k (secret)
- Both for encryption and decryption
- Secure channel without trusted third party

Public key cryptography

- pk, sk pair of keys (public, secret)
- pk at sender, sk at receiver
- distribute keys:
  - pk publicly sent (through authenticated channel)
  - pk broadcast, independent of sender
- multiple senders – one receiver
- 2-3 orders of magnitude slower :(

Definiition of public key scheme

Definition

A public key scheme is a triple $\Pi = (Gen, Enc, Dec)$ *with:*

- Gen, *the key generation, a prob. algorithm that has* 1^n *as input (security param.) and* (pk, sk) *as output (key pair). The public key is* pk, *the secret key is* sk, *and* $|pk|, |sk| \geq n$
- Enc, *the encryption, a PPT algo. with* pk *and message* $m \in \mathcal{M})$ *as inputs and* $c \in \mathcal{C}$, $c := Enc_{pk}(m)$ *as output (ciphertext)*
- Dec, *the decryption a deterministic algo. with* sk *and* $c \in \mathcal{C}$ *as inputs. The output is an element of* $\mathcal{M}$, $Dec_{sk}(c)$.

Properties of public key scheme

Correctness

We trivially need $\forall n, \forall pk, sk$ and $\forall m \in \mathcal{M}$, that

$$Dec_{sk} Enc_{pk}(m) = m.$$

.

Attacks against public key schemes

Definition (indistinguishability experiment with eavesdropping $PubK^{eav}_{\mathcal{A},\Pi}(n)$)

1. $Gen(1^n) = (pk, sk)$
2. *The adversary* $\mathcal{A}$ *issues messages* $m_0, m_1 \in \mathcal{M}$ *on input* pk*, where* $|m_0| = |m_1|$*.*
3. $k = Gen(1^n), b \in_R \{0,1\} : c = Enc_{pk}(m_b)$ *given to* $\mathcal{A}$
4. $\mathcal{A}$ *issues* $b' \in \{0,1\}$
5. $PubK^{eav}_{\mathcal{A},\Pi}(n) = 1$*, if* $b = b'$*, otherwise 0.*

Attacks against public key schemes

Definition

A scheme $\Pi = (Gen, Enc, Dec)$ is secure agains one eavesdropping if any PPT adversary $\forall \mathcal{A}$, $\exists e(.)$ negligible s.t.

$$P(PubK^{eav}_{\mathcal{A},\Pi}(n) = 1) \leq \frac{1}{2} + e(n).$$

Attacks against public key schemes

Definition (CPA indistinguishability experiment $PubK^{cpa}_{\mathcal{A},\Pi}(n)$)

1. $Gen(1^n) = (pk, sk)$
2. *The adversary $\mathcal{A}$ has oracle access to $Enc_{pk}(.)$ for pk, then issues m_0, m_1, with $|m_0| = |m_1|$*
3. $b \in_R \{0, 1\} : c = Enc_{pk}(m_b)$ *given to $\mathcal{A}$*
4. *$\mathcal{A}$ has renewed oracle access to $Enc_{pk}(.)$. Then issues $b' \in \{0, 1\}$*
5. $PubK^{cpa}_{\mathcal{A},\Pi}(n) = 1$, *if $b = b'$, otherwise 0.*

Attacks against public key schemes

Definition

A scheme $\Pi = (Gen, Enc, Dec)$ *is CPA-secure if for any PPT adversary* $\forall \mathcal{A}$, $\exists e(.)$ *negligible s.t.*

$$P(PubK^{cpa}_{\mathcal{A},\Pi}(n) = 1) \leq \frac{1}{2} + e(n).$$

Attacks against public key schemes

Definition (indistinguishability experiment with multiple eavesdroppings $PubK^{meav}_{\mathcal{A},\Pi}(n)$)

Slight modification of definition

1. *Adversary $\mathcal{A}$ issues* $M_0 = (m_{01}, \ldots, m_{0t}), M_1 = (m_{11}, \ldots, m_{1t})$ *sequences,* $\forall i : |m_{0i}| = |m_{1i}|$
2. $b \in_R \{0, 1\} : C = (c_1, \ldots, c_t) : c_i = Enc_{pk}(m_{bi})$ *is given to $\mathcal{A}$*

Attacks against public key schemes

Theorem

If Π is secure against one eavesdropping $\Rightarrow$ CPA-security folllows

Theorem

If Π is secure against one eavesdropping $\Rightarrow$ also secure against multiple eavesdroppings

Theorem

$\nexists \Pi$ perfectly secure scheme (i.e. $\forall \mathcal{A} : PubK^{eav}_{\mathcal{A},\Pi}(n) = 1/2$)

Number theory for cypto

Euler's totient function φ

- $\varphi(n) = |\{k : 1 \leq k \leq n, (k,n) = 1\}|$
- p prime: $\varphi(p) = p - 1, \varphi(p^m) = p^m - p^{m-1}$
- $\varphi(nm) = \varphi(n)\varphi(m)$, ha $(n,m) = 1$

Theorem (Euler–Fermat)

$\forall a : 1 \leq a \leq n, (a,n) = 1 \Rightarrow a^{\varphi(n)} \equiv 1 \mod n.$

Number theory for cypto

Theorem (Prime number theorem)

For $x > 0$ let $\pi(x)$ denote the number of primes up to x. We have $\pi(x) \sim \frac{x}{\log x}$

Corollary

$\exists c > 0 \forall n > 1 :$ *Number of n-bit primes roughly $c \cdot 2^{n-1}/n$.*

- n^2/c random picks will result in at least one prime with prob. $1 - 1/e^n$
- 2002: DPT primality test
- practice: PPT test
- e.g. Miller-Rabin

Textbook RSA

Textbook RSA

Gen

- For input 1^n, choose primes p, q with n-bits. Set $N = pq$
- Let $e \in \{2, \ldots, N-1\} : (e, \varphi(N)) = 1$
- Let $d \in \{2, \ldots, N-1\} : ed \equiv 1 \mod \varphi(N)$
- $pk = (N, e), sk = (p, q, d)$

Enc For message $m \in \mathbb{Z}_N^*$ and private key pk, let $c \equiv m^e \mod N$

Dec For ciphertext $c \in \mathbb{Z}_N^*$ and secret key sk, let $m \equiv c^d \mod N$

Seciruty of textbook RSA

Textbook RSA

Gen
- $N = pq, ed \equiv 1 \mod \varphi(N)$
- $pk = (N, e), sk = (p, q, d)$

Enc For $m \in \mathbb{Z}_N^*$ and pk, $c \equiv m^e \mod N$

Dec For $c \in \mathbb{Z}_N^*$ and sk, $m \equiv c^d \mod N$

Security

- correctness: $(x^e)^d) = x^{ed} \equiv x^{ed \bmod \varphi(N)} \equiv x^1 = x$
- Enc DPT $\Rightarrow$ no security unless randomization added

RSA

Factorization problem

For random RSA modulus input N, find $p, q : N = pq$.

RSA problem

For random RSA instance N, e, c, find $m : m^e \equiv c \mod N$.

Statement

Factoring tractable $\Rightarrow$ *RSA tractable. Note:* $\Leftarrow$ *only conjectured.*

RSA

Properties

- Rivest, Shamir, Adleman '76
- p, q 1024-bit Sophie-Germain primes ($2p + 1$ is also prime)
- $e = 2^{16} + 1$ (prím)
- PPP encryption: $m' = (r||m)$ with r fixed length random

Theorem

If RSA problem difficult $\Rightarrow$ *randomized RSA is CPA-secure*