Cryptography Goals
- Protect private communication in the public world
- Alice and Bob are shouting messages over a crowded room – no one 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

Basic Problem and Terminology
- Alice
- M → K1(M) → K2(M)
- Bob
- C

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

What Can Go Wrong?
- Cyphertext-only attack: Eve can attempt either to learn M or to learn how to decrypt other messages by observing many ciphertexts C
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: Mallory 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: Mallory 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:
- Mallory can substitute messages
- Mallory can modify messages
  - so that they have different meaning
  - so that they are scrambled
- Mallory can drop messages
- Mallory 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.

One-time Pad

- 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 is the only cipher that cannot be broken
- Sender and receiver must be perfectly synchronized
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
- Mallory could learn the key and replace it with her own (man-in-the-middle attack)
- Mallory 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

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 $x^{ab} \mod n$
- Bob does the same; now they both know a secret

Public Key Cryptography
- 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 $f(x)$:
    - Trapdoor one-way function: There is a secret $y$ such that given $f(x)$ and $y$ it is easy to compute $x$
    - Example: Factorization of a large number is hard. Selecting two large primes, multiplying them and obtaining a large number is easy. Knowing a large number and one factor, it is easy to get another factor.
Public Key Cryptography

(RSA)

- Choose two prime numbers $p$ and $q$
  - of equal length
- Choose public key $e$ relatively prime to $(p-1)*(q-1)$ — this means that $e$ and $(p-1)*(q-1)$ have no common divisors
- Calculate $d$ which is inverse of $e$ mod $(p-1)*(q-1)$
  $$d \cdot e \equiv 1 \pmod{(p-1)*(q-1)}$$

Common Practice

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

One-Way Hash Functions

- Take a variable-length input $M$ and produce fixed-length output (hash value or message digest)
  $$h = H(M)$$
- The idea is to fingerprint $M$
  - Given $M$ easy to compute $h$
  - Given $h$ very hard to compute $M$
  - One-bit change in $M$ changes many bits in $h$
  - Good one-way hash function is collision-free: given $M$ it is very hard to find $M'$ such that $H(M) = H(M')$
- One-way hash function is public

Digital Signatures

- Proof of authorship or agreement with contents of a document
  - Signature is authentic (no one but Alice could have signed a document with her signature)
  - Signature is unforgeable
  - Signature is not reusable
  - Signed document in unalterable
  - Signature cannot be repudiated
Public-Key Signatures
- Alice encrypts the document with her private key
- Sends the signed document to Bob who decrypts it with her public key
  - This signature is reusable, Bob can take the same message and claim he received it multiple times → add timestamps
  - Signing the whole document with public key is slow → sign a hash of the document produced by one-way hash function

Digital Signatures with Encryption
- Combining digital signatures with public-key cryptography we gain security and authenticity
  - Alice first signs the message (or message digest) with her private key: $S_A(M)$
  - Alice encrypts the signed message with Bob's public key: $E_B(S_A(M))$
  - Bob decrypts the message with his private key: $D_B(E_B(S_A(M))) = S_A(M)$
  - Bob verifies Alice's signature $V_A(S_A(M)) = M$

Revisiting Cryptography Goals
- Protect private communication in the public world (Symmetric and public key cryptography)
  - Alice and Bob are shouting in a crowded room
  - No guest can understand what they are saying
- Authentication (Digital signatures)
  - Bob can verify that Alice has created the message
- Integrity (Message digests)
  - Bob can verify that message has not been modified
- Non-repudiation (Digital sig. + timestamps)
  - Alice cannot deny that she indeed sent the message

Revisiting Common Practices
- Alice and Bob exchange symmetric key using:
  - Public-key encryption
    - If they first send each other public keys, Mallory can do man-in-the-middle attack
    - If they obtain public keys from a database, Mallory can poison public-key database with bad keys
  - Diffie-Hellman key exchange
    - Mallory can do man-in-the-middle attack

Man-in-the-Middle Attack on Key Exchange
- Alice to Bob her public key Pub(A)
- Mallory captures this and sends to Bob Pub(M)
- Bob sends to Alice his public key Pub(B)
- Mallory captures this and sends to Alice Pub(M)
- Now Alice and Bob correspond through Mallory who can read all their messages
Key Exchange with Interlock Protocol

- First four steps are the same
  - Alice to Bob her public key Pub(A)
  - Mallory captures this and sends to Bob Pub(M)
  - Bob sends to Alice his public key Pub(B)
  - Mallory captures this and sends to Alice Pub(M)
- Alice encrypts a message in Pub(M) but sends half to Bob – Mallory cannot recover this message and duplicate it
- This works if Mallory cannot mimic Alice’s and Bob’s messages

Delayed Key Exchange

- Alice and Bob need not exchange keys directly to communicate
  - Alice generates a random session key $K$
  - She obtains Bob’s public key from a database and encrypts $K$ with that $E_d(K)$
  - She sends both the message encrypted with $K$, $E_d(M)$ and a key $E_d(K)$ to Bob
- This is how most real-world protocols work

Authentication

- How does Alice prove her identity?
  - When she logs on
  - When she sends messages to Bob

Authentication on Log-on

- Hackers can compile a list of frequently used passwords, apply one-way function to each and store them in a table – dictionary attack
- Host adds random salt to password, applies one-way function to that and stores result and salt value
- SKEY – Alice will have different password each time she logs on
  - To set-up the system, Alice enters random number $R$
  - Host calculates $x_0 = f(R)$, $x_1 = f(f(R))$, $x_2 = f(f(f(R)))$, ..., $x_{100}$
  - Alice keeps this list, host sets her password to $x_{100}$
  - Alice logs on with $x_{100}$, host verifies $f(x_{100}) =$ $x_{101}$, resets password to $x_{100}$
  - Next time Alice logs on with $x_{99}$
Authentication on Log-on
- Someone sniffing on the network can learn the password
- Host keeps a file of every user’s public key
- Users keep their private keys
- When Alice attempts to log on, host sends her a random number $R$
- Alice encrypts $R$ with her private key and sends to host
- Host can now verify her identity by decrypting the message and retrieving $R$

Authentication and Key Exchange
- Alice wants to exchange keys with Bob
- How can she be sure that she is talking to Bob?
- How is this solved in the real world?
- Bob gets his ID from a trusted authority – government, DMV
- Bob shows his ID to Alice

Arbitrated Key Exchange
- Trent will play the role of trusted authority
- He will arbitrate key exchange and guarantee for Alice’s and Bob’s identity

Key Exchange with Digital Signatures
- Trent signs both Alice’s and Bob’s public keys – he generates public-key certificate
- When they receive keys they verify the signature
  - Everyone has Trent’s public key
  - Mallory cannot impersonate Alice or Bob because his key is signed as Mallory’s
- Certificate usually contains more than the public key
  - Name, network address, organization
  - Trent is known as Certificate Authority (CA)