PROBABILISTIC PUBLIC-KEY ENCRYPTION

By Azymbek Beibyt
A public-key cryptosystem is **semantically secure** if it's computationally infeasible for opponent to derive any information about plaintext given only ciphertext and public key.

**Semantic Security = Cryptosystem Indistinguishability**
Semantic security are obtained using trapdoor (one-way) functions:

- prime factorization in RSA
- discrete logarithm problem in ElGamal
Fundamentals
Semantic Security (Cont.)

Why not use trapdoor functions as before?

- The fact that $f$ is a trapdoor function does not rule out the possibility of computing $x$ from $f(x)$ when $x$ is of a special form.

- The fact that $f$ is a trapdoor function does not rule out the possibility of easily computing some partial information about $x$ (even every other bit of $x$) from $f(x)$. 
Probabilistic Encryption

New approach introduced by Shafi Goldwasser and Silvio Micali in 1983.

- Replace deterministic block encryption by probabilistic encryption of single bits.
- Proved to be hard to extract any information about plaintext under polynomially bounded computational resources, because it is based on intractability of deciding Quadratic Residuosity modulo composite numbers whose factorization is unknown.
Probabilistic Public-Key Cryptosystem

**General Idea**

\((P, C, K, E, D, R)\), where \(R\) is a set of randomizers, encryption is public and decryption is secret and following properties should be satisfied:

1. \(e_K: P \times R \rightarrow C\) \quad \(d_K: C \rightarrow P\)

\[ d_K(e_K(b, r)) = b \], where \(b \in P, r \in R\)

and \(e_K(x, r) \neq e_K(x', r)\) if \(x \neq x'\)
Probabilistic Public-Key Cryptosystem

General Idea (Cont.)

2. Let $\epsilon$ be specified security parameter.

Define probability distribution $P_{K,x}(y)$ on $C$, which denotes the probability that $y$ is the ciphertext given that $K$ is the key and $x$ is the plaintext.

Then for $x \neq x'$, the probability distributions $P_{K,x}$ and $P_{K,x'}$ are not $\epsilon$-distinguishable in polynomial time.
Definitions

**Quadratic Residue (QR):**

\( a \) is element of \( \text{QR} \) iff

\[ a^{(p-1)/2} \equiv 1 \pmod{n} \]

thus

\[ \text{QR}(n) = \{ x^2 \pmod{n} : x \in \mathbb{Z}_n^* \} \]
Definitions

Pseudo-Square modulo n ($\overline{QR}$):

$$\overline{QR}(n) = \{ x \in \mathbb{Z}_n^* \setminus QR(n) : \left(\frac{x}{n}\right) = 1 \}$$

then

$$\overline{QR}(n) = \{ x \in \mathbb{Z}_n^* : \left(\frac{x}{p}\right) = \left(\frac{x}{q}\right) = 1 \}$$
Definitions

Composite Quadratic Residue Problem: Let $n=pq$ and $p, q$ are odd primes. Then $x$ is element of $QR$ iff

$$\left( \frac{x}{p} \right) = \left( \frac{x}{q} \right) = 1$$

where, $\left( \frac{x}{p} \right)$ is Legendre symbol

Hard to solve if the factorization of $n$ is unknown
Blum Blum Shub Generator

- **Blum Blum Shub (B.B.S.)** is a pseudorandom number generator proposed in 1986 by Lenore Blum, Manuel Blum and Michael Shub.
- BBS takes the form: \( x_{n+1} = x_n^2 \mod M \)
- where \( M = pq \) is the product of two large primes \( p \) and \( q \).
- The output is commonly either the bit parity of \( x_{n+1} \) or one or more of the least significant bits of \( x_{n+1} \).
Definitions

Blum Blum Shub Generator (Cont.)

- \( \text{GCD}(x_0, M) = 1 \) and \( x_0 \neq 1 \)
- \( (p, q) = 3 \pmod{4} \)
  (this guarantees that each quadratic residue has one square root which is also a quadratic residue)
- \( \text{gcd}(\varphi(p-1), \varphi(q-1)) \) should be small (this makes the cycle length large).
Goldwasser and Micali Cryptosystem

Overview

Let $n = pq$, where $p$, $q$ are distinct odd primes, and let $m \in QR(n)$. Integers $n$, $m$ are public and factorization $n = pq$ is secret. Let $P = \{0, 1\}$, $C = R = \mathbb{Z}_n$ and define $K = \{(n, p, q, m)\}$. Then the encryption is:

$$e_K(x, r) = m^x r^2 \pmod{n}$$

and

$$d_K(y) = \begin{cases} 
0 & \text{if } y \in QR(n) \\
1 & \text{if } y \in QR(n) 
\end{cases}$$
Goldwasser and Micali Cryptosystem

**Key Generation**

- Alice generates two distinct odd large prime numbers $p$ and $q$, randomly and independently of each other.
- Alice computes $N = p \cdot q$.
- She then finds some non-residue $m$ such that the Legendre symbols satisfy $\left( \frac{m}{p} \right) = \left( \frac{m}{q} \right) = -1$ and hence the Jacobi Symbol is $\left( \frac{m}{N} \right) = 1$.
- The *public key* consists of $(m, N)$. The secret key is the factorization $(p, q)$.
- **NOTE:** If $(p, q) = 3 \mod 4$ (*Blum Integer*), then the value $m = N - 1$ is guaranteed to have the required property.
Suppose Bob wishes to send a message $m$ to Alice:

- Bob first encodes $m$ as a string of bits ($m_1, \ldots, m_n$).
- For every bit $m_i$, Bob generates a random value $r$ from the group of units modulo $N$, or $\text{GCD}(r,n) = 1$.
- He outputs the value $c_i = m_i r^2 \pmod{n}$.
- Bob sends the ciphertext $(c_1, \ldots, c_n)$. 

Goldwasser and Micali Cryptosystem

Encryption
Alice receives \((c_1, \ldots, c_n)\). She can recover \(m\) using the following procedure:

For each \(i\), using the prime factorization \((p, q)\), Alice determines whether the value \(c_i\) is a quadratic residue; if so, \(m_i = 0\), otherwise \(m_i = 1\).

Alice outputs the message \(m = (m_1, \ldots, m_n)\).
Goldwasser and Micali Cryptosystem

Conclusion

Note: Goldwasser-Micali Cryptosystem uses a very high data expansion, because every bit of plaintext is encrypted to a ciphertext of length $\log n$ bits.

So for the system to be secure against factoring $n$, Alice should take $n$ to be a 1024-bit integer.

There is more efficient cryptosystem, in terms of data expansion, proposed by Blum and Goldwasser.
Blum and Goldwasser Cryptosystem

Key Generation

- To allow for decryption, the modulus (n) should be a **Blum Integer**, meaning that the prime factors (p,q) of n must be congruent to 3 mod 4.

- Alice generates two large prime numbers and such that \( p \neq q \), randomly and independently of each other, where \((p,q) = 3 \text{ mod } 4\).

- Alice computes \( N = pq \).

- The *public key* is \( N \). The *secret key* is the factorization \((p,q)\).
Blum and Goldwasser Cryptosystem

Encryption

- Bob first encodes \( m \) (message) as a string of \( L \) bits

\[
(m_0, \ldots, m_{L-1})
\]

- Bob selects a random element \( r \), where \( 1 < r < N \), and computes

\[
x_0 = r^2 \mod N
\]

- Bob uses the BBS pseudo-random number generator to generate \( L \) random bits (the keystream), as follows:
  - For \( i = 0 \) to \( L - 1 \):
  - Set \( b_i \) equal to the least-significant bit of \( x_i \).
  - Increment \( i \).
  - Compute \( x_i = (x_{i-1})^2 \mod N \)
Blum and Goldwasser Cryptosystem

Encryption (Cont.)

- Compute the ciphertext by XORing the plaintext bits with the keystream:
  \[
  \vec{c} = \vec{m} \oplus \vec{b}, \quad y = x_0^{2^L} \mod N
  \]

- Bob sends the ciphertext \((c_0, \ldots, c_{L-1}), y\)
Blum and Goldwasser Cryptosystem

Decryption

Alice receives \((c_0, \ldots, c_{L-1}), y\). She can recover \(m\) using the following procedure:

- Using the prime factorization \((p, q)\), she computes
  
  \[
  r_p = y^{(p+1)/4} \mod p \quad \text{and} \quad r_q = y^{(q+1)/4} \mod q
  \]

- Compute the initial seed:
  
  \[
  x_0 = q(q^{-1} \mod p)r_p + p(p^{-1} \mod q)r_q \mod N
  \]

- From \(x_0\), recompute the keystream using the BBS generator, as in the encryption algorithm.

- Compute the plaintext by XORing the keystream with the ciphertext.

- Alice recovers the plaintext \((m_0, \ldots, m_{L-1})\)
References

Quiz

1. What is semantic security in terms of cryptography?
2. Who are the authors of the idea of probabilistic encryption?
3. What is Blum Integer?
4. Why is Goldwasser-Micali cryptosystem not efficient?
5. What Pseudo-random Number Generator is used in Blum-Goldwasser cryptosystem?

BONUS: In what university is Shafi Goldwasser teaching?