MIT 6.875

Foundations of Cryptography
Lecture 9

Lectures 7-10

Constructions of Public-key Encryption

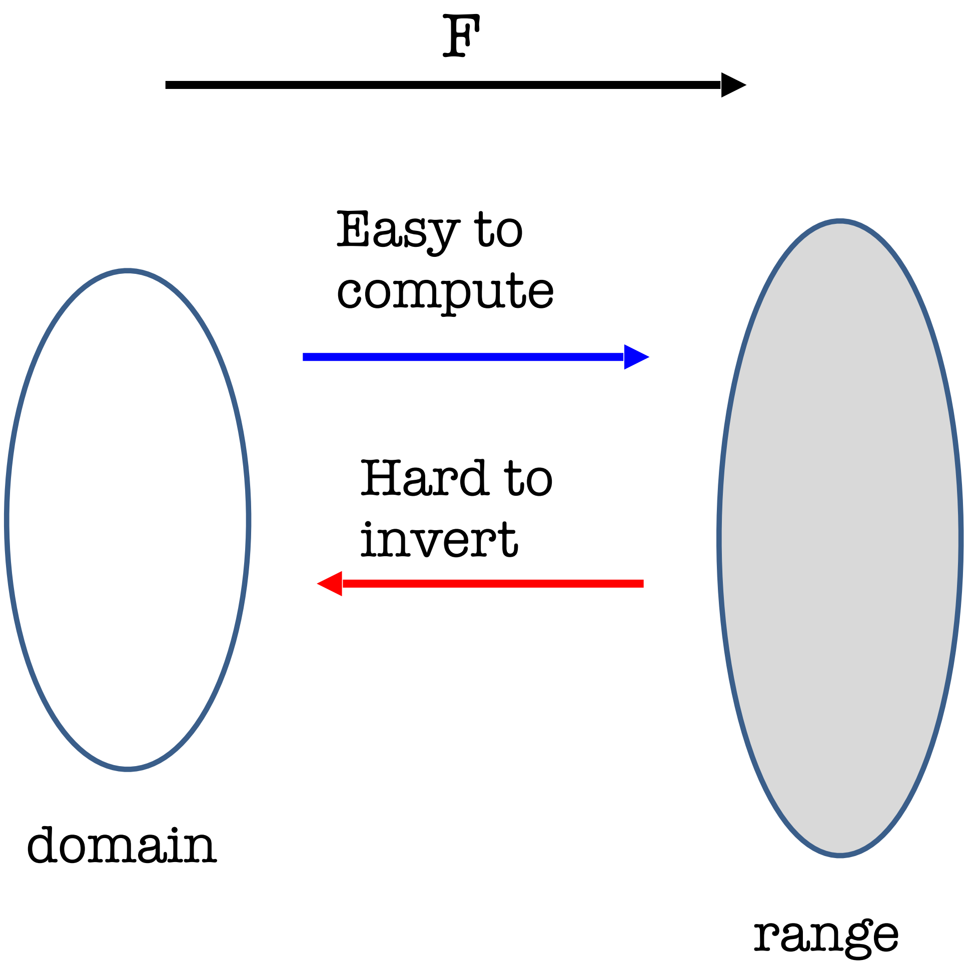
✅ Diffie-Hellman/El Gamal

2: Trapdoor Permutations (RSA)

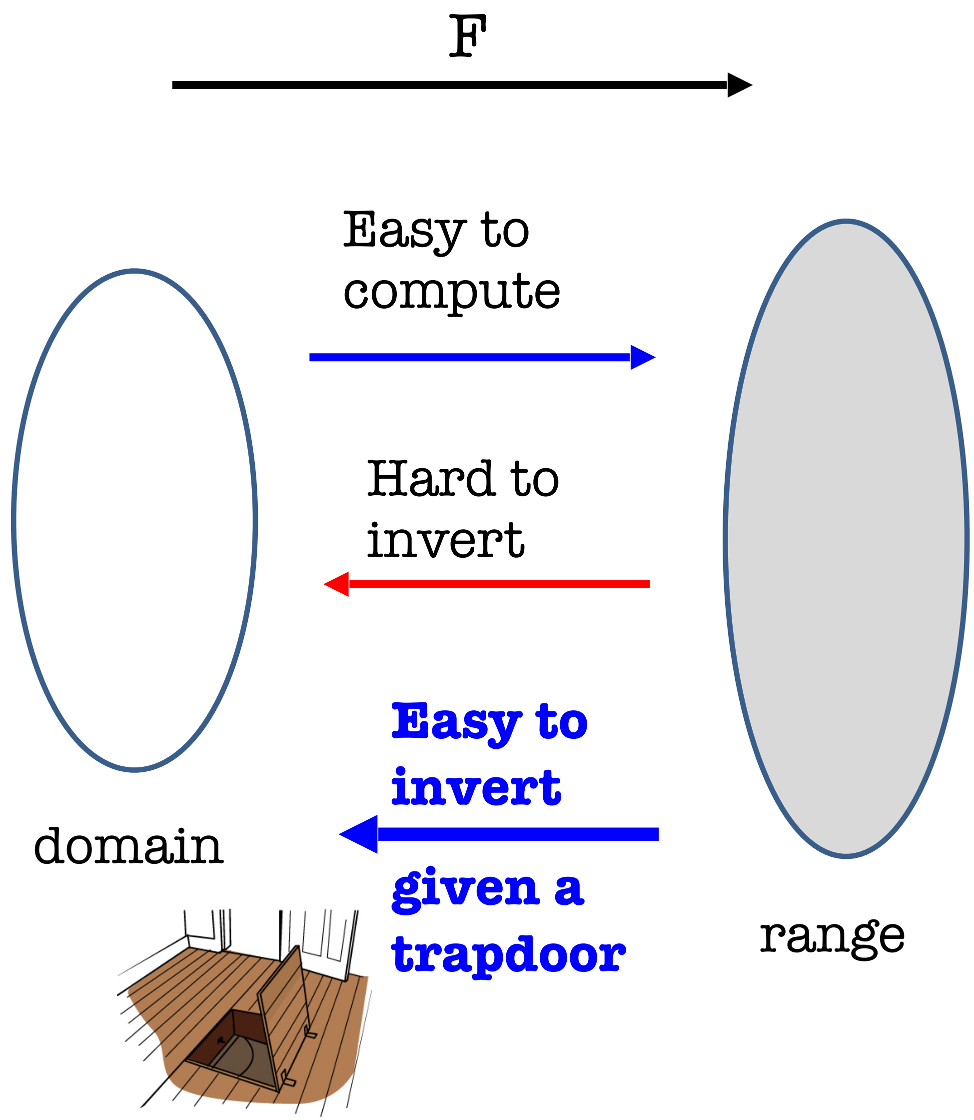
3: Quadratic Residuosity/Goldwasser-Micali

4: Post-Quantum Security & Lattice-based Encryption

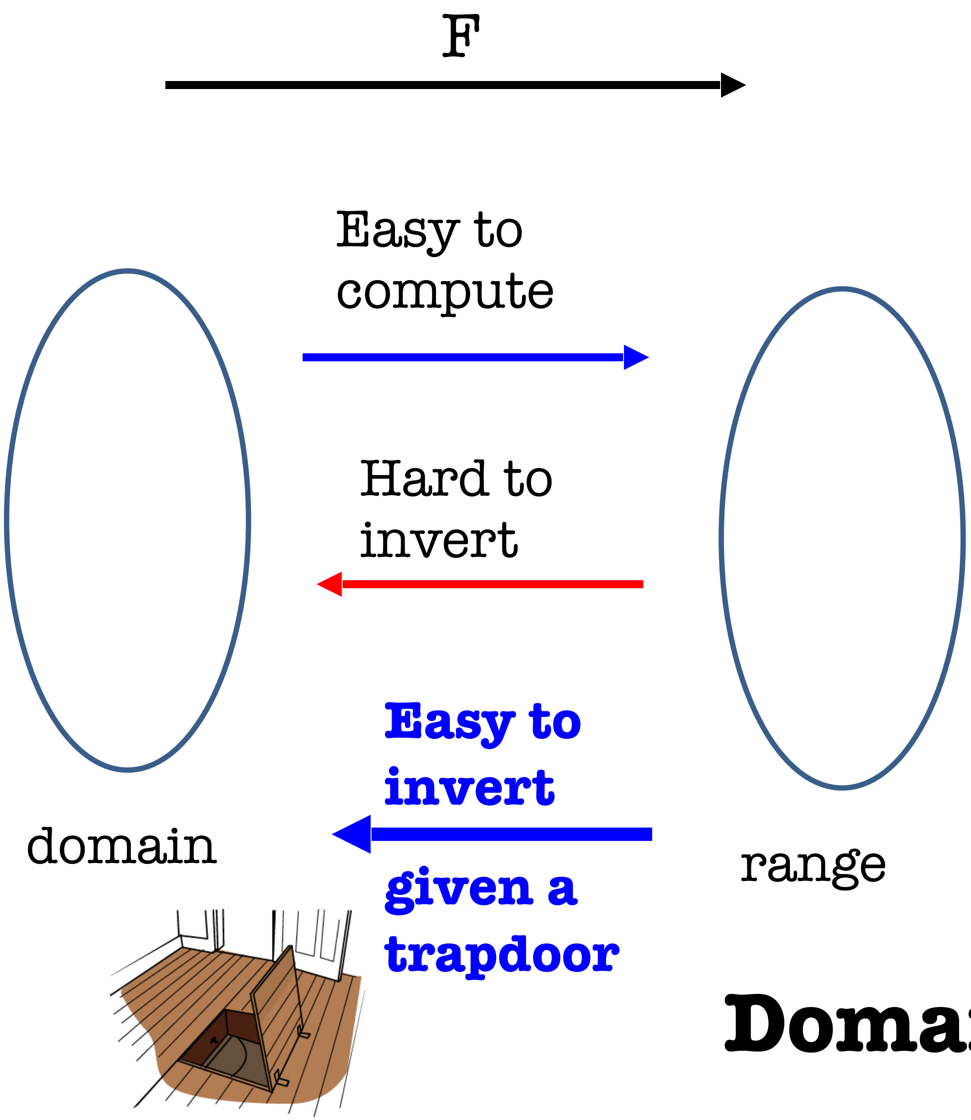
One-way Functions



Trapdoor One-way Functions



Trapdoor One-way Permutations



Domain = Range

Trapdoor Functions: The Definition

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: 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) $\mathscr{F} = \{\mathscr{F}_n\}_{n\in\mathbb{N}}$ where **each $\mathscr{F}_n$ is itself a collection of functions**

$\mathscr{F}_n = \{F_i: \{0,1\}^n \to \{0,1\}^{m(n)}\}_{i\in I_n}$ 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 $\mathcal{F}_n$ 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 .

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

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

From Trapdoor Permutations to IND-Secure Public-key Encryption

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

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

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

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

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

- $Enc\left(pk=i,m\right)$ where m is a bit: **Pick a random r .** **Output $c=(F_i(r), HCB(r) \bigoplus m)$.**

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

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 r . Output $c = (F_i(r), HCB(r) \bigoplus m)$.**

- $Dec(sk = t_i, c)$: 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

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 L8.

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

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

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

The RSA Trapdoor Permutation

Today: Let e be an integer with $\gcd\left(e, \phi(N)\right) = 1.$ 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.

The RSA Trapdoor Permutation

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

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

Proof: $(x^e)^d =$

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: $(x^e)^d = x^{k\phi(N)+1} =$

(for some integer k)

The RSA Trapdoor Permutation

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

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

Proof: $(x^e)^d = x^{k\phi(N)+1} = (x^{\phi(N)})^k \bullet x = x$

(for some integer k)

The RSA Trapdoor Permutation

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

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

Proof: $(x^e)^d = x^{k\phi(N)+1} = (x^{\phi(N)})^k \bullet x = x \bmod N$

(for some integer k)

The RSA Trapdoor Permutation

Today: Let e be an integer with $\gcd\left(e, \phi(N)\right) = 1$. 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: $(x^e)^d = x^{k\phi(N)+1} = (x^{\phi(N)})^k \bullet x = x \bmod N$

(for some integer k)

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

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

The RSA Trapdoor Permutation

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

The RSA Trapdoor Permutation

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

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

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

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 $r^e \bmod N$ and $\mathrm{LSB}(r) \bigoplus m$.

RSA Encryption

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

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

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

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

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 $r^e \bmod N$ and $\mathrm{LSB}(r) \bigoplus m$.

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

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

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

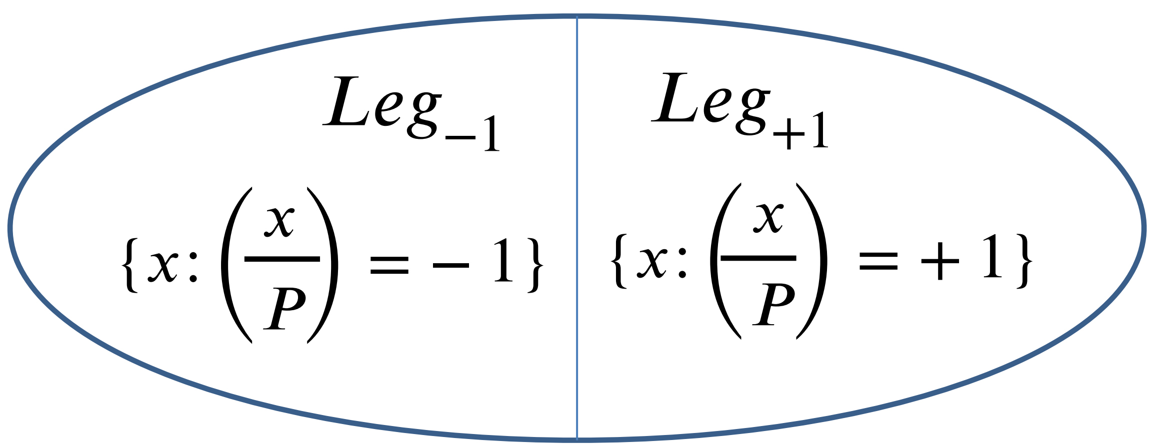
Quadratic Residues mod P

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.

Z_P^*



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

Z_P^*

Leg_{-1} $\quad$ Leg_{+1}

$\{x : \left(\frac{x}{P}\right) = -1\}$ $\quad$ $\{x : \left(\frac{x}{P}\right) = +1\}$

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).

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 $\pm\, x^{(P+1)/4}$

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

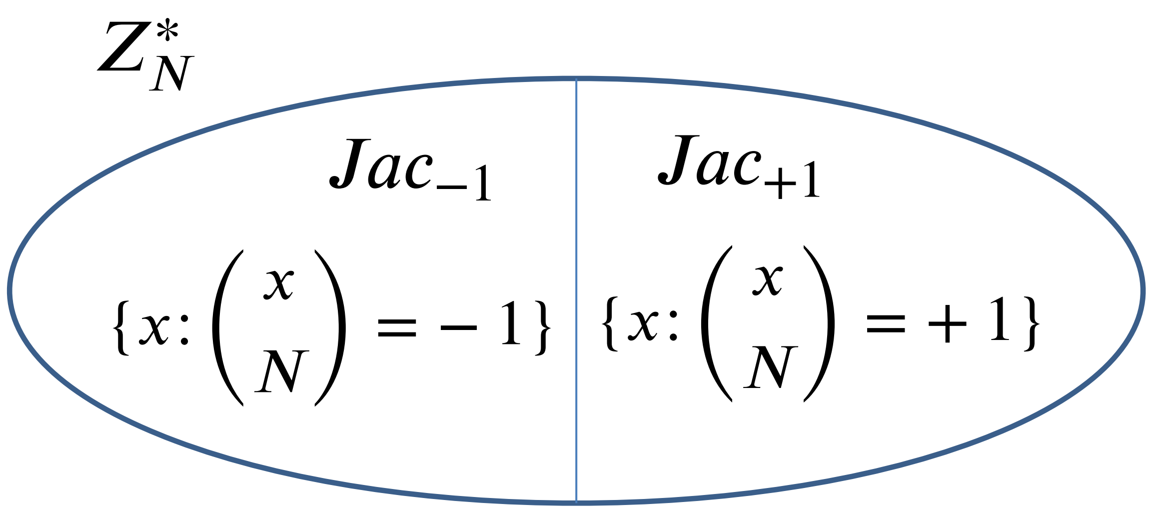
Quadratic Residues mod N

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

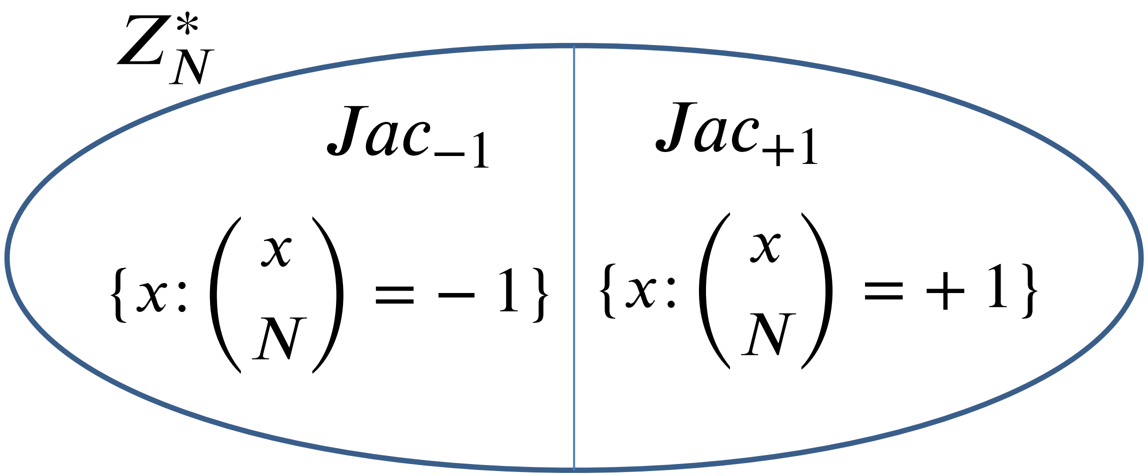
x is square mod N iff x is square mod P and it is a square mod Q.

Quadratic Residues mod 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.



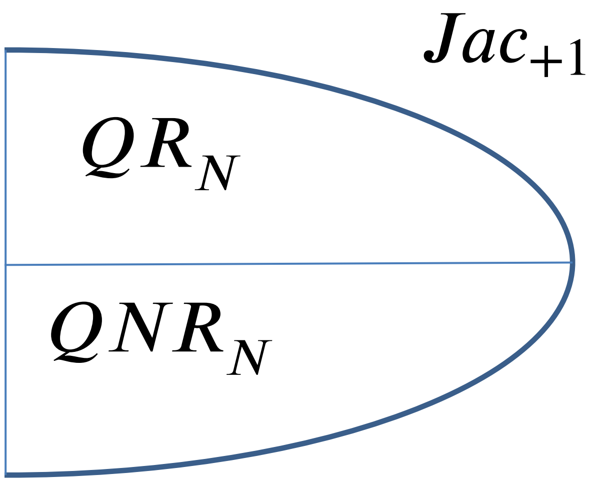
Quadratic Residues mod N



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.



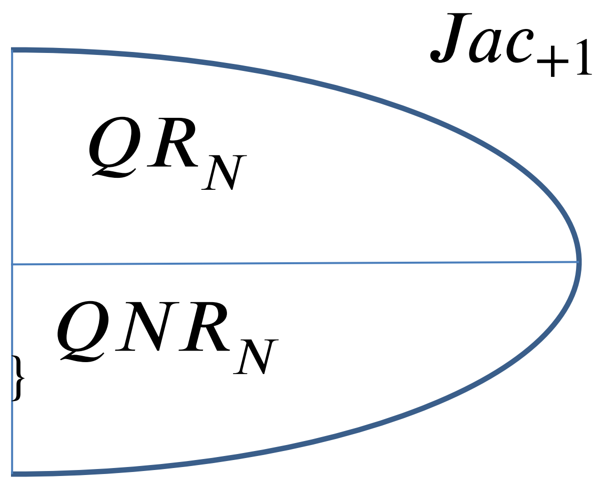
QR_N is the set of squares mod N and QNR_N is the set of non-squares mod N with Jacobi symbol +1.

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.

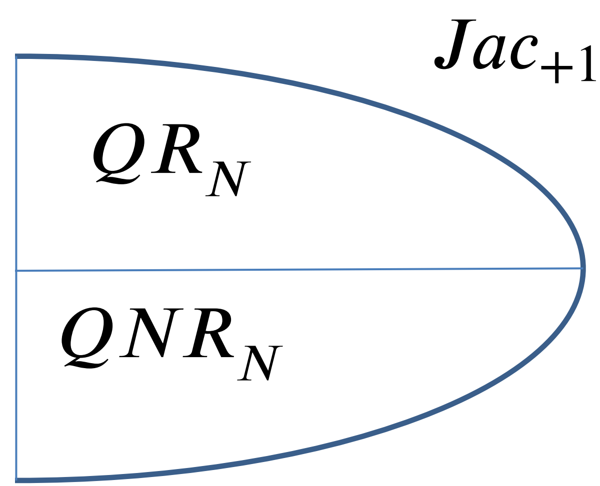
Recognizing Squares mod N

… seems hard

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

<u>Quadratic Residuosity Assumption (QRA)</u>

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 onl



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

$c_P = 1 \; mod \; P \; and \; 0 \; mod \; Q$

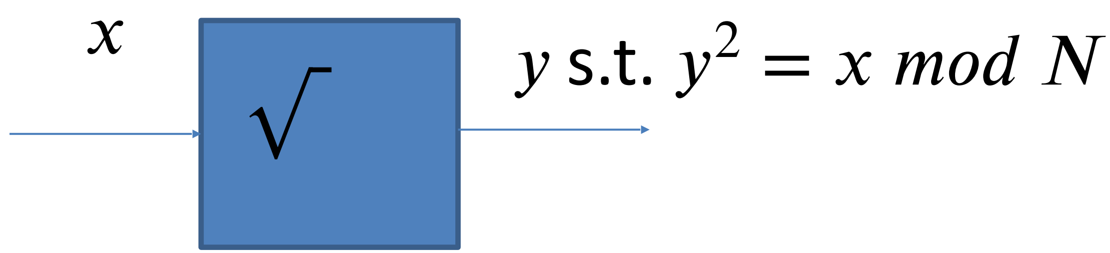
$$c_Q = 0 \; mod \; P \; and \; 1 \; mod \; Q$$

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

$\Rightarrow$ 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, $\gcd(z + y, N)$ is a non-trivial factor of 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 $sk = (p, q)$.

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 $b = 1$.

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 $b + b' mod\ 2$.

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 b or b'!

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 b or b'!

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

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 b or b'!

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

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