Cryptography Goals

- Protect private communication in the public world
- Alice and Bob are shouting messages over a crowded room
- No guest can understand what they are saying

Other Uses of Cryptography

- Authentication
  - Bob should be able to verify that Alice has created the message
- Integrity
  - Bob should be able to verify that message has not been modified
- Non-repudiation
  - Alice cannot deny that she indeed sent the message

Other Uses of Cryptography

- How to exchange a secret with someone you have never met, shouting in a room full of people
- How can someone convince us they know the secret without giving it away
- How to send encrypted messages to any subset of n people
- How to encrypt a message so that it can be decrypted only if m out of n people want to decrypt it

Why Do We Need a Key at All?

- Alice could give a message covertly
  - “Let’s meet where we meet last year”
  - Works only if Alice and Bob know each other well
- Alice could change the message in a secret way
  - Secret algorithms can be broken
  - In general good cryptography assumes knowledge of algorithm by anyone, secret lies in a key!!!
- Alice could hide her message in some other text - steganography

Basic Problem and Terminology

- Alice
- M – message
- K1 – encryption key
- EK1(M) – message M is encrypted using key K1
- C – ciphertext
- K2 – decryption key
- DK2(C) – message M is decrypted using key K2
- M – message

What Can Go Wrong?

- Ciphertext-only attack: Eve can attempt either to learn M or to learn how to decrypt other messages by observing many ciphertexts C

Cycles

- If K1 = K2 this is symmetric encryption
- If K1 ≠ K2 this is asymmetric (public key) encryption
What Can Go Wrong?

**Known-plaintext attack:** Eve can attempt to learn how to decrypt messages by observing many ciphertexts $C$ for known messages $M$.

**Chosen-plaintext attack:** Malory can feed chosen messages $M$ into encryption algorithm and look at resulting ciphertexts $C$. Thus she can attempt to learn either encryption key $K_1$, decryption key $K_2$ or messages $M$ that produce $C$. Assumption is that extremely few messages $M$ can produce same $C$.

**Adaptive-chosen-plaintext attack:** Malory can feed chosen messages $M$ into encryption algorithm and look at resulting ciphertexts $C$. Gain some knowledge or establish a hypothesis, then feed new $M$ to gain more knowledge or test the hypothesis. Thus she can attempt to learn either encryption key $K_1$ or how to decrypt messages.

**Man-in-the-middle attack:**
- Malory can substitute messages
- Malory can modify messages
- so that they have different meaning
- so that they are scrambled
- Malory can drop messages
- Malory can replay messages to Alice, Bob or the third party

**Brute-force attack:** Eve has caught a ciphertext and will try every possible key to try to decrypt it. This can be made infinitely hard by choosing a large keyspace.

**Cryptographic techniques**
- **Substitution** (confusion)
  - Goal: obscure relationship between plaintext and ciphertext
  - Substitute parts of plaintext with parts of ciphertext
- **Transposition** (diffusion)
  - Goal: dissipate redundancy of the plaintext by spreading it over ciphertext
Substitution

- **Monoalphabetic** – each character is replaced with another character
  - Caesar’s cipher – each letter is shifted by 3, a becomes d, b becomes e, etc.
  - Keep a mapping of symbols into other symbols
  - Drawback: frequency of symbols stays the same and can be used to break the cipher

Substitution

- **Homophonic** – each character is replaced with a character chosen randomly from a subset
  - Ciphertext alphabet must be larger than plaintext alphabet – we could replace letters by two-digit numbers
  - Number of symbols in the subset depend on frequency of the given letter in the plaintext
  - The resulting ciphertext has all alphabet symbols appearing with the same frequency

Substitution

- **Polygram** – each sequence of characters of length $n$ is replaced with another sequence of characters of length $n$
  - Like monoalphabetic cipher but works on $n$-grams
  - Can be viewed as simple block ciphers

Substitution

- **Polyalphabetic** – many alphabetic ciphers are used sequentially
  - First map is used for the first letter, second map for the second letter and so on
  - This can be broken if the number of used ciphers is not very large
  - Vigenère cipher
  - Rotors (Enigma)
  - Running-code cipher

One-time pad

- Polyalphabetic cipher with infinite key
  - Combine letters from the message with the letters from an infinite key, randomly generated
  - Never reuse the key
  - Key needs to be generated using a very good RNG (to avoid any patterns)
  - This cipher cannot be broken
  - Sender and receiver must be perfectly synchronized

XOR

- XOR bits of the message with bits of the key
- Polyalphabetic cipher in binary domain
- Can be broken in the following way:
  - Learn the length of the key by counting coincidences
  - Shift the ciphertext by this length and XOR it with itself
Transposition
- Shuffle characters in the message
- Since the frequency of characters in ciphertext is same as in the plaintext, frequency analysis can be used to break cipher
- Using several transposition ciphers in a chaining manner increases security somewhat

Symmetric Key Encryption
- Alice and Bob have never met before and they want to communicate
- Alice and Bob agree on a secret key
- Alice encrypts the message with the secret key, using substitution and transposition methods, and a few other tricks
- Bob reverses the process to decrypt the message

What Can Go Wrong?
- Eve could listen to shared key exchange and learn the key
- Malory could learn the key and replace it with her own (man-in-the-middle attack)
- Malory could prevent Alice and Bob from completing the protocol
- So how do Alice and Bob exchange shared key?

Exchanging a Shared Key
- Alice can send a key by courier to Bob
- Alice and Bob may have a mutual friend Trent:
  - They both trust Trent
  - They have already set up secret keys with Trent
  - Alice sends message to Trent with secret key for communication with Bob, encrypts the message with the key she shares with Trent
  - Trent decrypts the message, encrypts it with the key he shares with Bob
  - For n participants, there are \( \frac{n^2(n-1)}{2} \) keys
- Use Diffie-Hellman key exchange or public-key cryptography

Public Key Encryption
- Everyone has two keys:
  - Public key K1 that everyone knows
  - Private key K2 that only he knows
  - Encryption algorithm and key properties ensure that
    \[ D_{K2}(E_{K1}(M)) = M \]
- Alice creates a secret key, encrypts it with Bob’s public key and sends it off
- Bob decrypts the message with his private key
- They could even communicate this way but it’s slow

One-Way Functions
- Functions such that computing \( f(x) \), given \( x \) is easy, but computing \( x \) given \( f(x) \) is hard
- Hard means that it would take all computers on Earth millions of years to do it
- But for decryption we need to be able to calculate \( x \) given \( f(x) \):
  - Trapped door one-way function: There is a secret \( y \) such that given \( f(x) \) and \( y \) it is easy to compute \( x \)
Modular Arithmetic

- Observe all operations in Galois Field GF(n)
- Only numbers allowed are 0 ... n-1, smaller and larger numbers just wrap around
- $a \equiv b \mod n$ if $\forall k, a=kn+b$ e.g. 26 mod 16 = 10
- $b$ is called residue of $a$ modulo $n$
- $a$ is called congruent to $b$ modulo $n$
- Numbers 0 ... n-1 form complete set of residues for $n$
- Modulo operation (modular reduction) can be performed at any point, e.g.
  $$(a + b) \mod n = ((a \mod n) + (b \mod n)) \mod n$$

Example

$$4^{19} \mod 23$$
$$19=10011$$

$$4^{19} \mod 23 = (((4^2 \mod 23)^2 \mod 23)^2 \ast 4 \mod 23)^2 \ast 4 \mod 23$$
$$= (((16^2 \mod 23)^2 \ast 4 \mod 23)^2 \ast 4 \mod 23$$
$$= (3)^2 \ast 4 \mod 23$$
$$= (18)^2 \ast 4 \mod 23$$
$$= 9$$

Inverses Modulo a Number

- Multiplicative inverse $y$ for $x$ is a number that satisfies:
  $$x \ast y = 1$$
- In GF(n) inverse $y$ for $x$ modulo $n$ is a number that satisfies:
  $$x \ast y \mod n = 1$$
- Inverse $y$ can be found uniquely if $x$ and $n$ are relatively prime, otherwise it cannot be found
- If $n$ is prime than it is relatively prime to all numbers {0, n-1} and each number has its inverse in GF(n)

Prime Numbers

- A number $n$ is prime if it is only divisible by 1 and itself
- Numbers $x$ and $y$ are relatively prime if they share no factors greater than 1
  - E.g. 7 and 15 are relatively prime, 9 and 15 are not because they have 3 as common factor

Extended Euclidean Algorithm

- How to find an inverse $y$ for $x \mod n$
  $$x \ast y \mod n = 1$$
  $$x \ast y = 1 + k \ast n$$
  $$x \ast y - k \ast n = 1$$
- Euclidean algorithm will find $y$ and $k$ given $x$ and $n$
- It actually finds gcd(x,n) and coefficients $y$ and $k$
- If $x$ and $n$ are relatively prime than Euclidean algorithm will find inverse of $x \mod n$
Extended Euclidean Algorithm

1. \( x = 13, n = 225, y = ? \)

\[
\begin{array}{c|c|c|c}
 a & q & x & y \\
\hline
 225 & - & 1 & 0 \\
 13 & 0 & 1 & 1 \\
\end{array}
\]

2. \( 225 / 13 = 17 \text{ remainder } 4 \)

\[
\begin{array}{c|c|c|c}
 a & q & x & y \\
\hline
 225 & - & 1 & 0 \\
 13 & 17 & 0 & 1 \\
 4 & 3 & 0 & 1 \\
 1 & 4 & 1 & 0 \\
\end{array}
\]

3. \( 13 / 4 = 3 \text{ remainder } 1 \)

\[
\begin{array}{c|c|c|c}
 a & q & x & y \\
\hline
 225 & - & 1 & 0 \\
 13 & 17 & 0 & 1 \\
 4 & 3 & 0 & 1 \\
 1 & 4 & 1 & 0 \\
\end{array}
\]

4. \( 4 / 1 = 4 \text{ remainder } 0 \)

\[
\begin{array}{c|c|c|c}
 a & q & x & y \\
\hline
 225 & - & 1 & 0 \\
 13 & 17 & 0 & 1 \\
 4 & 3 & 0 & 1 \\
 1 & 4 & 1 & 0 \\
\end{array}
\]

Extended Euclidean Algorithm

5. \( x_{i+1} = x_{i-1} - q_i \cdot x_i \)

\[
\begin{array}{c|c|c|c}
 a & q & x & y \\
\hline
 225 & - & 1 & 0 \\
 13 & 17 & 0 & 1 \\
 4 & 3 & 0 & 1 \\
 1 & 4 & 1 & 0 \\
\end{array}
\]

6. \( 1 - 4 \cdot 0 = 1 \)

\[
\begin{array}{c|c|c|c}
 a & q & x & y \\
\hline
 225 & - & 1 & 0 \\
 13 & 17 & 0 & 1 \\
 4 & 3 & 0 & 1 \\
 1 & 4 & 1 & 0 \\
\end{array}
\]
Extended Euclidean Algorithm

\[ 0 - 3*1 = -3 \]

<table>
<thead>
<tr>
<th>a</th>
<th>q</th>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>225</td>
<td>-</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>17</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>1</td>
<td>-3</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>-3</td>
<td></td>
</tr>
</tbody>
</table>

\[ y_{i+1} = y_{i-1} - q_i y_i \]

Extended Euclidean Algorithm

\[ 0 - 17*1 = -17 \]

<table>
<thead>
<tr>
<th>a</th>
<th>q</th>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>225</td>
<td>-</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>17</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>1</td>
<td>-17</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>-3</td>
<td></td>
</tr>
</tbody>
</table>

\[ 1 - 3*(-17) = 52 \]

<table>
<thead>
<tr>
<th>a</th>
<th>q</th>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>225</td>
<td>-</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>17</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>1</td>
<td>-17</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>-3</td>
<td>52</td>
</tr>
</tbody>
</table>

52 is inverse for 13 mod 225

Factorization of Large Numbers

- It is hard to factor large numbers (and we have seen that exponentiation on GF can be performed efficiently)
- Various factoring algorithms exist: Number field sieve, Quadratic sieve …
- Generally factoring time of a large number \( n \) increases exponentially with each binary digit added to \( n \)

Public Key Cryptography (RSA)

- Created by Ron Rivest, Adi Shamir, and Leonard Adleman
- Choose two prime numbers \( p \) and \( q \) of equal length
- Compute \( n = p * q \), and Euler Totient function \( \phi(n) = (p-1)*(q-1) \)
- Choose public key \( e \) relatively prime to \( \phi(n) \)
Public Key Cryptography (RSA)

- Using extended Euclidean algorithm calculate \( d \) which is inverse of \( e \mod \phi(n) \)
  \[ d \times e = 1 \mod \phi(n) \]
- Publish \( e \) and \( n \), remember \( d \)
- Encryption:
  \[ E(M) = M^e \mod n \]
- Decryption:
  \[ D(C) = C^d \mod n = M^{de} \mod n = M \]

Fermat’s Little Theorem

- If \( m \) is prime and \( a \) is not multiple of \( m \) then
  \[ a^{m-1} \mod m = 1 \]

Chinese Remainder Theorem

- If
  \[ x = y \mod p \]
  And
  \[ x = y \mod q \]
- And \( p \) is relatively prime to \( q \), then
  \[ x = y \mod pq \]

Public Key Cryptography (RSA)

\[ D(C) = C^d \mod n = M^{de} \mod n = M \]
\[ M^{e(p-1)(q-1)} \mod n = M \]
\[ M^{e(p-1)(q-1)} \mod n \]

Consider
\[ X = M^{e(p-1)(q-1)} \mod p = (M^{(p-1)} \mod p)^{(q-1)} = 1 \]

Similarly
\[ X = M^{e(p-1)(q-1)} \mod q = 1 \]

Using Chinese remainder theorem
\[ M^{e(p-1)(q-1)} \mod pq = 1 \]

Public Key Cryptography (RSA)

- To summarize:
  - We can easily perform exponentiation in GF
  - We can calculate \( d \) out of \( e \) and \( n \) in polynomial time using extended Euclidean algorithm (because we know \( p \) and \( q \))
  - Enemy must factor large number \( n \) to learn \( p \) and \( q \) which is exponentially expensive
Common Practice

- Public-key cryptography is about 1500 times slower than symmetric cryptography
- Use public-key cryptography to exchange shared key
- Continue to communicate using symmetric cryptography

Diffie-Hellman Key Exchange

- Alice and Bob agree on $g$ and large $n$
- Alice chooses random number $a$ and sends to Bob $g^a \mod n$
- Bob chooses random number $b$ and sends to Alice $g^b \mod n$
- Alice takes Bob’s message and calculates $g^{ab} \mod n$
- Bob does the same; now they both know a secret