Lecture S1: Cryptology

Cryptology

Cryptography: science of creating secret codes.
Cryptanalysis: science of code breaking.
Cryptology: science of secret communication.

Goal: information security in presence of malicious adversaries.
- Confidentiality.
  - Keep communication private.
- Integrity.
  - Detect unauthorized alteration to communication.
- Authentication.
  - Confirm identity of sender.
- Authorization.
  - Establish level of access for trusted parties.
- Non-repudiation.
  - Prove that communication was received.

Analog Cryptography

Task.
- Protect information.
- Identification.
- Contract.
- Money transfer.
- Public auction.
- Poker.
- Public election.
- Public lottery.
- Anonymous communication.

Analog implementation.

Digital Cryptography

Our goal.
- Implement all tasks digitally and securely.
- Implement additional tasks that can’t be done with physics!
  - play poker over phone
  - anonymous elections where everyone learns winner, but nothing else

Fundamental questions.
- Is any of this possible?
- How?

Today.
- Give flavor of modern (digital) cryptography.
- Implement a few of these tasks.
- Sketch a few technical details.
Digital Cryptography Axioms

Axiom 1. Players can toss coins.
   - Crypto impossible without randomness.

Axiom 2. Players are computationally limited.
   - Polynomial time.

Axiom 3. Factoring is hard computationally.
   - 1-way trapdoor function.

Fact. Primality testing is easy.

Theorem. Digital cryptography exists.
   - Can do all previous tasks DIGITALLY.

Encryption

Encryption.
- Most basic problem in cryptography.
- Alice sends Bob an encrypted message $E(m)$.
- Easy for Bob to recover original message $m$.
- Hard for Eve to learn anything about $m$.

Private Key Encryption

Alice sends Bob a message.
- Encode message as binary string (e.g., ASCII).
- Alice and Bob share secret key $k$.
- Everything else is public.

Private Key Encryption: One Time Pad

Key distribution.
- Alice and Bob share N-bit secret key $k$.

Alice wants to send N-bit message $m$ to Bob.
- Alice computes $E(m) = m \oplus k$ and sends $E(m)$.

Bob receives ciphertext $c = E(m)$.
- Bob compute $D(c) = c \oplus k$.

Why does it work?
- $D(E(m)) = D(m \oplus k) = (m \oplus k) \oplus k = m$

Why is it secure?
- If $k$ is uniformly random, so is $m \oplus k$. 

\[ m = 010110, k = 011100 \]
\[ c = 011010 \]
\[ m \oplus k = 001010 \]
\[ (m \oplus k) \oplus k = 010110 \]
Private Key Encryption

Advantages.
- Provably secure if key is random.
- Simple to implement.

Disadvantages.
- Hard to generate uniformly random keys.
- Need new key for each message.
- Signature?
- Non-repudiation?
- Key distribution?

Other private key encryption schemes.
- Data Encryption Standard (DES).
- Advanced Encryption Standard (AES, Rijndael algorithm).
- Blowfish.

Public Key Encryption

Alice sends Bob a message.
- Bob has PUBLIC and PRIVATE keys.
- public key locks, private key opens
- Everything else is public.

Key distribution.
- Bob has PUBLIC key = published in digital phonebook (VeriSign).
- Bob has PRIVATE key = known only by Bob.

Alice wants to transmit N-bit private message m to Bob.
- Alice encrypts message using Bob’s public key: E().

Bob receives ciphertext c from Alice.
- Bob decrypts message using his private key: D().

Under what situations does it work? D(E(m)) = m.

What are necessary conditions for security?
- Can encrypt message efficiently with public key.
- Can decrypt message efficiently with private key.
- CANNOT decrypt message efficiently with public key alone.

Modular Arithmetic

Do all computations modulo some base n.
- 10 + 4 (mod 12) = 2
- 38 * 15 (mod 280) = 570 (mod 280) = 10
RSA Public-Key Cryptosystem

- Most widely used public-key cryptosystem (500 million users).
- Sun, Microsoft, Apple, browsers, cell phones, ATM machines, . . .
- Based on difference between testing primality and factoring.

Key generation.
- Select two large prime numbers p and q at random.
- Compute n = pq.

Fact. If p and q are prime, there exist efficiently computable integers e and d such that:
  - For all m: \((m^e)^d \equiv m \pmod{n}\)

Bob’s keys.
- Public key: \((e, n)\).
- Secret key: \((d, n)\).

RSA Example

Parameters.
- \(p = 47, q = 79, n = 3713, e = 17, d = 3377, m = 2003\)

Modular exponentiation.
- \(2003^{17} \pmod{3713} = 134454746427671370568340195448570911966902998629125654163 \pmod{3713} = 232\)

Fact. Can mod out by \(n\) after each multiplication.
- Intermediate numbers stay small.

Analysis.
- Suppose \(x, y, n\) are N-bit integers.
- # multiplications proportional to \(2^y\).

Repeated Squaring

Parameters.
- \(p = 47, q = 79, n = 3713, e = 17, d = 3377, m = 2003\)
- \(m^e \mod n = 232\)

Efficient alternative. (repeated squaring)
- \(2003^1 \pmod{3713} = 2003 \mod{3713} = 2003 \equiv 1\)
- \(2003^2 \pmod{3713} = 4012009 \mod{3713} = 1969 \equiv 0\)
- \(2003^4 \pmod{3713} = 1969^2 \mod{3713} = 589 \equiv 0\)
- \(2003^8 \pmod{3713} = 589^2 \mod{3713} = 1612 \equiv 0\)
- \(2003^{16} \pmod{3713} = 1612^2 \mod{3713} = 3157 \equiv 1\)

\[
\begin{align*}
2003^{17} \pmod{3713} & \\
& \equiv 2003^{16} \ast 2003^1 \pmod{3713} \\
& \equiv 3157 \ast 2003 \pmod{3713} \\
& \equiv 6323471 \pmod{3713} \\
& \equiv 232 \pmod{3713}
\end{align*}
\]
Repeated Squaring

Repeated squaring.

\[ x^y = \begin{cases} 1 & \text{if } y = 0 \\ x^{y/2} \cdot x^{y/2} & \text{if } y \text{ is even} \\ x \cdot x^{(y-1)/2} \cdot x^{(y-1)/2} & \text{if } y \text{ is odd} \end{cases} \]

Analysis.
- Suppose \( x, y, n \) are \( N \)-bit integers.
- All intermediate integers \( \leq 2N \)-bits.
- \( y \) decreases by a factor of 2 after at most 2 multiplications and mods.
- \# multiplications (or mods) \( \leq 2N \).

```
power(x, y, n) {
    if (y == 0)
        return 1
    t = power(x, y/2)
    c = (t * t) % n
    if (y is even)
        return c
    else
        return (x * c) % n
}
```

RSA Details

How large should \( n = pq \) be?
- 1,024 bits for long term security.
- Too small \( \Rightarrow \) easy to break.
- Too large \( \Rightarrow \) time consuming to encrypt/decrypt.

How to choose large "random" prime numbers?
- Number theory \( \Rightarrow \) \( n / \log n \) prime numbers between 2 and \( n \).
  - Primes are plentiful: \( 10^{151} \) with \( \leq 512 \) bits.
  - Will never run out, and no two people will pick same ones.
  - Database of primes: 1GB per gram \( \Rightarrow \) Chandrasekhar limit.
- Use Miller-Rabin algorithm to check whether integer is prime.
  - Randomized polynomial-time!
  - Guess, and use algorithm to check.

RSA Attacks

Factoring.
- Factor \( n = pq \).
- \( d \) is public.
- Use \( p, q, \) and \( d \) to compute \( e \).

Other means?
- Long-standing open research question.

Note: Diffie-Helman cryptosystem can be broken if and only if factoring is hard.
- Discrete log: given \( x, m, n \), find \( d \) such that \( x^d \text{ mod } n = m \).

RSA Tradeoffs

Advantages.
- One public and one private key per INDIVIDUAL (not per message).
- Digital signatures.
- Relatively easy to implement.

Disadvantages.
- Security relies on decryption being "computationally inefficient."
- Relatively expensive to decrypt
  (often used in hybrid system, e.g., with DES).
RSA in the Real World

Secure Internet communication.
- Browsers.
- S/MIME, SSL, S/WAN.
- PGP.
- Microsoft Outlook.

Operating systems.
- Sun, Microsoft, Apple, Novell.

Hardware.
- Cell phones.
- ATM machines.
- Wireless Ethernet cards.
- Smart cards (Mondex).
- Palm Pilots.