Modern Cryptography

Phan Duong Hieu

Telecom Paris, IPP

New Technologies & Security Challenges

Technologies

- IoT, Big Data, Cloud Computing

$\rightarrow$ huge real-life applications

Main concerns

- Security, Privacy
- Trust on Authorities

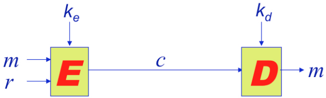
1. Multi-user Cryptography
2. Exploiting new technologies, without compromising privacy
3. Reducing the trust on Authorities $\rightarrow$ Decentralized Cryptography

Outline

1. Part 1 : Introduction to Modern Cryptography
2. Minicrypt: ZKP & Digital Signature
3. Cryptomania: Public-Key Encryption

Cryptography

- ***E*** – Encryption
- ***D*** – Decryption



- Symmetric Encryption: $k_e = k_d$
- Asymmetric Encryption: $k_e \neq k_d$

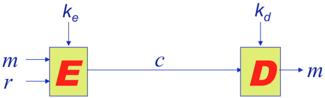
Public-key Encryption (Diffie-Helmann 1976)

- k_e could be published $\rightarrow$ encryption can be publicly computed.
- **RSA scheme**

$$(m^{e})^{(e^{-1} \mod \phi(N))} = m \mod N, \text{ where } N = pq$$

Cryptography

- ***E*** – Encryption
- ***D*** – Decryption



- Symmetric Encryption: $k_e = k_d$
- Asymmetric Encryption: $k_e \neq k_d$

Public-key Encryption (Diffie-Helmann 1976)

- k_e could be published $\rightarrow$ encryption can be publicly computed.
- **RSA scheme**

$$(m^e)^{(e^{-1} \mod \phi(N))} = m \mod N, \text{ where } N = pq$$

- **Elgamal scheme**

$$\frac{m(g^d)^r}{(g^r)^d} = m, \text{ where } g \text{ is a generator of a cyclic group}$$

Modern Cryptography

Beyond Encryption:

- Interactive proofs, zero-knowledge proofs, Identification
- Digital Signature
- Multi-party computation (for doing any cryptographic task imaginable!)

Main Theoretical Question (Complexity)

Does Cryptography really exist?

Centre question of Complexity: P vs. NP

- P: Problems for which solutions can be "efficiently" found
- NP: Problems for which solutions can be "efficiently" verified

Centre question of Complexity: P vs. NP

- P: Problems for which solutions can be "efficiently" found
- NP: Problems for which solutions can be "efficiently" verified

Efficiency

- Formal definition of algorithm (Turing machine)
- Church-Turing Thesis: everything that nature computes, can be emulated on a Turing machine
- Efficient algorithm: number of basic steps is bounded by a polynome on the size of the input

Centre question of Complexity: P vs. NP

- P: Problems for which solutions can be "efficiently" found
- NP: Problems for which solutions can be "efficiently" verified

Efficiency

- Formal definition of algorithm (Turing machine)
- Church-Turing Thesis: everything that nature computes, can be emulated on a Turing machine
- Efficient algorithm: number of basic steps is bounded by a polynome on the size of the input
- Example
  - P: multiplication, exponentiation modulo a prime number,...
  - NP: factorisation, discrete logarithm, 3-coloring problem, sodoku,...

Cryptography and the P vs. NP problem

(Trapdoor) one-way functions

A function $f : D \to R$ is a trapdoor function if it is

- Efficiently computable: $f(x)$ is efficiently computable for any $x \in D$
- Hard to invert: for a random $x \in D$, given $y = f(x)$, it is hard to find a $\bar{x}$ such that $y = f(\bar{x})$

Cryptography and the P vs. NP problem

(Trapdoor) one-way functions

A function $f : D \to R$ is a trapdoor function if it is

- Efficiently computable: $f(x)$ is efficiently computable for any $x \in D$
- Hard to invert: for a random $x \in D$, given $y = f(x)$, it is hard to find a $\bar{x}$ such that $y = f(\bar{x})$
- Trapdoor: given a trapdoor, it is easy to invert the function f.

Cryptography and the P vs. NP problem

(Trapdoor) one-way functions

A function $f : D \to R$ is a trapdoor function if it is

- Efficiently computable: $f(x)$ is efficiently computable for any $x \in D$
- Hard to invert: for a random $x \in D$, given $y = f(x)$, it is hard to find a $\bar{x}$ such that $y = f(\bar{x})$
- Trapdoor: given a trapdoor, it is easy to invert the function f.

Necessary conditions for the existence of cryptography

- One-way function for secret-key cryptography

Cryptography and the P vs. NP problem

(Trapdoor) one-way functions

A function $f : D \rightarrow R$ is a trapdoor function if it is

- Efficiently computable: $f(x)$ is efficiently computable for any $x \in D$
- Hard to invert: for a random $x \in D$, given $y = f(x)$, it is hard to find a $\bar{x}$ such that $y = f(\bar{x})$
- Trapdoor: given a trapdoor, it is easy to invert the function f.

Necessary conditions for the existence of cryptography

- One-way function for secret-key cryptography
- Trapdoor one-way function for public-key cryptography

Cryptography and the P vs. NP problem

(Trapdoor) one-way functions

A function $f : D \to R$ is a trapdoor function if it is

- Efficiently computable: $f(x)$ is efficiently computable for any $x \in D$
- Hard to invert: for a random $x \in D$, given $y = f(x)$, it is hard to find a $\bar{x}$ such that $y = f(\bar{x})$
- Trapdoor: given a trapdoor, it is easy to invert the function f.

Necessary conditions for the existence of cryptography

- One-way function for secret-key cryptography
- Trapdoor one-way function for public-key cryptography

The existence of one-way function implies P $\neq$ NP

5 Worlds in Impagliazzo's view

W1-Algorithmica: P = NP

One could use the method of verifying the solution to automatically solve the problem!

5 Worlds in Impagliazzo's view

W1-Algorithmica: P = NP

One could use the method of verifying the solution to automatically solve the problem!

W2-Heuristica: NP problems are hard in the worst case but easy on average.

There exist hard instances of NP problem, but to find such hard instances is itself a hard problem.

5 Worlds in Impagliazzo's view

W1-Algorithmica: P = NP

One could use the method of verifying the solution to automatically solve the problem!

W2-Heuristica: NP problems are hard in the worst case but easy on average.

There exist hard instances of NP problem, but to find such hard instances is itself a hard problem.

W3-Pessiland: NP problems hard on average but no one-way functions exist

It's easy to generate many hard instances of NP-problems, but no way to generate hard instances where we know the solution.

5 Worlds in Impagliazzo's view (cont.)

Minicrypt: One-way functions exist but public-key cryptography does not exist.

5 Worlds in Impagliazzo's view (cont.)

Minicrypt: One-way functions exist but public-key cryptography does not exist.

Cryptomania: Public-key cryptography is possible

It is possible for two parties to agree on a secret message using only public accessible channels

Outline

Minicrypt

Interactive proofs [Goldwasser, Micali, Rackoff 85]

"A proof is whatever convinces me" (Shimon Even)

Minicrypt

Interactive proofs [Goldwasser, Micali, Rackoff 85]

"A proof is whatever convinces me" (Shimon Even)

Zero-knowledge proofs, an example

Given g and $y = g^x$, I can convince you that I know x without revealing it

- I take a random r and send to you g^r
- You send me a random k
- I finally send back to you $t = r - kx$ that verifies $g^r = g^t y^k$

Minicrypt

Interactive proofs [Goldwasser, Micali, Rackoff 85]

"A proof is whatever convinces me" (Shimon Even)

Zero-knowledge proofs, an example

Given g and $y = g^x$, I can convince you that I know x without revealing it

- I take a random r and send to you g^r
- You send me a random k
- I finally send back to you $t = r - kx$ that verifies $g^r = g^t y^k$

Idea: representing g^r in the basis of $(g, y = g^x)$ requires the knowledge of x.

Why this is a ZK proof

(...on blackboard: extractor and simulator)

Minicrypt: Commitment

- Alice **commits** herself to some message m by giving Bob: $c = \text{Commit}(m, r)$, for a ramdom r.
- Bob should not learn anything about m given the commitment c.
- Alice can **open** the commitment by giving (m, r) to Bob to convince him that m was the value she committed herself to.

Formally:

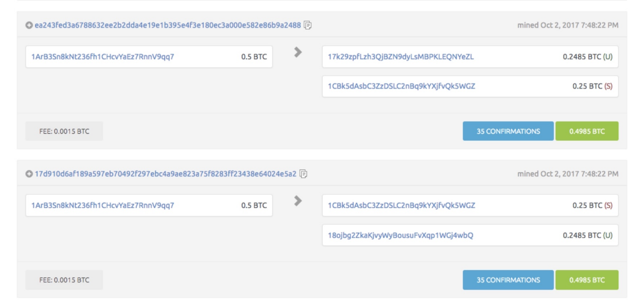
- **Hiding:** $\text{Commit}(m_0, U_n) \approx \text{Commit}(m_1, U_n)$ where U_n denotes the uniform distribution over $\{0, 1\}^n$.
- **Binding:** For all PPT adversaries A, we have

$$\Pr[\text{Commit}(m_0, r) = \text{Commit}(m_1, r') : (r, r') \leftarrow A(1^n)] = \text{negl}(n).$$

Application: ZKP for all NP problem

(...on blackboard)

A Bitcoin transaction



Privacy

- What is the problem with privacy in bitcoin?
- How we can use ZKP to solve this? $\rightarrow$ zkSNARKS.

Minicrypt: Digital Signatures (Idea)

If one-way functions exist, then every NP problem has a zero-knowledge proof. [Goldreich, Micali, Wigderson 91]

From zero-knowledge proof to digital signature (Schnorr scheme)

Given g and $y = g^x$, sign on the message m with the secret key x

- I take a random r and send to you g^r
- k is set to be $H(g^r, m)$ (H is modeled as a random oracle)
- I finally send to you the signature $(m, g^r, t = r - kx)$.
- Verification: checking whether $g^r = g^t y^{H(g^r, m)}$

Minicrypt: Digital Signatures

In Random Oracle Model

If one-way functions exist, then one can construct digital signature.

Minicrypt

- Zero-knowledge proofs, Identification, Digital Signature inspire from the notion of PKE.
- However, even if PKE dies one day, the above primitives would still be alive!

Digital Signatures: Formal treatment

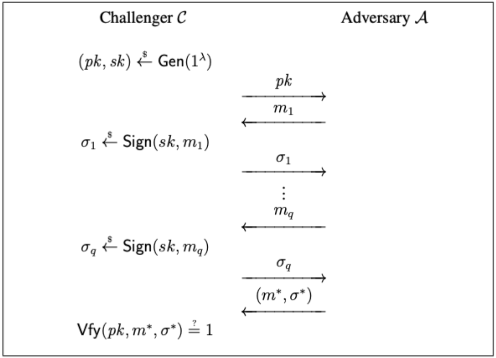
A signature scheme $S = (G, S, V)$

- $Gen(1^{\lambda}) \rightarrow (pk, sk)$ is a probabilistic algorithm that takes a security parameter λ and outputs a secret signing key *sk* and a public verification key *pk*.
- $Sign(sk, m) \rightarrow \sigma$ is a probabilistic algorithm that outputs a signature σ.
- $Vfy(pk, m, \sigma)$ outputs either accept (1) or reject (0).

We require that a signature generated by *S* is always accepted by *V*:

$$\Pr[V(pk, m, S(sk, m)) = accept] = 1$$

Digital Signatures: attack model (EUF-CMA)



Existential unforgeability under adaptive chosen message attacks

$$Adv(\mathcal{A}) = \Pr[Vfy(pk, m^\star, \sigma^\star) = 1]$$

The scheme is EUF-CMA secure si $\forall \mathcal{A}$, $Adv(\mathcal{A})$ is negligible.

Lamport's One-time Signatures from OWF f

- $Gen(1^\lambda) \to (pk, sk)$:

$$sk = \begin{pmatrix} x_{1,0} & x_{2,0} & \cdots & x_{\ell,0} \\ x_{1,1} & x_{2,1} & \cdots & x_{\ell,1} \end{pmatrix}$$

$$pk = \begin{pmatrix} y_{1,0} & y_{2,0} & \cdots & y_{\ell,0} \\ y_{1,1} & y_{2,1} & \cdots & y_{\ell,1} \end{pmatrix}$$

  where $x_{i,b} \in \{0,1\}^n, y_{i,b} = f(x_{i,b})$

- $Sign(sk, m = m_1 m_2 \ldots m_\ell \in \{0,1\}^\ell) \to \sigma$

$$\sigma = x_{1,m_1} x_{2,m_2} \ldots x_{\ell,m_\ell}$$

- $Vfy(pk, m, \sigma)$ check if $y_{i,m_i} = f(\sigma_i = x_{i,m_i}), \forall i = 1 \ldots \ell$

Theorem

If f is one-way, then the one-time signature is EUF-CMA.

Digital Signatures: from one-time to 2-times signatures

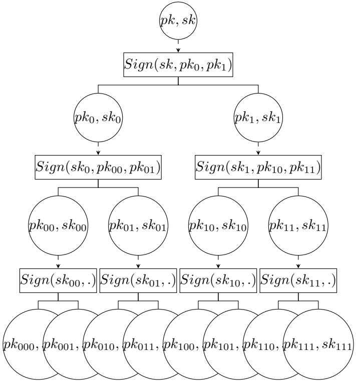
Exercices

1. Given 1-time signature, how can we construct a Stateful 2-time signature
2. Can we generalize the solution to a Stateful many-time signature? Estimate its efficiency.

$\rightarrow$ Stateful Chain-based Signature

Digital Signatures: from one-time to standard scheme

Hint: from this figure, describe the signature scheme.



Digital Signatures: Hash then Sign paradigm

Completer proof: OWF $\rightarrow$ Digital Signature

- Stateful *to* Stateless with PRF
- Sign on a long message $\rightarrow$ short message by using a hash function.

Exercices

Given:

- a collision resistant hash function $H : \{0,1\}^* \rightarrow H : \{0,1\}^n$
- a EUF-CMA singature on message of n bits

Construct another EUF-CMA singature that can sign on messages of abitrary size.

Signature Schemes in Practice

		Key exchange			Signatures		
	Hosts	**RSA**	**DH**	**ECDH**	**RSA**	**DSA**	**ECDSA**
HTTPS	39M	39%	10%	51%	99%	≈ 0	1%
SSH	17M	≈ 0	52%	48%	93%	7%	0.3%
IKEv1	1.1M	-	97%	3%	-	-	-
IKEv2	1.2M	-	98%	2%	-	-	-

FDH - RSA

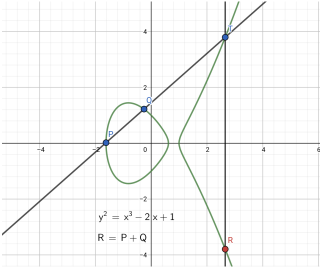
FDH - RSA

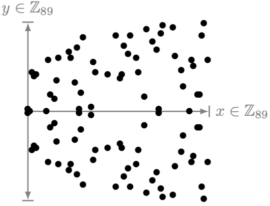
- $Gen(1^\lambda) \to (sk = d, pk = (N, e))$ as in RSA
- $Sign(sk, m) \to \sigma = H(m)^d$, where H is a **random oracle**.
- $Verfy(pk, m, \sigma)$ accept iff $\sigma^e = H(m)$

Security of FDH - RSA

If RSA problem is hard then FDH - RSA is EUF-CMA secure.
Proof: on blackboard.

Elliptic curve group





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

Elliptic curves on a field K (char(K) $\neq 2, 3$)

- Weierstrass equation: $y^2 = x^3 + ax + b$
- Points on a nonsingular elliptic curve (i.e., $4a^3 + 27b^2 \neq 0$) form a group under a special addition operation, with an additional point at infinity as the identity.

Elliptic Curve Cryptography

ElGamal encryption

- Setup: $G =< g >$ of order q.
- Secret key is a random $x \in \mathbb{Z}_q$, and public key is $y = g^x$
- Encryption $(c_1 = g^r, c_2 = y^r m)$
- Decryption $m = c_2/(c_1^x)$

Elliptic Curve ElGamal encryption

- The group can be chosen as $G =< g >$ where g is a point on an elliptic curve
- For the security, the group G should be big $\rightarrow$ the need of efficiency for points counting on elliptic curves (Schoof's algorithm)

Comparison

Symetric key size (bits)	Key size for RSA or Diffie-Hellman (bits)	Key size for Elliptic curve based shemes (bits)
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	521

Source: NIST Recommended Key Sizes

MOV attacks on ECM: the use of Pairings

Pairings on Elliptic curves

- E: a curve on a field $\mathbb{F}_q$
- $E[n] := \{P \in E(\bar{\mathbb{F}}_q) | nP = O\}$ (n-torsion subgroup in $E(\bar{\mathbb{F}}_q)$)
- Balasubramanian and Koblitz: $E[n] = E[n](\mathbb{F}_{q^k})$ for the smallest k such that $n|(q^k - 1)$ (k is called embedding degree).

Weil Pairings

$$e_n : E[n] \times E[n] \to \mathbb{F}_{q^k}$$

- Bilinear property: $e_n(aP, bQ) = e_n(P, Q)^{ab}$
- MOV attack: Reduce DL on Elliptic curve from DL on $\mathbb{F}_{q^k}$

Pairings in Cryptography

$$e : G \times G \rightarrow G_T$$

- bilinear map: $e(g^a, g^b) = e(g, g)^{ab}$
- non-degenerate map: $e(g^a, g^b) \neq 1$
- efficiently computable map: Miller's algorithm for (modified) Weil and Tate pairings.

Some problems are easy, some others are conjectured to be hard

- Decisional Diffie-Hellman problem on G is easy
- Computational Diffie-Hellman problem (given g^a, g^b, g^c, compute $e(g, g)^{abc}$) is conjectured to be hard

Pairings in Cryptography

Three-party key-exchange (Joux00)

- Secret keys of A, B, C are respectively a, b, c
- Public keys of A, B, C are respectively g^a, g^b, g^c
- Shared key $e(g, g)^{abc}$

Solution for Identity-based Encryption (Sakai-Ohgishi-Kasahara00, Boneh-Franklin01)

Will see in Advanced Primitives.

Aggregate Signature: BLS scheme

KeyGen:
- Let $e : G \times G \to G_T$ a pairing, where $G =< g >$.
- $H : \{0,1\}^\star \to G$ is a hash function, modelled as a random oracle.
- Randomly chooses s: the signing key $sk = s$, and the verification key $vk = g^s$;

$Sign(sk, m)$:

$$\sigma = H(m)^s$$

$Vfy(vk, \sigma, m)$: Checks

$$e(\sigma, g) = e(H(m), vk)$$

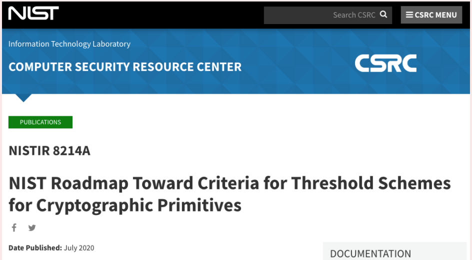
Exercice (Aggregate Technique)

How one can combine many signatures into just one signature?

Aggregate Signature in Practice

By 2020, BLS signatures were used in Ethereum blockchain.

Current Active Reseacrh Area: Multi-signer, Threshold signature (will see in Advanced Primitives)



Exercice: Collision Resistance from DL

Let $(\mathbb{G}, g, q) \leftarrow \mathsf{GroupGen}(1^n)$ be a group generation algorithm that generates a cyclic group $\mathbb{G} = \langle g \rangle$ with generator g of order $|\mathbb{G}| = q$ where q is a prime.

1. A hash function mapping $\mathbb{Z}_q^2 \to \mathbb{G}$: $H_{g,h}(x_1, x_2) = g^{x_1} h^{x_2}$.
2. A more compressing function that maps $\mathbb{Z}_q^m \to \mathbb{G}$:

$$H_{g_1,g_2,\ldots,g_m}(x_1, \ldots, x_m) = \prod_{i=1}^{m} g_i^{x_i}$$

   where $h, g_1 \ldots, g_m$ are random group elements.

Show that, under the DL assumption, the above functions are CR hash function. (Hint: given a discrete log challenge $g, h = g^x$ where your goal is to find x, define $g_i = g^{a_i} h^{b_i}$ for random $a_i, b_i \leftarrow \mathbb{Z}_q$.)

Secret Sharing → Threshold BLS Signature

Secret Sharing

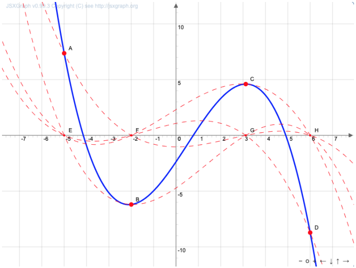
Dealer:

- On input a secret s, choose a polynomial P of degree d such that $P(0) = s$.
- Give to each user i a random point $(x_i, P(x_i))$

Goal:

- any $t = d + 1$ users can do a joint computation to get s
- any $k \leq d$ users get no information abour s.

Secret Sharing → Threshold BLS Signature



Simulation source: `https://inst.eecs.berkeley.edu/~cs70/sp15/hw/vlab7.html`

Tool: Lagrange Polynomial Interpolation

- Given a set of $t = d + 1$ points $(x_0, y_0), \ldots, (x_j, y_j), \ldots, (x_t, y_t)$
- The interpolation polynomial is a linear combination $L(x) := \sum_{j=0}^{k} y_j \ell_j(x)$ of Lagrange basis polynomials

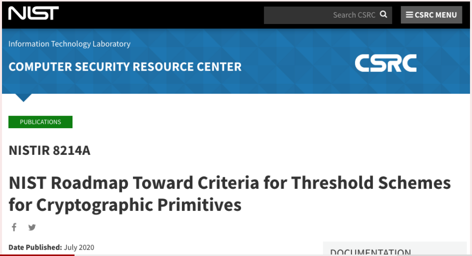
$$\ell_j(x) := \prod_{\substack{0 \le m \le k \\ m \ne j}} \frac{x - x_m}{x_j - x_m} = \frac{(x - x_0)}{(x_j - x_0)} \cdots \frac{(x - x_{j-1})}{(x_j - x_{j-1})} \frac{(x - x_{j+1})}{(x_j - x_{j+1})} \cdots \frac{(x - x_k)}{(x_j - x_k)}$$

Threshold BLS Signature

Exercice: Given a secret sharing scheme, propose a Threshold BLS Signature:

- Each signer receives from the Authority a secret key.
- Each signer signs the message m on its own.
- Any t signers can jointly produce a BLS signatures (Tool: Interpolation on exponents)
- No group of less than t signers can produce a valid BLS signature.

Threshold Cryptography (will see in Advanced Primitives)



Outline

Provable security: sufficient conditions for security

What we discussed

If factorization or DL problems are easy, then we can attack crypto systems that based on these problems

Question

Suppose that factorization and DL problems are hard. Could we prove the security for proposed crypto systems?

One wayness is enough?

$$E'(m1||m_2) := E(m_1)||m_2$$

- If E is one-way, then E' is also one-way
- But the security of E' is clearly not enough: at least half the message leaks!

In many situation, one bit (attack or not) is important...

Semantic security [Goldwasser-Micali '82]

Perfect Security vs. Semantic security

- Perfect security: the distribution of the ciphertext is perfectly independent of the plaintext
- Semantic security (computational version of perfect security): the distribution of the ciphertext is computationally independent of the plaintext

Semantic security [Goldwasser-Micali '82]

Perfect Security vs. Semantic security

- Perfect security: the distribution of the ciphertext is perfectly independent of the plaintext
- Semantic security (computational version of perfect security): the distribution of the ciphertext is computationally independent of the plaintext

Semantic Security

- Semantic Security is equivalent to the notion of Indistinguishability (IND): No adversary (modeled by a poly-time Turing machine) can distinguish a ciphertext of m_0 from a ciphertext of m_1.
- For public-key encryption: Probabilistic encryption is required!
- For secret-key encryption: deterministic encryption could be semantically secure [Phan-Pointcheval '04]

Semantic security is enough?

ElGamal Encryption

- Elgamal encryption can be proven to be IND, based on Decisional Diffie-Hellman assumption (given g^a, g^b, it is hard to distinguish between g^{ab} and a random element g^z).
- Elgamal encryption is homomorphic: $E(m_1 m_2) = E(m_1)E(m_2)$

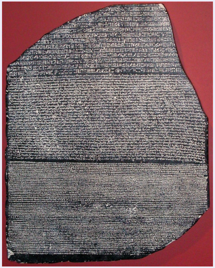
Private Auctions

The bids are encrypted. The authority then opens all the encrypted bids and the highest bid wins

- IND guarantees privacy of the bids
- Malleability: from $c = E(pk, b)$, without knowing b, one can generate $c' = E(pk, 2b)$: an unknown higher bid!
- Should consider adversaries with some more information.

Adversaries with additional information

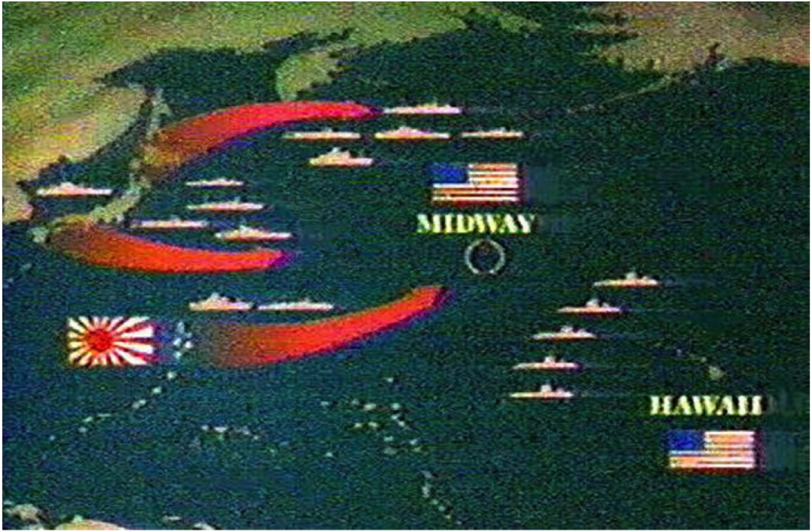
Rosetta Stone: A key element to decode Ancient Egyptian hieroglyphs



Chosen plaintext attacks (CPA)

The adversary can have access to encryption oracle (this only makes sense for symmetric encryption)

Interactive Adversaries: CCA attacks



Chosen plaintext and chosen ciphertext attacks

IND-CCA Security

- IND-CCA also implies non-malleability (NM-CCA)
- This is the standard notion for public-key encryption

Major problem in cryptography

Construction of IND-CCA encryption schemes.

Security of RSA & ElGamal PKE

Recall:

- k_e could be published $\rightarrow$ encryption can be publicly computed.
- **RSA scheme**

$$(m^{e})^{(e^{-1} \mod \phi(N))} = m \mod N, \text{ where } N = pq$$

Security of RSA & ElGamal PKE

Recall:

- k_e could be published $\rightarrow$ encryption can be publicly computed.
- **RSA scheme**

$$(m^{e})^{(e^{-1} \mod \phi(N))} = m \mod N, \text{ where } N = pq$$

- **ElGamal scheme**

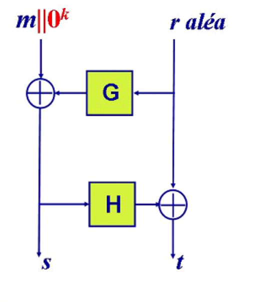
$$\frac{m(g^{d})^{r}}{(g^{r})^{d}} = m, \text{ where } g \text{ is a generator of a cyclic group}$$

Exercices

- Is RSA IND-CPA?
- Is ElGamal IND-CCA?

OAEP (Bellare-Rogaway94)

Random oracle model



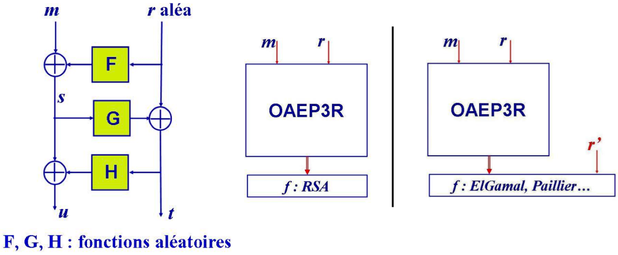
- It is believed that f-OAEP is IND-CCA for any trapdoor one-way permutation.
- In 2000, Shoup presented an attack on a very special trapdoor one-way permutation.

RSA-OAEP

RSA-OAEP is proven IND-CCA secure [Fujisaki-Okamoto-Pointcheval-Stern01]

- If f is partially one-way, then f-OAEP is secure
- RSA is partially one-way

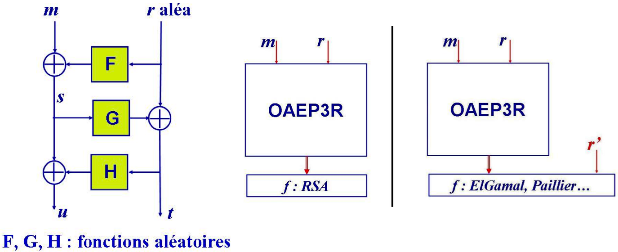
3-round OAEP (among others varieties of OAEP)



Advantages

- *f* does not need to be partially one-way
- *f* could also be one-way function (such as Elgamal, Paillier encryptions...)

3-round OAEP (among others varieties of OAEP)



Advantages

- f does not need to be partially one-way
- f could also be one-way function (such as Elgamal, Paillier encryptions...)

Current state

Many solutions in the standard model (without random oracle) but the practical implementations mostly rely on RSA-OAEP.

Proof of IND-CPA of ElGamal scheme, under DDH assumption

Let $\mathbb{G} = \langle g \rangle$ with generator g of order $|\mathbb{G}| = q$ where q is a prime.
Public key $pk = (g, h = g^x)$ and secret key $sk = x$.
Encryption:Enc$(pk, m) = (g^r, h^r \cdot m)$ where $r \leftarrow \mathbb{Z}_q$.

- **Game 0:** Real IND-CPA game, challenge ciphertext is $(g^r, h^r \cdot m_b)$
- **Game 1:** Replace (g, h, g^r, h^r) by $(g, h, g^r, h^{r'})$, for random r, r' The adversary cannot distinguish Game 0 and Game 1, otherwise we can solve DDH
- In Game 1: the adversary has no information about m_b.

Security Proofs: IND-CCA

Idea: Embed a ZK proof of knowedge in the ciphertext.

- Let $\mathbb{G} = \langle g \rangle$ with generator g of order $|\mathbb{G}| = q$ where q is a prime.
- Verifier chooses $\alpha, x_1, x_2 \leftarrow \mathbb{Z}_q$ and sets $g_1 = g, g_2 = g^\alpha, c = g_1^{x_1} g_2^{x_2}$ and sends g_1, g_2, c to prover.
- Prover chooses $r \leftarrow \mathbb{Z}_q$, sets $u_1 = g_1^r, u_2 = g_2^r$ and $v = c^r$
- Verifier checks whether $v = u_1^{x_1} u_2^{x_2}$.

Proof of IND-CCA1 of Cramer-Shoup Lite scheme

Public key $pk = (c = g_1^{x_1} g_2^{x_2}, h = g_1^z)$ and secret key $sk = (x_1, x_2, z)$.
Encryption: $\mathrm{Enc}(pk, m) = (u_1 = g_1^r, u_2 = g_2^r, e = h^r \cdot m, v = c^r)$ where $r \leftarrow \mathbb{Z}_q$.
Decryption: Check if $v = u_1^{x_1} u_2^{x_2}$, return $\frac{e}{u_1^z}$, otherwise return $\perp$
Proof: on blackboard, with sequences of games

Exercice: Homomorphism of ElGamal encryption

Let $\mathbb{G} = \langle g \rangle$ with generator g of order $|\mathbb{G}| = q$ where q is a prime.
Public key $pk = (g, h = g^x)$ and secret key $sk = x$.
Encryption:Enc$(pk, m) = (g^r, h^r \cdot m)$ where $r \leftarrow \mathbb{Z}_q$.

- Given a public key pk and an ciphertext c, show how to create a ciphertext c' which encrypts the same message under pk but with independent randomness.
- Given a public key pk and any two independently generated ciphertexts c_1, c_2 encrypting some unknown messages $m_1, m_2 \in \mathbb{G}$ under pk, create a new ciphertext c^* encrypting $m^* = m_1 \cdot m_2$ under pk without needing to know sk, m_1, m_2.

Application: Voting system.

Exercice: Broadcast attack on RSA

- For efficiency, the public key in RSA is often set to be $e = 3$.
- Suppose that three users have public keys $(N_1, 3)$, $(N_2, 3)$,$(N_3, 3)$.
- A center broadcasts a message m to these three people by using RSA aencryption and produces three ciphertexts c_1, c_2, c_3.

Can an adversary, by observing c_1, c_2, c_3, extract information about m?

Identity-based Encryption

Public key Encryption

- each user generates a couple of public-key/secret-key
- public-key is associated to the identity of the user via a certification $\rightarrow$ complicated public key infrastructure (PKI)

Identity-based Encryption

Shamir 1984 introduced the idea of using the identity of the user as the public-key $\rightarrow$ avoid the PKI.

- extract the secret-key from the public-key
- the extraction is done by an authority, from a trapdoor (master secret key)

Only at the begining of 2000, the first constructions of IBE were introduced.

PKE vs. IBE?

1. CCA PKE from CPA IBE [Boneh-Canetti-Halevi-Katz 2006]
2. No black-box construction of IBE from CCA-PKE [Dan Boneh-Papakonstantinou-Rackoff-Vahlis-Waters 2008]

Why is it difficult to construct an IBE?

1. Design:
   - In a PKE, one often generates a public key from a secret key. Well-formed public keys might be exponentially sparse.
   - In an IBE scheme:
     - any identity should be publicly mapped to a public key
     - extract secret key from public-key via a trapdoor.

Why is it difficult to construct an IBE?

1. Design:
   - In a PKE, one often generates a public key from a secret key. Well-formed public keys might be exponentially sparse.
   - In an IBE scheme:
     - any identity should be publicly mapped to a public key
     - extract secret key from public-key via a trapdoor.
2. Security: in IBE, the adversary can corrupt secret keys $\rightarrow$ the simulator should be able to simulate all key queries except the challenge identity.

Brief History of IBE

First idea by Shamir in 84.
There are five families of IBE schemes from:

- elliptic curves pairing: Sakai Ohgishi Kasahara in 2000, Boneh Franklin in 2001.
- quadratic residues: Cocks in 2001.
- lattice: Gentry Peikert Vaikuntanathan in 2008.
- computational Diffie-Hellman: Dottling-Garg in 2017.
- coding: Gabotit-Hauteville-Phan-Tillich in 2017

Brief History of IBE

First idea by Shamir in 84.
There are five families of IBE schemes from:

- elliptic curves pairing: Sakai Ohgishi Kasahara in 2000, Boneh Franklin in 2001.
- quadratic residues: Cocks in 2001.
- lattice: Gentry Peikert Vaikuntanathan in 2008.
- computational Diffie-Hellman: Dottling-Garg in 2017.
- coding: Gabotit-Hauteville-Phan-Tillich in 2017

Elgamal Encryption → IBE?

- $G =< g >$ of order q
- Secret key: $s \leftarrow \mathbb{Z}_q$
- Public key: $y = g^s$
- Ciphertext: $(g^r, y^r m)$, where $r \leftarrow \mathbb{Z}_q$
- Decryption: from s, compute $y^r = (g^r)^s$ and recover m

Transform to IBE:

1. Public key: define $y = H(id) = g^s \rightarrow$ can we extract s?
2. Possible in bilinear groups → Boneh-Franklin scheme

Elgamal Encryption $\rightarrow$ IBE? (with Pairings)

ElGamal:

- Secret key: random s
- Public key: $y = g^s$
- Ciphertext: $(g^r, y^r m)$, for a random r
- Decryption: from s, compute $y^r = (g^r)^s$ and recover m

Boneh-Franklin IBE [2001]

$$y_{id} = e(g, H(id))^s = e(g, H(id)^s) = e(g^s, H(id))$$

Elgamal Encryption $\rightarrow$ IBE? (with Pairings)

ElGamal:

- Secret key: random s
- Public key: $y = g^s$
- Ciphertext: $(g^r, y^r m)$, for a random r
- Decryption: from s, compute $y^r = (g^r)^s$ and recover m

Boneh-Franklin IBE [2001]

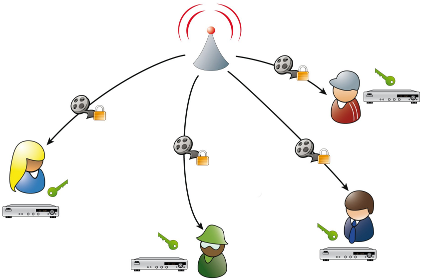
$$y_{id} = e(g, H(id))^s = e(g, H(id)^s) = e(g^s, H(id))$$

Considering s as trapdoor (master secret key), g^s as a public then:

- "Public key" $y_{id} = e(g^s, H(id)$ is computable from id
- Secret key can be extracted as $sk_{id} = H(id)^s$.
- Ciphertext: $(g^r, y_{id}^r m)$
- Decryption: from $H(id)^s$, compute $y_{id}^r = e(g^r, H(id)^s)$ and recover m

Multi-receiver Encryption

From “One-to-one” to ‘one-to-many” communications

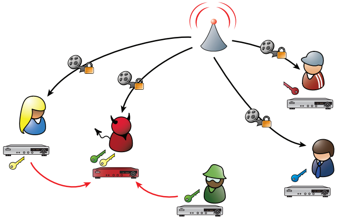


Provide all users with the same key $\rightarrow$ problems:

1. Impossibility to identify the source of the key leakage (traitor)
2. Impossibility to revoke a user, except by resetting the parameters

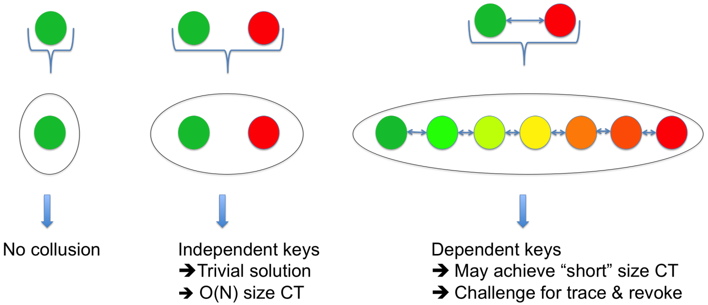
Broadcast Encryption

Revocation [Berkovist91, Fiat-Naor94] & Traitor Tracing [Chor-Fiat-Naor94]



1. Tracing traitors
    - From a pirate key $\rightarrow$ White-box tracing
    - From a pirate decoder (i.e., the pirate can obfuscate its own decryption algorithm and key)
        - ★ Black-box confirmation: tracer has a suspect list
        - ★ Black-box tracing: without any assumption
2. Revoke scheme: encrypt to all but revoked users

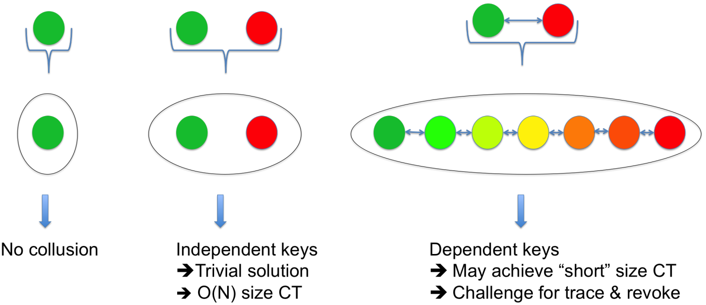
Pirate



Collusion of users → Pirate

The users' keys are not independent
→ A pirate (from only 2 keys) can produce many pirate keys

Pirate

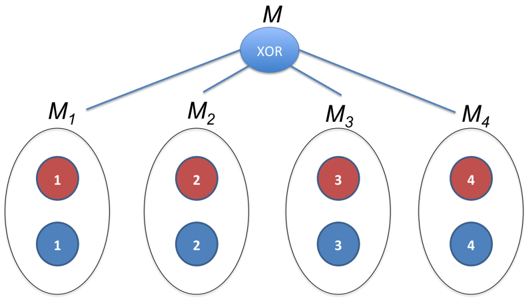


Collusion of users → Pirate

The users’ keys are not independent
→ A pirate (from only 2 keys) can produce many pirate keys
→ Tracing and revocation are non trivial, even for small collusions

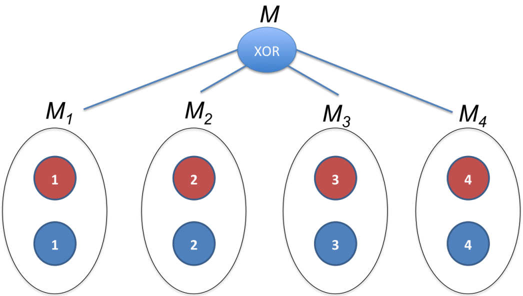
Example: Combinatorial Scheme

Combination of 2-user schemes → multi-user scheme [Boneh-Shaw95]

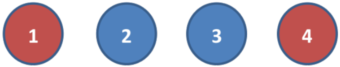


Example: Combinatorial Scheme

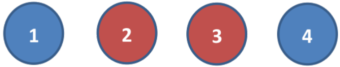
Combination of 2-user schemes → multi-user scheme [Boneh-Shaw95]



A key : one ball for each number



Collusion of 2 users could generate the whole set of the keys



Collusion secure Codes

Traitor 1	1	0	1	0	1	1	0	1	0	1	1	1	0	0	1	···	0	0	1	0	0
Traitor 2	1	0	1	0	1	0	1	1	0	1	1	0	0	0	1	···	0	0	1	0	1
Traitor 3	1	0	1	0	1	0	1	1	0	1	1	0	0	0	1	···	1	0	1	0	0

Collusion secure Codes

Traitor 1	1	0	1	0	1	1	0	1	0	1	1	1	0	0	1	···	0	0	1	0	0
Traitor 2	1	0	1	0	1	0	1	1	0	1	1	0	0	0	1	···	0	0	1	0	1
Traitor 3	1	0	1	0	1	0	1	1	0	1	1	0	0	0	1	···	1	0	1	0	0
Pirate	1	0	1	0	1			1	0	1	1		0	0	1	···		0	1	0	

Marking Assumption

At positions where all the traitors get the same bit, the pirate codeword must retain that bit

From Collusion Secure Codes to Traitor Tracing

KGen :

Table 0	$k_{0,1}$	$k_{0,2}$	$k_{0,3}$	$k_{0,4}$	$k_{0,5}$	...	$k_{0,\ell}$
Table 1	$k_{1,1}$	$k_{1,2}$	$k_{1,3}$	$k_{1,4}$	$k_{1,5}$	...	$k_{1,\ell}$

From Collusion Secure Codes to Traitor Tracing

KGen :

Table 0	$k_{0,1}$	$k_{0,2}$	$k_{0,3}$	$k_{0,4}$	$k_{0,5}$	...	$k_{0,\ell}$
Table 1	$k_{1,1}$	$k_{1,2}$	$k_{1,3}$	$k_{1,4}$	$k_{1,5}$	...	$k_{1,\ell}$
Codeword i	1	1	0	1	0	...	1

From Collusion Secure Codes to Traitor Tracing

KGen :

Table 0	$k_{0,1}$	$k_{0,2}$	$k_{0,3}$	$k_{0,4}$	$k_{0,5}$	...	$k_{0,\ell}$
Table 1	$k_{1,1}$	$k_{1,2}$	$k_{1,3}$	$k_{1,4}$	$k_{1,5}$	...	$k_{1,\ell}$

Codeword i	1	1	0	1	0	...	1
user i	$k_{1,1}$	$k_{1,2}$	$k_{0,3}$	$k_{1,4}$	$k_{0,5}$	...	$k_{1,\ell}$

From Collusion Secure Codes to Traitor Tracing

KGen :

Table 0	$k_{0,1}$	$k_{0,2}$	$k_{0,3}$	$k_{0,4}$	$k_{0,5}$	...	$k_{0,\ell}$
Table 1	$k_{1,1}$	$k_{1,2}$	$k_{1,3}$	$k_{1,4}$	$k_{1,5}$	...	$k_{1,\ell}$
Codeword i	1	1	0	1	0	...	1
user i	$k_{1,1}$	$k_{1,2}$	$k_{0,3}$	$k_{1,4}$	$k_{0,5}$	...	$k_{1,\ell}$

Enc :

Message	m_1	m_2	m_3	m_4	m_5	...	m_ℓ

From Collusion Secure Codes to Traitor Tracing

KGen :

Table 0	$k_{0,1}$	$k_{0,2}$	$k_{0,3}$	$k_{0,4}$	$k_{0,5}$	...	$k_{0,\ell}$
Table 1	$k_{1,1}$	$k_{1,2}$	$k_{1,3}$	$k_{1,4}$	$k_{1,5}$	...	$k_{1,\ell}$
Codeword i	1	1	0	1	0	...	1
user i	$k_{1,1}$	$k_{1,2}$	$k_{0,3}$	$k_{1,4}$	$k_{0,5}$	...	$k_{1,\ell}$

Enc :

Message	m_1	m_2	m_3	m_4	m_5	...	m_ℓ
Ciphertext	$c_{0,1}$	$c_{0,2}$	$c_{0,3}$	$c_{0,4}$	$c_{0,5}$	...	$c_{0,\ell}$
	$c_{1,1}$	$c_{1,2}$	$c_{1,3}$	$c_{1,4}$	$c_{1,5}$	...	$c_{1,\ell}$

From Collusion Secure Codes to Traitor Tracing

KGen :

Table 0	$k_{0,1}$	$k_{0,2}$	$k_{0,3}$	$k_{0,4}$	$k_{0,5}$	...	$k_{0,\ell}$
Table 1	$k_{1,1}$	$k_{1,2}$	$k_{1,3}$	$k_{1,4}$	$k_{1,5}$	...	$k_{1,\ell}$

Codeword i	1	1	0	1	0	...	1
user i	$k_{1,1}$	$k_{1,2}$	$k_{0,3}$	$k_{1,4}$	$k_{0,5}$	...	$k_{1,\ell}$

Enc :

Message	m_1	m_2	m_3	m_4	m_5	...	m_ℓ
Ciphertext	$c_{0,1}$	$c_{0,2}$	$c_{0,3}$	$c_{0,4}$	$c_{0,5}$	...	$c_{0,\ell}$
	$c_{1,1}$	$c_{1,2}$	$c_{1,3}$	$c_{1,4}$	$c_{1,5}$	...	$c_{1,\ell}$

Tracing Traitors

- At each position j, send $c_{0,j}$ and $c_{1,j}$ corresponding to two different messages m_j and $m'_j \rightarrow v_j \rightarrow$ a pirate codeword v
- From tracing algorithm of Secure Code, identify traitors

Exclusive Set System (ESS)

[ALO98]

$\mathcal{F}$ is an (N, ℓ, r, s)-ESS if:

- $\mathcal{F}$: a family of ℓ subsets of $[N]$
- For any $R \subseteq [N]$ of size at most r, there exists $S_1, \ldots S_s \in \mathcal{F}$ s.t.

$$[N] - R = \bigcup_{i=1}^{s} S_i$$

Exclusive Set System (ESS)

[ALO98]

$\mathcal{F}$ is an (N, ℓ, r, s)-ESS if:

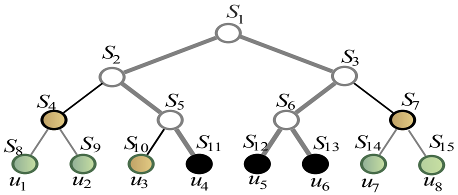
- $\mathcal{F}$: a family of ℓ subsets of $[N]$
- For any $R \subseteq [N]$ of size at most r, there exists $S_1, \dots S_s \in \mathcal{F}$ s.t.

$$[N] - R = \bigcup_{i=1}^{s} S_i$$

From ESS to Revoke System

- Each $S_i \in \mathcal{F}$ is associated to a key K_i
- User u receives all keys K_i that $u \in S_i$
- To revoke a set $R \subseteq [N]$ of size at most r:
  - Find $S_1, \dots S_s \in \mathcal{F}$ s.t. $[N] - R = \bigcup_{i=1}^{s} S_i$
  - Encrypt the message with each key K_i

NNL Schemes viewed as Exclusive Set Systems
[NNL01]



- $\mathcal{F} = \{S_1, S_2, \ldots, S_{15}\}$
- S_i contains all users (*i.e.* leaves) in the subtree of node i (e.g. $S_2 = \{u_1, u_2, u_3, u_4\}$)
- Revoked set $R = \{u_4, u_5, u_6\}$
- Encrypt with keys at S_4, S_7, S_{10}
- Complete-subtree is a $(N, 2N-1, r, r\log(N/r))$-ESS
- **Decentralized scheme** [Phan-Pointcheval-Strefler '12]

ElGamal Encryption Scheme

- $G = < g >$ of order q
- Secret key: $\alpha \leftarrow \mathbb{Z}_q$
- Public key: $y = g^\alpha$
- Ciphertext: $(g^r, y^r m)$, where $r \leftarrow \mathbb{Z}_q$
- Decryption: from α, compute $y^r = (g^r)^\alpha$ and recover m

Algebraic Schemes

Dependence between the keys: sharing some algebraic properties

ElGamal Encryption Scheme

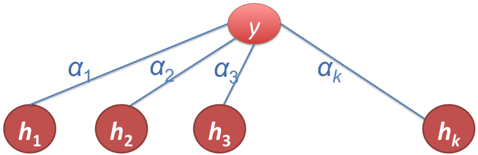
- $G =< g >$ of order q
- Secret key: $\alpha \leftarrow \mathbb{Z}_q$
- Public key: $y = g^\alpha$
- Ciphertext: $(g^r, y^r m)$, where $r \leftarrow \mathbb{Z}_q$
- Decryption: from α, compute $y^r = (g^r)^\alpha$ and recover m

Multi-receiver Encryption

Main problem: how to extend the same y to support many users?

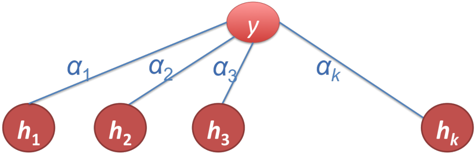
Algebraic Schemes

Dependent keys: sharing some algebraic properties [Boneh-Franklin99]



Algebraic Schemes

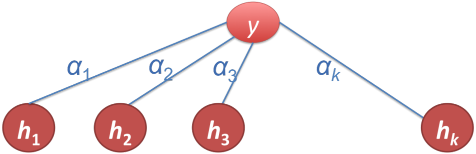
Dependent keys: sharing some algebraic properties [Boneh-Franklin99]



- $G =< g >$ of order q; Public key: $(y, h_1, \ldots, h_k) \in G^{k+1}$
- User key: a representation $(\alpha_1, \ldots, \alpha_k)$ of y in the basis $(h_1, \ldots, h_k)$: $(y = h_1^{\alpha_1} \ldots h_k^{\alpha_k})$

Algebraic Schemes

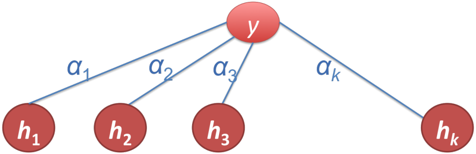
Dependent keys: sharing some algebraic properties [Boneh-Franklin99]



- $G =< g >$ of order q; Public key: $(y, h_1, \ldots, h_k) \in G^{k+1}$
- User key: a representation $(\alpha_1, \ldots, \alpha_k)$ of y in the basis $(h_1, \ldots, h_k)$: $(y = h_1^{\alpha_1} \ldots h_k^{\alpha_k})$
- Ciphertext: $(y^r m, h_1^r, \ldots, h_k^r)$, where $r \leftarrow \mathbb{Z}_q$
- Each user can compute y^r from $(h_1^r, \ldots, h_k^r)$ and recover m

Algebraic Schemes

Dependent keys: sharing some algebraic properties [Boneh-Franklin99]



- $G =< g >$ of order q; Public key: $(y, h_1, \ldots, h_k) \in G^{k+1}$
- User key: a representation $(\alpha_1, \ldots, \alpha_k)$ of y in the basis $(h_1, \ldots, h_k)$: $(y = h_1^{\alpha_1} \ldots h_k^{\alpha_k})$
- Ciphertext: $(y^r m, h_1^r, \ldots, h_k^r)$, where $r \leftarrow \mathbb{Z}_q$
- Each user can compute y^r from $(h_1^r, \ldots, h_k^r)$ and recover m

Collusion of 2 users

convex combination $\rightarrow$ q new pirate keys

From Encryption to Multi-receiver Encryption

ElGamal Encryption Scheme

- $G =< g >$ of order q
- Secret key: $\alpha \leftarrow \mathbb{Z}_q$
- Public key: $y = g^\alpha$
- Ciphertext: $(g^r, y^r m)$, where $r \leftarrow \mathbb{Z}_q$
- Decryption: from α, compute $y^r = (g^r)^\alpha$ and recover m

Boneh-Franklin Multi-receiver Encryption

- Each user receive a representation $(\alpha_1, \ldots, \alpha_k)$ of y in a public basis $(h_1, \ldots, h_k)$: $(y = h_1^{\alpha_1} \ldots h_k^{\alpha_k})$
- Each user can compute y^r from $(h_1^r, \ldots, h_k^r)$
- Public key: $(y, h_1, \ldots, h_k)$
- Ciphertext: $(y^r m, h_1^r, \ldots, h_k^r)$

Boneh-Franklin Scheme

Boneh-Franklin Traitor Tracing

- Transformation from Elgamal Encryption to Traitor Tracing: linear loss in the number of traitors
- Achieve black-box confirmation

Boneh-Franklin Scheme

Boneh-Franklin Traitor Tracing

- Transformation from Elgamal Encryption to Traitor Tracing: linear loss in the number of traitors
- Achieve black-box confirmation

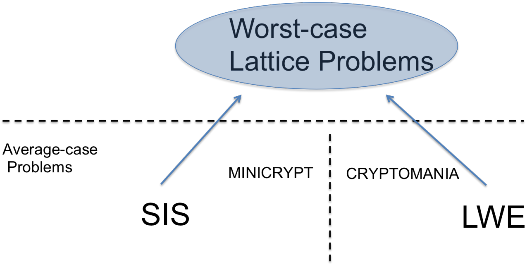
Our Work [Ling-Phan-Stehlé-Steinfeld, Crypto14]

- Study a variant of the Learning With Errors problem [Regev 05], namely k-LWE
- Get a more efficient transformation:

$$\text{LWE-based Encryption} \approx \text{LWE traitor tracing}$$

- Achieve black-box confirmation as in Boneh-Franklin scheme
- Resist quantum attacks

Short Integer Solution [Ajtai96] and Learning With Errors [Regev05] problems

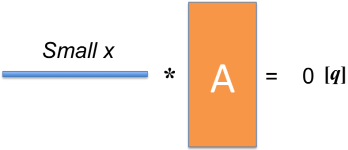


Post-quantum cryptography

- Lattice: (SIS and LWE) give solutions for almost all primitives
- Coding: give solutions for PKE, recently for Identity-based Encryption [Gaborit, Hauteville, Phan, Tillich, Crypto 2017]; still open for broadcast encryption, traitor tracing.
- Other tools: multi-variable, isogeny...

Short Integer Solution [Ajtai96] and Learning With Errors [Regev05] problems

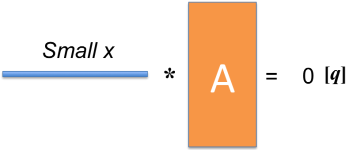
- Params: $m, n, q \geq 0$, $A \hookleftarrow U(\mathbb{Z}_q^{m \times n})$



SIS
Find small $\mathbf{x} \in \mathbb{Z}^m \setminus \mathbf{0}$
s.t. $\mathbf{x}^t A = \mathbf{0}\ [q]$

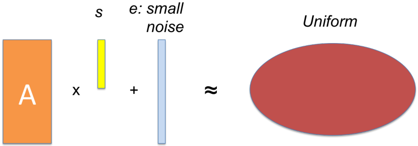
Short Integer Solution [Ajtai96] and Learning With Errors [Regev05] problems

- Params: $m, n, q \geq 0$, $A \hookleftarrow U(\mathbb{Z}_q^{m \times n})$



SIS

Find small $\mathbf{x} \in \mathbb{Z}^m \setminus \mathbf{0}$ s.t. $\mathbf{x}^t A = \mathbf{0}$ $[q]$

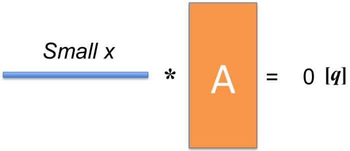


LWE

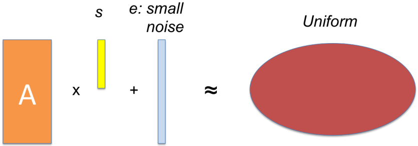
Dist. $A\mathbf{s} + \mathbf{e}$ $[q]$ and $U(\mathbb{Z}_q^m)$, for $\mathbf{s} \hookleftarrow U(\mathbb{Z}_q^n)$, noise $\mathbf{e} \in \mathbb{Z}^m$

Short Integer Solution [Ajtai96] and Learning With Errors [Regev05] problems

- Params: $m, n, q \geq 0$, $A \hookleftarrow U(\mathbb{Z}_q^{m \times n})$



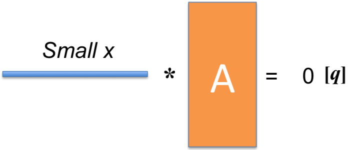
SIS
Find small $\mathbf{x} \in \mathbb{Z}^m \setminus \mathbf{0}$
s.t. $\mathbf{x}^t A = \mathbf{0}\ [q]$



LWE
Dist. $A\mathbf{s} + \mathbf{e}\ [q]$ and $U(\mathbb{Z}_q^m)$, for $\mathbf{s} \hookleftarrow U(\mathbb{Z}_q^n)$, noise $\mathbf{e} \in \mathbb{Z}^m$

SIS $\rightarrow$ k-SIS and LWE $\rightarrow$ k-LWE

- Params: $m, n, q \geq 0$, $A \hookleftarrow U(\mathbb{Z}_q^{m\times n})$
- k small hints $(\mathbf{x}_i)_{i\leq k}$ s.t. $\mathbf{x}_i^t A = \mathbf{0}$ $[q]$

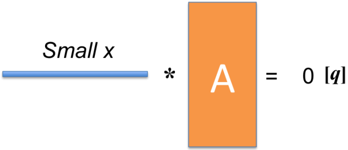


k-**SIS** [Boneh-Freeman11]
Find small $\mathbf{x} \in \mathbb{Z}^m$ s.t.

- $\mathbf{x}^t A = \mathbf{0}$ $[q]$
- $\mathbf{x} \notin \mathrm{Span}_{i\leq k}(\mathbf{x}_i)$

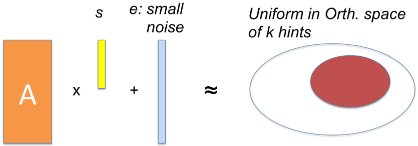
SIS $\to$ k-SIS and LWE $\to$ k-LWE

- Params: $m, n, q \geq 0$, $A \hookleftarrow U(\mathbb{Z}_q^{m \times n})$
- k small hints $(\mathbf{x}_i)_{i \leq k}$ s.t. $\mathbf{x}_i^t A = \mathbf{0}$ $[q]$



k-SIS [Boneh-Freeman11]
Find small $\mathbf{x} \in \mathbb{Z}^m$ s.t.

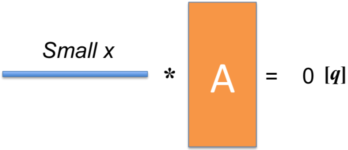
- $\mathbf{x}^t A = \mathbf{0}$ $[q]$
- $\mathbf{x} \notin \mathrm{Span}_{i \leq k}(\mathbf{x}_i)$



k-LWE
Distinguish $A\mathbf{s} + \mathbf{e}$ and $U(\mathrm{Span}_{i \leq k}(\mathbf{x}_i)^{\perp}) + \mathbf{e}'$ for $\mathbf{s} \hookleftarrow U(\mathbb{Z}_q^n)$ and small noises $\mathbf{e}, \mathbf{e}' \in \mathbb{Z}^m$

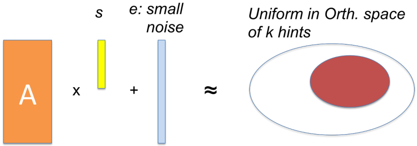
SIS $\rightarrow$ k-SIS and LWE $\rightarrow$ k-LWE

- Params: $m, n, q \geq 0$, $A \hookleftarrow U(\mathbb{Z}_q^{m\times n})$
- k small hints $(\mathbf{x}_i)_{i\leq k}$ s.t. $\mathbf{x}_i^t A = \mathbf{0}\ [q]$



k-**SIS** [Boneh-Freeman11]
Find small $\mathbf{x} \in \mathbb{Z}^m$ s.t.

- $\mathbf{x}^t A = \mathbf{0}\ [q]$
- $\mathbf{x} \notin \mathrm{Span}_{i\leq k}(\mathbf{x}_i)$

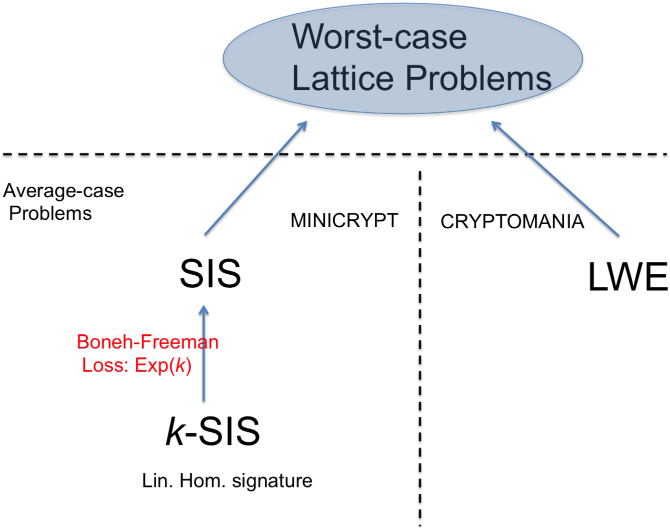


k-**LWE**
Distinguish $A\mathbf{s} + \mathbf{e}$
and $U(\mathrm{Span}_{i\leq k}(\mathbf{x}_i)^{\perp}) + \mathbf{e}'$
for $\mathbf{s} \hookleftarrow U(\mathbb{Z}_q^n)$ and small noises $\mathbf{e}, \mathbf{e}' \in \mathbb{Z}^m$

Original application of k-SIS: Homomorphic signatures [Boneh-Freeman11]

Hardness of *k*-SIS

[Boneh-Freeman11]

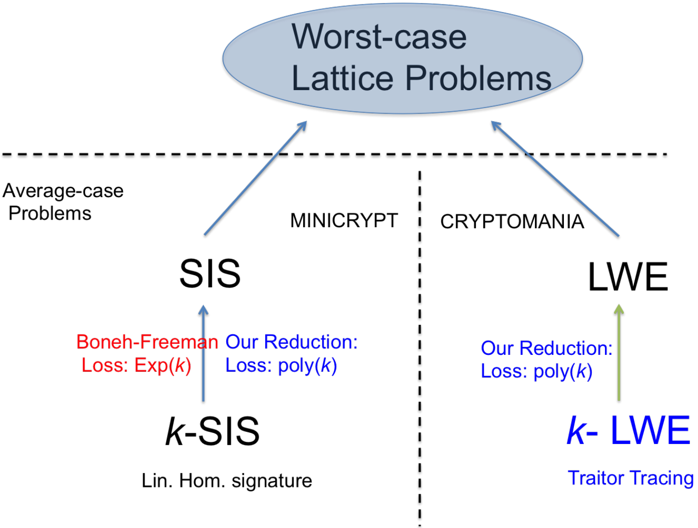


About BF reduction from SIS to k-SIS

[Boneh-Freeman11] " Our reduction degrades exponentially in k, which forces us to use a constant-size k if we want our linearly homomorphic scheme to be provably secure based on worst-case lattice problems. **It is an important open problem to give a tighter reduction**."

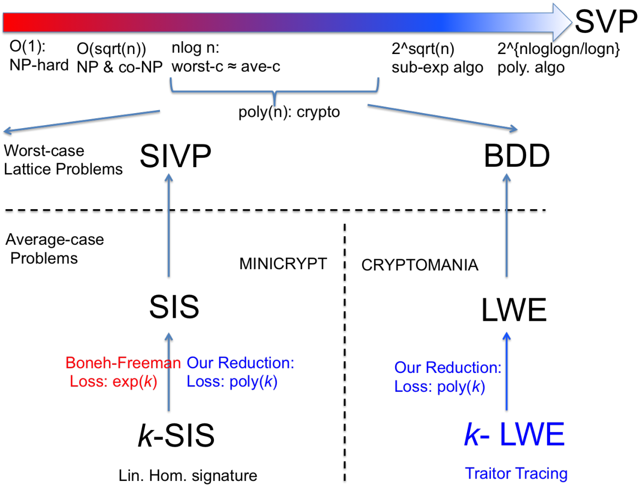
Hardness of *k*-LWE and *k*-SIS

[Ling-Phan-Stehlé-Steinfeld, Crypto14, Algorithmica16]



Hardness of *k*-LWE

[Ling-Phan-Stehlé-Steinfeld, Crypto14]



back

Computing on Encrypted Data

Computing on Encrypted Data: FHE/ Functional Encryption

Fully homomorphic encryption

- RSA is additionally homomorphic
- ElGamal is multiplicatively homomorphic

It was an long standing open question to construct a fully homomorphic encryption until the breakthrough of Gentry 09.

Computing on Encrypted Data: FHE/ Functional Encryption

Fully homomorphic encryption

- RSA is additionally homomorphic
- ElGamal is multiplicatively homomorphic

It was an long standing open question to construct a fully homomorphic encryption until the breakthrough of Gentry 09.

Functional Encryption

- Classical encryption: $\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(m)) = m$
- Functional encryption: $\mathcal{FE}.\mathsf{Dec}(\mathsf{sk}_f, \mathcal{FE}.\mathsf{Enc}(\mathbf{m})) = f(m)$

Functional Encryption / Inner-Product FE

Functional Encryption

- Classical encryption: $\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(m)) = m$
- Functional encryption: $\mathcal{FE}.\mathsf{Dec}(\mathsf{sk}_f, \mathcal{FE}.\mathsf{Enc}(\mathbf{m})) = f(m)$

Functional Encryption / Inner-Product FE

Functional Encryption

- Classical encryption: $\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(m)) = m$
- Functional encryption: $\mathcal{FE}.\mathsf{Dec}(\mathsf{sk}_f, \mathcal{FE}.\mathsf{Enc}(\mathbf{m})) = f(m)$

Inner-Product Functional Encryption over $\mathbb{Z}_p^\ell$

- secret key encodes a vector $\mathbf{x} \in \mathbb{Z}_p^\ell$: $\mathcal{FE}.\mathsf{KeyGen}(\mathbf{x}) \to \mathsf{sk}_\mathbf{x}$
- ciphertext encodes a vector $\mathbf{v} \in \mathbb{Z}_p^\ell$: $\mathcal{FE}.\mathsf{Enc}(\mathsf{pk}, \mathbf{v}) \to C$
- decryption recovers the inner product

$$\mathcal{FE}.\mathsf{Dec}(\mathsf{sk}_\mathbf{x}, C) \to \langle \mathbf{x}, \mathbf{v} \rangle \bmod p$$

Functional Encryption / Inner-Product FE

Functional Encryption

- Classical encryption: $\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(m)) = m$
- Functional encryption: $\mathcal{FE}.\mathsf{Dec}(\mathsf{sk}_f, \mathcal{FE}.\mathsf{Enc}(\mathbf{m})) = f(m)$

Inner-Product Functional Encryption over $\mathbb{Z}_p^\ell$

- secret key encodes a vector $\mathbf{x} \in \mathbb{Z}_p^\ell$: $\mathcal{FE}.\mathsf{KeyGen}(\mathbf{x}) \rightarrow \mathsf{sk}_\mathbf{x}$
- ciphertext encodes a vector $\mathbf{v} \in \mathbb{Z}_p^\ell$: $\mathcal{FE}.\mathsf{Enc}(\mathsf{pk}, \mathbf{v}) \rightarrow C$
- decryption recovers the inner product

$$\mathcal{FE}.\mathsf{Dec}(\mathsf{sk}_\mathbf{x}, C) \rightarrow \langle \mathbf{x}, \mathbf{v} \rangle \bmod p$$

- Efficient solutions [ADBP15, ALS15...]
- Our new result: Decentralized multi-client IPFE (Asiacrypt '18)

Different Tools for the Design of Advanced Primitives

- **Group, Pairings:** IBE, BE, TT [1], ABE, zk-SNARK, Voting, Inner-Product FE, Decentralized IPFE [2], 2-DNF FHE.
- **Lattice:** IBE, BE&TT [3,4], ABE, Inner-Product FE, FHE.
- **Coding:** IBE [5]
- **Combinatorics**: Group testing, Collusion secure code, IPP code, BE, Trace & Revoke code [6].

→ A large number of open problems!

Concluding Discussions

- Standard primitives:
  - Encryption for confidentiality
  - Hash functions for integrity
  - MAC, digital signature for authentification
  - Interactive, zero-knowledge proofs (used in IND-CCA PKE, multi-party computation,...)

Concluding Discussions

- Standard primitives:
  - Encryption for confidentiality
  - Hash functions for integrity
  - MAC, digital signature for authentification
  - Interactive, zero-knowledge proofs (used in IND-CCA PKE, multi-party computation,...)
- Advanced primitives:
  - Multi-user cryptography (BE, TT, ABE, GS...)
  - Computing in encrypted data (FHE, FE, machine learning/AI on encrypted data...)

Concluding Discussions

- Standard primitives:
  - Encryption for confidentiality
  - Hash functions for integrity
  - MAC, digital signature for authentification
  - Interactive, zero-knowledge proofs (used in IND-CCA PKE, multi-party computation,...)
- Advanced primitives:
  - Multi-user cryptography (BE, TT, ABE, GS...)
  - Computing in encrypted data (FHE, FE, machine learning/AI on encrypted data...)
- In these revolutionary years of technology:
  - Everyone should care about the privacy and the confidentiality
  - No abuse of data access, from the companies or from the governments
  - Should deal with powerful adversaries (quantum, collaborative attacks,...)