Breaking an Encryption Scheme: Ciphertext Only

- It is not difficult for a bad guy to obtain ciphertext. (If a bad guy can't access the encrypted data, then there would have been no need to encrypt the data in the first place!)
- One possible strategy (when the encryption function is known) is to search through all the keys and trying to decrypt operation with each key in turn.
- It is essential for this attack to be able to recognize when the decryption succeeds.
- It is also essential having enough ciphertext. For instance, using the example of a monoalphabetic cipher, if the only ciphertext available is XYZ, then there is not enough information. There are many possible letter substitutions that would lead to a legal three-letter English word like THE or CAT or HAT etc.

Breaking an Encryption Scheme: Known Plaintext

- Suppose an attacker has somehow obtained some <*plaintext,ciphertext*> pairs.
- For instance, the data might consist of specifying the next city to be attacked. Once the attack occurs, the plaintext to the previous day's ciphertext is now known.

- With a monoalphabetic cipher, a small amount of known plaintext would be a bonanza. Indeed from it, the attacker would learn the mappings of a substantial fraction of the most common letters.

Breaking an Encryption Scheme: Chosen Plaintext

- Rarely an attacker can choose any plaintext he wants, and get the system to tell him what the corresponding ciphertext is.
- How could such a thing be possible?
- Suppose the telegraph company offered a service in which they encrypt and transmit messages for you. Suppose Trudy had eavesdropped on Alice's encrypted message. Now she'd like to break the telegraph company's encryption scheme so that she can decrypt Alice's message.
- She can obtain the corresponding ciphertext to any message she chooses by paying the telegraph company to send the message for her, encrypted.
- For instance, if Trudy knew they were using a monoalphabetic cipher, she might send the message knowing that she would thereby get all the letters of the alphabet encrypted and then be able to decrypt with certainty any encrypted message.
- Clearly this is a further advantage for the attacker.

- A cryptosystem should resist all three sorts of attacks (ciphertext only, known plaintext, chosen plaintext)!

Types of Cryptographic Functions

- *Secret Key Cryptography (*sometimes referred to as *Symmetric Cryptography)*.

- *Public Key Cryptography* (sometimes referred to as *Asymmetric Cryptography*).

- *Hash Algorithms.*

Use of secret key

- Secret key cryptography involves the use of a single key. Given a message (plaintext) and the key, encryption produces unintelligible data which is about the same length as the plaintext was. Decryption is the reverse of encryption, and uses the same key as encryption.

Use of secret key: Transmitting Over an Insecure Channel

- It is often impossible to prevent eavesdropping when transmitting information. For instance, a telephone conversation can be tapped, a letter can be intercepted, a message transmitted on a LAN or packets travelling on internet can be received by unauthorized party.
- If you and I agree on a shared secret (a key), then by using secret key cryptography we can send messages to one another on a medium that can be tapped, without worrying about eavesdroppers. All we need to do is have the sender encrypt the messages and the receiver decrypt them using the shared secret. An eavesdropper will only see unintelligible data.

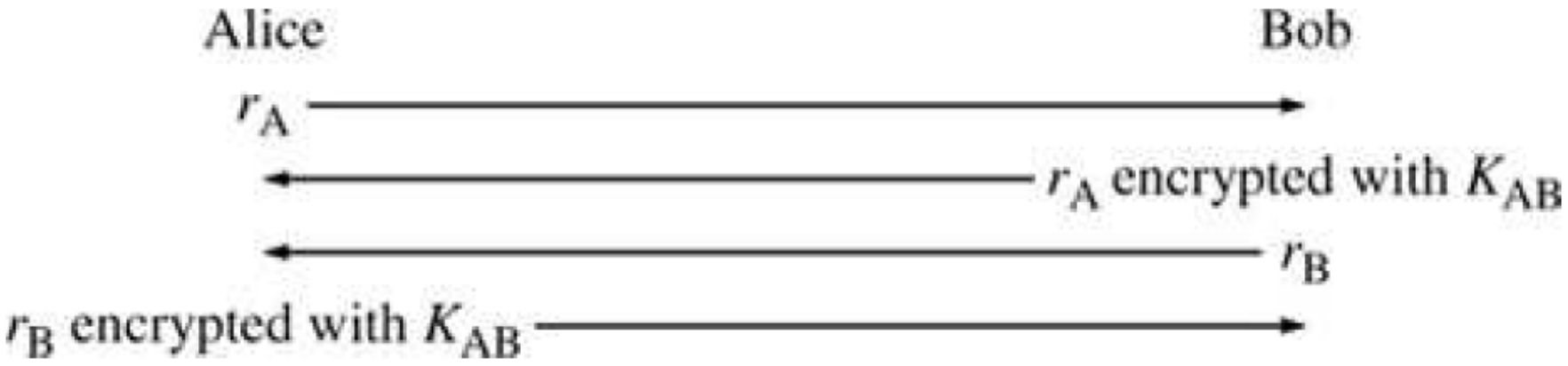
Use of secret key: Secure Storage on Insecure Media

- If I have information I want to preserve but which I want to ensure no one else can look at, I have to be able to store the media where I am sure no one can get it.
- There are very few places that are truly secure, and usually they are very expensive.
- If I invent a key and encrypt the information using the key, I can store it anywhere and it is safe so long as I can remember the key. Of course, forgetting the key makes the data irrevocably lost, so this must be used with great care!

Use of secret key: Strong Authentication

- In spy movies, when two secret agents who don't know each other must rendezvous, they are each given a password or pass phrase that they can use to recognize one another. This has the problem that anyone overhearing their conversation or initiating one falsely can gain information useful for replaying later and impersonating the person to whom they are talking.
- The term *strong authentication* means that someone can prove knowledge of a secret without revealing it.
- Strong authentication is possible with cryptography. Suppose Alice and Bob share a key K_{AB} and they want to verify they are speaking to each other. They each pick a random number, which is known as a challenge. Alice picks r_A. Bob picks r_B. The value x encrypted with the key K_{AB} is known as the response to the challenge. How Alice and Bob use challenges and responses to authenticate each other is shown in the next slide.

Use of secret key: Authentication



Use of secret key: Integrity Check

- An ordinary (noncryptographic) checksum protects against accidental corruption of a message.
- Break a message into fixed-length blocks (for instance, 32-bit words) and adding them up. The sum is sent along with the message. The receiver similarly breaks up the message, repeats the addition, and checks the sum.
- If the message had been garbled, the sum will not match the sum sent and the message is rejected.
- However it only protects against faulty hardware and not an intelligent attacker.

Use of secret key: Integrity Check

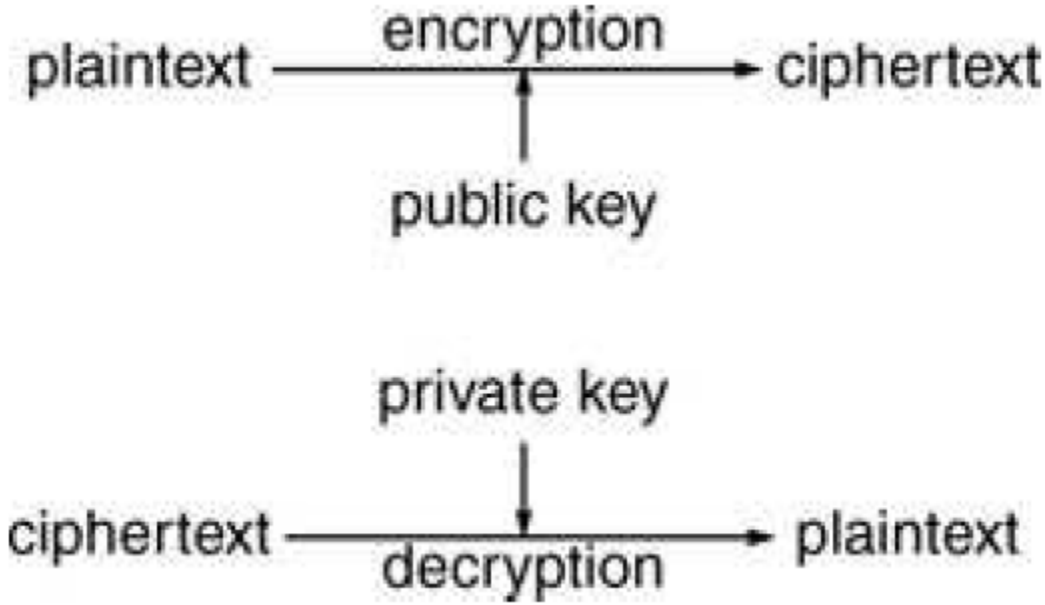
- To provide protection against malicious changes to a message, a secret checksum algorithm is required, such that an attacker not knowing the secret can't compute the right checksum for the message to be accepted as authentic.
- Given a key and a message, the algorithm produces a fixed-length message integrity code (MIC), (also MAC = Message Authentication code) that can be sent with the message.

- If anyone were to modify the message, and they didn't know the key, they would have to guess a MIC and the chance of getting it right depends on the length. A typical MIC is at least 48 bits long, so the chance of getting away with a forged message is only one in 280 trillion.

- Such message integrity codes have been in use to protect the integrity of large interbank electronic funds transfers for quite some time.

Use of public key

- Each individual has two keys: a private key that need not be revealed to anyone, and a public key that is preferably known to the entire world.
- Public key cryptography can do anything secret key cryptography can do, but in general public key cryptographic algorithms are slower than secret key cryptographic algorithms



Use of public key: Transmitting Over an Insecure Channel

- Suppose Alice's <public key, private key> pair is $<e_A, d_A>$.
- Suppose Bob's key pair is $<e_B, d_B>$.
- Assume Alice knows Bob's public key, and Bob knows Alice's public key.

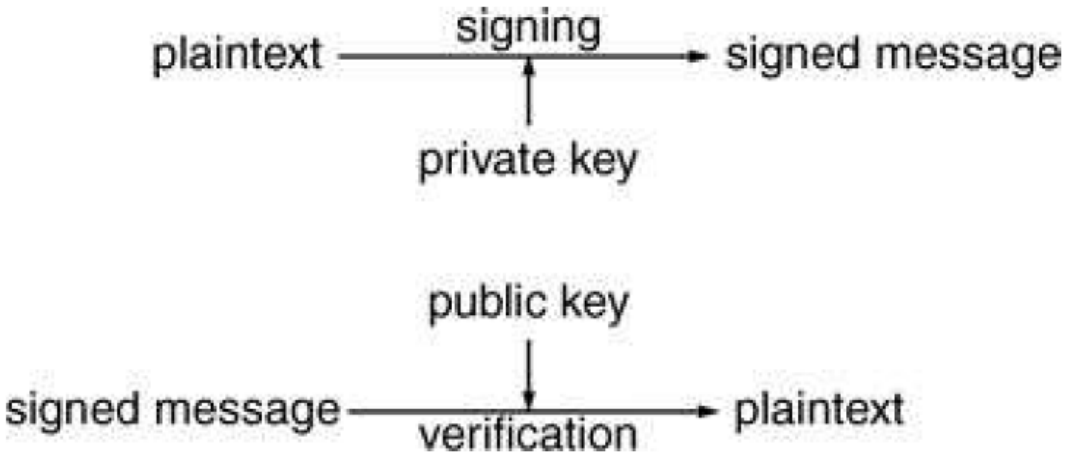
Use of public key: Authentication

- Here's an example of how Alice can use public key cryptography for verifying Bob's identity assuming Alice knows Bob's public key.
- Alice chooses a random number r, encrypts it using Bob's public key e_B, and sends the result to Bob. Bob proves he knows d_B by decrypting the message and sending r back to Alice.

- An important advantage of public key authentication is that Alice does not need to keep any secret information in order to verify Bob.

Use of public key: Digital Signatures

- A digital signature is a number associated with a message, like a checksum or the MIC described before.
- A public key signature differs from a secret key MIC because verification of a MIC requires knowledge of the same secret as was used to create it. Therefore anyone who can verify a MIC can also generate one, and so be able to substitute a different message and corresponding MIC.
- In contrast, verification of the signature only requires knowledge of the public key. So Alice can sign a message by generating a signature only she can generate, and other people can verify that it is Alice's signature, but cannot forge her signature.



Use of Hash Algorithms

- Hash algorithms are also known as message digests or one-way transformations.
- A cryptographic hash function is a mathematical transformation that takes a message of arbitrary length (transformed into a string of bits) and computes from it a fixed-length (short) number.
- We'll call the hash of a message *m*, *h(m)*. It has the following properties:
  - For any message m, it is relatively easy to compute h(m). This just means that in order to be practical it can't take a lot of processing time to compute the hash.
  - Given *h(m)*, there is no way to find an *m* that hashes to h(m) in a way that is substantially easier than going through all possible values of *m* and computing *h(m)* for each one.
  - Even though it's obvious that many different values of *m* will be transformed to the same value *h(m)* (because there are many more possible values of m), it is computationally infeasible to find two values that hash to the same thing.

Use of Hash Algorithms

- An example of the sort of function that might work is taking the message m, treating it as a number, adding some large constant, squaring it, and taking the middle n digits as the hash.
- You can see that while this would not be difficult to compute, it's not obvious how you could find a message that would produce a particular hash, or how one might find two messages with the same hash.

Use of Hash Algorithms: Password Hashing

- When a user types a password, the system has to be able to determine whether the user got it right.
- If the system stores the passwords unencrypted, then anyone with access to the system storage or backup tapes can steal the passwords. Luckily, it is not necessary for the system to know a password in order to verify its correctness.
- Instead of storing the password, the system can store a hash of the password. When a password is supplied, it computes the password's hash and compares it with the stored value. If they match, the password is deemed correct. If the hashed password file is obtained by an attacker, it is not immediately useful because the passwords can't be derived from the hashes.

Use of Hash Algorithms: Message integrity

- Cryptographic hash functions can be used to generate a MAC (or MIC) to protect the integrity of messages transmitted over insecure media in much the same way as secret key cryptography.
- Alice and Bob agree on a secret (key), Alice can use a hash to generate a MAC for a message to Bob by taking the message, concatenating the secret, and computing the hash of message|secret. This is called a keyed hash. Alice then sends the hash and the message (without the secret) to Bob. Bob concatenates the secret to the received message and computes the hash of the result. If that matches the received hash, Bob can have confidence the message was sent by someone knowing the secret.

Use of Hash Algorithms: Message fingerprint

- If you want to know whether some large data structure (e.g. a program) has been modified from one day to the next, you could keep a copy of the data and periodically compare it to the active version.
- With a hash function, you can save storage: you simply save the hash of the data. If the hash hasn't changed, you can be confident none of the data has.