

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

1

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

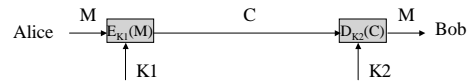
2

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

3

Basic Problem and Terminology



M – message

K1 – encryption key

$E_{K1}(M)$ – message M is encrypted using key K1

C – ciphertext

K2 – decryption key

$D_{K2}(M)$ – message M is decrypted using key K2

If $K1=K2$ this is symmetric encryption

If $K1 \neq K2$ this is asymmetric (public key) encryption

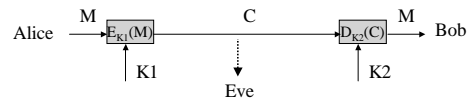
4

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

5

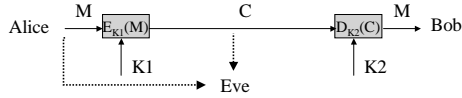
What Can Go Wrong?



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

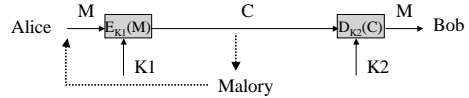
6

What Can Go Wrong?



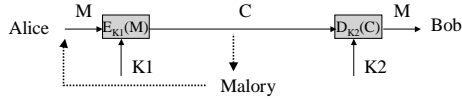
Known-plaintext attack: Eve can attempt to learn how to decrypt messages by observing many ciphertexts C for known messages M

What Can Go Wrong?



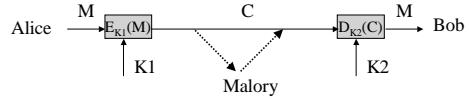
Chosen-plaintext attack: Malory can feed chosen messages M into encryption algorithm and look at resulting ciphertexts C. Thus she can attempt to learn either encryption key K1, decryption key K2 or messages M that produce C. Assumption is that extremely few messages M can produce same C.

What Can Go Wrong?



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 K1 or how to decrypt messages.

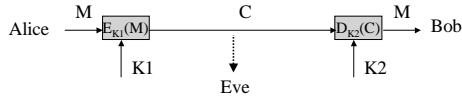
What Can Go Wrong?



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

What Can Go Wrong?



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

- **Monoalphabetical** – 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

13

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

14

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

15

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

16

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

17

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

18

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 $\frac{n*(n-1)}{2}$ keys
- Use Diffie-Hellman key exchange or public-key cryptography

Public Key Encryption

- Everyone has two keys:
 - Public key K_1 that everyone knows
 - Private key K_2 that only he knows
 - Encryption algorithm and key properties ensure that

$$D_{K_2}(E_{K_1}(M)) = M$$
- 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 $f(x)$:
 - **Trapdoor 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 $0 \dots n-1$, smaller and larger numbers just wrap around
 - $a \equiv b \pmod n$ if $\forall k, a=kn+b$ e.g. $26 \pmod{16} = 10$
 - b is called **residue of a modulo n**
 - a is called **congruent to b modulo n**
 - Numbers $0 \dots n-1$ form **complete set of residues** for n
 - Modulo operation (modular reduction) can be performed at any point, e.g.

$$(a + b) \pmod n = ((a \pmod n) + (b \pmod n)) \pmod n$$

25

Modular Arithmetic

- Exponentiation can be performed very efficiently (**addition chaining**):
 - We want to calculate $a^x \pmod n$
 - Write x as a binary number, $result=1$
 - Traverse x from left to right
 - If digit is 1, $result=result^2*a$
 - If digit is 0, $result=result^2$
 - Perform modular reduction often to keep result small

26

Example

$$\begin{aligned}
 &4^{19} \pmod{23} \\
 &19 = 10011 \\
 &4^{19} \pmod{23} = (((4^2 \pmod{23})^2 \pmod{23})^2 * 4 \pmod{23})^2 * 4 \pmod{23} \\
 &= (((16)^2 \pmod{23})^2 * 4 \pmod{23})^2 * 4 \pmod{23} \\
 &= ((3)^2 * 4 \pmod{23})^2 * 4 \pmod{23} \\
 &= (13)^2 * 4 \pmod{23} \\
 &= (18)^2 * 4 \pmod{23} \\
 &= 9
 \end{aligned}$$

27

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

28

Inverses Modulo a Number

- Multiplicative inverse y for x is a number that satisfies:

$$x*y = 1$$
- In GF(n) inverse y for x modulo n is a number that satisfies:

$$x*y \pmod n = 1$$
- Inverse y can be found uniquely if x and n are relatively prime, otherwise it cannot be found
- If n is prime then it is relatively prime to all numbers $\{0, n-1\}$ and each number has its inverse in GF(n)

29

Extended Euclidean Algorithm

- How to find an inverse y for $x \pmod n$

$$\begin{aligned}
 &x*y \pmod n = 1 \\
 &x*y = 1 + k*n \\
 &x*y - k*n = 1
 \end{aligned}$$
- 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 then Euclidean algorithm will find inverse of $x \pmod n$

30

Extended Euclidean Algorithm

> $x=13, n=225, y=?$

a	q	x	y
225	-	1	0
13		0	1

Extended Euclidean Algorithm

> $225/13 = 17$ remainder 4

a	q	x	y
225	-	1	0
13	17	0	1
4			

Extended Euclidean Algorithm

> $13/4 = 3$ remainder 1

a	q	x	y
225	-	1	0
13	17	0	1
4	3		
1			

Extended Euclidean Algorithm

> $4/1 = 4$ remainder 0

a	q	x	y
225	-	1	0
13	17	0	1
4	3		
1	4		

Extended Euclidean Algorithm

> $x_{i+1} = x_{i-1} - q_i * x_i$

a	q	x	y
225	-	1	0
13	17	0	1
4	3		
1	4		

Extended Euclidean Algorithm

> $1 - 17 * 0 = 1$

a	q	x	y
225	-	1	0
13	17	0	1
4	3	1	
1	4		

Extended Euclidean Algorithm

> $0 - 3 * 1 = -3$

a	q	x	y
225	-	1	0
13	17	0	1
4	3	1	
1	4	-3	

Extended Euclidean Algorithm

> $y_{i+1} = y_{i-1} - q_i * y_i$

a	q	x	y
225	-	1	0
13	17	0	1
4	3	1	
1	4	-3	

Extended Euclidean Algorithm

> $0 - 17 * 1 = -17$

a	q	x	y
225	-	1	0
13	17	0	1
4	3	1	-17
1	4	-3	

Extended Euclidean Algorithm

> $1 - 3 * (-17) = 52$

a	q	x	y
225	-	1	0
13	17	0	1
4	3	1	-17
1	4	-3	52

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 $e \bmod \phi(n)$

$$d * e = 1 \bmod \phi(n)$$

- Publish e and n , remember d
- Encryption:

$$E(M) = M^e \bmod n$$

- Decryption:

$$D(C) = C^d \bmod n = M^{de} \bmod n = M$$

43

Public Key Cryptography (RSA)

$$D(C) = C^d \bmod n = M^{de} \bmod n = M ?$$

- Fermat's little theorem
- Chinese Remainder theorem

44

Fermat's Little Theorem

- If m is prime and a is not multiple of m then

$$a^{m-1} \bmod m = 1$$

45

Chinese Remainder Theorem

- If
 - $x = y \bmod p$
- And
 - $x = y \bmod q$
- And p is relatively prime to q , then
 - $x = y \bmod pq$

46

Public Key Cryptography (RSA)

$$D(C) = C^d \bmod n = M^{de} \bmod n =$$

$$M^{k(p-1)(q-1)+1} \bmod n =$$

$$M * M^{k(p-1)(q-1)} \bmod n$$

- Consider $X = M^{k(p-1)(q-1)} \bmod p = (M^{(p-1)} \bmod p)^{k(q-1)} = 1$
- Similarly $X = M^{k(p-1)(q-1)} \bmod q = 1$
- Using Chinese remainder theorem

$$M^{k(p-1)(q-1)} \bmod pq = 1$$

47

Public Key Cryptography (RSA)

- So 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

48

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

$$g^a \bmod n$$

- Bob chooses random number b and sends to Alice

$$g^b \bmod n$$

- Alice takes Bob's message and calculates

$$g^{ab} \bmod n$$

- Bob does the same; now they both know a secret