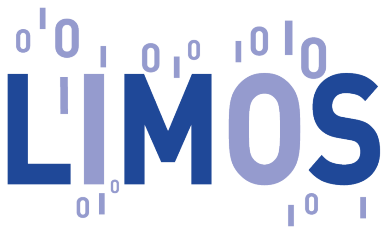
SSI Lecture 2
Public Key Cryptography

Pascal Lafourcade



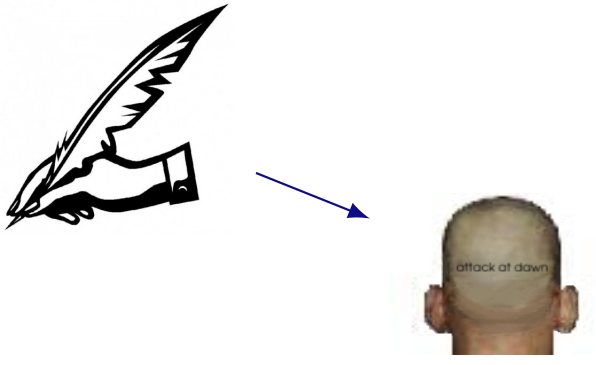
2022-2023

Outline

Outline

Information hiding

Information hiding



Stéganographie

Information hiding

Cryptography

Stéganographie

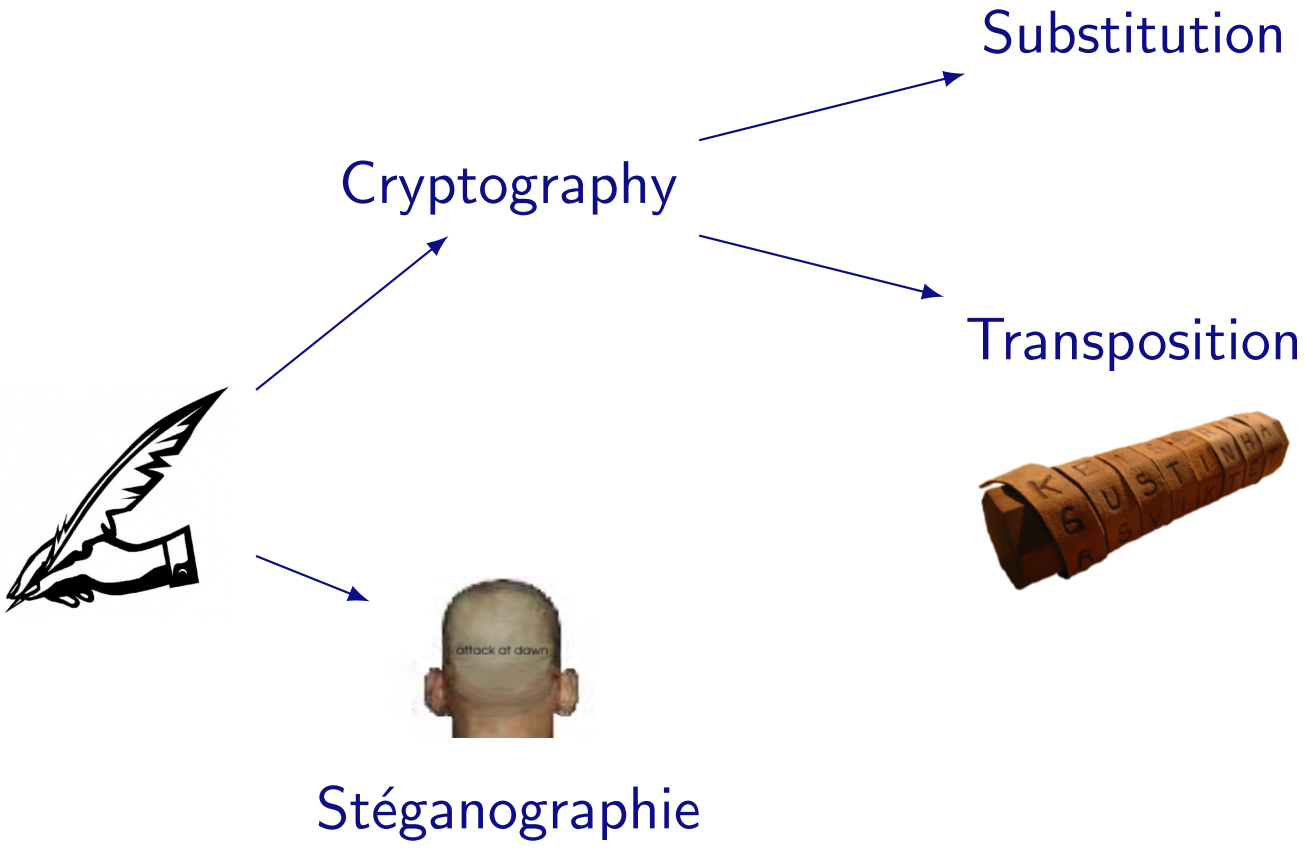
Information hiding

Cryptography

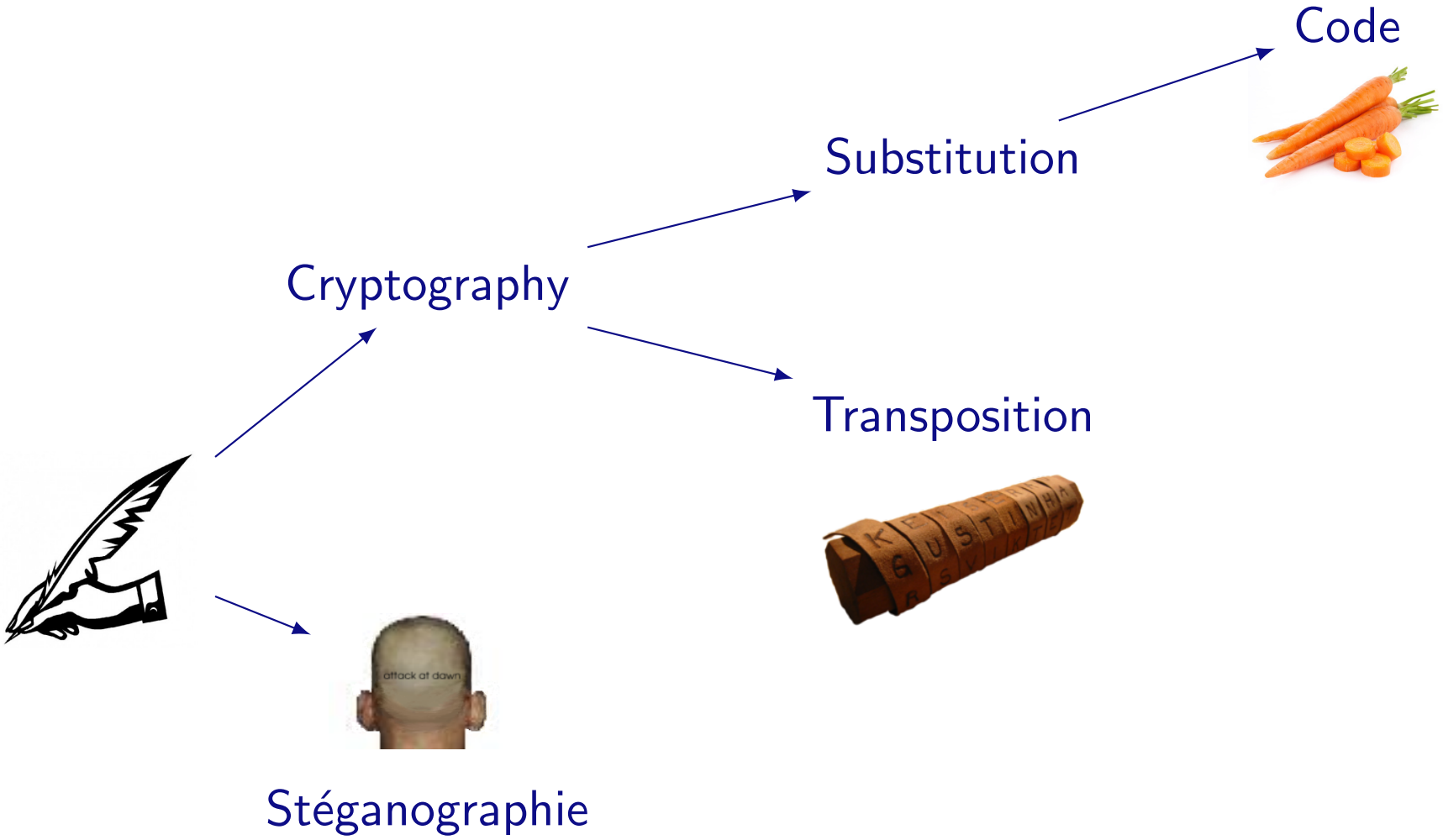
Transposition

Stéganographie

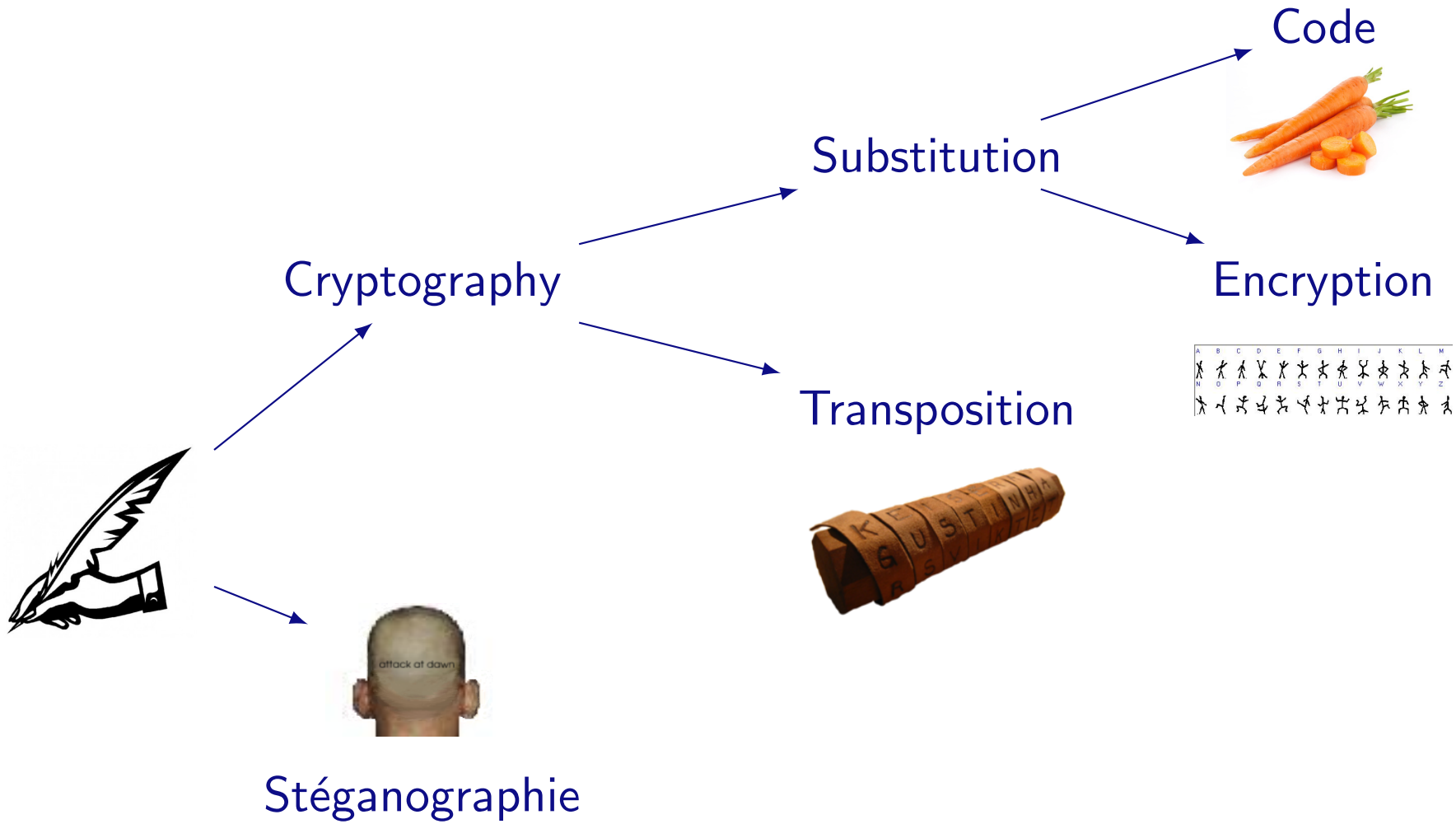
Information hiding



Information hiding



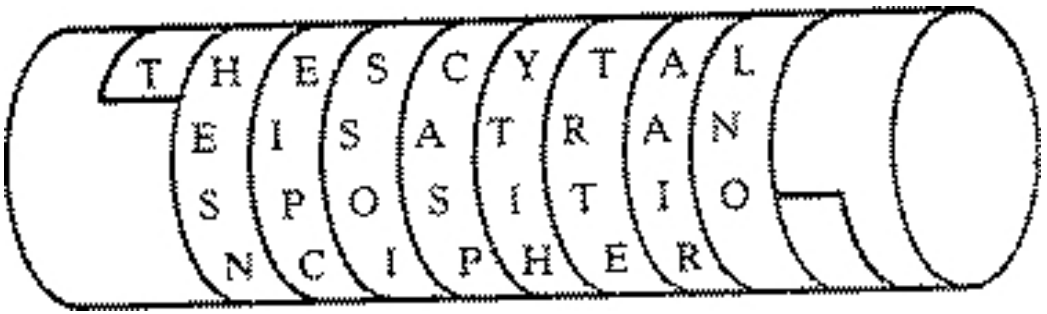
Information hiding



Egyptian time

Greeks and the Scythale

Greeks and the Scythale



Transposition

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.

Romans

Caesar Encryption
Substitution +3

Romans

Caesar Encryption
Substitution +3

Dyh Fhvdu

Romans

Caesar Encryption
Substitution +3

Dyh Fhvdu

Ave Cesar

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

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?

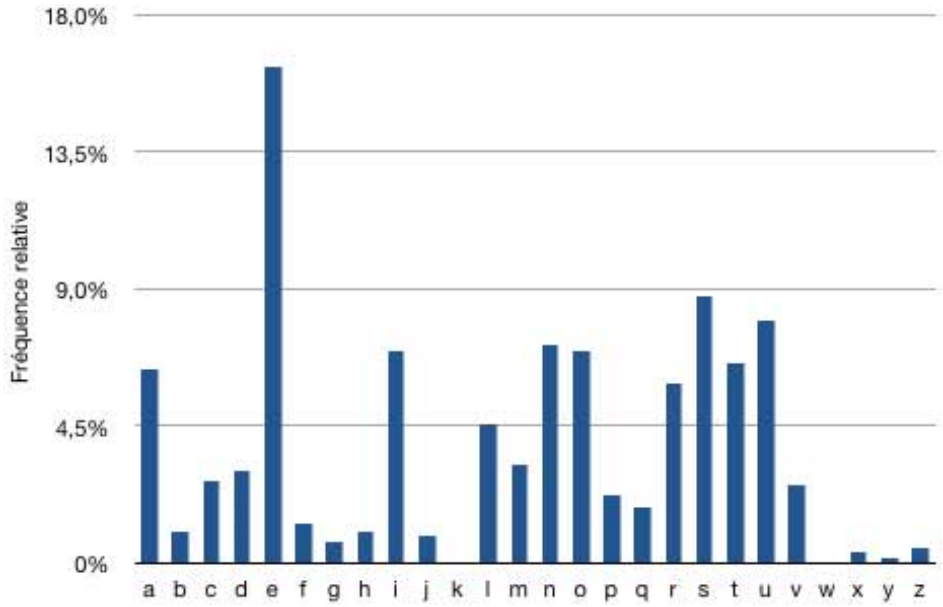
Is it secure?

Is it secure?

Key spaces are typically huge. 26 letters $\rightsquigarrow$ 26! possible keys.

Is it secure?

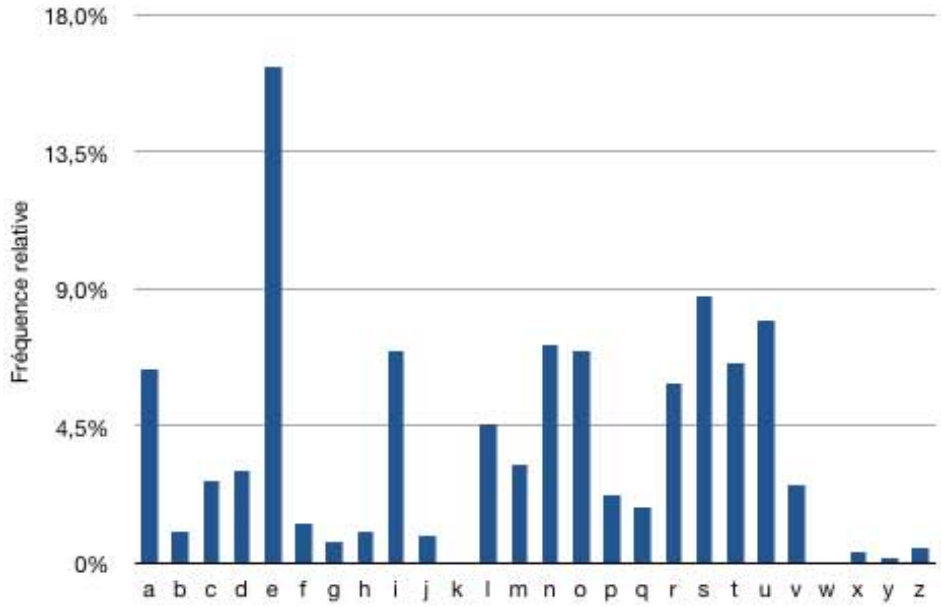
Key spaces are typically huge. 26 letters $\rightsquigarrow$ 26! possible keys.



Frequency analysis

Is it secure?

Key spaces are typically huge. 26 letters $\rightsquigarrow$ 26! possible keys.



Frequency analysis

Except for short, atypical texts

From Zanzibar to Zambia and Zaire, ozone zones make zebras run zany zigzags.

$\Rightarrow$ More sophistication required to mask statistical regularities

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

- 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 1553

- 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

m = THI SCI PHE RIS CER TAI NLY NOT SEC URE

then

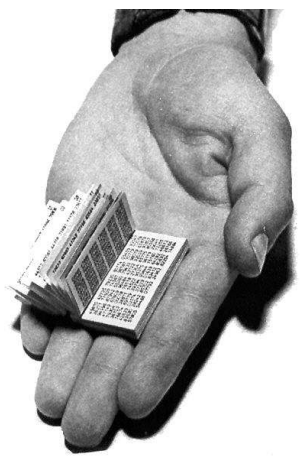
$E_e(m)$ = WOS VJS SOO UPC FLB WHS QSI QVD VLM XYO

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 $k_1 \cdots k_n$.

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

- Unconditional (information theoretic) security, if key isn't reused!



One-Time Pad (Vernam 1917)

Example:

$$\begin{array}{rcl} m & = & 010111 \\ k & = & 110010 \\ \hline c & = & 100101 \end{array}$$

Problem?

One-Time Pad (Vernam 1917)

Example:

$$\begin{array}{rcl} m & = & 010111 \\ k & = & 110010 \\ \hline c & = & 100101 \end{array}$$

Problem? Securely exchanging and synchronizing long keys.

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

Chiffrement : Enigma (Seconde guerre mondiale)

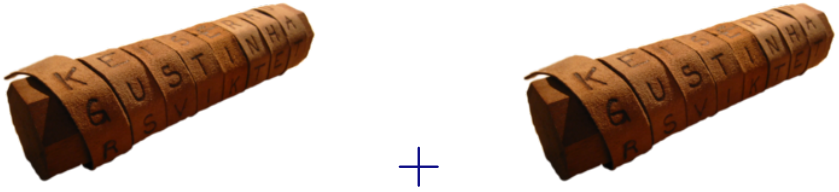

+

=

Chiffrement : Enigma (Seconde guerre mondiale)

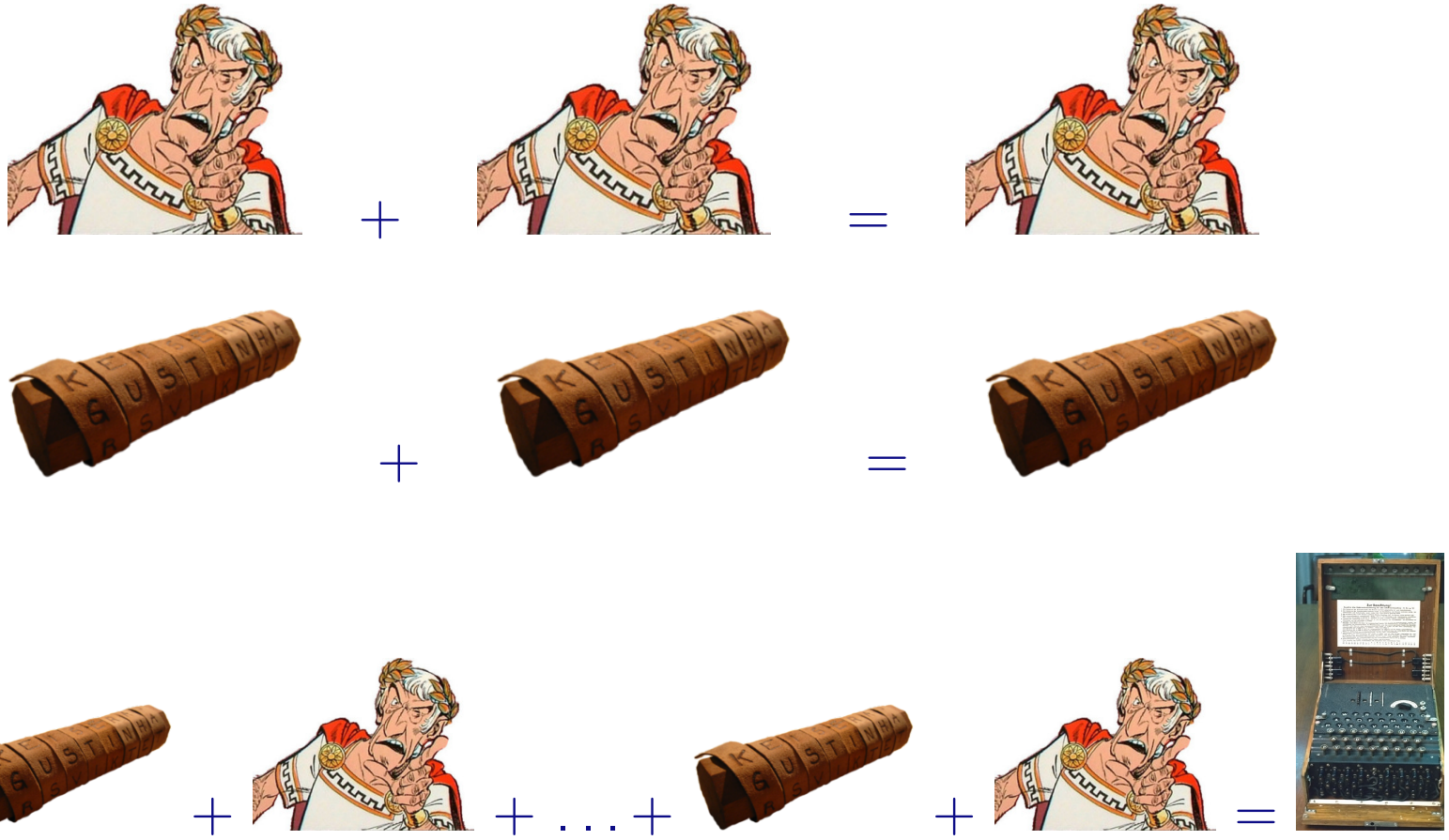
 +  = 

Chiffrement : Enigma (Seconde guerre mondiale)



Chiffrement : Enigma (Seconde guerre mondiale)

Chiffrement : Enigma (Seconde guerre mondiale)



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.

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)

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

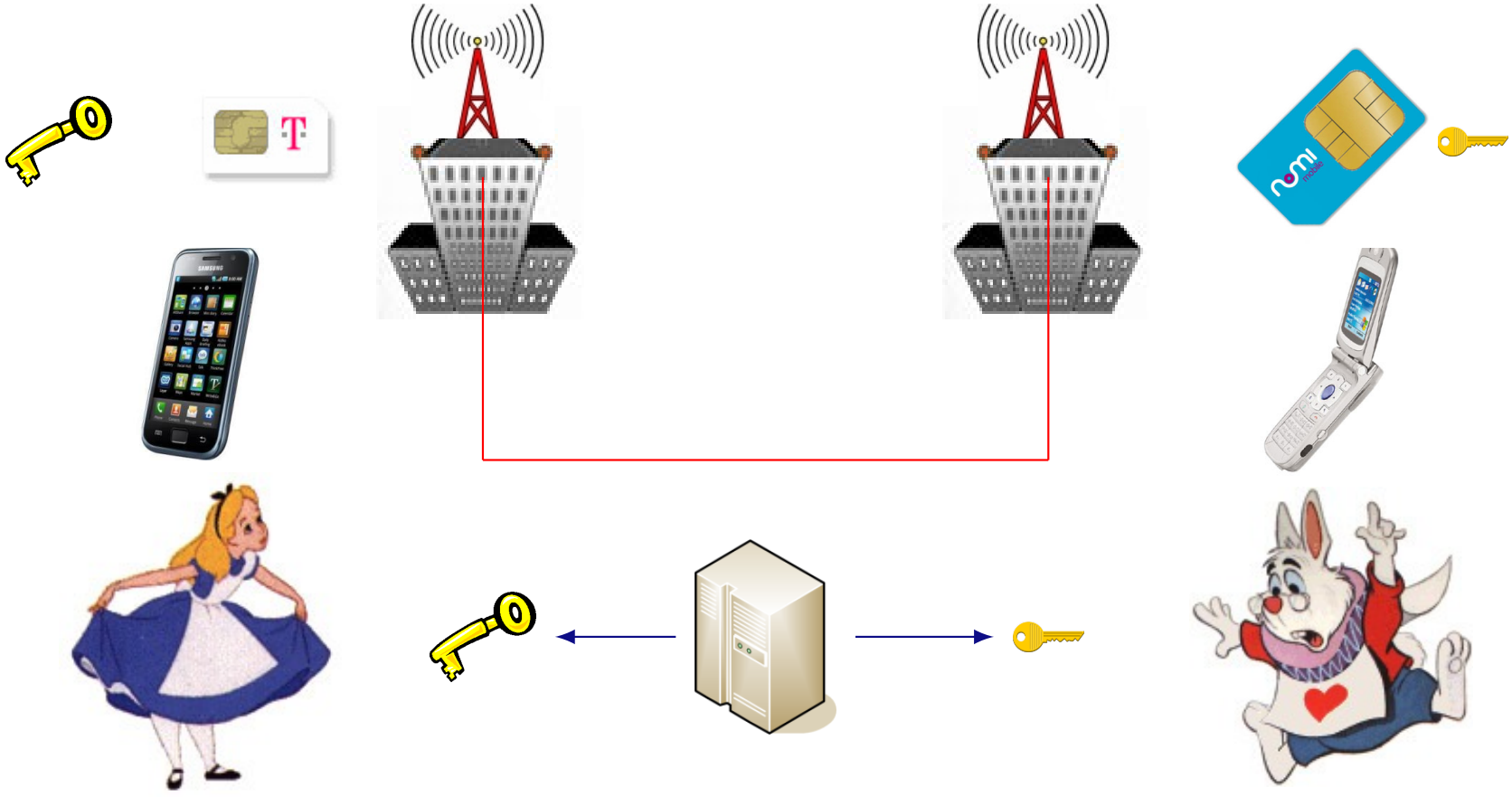
A good cipher design uses Confusion and Diffusion together

Symmetric Encryption

Examples

- DES
- AES

Cellphone Communications



Public Key Encryption

Examples

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

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

Outline

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).

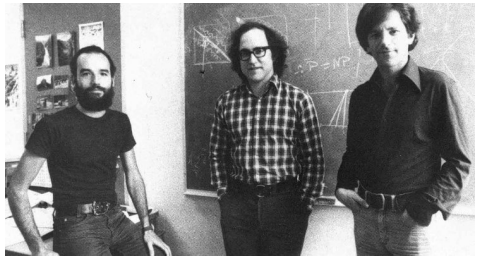
Integer Factoring

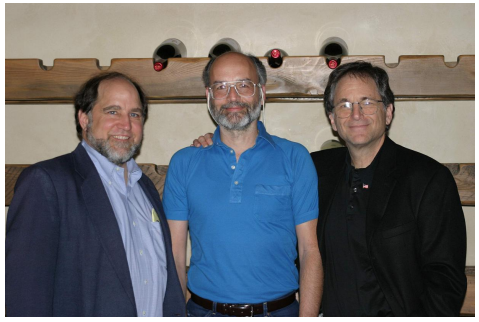
$\rightarrow$ Use of algorithmically hard problems.

Factorization

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

Rivest Shamir Adelmann (RSA 1978)





Let $n = pq$, p and q primes.

Public Key: (e, n)

Secret Key: d where $d = e^{-1} \mod \phi(n)$
and $gcd(e, \phi(n)) = 1$

Encryption: $c = m^e \mod n$

Decryption: $m = c^d$

Soundness

$c^d = m^{de} = m.m^{k\phi(n)} \mod n$

According to the Fermat Little Theorem $\forall x \in (Z/nZ)^*, x^{\phi(n)} = 1$

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.

Optimal Asymmetric Encryption Padding (OAEP)

The OAEP cryptosystem (K, E, D) obtained from a permutation f, whose inverse is denoted by g. And two hash functions:

$$G : \{0,1\}^{k_0} \to \{0,1\}^{k-k_0}$$

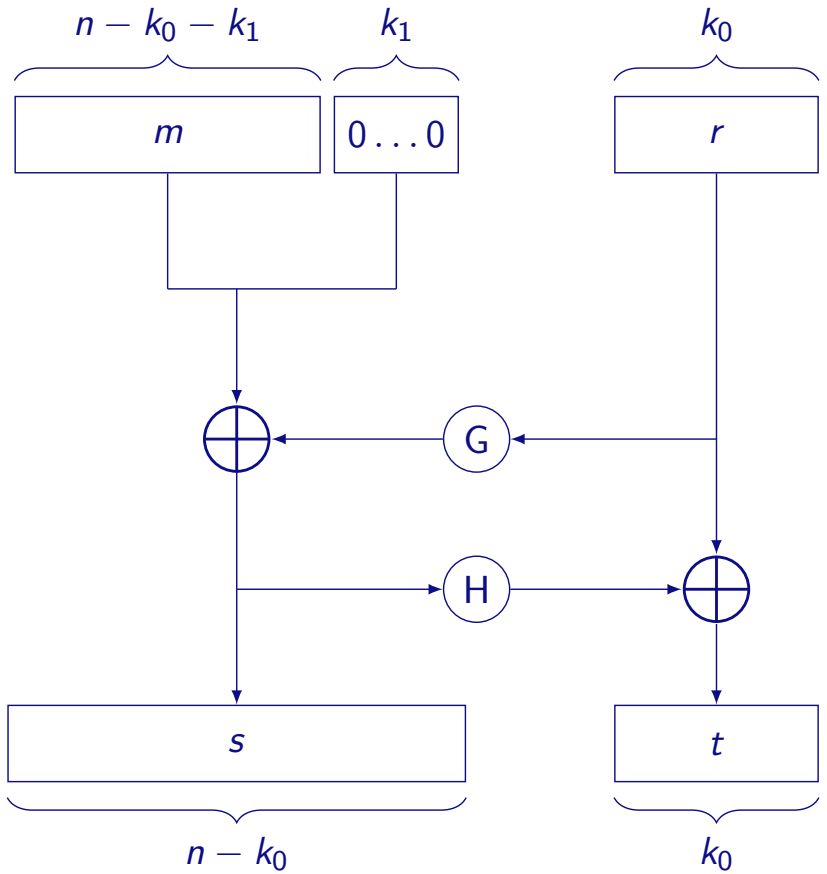
$$H : \{0,1\}^{k-k_0} \to \{0,1\}^{k_0}$$

$K(1^k)$: specifies an instance of the function f, and of its inverse g. The public key pk is therefore f and the private key sk is g.

OAEP: Encryption

$E_{pk}(m, r) = c = f(s, t)$ with $m \in \{0,1\}^n$, and $r \leftarrow \{0,1\}^{k_0}$

$$s = (m||0^{k_1}) \oplus G(r), t = r \oplus H(s)$$



OAEP: Decryption

$\mathcal{D}_{sk}(c)$

$$g(c) = (s, t)$$

$$r = t \oplus H(s)$$

$$M = s \oplus G(r)$$

If $[M]_{k_1} = 0^{k_1}$, the algorithm returns $[M]^n$, otherwise it returns "Reject"

- $[M]_{k_1}$ denotes the k_1 least significant bits of M
- $[M]^n$ denotes the n most significant bits of M

Results and References

OAEP was first proved IND-CPA then IND-CCA1 and finally IND-CCA2 secure under some assumptions.

1. M. Bellare, P. Rogaway. "Optimal Asymmetric Encryption – How to encrypt with RSA". Extended abstract in Advances in Cryptology - Eurocrypt '94 Proceedings, LNCS Vol. 950, A. Springer-Verlag, 1995.
2. Victor Shoup. "OAEP Reconsidered". IBM Zurich Research Lab, September 18, 2001.
3. Eiichiro Fujisaki, Tatsuaki Okamoto, David Pointcheval, and Jacques Stern. "RSA– OAEP is secure under the RSA assumption". In J. Kilian, ed., Advances in Cryptology – CRYPTO 2001, vol. 2139 of LNCS, SpringerVerlag, 2001.
4. P. Paillier and J. Villar, "Trading One-Wayness against Chosen-Ciphertext Security in Factoring-Based Encryption", Advances in Cryptology – Asiacrypt 2006.

Others Cryptosystems

- Bellare & Rogaway'93:

$$f(r)||x \oplus G(r)||H(x||r)$$

- Zheng & Seberry'93:

$$f(r)||G(r) \oplus (x||H(x))$$

- OAEP'94 (Bellare & Rogaway):

$$f(s||r \oplus H(s))$$

  where $s = x0^k \oplus G(r)$
- OAEP+'02 (Shoup):

$$f(s||r \oplus H(s))$$

  where $s = x \oplus G(r)||H'(r||x)$.
- Fujisaki & Okamoto'99:

$$\mathcal{E}((x||r); H(x||r))$$

  where $\mathcal{E}$ is IND-CPA.

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)

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

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 Resistance

Existential forgery (existential unforgeability, EUF):

GOAL: Forge at leat one couple (m, σ) is difficult.

Selective forgery (selective unforgeability, SUF):

m is imposed by the challenger before the attack.
GOAL: Forge at leat one couple (m, σ) is difficult.

Universal forgery (universal unforgeability, UUF):

GOAL : **For any message** m, forge (m, σ) is difficult.

Exercice RSA

Show that RSA signature is not EUF secure:

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

Hence $m' = m1 \cdot m2$ where $\sigma(m') = \sigma(m1 \cdot m2)$

Exercice RSA

Show that RSA signature is not EUF secure:

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

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: $c^d \mod n$

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

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

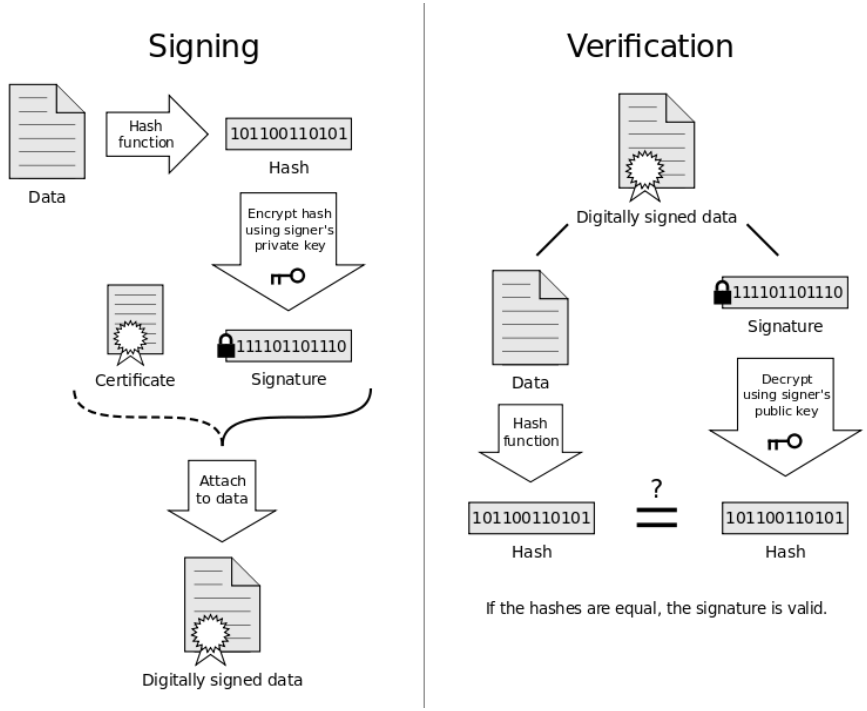
RSA Blind Signature

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

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

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

$$\begin{aligned} g^{H(m)} &\equiv g^{xr} g^{ks} \\ &\equiv (g^x)^r (g^k)^s \\ &\equiv (y)^r (r)^s \pmod{p}. \end{aligned}$$

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 $w = s^{-1} \bmod q$
- Calculate $u_1 = H(m) \cdot w \bmod q$
- Calculate $u_2 = r \cdot w \bmod q$
- Calculate $v = ((g^{u_1} y^{u_2}) \bmod p) \bmod q$

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 $s = k^{-1}(H(m) + xr) \mod q$

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

Since g has order q (mod p) we have

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

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

Schnorr Signature

Key generation

- Choose a private signing key, x.
- The public verification key is $y = g^x$.

Signing M

- Choose a random k
- Compute $r = g^k$.
- Let $e = H(r \parallel M)$, $\parallel$ denotes concatenation.
- Let $s = k - xe$.

The signature is $\sigma = (s, e)$

Verification of $\sigma = (s, e)$

- $r_v = g^s y^e = g^{k-xe} g^{xe}$
- $e_v = H(r_v \parallel M)$
- If $e_v = e = H(g^k \parallel M)$ then the signature is verified.

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: $e : G_1 \times G_2 \to G_T$, 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

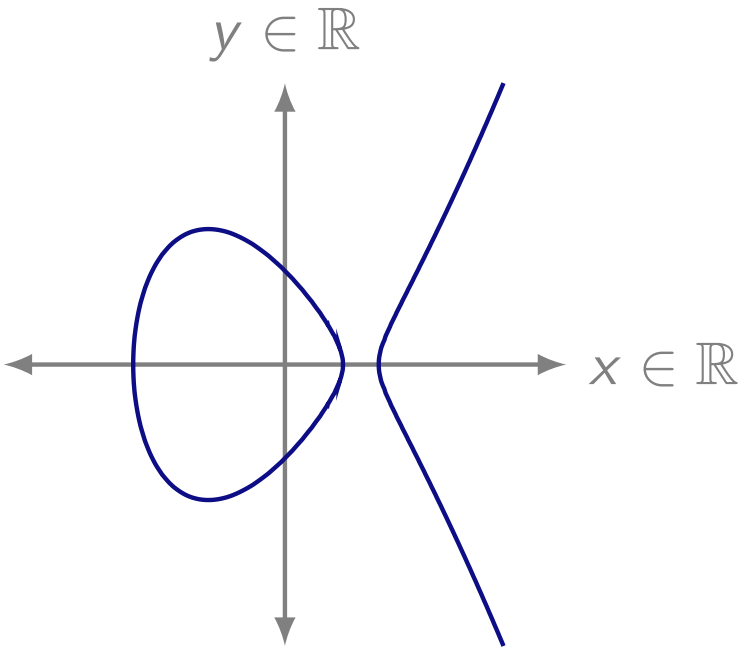
Boneh-Lynn-Shacham 2004

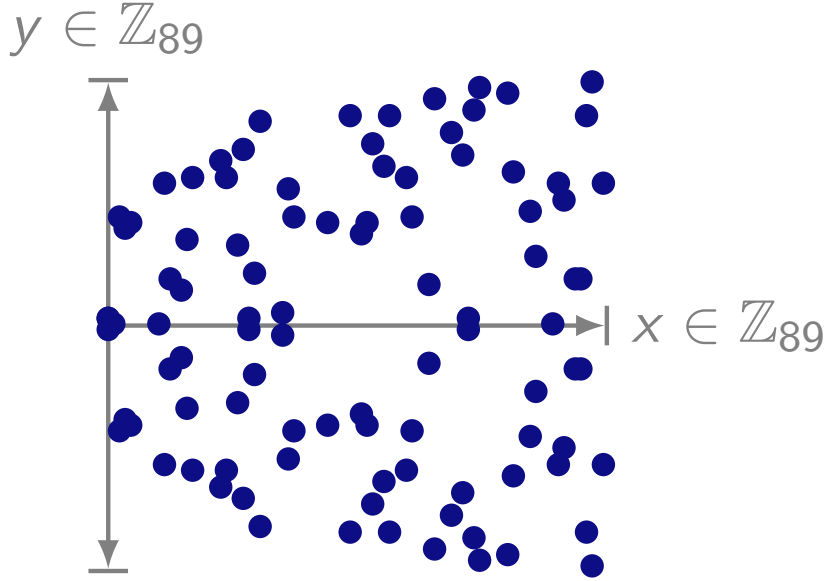
- Key generation :
  1. $x \leftarrow [0, r-1]$
  2. Private key is x
  3. Public key, g^x
- Signing : $h = H(m)$, $\sigma = h^x$
- Verification : $e(\sigma, g) = e(H(m), g^x)$

Outline

Introduction

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





$y^2 = x^3 - 2x + 1$ over $\mathbb{R}$ $y^2 = x^3 - 2x + 1$ over $\mathbb{Z}_{89}$

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

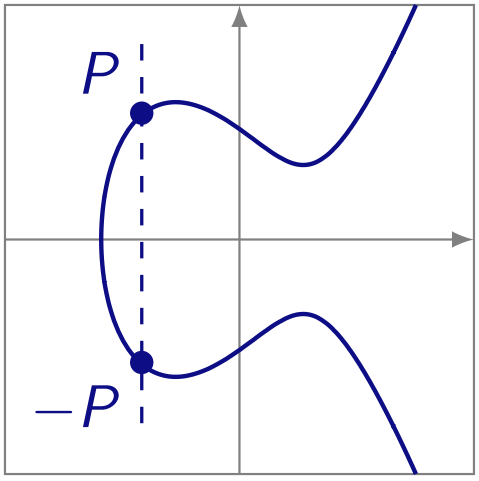
Weierstrass form if $\Delta = -16(4a^3 + 27b^2) \neq 0$ (if K is not of characteristic 2 or 3).

Laws

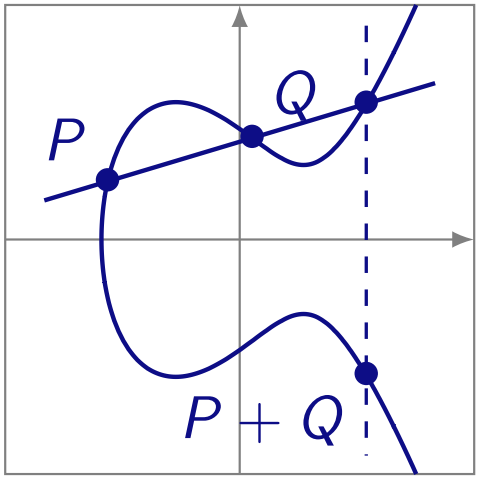
Theorem

- 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

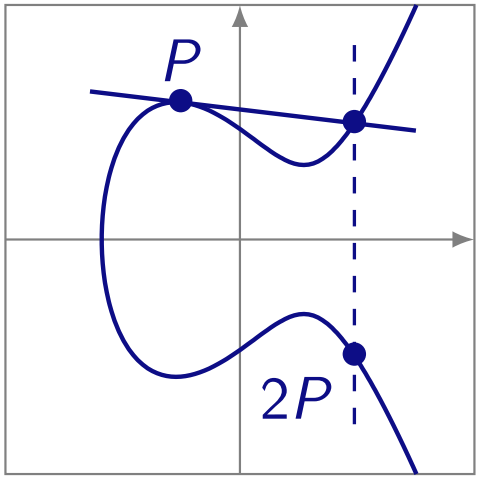
Laws



Inverse element $-P$



Addition $P + Q$
"Chord rule"



Doubling $P + P$
"Tangent rule"

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

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

"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 $k = d_A Q_B = d_A d_B G = d_B d_A G = d_B Q_A = k$.

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 $r = x_1 (\bmod n)$, where $(x_1, y_1) = kG$.
   If $r = 0$, go back to step 2.
4. Calculate $s = k^{-1}(e + rd_A)(\bmod n)$.
   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 $w = s^{-1}(\bmod n)$.
4. Calculate $u_1 = ew(\bmod n)$ and $u_2 = rw(\bmod n)$.
5. Calculate $(x_1, y_1) = u_1 G + u_2 Q_A$.
6. The signature is valid if $r = x_1(\bmod n)$, invalid otherwise.

ECDSA (Digital Signature Algorithm)

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

Hence
$k = s^{-1}(e + rd_A)(\text{ mod } n) = w(e + rd_A) = we + wrd_A = u_1 + u_2 d_A$
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 $r = x_1(\text{ mod } n)$ 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.

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

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

Limitations

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

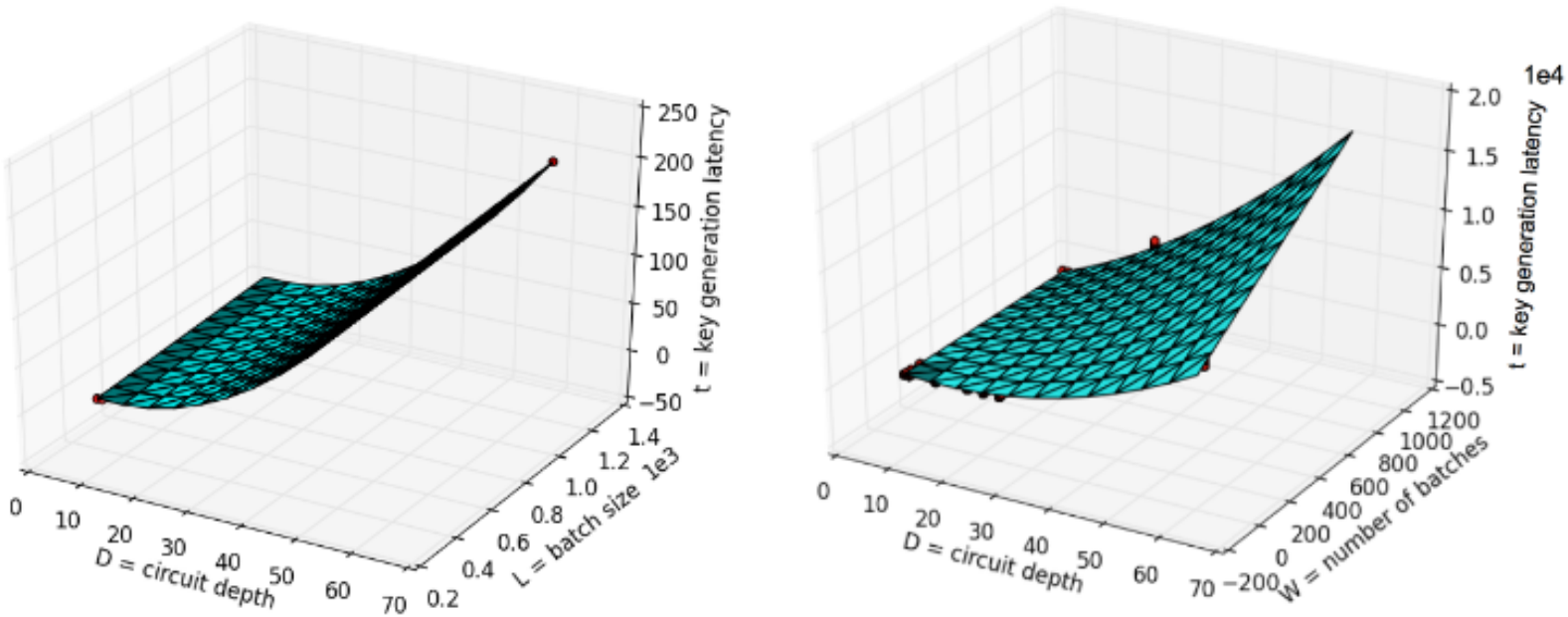


Fig. 9. Key generation time (left) and homomorphic evaluation time (right), in seconds

Outline

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 $H_2(g_{ID}^r)$, while for decryption, $H_2(e(d_{ID}, u))$ is applied.

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

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

Sakai-Kasahara: 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:

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

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:

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

Outline

Setup

Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data by Vipul Goyal, Omkant Pandey, Amit Sahai, Brent Waters, 2006

Pick random numbers $t_1 \dots t_n, y$, n is the number of attributes.

Public Key (PK) is $T_1 = g^{t_1}, \dots T_n = g^{t_n}$ and $Y = e(g, g)^y$

Master Key (MK) is $t_1 \dots t_n, y$.

Encryption

M a message, γ set of attributes, PK and a random s

$$E = (\gamma, MY^s, \{E_i = T_i^s\}_{i \in \gamma})$$

Key Generation

For each node x of the tree of γ pick a poynomial q_x of dergree $d_x = k_x - 1$ where k_x is the threshold value for x.
The root node r has $q_r(0) = y$ and for other
$q_x(0) = q_{parent(x)}(index(x))$

$$D_x = g^{\frac{q_x(0)}{t_i}}$$

where $i = att(x)$
$D = \{D_x\}_{x \in gamma}$

Decryption

With a Decryption key D and the root tree r
if $i \in \gamma$ we have :

$$DecryptNode(E, D, r) = e(D_x, E_i) = e(g^{\frac{q_x(0)}{t_i}}, g^{s \cdot t_i}) = e(g, g)^{s \cdot q_x(0)}$$

$$DecryptNode(E, D, r) = e(g, g)^y s = Y^s$$

with $E = (\gamma, MY^s, \{E_i = T_i^s\}_{i \in \gamma})$

Outline

Today

1. Motivation
2. Historical
3. Asymetric
4. Signature

Next Time

1. Modern Security Notions

Thank you for your attention.

Questions ?