RSA Cryptosystem

Michael Levin

Computer Science Department, Higher School of Economics

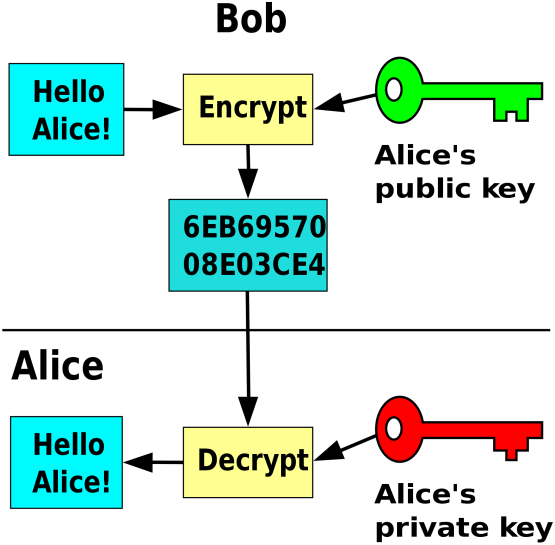
Outline

RSA

RSA

- Invented by Rivest, Shamir and Adleman in 1978
- Programs based on RSA are among the most frequently run
- Asymmetric or public key encryption

Asymmetric Encryption



Protocol

- Bob generates two random keys: public key E and private key D

Protocol

- Bob generates two random keys: public key E and private key D
- Bob publishes E for anyone to access

Protocol

- Bob generates two random keys: public key E and private key D
- Bob publishes E for anyone to access
- Anyone can encrypt message for Bob using E

Protocol

- Bob generates two random keys: public key E and private key D
- Bob publishes E for anyone to access
- Anyone can encrypt message for Bob using E
- Only Bob can decrypt an encrypted message using D

Protocol

- Bob generates two random keys: public key E and private key D
- Bob publishes E for anyone to access
- Anyone can encrypt message for Bob using E
- Only Bob can decrypt an encrypted message using D
- The encryption algorithm is public, so actually anyone can decrypt by trying all possible keys, but with known algorithms, it would take hundreds of years or more

Keys

- Bob generates two big random primes p and q
- Computes $n = p \cdot q$
- Generates random e coprime with $(p-1)(q-1)$
- Public key E is the pair (n, e)
- Private key D is the pair (p, q)

Encryption and Decryption

- Message m can be encoded as a sequence of bits and converted to an integer
- Needs to be between 0 and $n - 1$ — choose p and q so that this length in bits is sufficient
- Ciphertext $c = m^e \bmod n$ — use fast modular exponentiation
- Decryption: turns out that Bob can compute d such that $c^d \equiv m \bmod n$

Decryption

$$c^d \equiv (m^e)^d \equiv m^{ed} \bmod n$$

We need

$$m^{ed} \equiv m \bmod n$$

$n = pq$, so by Chinese Remainder Theorem it is equivalent to

$$m^{ed} \equiv m \bmod p, m^{ed} \equiv m \bmod q$$

Decryption

$$m^{ed} \equiv m \bmod p, m^{ed} \equiv m \bmod q$$

By Fermat's Little Theorem,
$m^k \equiv m^{k \bmod (p-1)} \bmod p$, so this holds if

$$ed \equiv 1 \bmod (p-1), ed \equiv 1 \bmod (q-1)$$

Decryption

$$ed \equiv 1 \bmod (p-1), ed \equiv 1 \bmod (q-1)$$

If $ed \equiv 1 \bmod (p-1)(q-1)$, then this holds.

Decryption

$$ed \equiv 1 \bmod (p-1), ed \equiv 1 \bmod (q-1)$$

If $ed \equiv 1 \bmod (p-1)(q-1)$, then this holds.

e is coprime with $(p-1)(q-1)$, so Bob can use Extended Euclid's Algorithm to compute such d that $ed \equiv 1 \bmod (p-1)(q-1)$.

Decryption Algorithm

- Compute $(p-1)(q-1)$
- Compute d such that $ed \equiv 1 \bmod (p-1)(q-1)$ using Extended Euclid's Algorithm
- Bob can compute and store this d right after generating p, q and e
- To decrypt ciphertext c, compute $c^d \bmod n$ using fast modular exponentiation

Communication Protocol

- Alice represents message m as a number between 0 and $n-1$
- Alice computes and sends ciphertext $c = m^e \bmod n$
- Bob receives c and computes $m = c^d \bmod n$

Discussion

- n is publicly known, but its factorization $n = pq$ is secret
- We rely on the difficulty of factorization
- If some day an efficient factorization algorithm appears, RSA will immediately become insecure

Discussion

- Actually we also rely on the fact that it is difficult to compute $(p-1)(q-1)$ without knowing p and q, otherwise Eve could compute d
- However, computing $(p-1)(q-1)$ is equivalent to factorizing $n = pq$
- Actually, $(p-1)(q-1) = \phi(pq) = \phi(n)$
- Encryption and decryption can be equivalently explained using Euler's ϕ function and Euler's theorem to note that for any m and k, $m^k \equiv m^{k \bmod \phi(n)} \mod n$

Modular Roots

- In general, to decrypt we need to solve the modular root problem: given c and e, find m such that $m^e \equiv c \bmod n$, or find e-th modular root of c
- Again, there are no known efficient algorithms for this problem
- However, there are known (inefficient) algorithms which solve modular root problem and avoid factorization

Breaking RSA

- Cryptoanalysts have been working on different attacks against RSA for decades
- There are accurate implementations which aren't broken
- However, there are a lot of subtle details
- Missing them leads to a breakable cipher
- You will learn several attacks and break several ciphers as an exercise!