Diffie-Hellman Key Exchange and Digital Signatures

Dr. Demetrios Glinos
University of Central Florida

CIS3360 - Security in Computing

Readings

- "Computer Security: Principles and Practice", 3rd Edition, by William Stallings and Lawrie Brown
  - Section 21.4, 2.4

Outline

- The Diffie-Hellman Key Exchange Protocol
  - Overview
  - Primitive Roots
  - The Protocol
  - Key Exchange Example
  - Man-in-the-middle Attack on DH
- Digital Signatures
  - The Digital Signature Problem
  - The RSA Signature Scheme
  - Using Hash Functions with Digital Signatures

Outline

- Topic 1: The Diffie-Hellman Key Exchange Protocol

Diffie-Hellman Key Exchange Overview

- First public-key algorithm
- A way for two parties to use *public key cryptography* to negotiate *a shared secret key* for *symmetric encryption*
- Global public elements (assumed to be known generally, even by attacker):
  - **p** – prime number
  - α – a primitive root of p
- Suppose also that each participant has his/her own **private key**.
- After the exchange,
  - both participants will have computed a **shared secret key** *without disclosing their private keys to each other!*
  - attacker doesn't know either private key or the shared secret key
- Depends for its effectiveness on the difficulty for an attacker to compute the shared secret key even knowing p, α, and intercepted communications

Primitive Roots

- A number α is a **primitive root** (or "generator") of a *prime number* **p if and only if**
  $\alpha^1, \alpha^2, \ldots\ldots, \alpha^{p-1}$ are all **distinct** (mod p)

- *i.e., the powers of α generate all the numbers from 1 to p-1, where $\alpha < p$, p prime*

- Example: For the prime number **7**, the numbers 3 and 5 are primitive roots. For the primitive root **3**, we have:

  $3^1 = 3 \bmod 7 = 3$,
  $3^2 = 9 \bmod 7 = 2$,
  $3^3 = 27 \bmod 7 = 6$,
  $3^4 = 81 \bmod 7 = 4$,
  $3^5 = 243 \bmod 7 = 5$,
  $3^6 = 729 \bmod 7 = 1$

- Example: For the prime number 19, the primitive roots are 2, 3, 10, 13, 14, 15

Diffie-Hellman Key Exchange Protocol

- Given the global public elements:
  - p – prime number
  - α - a primitive root of p

- Alice picks a random $x < p$ and computes ← *little x is Alice's private key*
  $X = \alpha^x \bmod p$, and sends this to Bob ← *big X is what Alice sends to Bob*

- Bob picks a random $y < p$ and computes ← *little y is Bob's private key*
  $Y = \alpha^y \bmod p$, and sends this to Alice ← *big Y is what Bob sends to Alice*

- Alice receives Y from Bob and computes: $K = Y^x \bmod p$
- Bob receives X from Alice and computes: $K = X^y \bmod p$

- Alice and Bob have computed the same secret key, because what Alice computed is $Y^x \bmod p = \alpha^{yx} \bmod p = \alpha^{xy} \bmod p = X^y \bmod p$, which is what Bob computed

Diffie-Hellman Key Exchange Example

- Given ***prime* p = 71** and ***a primitive root* 7**.
- Alice randomly selects private key **x = 5**

  So, Alice computes her public key $X = 7^5 \bmod 71$

  $= 7^3 \times 7^2 \bmod 71$

  $= 59 \times 49 \bmod 71 = \boxed{51}$ ← *what Alice sends to Bob*
- Bob randomly selects private key **y = 12**

  So, Bob computes the public key $Y = 7^{12} \bmod 71$

  $= 51 \times 51 \times 49 \bmod 71$

  $= 45 \times 49 \bmod 71 = \boxed{4}$ ← *what Bob sends to Alice*

Alice computes $K = 4^5 \bmod 71 = 1024 \bmod 71 = 30$ ← *the shared secret key*

Bob computes $K = 51^{12} \bmod 71 = 51^2 \times 51^2 \times 51^2 \times 51^2 \times 51^2 \times 51^2 \bmod 71$

$= 45 \times 45 \times 45 \times 45 \times 45 \times 45 \bmod 71 = 45^3 \times 45^3 \bmod 71$

$= 32 \times 32 \bmod 71 = 1024 \bmod 71 = 30$ ← *the shared secret key*

Differences from RSA

- RSA
  - PU=(e,n) where n is **composite** and e is **relatively prime** to ϕ(n)
  - private key d is the **inverse** of e modulo ϕ(n)

- Diffie-Hellman
  - use p and α, where p is **prime** and α is a **primitive root** of p
  - private keys for Alice and Bob can be **any values** less than p

Man-in-the-middle attack on DH

- Possible **if the attacker can intercept and modify messages**
- Global public elements:
  - **p – prime number; α - a primitive root of p**

- Evil **Man-in-the-middle Attacker (Eve)** picks **s** and **t** < **p**
- Alice sends $X = \alpha^x \bmod p$ to Bob. Eve reads it and replaces it by $T = \alpha^t \bmod p$.
- Bob sends $Y = \alpha^y \bmod p$ to Alice. Eve reads it and replaces it by $S = \alpha^s \bmod p$.
- Alice and Eve compute $K1 = S^x \bmod p$ ($= \alpha^{sx} \bmod p = \alpha^{xs} \bmod p = X^s \bmod p$)
- Bob and Eve computes $K2 = T^y \bmod p$ ($= \alpha^{ty} \bmod p = \alpha^{yt} \bmod p = Y^t \bmod p$)
- When Alice sends message to Bob encrypted with K1, Eve decrypts it, re-encrypts with K2 and sends to Bob
- When Bob sends message to Alice encrypted with K2, Eve decrypts it, re-encrypts with K1 and sends to Alice

→ Alice and Bob do not know they are using different keys!

Outline

- Topic 2: Digital Signatures

The Digital Signature Problem

- Suppose Bob receives a message *claiming* to be from Alice.
  - **How does he know that Alice sent it?**
  - **How does he know that it hasn't been altered?**

→ A **digitally signed** message from Alice provides these assurances:

- **Authentication** and **non-repudiation** – assurance that Alice sent it
- **Integrity** – assurance that the message was not altered in transit

The RSA Signature Scheme

- Recall:
- In the standard RSA encryption/decryption process,
  - ***sender*** encrypts using the *recipient's public key*
  - ***recipient*** who uses *his/her own private key*

- **For digital signatures**, however, it is the ***sender*** who uses the *private key*
  - The ***sender*** must use *his or her own private key*
  - The ***recipient*** uses the *sender's public key*

- *Why this works*
  - *When the recipient uses the sender's public key to decrypt, the recipient is assured that the sender sent the message and that the message was not altered in transit,* ***since only the sender could have encrypted it using the sender's private key****.*

Comparing RSA Operation and RSA Digital Signature

- Suppose Alice has public key $PU_A=\{ e_A, n_A \}$ and private key $PR_A =\{ d_A, n_A \}$
- Suppose Bob has public key $PU_B=\{ e_B, n_B \}$ and private key $PR_B =\{ d_B, n_B \}$

- *Alice sends encrypted message to Bob*
  - Using standard RSA encryption
  - Alice *encrypts using Bob's public key* by computing: $C = M^{e_B} \bmod n_B$
  - Bob *decrypts using his own private key* by computing: $M = C^{d_B} \bmod n_B$

- *Alice sends digitally signed message to Bob*
  - Using RSA algorithm in reverse
  - Alice ***digitally signs using her own private key*** by computing: $C = M^{d_A} \bmod n_A$
  - **Note: Since anyone, including Eve, can decrypt this message using Alice's public key, she should use encryption to send it to Bob**
  - ***Bob decrypts the ciphertext (if encrypted), and then decrypts digitally signed message using Alice's public key***: $M = C^{e_A} \bmod n_A$

Non-mutability

- Digital signature goal is to create something unique, that only Bob could have generated. This is called ***non-mutability***.

- Non-mutability is not always achieved by the RSA signature scheme
  - Suppose Eve, the attacker, has two valid signatures from Bob
    - $S_1 = M_1^d \bmod n$
    - $S_2 = M_2^d \bmod n$
  - Eve could then produce a new signature
    - $(S_1)(S_2) \bmod n = ((M_1)(M_2))^d \bmod n$
    - This would validate as a verifiable signature from Bob for $(M_1)(M_2)$
  - *This problem is avoided in practice by using digital signatures with cryptographic hash functions, which fixes this problem*

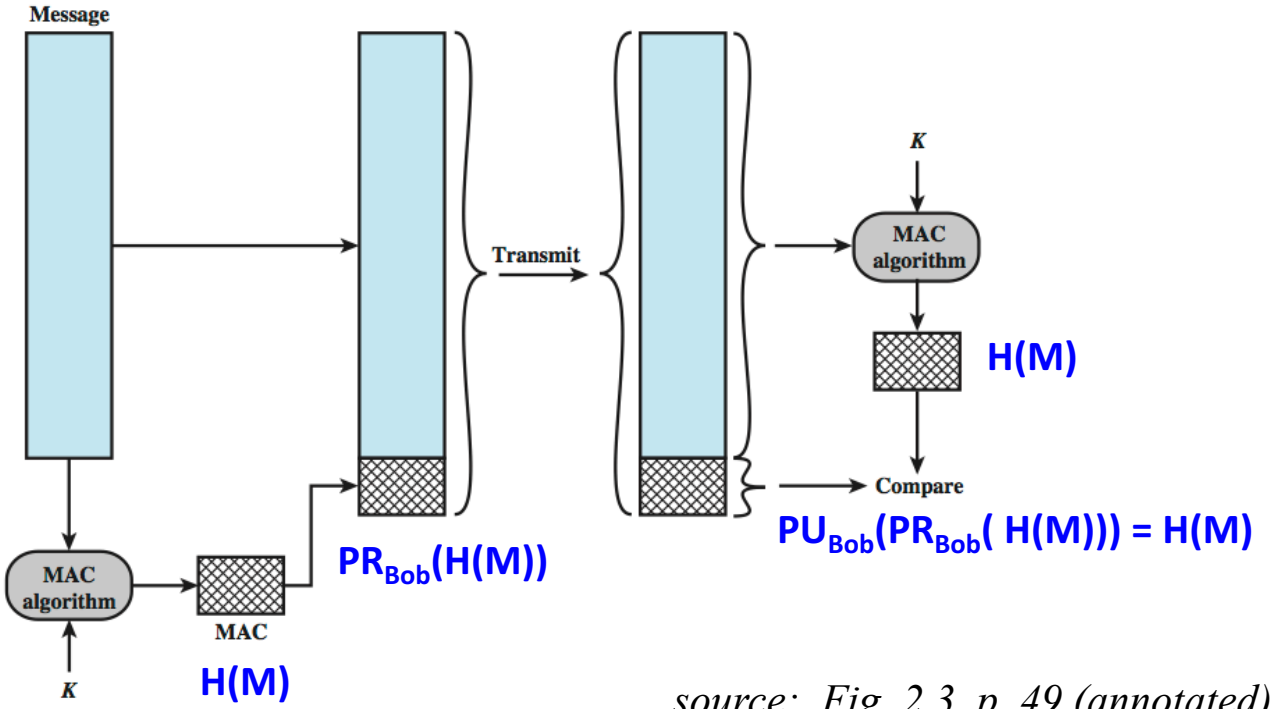
Using Hash Functions with Digital Signatures

- **In practice, digital signatures are not applied to entire messages**
  - Because public key encryption algorithms are slow compared with symmetric algorithms
  - And also because it is possible to construct valid RSA signatures on combined messages from existing RSA signatures (see previous slide)
- So, digital signatures are more commonly applied to **cryptographic hashes** of messages, which serve as **message authentication codes**
- Suppose Bob wishes to send a message, M, to Alice:
  - Cryptographic hash of message: **H(M)**
  - Bob *digitally signs the hash value* using his private key: **PR_{Bob}(H(M))**
  - What Bob sends to Alice: **PR_{Bob}(H(M)) and M**
  - On receipt, Alice ***separates*** message from digital signature
  - Alice uses Bob's public key to recover the hash value:

$$PU_{Bob}[PR_{Bob}(H(M))] = H(M)$$

  - Alice computes her own hash value from M and compares with received H(M)

Message Authentication using Digital Signature



source: Fig. 2.3, p. 49 (annotated)

If received and computed MACs match, the recipient is assured

1. the message has not been altered (since the hash values match)
2. the message came from the alleged sender (since only Bob know PR_{Bob})