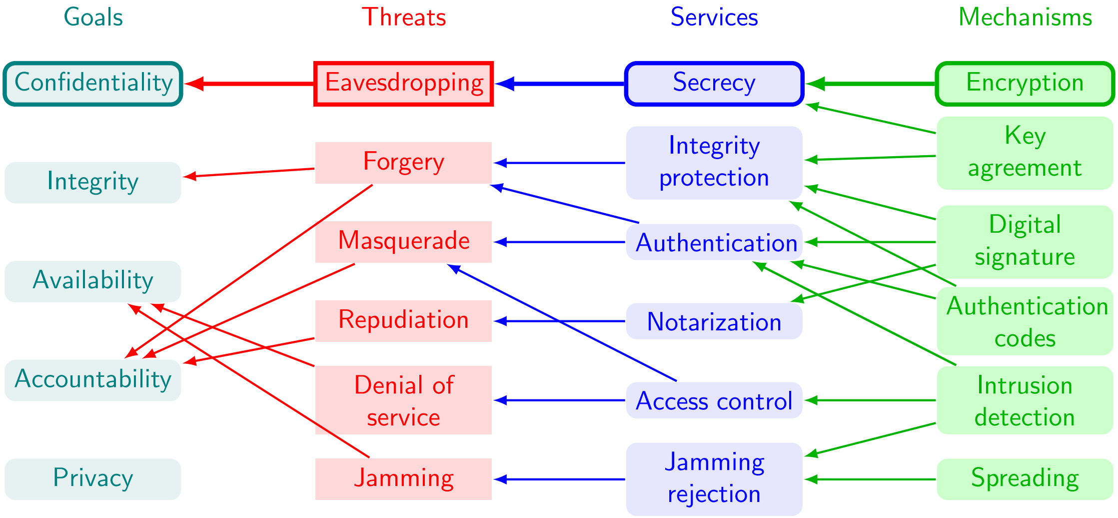
Lecture 9

Asymmetric (aka public key) encryption

Nicola Laurenti October 28, 2020

Lecture 9— Contents

Security goals, threats, services and mechanisms

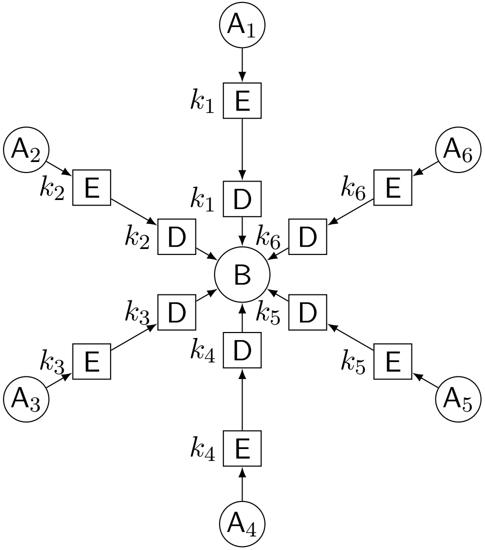


Motivation for asymmetric encryption

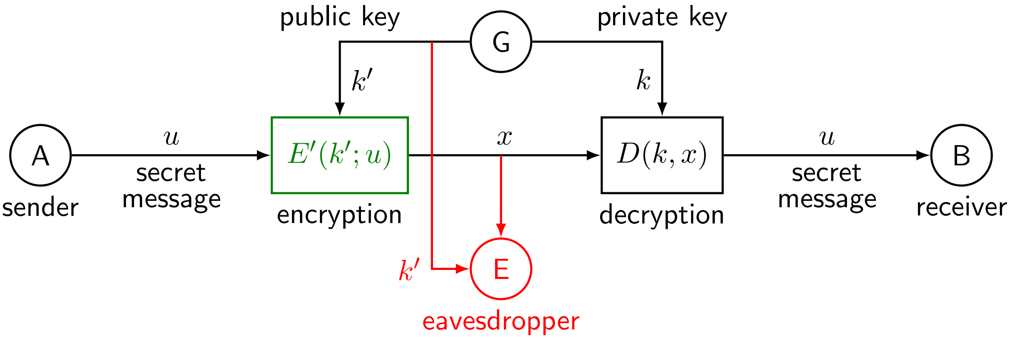
Consider the problem of a single user B having to receive confidential messages $u_1, \ldots, u_N$ from each of N different sources A$_i$, so that B obtains message u_i but any A$_i$ cannot learn any message u_j, $j \neq i$.

With a symmetric encryption mechanism $(\mathcal{M}, \mathcal{X}, \mathcal{K}, E, D, p_k, p_u)$, B must agree and share a different key k_i with any A$_i$

Can we build a mechanism where B uses a single key k_B ?



General model of an asymmetric encryption system



Glossary and notation

private key $k \in \mathcal{K}$ private key space

public key $k' \in \mathcal{K}'$ public key space

(reparametrized) encryption map $E' : \mathcal{K}' \times \mathcal{M} \mapsto \mathcal{X}$

$$E_{k'} : \mathcal{M} \mapsto \mathcal{X} \quad E_{k'}(u) \doteq E(k', u)$$

decryption map $D : \mathcal{K} \times \mathcal{X} \mapsto \mathcal{M}$

$$D_k : \mathcal{X} \mapsto \mathcal{M} \quad D_k(x) \doteq D(k, x)$$

Keys are random with joint probability mass distribution $p_{kk'} : \mathcal{K} \times \mathcal{K}' \mapsto [0, 1]$
typically $(k, k') \not\sim \mathcal{U}(\mathcal{K} \times \mathcal{K}')$ are uniform but not independent
often $k \sim \mathcal{U}(\mathcal{K})$ is random and uniform, $k' = f(k)$ is computed with $f : \mathcal{K} \mapsto \mathcal{K}'$ deterministic
The encryption system is completely specified as:

$$\mathcal{S} = (\mathcal{M}, \mathcal{X}, \mathcal{K}, \mathcal{K}', E', D, p_u, p_{kk'})$$

General assumptions

- (perfect reliability) The receiver must be able to recover the secret message perfectly

$$D_c = E_c^{-1} = (E'_{c'})^{-1} \quad \forall c \in \mathcal{K}, c' \in \mathcal{K}' \, : \, p_{kk'}(c, c') > 0 \quad \text{(or } c' = f(c)\text{)}$$

- (Kerchoff's assumption) The eavesdropper knows the system $\mathcal{S}$ (in particular the maps $E'(\cdot,\cdot)$ and $D(\cdot,\cdot)$)

Where does secrecy come from?

Secrecy can only be computational and is based on the following requirements

1. it is hard to derive k from k' (i.e., f is one-way)
2. it is hard to derive u from (k', x) (i.e., $E'_{k'}$ is one-way)
3. it is hard to derive k from (u, x) (i.e., $D(\cdot, x)$ is one-way)

One-way function: definitions

One-way functions are a fundamental tool in many computationally secure mechanisms and their analysis. They are informally referred to as "easy to compute and hard to invert".

Definition (concrete)

A function $f : \mathcal{X} \mapsto \mathcal{Y}$ is said to be $(\varepsilon_0, T_0; \varepsilon_1, T_1)$-one-way if

(easy to compute) there exists a probabilistic algorithm A such that

$$\forall x \in \mathcal{X} \quad , \quad \mathrm{P}\left[\{\mathtt{A}[x] \to f(x)\} \cap \{T_{\mathtt{A}} \leq T_1\}\right] \geq 1 - \varepsilon_1$$

(hard to invert) for any probabilistic algorithm B

$$\forall y \in \mathcal{Y} \quad , \quad \sum_{x \in f^{-1}(y)} \mathrm{P}\left[\{\mathtt{B}[y] \to x\} \cap \{T_{\mathtt{B}} \leq T_0\}\right] \leq \varepsilon_0$$

A deterministic variant for the easy to compute requires that there exists a deterministic algorithm A such that $T_{\mathtt{A}} \leq T_1$ and $\mathtt{A}[x] \to f(x)$, $\forall x \in \mathcal{X}$

One-way function: definitions

In order to provide an asymptotic definition we introduce a security parameter n

Definition (asymptotic)

A sequence $\{f_n\}, n \in \mathbb{N}$ of functions $f_n : \mathcal{X}_n \mapsto \mathcal{Y}_n$ is one-way if

(easy to compute) $\forall \varepsilon > 0$, there exists a sequence of probabilistic algorithms $\mathtt{A}_n$ and a polynomial $p(\cdot)$ such that

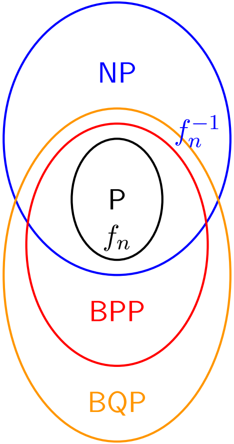
$$\forall n \in \mathbb{N} \ , \quad \forall x \in \mathcal{X}_n \quad , \qquad \mathrm{P}\left[\{\mathtt{A}_n[x] \to f_n(x)\} \cap \{T_{\mathtt{A}_n} \leq p(n)\}\right] \geq 1 - \varepsilon$$

(hard to invert) for any sequence of probabilistic algorithms $\mathtt{B}_n$, and any polynomials $q(\cdot), s(\cdot)$, there is a n_0 such that

$$\forall n > n_0 \ , \quad \forall y \in \mathcal{Y}_n \quad , \qquad \sum_{x \in f_n^{-1}(y)} \mathrm{P}\left[\{\mathtt{B}_n[y] \to x\} \cap \{T_{\mathtt{B}_n} \leq q(n)\}\right] \leq \frac{1}{s(n)}$$

Deterministic easy to compute requires a sequence of deterministic algorithms $\mathtt{A}_n$ such that $T_{\mathtt{A}_n} \leq p(n)$ and $\mathtt{A}_n[x] \to f_n(x)$, $\forall x \in \mathcal{X}_n$

Relationships between one-way functions and complexity classes



- The problem of computing a one-way function f_n must $\in$ BPP
- The problem of inverting a one-way function f_n must $\notin$ BPP
- Typically, the problem of computing a one-way function f_n $\in$ P and that of inverting it $\in$ NP, as a candidate inverse x can be verified by computing $f_n(x)$

The RSA cryptosystem [Rivest-Shamir-Adleman, '77]

Based on NP problems

- integer factorization

  easy given $p, q \in \mathbb{Z}$, compute $n = pq$

  hard given $n \in \mathbb{Z}$, find $p, q \in \mathbb{Z}$ such that $pq = n$

- finite logarithm and finite root

  easy given $n \in \mathbb{Z}$, $x, d \in \mathbb{Z}_n$ compute $y = x^d \bmod n$ (finite exponential)

  hard given $n \in \mathbb{Z}$, $x, y \in \mathbb{Z}_n$ find $d \in \mathbb{Z}_n$ such that $x^d \bmod n = y$

  hard given $n \in \mathbb{Z}$, $d, y \in \mathbb{Z}_n$ find $x \in \mathbb{Z}_n$ such that $x^d \bmod n = y$

The RSA cryptosystem

Key generation (ℓ-bit)

B chooses $p, q < 2^{\ell/2}$ primes
computes $n = pq$, $\varphi = (p-1)(q-1)$
chooses $d \in \mathbb{Z}_n$ such that $\gcd(\varphi, d) = 1$
computes $e \in \mathbb{Z}_n$ such that $ed = 1 \pmod{\varphi}$

private key $k = (p, q, d)$, $\mathcal{K} = \mathbb{Z}^3_{2^\ell}$

public key $k' = (n, e)$, $\mathcal{K}' = \mathbb{Z}^2_{2^\ell}$

Encryption by A (public key)

$$\mathcal{M} = \mathcal{X} = \mathbb{Z}_n$$

$$E' : \mathcal{K}' \times \mathcal{M} \mapsto \mathcal{X}$$

$$x = E'(k', u) = E'(n, e, u) = u^e \bmod n$$

Decryption by B (private key)

$$D : \mathcal{K} \times \mathcal{X} \mapsto \mathcal{M}$$

$$\hat{u} = D(k, x) = D(n, d, x) = x^d \bmod n$$

The RSA cryptosystem

Theorem (Euler's theorem)

Let $n, \varphi \in \mathbb{Z}$ *as in the key generation and* $u \in \mathbb{Z}_n$. *If* $\gcd(u, n) = 1$, *then* $u^\varphi = 1 \pmod n$

Correctness of RSA

We show that $\hat{u} = u$. Consider the equalities in $\mathbb{Z}_n$

$$\hat{u} = x^d = (u^e)^d = u^{ed} = u^{r\varphi+1}$$

with r an arbitrary integer. Then by Euler's theorem, in $\mathbb{Z}_n$

$$\hat{u} = (u^\varphi)^r u = 1^r u = u$$

The RSA cryptosystem

Computability

choosing p, q primes probabilistic algorithm $O(\ell)$: randomly generate them, then check if primes, else repeat. Probabilistic primality test run in $O(\ell)$ (e.g., Fermat test), the fastest deterministic primality test (Lenstra-Pomerance variant of the AKS test) has complexity $O(\ell^6)$ (still prohibitive)

computing n, φ is $O(\ell)$

choosing d probabilistic algorithm $O(\ell)$: randomly generate d, then check if coprime with φ, else repeat. Coprimality can be tested with Euclidean algorithm that is $O(\ell)$

computing e can be done with Euclidean algorithm

encryption and decryption finite exp $O(\ell^2)$ (typically, $e \ll n$, so encryption is fast)

The RSA cryptosystem

Security

$x = E'_{k'}(u)$ is one-way finding u from x and e is hard (finite root)

$k' = f(k)$ is one-way finding d from e, without knowing φ is hard

finding φ from n is hard (no easier than finding p, q)

finding p, q from n is hard (integer factorization)

$u = D(\cdot, x)$ is one-way finding d from (u, x) is hard (finite logarithm)

The Elgamal cryptosystem [Elgamal, ‘85]

Based on NP problem

finite logarithm

In a group $(\mathbb{G}, \circ)$, we denote $\alpha \overset{n}{\circ} = \underbrace{\alpha \circ \cdots \circ \alpha}_{n \text{ times}}$

easy given $\alpha \in \mathbb{G}, n \in \mathbb{N}$, compute $\beta = \alpha \overset{n}{\circ}$

hard given $\alpha, \beta \in \mathbb{G}$, find $n \in \mathbb{N}$ such that $\alpha \overset{n}{\circ} = \beta$

Key generation

Let $(\mathbb{G}, \circ)$ be a group with a primitive element $\alpha \in \mathbb{G}$, i.e. such that $\forall \beta \in \mathbb{G}, \exists n : \alpha \overset{n}{\circ} = \beta$.

private key space $\mathcal{K} = \{1, \ldots, |\mathbb{G}| - 1\} \subset \mathbb{N}$

public key space $\mathcal{K}' = \mathbb{G}$

Let $(\mathbb{G}, \circ)$ and α be publicly known. B generates $k \sim \mathcal{U}(\mathcal{K})$, then computes $k' = f(k) = \alpha \overset{k}{\circ}$

The Elgamal cryptosystem

Encryption by A (public key, probabilistic)

$\mathcal{M} = \mathbb{G} \quad , \quad \mathcal{X} = \mathbb{G}^2$

A generates $b \sim \mathcal{U}(\mathcal{K})$

$$x = E'_{k'}(u, b) = (x_1, x_2) \quad , \quad \begin{cases} x_1 = \alpha^{\overset{b}{\circ}} \\ x_2 = u \circ (k'^{\overset{b}{\circ}}) \end{cases}$$

Decryption by B (private key)

B need not know b

$$\hat{u} = D_k(x) = D_k(x_1, x_2) = x_2 \circ \left((x_1^{\overset{k}{\circ}})^{\overset{-1}{\circ}} \right)$$

where $\cdot^{\overset{-1}{\circ}}$ denotes the inverse in $(\mathbb{G}, \circ)$

The Elgamal cryptosystem

Correctness

We prove that $\hat{u} = u$

$$
\begin{aligned}
\hat{u} &= x_2 \circ \left(\left(x_1^{\overset{k}{\circ}} \right)^{\overset{-1}{\circ}} \right) \\
&= u \circ \left(k'^{\overset{b}{\circ}} \right) \circ \left(\left(\left(\alpha^{\overset{b}{\circ}} \right)^{\overset{k}{\circ}} \right)^{\overset{-1}{\circ}} \right) \\
&= u \circ \left(\left(\alpha^{\overset{k}{\circ}} \right)^{\overset{b}{\circ}} \right) \circ \left(\left(\left(\alpha^{\overset{b}{\circ}} \right)^{\overset{k}{\circ}} \right)^{\overset{-1}{\circ}} \right) \\
&= u \circ \left(\alpha^{\overset{kb}{\circ}} \right) \circ \left(\left(\alpha^{\overset{kb}{\circ}} \right)^{\overset{-1}{\circ}} \right) \\
&= u \circ e = u
\end{aligned}
$$

where e denotes the identity in $(\mathbb{G}, \circ)$

The Elgamal cryptosystem

Security

$x = E'_{k'}(u)$ is one-way given x and k', but not k nor b, it is hard to find u

$k' = \alpha \overset{k}{\circ}$ is one-way finding k from k' is hard (finite log problem)

$D(\cdot, x)$ is one-way given x and u, it is hard to find k from the equation $x_1 \overset{k}{\circ} = (u \overset{-1}{\circ}) \circ x_2$ (finite log problem)

Importance of b

secret if attacker learns b he can find $u = x_2 \circ (k' \overset{b}{\circ}) \overset{-1}{\circ}$

varied if the same b is used to encrypt both u and u', then $x_1 = x'_1$ and

$$x'_2 \circ (x_2 \overset{-1}{\circ}) = u' \circ (k' \overset{b}{\circ}) \circ \left(u \circ (k' \overset{b}{\circ})\right) \overset{-1}{\circ} = u' \circ (k' \overset{b}{\circ}) \circ ((k' \overset{b}{\circ}) \overset{-1}{\circ}) \circ (u \overset{-1}{\circ})$$

attacker can do a KPA, learn u' from u, x_2, x'_2

The finite logarithm problem

A strong requirement for security of the Elgamal encryption is that "exponentiation" $f_\alpha(n) = \alpha^{\overset{n}{\circ}}$ is a one-way function of n and this depends on the choice of the group $(\mathbb{G}, \circ)$.
If computing $\circ$ has linear complexity in $\ell = \log |\mathbb{G}|$, exponentiation can be computed with complexity $O(\ell^2)$, by iterative squaring and multiplying

For a general group $(\mathbb{G}, \circ)$

Consider the computation of $y = x^{\overset{n}{\circ}}$, with $n < |\mathbb{G}|$.
Let $\boldsymbol{b} = [b_0, b_1, \ldots, b_{\ell-1}]$ be the binary representation of n

$$n = \sum_{i=0}^{\ell-1} b_i 2^i$$

Then $y = x^{\overset{n}{\circ}} = x^{\overset{\sum_{i=0}^{\ell-1} b_i 2^i}{\circ}} = \left(x^{\overset{b_0 2^0}{\circ}}\right) \circ \cdots \circ \left(x^{\overset{b_{\ell-1} 2^{\ell-1}}{\circ}}\right)$

Iterative square and multiply

$c \leftarrow e$ (identity in $\mathbb{G}$)
$a \leftarrow x$
for $i = 0$ to $\ell - 1$ **do**
  if $b_i = 1$ **then**
    $c \leftarrow c \circ a$
  end if
  $a \leftarrow a \circ a$
end for
$y \leftarrow c$

The finite logarithm problem

Regarding inversion, different cases exist, for instance:

- if $(\mathbb{G}, \circ) = (\mathbb{Z}_p, \cdot)$ is the multiplicative group of integers modulo some prime p, it holds, the best algorithms run in $O(\sqrt{p}) = O(2^{\ell/2})$ time
- if $(\mathbb{G}, \circ) = (\mathbb{Z}_N, +)$ is the additive group of integers modulo N, it does not hold, infact:

$$f_\alpha(n) = \alpha \overset{n}{\circ} = n\alpha \mod N \quad \Rightarrow \quad f_\alpha^{-1}(\beta) = \beta/\alpha \bmod N$$

  and $f_\alpha^{-1}(\cdot)$ can be computed efficiently via the Euclidean algorithm
- in general the finite log problem is at most $O(|\mathbb{G}|) = O(2^\ell)$ time, one can find groups where the fastest known algorithms run in $O(2^\ell)$ time

Finite elliptic curve arithmetics

Given a finite field $(\mathbb{F}, +, \cdot)$, an elliptic curve on $\mathbb{F}$ is the set of points (locus)

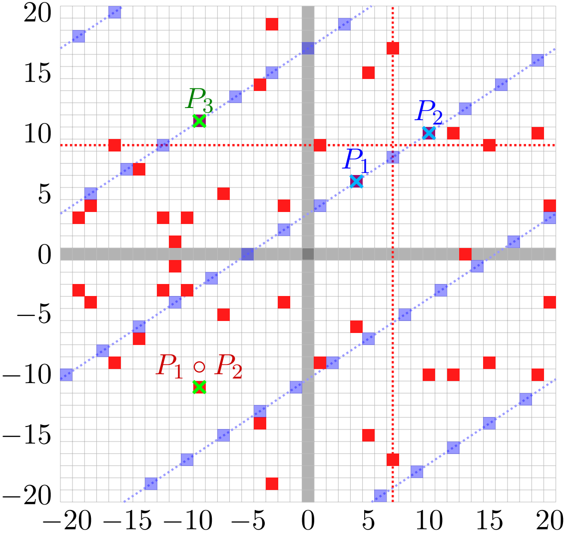
$$\mathcal{E} = \left\{(x, y) \in \mathbb{F}^2 \, : \, y^2 = x^3 + ax + b\right\}$$

for some coefficients $a, b \in \mathbb{F}$.

The set $\mathcal{E}$ can be made a group $(\mathcal{E}, \circ)$ by equipping it with an operation $\circ$ (called point addition) between two points P_1 and P_2, that yields a third point $P_1 \circ P_2$

In the elliptic curve group $(\mathcal{E}, \circ)$ the finite logarithm problem is harder than in $(\mathbb{Z}_p, \cdot)$ with the same cardinality.

Group operation between two points



$$\mathbb{F} = GF(41)$$

$$\mathcal{E} : y^2 = x^3 + 5x - 7 \quad , \quad |\mathcal{E}| = 43$$

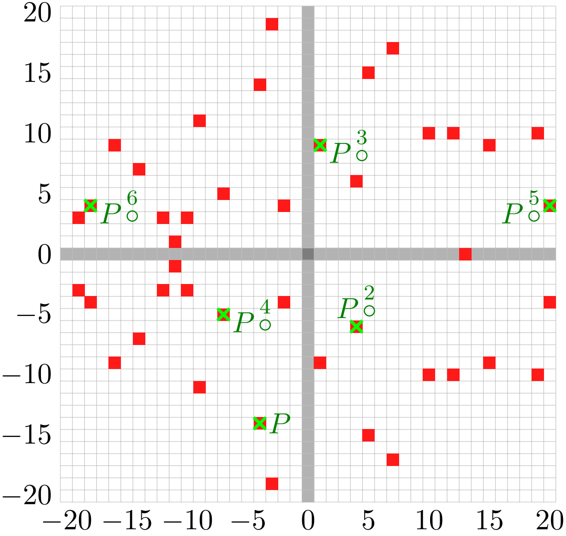
$$P_1 = (4, 6) \quad , \quad P_2 = (10, 10)$$

$$r : -8x + 12y + 1 = 0 \quad , \quad |r| = 41$$

$$P_3 = (-9, 11)$$

$$P_1 \circ P_2 = (-9, -11)$$

n-fold group operation



$$\mathbb{F} = GF(41)$$

$$\mathcal{E} \;:\; y^2 = x^3 + 5x - 7 \quad , \qquad |\mathcal{E}| = 43$$

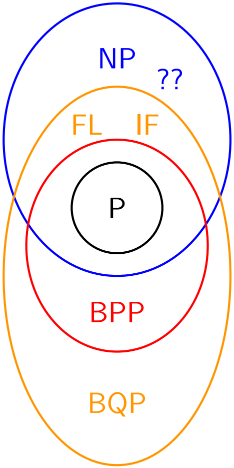
$$P = (-4, -14)$$

$$P^{\circ 2} = P \circ P = (4, -6)$$

$$P^{\circ 3} = (P^{\circ 2}) \circ P = (1, 9)$$

$$P^{\circ 4} = (P^{\circ 3}) \circ P = (P^{\circ 2}) \circ (P^{\circ 2}) = (-7, -5)$$

The Shor algorithm and post-quantum cryptography



In 1994, Peter Shor invented a quantum algorithm to efficiently compute the order of any element x in a group $(\mathbb{G}, \circ)$, that is the minimum positive integer n for which $x \overset{n}{\circ} = x$.
As a consequence,

- The finite logarithm problem is shown to $\in$ BQP $\Rightarrow$ ~~Elgamal~~
- The integer factorization problem is shown to $\in$ BQP $\Rightarrow$ ~~RSA~~
- Other NP problems may $\notin$ BQP

Mechanisms that rely only on NP problems that are not known to be $\in$ BQP are called post quantum.
We shall see an example shortly

The McEliece cryptosystem [McEliece, ‘78]

Based on NP problem

minimum Hamming distance (mHd) decoding of binary codes

In a (n, ℓ, t) linear binary FEC code (e.g., Goppa codes) with

n	codeword length
ℓ	code dimension = information word length
t	maximum nr. of correctable errors

easy given an information word $\boldsymbol{b} \in \mathbb{B}^{\ell}$ and a generating matrix $\boldsymbol{G} \in \mathbb{B}^{n \times \ell}$, compute the codeword $\boldsymbol{c} = \boldsymbol{G}\boldsymbol{b} \in \mathbb{B}^{n}$

hard given a received word (not necessarily a codeword) $\tilde{\boldsymbol{c}} \in \mathbb{B}^{n}$ and a generating matrix $\boldsymbol{G} \in \mathbb{B}^{n \times \ell}$, compute $\hat{\boldsymbol{b}} = \arg\min_{\boldsymbol{\beta} \in \mathbb{B}^{\ell}} d_{\mathsf{H}}(\tilde{\boldsymbol{c}}, \boldsymbol{G}\boldsymbol{\beta})$

easy given a received word (not necessarily a codeword) $\tilde{\boldsymbol{c}} \in \mathbb{B}^{n}$ and a generating matrix $\boldsymbol{G} \in \mathbb{B}^{n \times \ell}$ in canonical form, compute $\hat{\boldsymbol{b}} = \arg\min_{\boldsymbol{\beta} \in \mathbb{B}^{\ell}} d_{\mathsf{H}}(\tilde{\boldsymbol{c}}, \boldsymbol{G}\boldsymbol{\beta})$

The McEliece cryptosystem

Key generation

1. B chooses $\boldsymbol{G} \in \mathbb{B}^{n \times \ell}$ canonical generating matrix of a (n, ℓ, t) Goppa code
2. generates $\boldsymbol{S} \in \mathbb{B}^{\ell \times \ell}$ non singular
3. generates $\boldsymbol{P} \in \mathbb{B}^{n \times n}$ a permutation matrix (exactly one '1' in each row and column)
4. computes $\boldsymbol{S}^{-1}, \boldsymbol{P}^{-1}$, and $\boldsymbol{G}' = \boldsymbol{P}^{-1}\boldsymbol{G}\boldsymbol{S}^{-1} \in \mathbb{B}^{n \times \ell}$ noncanonical generating matrix of an equivalent (n, ℓ, t) Goppa code

private key $k = (\boldsymbol{G}, \boldsymbol{P}, \boldsymbol{S}) \quad , \qquad \mathcal{K} = \mathbb{B}^{n \times \ell} \times \mathbb{B}^{n \times n} \times \mathbb{B}^{\ell \times \ell}$

public key $k' = f(k) = (\boldsymbol{G}', t) \quad , \qquad \mathcal{K}' = \mathbb{B}^{n \times \ell} \times \mathbb{N}$

The McEliece cryptosystem

Encryption by A (public key, probabilistic)

$$\mathcal{M} = \mathbb{B}^\ell \quad , \quad \mathcal{X} = \mathbb{B}^n$$

A generates a random $\boldsymbol{e} \in \mathbb{B}^n$ such that $w_\mathsf{H}(\boldsymbol{e}) \leq t$ (i.e., a correctable error pattern)

$$E'_{k'} : \boldsymbol{x} = \boldsymbol{G}'\boldsymbol{u} + \boldsymbol{e}$$

Decryption by B (private key)

$\hat{u} = D(k, x) = D(\boldsymbol{G}, \boldsymbol{P}, \boldsymbol{S}, x)$ is computed as follows

1. B computes $\boldsymbol{x}' = \boldsymbol{P}\boldsymbol{x}$
2. B solves the mHd decoding of $\boldsymbol{x}'$ in the Goppa code with canonical $\boldsymbol{G}$, i.e.,
$$\boldsymbol{u}' = \arg\min_{\boldsymbol{\beta} \in \mathbb{B}^\ell} d_\mathsf{H}(\boldsymbol{x}', \boldsymbol{G}\boldsymbol{\beta})$$
3. B computes $\hat{\boldsymbol{u}} = \boldsymbol{S}\boldsymbol{u}'$

The McEliece cryptosystem

Correctness

We prove that $\hat{\boldsymbol{u}} = \boldsymbol{u}$

$$\begin{aligned} \boldsymbol{x}' &= \boldsymbol{P}\boldsymbol{x} \\ &= \boldsymbol{P}(\boldsymbol{G}'\boldsymbol{u} + \boldsymbol{e}) \\ &= \boldsymbol{P}\boldsymbol{P}^{-1}\boldsymbol{G}\boldsymbol{S}^{-1}\boldsymbol{u} + \boldsymbol{P}\boldsymbol{e} \\ &= \boldsymbol{G}\boldsymbol{S}^{-1}\boldsymbol{u} + \boldsymbol{e}' \\ &= \boldsymbol{G}\boldsymbol{u}' + \boldsymbol{e}' \end{aligned}$$

where $\boldsymbol{u}' = \boldsymbol{S}^{-1}\boldsymbol{u}$ is an information word, too, and $\boldsymbol{e}' = \boldsymbol{P}\boldsymbol{e}$ has $w_{\mathsf{H}}(\boldsymbol{e}') = w_{\mathsf{H}}(\boldsymbol{e}) \leq t$, so it is a correctable error pattern, too.
Therefore the mHd decoding of $\boldsymbol{x}'$ with $\boldsymbol{G}$ is $\boldsymbol{u}'$ and

$$\hat{\boldsymbol{u}} = \boldsymbol{S}\boldsymbol{u}' = \boldsymbol{S}\boldsymbol{S}^{-1}\boldsymbol{u} = \boldsymbol{u}$$

The McEliece cryptosystem

Security

$x = E'_{k'}(u)$ is one-way given x and noncanonical G', it is hard to find u (mHd decoding)

$k' = f(k)$ is one-way given the non canonical $\boldsymbol{G}' = \boldsymbol{P}^{-1}\boldsymbol{G}\boldsymbol{S}^{-1}$ it is hard to factor it into $\boldsymbol{P}, \boldsymbol{G}, \boldsymbol{S}$ with a canonical $\boldsymbol{G}$

Summary

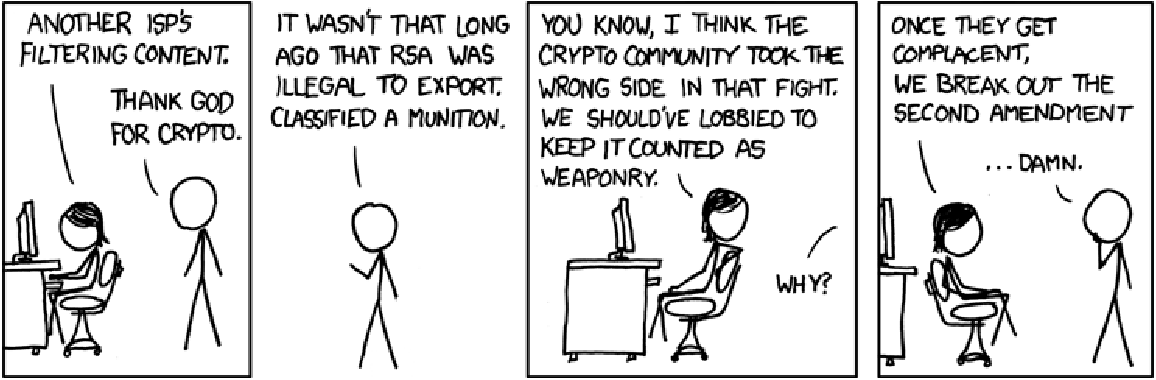
In this lecture we have:

- provided a general model for asymmetric encryption
- introduced the notion of one-way functions and described the RSA cryptosystem
- described the ElGamal cryptosystem and introduced elliptic curve cryptography
- introduced post quantum cryptography and described the McEliece cryptosystem

Assignment

- class notes
- textbook, §6.1, §6.3, §7.5, §9.1, §9.3

End of lecture



Legal Hacks, reproduced from xkcd URL: xkcd.com/504