Public Key Cryptography

Dhananjoy Dey

Indian Institute of Information Technology, Lucknow
ddey@iiitl.ac.in

January 3, 2024

Disclaimers

1

All the pictures used in this presentation are taken from freely available websites.

2

If there is a reference on a slide all of the information on that slide is attributable to that source whether quotation marks are used or not.

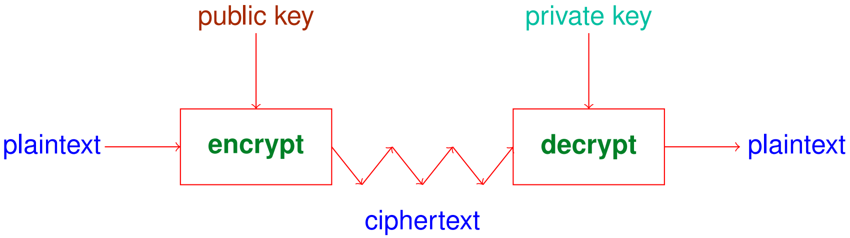
3

Any mention of commercial products or reference to commercial organizations is for information only; it does not imply recommendation or endorsement nor does it imply that the products mentioned are necessarily the best available for the purpose.

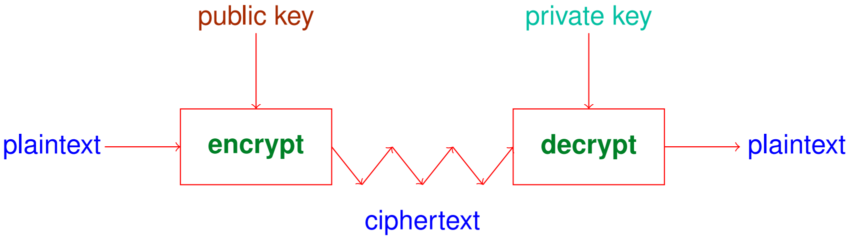
Outline

Outline

A Generic View of Public Key Crypto



A Generic View of Public Key Crypto

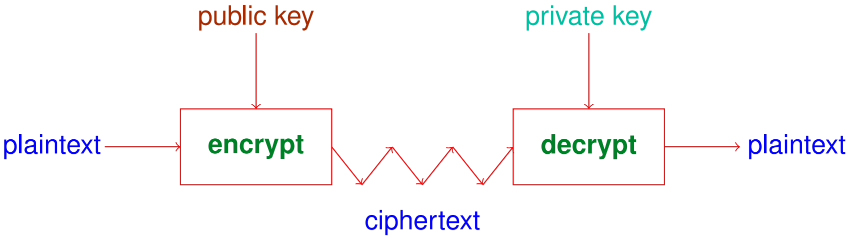


Advantages over symmetric-key

1. Better key distribution and management
   - No danger that public key compromised
2. New protocols
   - Digital Signature
3. Long-term encryption

Only disadvantage:

A Generic View of Public Key Crypto



Advantages over symmetric-key

1. Better key distribution and management
   - No danger that public key compromised
2. New protocols
   - Digital Signature
3. Long-term encryption

Only disadvantage: much more slower than symmetric key crypto

Definition

PKC

A public key cryptosystem is a pair of families $\{E_k : k \in \mathcal{K}\}$ and $\{D_k : k \in \mathcal{K}\}$ of algorithms representing invertible transformations,

$$E_k : \mathcal{M} \to \mathcal{C} \;\&\; D_k : \mathcal{C} \to \mathcal{M}$$

on a finite message space $\mathcal{M}$ and ciphertext space $\mathcal{C}$, such that

(i) for every $k \in \mathcal{K}$, D_k is the inverse of E_k and vice versa,

(ii) for every $k \in \mathcal{K}$, $M \in \mathcal{M}$ and $C \in \mathcal{C}$, the algorithms E_k and D_k are *easy* to compute.

(iii) for every $k \in \mathcal{K}$, it is feasible to compute inverse pairs E_k and D_k from k,

Definition

PKC

A public key cryptosystem is a pair of families $\{E_k : k \in \mathcal{K}\}$ and $\{D_k : k \in \mathcal{K}\}$ of algorithms representing invertible transformations,

$$E_k : \mathcal{M} \to \mathcal{C} \ \& \ D_k : \mathcal{C} \to \mathcal{M}$$

on a finite message space $\mathcal{M}$ and ciphertext space $\mathcal{C}$, such that

(i) for every $k \in \mathcal{K},\ D_k$ is the inverse of E_k and vice versa,

(ii) for every $k \in \mathcal{K},\ M \in \mathcal{M}$ and $C \in \mathcal{C}$, the algorithms E_k and D_k are *easy* to compute.

(iii) for every $k \in \mathcal{K}$, it is feasible to compute inverse pairs E_k and D_k from k,

(iv) for almost every $k \in \mathcal{K}$, each easily computed algorithm equivalent to D_k is computationally infeasible to derive from E_k, without knowing k.

Definition

Computationally Infeasible

A task is computationally infeasible if either the time taken or the memory required for carrying out the task is finite but impossibly large.

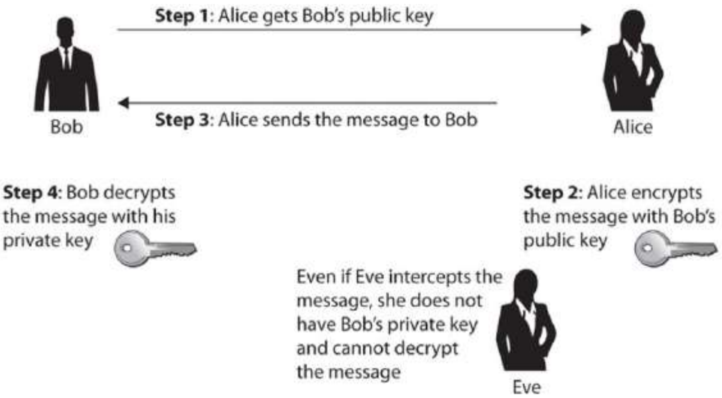
Definition

Computationally Infeasible

A task is computationally infeasible if either the time taken or the memory required for carrying out the task is finite but impossibly large.

Any computational task which takes $\geq 2^{112}$ bit operations, we say, it is computationally infeasible in present day scenario.

PKC


Step 1: Alice gets Bob's public key
Bob
Alice
Step 3: Alice sends the message to Bob
Step 4: Bob decrypts the message with his private key
Step 2: Alice encrypts the message with Bob's public key
Even if Eve intercepts the message, she does not have Bob's private key and cannot decrypt the message
Eve

Digital Signature

$\boxed{\textbf{Signing a Message } M}$

$\boxed{\text{Message } M}$

Digital Signature

Signing a Message M

$$\boxed{\text{Message } M} \xrightarrow{\text{Hash Function } h} \boxed{\text{Digest } h(M)}$$

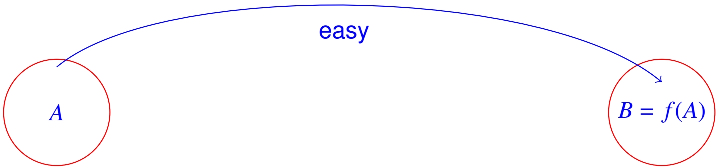
Digital Signature

$\boxed{\textbf{Signing a Message } M}$

$\boxed{\text{Message } M} \xrightarrow{\text{Hash Function } h} \boxed{\text{Digest } h(M)} \xrightarrow{\text{Private Key}} \boxed{\text{Signature}}$

Outline

One-way Function



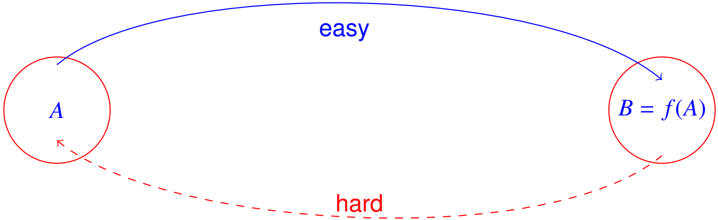
Definition

Easy: $\exists$ *a polynomial-time algorithm that, on input* $m \in A$ *outputs* $c = f(m)$.

Definition

Hard: *Every probabilistic polynomial-time algorithm trying, on input* $c (= f(m))$ *to find an inverse of* $c \in B$ *under* f, *may succeed only with negligible probability.*

One-way Function



Definition

Easy: $\exists$ *a polynomial-time algorithm that, on input* $m \in A$ *outputs* $c = f(m)$.

Definition

Hard: *Every probabilistic polynomial-time algorithm trying, on input* $c(= f(m))$ *to find an inverse of* $c \in B$ *under* f, *may succeed only with negligible probability.*

Examples of One-way Function

Examples of One-way Function

- Cryptographic hash functions, viz., SHA-2 and SHA-3 (Keccak) family.

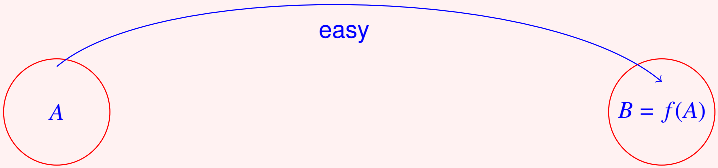
Examples of One-way Function

- Cryptographic hash functions, viz., SHA-2 and SHA-3 (Keccak) family.
- The function
$$f : \mathbb{Z}_p \to \mathbb{Z}_p,$$
$$x \mapsto x^{2^{24}+17} + a_1.x^{2^{24}+3} + a_2.x^3 + a_3.x^2 + a_4.x + a_5,$$
where $p = 2^{64} - 59$ and each $a_i\ (\in \mathbb{Z}_p)$ is 19-digit number for $1 \le i \le 5$.

Trapdoor One-way Function

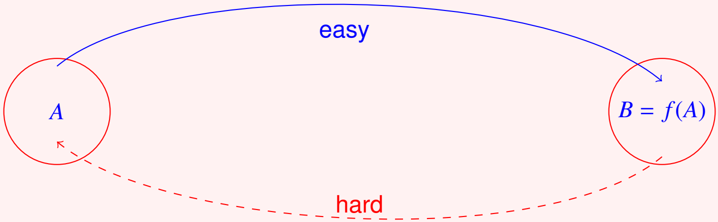
Trapdoor One-way Function

Trapdoor One-way Function



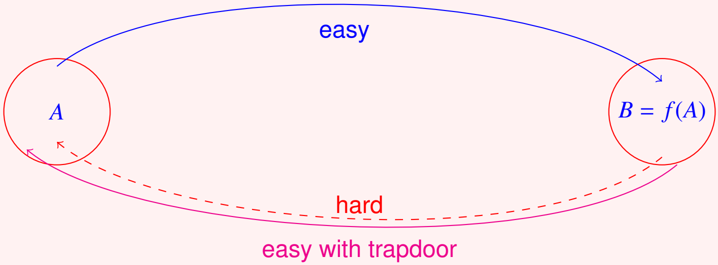
Trapdoor One-way Function

Trapdoor One-way Function



Trapdoor One-way Function

Trapdoor One-way Function



Trapdoor One-way Function

Definition

A trapdoor one-way function is a one-way function $f : \mathcal{M} \rightarrow \mathcal{C}$, satisfying the additional property that $\exists$ some additional information or trapdoor that makes it easy for a given $c \in f(\mathcal{M})$ to find out $m \in \mathcal{M} : f(m) = c$, but without the trapdoor this task becomes hard.

Examples Trapdoor One-way Function

- **Integer Factorization:** Given $n \in \mathbb{Z}^+$, find $n = p_1^{e_1} p_2^{e_2} \ldots p_k^{e_k}$ where the p_i are pairwise distinct primes and each $e_i \geq 0$ for $1 \leq i \leq k$. $\rightarrow$ **hard problem**.

Examples Trapdoor One-way Function

- **Integer Factorization:** Given $n \in \mathbb{Z}^+$, find $n = p_1^{e_1} p_2^{e_2} \dots p_k^{e_k}$ where the p_i are pairwise distinct primes and each $e_i \geq 0$ for $1 \leq i \leq k$. $\rightarrow$ **hard problem**.

$$IFP \stackrel{def}{=} \begin{cases} Input & : \quad n > 1 \\ Output & : \quad p_1^{e_1} p_2^{e_2} \dots p_k^{e_k} \end{cases}$$

Example

- Consider the number 37015031

Examples Trapdoor One-way Function

- **Integer Factorization:** Given $n \in \mathbb{Z}^+$, find $n = p_1^{e_1} p_2^{e_2} \ldots p_k^{e_k}$ where the p_i are pairwise distinct primes and each $e_i \geq 0$ for $1 \leq i \leq k$. $\rightarrow$ **hard problem**.

$$IFP \stackrel{def}{=} \begin{cases} Input & : \quad n > 1 \\ Output & : \quad p_1^{e_1} p_2^{e_2} \ldots p_k^{e_k} \end{cases}$$

Example

- Consider the number 37015031$= 6079 \times 6089$

Examples Trapdoor One-way Function

- **Integer Factorization:** Given $n \in \mathbb{Z}^+$, find $n = p_1^{e_1} p_2^{e_2} \ldots p_k^{e_k}$ where the p_i are pairwise distinct primes and each $e_i \geq 0$ for $1 \leq i \leq k$. $\rightarrow$ **hard problem**.

$$IFP \stackrel{def}{=} \begin{cases} Input & : \quad n > 1 \\ Output & : \quad p_1^{e_1} p_2^{e_2} \ldots p_k^{e_k} \end{cases}$$

Example

- Consider the number 37015031$= 6079 \times 6089$
- Consider the number 96679789

Examples Trapdoor One-way Function

- **Integer Factorization:** Given $n \in \mathbb{Z}^+$, find $n = p_1^{e_1} p_2^{e_2} \dots p_k^{e_k}$ where the p_i are pairwise distinct primes and each $e_i \geq 0$ for $1 \leq i \leq k$. $\rightarrow$ **hard problem**.

$$IFP \stackrel{def}{=} \begin{cases} Input & : \quad n > 1 \\ Output & : \quad p_1^{e_1} p_2^{e_2} \dots p_k^{e_k} \end{cases}$$

Example

- Consider the number 37015031$= 6079 \times 6089$
- Consider the number 96679789$= 9743 \times 9923$

Examples Trapdoor One-way Function

- **Discrete Logarithm Problem:** Given an abelian group $(G, \ .)$ and $g \in G$ of order n. Given $h \in G$ such that $h = g^x$ find x ($DLP(g, h) \rightarrow x$). $\rightarrow$ **hard problem**.

Examples Trapdoor One-way Function

- **Discrete Logarithm Problem:** Given an abelian group $(G, \ .)$ and $g \in G$ of order n. Given $h \in G$ such that $h = g^x$ find x $(DLP(g,h) \to x)$. $\to$ **hard problem**.
  The DLP over the multiplicative group $\mathbb{Z}_n^* = \{a \ : \ 1 \leq a \leq n, \gcd(a,n) = 1\}$. DLP may be defined as follows:

$$DLP \stackrel{def}{=} \begin{cases} Input & : \quad x, y \in \mathbb{Z}_n^* \ \& \ n \\ Output & : \quad k \ s/t \ y \equiv x^k \mod n \end{cases}$$

Examples Trapdoor One-way Function

- **Discrete Logarithm Problem:** Given an abelian group $(G, \ .)$ and $g \in G$ of order n. Given $h \in G$ such that $h = g^x$ find x ($DLP(g, h) \to x$). $\to$ **hard problem**.
  The DLP over the multiplicative group $\mathbb{Z}_n^* = \{a \ : \ 1 \leq a \leq n, \gcd(a, n) = 1\}$. DLP may be defined as follows:

$$DLP \stackrel{def}{=} \begin{cases} Input & : \quad x, y \in \mathbb{Z}_n^* \ \& \ n \\ Output & : \quad k \ s/t \ y \equiv x^k \mod n \end{cases}$$

Example

- Let $p = 97$. Then $\mathbb{Z}_{97}^*$ is a cyclic group of order $n = 96$.
  5 is a generator of $\mathbb{Z}_{97}^*$.
  Now, $5^x \equiv 35 \mod 97$, find the value of x.

Examples Trapdoor One-way Function

- **Discrete Logarithm Problem:** Given an abelian group $(G, \ .)$ and $g \in G$ of order n. Given $h \in G$ such that $h = g^x$ find x $(DLP(g,h) \rightarrow x)$. $\rightarrow$ **hard problem**.
  The DLP over the multiplicative group $\mathbb{Z}_n^* = \{a \ : \ 1 \leq a \leq n, \gcd(a,n) = 1\}$. DLP may be defined as follows:

$$DLP \stackrel{def}{=} \begin{cases} Input & : \quad x, y \in \mathbb{Z}_n^* \ \& \ n \\ Output & : \quad k \ s/t \ y \equiv x^k \mod n \end{cases}$$

Example

- Let $p = 97$. Then $\mathbb{Z}_{97}^*$ is a cyclic group of order $n = 96$.
  5 is a generator of $\mathbb{Z}_{97}^*$.
  Now, $5^x \equiv 35 \mod 97$, find the value of x.

Example Trapdoor One-way Function

- **Computational Diffie-Hellman Problem:** Given $a = g^x$ and $b = g^y$ find $c = g^{xy}$. ($CDH(g, a, b) \rightarrow c$). $\rightarrow$ **hard problem**.

Example Trapdoor One-way Function

- **Computational Diffie-Hellman Problem:** Given $a = g^x$ and $b = g^y$ find $c = g^{xy}$. ($CDH(g, a, b) \rightarrow c$). $\rightarrow$ **hard problem**.
- **Elliptic Curve Discrete Logarithm Problem (ECDLP):** $\mathbb{E}$ denotes the collections of points on a elliptic curve and $P \in \mathbb{E}$. Let $\mathcal{S}$ be the cyclic subgroup of $\mathbb{E}$ generated by P. Given $Q \in \mathcal{S}$, find an integer x such that $Q = x.P$. $\rightarrow$ **hard problem**.

Outline

DH Key Exchange

DH Key Exchange

DH Key Exchange

- k is the shared secret key.

DH Key Exchange

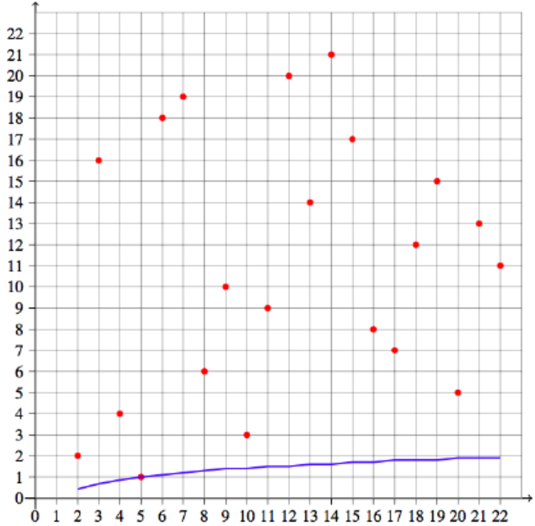
- k is the shared secret key.
- Knowing $g, \ g^a \ \& \ g^b$, it is hard to find g^{ab}.
- **Idea of this protocol:** The enciphering key can be made public since it is computationally infeasible to obtain the deciphering key from enciphering key.
- This protocol was (supposed to be) the door-opener to PKC.

DH Key Exchange

- k is the shared secret key.
- Knowing $g, \ g^a \ \& \ g^b$, it is hard to find g^{ab}.
- **Idea of this protocol:** The enciphering key can be made public since it is computationally infeasible to obtain the deciphering key from enciphering key.
- This protocol was (supposed to be) the door-opener to PKC.
- PKCS #3 (Version 1.4): Diffie-Hellman Key-Agreement Standard, An RSA Laboratories Technical Note – Revised November 1, 1993.

Discrete Logarithm $\mod 23$ to the Base 5

Discrete Logarithm $\bmod 23$ to the Base 5



- Clifford Cocks, Malcolm Williamson & James Ellis developed Nonsecret Encryption between 1969 and 1974.

Clifford Cocks, Malcolm Williamson, and James Ellis.

- All were at GCHQ, so this stayed secret until 1997.

Nonsecret Encryption

Key Generation

1. Select 2 large distinct primes p & q such that $p \nmid (q-1)$ and $q \nmid (p-1)$.

   Public key: $n = pq$.

2. Find numbers r & s, s/t $p.r \equiv 1 \mod (q-1)$ and $q.s \equiv 1 \mod (p-1)$.
3. Find u & v, s/t $u.p \equiv 1 \mod q$ and $v.q \equiv 1 \mod p$.

   Private key: (p, q, r, s, u, v).

Nonsecret Encryption

Encryption

$$C \equiv M^n \mod n \text{ for } 0 \leq M < n.$$

Decryption

1. $a \equiv C^s \mod p$ and $b \equiv C^r \mod q$.
2. $M \equiv a.q.v + b.p.u \mod n$.

Outline

RSA Key Generation

- Generate two large distinct random primes p & q.
- Compute $n = pq$ and $\phi(n) = (p-1)(q-1)$.
- Select a random integer $e,\ 1 < e < \phi(n)$ s/t $\gcd(e, \phi(n)) = 1$.

RSA Key Generation

- Generate two large distinct random primes p & q.
- Compute $n = pq$ and $\phi(n) = (p-1)(q-1)$.
- Select a random integer $e,\ 1 < e < \phi(n)$ s/t $\gcd(e, \phi(n)) = 1$.
- Compute the unique integer $d,\ 1 < d < \phi(n)$ s/t

$$ed \equiv 1 \mod \phi(n).$$

Public key is (n, e); Private key is (p, q, d).

RSA Encryption/Decryption

Encryption:

$$c \equiv m^e \mod n,$$

Plaintext m and ciphertext $c \in \mathbb{Z}_n$.

Decryption:

$$m' \equiv c^d \mod n.$$

RSA Encryption/Decryption

Encryption:

$$c \equiv m^e \mod n,$$

Plaintext m and ciphertext $c \in \mathbb{Z}_n$.

Decryption:

$$m' \equiv c^d \mod n.$$

PKCS #1 v2.2: RSA Cryptography Standard, RSA Laboratories – October 27, 2012.

RSA Validation

SBI Public Key Information

Public Key Info

Algorithm	RSA
Key Size	2048
Exponent	65537
Modulus	A6:55:7F:B2:9C:23:FC:79:F8:9D:90:F6:75:4E:CE:3A:26:90:B8:37:EA:8E:6E: D6:18:8A:FC:F6:CA:7C:6F:4B:45:4D:98:DE:4F:3D:A3:78:5E:0C:4A:1A:81:8D: 6F:C3:BB:4C:38:6E:04:0B:1F:BB:CB:50:8B:42:E9:E2:17:65:E2:C0:D0:CA:F4: E5:C6:0A:C9:47:53:32:15:69:F6:C4:EC:B0:E0:B0:FC:CB:BA:DE:DF:BE:ED:2 B:44:3D:F6:2B:B3:0A:CA:B8:FC:D1:5F:84:2C:34:1E:15:52:76:4E:90:FA:85:7 0:BB:05:C3:02:03:17:74:B3:80:A1:59:1F:19:7B:3A:2B:C3:D5:59:CF:BA:5D:B E:DF:3B:3A:8E:52:C1:D3:A3:8C:06:D2:2A:98:2F:4D:82:7F:28:F1:B1:D3:71:7 E:CF:4C:B1:26:F4:6F:EA:09:F9:7F:5A:D6:15:46:5C:92:50:D4:F4:F3:CA:60:2 5:4D:9A:66:91:1D:EA:74:D4:B1:71:D9:30:15:4C:BB:B6:CD:C6:18:82:F8:B7:4 8:97:AF:2F:22:15:94:FE:EB:E7:DE:EF:CA:A3:6E:CC:26:69:D5:92:5B:68:89:5 6:2B:B3:72:60:62:49:8B:C5:59:45:43:C1:F4:7E:8F:2B:C4:DD:C1:BB:39:D4:B C:5C:51:53

Strong Prime Number

Definition

A prime p *is called a strong prime if*

Strong Prime Number

Definition

A prime p is called a strong prime if

(i) $p-1$ *has a large prime factor, say* r*,*

(ii) $p+1$ *has a large prime factor, and*

(iii) $r-1$ *has a large prime factor.*

Definition

For $n \geq 1$, let $\phi(n)$ denote the number of integers in the interval $[1,\ n]$ which are relatively prime to n. The function ϕ is called the ***Euler phi function****.*

Definition

For $n \geq 1$, let $\phi(n)$ denote the number of integers in the interval $[1, \ n]$ which are relatively prime to n. The function ϕ is called the ***Euler phi function****.*

Properties of Euler phi function

i. If p is a prime, then $\phi(p) = p - 1$.

Definition

For $n \geq 1$, let $\phi(n)$ denote the number of integers in the interval $[1,\ n]$ which are relatively prime to n. The function ϕ is called the ***Euler phi function****.*

Properties of Euler phi function

i. If p is a prime, then $\phi(p) = p - 1$.

ii. The Euler phi function is multiplicative. That is, if $gcd(m, n) = 1$, then

$$\phi(mn) = \phi(m)\phi(n).$$

Definition

For $n \geq 1$, let $\phi(n)$ denote the number of integers in the interval $[1,\ n]$ which are relatively prime to n. The function ϕ is called the ***Euler phi function****.*

Properties of Euler phi function

i. If p is a prime, then $\phi(p) = p - 1$.

ii. The Euler phi function is multiplicative. That is, if $gcd(m, n) = 1$, then

$$\phi(mn) = \phi(m)\phi(n).$$

iii. If $n = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}$, is the prime factorization of n, then

$$\phi(n) = n \left(1 - \frac{1}{p_1}\right)\left(1 - \frac{1}{p_2}\right)\cdots\left(1 - \frac{1}{p_k}\right).$$

Modular Arithmetic

- The multiplicative group of $\mathbb{Z}_n$ is $\mathbb{Z}_n^* = \{a \in \mathbb{Z}_n \ : \ gcd(a,n) = 1\}$.

Modular Arithmetic

- The multiplicative group of $\mathbb{Z}_n$ is $\mathbb{Z}_n^* = \{a \in \mathbb{Z}_n \ : \ gcd(a,n) = 1\}$.
- **Fermat's theorem:** If $gcd(a,p) = 1$, for a prime p then $a^{p-1} \equiv 1 \ mod \ p$.

Modular Arithmetic

- The multiplicative group of $\mathbb{Z}_n$ is $\mathbb{Z}_n^* = \{a \in \mathbb{Z}_n \ : \ gcd(a,n) = 1\}$.
- **Fermat's theorem:** If $gcd(a,p) = 1$, for a prime p then $a^{p-1} \equiv 1 \ mod \ p$.
- Let n be an odd composite integer. An integer $a, \ 1 \leq a \leq n-1, \ni \ a^{n-1} \not\equiv 1 \mod n$ is called a **Fermat witness** (to compositeness) for n.

Modular Arithmetic

- The multiplicative group of $\mathbb{Z}_n$ is $\mathbb{Z}_n^* = \{a \in \mathbb{Z}_n \;:\; gcd(a,n) = 1\}$.
- **Fermat's theorem:** If $gcd(a,p) = 1$, for a prime p then $a^{p-1} \;\equiv\; 1 \; mod \; p$.
- Let n be an odd composite integer. An integer $a,\; 1 \leq a \leq n-1, \ni \; a^{n-1} \not\equiv 1 \mod n$ is called a **Fermat witness** (to compositeness) for n.
- **Euler's theorem:** If $a \in \mathbb{Z}_n^*$, then

$$a^{\phi(n)} \;\equiv\; 1 \; mod \; n.$$

Pseudoprime

Definition

*If n is an odd composite number and b is an integer s/t $\gcd(n,b)=1$ and $b^{n-1}\equiv 1 \mod n$ then n is called a **pseudoprime** to the base b. The integer b is called a **Fermat liar** (to primality) for n.*

Pseudoprime

Definition

*If n is an odd composite number and b is an integer s/t $\gcd(n, b) = 1$ and $b^{n-1} \equiv 1 \mod n$ then n is called a **pseudoprime** to the base b. The integer b is called a **Fermat liar** (to primality) for n.*

Example

1. The number $n = 91$ is a pseudoprime to the base $b = 3$,

Pseudoprime

Definition

*If n is an odd composite number and b is an integer s/t $\gcd(n, b) = 1$ and $b^{n-1} \equiv 1 \mod n$ then n is called a **pseudoprime** to the base b. The integer b is called a **Fermat liar** (to primality) for n.*

Example

1. The number $n = 91$ is a pseudoprime to the base $b = 3$,

$$\because 3^{90} \equiv 1 \mod 91.$$

Pseudoprime

Definition

If n is an odd composite number and b is an integer s/t $\gcd(n,b) = 1$ and $b^{n-1} \equiv 1 \mod n$ then n is called a ***pseudoprime*** *to the base b. The integer b is called a* ***Fermat liar*** *(to primality) for n.*

Example

1. The number $n = 91$ is a pseudoprime to the base $b = 3$,

$$\because 3^{90} \equiv 1 \mod 91.$$

2. However, 91 is not a pseudoprime to the base 2,

$$\because 2^{90} \equiv$$

3. The composite integer $n = 341 (= 11 \times 31)$ is a pseudoprime to the base $2, \because 2^{340} \equiv 1 \mod 341$.

Carmichael Number

Definition

A Carmichael number is a composite integer n *s/t*

$$b^{n-1} \equiv 1 \mod n,$$

for every $b \in \mathbb{Z}_n^*$.

Carmichael Number

Definition

A Carmichael number is a composite integer n s/t

$$b^{n-1} \equiv 1 \mod n,$$

for every $b \in \mathbb{Z}_n^*$.

Example

1. $n = 561 = 3 \times 11 \times 17$ is a Carmichael number. This is the smallest Carmichael number.

Carmichael Number

Definition

A *Carmichael number* *is a composite integer* n *s/t*

$$b^{n-1} \equiv 1 \mod n,$$

for every $b \in \mathbb{Z}_n^*$.

Example

1. $n = 561 = 3 \times 11 \times 17$ is a Carmichael number. This is the smallest Carmichael number.
2. The following are Carmichael numbers:
   (a) $1105 = 5 \times 13 \times 17$
   (b) $1729 = 7 \times 13 \times 19$
   (c) $2465 = 5 \times 17 \times 29$

Carmichael Number

- A composite integer n is a Carmichael number iff the following two conditions are satisfied:
  (i) n is square-free, and
  (ii) $p-1$ divides $n-1$ for every prime divisor p of n.

Carmichael Number

- A composite integer n is a Carmichael number iff the following two conditions are satisfied:
  (i) n is square-free, and
  (ii) $p-1$ divides $n-1$ for every prime divisor p of n.
- A Carmichael number must be the product of at least three distinct primes.
- There are an infinite number of Carmichael numbers.

Quadratic Residue

Definition

*Let $a \in \mathbb{Z}_n^*$; a is said to be a **quadratic residue** modulo n, if*

$$\exists\, x \in \mathbb{Z}_n^* \ni x^2 \equiv a \mod n.$$

*If no such x exists, then a is called a **quadratic nonresidue** modulo n.*

The set of all quadratic residues modulo n is denoted by Q_n and the set of all quadratic nonresidues is denoted by $\overline{Q_n}$.

Quadratic Residue

Definition

Let $a \in \mathbb{Z}_n^*$; a is said to be a ***quadratic residue*** *modulo n, if*

$$\exists\, x \in \mathbb{Z}_n^* \ni x^2 \equiv a \mod n.$$

If no such x exists, then a is called a ***quadratic nonresidue*** *modulo n.*

The set of all quadratic residues modulo n is denoted by Q_n and the set of all quadratic nonresidues is denoted by $\overline{Q_n}$.

- Let p be an odd prime and let α be a generator of $\mathbb{Z}_p^*$. Then $a \in \mathbb{Z}_p^*$ is a quadratic residue modulo $p \Leftrightarrow a \equiv \alpha^i \mod p$, where i is an even integer.

Quadratic Residue

Definition

Let $a \in \mathbb{Z}_n^*$; a is said to be a ***quadratic residue*** *modulo n, if*

$$\exists\, x \in \mathbb{Z}_n^* \ni x^2 \equiv a \mod n.$$

If no such x exists, then a is called a ***quadratic nonresidue*** *modulo n.*

The set of all quadratic residues modulo n is denoted by Q_n and the set of all quadratic nonresidues is denoted by $\overline{Q_n}$.

- Let p be an odd prime and let α be a generator of $\mathbb{Z}_p^*$. Then $a \in \mathbb{Z}_p^*$ is a quadratic residue modulo $p \Leftrightarrow a \equiv \alpha^i \mod p$, where i is an even integer.
- It follows that $\#Q_p = \frac{p-1}{2}$ and $\#\overline{Q_p} = \frac{p-1}{2}$.

Quadratic Residue

Example

$\alpha = 6$ is a generator of $\mathbb{Z}_{13}^*$. The powers of α are

i	0	1	2	3	4	5	6	7	8	9	10	11
$\alpha^i \mod 13$	1	6	10	8	9	2	12	7	3	5	4	11

Quadratic Residue

Example

$\alpha = 6$ is a generator of $\mathbb{Z}_{13}^*$. The powers of α are

i	0	1	2	3	4	5	6	7	8	9	10	11
$\alpha^i \mod 13$	1	6	10	8	9	2	12	7	3	5	4	11

Hence $Q_{13} = \{1, 3, 4, 9, 10, 12\}$ and $\overline{Q_{13}} = \{2, 5, 6, 7, 8, 11\}$.

Quadratic Residue

Example

$\alpha = 6$ is a generator of $\mathbb{Z}_{13}^*$. The powers of α are

i	0	1	2	3	4	5	6	7	8	9	10	11
$\alpha^i \mod 13$	1	6	10	8	9	2	12	7	3	5	4	11

Hence $Q_{13} = \{1, 3, 4, 9, 10, 12\}$ and $\overline{Q_{13}} = \{2, 5, 6, 7, 8, 11\}$.

- Let $n = p.q$ be a product of two distinct odd primes. Then $a \in \mathbb{Z}_n^*$ is a quadratic residue modulo $n \Leftrightarrow a \in Q_p \;\&\; a \in Q_q$.
- It follows that $\#Q_n = \frac{(p-1)(q-1)}{4}$ and $\#\overline{Q_n} = \frac{3(p-1)(q-1)}{4}$.

Quadratic Residue

Example

$\alpha = 6$ is a generator of $\mathbb{Z}_{13}^*$. The powers of α are

i	0	1	2	3	4	5	6	7	8	9	10	11
$\alpha^i \mod 13$	1	6	10	8	9	2	12	7	3	5	4	11

Hence $Q_{13} = \{1, 3, 4, 9, 10, 12\}$ and $\overline{Q_{13}} = \{2, 5, 6, 7, 8, 11\}$.

- Let $n = p.q$ be a product of two distinct odd primes. Then $a \in \mathbb{Z}_n^*$ is a quadratic residue modulo $n \Leftrightarrow a \in Q_p \;\&\; a \in Q_q$.
- It follows that $\#Q_n = \frac{(p-1)(q-1)}{4}$ and $\#\overline{Q_n} = \frac{3(p-1)(q-1)}{4}$.

  Let $n = 21$.
  Then $Q_{21} = \{1, 4, 16\}$ and $\overline{Q_{21}} = \{2, 5, 8, 10, 11, 13, 17, 19, 20\}$.

The Legendre and Jacobi Symbols

- Let p be an odd prime and a an integer. The **Legendre symbol** $\left(\frac{a}{p}\right)$ is defined to be

$$\left(\frac{a}{p}\right) = \begin{cases} 0, & \text{if } p \mid a, \\ 1, & \text{if } a \in Q_p, \\ -1, & \text{if } a \in \overline{Q_p}. \end{cases}$$

The Legendre and Jacobi Symbols

- Let p be an odd prime and a an integer. The **Legendre symbol** $\left(\frac{a}{p}\right)$ is defined to be

$$\left(\frac{a}{p}\right) = \begin{cases} 0, & \text{if } p \mid a, \\ 1, & \text{if } a \in Q_p, \\ -1, & \text{if } a \in \overline{Q_p}. \end{cases}$$

- Let $n \geq 3$ be odd with prime factorization $n = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}$. Then the **Jacobi symbol** $\left(\frac{a}{n}\right)$ is defined to be

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right)^{e_1} \left(\frac{a}{p_2}\right)^{e_2} \cdots \left(\frac{a}{p_k}\right)^{e_k}$$

Properties of Legendre Symbol

(i) $\left(\frac{a}{p}\right) = a^{(p-1)/2} \mod p$. In particular, $\left(\frac{1}{p}\right) = 1$ and $\left(\frac{-1}{p}\right) = (-1)^{(p-1)/2}$. Hence, $-1 \in Q_p$ if $p \equiv 1 \mod 4$, and $-1 \in \overline{Q_p}$ if $p \equiv 3 \mod 4$.

Properties of Legendre Symbol

(i) $\left(\frac{a}{p}\right) = a^{(p-1)/2} \mod p$. In particular, $\left(\frac{1}{p}\right) = 1$ and $\left(\frac{-1}{p}\right) = (-1)^{(p-1)/2}$. Hence, $-1 \in Q_p$ if $p \equiv 1 \mod 4$, and $-1 \in \overline{Q_p}$ if $p \equiv 3 \mod 4$.

(ii) $\left(\frac{ab}{p}\right) = \left(\frac{a}{p}\right)\left(\frac{b}{p}\right)$. Hence if $a \in \mathbb{Z}_p^*$, then $\left(\frac{a^2}{p}\right) = 1$.

Properties of Legendre Symbol

- (i) $\left(\frac{a}{p}\right) = a^{(p-1)/2} \mod p$. In particular, $\left(\frac{1}{p}\right) = 1$ and $\left(\frac{-1}{p}\right) = (-1)^{(p-1)/2}$. Hence, $-1 \in Q_p$ if $p \equiv 1 \mod 4$, and $-1 \in \overline{Q_p}$ if $p \equiv 3 \mod 4$.
- (ii) $\left(\frac{ab}{p}\right) = \left(\frac{a}{p}\right)\left(\frac{b}{p}\right)$. Hence if $a \in \mathbb{Z}_p^*$, then $\left(\frac{a^2}{p}\right) = 1$.
- (iii) If $a \equiv b \mod p$, then $\left(\frac{a}{p}\right) = \left(\frac{b}{p}\right)$.

Properties of Legendre Symbol

1. (i) $\left(\frac{a}{p}\right) = a^{(p-1)/2} \mod p$. In particular, $\left(\frac{1}{p}\right) = 1$ and $\left(\frac{-1}{p}\right) = (-1)^{(p-1)/2}$. Hence, $-1 \in Q_p$ if $p \equiv 1 \mod 4$, and $-1 \in \overline{Q_p}$ if $p \equiv 3 \mod 4$.
2. (ii) $\left(\frac{ab}{p}\right) = \left(\frac{a}{p}\right)\left(\frac{b}{p}\right)$. Hence if $a \in \mathbb{Z}_p^*$, then $\left(\frac{a^2}{p}\right) = 1$.
3. (iii) If $a \equiv b \mod p$, then $\left(\frac{a}{p}\right) = \left(\frac{b}{p}\right)$.
4. (iv) **Law of quadratic reciprocity:** If q is an odd prime distinct from p, then

$$\left(\frac{p}{q}\right) = \left(\frac{q}{p}\right)(-1)^{(p-1)(q-1)/4}.$$

Fermat Test for Primality – Probabilistic Algorithm

Fermat Test for Primality

Input: n
Output: YES if n is composite, NO otherwise.
Choose a random $b,\ 0 < b < n$
if $\gcd(b, n) > 1$ **then**
  return YES
end
else ;
if $b^{n-1} \not\equiv 1 \mod n$ **then**
  return YES
end
else ;
return NO

The Euler Test – Probabilistic Algorithm

- If n is an odd prime, we know that an integer can have at most two square roots, $\mod n$. In particular, the only square roots of $1 \mod n$ are ± 1.
- If $a \not\equiv 0 \mod n$, $a^{(n-1)/2}$ is a square root of $a^{n-1} \equiv 1 \mod n$, so $a^{(n-1)/2} \equiv \pm 1 \mod n$.

The Euler Test – Probabilistic Algorithm

- If n is an odd prime, we know that an integer can have at most two square roots, $\mod n$. In particular, the only square roots of $1 \mod n$ are ± 1.
- If $a \not\equiv 0 \mod n$, $a^{(n-1)/2}$ is a square root of $a^{n-1} \equiv 1 \mod n$, so $a^{(n-1)/2} \equiv \pm 1 \mod n$.
- If $a^{(n-1)/2} \not\equiv \pm 1 \mod n$ for some a with $a \not\equiv 0 \mod n$, then n is composite.

The Euler Test – Probabilistic Algorithm

- For a randomly chosen a with $a \not\equiv 0 \mod n$, compute $a^{(n-1)/2} \mod n$.

The Euler Test – Probabilistic Algorithm

- For a randomly chosen a with $a \not\equiv 0 \mod n$, compute $a^{(n-1)/2} \mod n$.
  - (i) If $a^{(n-1)/2} \equiv \pm 1 \mod n$, declare n a **probable prime**, and optionally repeat the test a few more times.

The Euler Test – Probabilistic Algorithm

- For a randomly chosen a with $a \not\equiv 0 \mod n$, compute $a^{(n-1)/2} \mod n$.

  (i) If $a^{(n-1)/2} \equiv \pm 1 \mod n$, declare n a **probable prime**, and optionally repeat the test a few more times.

  If n is large and chosen at random, the probability that n is prime is very close to 1.

  (ii) If $a^{(n-1)/2} \not\equiv \pm 1 \mod n$, declare n **composite**.

  This is always correct.

The Euler Test – Probabilistic Algorithm

- For a randomly chosen a with $a \not\equiv 0 \mod n$, compute $a^{(n-1)/2} \mod n$.

  (i) If $a^{(n-1)/2} \equiv \pm 1 \mod n$, declare n a **probable prime**, and optionally repeat the test a few more times.

  If n is large and chosen at random, the probability that n is prime is very close to 1.

  (ii) If $a^{(n-1)/2} \not\equiv \pm 1 \mod n$, declare n **composite**.

  This is always correct.

The Euler test is more powerful than the Fermat test.

The Euler Test – Probabilistic Algorithm

The Euler test is more powerful than the Fermat test.

- If the Fermat test finds that n is composite, so does the Euler test.
- If n is an odd composite integer (other than a prime power), 1 has at least 4 square roots $\mod n$.

  So we can have $a^{(n-1)/2} \equiv \beta \mod n$, where $\beta \neq \pm 1$ is a square root of 1.

  Then $a^{n-1} \equiv 1 \mod n$. In this situation, the Fermat Test (incorrectly) declares n a probable prime, but the Euler test (correctly) declares n composite.

Miller-Rabin Test – Probabilistic Algorithm

- The Euler test improves upon the Fermat test by taking advantage of the fact, if 1 has a square root other than $\pm 1 \mod n$, then n must be composite.
- If $a^{(n-1)/2} \not\equiv \pm 1 \mod n$, where $\gcd(a, n) = 1$, then n must be composite for one of two reasons:
  - (i) If $a^{n-1} \not\equiv 1 \mod n$, then n must be composite by Fermat's Little Theorem
  - (ii) If $a^{n-1} \equiv 1 \mod n$, then n must be composite because $a^{(n-1)/2}$ is a square root of $1 \mod n$ different from ± 1.

Miller-Rabin Test – Probabilistic Algorithm

- The Euler test improves upon the Fermat test by taking advantage of the fact, if 1 has a square root other than $\pm 1 \mod n$, then n must be composite.
- If $a^{(n-1)/2} \not\equiv \pm 1 \mod n$, where $\gcd(a, n) = 1$, then n must be composite for one of two reasons:
  1. If $a^{n-1} \not\equiv 1 \mod n$, then n must be composite by Fermat's Little Theorem
  2. If $a^{n-1} \equiv 1 \mod n$, then n must be composite because $a^{(n-1)/2}$ is a square root of $1 \mod n$ different from ± 1.
- The limitation of the Euler test is that is does not go to any special effort to find square roots of 1, different from ± 1. The Miller-Rabin test does this.

Miller-Rabin Test – Probabilistic Algorithm

Miller-Rabin Test

Input: an odd integer $n \geq 3$ and security parameter $t \geq 1$.
Output: an answer "prime" or "composite" to the question: "Is n prime?"
Write $n - 1 = 2^s.r$ s/t r is odd.
for $i = 1$ *to* t **do**
  Choose a random integer a s/t $2 \leq a \leq n - 2$.
  Compute $y \equiv a^r \mod n$
  if $y \neq 1$ & $y \neq n - 1$ **then**
    $j \leftarrow 1$.
    while $j \leq s - 1$ & $y \neq n - 1$ **do**
      Compute $y \leftarrow y^2 \mod n$.
      If $y = 1$ then **return**("composite").
      $j \leftarrow j + 1$.
    end
    If $y \neq n - 1$ then **return** ("composite").
  end
end
Return("prime").

Deterministic Polynomial Time Algorithm

The AKS Algorithm

Input: a positive integer $n > 1$
Output: n is **Prime** or **Composite** in deterministic polynomial-time
If $n = a^b$ with $a \in \mathbb{N}$ & $b > 1$, then output **COMPOSITE.**

Deterministic Polynomial Time Algorithm

The AKS Algorithm

Input: a positive integer $n > 1$
Output: n is **Prime** or **Composite** in deterministic polynomial-time
If $n = a^b$ with $a \in \mathbb{N}$ & $b > 1$, then output **COMPOSITE.**
Find the smallest r such that $ord_r(n) > 4(\log n)^2$.
If $1 < \gcd(a, n) < n$ for some $a \leq r$, then output **COMPOSITE**.

Deterministic Polynomial Time Algorithm

The AKS Algorithm

Input: a positive integer $n > 1$

Output: n is **Prime** or **Composite** in deterministic polynomial-time

If $n = a^b$ with $a \in \mathbb{N}$ & $b > 1$, then output **COMPOSITE.**

Find the smallest r such that $ord_r(n) > 4(\log n)^2$.

If $1 < \gcd(a, n) < n$ for some $a \leq r$, then output **COMPOSITE.**

If $n \leq r$, then output **PRIME.**

Deterministic Polynomial Time Algorithm

The AKS Algorithm

Input: a positive integer $n > 1$
Output: n is **Prime** or **Composite** in deterministic polynomial-time
If $n = a^b$ with $a \in \mathbb{N}$ & $b > 1$, then output **COMPOSITE.**
Find the smallest r such that $ord_r(n) > 4(\log n)^2$.
If $1 < \gcd(a, n) < n$ for some $a \leq r$, then output **COMPOSITE**.
If $n \leq r$, then output **PRIME.**
for $a = 1$ *to* $\lfloor 2\sqrt{\phi(r)}\log n \rfloor$ **do**
    if $(x - a)^n \not\equiv (x^n - a) \mod (x^r - 1, n)$,
    then output **COMPOSITE.**
end
Return("PRIME").

RSA Example

- Suppose A wants to send the following message to B

  RSAISTHEKEYTOPUBLICKEYCRYPTOGRAPHY

- B chooses his $n = 737 = 11 \times 67$. Then $\phi(n) = 660$. Suppose he picks $e = 7, \Rightarrow d = 283$.

RSA Example

- Suppose A wants to send the following message to B

  RSAISTHEKEYTOPUBLICKEYCRYPTOGRAPHY

- B chooses his $n = 737 = 11 \times 67$. Then $\phi(n) = 660$. Suppose he picks $e = 7, \Rightarrow d = 283$.
- $\because \; 26^2 < n < 26^3 \quad \therefore$

RSA Example

- Suppose A wants to send the following message to B

  RSAISTHEKEYTOPUBLICKEYCRYPTOGRAPHY
- B chooses his $n = 737 = 11 \times 67$. Then $\phi(n) = 660$. Suppose he picks $e = 7, \Rightarrow d = 283$.
- $\because \; 26^2 < n < 26^3 \quad \therefore$ the block size of the plaintext $= 2$.

$$m_1 = \text{‘}RS\text{'} = 17 \times 26 + 18 = 460$$

$$c_1 = 460^7 \equiv 697 \mod 737 = 1.26^2 + 0.26 + 21 = BAV$$

RSA Example

	RS	AI	ST	HE	KE	YT	OP	UB
m_b	460	8	487	186	264	643	379	521
c_b	697	387	229	340	165	223	586	5

LI	CK	EY	CR	YP	TO	GR	AP	HY
294	62	128	69	639	508	173	15	206
189	600	325	262	100	689	354	665	673

RSA Example

- Suppose A wants to send the following message to B

power

- B chooses his $n = 1943 = 29 \times 67$. Then $\phi(n) = 1848$. Suppose he picks $e = 701, \Rightarrow d = 29$.
- $\because\ 26^2 < n < 26^3 \quad \therefore$ the block size of the plaintext $= 2$.
- $m_1 = \text{`}po' = 15 \times 26 + 14 = 404,\ m_2 = \text{`}we' = 22 \times 26 + 4 = 576,\ m_3 = \text{`}ra' = 17 \times 26 + 0 = 442$.
- $c_1 = 404^{701} \equiv 1419 \mod 1943 = 2.26^2 + 2.26 + 15 = ccp$.
- $\|ly,\ c_2 = 344 = 13.26 + 6 = ang$ & $c_3 = 210 = 8.26 + 2 = aic$.
- The cipher text is

ccpangaic

Security of RSA

Security

If we know n and $\phi(n)$, we can find p & q.

Security of RSA

Security

If we know n and $\phi(n)$, we can find p & q.

We have

$$\phi(n) = pq - p - q + 1 = n - (p + q) + 1.$$

Since we know n, we can find $p + q$ from the above equation. Since we know $pq = n$ and $p + q$, we can find p & q by factoring the quadratic equation

$$x^2 - (p + q)x + pq = 0.$$

Security of RSA

- Security of RSA relies on difficulty of finding d given n & e.
- Breaking RSA is no harder than Factoring.
- It is not secure against chosen ciphertext attacks (CCA).

Security of RSA

- Security of RSA relies on difficulty of finding d given n & e.
- Breaking RSA is no harder than Factoring.
- It is not secure against chosen ciphertext attacks (CCA).
  - **Input challenge** ciphertext $c \equiv m^e \mod N$.

Security of RSA

- Security of RSA relies on difficulty of finding d given n & e.
- Breaking RSA is no harder than Factoring.
- It is not secure against chosen ciphertext attacks (CCA).
  - **Input challenge** ciphertext $c \equiv m^e \mod N$.
  - Submit ciphertext $c' \equiv r^e c \mod N$ for decryption.
  - Receive message $m' = rm$.
  - Original message is $r^{-1}m' \mod N \equiv m$.
- RSA is secure against chosen plaintext attack (CPA).

IND-CCA

Security notion for encryption.

- From a ciphertext c, an attacker should not be able to derive any information from the corresponding plaintext m.
- Even if the attacker can obtain the decryption of any ciphertext, c excepted.
- This is called indistinguishability against a chosen ciphertext attack (IND-CCA).

SBI Public Key Information

Public Key Info

Algorithm	RSA
Key Size	2048
Exponent	65537
Modulus	A6:55:7F:B2:9C:23:FC:79:F8:9D:90:F6:75:4E:CE:3A:26:90:B8:37:EA:8E:6E: D6:18:8A:FC:F6:CA:7C:6F:4B:45:4D:98:DE:4F:3D:A3:78:5E:0C:4A:1A:81:8D: 6F:C3:BB:4C:38:6E:04:0B:1F:BB:CB:50:8B:42:E9:E2:17:65:E2:C0:D0:CA:F4: E5:C6:0A:C9:47:53:32:15:69:F6:C4:EC:B0:E0:B0:FC:CB:BA:DE:DF:BE:ED:2 B:44:3D:F6:2B:B3:0A:CA:B8:FC:D1:5F:84:2C:34:1E:15:52:76:4E:90:FA:85:7 0:BB:05:C3:02:03:17:74:B3:80:A1:59:1F:19:7B:3A:2B:C3:D5:59:CF:BA:5D:B E:DF:3B:3A:8E:52:C1:D3:A3:8C:06:D2:2A:98:2F:4D:82:7F:28:F1:B1:D3:71:7 E:CF:4C:B1:26:F4:6F:EA:09:F9:7F:5A:D6:15:46:5C:92:50:D4:F4:F3:CA:60:2 5:4D:9A:66:91:1D:EA:74:D4:B1:71:D9:30:15:4C:BB:B6:CD:C6:18:82:F8:B7:4 8:97:AF:2F:22:15:94:FE:EB:E7:DE:EF:CA:A3:6E:CC:26:69:D5:92:5B:68:89:5 6:2B:B3:72:60:62:49:8B:C5:59:45:43:C1:F4:7E:8F:2B:C4:DD:C1:BB:39:D4:B C:5C:51:53

LinkedIn Public Key Information

Public Key Info

Algorithm	RSA
Key Size	2048
Exponent	65537
Modulus	D4:8A:8B:DF:28:F5:5C:7B:B6:79:74:E5:F4:4A:5B:E7:38:94:69:B7:BA:19:4D: A7:A9:73:64:6F:DD:B8:4C:99:5A:91:E8:F5:C8:D7:B1:1E:5B:3E:3E:AE:77:6B: A3:E3:DF:D3:29:38:59:E8:66:59:5D:37:FF:75:20:4E:66:1B:D0:C8:73:9E:A0: 38:6E:16:98:BD:DB:CC:D8:95:CF:87:AE:5E:42:10:F8:10:34:BF:E8:1F:5A:0A: 4B:A3:28:25:55:3F:FD:15:D0:3D:25:EF:09:6C:E4:C0:E4:9F:E7:4E:28:C6:D0: 63:2C:07:4C:CE:4F:4E:EE:B1:70:66:07:96:40:E3:51:1B:23:91:84:12:AE:A5:F A:2D:B0:3E:1E:C1:AC:BF:80:90:31:81:88:C7:5C:66:0E:34:5F:62:B5:CF:03:8 E:C8:74:82:77:01:A1:E8:A1:D3:1D:4B:43:6A:87:F2:E2:22:48:58:B2:3A:88:C7: F8:DC:9D:70:D9:BE:83:E1:B2:E9:BA:AC:C5:EF:B0:CB:76:9D:6E:10:F7:C9:80: 6E:B7:C7:30:5B:85:5F:D9:6C:26:B1:B9:59:24:17:C5:F6:01:CD:67:FA:21:E8:B B:1D:24:44:20:6B:09:CA:8F:5B:10:AF:76:B0:AB:33:9F:28:B2:B1:C8:FC:2F:E 5:71

IIITL Public Key Information

Public Key Info

Algorithm	RSA
Key Size	2048
Exponent	65537
Modulus	BF:26:C8:BA:E3:2F:68:5A:8F:C1:82:43:AC:0A:82:B5:0D:4E:04:6E:B1:85:35: 8E:14:51:AC:7A:44:4F:A5:CF:A2:3C:4C:8B:97:7E:0E:8C:4A:F6:05:1F:53:5C:4 E:D1:1D:23:84:8C:8F:C7:B6:99:AA:6D:00:36:E4:FF:53:7F:EC:FF:9F:42:B9:2 B:F5:EF:39:9B:7C:F3:51:75:0F:0C:B1:AA:FB:4C:59:40:06:C5:60:0F:5D:2F:A 8:47:CE:47:CF:69:73:0B:AB:71:44:51:01:6D:E1:C8:9A:EF:FA:96:A4:E7:AF:5E: 1F:4B:A7:6C:26:8A:7B:4E:A9:14:7A:EC:74:7B:7B:D3:9B:51:C7:60:1F:E7:CB:7 F:E9:A8:F2:C5:6F:22:4A:42:AB:60:B5:BF:D9:9D:CA:D7:6D:F2:8C:06:6E:30: A5:F1:AB:EC:32:73:D3:E8:67:93:E3:06:C9:58:C5:99:43:8C:5E:3C:C2:7A:B9: 1B:27:47:29:B7:9E:9A:DC:FB:63:6A:E0:A1:BC:33:B0:FE:C1:12:6F:01:73:A7:A B:3E:C9:92:EB:45:FE:5D:86:CA:4D:99:87:6E:75:4C:B3:CD:85:F0:AE:61:9B:B C:C6:9E:A4:3A:D2:53:76:EE:73:D9:3A:52:0C:CD:D1:73:70:7A:D5:BC:DC:5E: 58:7D

Choice of Encryption Key e

- The encryption exponent e should not be too small.

Choice of Encryption Key e

- The encryption exponent e should not be too small.
- Suppose $e = 3$ and there are 3 recipients having the same encryption exponent 3, but with different modulus $n_i, \quad i = 1, 2, 3$.
- Then, ciphertexts $y_i \equiv M^3 \mod n_i$ for $i = 1, 2, 3$ and send them to the recipients.
- Assume that n_i for $i = 1, 2, 3$ are pairwise coprime.

Choice of Encryption Key e

- The encryption exponent e should not be too small.
- Suppose $e = 3$ and there are 3 recipients having the same encryption exponent 3, but with different modulus $n_i, \quad i = 1, 2, 3$.
- Then, ciphertexts $y_i \equiv M^3 \mod n_i$ for $i = 1, 2, 3$ and send them to the recipients.
- Assume that n_i for $i = 1, 2, 3$ are pairwise coprime.
- Suppose two of them, say n_1 & n_2, are not coprime.

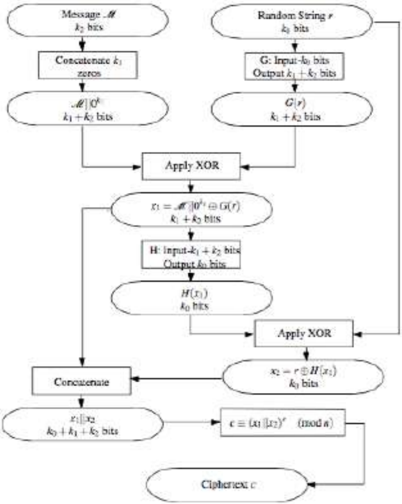
Choice of Encryption Key e

- The encryption exponent e should not be too small.
- Suppose $e = 3$ and there are 3 recipients having the same encryption exponent 3, but with different modulus $n_i, \quad i = 1, 2, 3$.
- Then, ciphertexts $y_i \equiv M^3 \mod n_i$ for $i = 1, 2, 3$ and send them to the recipients.
- Assume that n_i for $i = 1, 2, 3$ are pairwise coprime.
- Suppose two of them, say n_1 & n_2, are not coprime. Then, $\gcd(n_1, n_2)$ is a nontrivial factor of n_1 & n_2 and any adversary can factorise both of them.
- If adversary gets hold of the messages $y_i, \ 1 \leq i \leq 3$, (s)he can compute $M^3 \mod n_1 n_2 n_3$ using Chinese remainder theorem since $\gcd(n_i, n_j) = 1$ for $i \neq j$.

Choice of Encryption Key e

- The encryption exponent e should not be too small.
- Suppose $e = 3$ and there are 3 recipients having the same encryption exponent 3, but with different modulus $n_i, \quad i = 1, 2, 3$.
- Then, ciphertexts $y_i \equiv M^3 \mod n_i$ for $i = 1, 2, 3$ and send them to the recipients.
- Assume that n_i for $i = 1, 2, 3$ are pairwise coprime.
- Suppose two of them, say n_1 & n_2, are not coprime. Then, $\gcd(n_1, n_2)$ is a nontrivial factor of n_1 & n_2 and any adversary can factorise both of them.
- If adversary gets hold of the messages $y_i, \ 1 \leq i \leq 3$, (s)he can compute $M^3 \mod n_1 n_2 n_3$ using Chinese remainder theorem since $\gcd(n_i, n_j) = 1$ for $i \neq j$.
- $\because \ m < n_i, \ m^3 < n_1 n_2 n_3$. So, $M^3 \mod n_1 n_2 n_3 = M^3$ and the adversary can find M by taking the cube root of $M^3 \mod n_1 n_2 n_3$.

RSA in Practice – Optimal Asymmetric Encryption Padding (OAEP)



Optimal Asymmetric Encryption Padding (OAEP) I

- To encrypt a message M of k_2-bit, first concatenates the message with 0^{k_1}.
- Expands the message to $M||0^{k_1}$.
- After that, select a random string r of length k_0 bits.
- Use it as the random seed for $G(r)$ and computes

$$x_1 = (M||0^{k_1}) \oplus G(r), \quad x_2 = r \oplus H(x_1)$$

- If $x_1||x_2$ is a binary number bigger than n, Alice chooses another random string r and computes the new values of x_1 & x_2.
- If $G(r)$ produces fairly random outputs, $x_1||x_2$ will be less than n in binary with a probability greater than $\frac{1}{2}$.

Optimal Asymmetric Encryption Padding (OAEP) II

- After getting a string r with $x_1 \| x_2 < n$, Alice then encrypts $x_1 \| x_2$ to get the ciphertext

$$E(M) = (x_1 \| x_2)^e \equiv c \mod n$$

ElGamal PKC in $\mathbb{Z}_p^*$

This was designed by Taher ElGamal in 1985

ElGamal PKC in $\mathbb{Z}_p^*$

This was designed by Taher ElGamal in 1985

Key Generation:

- $<\alpha>= \mathbb{Z}_p^*,\ \mathcal{P} = \mathbb{Z}_p^*\ \&\ \mathcal{C} = \mathbb{Z}_p^* \times \mathbb{Z}_p^*$.
- $\beta \equiv \alpha^a \mod p$.
- Public: p, α, β and Private: a.

ElGamal PKC in $\mathbb{Z}_p^*$

This was designed by Taher ElGamal in 1985

Key Generation:

- $<\alpha>= \mathbb{Z}_p^*,\ \mathcal{P} = \mathbb{Z}_p^*\ \&\ \mathcal{C} = \mathbb{Z}_p^* \times \mathbb{Z}_p^*$.
- $\beta \equiv \alpha^a \mod p$.
- Public: p, α, β and Private: a.

Encryption:

- Select a random $k \in \mathbb{Z}_{p-1}$.
- $Enc_k(x) = (y_1, y_2)$

$$y_1 \equiv \alpha^k \mod p, \quad y_2 \equiv x.\beta^k \mod p.$$

ElGamal PKC in $\mathbb{Z}_p^*$

This was designed by Taher ElGamal in 1985

Key Generation:

- $<\alpha>= \mathbb{Z}_p^*,\ \mathcal{P} = \mathbb{Z}_p^*\ \&\ \mathcal{C} = \mathbb{Z}_p^* \times \mathbb{Z}_p^*$.
- $\beta \equiv \alpha^a \mod p$.
- Public: p, α, β and Private: a.

Encryption:

- Select a random $k \in \mathbb{Z}_{p-1}$.
- $Enc_k(x) = (y_1, y_2)$

$$y_1 \equiv \alpha^k \mod p, \quad y_2 \equiv x.\beta^k \mod p.$$

Decryption:

$$Dec_k(y_1, y_2) \equiv y_2.(y_1^a)^{-1} \mod p.$$

ElGamal PKC in $\mathbb{Z}_p^*$

Example

- Let $p = 29$ and $\alpha = 2$, α is a primitive element $\mod 29$.
- Let $a = 5$,

ElGamal PKC in $\mathbb{Z}_p^*$

Example

- Let $p = 29$ and $\alpha = 2$, α is a primitive element $\mod 29$.
- Let $a = 5$, $\therefore \beta \equiv 2^5 \mod \equiv 3 \mod 29$.
- **Public Key:** $(29, 2, 3)$ and **Private Key:** 5
- **Plaintext:** $x = 6$ & random number $k = 14 \in \mathbb{Z}_{28}$

ElGamal PKC in $\mathbb{Z}_p^*$

Example

- Let $p = 29$ and $\alpha = 2$, α is a primitive element $\mod 29$.
- Let $a = 5$, $\therefore \beta \equiv 2^5 \mod \equiv 3 \mod 29$.
- **Public Key:** $(29, 2, 3)$ and **Private Key:** 5
- **Plaintext:** $x = 6$ & random number $k = 14 \in \mathbb{Z}_{28}$
- $$y_1 \equiv 2^{14} \equiv 28 \mod 29 \;\&\; y_2 \equiv 6.3^{14} \equiv 23 \mod 29$$
- **Ciphertext:** $(28, 23)$.

Security of ElGamal Ciphertexts

Security of ElGamal Ciphertexts

- Suppose Eve claims to have obtained the plaintext m for an RSA ciphertext c.
- It is easy to verify her claim

Security of ElGamal Ciphertexts

- Suppose Eve claims to have obtained the plaintext m for an RSA ciphertext c.
- It is easy to verify her claim
- Now suppose instead that Eve claims to possess the message m corresponding to an ElGamal encryption (r, t).
- Can you verify her claim?

Security of ElGamal Ciphertexts

- Suppose Eve claims to have obtained the plaintext m for an RSA ciphertext c.
- It is easy to verify her claim
- Now suppose instead that Eve claims to possess the message m corresponding to an ElGamal encryption (r, t).
- Can you verify her claim?
- This is as hard as the decision Diffie-Hellman problem.

Elliptic Curves

- Elliptic curve[1] E over field $\mathbb{K}$ is defined by

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, \quad a_i \in \mathbb{K}$$

- The set of $\mathbb{K}$-rational points $E(\mathbb{K})$ is defined as

$$E(\mathbb{K}) = \{(x, y) \in \mathbb{K} \times \mathbb{K} : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6\} \cup \{\mathcal{O}\}$$

[1] It is called a (generalized) Weierstrass equation. The equation defines a cubic curve called a Weierstrass curve.

Elliptic Curves

- Elliptic curve[1] E over field $\mathbb{K}$ is defined by

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, \quad a_i \in \mathbb{K}$$

- The set of $\mathbb{K}$-rational points $E(\mathbb{K})$ is defined as

$$E(\mathbb{K}) = \{(x, y) \in \mathbb{K} \times \mathbb{K} : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6\} \cup \{\mathcal{O}\}$$

Theorem

There exists an addition law on E and the set $E(K)$ with that addition forms a group.

[1] It is called a (generalized) Weierstrass equation. The equation defines a cubic curve called a Weierstrass curve.

Elliptic Curves

1. Let $\mathbb{K}$ be a field of characteristic $\neq 2, 3$, and let $x^3 + ax + b$ be a cubic polynomial with no multiple roots, i.e., when

$$-16(4a^3 + 27b^2) \neq 0 \Rightarrow 4a^3 + 27b^2 \neq 0.$$

An elliptic curve over $\mathbb{K}$ is the set of points (x, y) with $x, y \in K$ which satisfy the equation

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

together with a single element denoted O and called the *point at infinity*.

Elliptic Curves

1. Let $\mathbb{K}$ be a field of characteristic $\neq 2, 3$, and let $x^3 + ax + b$ be a cubic polynomial with no multiple roots, i.e., when

$$-16(4a^3 + 27b^2) \neq 0 \Rightarrow 4a^3 + 27b^2 \neq 0.$$

   An elliptic curve over $\mathbb{K}$ is the set of points (x, y) with $x, y \in K$ which satisfy the equation

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

   together with a single element denoted O and called the *point at infinity*.

2. If char $K = 2$, then an elliptic curve over $\mathbb{K}$ is the set of points satisfying an equation of type either

$$y^2 + cy = x^3 + ax + b \text{ or } y^2 + xy = x^3 + ax + b$$

   together with the *point at infinity* O.

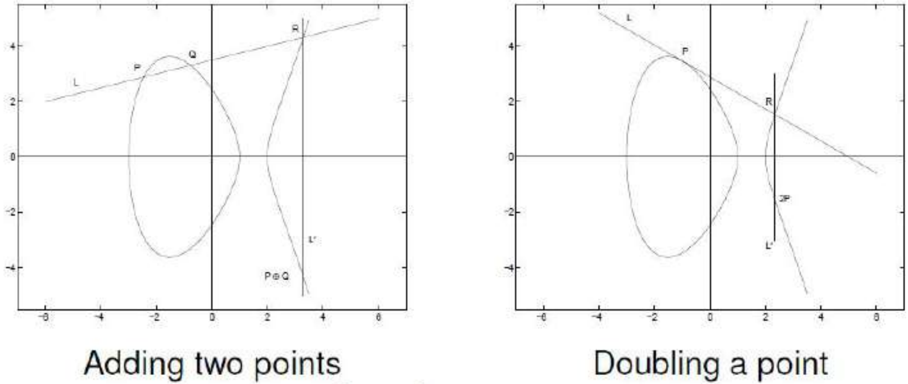
Elliptic Curves

3. If char $K = 3$, then an elliptic curve over $\mathbb{K}$ is the set of points satisfying the equation

$$y^2 = x^3 + ax^2 + bx + c$$

together with the *point at infinity* O.

Addition Law on Elliptic Curves



$$y^2 = x^3 - 7x + 6$$

Addition Law on Elliptic Curves I

- Suppose E is a nonsingular elliptic curve.
- The point at infinity $\mathcal{O}$, will be the identity element, so $P + \mathcal{O} = \mathcal{O} + P = P \; \forall \; P \in E$.
- Suppose $P, Q \in E$, where $P = (x_1, y_1)$ & $Q = (x_2, y_2)$

  (i) $x_1 \neq x_2$

  - L is the line through P and Q.
  - L intersects E in the two points P and Q
  - L will intersect E in one further point R'.
  - If we reflect R' in the x-axis, then we get a point R.

$$P + Q = R.$$

Addition Law on Elliptic Curves II

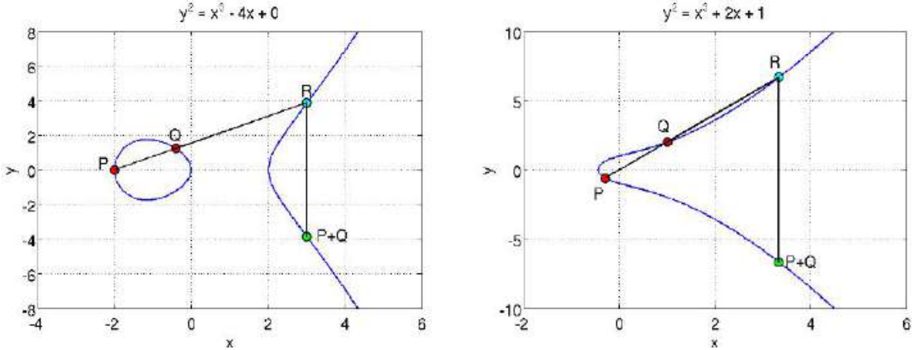
(ii) $x_1 = x_2 \;\&\; y_1 = -y_2$

$$(x, y) + (x, -y) = \mathcal{O}$$

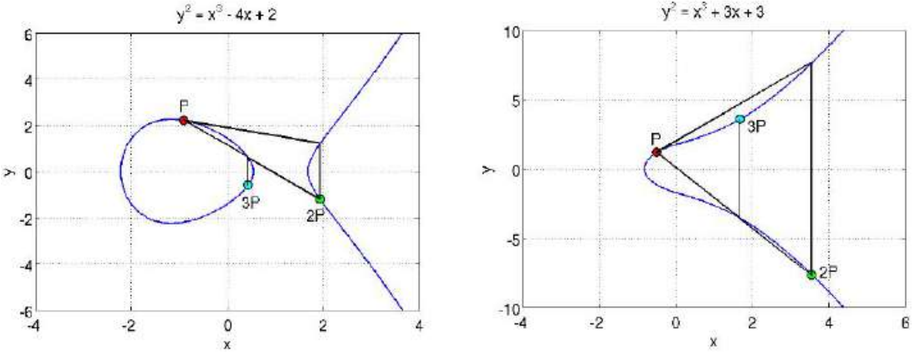
(iii) $x_1 = x_2 \;\&\; y_1 = y_2$

- Draw a tangent line L through P
- Follow step (i)

Addition Law on Elliptic Curves



Addition Law on Elliptic Curves



Addition Law on Elliptic Curves

- Suppose that we want to add the points $P_1 = (x_1, y_1)$ & $P_2 = (x_2, y_2)$ on the elliptic curve

$$E \; : \; y^2 = x^3 + ax + b.$$

Addition Law on Elliptic Curves

- Suppose that we want to add the points $P_1 = (x_1, y_1)$ & $P_2 = (x_2, y_2)$ on the elliptic curve

$$E \ : \ y^2 = x^3 + ax + b.$$

- Let the line connecting P_1 to P_2 be

$$L \ : \ y = \lambda x + \nu$$

- Explicitly, the slope and y-intercept of L are given by

Addition Law on Elliptic Curves

- Suppose that we want to add the points $P_1 = (x_1, y_1)$ & $P_2 = (x_2, y_2)$ on the elliptic curve

$$E \ : \ y^2 = x^3 + ax + b.$$

- Let the line connecting P_1 to P_2 be

$$L \ : \ y = \lambda x + \nu$$

- Explicitly, the slope and y-intercept of L are given by

$$\lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} & \text{if } P_1 \neq P_2 \\ \frac{3x_1^2 + a}{2y_1} & \text{if } P_1 = P_2 \end{cases} \qquad \text{and} \qquad \nu = y_1 - \lambda x_1$$

Addition Law on Elliptic Curves

- Thus, we have

$$P_1 + P_2 = (x_3, -y_3),$$

where $x_3 = \lambda^2 - x_1 - x_2$ and $y_3 = \lambda x_3 + \nu$.

Addition Law on Elliptic Curves

- Thus, we have
$$P_1 + P_2 = (x_3, -y_3),$$
where $x_3 = \lambda^2 - x_1 - x_2$ and $y_3 = \lambda x_3 + \nu$.
- If $P_1 \neq P_2$ and $x_1 = x_2$, then $P_1 + P_2 = \mathcal{O}$.

Addition Law on Elliptic Curves

- Thus, we have

$$P_1 + P_2 = (x_3, -y_3),$$

  where $x_3 = \lambda^2 - x_1 - x_2$ and $y_3 = \lambda x_3 + \nu$.
- If $P_1 \neq P_2$ and $x_1 = x_2$, then $P_1 + P_2 = \mathcal{O}$.
- If $P_1 = P_2$ and $y_1 = 0$, then $P_1 + P_2 = 2P_1 = \mathcal{O}$.

Addition Law on Elliptic Curves

- Thus, we have

$$P_1 + P_2 = (x_3, -y_3),$$

  where $x_3 = \lambda^2 - x_1 - x_2$ and $y_3 = \lambda x_3 + \nu$.
- If $P_1 \neq P_2$ and $x_1 = x_2$, then $P_1 + P_2 = \mathcal{O}$.
- If $P_1 = P_2$ and $y_1 = 0$, then $P_1 + P_2 = 2P_1 = \mathcal{O}$.

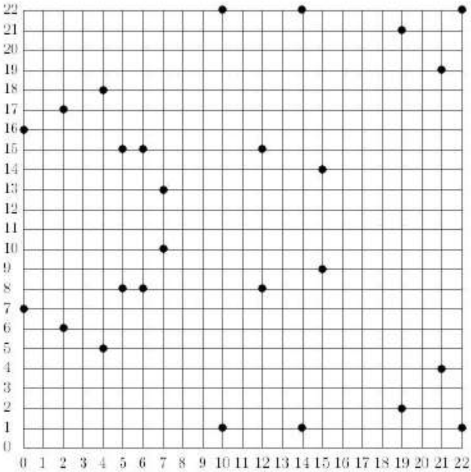
Visualizing Elliptic Curve Cryptography

Elliptic Curves over Finite Fields

Example

Let E be the elliptic curve $y^2 = x^3 + x + 3$ over $\mathbb{F}_{23}$. Then write down all the points of E over $\mathbb{F}_{23}$. Draw the elliptic curve E along with the grid.

Elliptic Curves over Finite Fields



The elliptic curve $y^2 = x^3 + x + 3 \bmod 23$

Elliptic Curves over Finite Fields

Problem

Let E be the elliptic curve $y^2 = x^3 + x + 1$ over $\mathbb{F}_{11}$. Then write down all the points of E over $\mathbb{F}_{11}$. Draw the elliptic curve E along with the grid.

Elliptic Curves over Finite Fields

Solution

NIST's Primes for ECC

$$
\begin{aligned}
p_{192} &= 2^{192} - 2^{64} - 1 \\
p_{224} &= 2^{224} - 2^{96} + 1 \\
p_{256} &= 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1 \\
p_{384} &= 2^{384} - 2^{128} - 2^{96} + 2^{32} - 1 \\
p_{521} &= 2^{521} - 1
\end{aligned}
$$

$$
\begin{aligned}
\text{W-}25519 &= 2^{255} - 19 \\
\text{W-}448 &= 2^{448} - 2^{224} - 1
\end{aligned}
$$

$$
\begin{aligned}
\text{Edwards}25519 &= 2^{255} - 19 \\
\text{Edwards}448 &= 2^{448} - 2^{224} - 1
\end{aligned}
$$

Recommendations for Discrete Logarithm-Based Cryptography: Elliptic Curve Domain Parameters

ElGamal Cryptosystems on Elliptic Curves

- First choose two public elliptic curve points P and Q s/t

$$Q = sP,$$

where s is the private key.

ElGamal Cryptosystems on Elliptic Curves

- First choose two public elliptic curve points P and Q s/t

$$Q = sP,$$

where s is the private key.

- To encrypt choose a random k
- $Enc_k(m) = (y_1, y_2)$

$$y_1 = kP, \quad y_2 = m + kQ.$$

ElGamal Cryptosystems on Elliptic Curves

- First choose two public elliptic curve points P and Q s/t

$$Q = sP,$$

where s is the private key.

- To encrypt choose a random k
- $Enc_k(m) = (y_1, y_2)$

$$y_1 = kP, \quad y_2 = m + kQ.$$

- **Decryption:**

$$Dec_k(y_1, y_2) = y_2 - s.y_1$$

ElGamal Cryptosystems on Elliptic Curves

- The plaintext space in general may not consist of the points on the curve E.
- Convert the plaintext as an arbitrary element in $\mathbb{Z}_p$.
- Apply a suitable hash function $h : E \to \mathbb{Z}_p$ to kQ

ElGamal Cryptosystems on Elliptic Curves

- The plaintext space in general may not consist of the points on the curve E.
- Convert the plaintext as an arbitrary element in $\mathbb{Z}_p$.
- Apply a suitable hash function $h : E \to \mathbb{Z}_p$ to kQ
- To encrypt a message m choose a random k
- The ciphertext $c = Enc_k(m) = (y_1, y_2)$

$$y_1 = kP, \quad y_2 \equiv (m + h(kQ)) \mod p.$$

- **Decryption:**

ElGamal Cryptosystems on Elliptic Curves

- The plaintext space in general may not consist of the points on the curve E.
- Convert the plaintext as an arbitrary element in $\mathbb{Z}_p$.
- Apply a suitable hash function $h : E \to \mathbb{Z}_p$ to kQ
- To encrypt a message m choose a random k
- The ciphertext $c = Enc_k(m) = (y_1, y_2)$

$$y_1 = kP, \quad y_2 \equiv (m + h(kQ)) \mod p.$$

- **Decryption:**
  - Compute $h(kQ)$
  - Compute $c \equiv (y_2 - h(kQ)) \mod p$

ElGamal Cryptosystems on Elliptic Curves

Key Generation

- Let E be an elliptic curve defined over $\mathbb{Z}_p$ (where $p > 3$ is prime) s/t E contains a cyclic subgroup $H = \langle P \rangle$ of prime order n in which the **Discrete Logarithm Problem** is infeasible.
- Let $h : E \to \mathbb{Z}_p$ be a secure hash function.
- Let $\mathcal{P} = \mathbb{Z}_p$ and $C = (\mathbb{Z}_p \times \mathbb{Z}_2) \times \mathbb{Z}_p$. Define

$$\mathcal{K} = \{(E, P, s, Q, n, h) \ : \ Q = sP\},$$

where P and Q are points on E and $s \in \mathbb{Z}_n^*$.

ElGamal Cryptosystems on Elliptic Curves

Key Generation

- Let E be an elliptic curve defined over $\mathbb{Z}_p$ (where $p > 3$ is prime) s/t E contains a cyclic subgroup $H = \langle P \rangle$ of prime order n in which the **Discrete Logarithm Problem** is infeasible.
- Let $h : E \to \mathbb{Z}_p$ be a secure hash function.
- Let $\mathcal{P} = \mathbb{Z}_p$ and $C = (\mathbb{Z}_p \times \mathbb{Z}_2) \times \mathbb{Z}_p$. Define

$$\mathcal{K} = \{(E, P, s, Q, n, h) \ : \ Q = sP\},$$

where P and Q are points on E and $s \in \mathbb{Z}_n^*$.

The values E, P, Q, n, and h are the public key and s is the private key.

ElGamal Cryptosystems on Elliptic Curves

Encryption

- To encrypt a message m sender selects a random number $k \in \mathbb{Z}_n^*$ and compute the ciphertext

$$y = e_K(m, k) = (y_1, y_2) = (\text{ POINT-COMPRESS}(kP), m + h(kQ) \mod p),$$

where $y_1 \in \mathbb{Z}_p \times \mathbb{Z}_2$ and $y_2 \in \mathbb{Z}_p$.

ElGamal Cryptosystems on Elliptic Curves

Encryption

- To encrypt a message m sender selects a random number $k \in \mathbb{Z}_n^*$ and compute the ciphertext

$$y = e_K(m, k) = (y_1, y_2) = (\text{ POINT-COMPRESS}(kP), m + h(kQ) \mod p),$$

where $y_1 \in \mathbb{Z}_p \times \mathbb{Z}_2$ and $y_2 \in \mathbb{Z}_p$.

Decryption

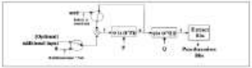
$$d_K(y) = y_2 - h(R) \mod p,$$

where $R = s\text{POINT-DECOMPRESS}(y_1)$.

The Many Flaws of Dual_EC_DRBG

Matthew Green in Dual EC, NSA, RNGs · September 18, 2013 · 3,035 Words

The Many Flaws of Dual_EC_DRBG



The Dual_EC_DRBG generator from NIST SP800-90A.

Update 9/19: *RSA warns developers not to use the default Dual_EC_DRBG generator in BSAFE. Oh lord.*

As a technical follow up to my previous post about the NSA's war on crypto, I wanted to make a few specific points about standards. In particular I wanted to address the allegation that NSA inserted a backdoor into the Dual-EC pseudorandom number generator.

For those not following the story, Dual-EC is a pseudorandom number generator proposed by NIST for international use back in 2006. Just a few months later, Shumow and Ferguson made cryptographic history by pointing out that there might be an NSA backdoor in the algorithm. This possibility — fairly remarkable for an algorithm of this type — looked bad and smelled worse. If true, it spelled almost certain doom for anyone relying on Dual-EC to keep their system safe from spying eyes.

Key Comparison

Symmetric Key Size (in bits)	Based on Factoring (in bits)	Based on DLP (in bits)	Based on ECDLP (in bits)
80	1024	1024	160
112	2048	2048	224
128	3072	3072	256
192	7680	7680	384
256	15360	15360	512

Outline

Integer Factorization

Integer Factorization

- **Basic Method:** divide n by all primes $p \leq \sqrt{n}$

Integer Factorization

- **Basic Method:** divide n by all primes $p \leq \sqrt{n}$
- **Fermat factorization:**

Integer Factorization

- **Basic Method:** divide n by all primes $p \leq \sqrt{n}$
- **Fermat factorization:**
  - **The idea:** express n as a difference of two squares:
  $$n = x^2 - y^2$$

Integer Factorization

- **Basic Method:** divide n by all primes $p \leq \sqrt{n}$
- **Fermat factorization:**
  - **The idea:** express n as a difference of two squares:

$$n = x^2 - y^2$$

Example

1. Factor $n = 295927$

Integer Factorization

- **Basic Method:** divide n by all primes $p \leq \sqrt{n}$
- **Fermat factorization:**
  - **The idea:** express n as a difference of two squares:

$$n = x^2 - y^2$$

Example

1. Factor $n = 295927$

$$\begin{aligned} 295927 + 1^2 &= 295928 \neq \text{perfect square} \\ 295927 + 2^2 &= 295931 \neq \text{perfect square} \\ 295927 + 3^2 &= 295936 \end{aligned}$$

Integer Factorization

- **Basic Method:** divide n by all primes $p \leq \sqrt{n}$
- **Fermat factorization:**
  - **The idea:** express n as a difference of two squares:

$$n = x^2 - y^2$$

Example

1. Factor $n = 295927$

$$\begin{aligned} 295927 + 1^2 &= 295928 && \neq \text{perfect square} \\ 295927 + 2^2 &= 295931 && \neq \text{perfect square} \\ 295927 + 3^2 &= 295936 && = 544^2 \end{aligned}$$

$$295927 = 544^2 - 3^2 = 547 \times 541$$

Integer Factorization

n can be factored if $k\phi(n)$ is given

- Factorize n, with a *high probability*, if any multiple of $\phi(n)$ is known;

Integer Factorization

n can be factored if $k\phi(n)$ is given

- Factorize n, with a *high probability*, if any multiple of $\phi(n)$ is known;

$$\because ed = k \times \phi(n) = k(p-1)(q-1).$$

Integer Factorization

n can be factored if $k\phi(n)$ is given

- Factorize n, with a *high probability*, if any multiple of $\phi(n)$ is known;

$$\because\ ed = k \times \phi(n) = k(p-1)(q-1).$$

- Find an exponent r s/t

$$b^r \equiv 1 \mod n,\ \forall\ b \text{ with } \gcd(b,n) = 1$$

Integer Factorization

n can be factored if $k\phi(n)$ is given

- Factorize n, with a *high probability*, if any multiple of $\phi(n)$ is known;

$$\because\ ed = k \times \phi(n) = k(p-1)(q-1).$$

- Find an exponent r s/t

$$b^r \equiv 1 \mod n,\ \forall\ b \text{ with } \gcd(b,n) = 1$$

- Write $r = a \cdot 2^s$ with a odd.
- Choose a random b with $1 < b < n-1$.
- If $\gcd(b,n) \neq 1$ we have found a factor of n.

Integer Factorization

n can be factored if $k\phi(n)$ is given

- Otherwise, let $b_0 \equiv b^a \mod n$. We compute
$$b_1 \equiv b_0^2 \mod n,\ b_2 \equiv b_1^2 \mod n, b_3 \equiv b_2^2 \mod n, \dots$$
- If $b_0 \equiv 1 \mod n$, we choose another b and repeat the procedure.
- Also, if $b_k \equiv -1 \mod n$ for some k, we choose a different b and repeat the procedure.
- If $b_{k+1} \equiv 1 \mod n$ & $b_k \not\equiv \pm 1 \mod n$ for some k,

  $\gcd(b_k - 1, n)$ gives a nontrivial divisor of n.

Integer Factorization

n can be factored if $k\phi(n)$ is given

- Otherwise, let $b_0 \equiv b^a \mod n$. We compute
$$b_1 \equiv b_0^2 \mod n,\ b_2 \equiv b_1^2 \mod n, b_3 \equiv b_2^2 \mod n, \ldots$$
- If $b_0 \equiv 1 \mod n$, we choose another b and repeat the procedure.
- Also, if $b_k \equiv -1 \mod n$ for some k, we choose a different b and repeat the procedure.
- If $b_{k+1} \equiv 1 \mod n \ \&\ b_k \not\equiv \pm 1 \mod n$ for some k,

  $\gcd(b_k - 1, n)$ gives a nontrivial divisor of n.

So, if the decryption exponent leaks out, changing only e and d is not enough.

Integer Factorization

Example

- Suppose $n = 667, e = 39, d = 79$. We have $(39 \times 79) - 1 = 2^3 \times 385$.
- First select $b = 3$, so $\gcd(3, 667) = 1$.

Integer Factorization

Example

- Suppose $n = 667, e = 39, d = 79$. We have $(39 \times 79) - 1 = 2^3 \times 385$.
- First select $b = 3$, so $\gcd(3, 667) = 1$.
- We have

$$b_0 = 3^{385} \equiv 162 \mod 667$$

Integer Factorization

Example

- Suppose $n = 667, e = 39, d = 79$. We have $(39 \times 79) - 1 = 2^3 \times 385$.
- First select $b = 3$, so $\gcd(3, 667) = 1$.
- We have

$$\begin{aligned} b_0 &= 3^{385} \equiv 162 \mod 667 \\ b_1 &= b_0^2 \equiv 231 \mod 667 \end{aligned}$$

Integer Factorization

Example

- Suppose $n = 667, e = 39, d = 79$. We have $(39 \times 79) - 1 = 2^3 \times 385$.
- First select $b = 3$, so $\gcd(3, 667) = 1$.
- We have

$$\begin{aligned} b_0 &= 3^{385} &\equiv 162 \quad &\mod 667 \\ b_1 &= b_0^2 &\equiv 231 \quad &\mod 667 \\ b_2 &= b_1^2 &\equiv 1 \quad &\mod 667 \end{aligned}$$

- We have $b_2 \equiv 1 \mod 667$ & $b_1 \not\equiv \pm 1 \mod 667$.

Integer Factorization

Example

- Suppose $n = 667, e = 39, d = 79$. We have $(39 \times 79) - 1 = 2^3 \times 385$.
- First select $b = 3$, so $\gcd(3, 667) = 1$.
- We have

$$
\begin{aligned}
b_0 &= 3^{385} \equiv 162 \mod 667 \\
b_1 &= b_0^2 \equiv 231 \mod 667 \\
b_2 &= b_1^2 \equiv 1 \mod 667
\end{aligned}
$$

- We have $b_2 \equiv 1 \mod 667$ & $b_1 \not\equiv \pm 1 \mod 667$.
- $$\gcd(b_1 - 1, 667) = (230, 667)$$

Integer Factorization

Example

- Suppose $n = 667, e = 39, d = 79$. We have $(39 \times 79) - 1 = 2^3 \times 385$.
- First select $b = 3$, so $\gcd(3, 667) = 1$.
- We have

$$\begin{aligned} b_0 &= 3^{385} &\equiv& 162 \mod 667 \\ b_1 &= b_0^2 &\equiv& 231 \mod 667 \\ b_2 &= b_1^2 &\equiv& 1 \mod 667 \end{aligned}$$

- We have $b_2 \equiv 1 \mod 667$ & $b_1 \not\equiv \pm 1 \mod 667$.
- $$\gcd(b_1 - 1, 667) = (230, 667) = 23 \Rightarrow 667 = 23 \times 29$$

Integer Factorization

Pollard's $p - 1$ method

- It works if $p \mid n$ and $p - 1$ has only small prime factors.

Integer Factorization

Pollard's $p-1$ method

- It works if $p \mid n$ and $p-1$ has only small prime factors.
- Choose an integer $a > 1$; let $a = 2$.
- We choose a bound B and compute $b \equiv a^{B!} \mod n$
- If $p-1$ has only small prime factors. Then $B!$ is likely to be divisible by $p-1$, say $B! = (p-1)k$. We have

$$b \equiv a^{B!} \equiv \left(a^{p-1}\right)^k \equiv 1 \mod p$$

Integer Factorization

Pollard's $p - 1$ method

- It works if $p \mid n$ and $p - 1$ has only small prime factors.
- Choose an integer $a > 1$; let $a = 2$.
- We choose a bound B and compute $b \equiv a^{B!} \mod n$
- If $p - 1$ has only small prime factors. Then $B!$ is likely to be divisible by $p - 1$, say $B! = (p - 1)k$. We have

$$b \equiv a^{B!} \equiv \left(a^{p-1}\right)^k \equiv 1 \mod p \Rightarrow \gcd(b - 1, n) = p$$

Pollard's $p-1$ method

Algorithm

Input: Integer n to be factored

1. Set $a = 2$ (or some other convenient value)
2. For$\{j = 2, 3, 4, \ldots$ up to a specified bound.$\}\{$
   - (i) Set $a \equiv a^j \mod n$
   - (ii) Compute $d \equiv \gcd(a-1, n)$
   - (iii) If $1 < d < n$ then success, *return* d.

   $\}$
3. Increment j and loop again at Step 2.

Integer Factorization

Example

Factor $n = 13927189$ starting with $\gcd(2^{9!} - 1, n)$

Integer Factorization

Example

Factor $n = 13927189$ starting with $\gcd(2^{9!} - 1, n)$

$$
\begin{array}{rcll|rcl}
2^{9!} - 1 & \equiv & 13867883 & \bmod 13927189, & \gcd(2^{9!} - 1, 13927189) & = & 1, \\
2^{10!} - 1 & \equiv & 5129508 & \bmod 13927189, & \gcd(2^{10!} - 1, 13927189) & = & 1, \\
2^{11!} - 1 & \equiv & 4405233 & \bmod 13927189, & \gcd(2^{11!} - 1, 13927189) & = & 1, \\
2^{12!} - 1 & \equiv & 6680550 & \bmod 13927189, & \gcd(2^{12!} - 1, 13927189) & = & 1, \\
2^{13!} - 1 & \equiv & 6161077 & \bmod 13927189, & \gcd(2^{13!} - 1, 13927189) & = & 1, \\
2^{14!} - 1 & \equiv & 879290 & \bmod 13927189, & \gcd(2^{14!} - 1, 13927189) & = & 3823.
\end{array}
$$

Integer Factorization

Example

Factor $n = 13927189$ starting with $\gcd(2^{9!} - 1, n)$

$$\begin{array}{rcll|rcl}
2^{9!} - 1 & \equiv & 13867883 & \bmod 13927189, & \gcd(2^{9!} - 1, 13927189) & = & 1, \\
2^{10!} - 1 & \equiv & 5129508 & \bmod 13927189, & \gcd(2^{10!} - 1, 13927189) & = & 1, \\
2^{11!} - 1 & \equiv & 4405233 & \bmod 13927189, & \gcd(2^{11!} - 1, 13927189) & = & 1, \\
2^{12!} - 1 & \equiv & 6680550 & \bmod 13927189, & \gcd(2^{12!} - 1, 13927189) & = & 1, \\
2^{13!} - 1 & \equiv & 6161077 & \bmod 13927189, & \gcd(2^{13!} - 1, 13927189) & = & 1, \\
2^{14!} - 1 & \equiv & 879290 & \bmod 13927189, & \gcd(2^{14!} - 1, 13927189) & = & 3823.
\end{array}$$

$p = 3823$ of n. Thus $q = \frac{n}{p} = \frac{13927189}{3823} = 3643$.

Factorization via Difference of Squares

$$X^2 - Y^2 = (X+Y)(X-Y).$$

Factorization via Difference of Squares

$$X^2 - Y^2 = (X + Y)(X - Y).$$

- Search for an integer b s/t $n + b^2$ is a perfect square, say equal to a^2.

Factorization via Difference of Squares

$$X^2 - Y^2 = (X + Y)(X - Y).$$

- Search for an integer b s/t $n + b^2$ is a perfect square, say equal to a^2.
- Then $n + b^2 = a^2$, so

$$n = a^2 - b^2 = (a + b)(a - b),$$

and we have found the factors of n.

Factorization via Difference of Squares

$$X^2 - Y^2 = (X + Y)(X - Y).$$

- Search for an integer b s/t $n + b^2$ is a perfect square, say equal to a^2.
- Then $n + b^2 = a^2$, so

$$n = a^2 - b^2 = (a + b)(a - b),$$

  and we have found the factors of n.
- It is called Fermat factorisation method.

Factorization via Difference of Squares

Factor $n = 25217$ by looking for an integer b making $n + b^2$ a perfect square

Factorization via Difference of Squares

Factor $n = 25217$ by looking for an integer b making $n + b^2$ a perfect square

Example

$$\begin{aligned}
25217 + 1^2 &= 25218 && \textit{not a square},\\
25217 + 2^2 &= 25221 && \textit{not a square},\\
25217 + 3^2 &= 25226 && \textit{not a square},\\
25217 + 4^2 &= 25233 && \textit{not a square},\\
25217 + 5^2 &= 25242 && \textit{not a square},\\
25217 + 6^2 &= 25253 && \textit{not a square},\\
25217 + 7^2 &= 25266 && \textit{not a square},
\end{aligned}$$

Factorization via Difference of Squares

Factor $n = 25217$ by looking for an integer b making $n + b^2$ a perfect square

Example

$$\begin{array}{rcll}
25217 + 1^2 & = & 25218 & \quad \textit{not a square}, \\
25217 + 2^2 & = & 25221 & \quad \textit{not a square}, \\
25217 + 3^2 & = & 25226 & \quad \textit{not a square}, \\
25217 + 4^2 & = & 25233 & \quad \textit{not a square}, \\
25217 + 5^2 & = & 25242 & \quad \textit{not a square}, \\
25217 + 6^2 & = & 25253 & \quad \textit{not a square}, \\
25217 + 7^2 & = & 25266 & \quad \textit{not a square}, \\
25217 + 8^2 & = & 25281 \; = 159^2 & \quad \textit{Eureka!}
\end{array}$$

Factorization via Difference of Squares

Factor $n = 25217$ by looking for an integer b making $n + b^2$ a perfect square

Example

$$\begin{array}{rclll} 25217 + 1^2 & = & 25218 & & \textit{not a square}, \\ 25217 + 2^2 & = & 25221 & & \textit{not a square}, \\ 25217 + 3^2 & = & 25226 & & \textit{not a square}, \\ 25217 + 4^2 & = & 25233 & & \textit{not a square}, \\ 25217 + 5^2 & = & 25242 & & \textit{not a square}, \\ 25217 + 6^2 & = & 25253 & & \textit{not a square}, \\ 25217 + 7^2 & = & 25266 & & \textit{not a square}, \\ 25217 + 8^2 & = & 25281 & = 159^2 & \textit{Eureka}! \end{array}$$

Then we compute

$$25217 = 159^2 - 8^2 = (159 + 8)(159 - 8) = 167 \times 151.$$

Factorization via Difference of Squares

- If n is large, then it is unlikely that a randomly chosen value of b will make $n + b^2$ into a perfect square.

Factorization via Difference of Squares

- If n is large, then it is unlikely that a randomly chosen value of b will make $n + b^2$ into a perfect square.
- It often suffices to write some multiple kn of n as a difference of 2 squares, since if

$$kn = a^2 - b^2 = (a + b)(a - b),$$

then there is a reasonable chance that the factors of n are separated by the right-hand side of the equation.
- n has a nontrivial factor in common with each of $a + b$ and $a - b$.
- Recover the factors by computing $\gcd(n, a + b)$ & $\gcd(n, a - b)$.

Dixon's Factorization Method

- In 1981, John D. Dixon developed this method.
- **The Idea:**
  - Generate a large number of integer pairs (x, y) s/t

$$x^2 \equiv y^2 \mod n,$$

    where $x \neq \pm y \mod n$
  - $x^2 \mod n$ and $y^2 \mod n$ can be completely factorized over the chosen factor base.

Dixon's Factorization Method

- In 1981, John D. Dixon developed this method.
- **The Idea:**
  - Generate a large number of integer pairs (x, y) s/t

$$x^2 \equiv y^2 \mod n,$$

    where $x \neq \pm y \mod n$
  - $x^2 \mod n$ and $y^2 \mod n$ can be completely factorized over the chosen factor base.

Definition

A positive integer is called B-smooth if none of its prime factors is greater than B.

Dixon's Factorization Method

- In 1981, John D. Dixon developed this method.
- **The Idea:**
  - Generate a large number of integer pairs (x, y) s/t
  $$x^2 \equiv y^2 \mod n,$$
  where $x \neq \pm y \mod n$
  - $x^2 \mod n$ and $y^2 \mod n$ can be completely factorized over the chosen factor base.

Definition

A positive integer is called B-smooth if none of its prime factors is greater than B.

Example

- $720 = 2^4 \times 3^2 \times 5^1$; thus 720 is 5-smooth

Dixon's Factorization Method

Example

Factor $n = 84923$ using bound $B = 7$

- Randomly search for integers between $4\lceil \sqrt{n} \rceil = 292$ and n whose squares are B-smooth
-

$$513^2 \mod n = 8400 =$$

Dixon's Factorization Method

Example

Factor $n = 84923$ using bound $B = 7$

- Randomly search for integers between $4\lceil\sqrt{n}\rceil = 292$ and n whose squares are B-smooth
-

$$513^2 \mod n = 8400 = \; = 2^4 \times 3^1 \times 5^2 \times 7^1$$

$$537^2 \mod n = 33600 =$$

Dixon's Factorization Method

Example

Factor $n = 84923$ using bound $B = 7$

- Randomly search for integers between $4\lceil\sqrt{n}\rceil = 292$ and n whose squares are B-smooth
- $$513^2 \mod n = 8400 = \ = 2^4 \times 3^1 \times 5^2 \times 7^1$$
  $$537^2 \mod n = 33600 = 2^6 \times 3^1 \times 5^2 \times 7$$
- $(513 \times 537)^2 \mod n = 2^{10} \times 3^2 \times 5^4 \times 7^2 = \left(2^5.3.5^2.7\right)^2 = (16800)^2$
  $\Rightarrow (275481)^2 \equiv (16800)^2 \mod 84923 \Rightarrow (20712)^2 \equiv (16800)^2$

Dixon's Factorization Method

Example

Factor $n = 84923$ using bound $B = 7$

- Randomly search for integers between $4\lceil\sqrt{n}\rceil = 292$ and n whose squares are B-smooth
-
$$513^2 \mod n = 8400 = \; = 2^4 \times 3^1 \times 5^2 \times 7^1$$
$$537^2 \mod n = 33600 = 2^6 \times 3^1 \times 5^2 \times 7$$
- $(513 \times 537)^2 \mod n = 2^{10} \times 3^2 \times 5^4 \times 7^2 = \left(2^5.3.5^2.7\right)^2 = (16800)^2$ $\Rightarrow (275481)^2 \equiv (16800)^2 \mod 84923 \Rightarrow (20712)^2 \equiv (16800)^2$
- $84923 = \gcd(20712 - 16800, 84923) \times \gcd(20712 + 16800, 84923)$ $= 163 \times 521$

A Bad Way to Solve DLP

Problem

Find x s/t $y \equiv g^x \mod p$

A Bad Way to Solve DLP

Problem

Find x *s/t* $y \equiv g^x \mod p$

Solution

- ***Input:*** y
- *For* $x = 0$ *to* $p - 1$
    - *Compute* g^x
    - *If* $g^x \equiv y \mod p$ *then output*(x) *and* ***STOP***

A Bad Way to Solve DLP

Problem

Find x *s/t* $y \equiv g^x \mod p$

Solution

- ***Input:*** y
- *For* $x = 0$ *to* $p - 1$
  - *Compute* g^x
  - *If* $g^x \equiv y \mod p$ *then output*(x) *and* ***STOP***

The worst case $\approx p$ *steps*

Shanks's Babystep-Giantstep Algorithm

DLP

Find $g^x \equiv h \mod p$ in $\mathcal{O}(\sqrt{p}.\log p)$ steps using $\mathcal{O}(\sqrt{p})$ storage.

Shanks's Babystep-Giantstep Algorithm

DLP

Find $g^x \equiv h \mod p$ in $\mathcal{O}(\sqrt{p}.\log p)$ steps using $\mathcal{O}(\sqrt{p})$ storage.

1. Let $m = 1 + \lfloor\sqrt{p}\rfloor$, so in particular, $m > \sqrt{p}$.

Shanks's Babystep-Giantstep Algorithm

DLP

Find $g^x \equiv h \mod p$ in $\mathcal{O}(\sqrt{p}.\log p)$ steps using $\mathcal{O}(\sqrt{p})$ storage.

1. Let $m = 1 + \lfloor \sqrt{p} \rfloor$, so in particular, $m > \sqrt{p}$.
2. Create two lists,

   List 1: $e, g, g^2, g^3, \ldots, g^m,$

   List 2: $h, h.g^{-m}, h.g^{-2m}, h.g^{-3m}, \ldots, h.g^{-m^2}.$

Shanks's Babystep-Giantstep Algorithm

DLP

Find $g^x \equiv h \mod p$ in $\mathcal{O}(\sqrt{p}.\log p)$ steps using $\mathcal{O}(\sqrt{p})$ storage.

1. Let $m = 1 + \lfloor \sqrt{p} \rfloor$, so in particular, $m > \sqrt{p}$.
2. Create two lists,

   List 1: $e, g, g^2, g^3, \ldots, g^m,$

   List 2: $h, h.g^{-m}, h.g^{-2m}, h.g^{-3m}, \ldots, h.g^{-m^2}.$
3. Find a match between the 2 lists, say $g^i = h.g^{-j.m}$.

Shanks's Babystep-Giantstep Algorithm

DLP

Find $g^x \equiv h \mod p$ in $\mathcal{O}(\sqrt{p}.\log p)$ steps using $\mathcal{O}(\sqrt{p})$ storage.

1. Let $m = 1 + \lfloor \sqrt{p} \rfloor$, so in particular, $m > \sqrt{p}$.
2. Create two lists,

   List 1: $e, g, g^2, g^3, \ldots, g^m,$

   List 2: $h, h.g^{-m}, h.g^{-2m}, h.g^{-3m}, \ldots, h.g^{-m^2}.$
3. Find a match between the 2 lists, say $g^i = h.g^{-j.m}$.
4. Then $x = i + j.m$ is a solution to $g^x = h$.

Shanks's Babystep-Giantstep Algorithm

Example

Solve the discrete logarithm problem $g^x = h$ in $\mathbb{F}_p^*$ with $g = 9704, h = 13896,\ \&\ p = 17389$.

Shanks's Babystep-Giantstep Algorithm

Example

Solve the discrete logarithm problem $g^x = h$ in $\mathbb{F}_p^*$ with $g = 9704, h = 13896,\ \&\ p = 17389$.

- The number 9704 has order[a] 1242 in $\mathbb{F}_{17389}^*$.
- Set $m = 1 + \lfloor \sqrt{1242} \rfloor = 36$ and

  $u = g^{-m} = 9704^{-36} \equiv 2494 \mod 17389$.

[a] Lagrange's theorem says that the order of g divides $17388 = 2^2.3^3.7.23$. So we can determine the order of g by computing g^n for the 48 distinct divisors of 17388

Shanks's Babystep-Giantstep Algorithm

Example

Solve the discrete logarithm problem $g^x = h$ in $\mathbb{F}_p^*$ with $g = 9704, h = 13896,\ \&\ p = 17389$.

k	g^k	$h \cdot u^k$	k	g^k	$h \cdot u^k$	k	g^k	$h \cdot u^k$	k	g^k	$h \cdot u^k$
1	9704	347	9	15774	16564	17	10137	10230	25	4970	12260
2	6181	13357	10	12918	11741	18	17264	3957	26	9183	6578
3	5763	12423	11	16360	16367	19	4230	9195	27	10596	7705
4	1128	13153	12	13259	7315	20	9880	13628	28	2427	1425
5	8431	7928	13	4125	2549	21	9963	10126	29	6902	6594
6	16568	1139	14	16911	10221	22	15501	5416	30	11969	12831
7	**14567**	6259	15	4351	16289	23	6854	13640	31	6045	4754
8	2987	12013	16	1612	4062	24	15680	5276	32	7583	**14567**

Shanks's Babystep-Giantstep Algorithm

Example

Solve the discrete logarithm problem $g^x = h$ in $\mathbb{F}_p^*$ with $g = 9704, h = 13896,\ \&\ p = 17389$.

k	g^k	$h \cdot u^k$	k	g^k	$h \cdot u^k$	k	g^k	$h \cdot u^k$	k	g^k	$h \cdot u^k$
1	9704	347	9	15774	16564	17	10137	10230	25	4970	12260
2	6181	13357	10	12918	11741	18	17264	3957	26	9183	6578
3	5763	12423	11	16360	16367	19	4230	9195	27	10596	7705
4	1128	13153	12	13259	7315	20	9880	13628	28	2427	1425
5	8431	7928	13	4125	2549	21	9963	10126	29	6902	6594
6	16568	1139	14	16911	10221	22	15501	5416	30	11969	12831
7	**14567**	6259	15	4351	16289	23	6854	13640	31	6045	4754
8	2987	12013	16	1612	4062	24	15680	5276	32	7583	**14567**

- Find the collision $9704^7 \equiv 14567 \equiv 13896.2494^{32} \mod 17389$
- Using the fact that $2494 \equiv 9704^{-36}$, we compute

Shanks's Babystep-Giantstep Algorithm

Example

Solve the discrete logarithm problem $g^x = h$ in $\mathbb{F}_p^*$ with $g = 9704, h = 13896,\ \&\ p = 17389$.

k	g^k	$h \cdot u^k$	k	g^k	$h \cdot u^k$	k	g^k	$h \cdot u^k$	k	g^k	$h \cdot u^k$
1	9704	347	9	15774	16564	17	10137	10230	25	4970	12260
2	6181	13357	10	12918	11741	18	17264	3957	26	9183	6578
3	5763	12423	11	16360	16367	19	4230	9195	27	10596	7705
4	1128	13153	12	13259	7315	20	9880	13628	28	2427	1425
5	8431	7928	13	4125	2549	21	9963	10126	29	6902	6594
6	16568	1139	14	16911	10221	22	15501	5416	30	11969	12831
7	**14567**	6259	15	4351	16289	23	6854	13640	31	6045	4754
8	2987	12013	16	1612	4062	24	15680	5276	32	7583	**14567**

- Find the collision $9704^7 \equiv 14567 \equiv 13896.2494^{32} \mod 17389$
- Using the fact that $2494 \equiv 9704^{-36}$, we compute

$$13896 \equiv 9704^7.2494^{-32} \equiv 9704^7 \left(9704^{-36}\right)^{-32} \equiv 9704^{1159}$$

Outline

Signature Scheme

Definition

A signature scheme is a five-tuple $(\mathcal{P}, \mathcal{A}, \mathcal{K}, \mathcal{S}, \mathcal{V})$, where the following conditions are satisfied:

Signature Scheme

Definition

A signature scheme is a five-tuple $(\mathcal{P}, \mathcal{A}, \mathcal{K}, \mathcal{S}, \mathcal{V})$, where the following conditions are satisfied:

(i) $\mathcal{P}$ is a finite set of possible messages

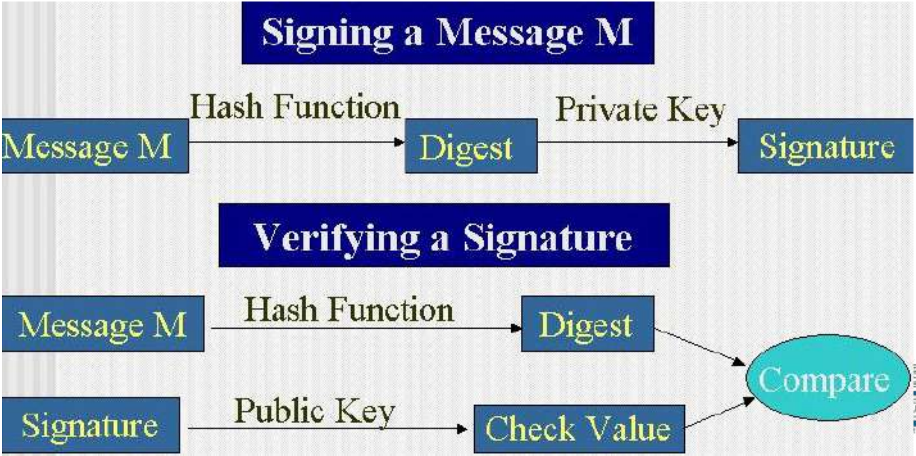
(ii) $\mathcal{A}$ is a finite set of possible signatures

(iii) $\mathcal{K}$, the keyspace, is a finite set of possible keys

(iv) For each $K \in \mathcal{K}$, there is a signing algorithm $sig_K \in \mathcal{S}$ and a corresponding verification algorithm $ver_K \in \mathcal{V}$. Each $sig_K : \mathcal{P} \to \mathcal{A}$ and $ver_K : \mathcal{P} \times \mathcal{A} \to \{true, \ false\}$ are functions s/t the following equation is satisfied for every message $x \in \mathcal{P}$ and for every signature $y \in \mathcal{A}$

$$ver_K = \begin{cases} \text{true} & \text{if} \quad y = sig_K(x) \\ \text{false} & \text{if} \quad y \neq sig_K(x) \end{cases}$$

A pair (x, y) with $x \in \mathcal{P}$ and $y \in \mathcal{A}$ is called a signed message.


Signing a Message M
Message M
Hash Function
Digest
Private Key
Signature
Verifying a Signature
Message M
Hash Function
Digest
Compare
Signature
Public Key
Check Value

RSA Signature Scheme

Signature Generation

A signs a message m. Any entity B can verify A's signature and recover the message m from the signature.

- Compute $\tilde{m} = R(m)$, where $R : \mathcal{M} \to \mathbb{Z}_n$.
- Compute $s \equiv \tilde{m}^d \mod n$.
- A's signature for m is s.

RSA Signature Scheme

Signature Generation

A signs a message m. Any entity B can verify A's signature and recover the message m from the signature.

- Compute $\tilde{m} = R(m)$, where $R : \mathcal{M} \rightarrow \mathbb{Z}_n$.
- Compute $s \equiv \tilde{m}^d \mod n$.
- A's signature for m is s.

Signature Verification

To verify A's signature s and recover the message m, B should:

- Obtain A's authentic public key (n, e).
- Compute $\tilde{m} \equiv s^e \mod n$.
- Verify that $\tilde{m} \in$ range of $\mathcal{M}$; if not, reject the signature.
- Recover $m = R^{-1}(\tilde{m})$.

DSA

Key Generation

1. Choose a hash function h.
2. Decide a key length L.
3. Choose prime q with with same number of bits as output of h.
4. Choose α-bit prime p such that $q|(p-1)$.
5. Choose g such that $g^q \equiv 1 \mod p$.

Choose x : $0 < x < q$.

Calculate : $y \equiv g^x \mod p$.

(p, q, g, y) $\longrightarrow$ Public Key

x $\longrightarrow$ Private Key

DSA

Signature Generation

1. Generate random k such that $0 < k < q$.
2. Calculate $r \equiv (g^k \mod p) \mod q$.
3. Calculate $s \equiv (k^{-1}(h(m) + xr)) \mod q$.
4. Signature is (r, s).

DSA

Signature Generation

1. Generate random k such that $0 < k < q$.
2. Calculate $r \equiv (g^k \mod p) \mod q$.
3. Calculate $s \equiv (k^{-1}(h(m) + xr)) \mod q$.
4. Signature is (r, s).

Signature Verification

1. $w \equiv s^{-1} \mod q$.
2. $u_1 \equiv (h(m).w) \mod q$.
3. $u_2 \equiv rw \mod q$.
4. $v \equiv (g^{u_1}.y^{u_2} \mod p) \mod q$.
5. Verify $v = r$.

Schnorr Signature Scheme

Key Generation

- Let p be a prime s/t the DLP in $\mathbb{Z}_p^*$ is intractable, and let q be a prime and $q \mid (p-1)$. Let $\alpha \in \mathbb{Z}_p^*$ be a q^{th} root of unity modulo p. Let $\mathcal{P} = \{0,1\}^*$, $\mathcal{A} = \mathbb{Z}_q \times \mathbb{Z}_q$, and define

$$\mathcal{K} = \{(p, q, \alpha, a, \beta) \ : \ \beta \equiv \alpha^a \mod p\},$$

  where $0 \le a \le q-1$.

  The values p, q, α, and β are the public key, and a is the private key.

  Finally, let $h \ : \ \{0,1\}^* \to \mathbb{Z}_q$ be a secure hash function.

Schnorr Signature Scheme

Signature Generation

- Signer first selects a (secret) random number $k, \ 1 \leq k \leq q-1$, define

$$sig_K(x,k) = (\gamma, \delta),$$

where

$$\gamma = h(x\|\alpha^k \ mod \ p) \ \& \ \delta = k + a\gamma \ mod \ q.$$

Verification

- For $x \in \{0,1\}^*$ and $\gamma, \delta \in \mathbb{Z}_q$, verification is done by performing the following computations:

$$ver_K(x, (\gamma, \delta)) = true \iff h(x\|\alpha^{\delta}\beta^{-\gamma} \ mod \ p) = \gamma.$$

W Diffie & M Hellman,
New Directions in Cryptography, IEEE Transactions on Information Theory, 22(6), 1976.

J. Hoffstein, J. Pipher & J. H. Silverman,
An Introduction to Mathematical Cryptography, Second Edition, Springer, 2014.

J. Katz & Y. Lindell,
Introduction to Modern Cryptography, CRC Press, 2021.

Neal Koblitz,
A Course in Number Theory and Cryptography, Springer- Verlag, 1994.

A. Menezes, P. Oorschot & S. Vanstone,
Handbook of Applied Cryptography, CRC Press, 1997, Available Online at
`http://www.cacr.math.uwateroo.ca/hac/`

D. R. Stinson & M. B. Paterson,
Cryptography - Theory and Practice, Chapman & Hall/CRC, 2019.

The End

Thanks a lot for your attention!