Introduction to Information Security

Dr. Ashok Kumar Das

Center for Security, Theory and Algorithmic Research
International Institute of Information Technology, Hyderabad

E-mail: *ashok.das@iiit.ac.in*
URL: http://www.iiit.ac.in/people/faculty/ashokkdas
Personal Home Page: http://sites.google.com/view/iitkgpakdas/

Diffie-Hellman Key Exchange Protocol

Overview

- Diffie-Hellman key agreement (also called exponential key exchange or Diffie-Hellman key exchange) provided the first practical solution to the secret key distribution problem.
- It is based on public-key cryptography.
- This protocol enables two parties, say *A* and *B*, which have never communicated before, to establish a mutual secret key by exchanging messages over a public channel.

Inventors

Figure: Whitfield Diffie

Inventors

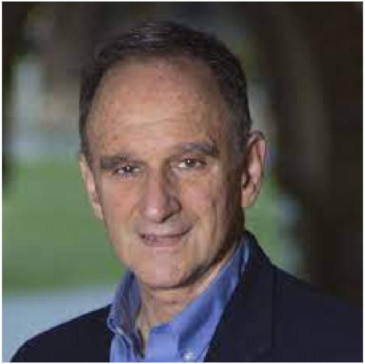

Figure: Martin Hellman

I and Prof. Martin Hellman at IIIT Hyderabad (15 February 2018)

Diffie-Hellman Key Exchange Protocol (continued...)

Global Public Elements

- q : a sufficiently large prime, such that it is intractible to compute the discrete logarithms in $Z_q^* = \{1, 2, \cdots, q-1\}$
(Given α, q and $y = \alpha^x \pmod{q}$, to find discrete logarithm $x \in Z_q^*$).
- α : $\alpha < q$ and α a primitive root of q.
(Compute $\alpha^1 \pmod{q}$, $\alpha^2 \pmod{q}$, $\cdots$, $\alpha^{q-1} \pmod{q}$).
If all are distinct and $\alpha^{q-1} \pmod{q} = 1$, α is primitive root of q)

User A Key Generation

- Select private X_A such that $X_A < q$
- Calculate public Y_A such that $Y_A = \alpha^{X_A} \mod q$

$A \rightarrow B : \{Y_A, q, \alpha\}$
Here $A \rightarrow B : M$ denotes party A sends a message M to party B.

Diffie-Hellman Key Exchange Protocol (continued...)

User B Key Generation

- Select private X_B such that $X_B < q$
- Calculate public Y_B such that $Y_B = \alpha^{X_B} \mod q$

$B \rightarrow A : \{Y_B\}$

Generation of secret key by User A

- $K_{A,B} = (Y_B)^{X_A} \mod q$

Generation of secret key by User B

- $K_{B,A} = (Y_A)^{X_B} \mod q$

Diffie-Hellman Key Exchange Protocol (continued...)

Summary

User A	User B
1. Select private X_A	
2. Calculate public Y_A	
3. $\xrightarrow{Y_A = \alpha^{X_A} \mod q}$	
	1. Select private X_B
	2. Calculate public Y_B
	3. $\xleftarrow{Y_B = \alpha^{X_B} \mod q}$
4. $K_{A,B} = (Y_B)^{X_A} \mod q$	4. $K_{B,A} = (Y_A)^{X_B} \mod q$

Diffie-Hellman Key Exchange Protocol (continued...)

Correctness Proof

$$\begin{aligned}
K_{A,B} &= (Y_B)^{X_A} \mod q \text{ [User A]} \\
&= (\alpha^{X_B} \mod q)^{X_A} \mod q \\
&= (\alpha)^{X_B \cdot X_A} \mod q \\
&= (\alpha^{X_A})^{X_B} \mod q \\
&= (\alpha^{X_A} \mod q)^{X_B} \mod q \\
&= (Y_A)^{X_B} \mod q \\
&= K_{B,A} \text{ [User B]}
\end{aligned}$$

Diffie-Hellman Key Exchange Protocol (continued...)

Problem [Diffie-Hellman Key Exchange]

Users A and B use the Diffie-Hellman key exchange technique with a common prime $q = 71$ and a primitive root $\alpha = 7$.
(a) If user A has private key $X_A = 5$, what is the A's public key Y_A?
(b) If user B has private key $X_B = 12$, what is the B's public key Y_B?
(c) What is the secret shared key?

Solution: Here $q = 71$ and $\alpha = 7$.
(a) A's public key Y_A is given by

$$\begin{aligned} Y_A &= \alpha^{X_A} \mod q \\ &= 7^5 \mod 71 \\ &= (7^1 \mod 71) \times (7^4 \mod 71) \mod 71 \\ &= 51 \end{aligned}$$

Diffie-Hellman Key Exchange Protocol (continued...)

Problem [Diffie-Hellman Key Exchange] (Continued...)

(b) *B*'s public key Y_B is given by

$$\begin{aligned} Y_B &= \alpha^{X_B} \bmod q \\ &= 7^{12} \bmod 71 \\ &= (7^4 \bmod 71) \times (7^8 \bmod 71) \bmod 71 \\ &= 4 \end{aligned}$$

(c) The secret shared key K is given by

$$\begin{aligned} K_{A,B} &= (Y_B)^{X_A} \bmod q \text{ [User A]} \\ &= 4^5 \bmod 71 \\ &= 30 \end{aligned}$$

Diffie-Hellman Key Exchange Protocol (continued...)

Problem [Diffie-Hellman Key Exchange] (Continued...)

$$
\begin{aligned}
K_{B,A} &= (Y_A)^{X_B} \mod q \text{ [User B]} \\
&= 51^{12} \mod 71 \\
&= 30
\end{aligned}
$$

$K = K_{A,B} = K_{B,A} = 30$ is the required secret shared key between A and B. ■

Diffie-Hellman Key Exchange Protocol (continued...)

Online Demo on Diffie-Hellman Key Exchange Protocol

- Generating primitive root of prime
- Computing the shared session key between two parties

`http://www.irongeek.com/diffie-hellman.php?`

Diffie-Hellman Key Exchange Protocol (continued...)

Discrete Logarithm Problem (DLP)

Instance: A multiplicative group $(G, \cdot)$, an element $g \in G$ having order n and $y = g^x \mod n$.
Question: Find x.
This problem is computationally infeasible when n is large.

Diffie-Hellman Key Exchange Protocol (continued...)

Formal definition of discrete logarithm problem

Let G be a cyclic group of order n, g a generator of G, and A_1 an algorithm that returns an integer in Z_n, where $Z_n = \{0, 1, \ldots, n-1\}$. Let $a \in_R S$ denote that a is chosen randomly from the set S. Consider the following experiment, $EXP1^{DLP}_{G,g}(A_1)$ in Algorithm 1.

Algorithm 1: $EXP1^{DLP}_{G,g}(A_1)$

$x \in_R Z_n$
$X \leftarrow g^x \mod n$
$x' \leftarrow A_1(X)$
if $g^{x'} = X \mod n$ **then**
  return 1 (Success)
else
  return 0 (Failure)
end if

Active Attack on Diffie-Hellman Key Exchange: Man-in-the-middle attack

Table: Man-in-the-middle attack

Alice (User A)	Eve (attacker) C	Bob (User B)
1. private: $X_A < q$ public: $Y_A = \alpha^{X_A} \bmod q$ $\xrightarrow{\langle Y_A \rangle}$		
	2. private: $X_C < q$ public: $Y_C = \alpha^{X_C} \bmod q$ $\xleftarrow{\langle Y_C \rangle}$ $\xrightarrow{\langle Y_C \rangle}$	
		3. private: $X_B < q$ public: $Y_B = \alpha^{X_B} \bmod q$ $\xleftarrow{\langle Y_B \rangle}$

Active Attack on Diffie-Hellman Key Exchange: Man-in-the-middle attack (Continued...)

Table: Man-in-the-middle attack (continued...)

Alice (User A)	Eve (attacker) C	Bob (User B)
4. Computes $K_1 = Y_C^{X_A} \bmod q$		
	5. Computes $K_1 = Y_A^{X_C} \bmod q$ $K_2 = Y_B^{X_C} \bmod q$	
		6. Computes $K_2 = Y_C^{X_B} \bmod q$

Alice-Eve key, $K_1 = Y_C^{X_A} \bmod q = Y_A^{X_C} \bmod q = \alpha^{X_A X_C} \bmod q$.

Eve-Bob key, $K_2 = Y_C^{X_B} \bmod q = Y_B^{X_C} \bmod q = \alpha^{X_C X_B} \bmod q$.

Active Attack on Diffie-Hellman Key Exchange: Man-in-the-middle attack (Continued...)

- Alice (User A) chooses $X_A (< q)$, calculates $Y_A = \alpha^{X_A} \bmod q$, and sends Y_A to Bob (User B).
- Eve, the intruder, intercepts Y_A. She chooses $X_C (< q)$, calculates $Y_C = \alpha^{X_C} \bmod q$, and sends Y_C to both Bob and Alice.
- Bob (User B) chooses $X_B (< q)$, calculates $Y_B = \alpha^{X_B} \bmod q$, and sends Y_B to Alice . Y_B is intercepted by Eve and never reaches Alice.
- Alice and Eve calculate $K_1 = Y_C^{X_A} \bmod q = Y_A^{X_C} \bmod q$ $= \alpha^{X_A X_C} \bmod q$, which becomes a shared key between Alice and Eve. Alice, however, thinks that it is a key shared between Bob and Alice.
- Eve and Bob calculate $K_2 = Y_C^{X_B} \bmod q = Y_B^{X_C} \bmod q$ $= \alpha^{X_B X_C} \bmod q$, which becomes a shared key between Bob and Eve. Bob, however, thinks that it is a key shared between Alice and Bob.

Consequences: Active Attack on Diffie-Hellman Key Exchange: Man-in-the-middle attack (Continued...)

- Two keys, instead of one, are created during this attack: one (K_1) between Alice and Eve and other (K_2) between Bob and Eve.
- Suppose Alice wants to send data to Bob.
- Alice encrypts data using the key K_1 and sends to Bob.
- Eve can deciphered the message using the key K_1 and read all the messages.
- Eve can send the message to Bob encrypted using the key K_2
  or
  even change the message
  or
  send a totally new message.
- Bob is fooled into believing that the message has come from Alice.
- Similar situation, when Bob sends messages to Alice.

Defense: Active Attack on Diffie-Hellman Key Exchange: Man-in-the-middle attack (Continued...)

- The station-to-station key agreement method based on the Diffie-Hellman uses authentication to thwart this serious attack.
- This station-to-station key agreement method uses certificates.
- Self study for station-to-station key agreement method.
- Reference: Behrouz A. Forouzan, “Cryptography and Network Security,” Special Indian Edition.

ElGamal Encryption

- In 1985, T. Elgamal announced a public-key scheme based on discrete logarithms, closely related to the Diffie-Hellman technique.
  Reference: T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory, 31:469-472, July 1985.
- As with Diffie-Hellman, the global public elements of the ElGamal scheme are:
  a prime number q and α, a primitive root of q (i.e., α is a primitive root in Z_q^*).
- We start with a very large finite field. We take the field Z_q, with q a large prime.

ElGamal Encryption

- Suppose that the user A (Alice) wants to send some secret messages to the user B (Bob).
- **Key Generation**
  The recipient of message, Bob (user B), proceeds as follows:
  - Step 1. He chooses a large prime q, such that $q - 1$ has a big prime factor and a primitive root $\alpha \in Z_q^*$.
  - Step 2. He chooses an integer $X_B (< q)$ in the range $1 \leq X_B \leq q - 1$ at random.
    X_B is the secret key (private key) of Bob.
  - Step 3. He computes $Y_B = \alpha^{X_B} \pmod{q}$.
    The public key of Bob is (q, α, Y_B), and X_B is kept secret.

ElGamal Encryption

- **Encryption**
  Alice (user A) encrypts a plaintext message $M < q$ intended for user B (bob) as follows:
  - Step 1. Choose a random number X_A such that $1 \leq X_A \leq q - 1$.
  - Step 2. Compute $K = Y_B^{X_A} \pmod{q}$.
  - Step 3. Encrypt M as the pair of integers (C_1, C_2), where $C_1 = \alpha^{X_A} \pmod{q}$, and $C_2 = KM \pmod{q}$.

ElGamal Encryption

- **Decryption**
  Bob (user B) recovers the plaintext message M as follows:
  - Step 1. Compute $K = C_1^{X_B} \pmod{q}$.
  - Step 2. Recover M as $M = C_2 K^{-1} \pmod{q}$.

ElGamal Encryption

Problem: Consider an ElGamal scheme with a common prime number $q = 71$ and a primitive root $\alpha = 7$. If the recipient B has the public key $Y_B = 3$ and the sender A chooses the random integer $X_A = 2$, what is the ciphertext of the plaintext message $M = 30$?

- **Solution:**
  - $K = Y_B^{X_A} \pmod{q} = 3^2 \pmod{71} = 9.$
  - The ciphertext of $M = 30$ is the pair of integers (C_1, C_2), where $C_1 = \alpha^{X_A} \pmod{q}$, and $C_2 = KM \pmod{q}$.
  - $C_1 = 7^2 \pmod{71} = 49.$
  - $C_2 = 9 \times 30 \pmod{71} = 57.$

Digital Signatures

Authentication Functions

One-way hash function

- A cryptographic hash function is an algorithm which accepts a variable length block of data as input and produces a fixed-size bit string, known as cryptographic hash value.
- Hash function can be applied to a large set of inputs which will produce outputs that are evenly distributed, and apparently random.
- Hash function provides data integrity.
- A change to any bit or bits in input data results, with high probability, in a change to the hash value.
- Mathematically, a one-way hash function $h : \{0,1\}^* \rightarrow \{0,1\}^l$ takes an arbitrary-length input $x \in \{0,1\}^*$, and produces a fixed-length (say, l-bits) output $h(x) \in \{0,1\}^l$, called the message digest or hash value.

Authentication Functions

Hash function

The hash function may be the fingerprint of a file, a message, or other data blocks, and has the following attributes.

- h can be applied to a data block of all sizes.
- For any given input x, the message digest $h(x)$ is easy to operate, enabling easy implementation in software and hardware.
- The output length of the message digest $h(x)$ is fixed.
- Deriving the input x from the given hash value $y = h(x)$ and the given hash function $h(\cdot)$ is computationally infeasible. This property is called the *one-way or pre-image resistance* property.
- For any given input x, finding any other input $y \neq x$ so that $h(y) = h(x)$ is computationally infeasible [*weak-collision resistant or second pre-image resistance* property].
- Finding a pair of inputs (x, y), with $x \neq y$, so that $h(x) = h(y)$ is computationally infeasible [*strong-collision resistant* property].

One-way Hash Functions

- **MD family:** Ron Rivest designed *MD*5 digest algorithm with 128 bits digest length in order to replace *MD*4 in 1991. Though *MD*5 algorithm has several vulnerabilities, it remains as a widely used digest algorithm and it is still used in non-cryptographic applications like computing checksum for unintentional data corruption.
- **SHA family:** The secure hash algorithm (SHA-1) was published by the United States National Security Agency (NSA), in the year 1995, by adding error correcting codes to *MD*5 digest algorithm. SHA-1 produces a digest of length 20 bytes or 160 bits.

  Subsequently, in the year 2001, NIST published its successor, known as **SHA-2 digest algorithm**, which has several variants, such as **SHA-224, SHA-256, SHA-384 & SHA-512** with digest lengths of 28 bytes (224 bits), 16 bytes (256 bits), 48 bytes (384 bits) and 32 bytes (512 bits), respectively.

Digital Signatures

Signature Schemes

- A *signature scheme* is a five-tuple $(\mathcal{P}, \mathcal{A}, \mathcal{K}, \mathcal{S}, \mathcal{V})$, where the following conditions are satisfied:
- 1. $\mathcal{P}$ is a finite set of possible messages;
- 2. $\mathcal{A}$ is a finite set of possible signatures;
- 3. $\mathcal{K}$, the key space, is a finite set of possible keys;
- 4. For each $k \in \mathcal{K}$, there is a signing algorithm $sig_k \in \mathcal{S}$ and a corresponding verification algorithm $ver_k \in \mathcal{V}$. Each $sig_k : \mathcal{P} \to \mathcal{A}$ and $ver_k : \mathcal{P} \times \mathcal{A} \to \{true, false\}$ are functions such that the following equation is satisfied for every message $x \in \mathcal{P}$ and for every signature $y \in \mathcal{A}$:
  $ver_k(x, y) =$ true, if $y = sig_k(x)$,
  $ver_k(x, y) =$ false, if $y \neq sig_k(x)$.
- The pair (x, y) with $x \in \mathcal{P}$ and $y \in \mathcal{A}$ is called a *signed message*.

Digital Signatures

The Digital Signature Algorithm (DSA)

- The DSA is based on the difficulty of computing logarithms and is based on schemes originally presented by ElGamal and Schnorr.

Table: Global Public-Key Components

p	prime number where $2^{L-1} < p < 2^{L}$ for $512 \leq L \leq 1024$ and L is a multiple of 64.
q	prime divisor of $(p-1)$, where $2^{159} < q < 2^{160}$; i.e., bit length of 160 bits.
g	$= h^{(p-1)/q} \mod p$, where h is any integer with $1 < h < p-1$ such that $h^{(p-1)/q} \mod p > 1$.

Digital Signatures

The Digital Signature Algorithm (DSA) (Continued...)

Table: User's Private Key

x	random or psuedo-random integer with $0 < x < q$.

Table: User's Public Key

y	$= g^x \mod p$.

Table: User's Per-Message Secret Number

k	random or psuedo-random integer with $0 < k < q$.

Digital Signatures

The Digital Signature Algorithm (DSA) (Continued...)

Table: Signing Phase

$r = (g^k \mod p) \mod q$
$s = [k^{-1}(H(M) + x.r)] \mod q$
Signature = (r, s)
Send $(M, (r, s))$ to reviever.

M : message to be signed

Digital Signatures

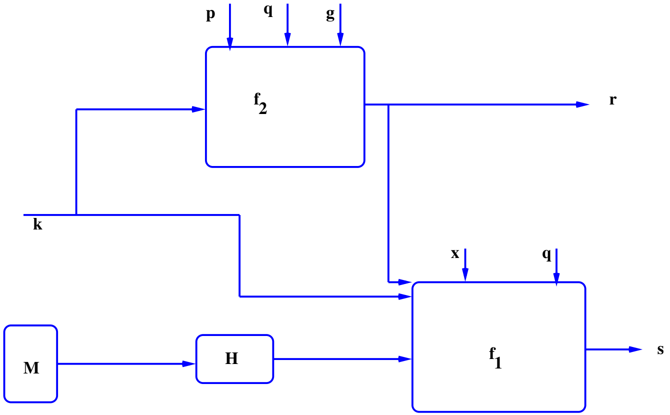

Figure: (a) Signing

$$s = f_1(H(M), k, x, r, q) = [k^{-1}(H(M) + x.r)] \bmod q$$
$$r = f_2(k, p, q, g) = (g^k \bmod p) \bmod q$$

Digital Signatures

The Digital Signature Algorithm (DSA) (Continued...)

Table: Verification Phase

$w = (s')^{-1} \mod q$
$u1 = [H(M').w] \mod q$
$u2 = (r').w \mod q$
$v = (g^{u1}.y^{u2} \mod p) \mod q$
TEST: $v = r'$. If so accept; otherwise reject.

M', r', s' : received versions of M, r, s

Digital Signatures

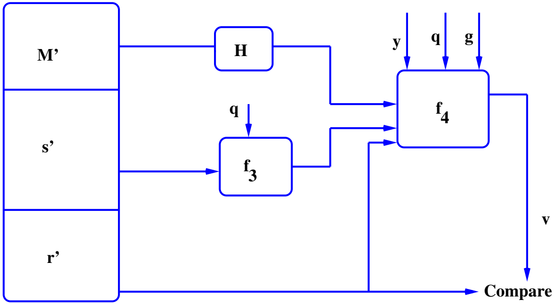

Figure: (b) Verifying

$w = f_3(s', q) = (s')^{-1} \mod q$
$v = f_4(y, q, g, H(M'), w, r')$
$= ((g^{(H(M').w) \mod p}.y^{(r'.w) \mod q}) \mod p) \mod q$

DSA Digital Signature and Verification Demo

Online Demo on DSA

- DSA Key Generation
- Signing File
- Verify Signature

Demo Link:

`https://8gwifi.org/dsafunctions.jsp`

Further Readings (Cryptography and Network Security)

- William Stallings, "Cryptography and Network Security: Principles and Practices", Pearson Education, 2010.
- Behrouz A. Forouzan, "Cryptography and Network Security", Special Indian Edition.
- Bernard Menezes, "Network Security and Cryptography", Cengage Learning, 2010.
- A. Menezes, P. Oorschot and S. Vanstone, "Handbook of Applied Cryptography", CRC Press.
- B. Schneier, "Applied Cryptography", Reading, MA: Addison-Wesley, 2006.
- D. Stinson, "Cryptography: Theory and Practice", Chapman & Hall/CRC, 2006.
- Neal Koblitz, "A course in number theory and cryptography", Springer.

Thank you