MIT 6.875

Foundations of Cryptography
Lecture 10

Lectures 8-10

<u>Constructions of Public-key Encryption</u>

✅ Diffie-Hellman/El Gamal

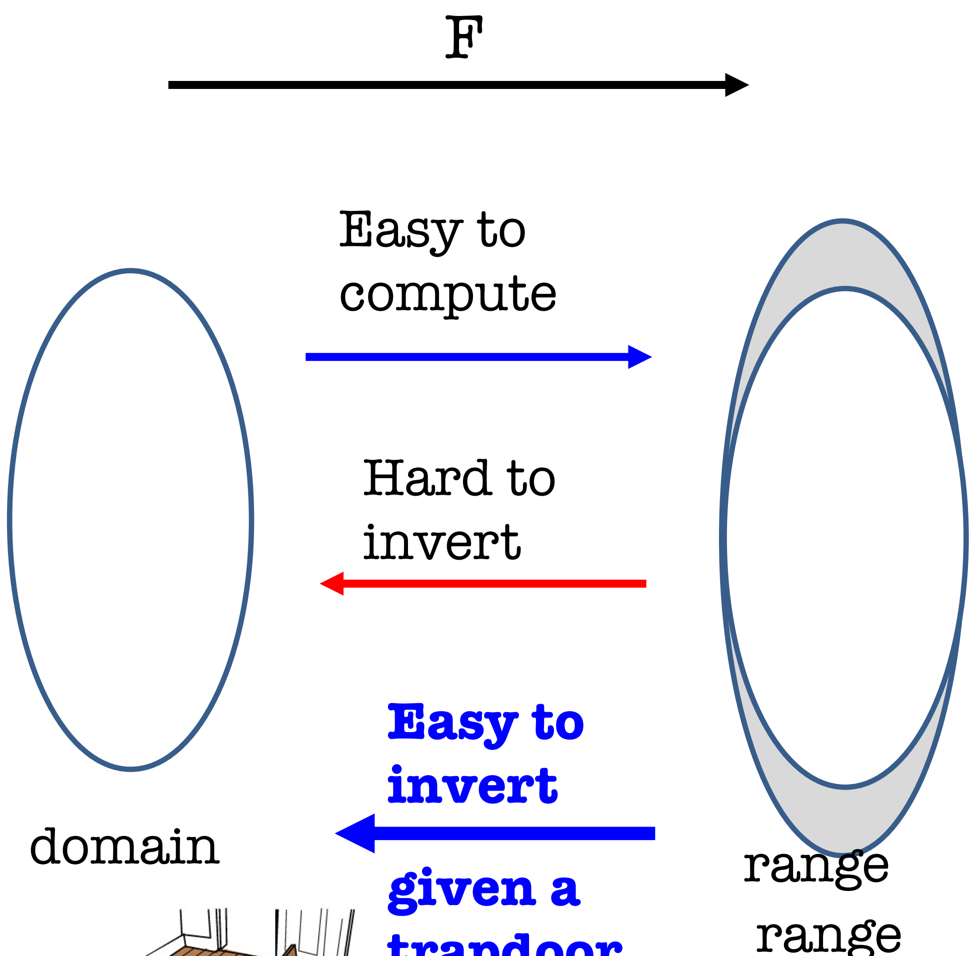
2: Trapdoor Permutations (RSA)

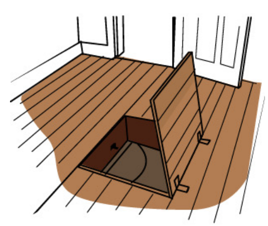
3: Quadratic Residuosity/Goldwasser-Micali

4: Post-Quantum Security & Lattice-based Encryption

Trapdoor One-Way Permutations

One Way Functions





Domain = Range

Trapdoor Functions: The Definition

A function (family) $\mathcal{F} = \{\mathcal{F}_n\}_{n\in\mathbb{N}}$ where **each $\mathcal{F}_n$ is itself a collection of functions** $\mathcal{F}_n = \{F_i: \{0,1\}^n \rightarrow \{0,1\}^{m(n)}\}_{i\in I_n}$ is a trapdoor one-way function family if:

- Easy to sample function index with a trapdoor: There is a PPT algorithm $Gen(1^n)$ that outputs a function index $i \in I_n$ together with a trapdoor t_i.

Trapdoor Functions: The Definition

A function (family) $\mathcal{F} = \{\mathcal{F}_n\}_{n\in\mathbb{N}}$ where **each $\mathcal{F}_n$ is itself a collection of functions** $\mathcal{F}_n = \{F_i: \{0,1\}^n \rightarrow \{0,1\}^{m(n)}\}_{i\in I_n}$ is a trapdoor one-way function family if:

- Easy to sample function index with a trapdoor.
- Easy to compute $F_i(x)$ given i and x.

Trapdoor Functions: The Definition

A function (family) $\mathcal{F} = \{\mathcal{F}_n\}_{n\in\mathbb{N}}$ where **each $\mathcal{F}_n$ is itself a collection of functions** $\mathcal{F}_n = \{F_i: \{0,1\}^n \to \{0,1\}^{m(n)}\}_{i\in I_n}$ is a trapdoor one-way function family if:

- Easy to sample function index with a trapdoor.
- Easy to compute $F_i(x)$ given i and x.

- Easy to compute an inverse of $F_i(x)$ given t_i.

Trapdoor Functions: The Definition

A function (family) $\mathcal{F} = \{\mathcal{F}_n\}_{n\in\mathbb{N}}$ where **each $\boldsymbol{\mathcal{F}_n}$ is itself a collection of functions** $\mathcal{F}_n = \{F_i: \{0,1\}^n \rightarrow \{0,1\}^{m(n)}\}_{i\in I_n}$ is a trapdoor one-way function family if:

- Easy to sample function index with a trapdoor.
- Easy to compute $F_i(x)$ given i and x.
- Easy to compute an inverse of $F_i(x)$ given t_i.

- It is one-way: that is, for every p.p.t. A, there is a negligible function μ s.t.

$$\Pr\left[\begin{matrix}\boldsymbol{(i,t) \leftarrow Gen(1^n)};\ x \leftarrow \{0,1\}^n; y = F_i(x); \\ A(1^n, i, y) = x' : y = F_i(x')\end{matrix}\right] \le \mu(n)$$

From Trapdoor Permutations to IND-Secure Public-key Encryption

- $Gen(1^n)$: Sample function index i with a trapdoor t_i. The public key is i and the private key is t_i.

- $Enc(pk = i, m)$: Output $c = F_i(m)$ as the ciphertext.

- $Dec(sk = t_i, c)$: Output $F_i^{-1}(c)$ computed using the private key t_i.

Could reveal partial info about m!
So, not IND-secure!

From Trapdoor Permutations to IND-Secure Public-key Encryption

- $Gen(1^n)$: Sample function index i with a trapdoor t_i. The public key is i and the private key is t_i.

- $Enc(pk = i, m)$ where m is a bit: **Pick a random $\boldsymbol{r}$. Output $\boldsymbol{c = (F_i(r), HCB(r) \oplus m)}$.**

- $Dec(sk = t_i, c)$: Recover r using the private key t_i, and using it m.

This is IND-CPA secure:
Proof by Hybrid argument (exercise).

Trapdoor Permutations: Candidates

Trapdoor Permutations are *exceedingly* rare.

Two candidates (both need factoring to be hard):

- **The RSA (Rivest-Shamir-Adleman) Function**
- The Rabin/Blum-Williams Function

Review: Number Theory

Let's review some number theory from L9.

Let $N = pq$ be a product of two large primes.

<ins>Fact</ins>: $Z_N^* = \{a \ \in Z_N : \mathrm{gcd}(\mathrm{a}, \mathrm{N}) = 1\}$ is a group.

- group operation is multiplication mod N.
- inverses exist and are easy to compute.
- the order of the group is $\phi(N) = (p-1)(q-1)$

The RSA Trapdoor Permutation

Today: Let e be an integer with $\gcd(e, \phi(N)) = 1$. Then, the map $F_{N,e}(x) = x^e \bmod N$ is a trapdoor permutation.

Key Fact: Given d such that $ed = 1 \bmod \phi(N)$, it is easy to compute x given x^e.

Proof: $(x^e)^d$

This gives us the RSA trapdoor permutation collection.

$$\{F_{N,e} : \gcd(e, N) = 1\}$$

Trapdoor for inversion: $d = e^{-1} \bmod \phi(N)$.

The RSA Trapdoor Permutation

Today: Let e be an integer with $\gcd(e, \phi(N)) = 1$. Then, the map $F_{N,e}(x) = x^e \bmod N$ is a trapdoor permutation.

Hardness of inversion without trapdoor = **RSA assumption**

given N, e (as above) and x^e mod N, hard to compute x.

We know that if factoring is easy, RSA is broken (and that's the only *known* way to break RSA)

Major Open Problem: Are factoring and RSA equivalent?

The RSA Trapdoor Permutation

<u>Today</u>: Let e be an integer with $\gcd(e, \phi(N)) = 1$. Then, the map $F_{N,e}(x) = x^e \bmod N$ is a trapdoor permutation.

Hardcore bits (galore) for the RSA trapdoor one-way perm:

- The Goldreich-Levin bit $\mathrm{GL}(r; r') = \langle r, r' \rangle \bmod 2$
- The least significant bit $\mathrm{LSB}(r)$
- The "most significant bit" $HALF_N(r) = 1$ iff $r < N/2$
- In fact, any single bit of r is hardcore.

RSA Encryption

- $Gen(1^n)$: Let $N = pq$ and (e, d) be such that $ed = 1 \; mod \; \phi(N)$.

  Let $pk = (N, e)$ and let $sk = d$.

- $Enc(pk, b)$ where b is a bit: Generate random $r \in Z_N^*$ and output $r^e \text{ mod } N$ and $\text{LSB}(r) \oplus m$.

- $Dec(sk, c)$: Recover r via RSA inversion.

<u>IND-secure under the RSA assumption</u>: given N, e (as above) and r^e mod N, hard to compute r.

Lectures 8-10

Constructions of Public-key Encryption

✅ Diffie-Hellman/El Gamal

✅ Trapdoor Permutations (RSA)

3: Quadratic Residuosity/Goldwasser-Micali

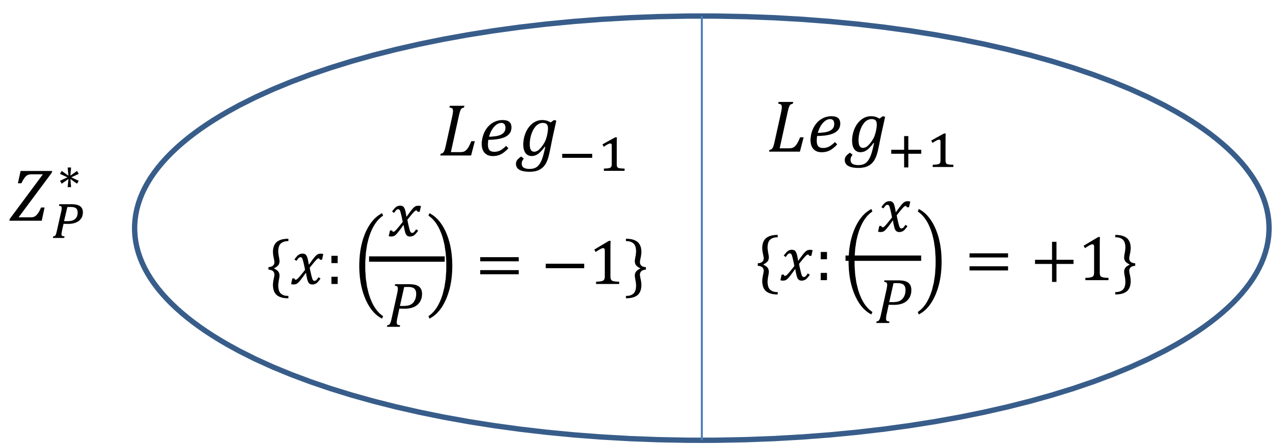
4: Post-Quantum Security & Lattice-based Encryption

Quadratic Residues mod P

Let P be prime. We saw that exactly half of Z_P^* are squares.

Define the Legendre Symbol $\left(\frac{x}{P}\right) = 1$ if x is a square, -1 if x is not a square, and 0 if x = 0 mod P.

So: $\left(\frac{x}{P}\right) = x^{(P-1)/2}$



Quadratic Residues mod P

Let P be prime. We saw that exactly half of Z_P^* are squares.

It is easy to compute square roots mod P. We will show it for the case where P = 3 (mod 4).

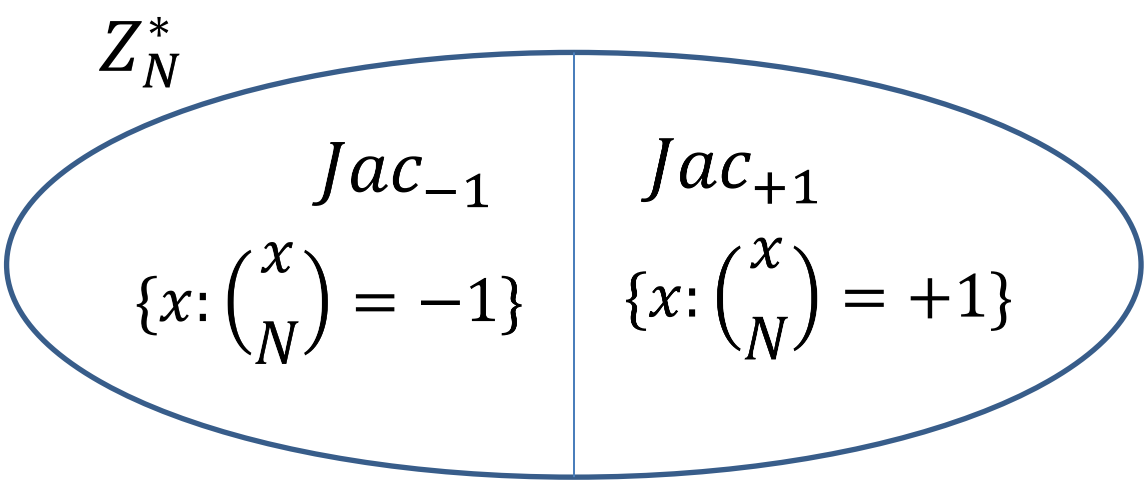
Claim: The square roots of x mod P are $\pm x^{(P+1)/4}$

Proof: $(\pm x^{(P+1)/4})^2 = x^{(P+1)/2} = x \cdot x^{(P-1)/2} = x \bmod P$

Quadratic Residues mod N

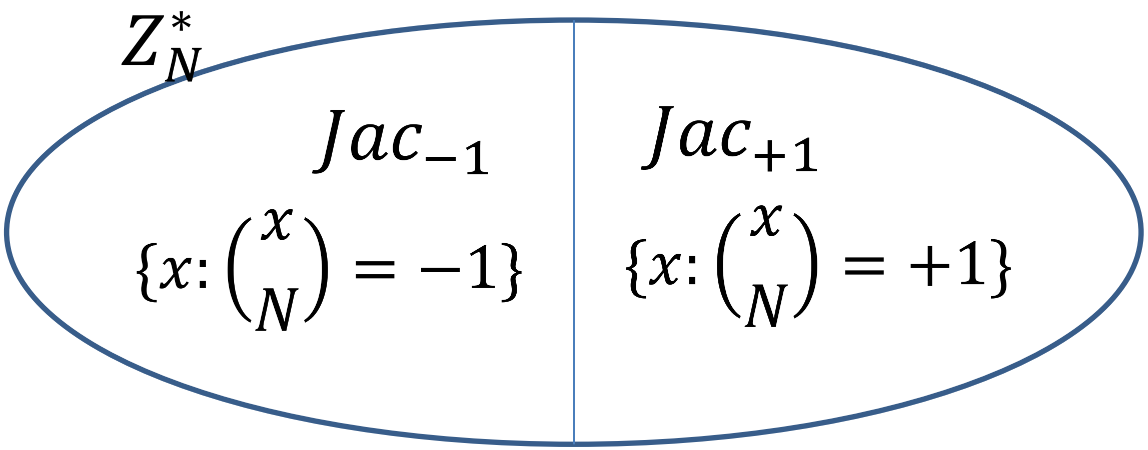
Now, let N = PQ be a product of two primes and look at Z_N^*

Define the Jacobi symbol $\left(\frac{x}{N}\right) = \left(\frac{x}{P}\right)\left(\frac{x}{Q}\right)$ to be +1 if x is a square mod both P and Q or a non-square mod both P and Q.



Quadratic Residues mod N

Let $N = PQ$ be a product of two large primes.



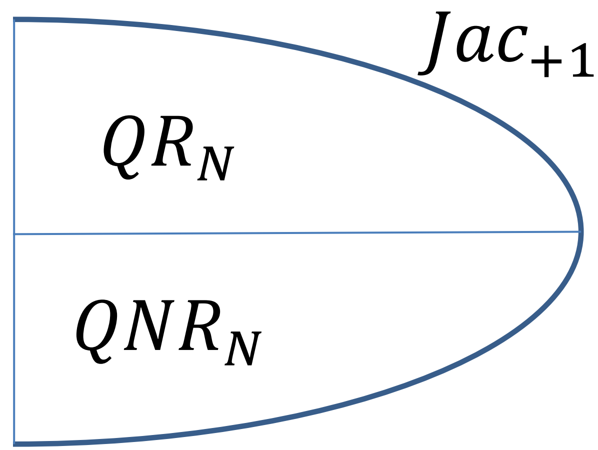
Surprising fact: Jacobi symbol $\left(\frac{x}{N}\right) = \left(\frac{x}{P}\right)\left(\frac{x}{Q}\right)$ is computable in poly time without knowing P and Q.

Quadratic Residues mod N

x is square mod N iff x is square mod P and it is a square mod Q.

So: $QR_N = \{x : \left(\frac{x}{P}\right) = \left(\frac{x}{Q}\right) = +1\}$

$QNR_N = \{x : \left(\frac{x}{P}\right) = \left(\frac{x}{Q}\right) = -1\}$



QR_N is the set of squares mod N and QNR_N is the set of non-squares mod N with Jacobi symbol +1.

Finding Square Roots Mod N

... is as hard as factoring N

$\Leftarrow$ Suppose you know P and Q and you want to find the square root of x mod N.

Find the square roots of y mod P and mod Q.

$$x = y_P^2 \bmod P \qquad\qquad x = y_Q^2 \bmod Q$$

Use the Chinese remainder theorem. Let $y = c_P y_P + c_Q y_Q$ where the CRT coefficients

$$c_P = 1\, mod\ P\ and\ c_P = 0\, mod\ Q$$
$$c_Q = 0\, mod\ P\ and\ c_Q = 1\, mod\ Q$$

Then y is a square root of x mod N.

Finding Square Roots Mod N

… is as hard as factoring N

Suppose you know P and Q and you want to find the square root of x mod N.

Find the square roots of y mod P and mod Q.

$$x = y_P^2 \bmod P \qquad\qquad x = y_Q^2 \bmod Q$$

Let $y = c_P y_P + c_Q y_Q$ where the CRT coefficients

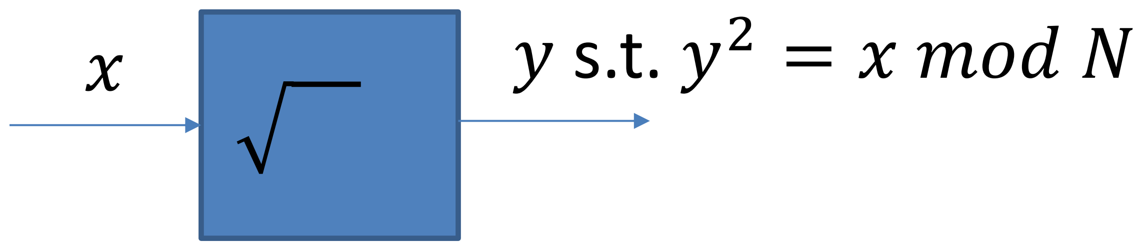
$$c_P = 1\, mod\ P\ and\ 0\ mod\ Q$$
$$c_Q = 0\, mod\ P\ and\ 1\ mod\ Q$$

So, if x is a square, it has 4 distinct square roots mod N.

Finding Square Roots Mod N

... is as hard as factoring N

⇒ Suppose you have a box that computes square roots mod N. Can we use it to factor N?



Feed the box $x = z^2 \; mod \; N$ for a random z.

Claim (Pf on the board): with probability 1/2, $\mathrm{gcd}(\mathrm{z} + \mathrm{y}, \mathrm{N})$ is a non-trivial factor of N.

Recognizing Squares mod N

… also seems hard

Let $N = PQ$ be a product of two large primes.

Quadratic Residuosity Assumption (QRA)

Let $N = PQ$ be a product of two large primes.
No PPT algorithm can distinguish between a random element of QR_N from a random element of QNR_N given only N.

Goldwasser-Micali (GM) Encryption

$Gen(1^n)$: Generate random n-bit primes p and q and let $N = pq$. Let $y \in QNR_N$ be some quadratic non-residue with Jacobi symbol +1.

Let $pk = (N, y)$ and let $sk = (p, q)$.

$Enc(pk, b)$ where b is a bit:
Generate random $r \in Z_N^*$ and output $r^2 \bmod N$ if $b = 0$ and $r^2 y \bmod N$ if $b = 1$.

$Dec(sk, c)$: Check if c $\in Z_N^*$ is a quadratic residue using p and q. If yes, output 0 else 1.

Goldwasser-Micali (GM) Encryption

$Enc(pk, b)$ where b is a bit:
Generate random $r \in Z_N^*$ and output $r^2 \bmod N$ if $b = 0$ and $r^2 y \bmod N$ if $b = 1$.

IND-security follows directly from the quadratic residuosity assumption.

GM is a Homomorphic Encryption

Given a GM-ciphertext of b and a GM-ciphertext of b', I can compute a GM-ciphertext of $b + b' mod\ 2$.

without knowing anything about $\boldsymbol{b}$ or $\boldsymbol{b'}$!

$Enc(pk, b)$ where b is a bit:
Generate random $r \in Z_N^*$ and output $r^2 y^b \bmod N$.

Claim: $Enc(pk, b) \cdot Enc(pk, b')$ is an encryption of $b \oplus b' = b + b' mod\ 2$.

Lectures 8-10

Constructions of Public-key Encryption

✅ Diffie-Hellman/El Gamal

✅ Trapdoor Permutations (RSA)

✅ Quadratic Residuosity/Goldwasser-Micali

4: Post-Quantum Security & Lattice-based Encryption

Solving Linear Equations

Solving *Noisy* Linear Equations

$$(\boldsymbol{s_1}|\boldsymbol{s_2})\begin{bmatrix}\mathbf{5} & \mathbf{1} & \mathbf{3}\\ \mathbf{6} & \mathbf{2} & \mathbf{1}\end{bmatrix} = [11 \quad 3 \quad 9]$$

Easy!

Find $(\boldsymbol{s_1}|\boldsymbol{s_2})$

How about:

$$(\boldsymbol{s_1}|\boldsymbol{s_2})\begin{bmatrix}\mathbf{5} & \mathbf{1} & \mathbf{3}\\ \mathbf{6} & \mathbf{2} & \mathbf{1}\end{bmatrix} + [e_1 \quad e_2 \quad e_3] = [11 \quad 3 \quad 9]$$

(e_1,e_2,e_3) are "small" numbers

Very hard!

in large dimensions

Find $\vec{s}$

Learning with Errors (LWE)

[Regev05, following BFKL93, Ale03]

very hard!

LWE: $(\mathbf{A}, s\mathbf{A}+e) \Rightarrow$ Find s

($\mathbf{A} \in Z_q^{nXm}$
$\mathbf{s} \in Z_q^n$ random “small” secret vector
$\boldsymbol{e} \in Z_q^n$: random “small” error vector)

Decisional LWE:

$$(\mathbf{A}, \mathrm{s}\mathbf{A}+e) \stackrel{c}{\approx} (\mathbf{A}, \mathbf{b})$$

(b uniformly random)

“Decisional LWE is as hard as LWE”.

Basic (Secret-key) Encryption

[Regev05]

n = security parameter, q = "small" prime

- Secret key sk = Uniformly random vector $\mathbf{s} \in Z_q^n$
- Encryption $\mathrm{Enc}_{\mathbf{s}}(m)$: // $m \in \{0,1\}$
  - Sample uniformly random $\mathbf{a} \in Z_q^n$, "short" noise $e \in Z$
  - The ciphertext **c = (a**, b = ⟨**a, s**⟩ + e + m)
- Decryption $\mathrm{Dec}_{sk}(\mathbf{c})$: Output $_2$(b − ⟨**a, s**⟩ mod q)

  // correctness as long as |e| < q/4

Basic (Secret-key) Encryption

[Regev05]

This is an incredibly cool scheme. In particular, additively homomorphic.

$$\boldsymbol{c} = (\mathbf{a},\ b = \langle \mathbf{a}, \mathbf{s} \rangle + e + m \lfloor q/2 \rfloor) \qquad +$$

$$\boldsymbol{c}' = (\mathbf{a}',\ b' = \langle \mathbf{a}', \mathbf{s} \rangle + e' + m' \lfloor q/2 \rfloor)$$

$$\boldsymbol{c} + \boldsymbol{c}' = (\mathbf{a}+\mathbf{a}',\ b+b' = \langle \mathbf{a}+\mathbf{a}', \mathbf{s} \rangle + (e+e') + (m+m') \lfloor q/2 \rfloor)$$

In words: $c + c'$ is an encryption of $m+m'$ (mod 2)

Public-key Encryption

[Regev05]

Here is a crazy idea. Public key has an encryption of 0 (call it c_0) and an encryption of 1 (call it c_1).
If you want to encrypt 0, output c_0 and if you want to encrypt 1, output c_1.

Well, turns out to be a crazy *bad* idea.

If only we could produce *fresh* encryptions of 0 or 1 given just the pk…

Public-key Encryption

[Regev05]

Here is another crazy idea.

Public key has *many* encryptions of 0 and an encryption of 1 (call it c_1).

If you want to encrypt 0, output a random linear combination of the 0-encryptions.

If you want to encrypt 1, output a random linear combination of the 0-encryptions plus c_1.

This one turns out to be a crazy ***good*** idea.

Public-key Encryption

[Regev05]

- Secret key sk = Uniformly random vector $\mathbf{s} \in Z_q^n$
- Public key pk: for $i\ from\ 1\ to\ k = poly(n)$

$$\left(\boldsymbol{c_0} = (\boldsymbol{a_0}, \langle \boldsymbol{a_0}, \boldsymbol{s} \rangle + e_0 + \left\lfloor \frac{q}{2} \right\rfloor), \boldsymbol{c_i} = (\boldsymbol{a_i}, \langle \boldsymbol{a_i}, \boldsymbol{s} \rangle + e_i)\right)$$

- Encrypting a bit m: pick k random bits $r_1, \ldots, r_k$

$$\sum_{i=1}^{k} r_i \boldsymbol{c_i} + m \cdot \boldsymbol{c_0}$$

Correctness: additive homomorphism

Security: decisional LWE + "Leftover Hash Lemma"

Practical Considerations

I want to encrypt to Bob. How do I know his public key?

Public-key Infrastructure: a directory of identities together with their public keys.

<u>Needs to be "authenticated":</u>

otherwise Eve could replace Bob's pk with her own.

Practical Considerations

Public-key encryption is orders of magnitude slower than secret-key encryption.

1. We mostly showed how to encrypt bit-by-bit! Super-duper inefficient.
2. Exponentiation takes $O(n^2)$ time as opposed to typically linear time for secret key encryption (AES).
3. The n itself is large for PKE (RSA: $n \geq 2048$) compared to SKE (AES: $n = 128$).

(For Elliptic Curve El-Gamal, it's 320 bits)

Can solve problem 1 and minimize problems 2&3 using **hybrid encryption**.

Hybrid Encryption

To encrypt a long message m (think 1 GB):

Pick a random key K (think 128 bits) for a secret-key encryption

Encrypt K with the PKE: $PKE.Enc(pk, K)$

Encrypt m with the SKE: $\mathrm{SKE}.Enc(K, m)$

To decrypt: recover K using sk. Then using K, recover m