Public Key Cryptography

Dhananjoy Dey

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

March 9, 2021

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

1. Introduction to Public Key Cryptography
2. Requirements to Design a PKC
3. Origin of PKC
   - Diffie Hellman Key Exchange Protocol
   - Non-secret Encryption
4. PKC
   - RSA
   - ElGamal
   - Elliptic Curve
5. Digital Signature
   - Digital Signature Algorithm (DSA)

Outline

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 almost every $k \in \mathcal{K}$, each easily computed algorithm equivalent to D_k is computationally infeasible to derive from E_k,

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

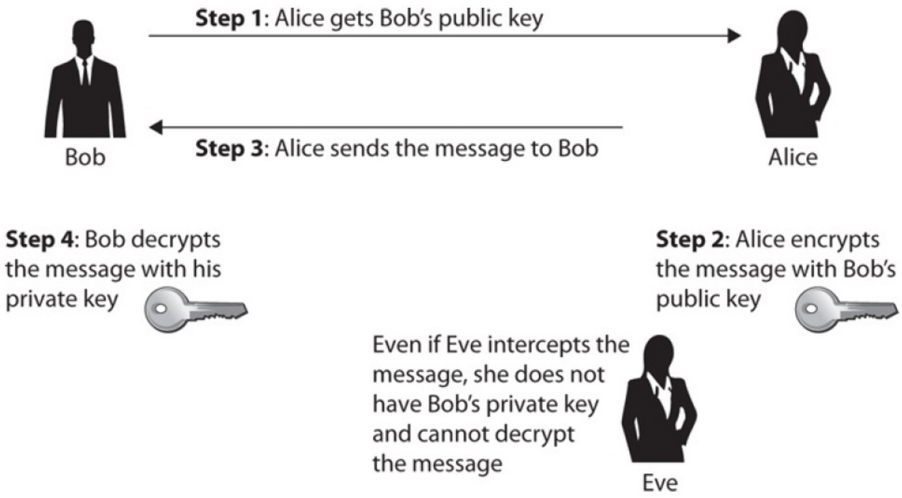
Definition

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
Step 3: Alice sends the message to Bob
Alice
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

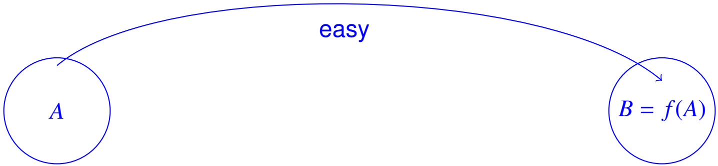
Advantages of PKC

Advantages over symmetric-key

1. Better key distribution and management
   - No danger that public key compromised
   - Convert authenticated channel to secure channel in interactive setting
2. New protocols
   - Digital Signature
3. Long-term encryption

Outline

One-way Function



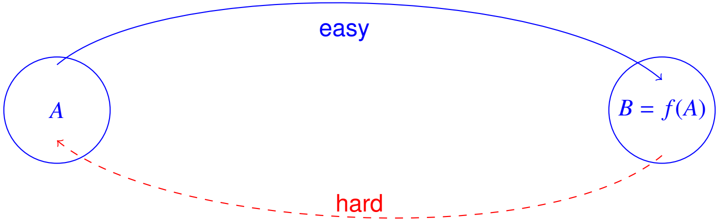
Definition

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

- Cryptographic hash functions, viz., RIPEMD-160, SHA-2 family and SHA-3 (Keccak).
- 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 $1 \leq i \leq 5$.

Trapdoor One-way Function

Definition

A trapdoor one-way function is a one-way function $f : \mathcal{M} \to \mathcal{C}$, 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 $n = p_1^{e_1} p_2^{e_2} \ldots p_k^{e_k}$ where the p_i are pairwise distinct primes and each $e_i \geq 0$ for $1 \leq i \leq k$. $\rightarrow$ **hard problem**.

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

- **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

- **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**.

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

- **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 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**.

विद्या ददाति विनयं विनयाद् याति पात्रताम्

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

Both parties know p and g

Alice

1. Alice generates a
2. Alice's public value is $g^a \bmod p$
3. Alice computes $g^{ab} = (g^b)^a \bmod p$

Bob

1. Bob generates b
2. Bob's public value is $g^b \bmod p$
3. Bob computes $g^{ba} = (g^a)^b \bmod p$

Since $g^{ab} = g^{ba}$ they now have a shared secret key usually called k ($K = g^{ab} = g^{ba}$)

DH Key Exchange

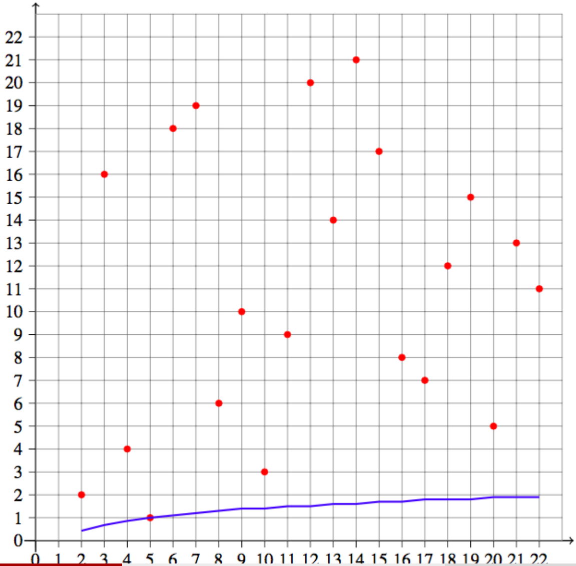
- k is the shared secret key.
- Knowing $g,\ g^a\ \&\ g^b$, it is hard to find g^{ab}.
- **Idea of this protocol:** The enciphering key can be made public since it is computationally infeasible to obtain the deciphering key from enciphering key.
- This protocol was (supposed to be) the door-opener to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

विद्या ददाति विनयं विनयाद् याति पात्रताम्

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 $\bmod 23$ to The Base 5



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

Clifford Cocks, Malcolm Williamson, and James Ellis.

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

Chinese Remainder Theorem

Theorem

Suppose $m_1, m_2, \cdots, m_r \in \mathbb{Z}^+ \ : \ gcd(m_i, m_j) = 1$ *for* $i \neq j$.
Then $x \equiv a_i \mod m_i$ *has ! solution* $\mod M(= \prod_{i=1}^{r} m_i)$, *which is given by*

$$x \equiv \sum_{i=1}^{r} a_i.M_i.y_i \mod M,$$

where $M_i = \frac{M}{m_i}$ & $y_i = M_i^{-1} \mod m_i$ *for* $1 \leq i \leq r$.

Chinese Remainder Theorem

Problem

Find x *s/t*

$x \equiv 5 \mod 7$, $x \equiv 3 \mod 11$, $x \equiv 10 \mod 13$

Chinese Remainder Theorem

Problem

Find x *s/t*

$x \equiv 5 \mod 7$, $x \equiv 3 \mod 11$, $x \equiv 10 \mod 13$

Solution

1. *First we calculate* $M = 7 \times 11 \times 13 = 1001$

Chinese Remainder Theorem

Problem

Find x *s/t*

$x \equiv 5 \mod 7, x \equiv 3 \mod 11, x \equiv 10 \mod 13$

Solution

1. *First we calculate* $M = 7 \times 11 \times 13 = 1001$
2. *After that we compute* M_1, M_2, M_3
   $M_1 = \frac{M}{7} = 11 \times 13 = 143$
   $M_2 = \frac{M}{11} = 7 \times 13 = 91$
   $M_3 = \frac{M}{13} = 7 \times 11 = 77$

Chinese Remainder Theorem

Problem

Find x *s/t*

$x \equiv 5 \mod 7$, $x \equiv 3 \mod 11$, $x \equiv 10 \mod 13$

Solution

1. *First we calculate* $M = 7 \times 11 \times 13 = 1001$
2. *After that we compute* M_1, M_2, M_3
   $M_1 = \frac{M}{7} = 11 \times 13 = 143$
   $M_2 = \frac{M}{11} = 7 \times 13 = 91$
   $M_3 = \frac{M}{13} = 7 \times 11 = 77$
3. *Now we have to compute*
   $(143)^{-1} \mod 7 \equiv 3^{-1} \mod 7 \equiv 5 \mod 7$
   $(91)^{-1} \mod 11 \equiv 3^{-1} \mod 11 \equiv 4 \mod 11$
   $77^{-1} \mod 13 \equiv (12)^{-1} \mod 13 \equiv 12 \mod 13$

Chinese Remainder Theorem

Solution

4. *So, the solution for x is given below:*

$$
\begin{aligned}
x &\equiv [(5 \times 143 \times 5) + (3 \times 91 \times 4) + (10 \times 77 \times 12)] \div 1001 \\
&\equiv [3575 + 1092 + 9240] \mod 1001 \\
&\equiv 13907 \mod 1001
\end{aligned}
$$

विद्या ददाति विनयं विनयाद् याति पात्रताम्

Chinese Remainder Theorem

Solution

4. *So, the solution for x is given below:*

$$
\begin{aligned}
x &\equiv [(5 \times 143 \times 5) + (3 \times 91 \times 4) + (10 \times 77 \times 12)] \div 1001 \\
&\equiv [3575 + 1092 + 9240] \mod 1001 \\
&\equiv 13907 \mod 1001 \\
&\equiv 894 \mod 1001
\end{aligned}
$$

Modular Arithmetic

Euclidean algorithm for computing the $gcd(a, b)$

Input: 2 non-negative integers a & b, with $a \geq b$.

Output: $gcd(a, b)$

1. While $(b \neq 0)$ do
   1.1 Set $r \leftarrow a \bmod b$, $a \leftarrow b,\ b \leftarrow r$.
2. *Return*(a)

Modular Arithmetic

Euclidean algorithm for computing the $gcd(a, b)$

Input: 2 non-negative integers a & b, with $a \geq b$.
Output: $gcd(a, b)$

1. While $(b \neq 0)$ do
   1.1 Set $r \leftarrow a \ mod \ b$, $a \leftarrow b, \ b \leftarrow r$.
2. *Return*(a)

$gcd(4864, 3458)$

Modular Arithmetic

Euclidean algorithm for computing the $gcd(a, b)$

Input: 2 non-negative integers a & b, with $a \geq b$.

Output: $gcd(a, b)$

1. While $(b \neq 0)$ do
   1.1 Set $r \leftarrow a \bmod b$, $a \leftarrow b,\ b \leftarrow r$.
2. *Return*(a)

$gcd(4864, 3458)$

$$\begin{aligned}
4864 &= 1.3458 + 1406 \\
3458 &= 2.1406 + 646 \\
1406 &= 2.646 + 114 \\
646 &= 5.114 + 76 \\
114 &= 1.76 + 38 \\
76 &= 2.38 + 0.
\end{aligned}$$

Modular Arithmetic

Euclidean algorithm for computing the $gcd(a, b)$

Input: 2 non-negative integers a & b, with $a \geq b$.

Output: $gcd(a, b)$

1. While $(b \neq 0)$ do
   1.1 Set $r \leftarrow a \bmod b$, $a \leftarrow b$, $b \leftarrow r$.
2. *Return*(a)

$gcd(4864, 3458)$

$$
\begin{aligned}
4864 &= 1.3458 + 1406 \\
3458 &= 2.1406 + 646 \\
1406 &= 2.646 + 114 \\
646 &= 5.114 + 76 \\
114 &= 1.76 + 38 \\
76 &= 2.38 + 0.
\end{aligned}
$$

Bezout's Lemma

$\forall\ a, b \in \mathbb{Z},\ \exists\ s, t \in \mathbb{Z}$ s/t $gcd(a, b) = s.a + t.b$

Modular Arithmetic

Extended Euclidean algorithm

Input: 2 non-negative integers a & b, with $a \geq b$.
Output: $d = gcd(a, b)$ & $x, y \in \mathbb{Z}$ s/t $ax + by = d$.

1. If $b = 0$ then set $d \leftarrow a,\ x \leftarrow 1,\ y \leftarrow 0$, and $return(d, x, y)$.
2. Set $x_2 \leftarrow 1,\ x_1 \leftarrow 0,\ y_2 \leftarrow 0,\ y_1 \leftarrow 1$.
3. While $(b > 0)$ do
   - 3.1 $q \leftarrow \lfloor a/b \rfloor,\ r \leftarrow a - qb$,
     $x \leftarrow x_2 - qx_1,\ y \leftarrow y_2 - qy_1$.
   - 3.2 $a \leftarrow b,\ b \leftarrow r,\ x_2 \leftarrow x_1$,
     $x_1 \leftarrow x,\ y_2 \leftarrow y_1$, and $y_1 \leftarrow y$.
4. Set $d \leftarrow a,\ x \leftarrow x_2,\ y \leftarrow y_2$, and $return(d, x, y)$.

Modular Arithmetic

Extended Euclidean algorithm

Input: 2 non-negative integers a & b, with $a \geq b$.
Output: $d = gcd(a, b)$ & $x, y \in \mathbb{Z}$ s/t $ax + by = d$.

1. If $b = 0$ then set $d \leftarrow a,\ x \leftarrow 1,\ y \leftarrow 0$, and $return(d, x, y)$.
2. Set $x_2 \leftarrow 1,\ x_1 \leftarrow 0,\ y_2 \leftarrow 0,\ y_1 \leftarrow 1$.
3. While $(b > 0)$ do
   - 3.1 $q \leftarrow \lfloor a/b \rfloor,\ r \leftarrow a - qb$, $x \leftarrow x_2 - qx_1,\ y \leftarrow y_2 - qy_1$.
   - 3.2 $a \leftarrow b,\ b \leftarrow r,\ x_2 \leftarrow x_1$, $x_1 \leftarrow x,\ y_2 \leftarrow y_1$, and $y_1 \leftarrow y$.
4. Set $d \leftarrow a,\ x \leftarrow x_2,\ y \leftarrow y_2$, and $return(d, x, y)$.

$$a = 4864,\ b = 3458$$

Modular Arithmetic

Extended Euclidean algorithm

Input: 2 non-negative integers a & b, with $a \geq b$.
Output: $d = gcd(a, b)$ & $x, y \in \mathbb{Z}$ s/t $ax + by = d$.

1. If $b = 0$ then set $d \leftarrow a,\ x \leftarrow 1,\ y \leftarrow 0$, and $return(d, x, y)$.
2. Set $x_2 \leftarrow 1,\ x_1 \leftarrow 0,\ y_2 \leftarrow 0,\ y_1 \leftarrow 1$.
3. While $(b > 0)$ do
   - 3.1 $q \leftarrow \lfloor a/b \rfloor,\ r \leftarrow a - qb$, $x \leftarrow x_2 - qx_1,\ y \leftarrow y_2 - qy_1$.
   - 3.2 $a \leftarrow b,\ b \leftarrow r,\ x_2 \leftarrow x_1$, $x_1 \leftarrow x,\ y_2 \leftarrow y_1$, and $y_1 \leftarrow y$.
4. Set $d \leftarrow a,\ x \leftarrow x_2,\ y \leftarrow y_2$, and $return(d, x, y)$.

$a = 4864,\ b = 3458$

q	r	x	y	a	b	x_2	x_1	y_2	y
–	–	–	–	4864	3458	1	0	0	
1	1406	1	−1	3458	1406	0	1	1	–
2	646	−2	3	1406	646	1	−2	−1	
2	114	5	−7	646	114	−2	5	3	–
5	76	−27	38	114	76	5	−27	−7	3
1	38	32	−45	76	38	−27	32	38	−4
2	0	−91	128	38	0	32	−91	−45	12

$38 = 32.4864 - 45.3458$

Modular Arithmetic

Problem

Find $7^{-1} \mod 26$

Modular Arithmetic

Problem

Find $7^{-1} \mod 26$

Solution

1. *First we find the* $\gcd(7, 26)$

Modular Arithmetic

Problem

Find $7^{-1} \mod 26$

Solution

1. *First we find the* $\gcd(7, 26)$

$$\begin{aligned} 26 &= 3 \times 7 + 5 \\ 7 &= 1 \times 5 + 2 \\ 5 &= 2 \times 2 + 1 \\ 2 &= 1 \times 2 + 0 \end{aligned}$$

2. *Now, we compute the inverse of* $7 \mod 26$

$$\begin{aligned} 1 &= 5 - 2.2 \\ &= 5 - 2(7 - 5) \\ &= 3.5 - 2.7 \\ &= 3(26 - 3.7) - 2.7 \\ &= 3.26 - 11.7 \end{aligned}$$

Modular Arithmetic

Problem

Find $7^{-1} \mod 26$

Solution

1. *First we find the* $\gcd(7, 26)$

$$\begin{aligned} 26 &= 3 \times 7 + 5 \\ 7 &= 1 \times 5 + 2 \\ 5 &= 2 \times 2 + 1 \\ 2 &= 1 \times 2 + 0 \end{aligned}$$

2. *Now, we compute the inverse of* $7 \mod 26$

$$\begin{aligned} 1 &= 5 - 2.2 \\ &= 5 - 2(7 - 5) \\ &= 3.5 - 2.7 \\ &= 3(26 - 3.7) - 2.7 \\ &= 3.26 - 11.7 \\ 1 &\equiv (3.26 + 15.7) \mod 26 \end{aligned}$$

Modular Arithmetic

Problem

Find $7^{-1} \mod 26$

Solution

1. *First we find the* $\gcd(7, 26)$

$$\begin{aligned} 26 &= 3 \times 7 + 5 \\ 7 &= 1 \times 5 + 2 \\ 5 &= 2 \times 2 + 1 \\ 2 &= 1 \times 2 + 0 \end{aligned}$$

2. *Now, we compute the inverse of* $7 \mod 26$

$$\begin{aligned} 1 &= 5 - 2.2 \\ &= 5 - 2(7 - 5) \\ &= 3.5 - 2.7 \\ &= 3(26 - 3.7) - 2.7 \\ &= 3.26 - 11.7 \\ 1 &\equiv (3.26 + 15.7) \mod 26 \\ 1 &\equiv 15.7 \mod 26 \end{aligned}$$

$$15 \equiv 7^{-1} \mod 26$$

Non-secret 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).

Non-secret Encryption

Encryption

$$C \equiv M^n \mod n \quad for\ 0 \leq M < n.$$

Decryption

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

Modular Exponentiation by The Repeated Squaring I

Compute $b^n \mod m$

1. Use a to denote the partial product.
2. We'll have $a \equiv b^n \mod m$.
3. We start out with $a = 1$.
4. Let $n_0, n_1, \ldots n_{k-1}$ denote the binary digits of n, i.e.,

$$n = n_0 + 2n_1 + 4n_2 + \ldots + 2^{k-1}n_{k-1}.$$

5. If $n_0 = 1$, change a to b (otherwise keep $a = 1$).
   Then set $b_1 = b^2 \mod m$
6. If $n_1 = 1$, multiply a by b_1 (and reduce $\mod m$); otherwise keep a unchanged.
7. Next square b_1, and set $b_2 = b_1^2 \mod m$

Modular Exponentiation by The Repeated Squaring II

8. If $n_2 = 1$, multiply a by b_2 (and reduce $\mod m$); otherwise keep a unchanged.
9. Continue in this way. You see that in the j-th step you have computed $b_j \equiv b^{2^j} \mod m$.
10. If $n_j = 1$, i.e., if 2^j occurs in the binary expansion of n, then you include b_j in the product for a (if 2^j is absent from n, then you do not).
11. It is easy to see that after the $(k-1)$-st step you'll have the desired

$$a \equiv b^n \mod m.$$

$$\text{Time}(b^n \mod m) = O((\log n)(\log^2 m)).$$

Modular Exponentiation by The Repeated Squaring

Example

Let us compute $5^{100} \mod 33$.

Modular Exponentiation by The Repeated Squaring

Example

Let us compute $5^{100} \mod 33$.

$$
\begin{aligned}
5^1 &= 5 \\
5^2 &= 25 \\
5^4 &= 25 \times 25 \equiv 31 \mod 33 \\
5^8 &\equiv 31 \times 31 \equiv 4 \mod 33 \\
5^{16} &\equiv 4 \times 4 \equiv 16 \mod 33 \\
5^{32} &\equiv 16 \times 16 \equiv 25 \mod 33 \\
5^{64} &\equiv 25 \times 25 \equiv 31 \mod 33 \\
5^{96} &\equiv 31 \times 25 \equiv 16 \mod 33 \\
5^{100} &\equiv 16 \times 31 \equiv 1 \mod 33
\end{aligned}
$$

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 $\gcd(e, \phi(n)) = 1$.
- 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.

Strong Prime Number

Definition

A prime p is called a strong prime if

1. (i) $p-1$ *has a large prime factor, say* r*,*
2. (ii) $p+1$ *has a large prime factor, and*
3. (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

1. If p is a prime, then $\phi(p) = p - 1$.
2. The Euler phi function is multiplicative. That is, if $gcd(m, n) = 1$, then

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

3. 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

$$a^{p-1} \equiv 1 \; mod \; p.$$

- **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.*

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.*

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,*
$$\because 2^{90} \equiv 64 \mod 91.$$

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:*
   1. $1105 = 5 \times 13 \times 17$
   2. $1729 = 7 \times 13 \times 19$
   3. $2465 = 5 \times 17 \times 29$

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:*
    1. $1105 = 5 \times 13 \times 17$
    2. $1729 = 7 \times 13 \times 19$
    3. $2465 = 5 \times 17 \times 29$

RSA Validation

We have to prove $c^d \equiv m \mod n$

RSA Validation

We have to prove $c^d \equiv m \mod n$

First we see the $\gcd(m, n) =$

RSA Validation

We have to prove $c^d \equiv m \mod n$

First we see the $\gcd(m, n) = 1, p, q,$ or n

- **Case-1:** If $\gcd(m, n) = n$,

RSA Validation

We have to prove $c^d \equiv m \mod n$

First we see the $\gcd(m, n) = 1, p, q,$ or n

- **Case-1:** If $\gcd(m, n) = n$, $m = 0$.

$$c^d \equiv m^{ed} \equiv 0^{ed} \equiv 0 \equiv m \mod n.$$

RSA Validation

We have to prove $c^d \equiv m \mod n$

First we see the $\gcd(m, n) = 1, p, q,$ or n

- **Case-1:** If $\gcd(m, n) = n$, $m = 0$.

$$c^d \equiv m^{ed} \equiv 0^{ed} \equiv 0 \equiv m \mod n.$$

- **Case-2:** If $\gcd(m, n) = 1$

RSA Validation

We have to prove $c^d \equiv m \mod n$

First we see the $\gcd(m, n) = 1, p, q,$ or n

- **Case-1:** If $\gcd(m, n) = n$, $m = 0$.

$$c^d \equiv m^{ed} \equiv 0^{ed} \equiv 0 \equiv m \mod n.$$

- **Case-2:** If $\gcd(m, n) = 1$

$$c^d \equiv m^{ed} \equiv m.(m^{\phi(n)})^k \equiv m \; mod \; n$$

RSA Validation

We have to prove $c^d \equiv m \mod n$

First we see the $\gcd(m, n) = 1, p, q,$ or n

- **Case-1:** If $\gcd(m, n) = n$, $m = 0$.

$$c^d \equiv m^{ed} \equiv 0^{ed} \equiv 0 \equiv m \mod n.$$

- **Case-2:** If $\gcd(m, n) = 1$

$$c^d \equiv m^{ed} \equiv m.(m^{\phi(n)})^k \equiv m \; mod \; n$$

by using Euler's theorem $m^{\phi(n)} \equiv 1 \; mod \; n$.

RSA Validation

We have to prove $c^d \equiv m \mod n$

RSA Validation

We have to prove $c^d \equiv m \mod n$

- **Case-3:** If $\gcd(m,n) \neq 1$ and $\gcd(m,n) \neq n$
  - $p \mid m \ \& \ q \nmid m$.

$$\because p \mid m, \ \therefore m \equiv 0 \ mod \ p \Rightarrow m^{ed} \equiv 0 \equiv m \ mod \ p \Rightarrow p \mid (m^{ed} - m).$$

$$\because q \nmid m \Rightarrow \ gcd(m,q) = 1 \Rightarrow m^{q-1} \equiv 1 \ mod \ q.$$

$$\therefore m^{1+k.(p-1)(q-1)} \equiv m \ mod \ q \Rightarrow q \mid (m^{ed} - m).$$

$$\because gcd(p,q) = 1 \Rightarrow pq \mid (m^{ed} - m) \Rightarrow c^d \equiv m \ mod \ n.$$

RSA Validation

We have to prove $c^d \equiv m \mod n$

- **Case-3:** If $\gcd(m,n) \neq 1$ and $\gcd(m,n) \neq n$
  - $p \mid m \;\&\; q \nmid m$.

$$\because p \mid m,\ \therefore m \equiv 0\ mod\ p \Rightarrow m^{ed} \equiv 0 \equiv m\ mod\ p \Rightarrow p \mid (m^{ed} - m).$$

$$\because q \nmid m \Rightarrow\ gcd(m,q) = 1 \Rightarrow m^{q-1} \equiv 1\ mod\ q.$$

$$\therefore m^{1+k.(p-1)(q-1)} \equiv m\ mod\ q \Rightarrow q \mid (m^{ed} - m).$$

$$\because gcd(p,q) = 1 \Rightarrow pq \mid (m^{ed} - m) \Rightarrow c^d \equiv m\ mod\ n.$$

  - $p \nmid m \;\&\; q \mid m$. The proof for this case is same as the above by interchanging the role of $p \;\&\; q$.

Primality Testing – Probabilistic Algorithm

Pseudo-prime Test

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

Primality Testing – 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 ≤ a ≤ n − 2.
    Compute y ≡ a^r mod n
    if y ≠ 1 & y ≠ n − 1 then
        j ← 1.
        while j ≤ s − 1 & y ≠ n − 1 do
            Compute y ← y^2 mod n.
            If y = 1 then return("composite").
            j ← j + 1.
        end
        If y ≠ 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**

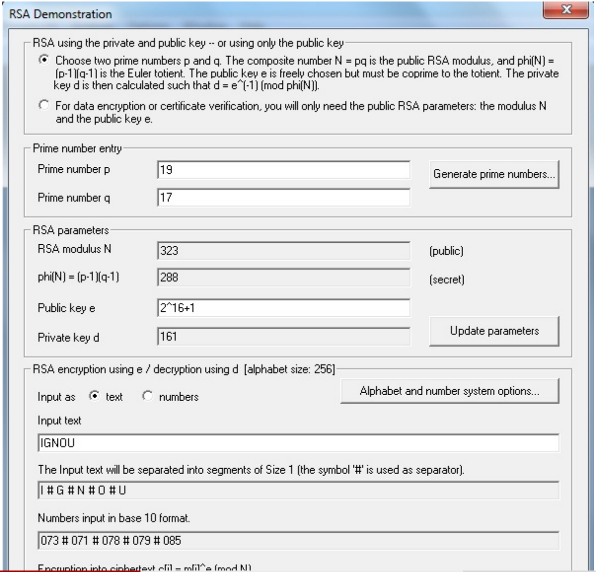
  if $(x - a)^n \not\equiv (x^n - a) \mod (x^r - 1, n)$,

  then output **COMPOSITE.**

end

Return("PRIME").

RSA Example


RSA Demonstration
RSA using the private and public key -- or using only the public key
Choose two prime numbers p and q. The composite number N = pq is the public RSA modulus, and phi(N) = (p-1)(q-1) is the Euler totient. The public key e is freely chosen but must be coprime to the totient. The private key d is then calculated such that d = e^(-1) (mod phi(N)).
For data encryption or certificate verification, you will only need the public RSA parameters: the modulus N and the public key e.
Prime number entry
Prime number p
19
Prime number q
17
Generate prime numbers...
RSA parameters
RSA modulus N
323
(public)
phi(N) = (p-1)(q-1)
288
(secret)
Public key e
2^16+1
Private key d
161
Update parameters
RSA encryption using e / decryption using d [alphabet size: 256]
Input as
text
numbers
Alphabet and number system options...
Input text
IGNOU
The Input text will be separated into segments of Size 1 (the symbol '#' is used as separator).
I # G # N # O # U
Numbers input in base 10 format.
073 # 071 # 078 # 079 # 085

RSA Example

```
                  GP/PARI CALCULATOR Version 2.6.1 (alpha)
          i686 running mingw (ix86/GMP-5.0.1 kernel) 32-bit version
               compiled: Sep 20 2013, gcc version 4.6.3 (GCC)
               (readline v6.2 enabled, extended help enabled)

                   Copyright (C) 2000-2013 The PARI Group

PARI/GP is free software, covered by the GNU General Public License, and comes
WITHOUT ANY WARRANTY WHATSOEVER.

Type ? for help, \q to quit.
Type ?12 for how to get moral (and possibly technical) support.

parisize = 4000000, primelimit = 500000
? N=323
%1 = 323
? e = 2^16+1
%2 = 65537
? 73^e%N
%3 = 158
? 71^e%N
%4 = 224
? 78^e%N
%5 = 10
? 79^e%N
%6 = 317
? 85^e%N
%7 = 17
? _
```

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 $= 2$.

$$m_1 = `RS' = 17 \times 26 + 18 = 460$$

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

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 $= 2$.
- $m_1 = `po' = 15 \times 26 + 14 = 404, \; m_2 = `we' = 22 \times 26 + 4 = 576, \; m_3 = `ra' = 17 \times 26 + 0 = 442$.
- $c_1 = 404^{701} \equiv 1419 \mod 1943 = 2.26^2 + 2.26 + 15 = ccp$.
- $\| ly, \; c_2 = 344 = 13.26 + 6 = ang$ & $c_3 = 210 = 8.26 + 2 = aic$.
- 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).
- 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).

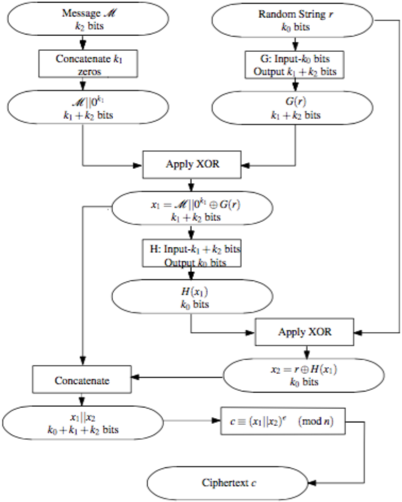
Choice of Encryption Key e

- 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.
- Suppose two of them, say $n_1 \ \& \ n_2$, are not coprime. Then, $\gcd(n_1, n_2)$ is a non-trivial factor of $n_1 \ \& \ n_2$ and any adversary can factorise both of them.
- So, we can always assume that n_i for $i = 1, 2, 3$ are pairwise coprime.
- If adversary gets hold of the messages $y_i, \ 1 \leq i \leq 3$, she can compute $M^3 \mod n_1 n_2 n_3$ using Chinese remainder theorem since $\gcd(n_i, n_j) = 1$ for $i \neq j$.
- Since $m < n_i, \ m^3 < n_1 n_2 n_3$. 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 $\mathbb{Z}_p^*$

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:

$$Dec_k(y_1,y_2)=y_2.(y_1^a)^{-1}.$$

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

Example

- *Let* $p = 29$ *and* $\alpha = 2$, α *is a primitive element* $\mod 29$.
- *Let* $a = 5, \therefore \beta \equiv 2^5 \mod \equiv 3 \mod 29$.

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).*

Elliptic Curves

- Elliptic curve 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.

Elliptic Curves I

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 $(4a^3 + 27b^2 \neq 0)$.
   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
   $$y^2 + cy = x^3 + ax + b$$
   or
   $$y^2 + xy = x^3 + ax + b$$
   together with the *point at infinity* O.

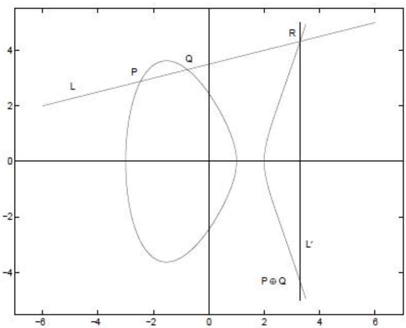
Elliptic Curves II

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

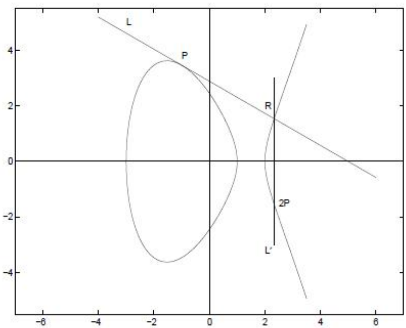
$$y^2 = x^3 + ax^2 + bx + c$$

together with the *point at infinity* O.

Addition Law on Elliptic Curves



Adding two points



Doubling a point

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

Addition Law on Elliptic Curves

- Suppose E is a non-singular 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.

$$P + Q = R.$$

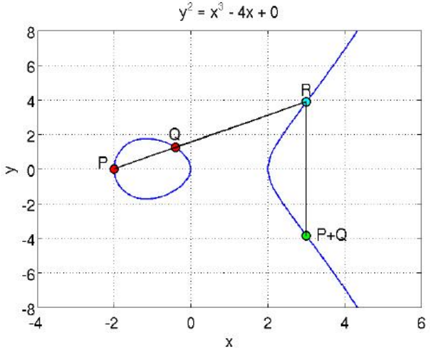
(ii) $x_1 = x_2 \,\&\, y_1 = -y_2$

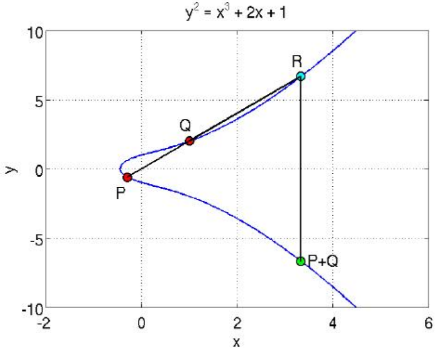
$$(x, y) + (x, -y) = \mathcal{O}$$

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

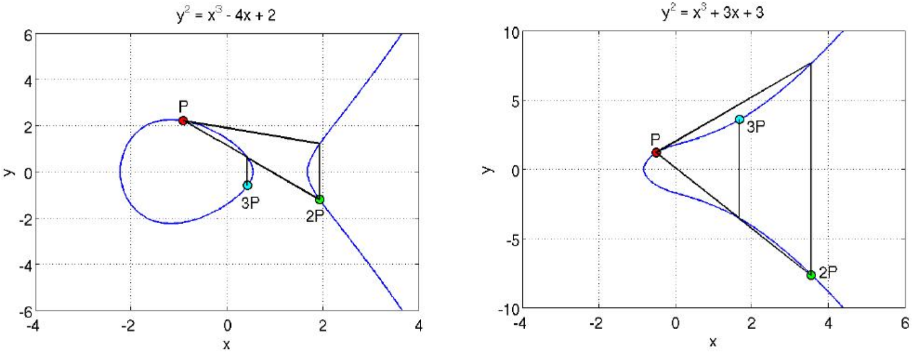
- Draw a tangent line L through P
- Follow step (i)

Addition Law on Elliptic Curves

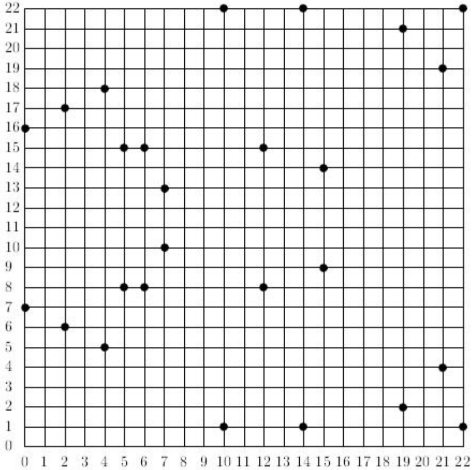




Addition Law on Elliptic Curves



Elliptic Curves over Finite Fields



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

NIST's Primes for ECC

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

NIST's Primes for ECC

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

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.

- 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, because there is no convenient method known of deterministically generating points on E.
- So, we convert the plaintext as an arbitrary element in $\mathbb{Z}_p$.
- For that, we can apply a suitable hash function $h : E \rightarrow \mathbb{Z}_p$ to kQ
- Add the result *modulo* p to$x$ in order to encrypt it.
- To decrypt, the private key s will allow kQ to be computed from kP.
- Then the result is hashed and *modulo* p from the ciphertext.

विद्या ददाति विनयं विनयाद् याति पात्रताम्

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 $\mathcal{C} = (\mathbb{Z}_p \times \mathbb{Z}_2) \times \mathbb{Z}_p$. 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 x sender selects a random number $k \in \mathbb{Z}_n^*$ and compute the ciphertext

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

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 x sender selects a random number $k \in \mathbb{Z}_n^*$ and compute the ciphertext

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

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 $R = sPOINT - DECOMPRESS(y_1)$.

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:

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

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

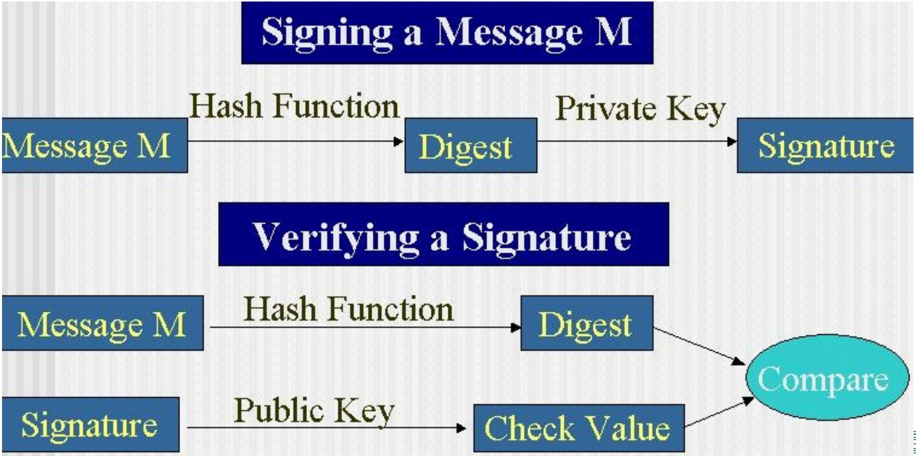
(iii) $\mathcal{K}$, the keyspace, is a finite set of possible keys

(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 $y \in \mathcal{A}$

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

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

Digital Signature



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.

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.

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 $0 \leq a \leq q-1$.

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

Finally, let $h \; : \; \{0,1\}^* \rightarrow \mathbb{Z}_q$ 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:

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

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, 2015. CRC Press, 2015.

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

and

QUESTIONS Please!