CIS 6930 Emerging Topics in Network Security

Topic 2. Network Security Primitives

Outline

- Absolute basics
  - Encryption/Decryption;
  - Digital signatures;
  - D-H key exchange;
  - Hash functions;
  - Application of hash functions
  - Pseudo random generators;
  - traditional key distribution techniques
  - Authentication
  - Public key infrastructure

CIS 6930 Emerging Topics in Network Security

Topic 2.1 Absolute Basics

Cryptography

- *Cryptography*: the art of secret writing
- Converts data into unintelligible (random-looking) form
  - Must be *reversible* (can recover original data without loss or modification)

Encryption/Decryption

- Plaintext: a message in its original form
- Ciphertext: a message in the transformed, unrecognized form
- Encryption: the process that transforms a plaintext into a ciphertext
- Decryption: the process that transforms a ciphertext to the corresponding plaintext
- Key: the value used to control encryption/decryption.

Cryptanalysis

- Cryptanalysis: the art of revealing the secret
  - Defeat cryptographic security systems
  - Gain access to the real contents of encrypted messages
  - Cryptographic keys can be unknown
- Difficulty depends on
  - Sophistication of the encryption/decryption
  - Amount of information available to the code breaker

Ciphertext Only Attacks

- An attacker intercepts a set of ciphertexts
- Breaking the cipher: analyze patterns in the ciphertext
  - provides clues about the plaintext and key

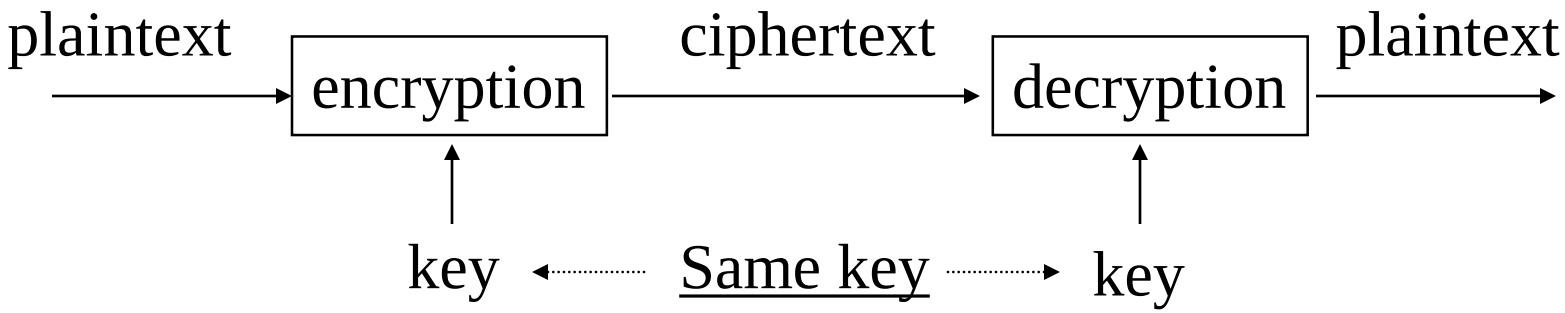
Known Plaintext Attacks

- An attacker has samples of both the plaintext and its encrypted version, the ciphertext

Chosen Plaintext Attacks

- An attacker has the capability to choose arbitrary plaintexts to be encrypted and obtain the corresponding ciphertexts
  - Difference between known plaintext and chosen plaintext attacks

Secret Key Cryptography



- Same key is used for encryption and decryption
- Also known as
  - Symmetric cryptography
  - Conventional cryptography

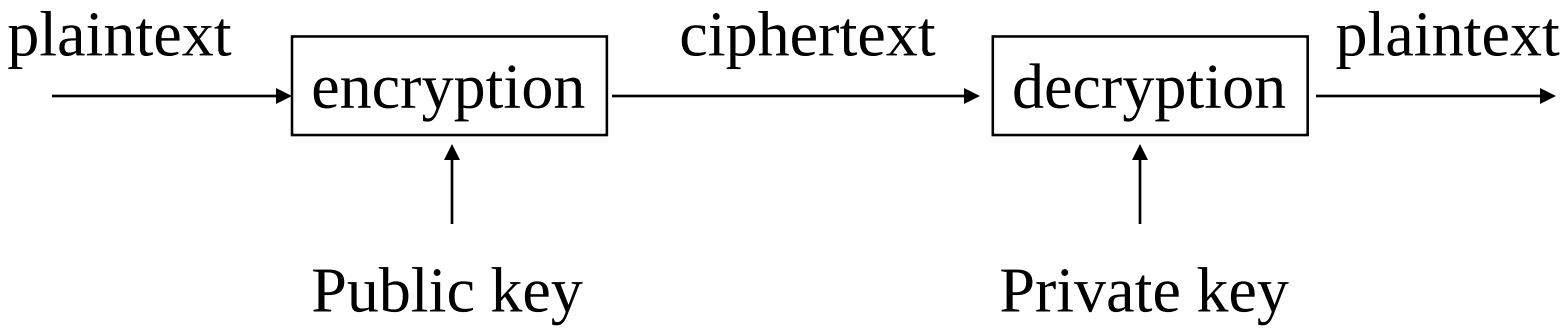
Exercise

- Alice wants to send a message to Bob. The message content is “sell the business”.
- Alice encrypts the message by replacing each letter with the one 3 letters later in the alphabet (e.g., a -> d, b - > e)
  - What is the plaintext?
    - sell the business
  - What is the ciphertext?
    - vhoo wkh exvlqhvv
  - How can Bob recover the original plaintext?
  - What is the key in this cryptosystem?
    - 3
  - Cryptanalysis attacks?
    - Known plaintext attacks

Exercise

- Alice wants to send a message to Bob. The message content is “surrender” and the ASCII codes of the message is 73 75 72 72 65 6E 64 65 72 (in hexdecimal)
- Alice encrypts the message by XORing the message with a random bit stream 24 61 80 62 1F 7A 5D 36 BC (e.g., 1 XOR 0 = 1)
  - What is the plaintext (in ASCII codes) ?
    - 73 75 72 72 65 6E 64 65 72
  - What is the ciphertext?
    - 57 14 F2 10 7A 14 39 53 CE
  - How can Bob recover the original plaintext?
  - What is the key in this cryptosystem?
    - 24 61 80 62 1F 7A 5D 36 BC
  - Cryptanalysis attacks?
    - Known plaintext attacks

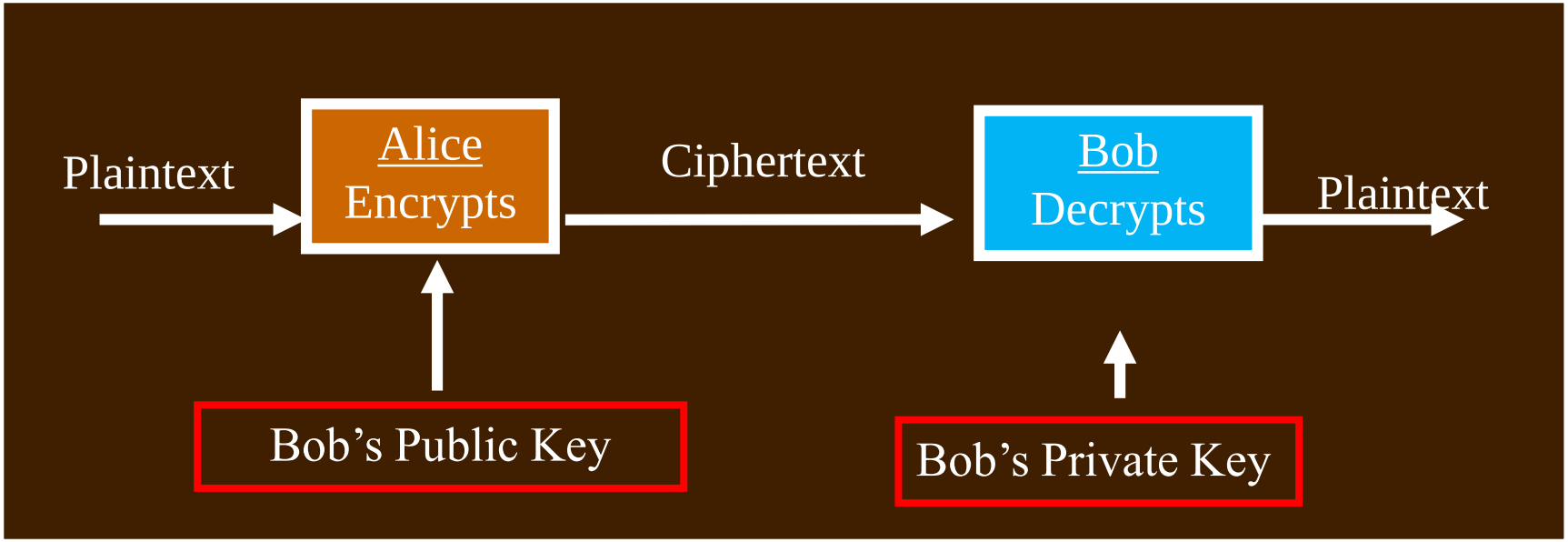
Public Key Cryptography



- Invented/published in 1975
- A public/private key pair is used
  - Public key can be publicly known
  - Private key is kept secret by the owner of the key
- Much slower than secret key cryptography
- Also known as
  - Asymmetric cryptography

Applications

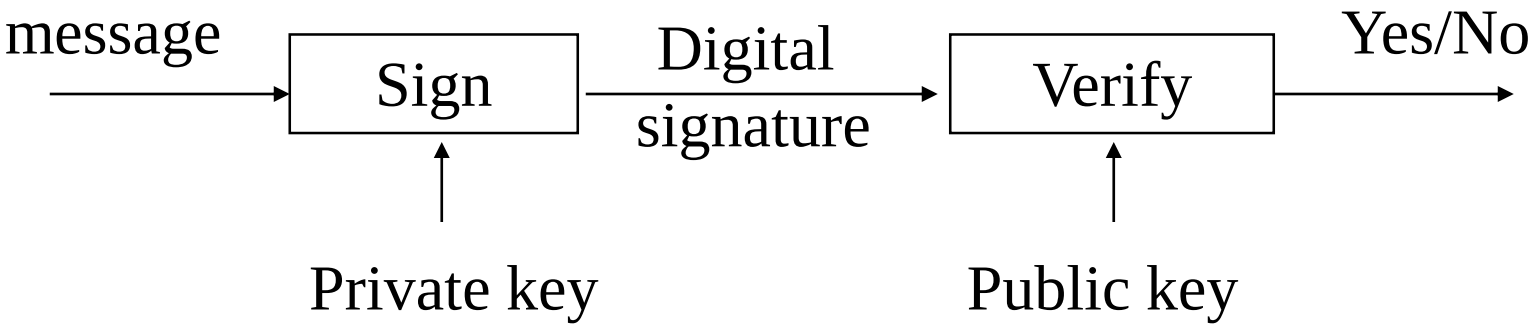
1. Communicating securely over an insecure channel
   - Alice encrypts plaintext using Bob's public key, and Bob decrypts ciphertext using his private key
   - No one else can decrypt the message (because they don't have Bob's private key)



Applications (Cont'd)

2. Secure storage
   - Users encrypt data using the storage provider's public key
3. *User Authentication*
   - Bob proves his identity to Alice by using his private key to perform an operation (without disclosing his private key)
   - Alice verifies result using Bob's public key
4. Establishing pre-shared secret key

Public Key Cryptography (Cont'd)



- Another mode: digital signature
  - Only the party with the private key can create a digital signature.
  - The digital signature is verifiable by anyone who knows the public key.
  - The signer cannot deny that he/she has done so.

RSA (Rivest, Shamir, Adleman)

- The most popular public key method
  - provides both public key encryption and digital signatures
- Basis: factorization of large numbers is hard
- Variable key length (1024 bits or greater)
- Variable plaintext block size
  - plaintext block size must be smaller than key size
  - ciphertext block size is same as key size

Some Terms in Number Theory

- *Prime number*
  - *a is prime* if it can only be divided by 1 and itself
  - examples: 2, 3, 5, 7, 11, 13, 17, 19, 31,…
- Greatest common divisor
  - $gcd(a,b) = \max\{k \mid k|a \text{ and } k|b\}$
  - Example: gcd(60,24) = 12
- Relatively prime
  - Integers *a* and *b* are *relatively prime* iff $\gcd(a,b) = 1$
  - example: 8 and 15 are relatively prime

Some Terms in Number Theory (Cont'd)

- *Modular operation*
  - For any integer *a* and any positive integer *n*, there are two unique integers *q* and *r*, such that $0 \leq r < n$ and $a = qn + r$. Modular operation finds r
  - Example: 13 mod 5 = 3 (13 = 2*5 + 3)
- Multiplicative inverse
  - z is the multiplicative inverse of m over modular n if (m * z) mod n = 1 (0 < m < n)
  - Example: 12 and 3 are multiplicative inverses of each other (over modular 35), because $12*3 \bmod 35 = 1$
  - May not always exist. A positive integer m has a multiplicative inverse over modular n iff m and n are relatively prime

Generating a Public/Private Key Pair

- Find large primes *p* and *q*
- Let $n = p*q$
  - $\phi(n) = (p-1)*(q-1)$
    - $\phi(n)$: the number of positive integers that are less than n and relatively prime to n
    - If n can be factored into the product of two prime numbers, then $\phi(n)$ *can be computed by* $(p-1)*(q-1)$
- Choose an *e* that is relatively prime to $\phi(n)$
  - **public** key = $<e,n>$
- Find *d* = multiplicative inverse of *e* mod $\phi(n)$ (i.e., $e*d = 1 \bmod \phi(n)$)
  - **private** key = $<d,n>$

RSA Operations

- For plaintext message m and ciphertext c

Encryption: $c = m^e \bmod n$, $m < n$

Decryption: $m = c^d \bmod n$

Signing: $s = m^d \bmod n$, $m < n$

Verification: $m = s^e \bmod n$

RSA Example: Encryption and Signing

- Choose p = 23, q = 11 (both primes)
  - $n = p*q = 253$
  - $\phi(n) = (p-1)(q-1) = 220$
- Choose e = **39** (relatively prime to 220)
  - public key = <**39**, 253>
- Find e^{-1} mod 220 = d = **79**
  (note: $39*79 \equiv 1 \bmod 220$)
  - private key = <**79**, 253>

Example (Cont'd)

- Suppose plaintext **m = 80**

Encryption

c = 80^{39} mod 253 = **37** ($c = m^e \bmod n$)

Decryption

m = 37^{79} mod 253 = **80** ($c^d \bmod n$)

Signing (in this case, for entire message **m**)

s = $\mathbf{80}^{79}$ mod 253 = 224 ($s = m^d \bmod n$)

Verification

m = 224^{39} mod 253 = **80** ($s^e \bmod n$)

Hash Algorithms

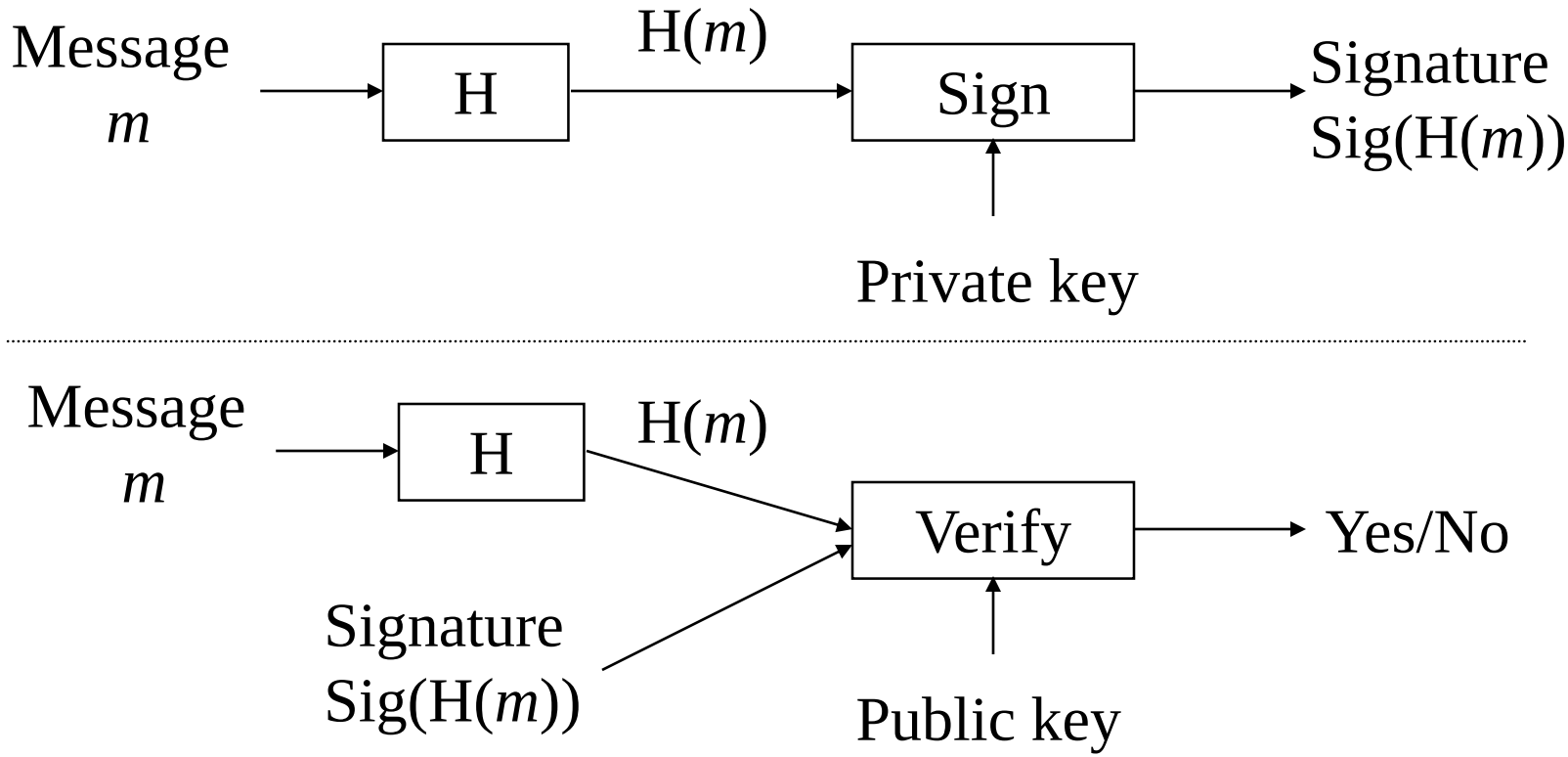
- Also known as
  - Message digests
  - One-way transformations
  - One-way functions
  - Hash functions
- Length of $H(m)$ much shorter then length of m
- Usually fixed lengths: 128 or 160 bits

Hash Algorithms (Cont'd)

- Desirable properties of hash functions
  - Performance: Easy to compute $H(m)$
  - One-way property: Given $H(m)$ but not m, it is computationally infeasible to find m
  - Weak collision free: Given $H(m)$, it is computationally infeasible to find m' such that $H(m') = H(m)$.
  - Strong collision free: Computationally infeasible to find m_1, m_2 such that $H(m_1) = H(m_2)$
- Example algorithms
  - MD5
  - SHA-1
  - SHA-256

Applications of Hash Functions

- Primary application
  - Generate/verify digital signature



Applications of Hash Functions (Cont'd)

- Password hashing
  - Doesn't need to know password to verify it
  - Store $H(password+salt)$ and salt, and compare it with the user-entered password
  - Salt makes dictionary attack more difficult
- Message integrity
  - Agree on a secrete key k
  - Compute $H(m|k)$ and send with m
  - Doesn't require encryption algorithm, so the technology is exportable

Applications of Hash Functions (Cont'd)

- Authentication
  - Give $H(m)$ as an authentication token
  - Later release m
  - Which property is used?

Pseudo Random Generator

- Definition
  - A cryptographically secure pseudorandom bit generator is an efficient algorithm that will expand a random *n*-bit seed to a longer sequence that is computationally indistinguishable from a truly random sequence.
- Theorem [Levin]
  - A one-way function can be used to construct a cryptographically secure pseudo-random bit generator.

Key Agreement

- Establish a key between two or among multiple parties
  - Classical algorithm
    - Diffie-Hellman

Diffie-Hellman Key Exchange

Diffie-Hellman Protocol

- For negotiating a shared secret key using only public communication
- Does not provide authentication of communicating parties
- What's involved?
  - *p* is a large prime number (about 512 bits)
  - *g* is an integer less than *p (g is chosen based on p)*
  - *p* and *g* are publicly known

D-H Key Exchange Protocol

Alice	Bob
Publishes g and p	Reads g and p
Picks random number S_A *(and keeps private)*	Picks random number S_B *(and keeps private)*
Computes $T_A = g^{S_A} \bmod p$	Computes $T_B = g^{S_B} \bmod p$
Sends T_A to Bob,	Sends T_B to Alice,
Computes $T_B^{S_A} \bmod p$ =	Computes $T_A^{S_B} \bmod p$

Key Exchange (Cont'd)

Alice and Bob have now both computed the same secret $g^{S_A S_B}$ mod p, which can then be used as the shared secret key K

S_A is called the discrete logarithm of g^{S_A} mod p and

S_B is called the discrete logarithm of g^{S_B} mod p

D-H Example

- Let p = 353, g = 3
- Let random numbers be S_A = 97, S_B = 233
- Alice computes T_A = ___ mod __ = 40 = $g^{S_A} \bmod p$
- Bob computes T_B = ___ mod ___ = 248 = $g^{S_B} \bmod p$
- They exchange T_A and T_B
- Alice computes K = __ mod __ = **160** = $T_B^{S_A} \bmod p$
- Bob computes K = __ mod ___ = **160** = $T_A^{S_B} \bmod p$

Why is This Secure?

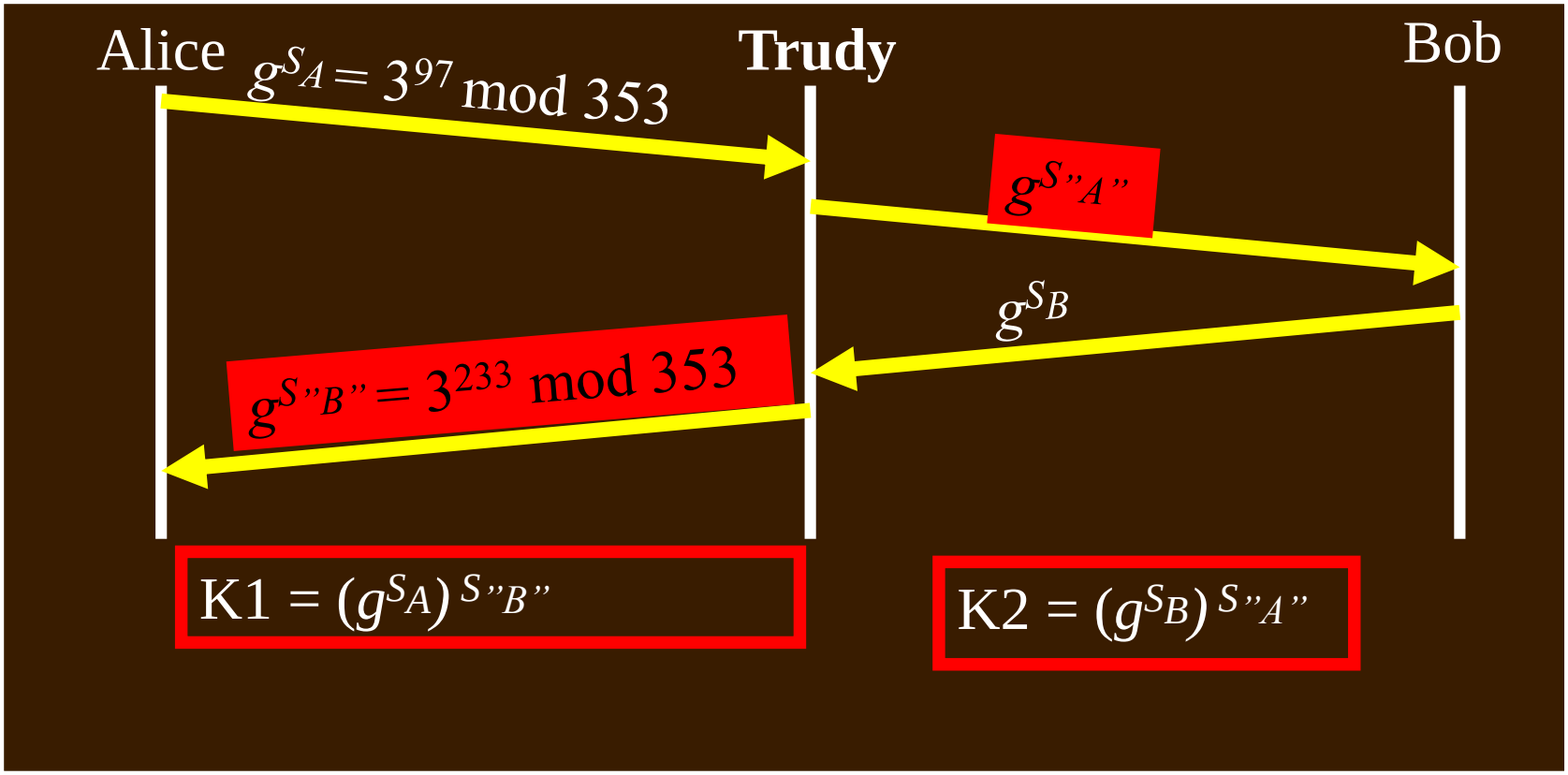
- Discrete log problem is hard:
  - given a^x mod b, a, and b, it is computationally infeasible to compute x

D-H Limitations

- Expensive exponential operation is required
- Algorithm is useful for key negotiation only
  - i.e., not for public key encryption/verification
- Not for user authentication
  - In fact, you can negotiate a key with a complete stranger!

Man-In-The-Middle Attack

- Trudy impersonates as Alice to Bob, and also impersonates as Bob to Alice

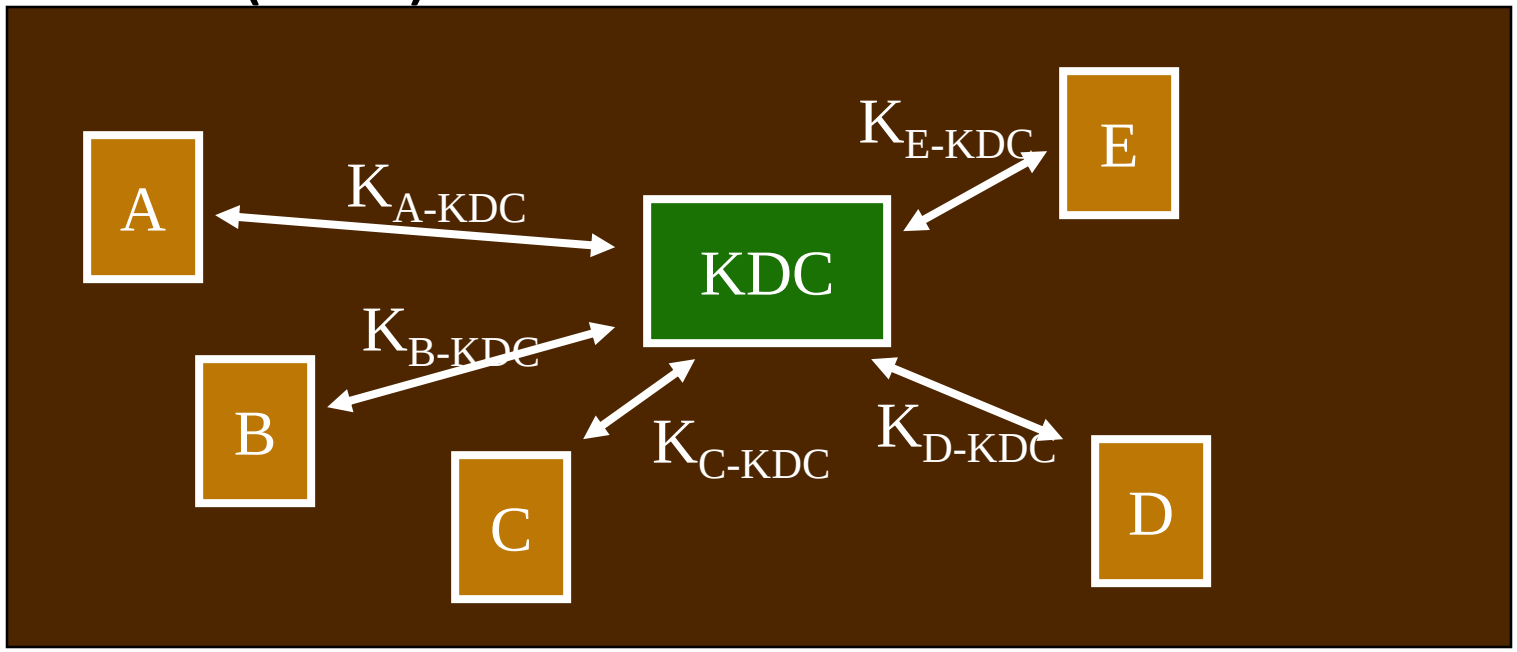


Key Distribution

- Involves a (trusted) third party to help establish keys.
- Based on
  - Symmetric key cryptography, or
  - Public key cryptography

KDC (Cont'd)

- Shared keys between the Key Distribution Center (KDC) and users



Key Distribution Center (KDC)

- Key Distribution Center (KDC)
  - Communication parties depend on KDC to establish a pair-wise key.
  - The KDC generates the cryptographic key
  - Pull based
    - Alice communicates with the KDC before she communicates with Bob
  - Push based
    - Alice communicates with Bob, and it's Bob's responsibility to contact the KDC to get the pair-wise key.

When Public Key Cryptography is Used

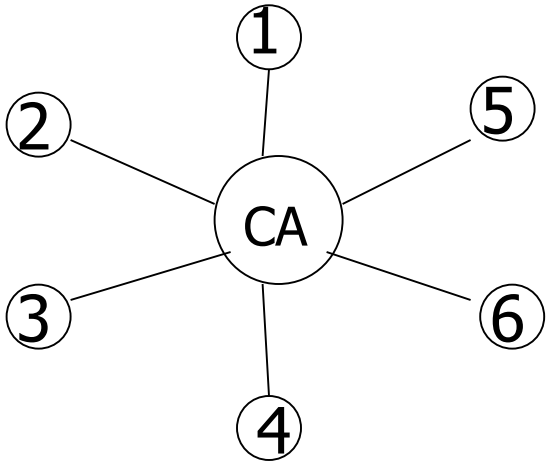
- Need to authenticate public keys
- Public key certificate
  - Bind an identity and a public key together
  - Verify the authenticity of a party's public key

What Is Public Key Infrastructure (PKI)

- Informally, the infrastructure supporting the use of public key cryptography.
- A PKI consists of
  - Certificate Authority (CA)
  - Certificates
  - A repository for retrieving certificates
  - A method of revoking/updating certificates

Certification Authorities (CA)

- A CA is a trusted node that maintains the public keys for all nodes (Each node maintains its own private key)



If a new node is inserted in the network, only that new node and the CA need to be configured with the public key for that node

Certificates

- A CA is involved in authenticating users' public keys by generating certificates
- A certificate is a signed message vouching that a particular name goes with a particular public key
- Example: $[\text{Alice's public key is } 876234]_{carol}$

- Knowing the CA's public key, users can verify the certificate and authenticate Alice's public key

Certificates

- Certificates can hold expiration date and time
- Alice keeps the same certificate as long as she has the same public key and the certificate does not expire
- Alice can append the certificate to her messages so that others know for sure her public key

CA Advantages

1. The CA does not need to be online. [Why?]

2. If a CA crashes, then nodes that already have their certificates can still operate.

3. Certificates are not security sensitive (in terms of confidentiality).
   - Can a compromised CA decrypt a conversation between two parties?
   - Can a compromised CA fool Alice into accepting an incorrect public key for Bob, and then impersonate Bob to Alice?

Summary

- An important security term
  - Authentication
- Cryptography
  - Secret key cryptography (one shared key)
  - Public key cryptography (two keys: public and private keys)
  - Hash functions (output is of fixed length)
    - One-way property
- Diffie-Hellman key establishment (man-in-the-middle attacks)
- Key distribution - KDC (based on secret key cryptography)
- Public key infrastructure PKI (certificate)