Implementation and Analysis of several Public-Key Encryption Algorithms

By
Sreekanth Yalamanchili

Public-Key Encryption Algorithms

1. Rabin public-key encryption
2. McEliece public-key encryption
3. Merkle-Hellman knapsack encryption
4. Goldwasser-Micali probabilistic encryption

Rabin public-key encryption

Key generation for Rabin public-key encryption

Each entity creates a public key and a corresponding private key.

Each entity A should do the following:

1. Generate two large random (and distinct) primes p and q, each roughly the same size.
2. Compute n = pq.
3. A’s public key is n; A’s private key is (p, q).

Rabin public-key encryption

B encrypts a message m for A, which A decrypts.

Encryption:B should do the following:

(a) Obtain A’s authentic public key n.

(b) Represent the message as an integer m in the range $\{0, 1, . . . , n - 1\}$.

(c) Compute $c = m^2 \bmod n$.

(d) Send the cipher-text c to A.

Rabin public-key encryption

Decryption: To recover plaintext m from c, A should do the following:

(a) Find the four square roots m1 , m2 , m3 , and m4 of c modulo n.

(b) The message sent was either m1 , m2 , m3 , or m4 . A somehow decides which of these is m.

Rabin public-key encryption

1. Use the extended Euclidean algorithm to find integers a and b satisfying $ap + bq = 1$. Note that a and b can be computed once and for all during the key generation stage.
2. Compute $r = c(p+1)/4 \bmod p$.
3. Compute $s = c(q+1)/4 \bmod q$.
4. Compute $x = (aps + bqr) \bmod n$.
5. Compute $y = (aps - bqr) \bmod n$.
6. The four square roots of c modulo n are x, $-x \bmod n$, y, and $-y \bmod n$.

McEliece public-key encryption

<u>*Key generation for McEliece public-key encryption*</u>

Each entity creates a public key and a corresponding private key.

1. Integers k, n, and t are fixed as common system parameters.
2. Each entity A should perform steps 3 – 7.
3. Choose a $k \times n$ generator matrix G for a binary (n, k)-linear code which can correct t errors, and for which an efficient decoding algorithm is known.
4. Select a random $k \times k$ binary non-singular matrix S.
5. Select a random $n \times n$ permutation matrix P .
6. Compute the $k \times n$ matrix G` = SGP .
7. A's public key is (G`, t); A's private key is (S, G, P).

McEliece public-key encryption

SUMMARY: B encrypts a message m for A, which A decrypts.

Encryption:B should do the following:

(a) Obtain A’s authentic public key (G`, t).

(b) Represent the message as a binary string m of length k.

(c) Choose a random binary error vector z of length n having at most t 1’s.

(d) Compute the binary vector c = mG` + z.

(e) Send the ciphertext c to A.

McEliece public-key encryption

Decryption:To recover plaintext m from c, A should do the following:

(a) Compute $c = cP^{-1}$, where P^{-1} is the inverse of the matrix P .

(b) Decode code generated by G to decode c\` to m\`.

(c) Compute $m = m`S^{-1}$.

Merkle-Hellman knapsack encryption

Key generation for basic Merkle-Hellman knapsack encryption

SUMMARY: Each entity creates a public key and a corresponding private key.

1. An integer n is fixed as a common system parameter.
2. Each entity A should perform steps 3 – 7.
3. Choose a superincreasing sequence $(b_1, b_2, \ldots, b_n)$ and modulus M such that $M > b_1 + b_2 + \cdots + b_n$.
4. Select a random integer W, $1 \leq W \leq M - 1$, such that $\gcd(W, M) = 1$.
5. Select a random permutation π of the integers $\{1, 2, \ldots, n\}$.
6. Compute $a_i = W b_{\pi(i)} \bmod M$ for $i = 1, 2, \ldots, n$.
7. A's public key is $(a_1, a_2, \ldots, a_n)$;

   A's private key is $(\pi, M, W, (b_1, b_2, \ldots, b_n))$.

Merkle-Hellman knapsack encryption

B encrypts a message m for A, which A decrypts.

Encryption:B should do the following:

(a) Obtain A’s authentic public key $(a_1, a_2, \ldots, a_n)$.

(b) Represent the message m as a binary string of length n, $m = m_1 m_2 \cdots m_n$.

(c) Compute the integer $c = m_1 a_1 + m_2 a_2 + \cdots + m_n a_n$.

(d) Send the ciphertext c to A.

Merkle-Hellman knapsack encryption

Decryption:To recover plaintext m from c, A should do the following:

(a) Compute $d = W^{-1} c \bmod M$.

(b) By solving a superincreasing subset sum problem , find integers $r_1 , r_2 , \ldots , r_n$, $r_i \in \{0, 1\}$, such that $d = r_1 b_1 + r_2 b_2 + \cdots + r_n b_n$.

(c) The message bits are $m_i = r_{\pi(i)}$, $i = 1, 2, \ldots , n$.

Merkle-Hellman knapsack encryption

Solving a superincreasing subset sum problem

INPUT: a superincreasing sequence (b_1 , b_2 , ... , b_n) and an integer s which is the sum of a subset of the b_i .

OUTPUT: (x_1 , x_2 , ... , x_n) where $x_i \in \{0, 1\}$, such that

```
for(i=0;i<=n)
    xi bi = s;
```

1. $i \leftarrow n$.
2. While $i \geq 1$ do the following:
   2.1 If $s \geq b_i$ then $x_i \leftarrow 1$ and $s \leftarrow s - b_i$. Otherwise $x_i \leftarrow 0$.
   2.2 $i \leftarrow i - 1$.
3. Return((x_1 , x_2 , ... , x_n)).

Goldwasser-Micali probabilistic encryption

B encrypts a message m for A, which A decrypts.

Encryption:B should do the following:

(a) Obtain A’s authentic public key n.

(b) Represent the message m as a binary string

$m = m_1 m_2 \cdot \cdot \cdot m_t$ of length t.

(c) For i from 1 to t do:

i. Pick an $r \in Z_n^*$ at random.

ii. If $m_i = 0$ then set $c_i \leftarrow r^2 \bmod n$;
otherwise set $c_i \leftarrow -r^{\wedge} \bmod n$.

(d) Send the t-tuple $c = (c_1 , c_2 , ... , c_t)$ to A.

Goldwasser-Micali probabilistic encryption

Decryption:

Decide whether c is a quadratic residue mod n.

A quadratic residue is a residue class that has square root mod n.

If c is a quadratic residue, then m=0.

Otherwise, m=1.

Analysis

From the results:

1. Rabin encryption is an extremely fast operation as it only involves a single modular squaring.
2. Rabin decryption is slower than encryption.
3. Although the encryption and decryption operations are relatively fast, the McEliece scheme suffers from the drawback that the publickey is very large.

Analysis(Contd...)

4.The encryption and decryption algorithms of Merkle-Hellman algorithm works really fast!!
5.Goldwasser-Micali probabilistic encryption is quick enough but the decryption takes a lot of time.

Q & A