قســـم الامـــــن الـــــــســيبرانــــــــي

DEPARTMENT OF CYBER SECURITY

SUBJECT:

PUBLIC KEY ENCRYPTION

CLASS:

THIRD

LECTURER:

ASST. LECTURER QUSAI AL-DURRAH

LECTURE (3 & 4):

DIFFIE–HELLMAN KEY EXCHANGE AND ASYMMETRIC CRYPTOSYSTEMS

1. Introduction

This lecture introduces one of the most influential concepts in modern cryptography—the **Diffie–Hellman Key Exchange**—and its role within **asymmetric (public-key) cryptosystems**. Public-key cryptography, made publicly known in 1976 by **Whitfield Diffie** and **Martin Hellman**, revolutionized secure communication by allowing two parties to establish a shared secret over an insecure channel without ever meeting or sharing a pre-existing key.

This lecture explains the historical context of this innovation, the mathematical foundations of exponential key exchange, and how the Diffie–Hellman protocol allows users to create a **common secret key** using modular arithmetic. It also discusses other public-key algorithms (RSA, ElGamal, Schnorr, ECC, and DSA) and their respective uses in encryption and digital signatures.

2. Learning Outcomes

By the end of this lecture, students will be able to:

1. **Explain** the historical development of public-key cryptography and identify key contributors such as Diffie and Hellman.
2. **Describe** the concept of the Diffie–Hellman key exchange and the mathematical principles underlying modular exponentiation and discrete logarithms.
3. **Illustrate** the step-by-step process of generating a shared key using the Diffie–Hellman scheme.
4. **Differentiate** between key exchange algorithms and encryption or signature algorithms such as RSA, ElGamal, and ECC.
5. **Apply** numerical examples to demonstrate key generation and verify the shared secret.
6. **Evaluate** the strengths and limitations of Diffie–Hellman in terms of computational complexity and real-world security.

3. Overview of Asymmetric Public-Key Cryptosystems

Public-key cryptography became widely recognized after Diffie and Hellman proposed their **exponential key exchange** method in 1976. Since then, many public-key algorithms have been developed, though only a few remain both **secure** and **practical** for real-world use.

3.1 Classification of Public-Key Algorithms

- **RSA (1978):** Supports both encryption and digital signatures.
- **ElGamal (1985):** Based on the discrete logarithm problem; suitable for encryption and signatures.
- **Schnorr (1990):** Used primarily for efficient digital signatures.
- **Elliptic Curve Cryptography (ECC, 1985):** Offers equivalent security with smaller key sizes.
- **Digital Signature Algorithm (DSA, 1991):** Designed exclusively for digital signatures.

The effectiveness of any public-key scheme depends on:

- The **key length** (bit size).
- The **difficulty** of solving the underlying mathematical problem (e.g., factoring or discrete logarithm).

4. The Diffie–Hellman Exponential Key Exchange

The Diffie–Hellman (DH) scheme provides a secure way for two parties to establish a **shared secret key** through public communication channels. The protocol relies on the **mathematical asymmetry** between **easy exponentiation** and **hard discrete logarithms**.

4.1 Mathematical Foundation

Let (p) be a prime number and (α) a **primitive element** (generator) of the multiplicative group GF(p). Where A **Galois Field**, abbreviated as **GF**, is a *finite mathematical field* that contains a limited number of elements. All arithmetic operations are performed **modulo** a fixed number, and the results always remain within the field.

For any integer (Y) and base (α), there exists a unique exponent (X) such that:

$Y \equiv \alpha^X \pmod p$ where $2 \leq X \leq p - 1$

Here, (X) is called the **discrete logarithm** of (Y) to the base (α).

- Computing ($Y = \alpha^X \bmod p$) is easy (exponentiation).
- Computing (X) from (Y) is extremely hard (discrete logarithm problem).

This **one-way property** forms the core of Diffie–Hellman security.

5. Protocol Steps for Key Exchange

Assume two users, **A** and **B**, want to establish a shared key.

1. **Public parameters**:
   Both agree on a prime (p) and a primitive root (α).
   These values are public.
2. **Private keys**:
   - A selects a private number (X_A).
   - B selects a private number (X_B).
     Both keep these numbers secret.
3. **Public keys**:
   - A computes ($Y_A = \alpha^{X_A} \bmod p$).
   - B computes ($Y_B = \alpha^{X_B} \bmod p$).
     Each sends their public key to the other.
4. **Shared secret computation**:
   - A computes ($KAB = (Y_B)^{X_A} \bmod p$).
   - B computes ($KBA = (Y_A)^{X_B} \bmod p$).

Because $Y_B = \alpha^{X_B}$ and $Y_A = \alpha^{X_A}$, both calculations yield:

$K=\alpha^{X_A X_B} \bmod p$

Thus, both A and B obtain the same **shared secret key**, which can later be used for symmetric encryption.

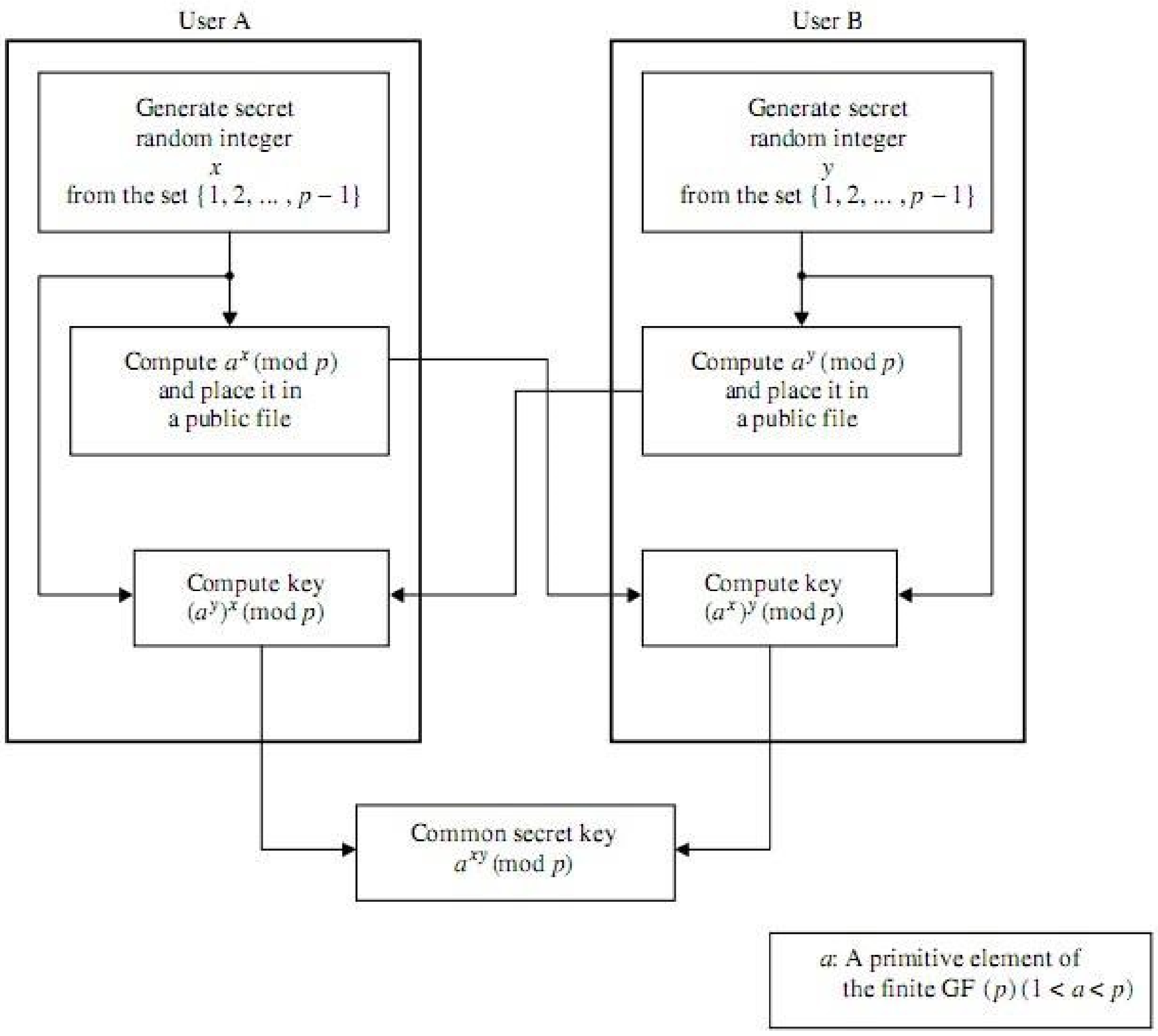
User A
User B
Generate secret random integer x from the set {1, 2, ... , p – 1}
Generate secret random integer y from the set {1, 2, ... , p – 1}
Compute a^x (mod p) and place it in a public file
Compute a^y (mod p) and place it in a public file
Compute key (a^y)^x (mod p)
Compute key (a^x)^y (mod p)
Common secret key a^xy (mod p)
a: A primitive element of the finite GF (p) (1 < a < p)

Figure 1: *The Diffie–Hellman Exponential Key Exchange Scheme*

This figure illustrates the sequence of operations in the Diffie–Hellman protocol. Each user selects a private random integer, computes a public value using modular exponentiation, exchanges public values, and derives the same shared secret key $\alpha^{X_A Y_B} \bmod p$.

6. Calculation Example and code

Given:

- Prime modulus (p = 11)
- Primitive root (α = 2)

Step 1:

User A chooses a private key ($X_A = 5$).

User B chooses ($X_B = 7$).

Step 2: Compute public keys

$Y_A=2^5 \bmod 11 = 32 \bmod 11 = 10$

$Y_B = 2^7 \bmod 11 = 128 \bmod 11 = 7$

Step 3: Compute shared key

$K_{AB}=Y_B^{X_A} \bmod 11=7^5 \bmod 11 = 10$

$K_{BA}=Y_A^{X_B} \bmod 11 =10^7 \bmod 11=10$

Both users derive the same shared secret: **K = 10**

Code:

```
import random
# Step 1: Publicly agreed parameters (prime p and primitive root α)
p = 23          # A prime number
alpha = 5        # A primitive root modulo p
print("Publicly shared values:")
print("Prime (p):", p)
print("Primitive root (α):", alpha)
print("-" * 40)

# Step 2: Each user chooses a private key
Xa = random.randint(2, p-1)   # Private key for User A
Xb = random.randint(2, p-1)   # Private key for User B
print("Private keys:")
print("User A private key (Xa):", Xa)
print("User B private key (Xb):", Xb)
print("-" * 40)
```

```
# Step 3: Each user computes their public key
Ya = pow(alpha, Xa, p)   # A's public key = α^Xa mod p
Yb = pow(alpha, Xb, p)   # B's public key = α^Xb mod p
print("Public keys:")
print("User A public key (Ya):", Ya)
print("User B public key (Yb):", Yb)
print("-" * 40)

# Step 4: Exchange public keys and compute shared secret
Ka = pow(Yb, Xa, p)      # A computes shared key
Kb = pow(Ya, Xb, p)      # B computes shared key
print("Shared secret keys:")
print("Key computed by A:", Ka)
print("Key computed by B:", Kb)
print("-" * 40)

# Verify both keys are identical
if Ka == Kb:
    print(" Success: Common secret key established:", Ka)
else:
    print(" Error: Keys do not match!")
```

Explanation

Step	**Operation**	**Description**
1	p, alpha	Public parameters agreed upon by both users
2	Xa, Xb	Random private keys selected secretly
3	Ya, Yb	Public keys calculated using modular exponentiation
4	Ka, Kb	Shared secret key derived independently by both users

Output:

```
Publicly shared values:
Prime (p): 23
Primitive root (α): 5
----------------------------------------
Private keys:
User A private key (Xa): 6
User B private key (Xb): 15
----------------------------------------
Public keys:
User A public key (Ya): 8
User B public key (Yb): 19
----------------------------------------
Shared secret keys:
Key computed by A: 2
Key computed by B: 2
----------------------------------------
Success: Common secret key established: 2
```

7. Security Considerations

The security of Diffie–Hellman depends on:

- The **size of the prime number (p)**: larger primes make discrete logarithms computationally infeasible.
- The **choice of primitive root (α)**: ensures that generated numbers span the full multiplicative group.
- Avoiding small subgroup attacks and man-in-the-middle attacks by incorporating authentication mechanisms (e.g., signed public keys).

Although Diffie–Hellman is secure in theory, without authentication it is vulnerable to interception during key exchange.

8. Variants and Applications

- **Ephemeral Diffie–Hellman (DHE):** Uses temporary keys for each session, providing *forward secrecy*.
- **Elliptic Curve Diffie–Hellman (ECDH):** Implements DH on elliptic curves, offering equal security with shorter keys.
- **TLS/SSL:** Modern web browsers use Diffie–Hellman variants to secure HTTPS sessions.
- **Virtual Private Networks (VPNs):** Use DH or ECDH to establish session keys for encrypted tunnels.

9. Advantages and Limitations

Advantages	**Limitations**
Eliminates need for prior key sharing	Susceptible to man-in-the-middle if unauthenticated
Based on mathematically hard discrete logarithm problem	Computationally expensive for very large primes
Enables secure communication over open channels	Requires additional mechanisms for authentication

10. Conclusion

The **Diffie–Hellman Key Exchange** remains one of the foundational pillars of modern cybersecurity.
Its elegant mathematical foundation enables secure establishment of shared keys without prior trust, forming the basis for many modern protocols such as **TLS**, **SSH**, and **IPSec**. Despite its vulnerabilities when used alone, its combination with digital signatures and modern key agreement extensions ensures both **confidentiality** and **authenticity** in secure communication systems.

References

- Diffie, W., & Hellman, M. (1976). *New Directions in Cryptography.* IEEE Transactions on Information Theory.
- Stallings, W. (2017). *Cryptography and Network Security: Principles and Practice.* 7th Edition, Pearson.

Homework Assignment 3:

Using your knowledge from The Lecture, explain in your own words **how two users (A and B)** establish a shared secret key using the **Diffie–Hellman key exchange algorithm**.

Your explanation should include:

1. The meaning of the public parameters *(p)* and *(α)*.
2. How users select private and public keys.
3. How both users independently compute the same shared secret key.
4. One short **numerical example** to support your answer (you may use the values $p = 11$, $\alpha = 2$).
5. A short note on **one security risk** (e.g., man-in-the-middle attack) and **how it can be mitigated**.

Format Requirements:

- Length: **250–300 words**
- Language: **English or Arabic**
- File type: **Word** or **PDF**
- Submission: **Google Classroom → Homework Assignment 3**
- **Deadline:** One week after the lecture