Contents

1. Introduction to cryptography
2. Cryptosystem and cryptanalysis
3. The RSA cryptosystem
4. Digital signatures
5. Secret sharing schemes
Cryptography is the field concerned with techniques for securing information, particularly in communications;

Cryptography focuses on the following paradigms:

- **Authentication** – the process of proving one’s identity (the primary forms of host-to-host authentication on the Internet today are name-based or address-based, both of which are notoriously weak);

- **Privacy/confidentiality** – ensuring that no one can read the message except the intended receiver;

- **Integrity** – assuring the receiver that the received message has not been altered in any way from the original;

- **Non-repudiation** – a mechanism to prove that the sender really sent this message.
Applications of cryptography:

- computer and information security: cryptography is necessary when communicating over any untrusted medium, which includes just about any network, particularly the Internet;

- e-commerce, e-payment, e-voting, e-auction, e-lottery, and e-gambling schemes, are all based on cryptographic (security) protocols.
Introduction to cryptography

A few examples of concrete applications are in order:

- **IPsec = IP Security Protocol**
  - Standard for cryptographically-based authentication, integrity, and confidentiality services at the IP datagram layer
  - Uses RSA, DH, MD5, DES, 3DES, and SHA1

- **SSL & TLS**
  - SSL = Secure Sockets Layer
  - Allows a “secure pipe” between any two applications for secure transfer of data and mutual authentication
  - TLS = Transport Layer Security
  - TLS is the latest enhancement of SSL
  - Uses RSA, DH, RC4, MD5, DES, 3DES, and SHA1
Introduction to cryptography

**DNSSEC**
- **DNSSEC = Domain Name System Security Extensions**
- Protocol for secure distributed name services such as hostname and IP address lookup
- Uses RSA, MD5, and DSA

**IEEE 802.11**
- Protocol standard for secure wireless Local Area Network products
- Uses RC4 and MD5
DOCSIS,

- **DOCSIS** = Data Over Cable Service Interface Specification
- Cable modem standard for secure transmission of data with protection from theft-of-service and denial-of-service attacks and for protecting the privacy of cable customers
- Uses RSA, DES, HMAC, and SHA1

CDPD

- **CDPD** = Cellular Digital Packet Data
- It is a standard designed to enable customers to send computer data over existing cellular networks
- Uses DH and RC4

PPTP, SET, S/MIME etc.
A brief history of cryptography:

- The oldest forms of cryptography date back to at least Ancient Egypt, when derivations of the standard hieroglyphs of the day were used to communicate;

- Julius Caesar (100-44 BC) used a simple substitution cipher with the normal alphabet (just shifting the letters a fixed amount) in government communications (Caesar cipher);

- Thomas Jefferson, the father of American cryptography, invented a wheel cipher in the 1790’s, which would be redeveloped as the Strip Cipher, M-138-A, used by the US Navy during World War II;
During World War II, two notable machines were employed: the German’s **Enigma machine**, developed by Arthur Scherbius, and the Japanese **Purple Machine**, developed using techniques first discovered by Herbert O. Yardley;

William Frederick Friedman, the father of American cryptanalysis, led a team which broke in 1940 the Japanese Purple Code;

In the 1970s, Horst Feistel developed a “family” of ciphers, the **Feistel ciphers**, while working at IBM’s Watson Research Laboratory. In 1976, The National Security Agency (NSA) worked with the Feistel ciphers to establish FIPS PUB-46, known today as **DES**;
In 1976, Martin Hellman, Whitfield Diffie, and Ralph Merkle, have introduced the concept of public-key cryptography;

In 1977, Ronald L. Rivest, Adi Shamir and Leonard M. Adleman proposed the first public-key cipher which is still secure and used (it is known as RSA);

The Electronic Frontier Foundation (EFF) built the first unclassified hardware for cracking messages encoded with DES. On July 17, 1998, the EFF DES Cracker was used to recover a DES key in 22 hours. The consensus of the cryptographic community was that DES was not secure;

In October 2001, after a long searching process, NIST selected the Rijndael cipher, invented by Joan Daemen and Vincent Rijmen, as the Advanced Encryption Standard. The standard was published in November 2002.
Definition 1  A cryptosystem or cipher is a 5-tuple $S = (\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$, where:

1. $\mathcal{P}$ is a non-empty finite set of plaintext symbols;
2. $\mathcal{C}$ is a non-empty finite set of cryptotext symbols;
3. $\mathcal{K}$ is a non-empty finite set of keys;
4. $\mathcal{E}$ and $\mathcal{D}$ are two sets of functions (algorithms)

$$
\mathcal{E} = \{ e_K : \mathcal{P} \rightarrow \mathcal{C} | K \in \mathcal{K} \} \quad \text{and} \quad \mathcal{D} = \{ d_K : \mathcal{C} \rightarrow \mathcal{P} | K \in \mathcal{K} \},
$$

such that $d_K(e_K(x)) = x$, for any $K \in \mathcal{K}$ and $x \in \mathcal{P}$.

$e_K$ is the encryption rule (algorithm), and $d_K$ is the decryption rule (algorithm), induced by $K$. 


Let $S = (\mathcal{P}, \mathcal{C}, K, E, D)$ be a cipher. A plaintext (cryptotext) is a finite sequence of plaintext (cryptotext) symbols. If $x = x_1 \cdots x_n$ is a plaintext, then it can be encrypted by $S$ in one of the following two ways:

- **(Fixed-key encryption).** Generate a key $K$ and encrypt each plaintext symbol by $e_K$:

  $$y = e_K(x_1) \cdots e_K(x_n);$$

- **(Variable-key encryption).** Generate a sequence of keys $K_1, \ldots, K_n$ and encrypt each plaintext symbol $x_i$ by $e_{K_1}$:

  $$y = e_{K_1}(x_1) \cdots e_{K_n}(x_n).$$

**Remark 1** We will mainly use the fixed-key encryption mode.
Cryptosystem and cryptanalysis

Figure 1: Communication between two entities $A$ and $B$ via a cryptosystem
Cryptosystems can be classified into:

- **symmetric (private-key, single-key) cryptosystems** – characterized by the fact that it is easy to compute the decryption rule $d_K$ from $e_K$, and vice-versa.
asymmetric (public-key) cryptosystems – characterized by the fact that it is hard to compute \( d_K \) from \( e_K \). With such cryptosystems, the key \( K \) is split into two subkeys, \( K_e \), for encryption, and \( K_d \), for decryption. Moreover, \( K_e \) can be made public without endangering security.
Most cryptosystems are based on number theory and, therefore, it is customary to view each plaintext symbol as an integer, for instance, based on a one-to-one correspondence like the one below:

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
<th>g</th>
<th>h</th>
<th>i</th>
<th>j</th>
<th>k</th>
<th>l</th>
<th>m</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>n</th>
<th>o</th>
<th>p</th>
<th>q</th>
<th>r</th>
<th>s</th>
<th>t</th>
<th>u</th>
<th>v</th>
<th>w</th>
<th>x</th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
</tr>
</tbody>
</table>

For instance, the plaintext “home” becomes the sequence of integers “7,14,12,4”.
Example 1 (Affine Cryptosystems)

- \( \mathcal{P} = \mathcal{C} = \mathbb{Z}_{26} \);
- \( \mathcal{K} = \{ (a, b) \in \mathbb{Z}_{26} \times \mathbb{Z}_{26} \mid \gcd(a, 26) = 1 \} \);
- for any key \( K = (a, b) \) and \( x, y \in \mathbb{Z}_{26} \),

\[
e_K(x) = (ax + b) \mod 26 \quad \text{and} \quad d_K(y) = (a^{-1}(y - b)) \mod 26.
\]

Let \( K = (7, 3) \) and the plaintext \( pt = hot \) (\( pt = 7, 14, 19 \)). Then,

\[
e_K(pt) = e_K(7), e_K(14), e_K(19) = 0, 23, 6,
\]

that is, the cryptotext is \( ct = axg \).
Affine cryptosystems can be easily broken by exhaustive key search (EKS), also known as brute-force search, which consists of trying every possible key until you find the right one.

Question: If you have a chunk of cryptotext and decrypt it with one key after the other, how does you know when you found the correct plaintext?

Answer: You know that you have found the plaintext because it looks like plaintext. Plaintext tends to look like plaintext. It’s an English-language message, or a data file from a computer application (e.g., programs like Microsoft Word have large known headers), or a database in a reasonable format. When you look at a decrypted file, it looks like something understandable. When you look at a cryptotext file, or a file decrypted with the wrong key, it looks like gibberish.
Question: How many keys are?

Answer: If an affine cryptosystem is developed over $\mathbb{Z}_{26}$, then there are only $\phi(26) \times 26 = 12 \times 26 = 312$ possible keys.

As a conclusion, given an affine cryptosystem, it is very easy to enumerate all its keys and break it using a laptop (assuming that you have a chunk of cryptotext).
The RSA cryptosystem

In 1977, Ronald Rivest, Adi Shamir, and Leonard Adleman, proposed the first public-key cryptosystem which is still secure and used.

**The RSA cryptosystem**

- Let $p$ and $q$ be two distinct primes, and $n = pq$;
- $\mathcal{P} = \mathcal{C} = \mathbb{Z}_n$;
- $\mathcal{K} = \{(n, p, q, e, d) | e \in \mathbb{Z}_n^* \land ed \equiv 1 \mod \phi(n)\}$;
- For any $K = (n, p, q, e, d) \in \mathcal{K}$ and $x, y \in \mathbb{Z}_n$,

  $$e_K(x) = x^e \mod n \quad \text{and} \quad d_K(y) = y^d \mod n;$$

- $(n, e)$ is the public key, and $(p, q, d)$ is the secret key.
The RSA cryptosystem

Example 2 (with artificially small parameters)
Let \( p = 61 \) and \( q = 53 \). Then:

- \( n = pq = 3233 \) and \( \phi(n) = 3120 \);
- if we chose \( e = 17 \), then \( d \) can be computed with the extended Euclidean algorithm. We obtain \( d = e^{-1} \mod 3120 = 2753 \);
- \( n = 3233 \) and \( e = 17 \) are public parameters; \( p, q, \) and \( d \) secret;

Let \( x = 123 \) be a plaintext. The cryptotext is

\[
y = 123^{17} \mod 3233 = 855.
\]

In order to decrypt \( y \) we have to compute

\[
855^{2753} \mod 3233 = 123.
\]
The RSA cryptosystem

Security issues:
- If $p$ or $q$ is recovered (e.g., by factoring $n$ in reasonable time), then the system is completely broken;
- If $\phi(n)$ can be computed in reasonable time, then the system is completely broken;
- If $d$ can be easily computed from $n$ and $e$, then the system is completely broken.

In practice:
- $p$ and $q$ are 512-bit primes (or even larger);
- $e$ is small (fast encryption) but chosen such that $d > \frac{4}{\sqrt{n}}$ (otherwise, an efficient attack can be mounted).

Digital signatures

Public key cryptography solves another problem crucial to e-commerce and Internet cyber relationship: it lets you emulate written signatures. This use of public key technology is called a digital signature.

A digital signature must provide:

- **authenticity and integrity.** That is, it must be “impossible” for anyone who does not have access to the secret key to forge \((x, \sigma)\) (\(x\) is the original data and \(\sigma\) is its associated signature);
- **non-repudiation.** That is, it must be impossible for the legitimate signer to repudiate his own signature.

Signing (encrypting with a private key) is extremely slow, so you usually add a time-saving (and space-saving) step before you encrypt messages. It is called message digesting or hashing.
A hash algorithm (function) is an algorithm (function) which, applied to an arbitrary-length input data, produces a fixed-length output data (called a hash value or message digest or fingerprint). A hash algorithm should be resistant to collisions.
Digital signatures

Any public key cipher can be used to produce digital signatures:

- If $K = (K_e, K_d)$ is $A$’s key, then the encryption of a message $x$ by $K_d$ (which is $A$’s private key) is the **digital signature associated** to $x$. It can be **verified** by the public key $K_e$:

$$x \equiv d_{K_e}(e_{K_e}(x)).$$

The **RSA signature** is obtained from the RSA public key cipher.
Secret sharing schemes

An important application of the Chinese remainder theorem concerns the construction of \((k, n)\)-threshold sharing schemes.

A \((k, n)\)-threshold sharing scheme consists of \(n\) people \(P_1, \ldots, P_n\) sharing a secret \(S\) in such a way that the following properties hold:

- \(k \leq n\);
- each \(P_i\) has an information \(I_i\);
- knowledge of any \(k\) of \(I_1, \ldots, I_k\) enables one to find \(S\) easily;
- knowledge of less than \(k\) of \(I_1, \ldots, I_k\) does not enable one to find \(S\) easily.
Secret sharing schemes

We will show how a \((k, n)\)-threshold sharing scheme can be constructed:

- let

\[
\begin{align*}
m_1 < & \cdots < m_k < \cdots < m_{n-k+2} < \cdots < m_n
\end{align*}
\]

be a sequence of pairwise co-prime numbers such that

\[\alpha = m_1 \cdots m_k > m_{n-k+2} \cdots m_n = \beta;\]

- let \(S\) be a secret, \(\beta < S < \alpha;\)

- each \(P_i\) gets the information \(I_i = S \mod m_i.\)

This is called Mignotte’s secret sharing scheme.
Secret sharing schemes

Any group of $k$ people, $P_{i_1}, \ldots, P_{i_k}$, can recover uniquely the secret $S$ by solving the system:

\[
\begin{align*}
  x &\equiv I_{i_1} \mod m_{i_1} \\
  &\quad \ldots \\
  x &\equiv I_{i_k} \mod m_{i_k}
\end{align*}
\]

According to the Chinese remainder theorem, this system has a unique solution modulo $m_{i_1} \cdots m_{i_k}$, and this solution is $S$ because

\[ S < \alpha < m_{i_1} \cdots m_{i_k}. \]
No group of \( k - 1 \) people, \( P_{j_1}, \ldots, P_{j_k} \), can recover uniquely the secret \( S \) by solving the system:

\[
\begin{align*}
\left\{ \begin{array}{l}
x \equiv I_{j_1} \mod m_{j_1} \\
\quad \cdots \\
\quad x \equiv I_{j_{k-1}} \mod m_{j_{k-1}}
\end{array} \right.
\]

According to the Chinese remainder theorem, this system has a unique solution modulo \( m_{j_1} \cdots m_{j_{k-1}} \), and this solution, denoted \( x_0 \), satisfies

\[ x_0 < m_{j_1} \cdots m_{j_{k-1}} \leq \beta. \]