사이버보안개론

Introduction to Cybersecurity

유준수 (Joon Soo Yoo, Ph.D.)

Fall 2026

Defense & Advanced Security Lab (DAS Lab)
Department of Advanced Defense Technology and Industry
Jeonbuk National University

Chapter 2. Cryptographic Tools

2.1 Symmetric Encryption

Confidentiality with a Shared Secret Key

Learning Objectives

이번 Section에서는 Symmetric Encryption의 기본 구조와 주요 개념을 학습한다.

Learning Goals

- Symmetric Encryption의 기본 구조를 이해한다.
- Plaintext, Ciphertext, Secret Key의 역할을 이해한다.
- 안전한 Symmetric Encryption의 조건을 이해한다.
- Secret Key Distribution이 왜 중요한지 설명할 수 있다.
- Cryptanalysis와 Brute-Force Attack을 구분할 수 있다.
- Block Cipher와 Stream Cipher의 차이를 이해한다.
- DES, 3DES, AES의 발전 흐름을 이해한다.
- ECB Mode의 Pattern Leakage 문제를 이해한다.

1. Symmetric Encryption

Encryption의 가장 기본적인 목적은 Data의 **Confidentiality**를 보호하는 것이다.

읽을 수 있는 Plaintext를 Unauthorized Entity가 이해하기 어려운 Ciphertext로 변환한다.

Plaintext

⇓

Encryption

⇓

Ciphertext

Symmetric Encryption에서는 Sender와 Receiver가 **동일한 Secret Key**를 사용한다.

$$Y = E_K(X)$$

$$X = D_K(Y)$$

핵심

Symmetric Encryption은 동일한 Secret Key를 이용하여 Encryption과 Decryption을 수행한다.

1-1. Components of Symmetric Encryption

Symmetric Encryption에는 다섯 가지 기본 요소가 있다.

Basic Components

Component	Meaning
Plaintext	암호화하기 전의 원래 Data
Encryption Algorithm	Plaintext를 Ciphertext로 변환하는 Cipher
Secret Key	Encryption과 Decryption에 사용되는 비밀값
Ciphertext	Encryption의 결과로 만들어진 암호화된 Data
Decryption Algorithm	Ciphertext를 다시 Plaintext로 복원

Encryption Algorithm의 동작은 Secret Key에 따라 달라진다.

핵심

Symmetric Encryption의 Security는 Cipher와 Secret Key 모두에 의존한다.

1-2. Requirements for Secure Symmetric Encryption

Symmetric Encryption을 안전하게 사용하려면 두 가지 조건이 중요하다.

1. **Strong Encryption Algorithm**
   공격자가 Algorithm을 알고 있고 Ciphertext를 가지고 있어도,
   Plaintext를 복구하거나 Secret Key를 알아내는 것이 현실적으로 어려워야 한다.

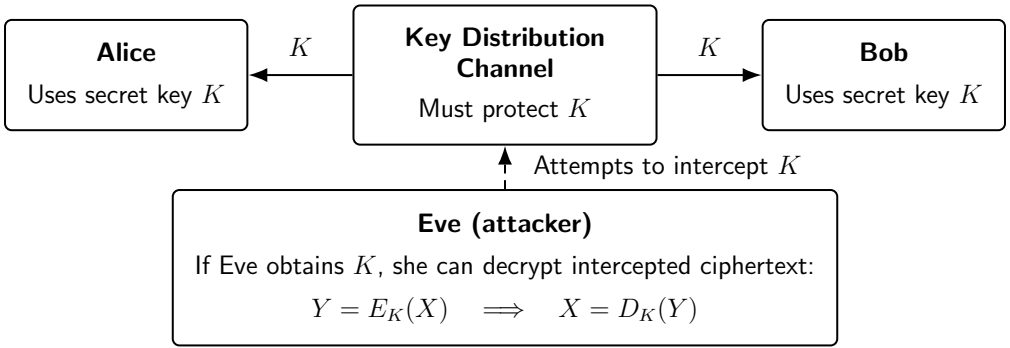
2. **Secure Secret Key**
   Sender와 Receiver가 Secret Key를 안전하게 공유하고 보호해야 한다.

중요

Secret Key가 유출되면 강한 Encryption Algorithm을 사용하더라도 해당 Key로 암호화된 Data는 보호할 수 없다.

How do Alice and Bob securely obtain K?

The same secret key must be shared before encrypted communication.



Secure key distribution is itself a security problem.

1-3. The Key Distribution Problem

Symmetric Encryption에서는 Alice와 Bob이 같은 Secret Key K를 가져야 한다.

따라서 Encryption을 시작하기 전에 먼저 해결해야 하는 문제가 있다.

How do Alice and Bob securely share K?

Secret Key를 안전하지 않은 Network를 통해 그대로 전송한다면,

공격자가 통신을 도청하여 Secret Key를 획득할 수 있다.

핵심

Symmetric Encryption에서는 Data를 안전하게 암호화하는 것뿐 아니라 Secret Key를 안전하게 공유하는 것도 중요하다.

2. Attacking Symmetric Encryption

Symmetric Encryption을 공격하는 방법은 크게 두 가지 관점에서 생각할 수 있다.

Cryptanalysis

Cipher의 구조와 특성을 분석한다.

공격자가 사용할 수 있는 정보:

- Ciphertext
- Plaintext의 일반적인 특성
- Plaintext–Ciphertext Pair

Brute-Force Attack

Cipher의 구조적 특성을 이용하지 않고 가능한 Secret Key를 직접 시도한다.

$$K_1, K_2, K_3, \ldots$$

Key Space를 탐색하여 올바른 Key를 찾는다.

핵심

Cryptanalysis는 Cipher의 특성을 이용하고, Brute Force는 Key Space 자체를 탐색한다.

2-1. Cryptanalysis

Cryptanalysis는 Cipher의 구조와 특성을 분석하여 Encryption을 공격하는 방법이다.

공격자는 다음과 같은 정보를 활용할 수 있다.

- Encryption Algorithm의 구조
- Ciphertext의 특징
- Plaintext의 일반적인 특징
- 알려진 Plaintext–Ciphertext Pair

Cryptanalysis의 주요 목표는 다음과 같다.

- 특정 Plaintext를 Recover
- 사용된 Secret Key를 Recover

Secret Key를 알아내면 해당 Key로 암호화된 다른 Message도 복호화할 수 있다.

핵심

Cryptanalysis는 가능한 모든 Key를 단순히 시도하는 것이 아니라, Cipher의 특성을 이용하여 공격을 더 효율적으로 수행하려는 접근이다.

2-2. Brute-Force Attack

Brute-Force Attack은 가능한 Secret Key를 하나씩 직접 시도하는 공격이다.

Key Length가 k bit라면 가능한 Secret Key의 수는

$$2^k$$

이다.

평균적으로 올바른 Key를 찾는 데 필요한 시도 횟수는 대략

$$2^{k-1}$$

이다.

예를 들어,

$$56\text{-bit Key} \rightarrow 2^{56}$$

$$128\text{-bit Key} \rightarrow 2^{128}$$

개의 가능한 Key가 존재한다.

핵심

Key Length가 증가하면 Brute-Force Attack이 탐색해야 하는 Key Space는 지수적으로 증가한다.

3. Symmetric Block Encryption

가장 널리 사용되는 Symmetric Encryption Algorithm의 형태는 **Block Cipher**이다.

Block Cipher는 Plaintext를 고정된 크기의 Block 단위로 처리한다.

긴 Plaintext는

$$P_1, P_2, P_3, \ldots, P_n$$

과 같은 Fixed-Size Block으로 나뉜다.

각 Plaintext Block은 Secret Key K를 이용하여

$$C_i = E_K(P_i)$$

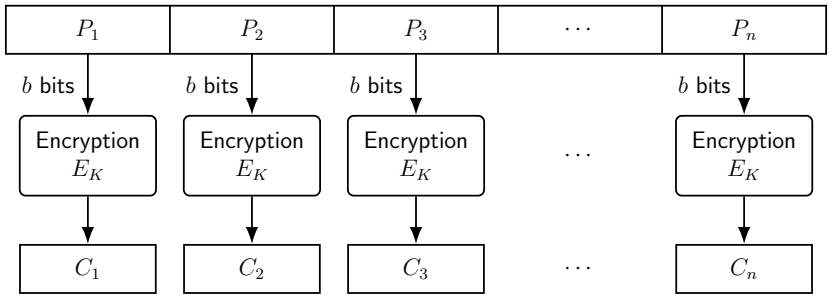
와 같이 Ciphertext Block으로 변환된다.

핵심

Block Cipher는 Data를 일정한 크기의 Block으로 나누어 각 Block을 암호화한다.

Block Cipher: Basic Concept

Plaintext stream partitioned into b-bit blocks



Every E_K uses the same secret key K

Fixed-size blocks $P_i, C_i \in \{0,1\}^b$

$$C_i = E_K(P_i)$$

Single-block transformation shown. A mode of operation specifies how the block cipher is used to encrypt a complete message.

3-1. DES, 3DES, and AES

대표적인 Symmetric Block Cipher에는 DES, Triple DES, AES가 있다.

Symmetric Block Ciphers

Algorithm	Block Size	Key Size
DES	64 bits	56 bits
Triple DES	64 bits	112 or 168 bits
AES	128 bits	128, 192, or 256 bits

$$\text{DES} \longrightarrow \text{3DES} \longrightarrow \text{AES}$$

각 Algorithm은 Security와 Efficiency 측면에서 서로 다른 특징을 가진다.

핵심

DES의 한계를 보완하기 위해 3DES가 사용되었고, 이후 AES가 현대적인 Symmetric Encryption Standard로 자리 잡았다.

3-2. Data Encryption Standard (DES)

DES는 1977년 표준화된 대표적인 초기 Symmetric Block Cipher이다.

- Block Size: 64 bits
- Key Size: 56 bits
- 오랫동안 널리 사용됨
- 많은 Cryptanalysis 연구의 대상이 됨

DES의 가장 큰 실질적인 한계는 **56-bit Key Length**이다.

가능한 Key의 수는

$$2^{56}$$

이므로 Computing Power가 증가하면서 Brute-Force Attack에 충분히 안전하지 않게 되었다.

핵심

DES의 대표적인 Security Problem은 짧은 Key Length로 인한 Brute-Force Attack 가능성이다.

3-3. Triple DES

DES의 Key Length 문제를 보완하기 위해 DES Operation을 세 번 적용하는 Triple DES가 사용되었다.

Plaintext
⇓
DES
⇓
DES
⇓
DES
⇓
Ciphertext

Triple DES는 두 개 또는 세 개의 Key를 이용하여

- 112-bit
- 168-bit

수준의 Key를 사용할 수 있다.

하지만 DES Operation을 세 번 수행하기 때문에 상대적으로 느리다는 문제가 있다.

핵심

3DES는 DES의 짧은 Key Length 문제를 보완하지만, Efficiency 측면에서 한계를 가진다.

3-4. Advanced Encryption Standard (AES)

DES와 3DES를 대체하기 위해 NIST는 새로운 Encryption Standard를 선정하였다.

최종적으로 Rijndael Algorithm이 선택되어 AES가 되었다.

AES의 주요 특징은 다음과 같다.

- Symmetric Block Cipher
- Block Size: 128 bits
- Key Size: 128, 192, 256 bits
- Hardware와 Software에서 효율적인 구현 가능

AES-128 AES-192 AES-256

핵심

AES는 DES와 3DES를 대체한 대표적인 현대 Symmetric Block Cipher이다.

4. Encrypting Multiple Blocks

실제 Message나 File은 하나의 Block보다 훨씬 크다.

따라서 실제 Data를 암호화하려면 긴 Plaintext를 여러 Block으로 나누어야 한다.

$$P_1, P_2, \ldots, P_n$$

그러면 새로운 질문이 생긴다.

How should we encrypt multiple blocks?

각 Block을 어떻게 연결하여 암호화하는지에 따라 Security 특성이 달라질 수 있다.

핵심

Block Cipher 자체가 안전하더라도 여러 Block에 Cipher를 적용하는 방식이 안전하지 않을 수 있다.

4-1. Electronic Codebook (ECB)

여러 Block을 암호화하는 가장 단순한 방법 중 하나가 Electronic Codebook Mode이다.

ECB에서는 각 Plaintext Block을 같은 Secret Key를 이용하여 독립적으로 암호화한다.

$$C_i = E_K(P_i)$$

예를 들어,

$$P_1 \rightarrow C_1$$

$$P_2 \rightarrow C_2$$

$$P_3 \rightarrow C_3$$

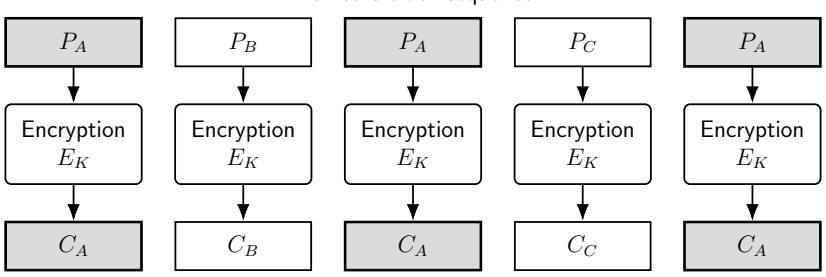
와 같이 각 Block이 서로 독립적으로 처리된다.

핵심

ECB는 구조가 단순하지만, 각 Block을 독립적으로 암호화한다는 특징을 가진다.

ECB: Repeated Blocks Reveal Patterns

Each block is encrypted independently with the same secret key K.



Ciphertext block sequence: the repeated positions remain visible

$$P_i = P_j \quad \Longrightarrow \quad E_K(P_i) = E_K(P_j)$$

Pattern leakage: ciphertext reveals equality of plaintext blocks.

4-2. The ECB Pattern Problem

ECB에서는 같은 Key로 동일한 Plaintext Block을 암호화하면 동일한 Ciphertext Block이 만들어진다.

$$P_i = P_j$$

이면,

$$E_K(P_i) = E_K(P_j)$$

이다.

따라서 공격자가 Plaintext의 실제 내용은 알지 못하더라도

- 어떤 Block이 반복되는지
- Data에 어떤 Pattern이 존재하는지

를 관찰할 수 있다.

중요

Strong Cipher를 사용하더라도 Cipher를 사용하는 방식이 잘못되면 정보가 노출될 수 있다.

4-3. Block Cipher Modes of Operation

ECB의 문제를 해결하기 위해 Block Cipher를 여러 Block에 적용하는 다양한 Mode of Operation이 개발되었다.

핵심 목표는 Plaintext의 반복적인 Pattern이 Ciphertext에 그대로 나타나지 않도록 하는 것이다.

즉,

$$P_i = P_j$$

라고 하더라도 항상

$$C_i = C_j$$

가 나타나지 않도록 Block 사이의 관계나 추가적인 값을 이용한다.

핵심

안전한 Encryption을 위해서는 좋은 Cipher를 선택하는 것뿐 아니라 적절한 Mode of Operation을 사용하는 것도 중요하다.

5. Stream Ciphers

Stream Cipher는 Block Cipher와 달리 Data를 연속적인 Stream 형태로 처리한다.

Secret Key K를 이용하여 Pseudorandom Keystream을 생성한다.

Secret Key K

$\Downarrow$

Keystream Generator

$\Downarrow$

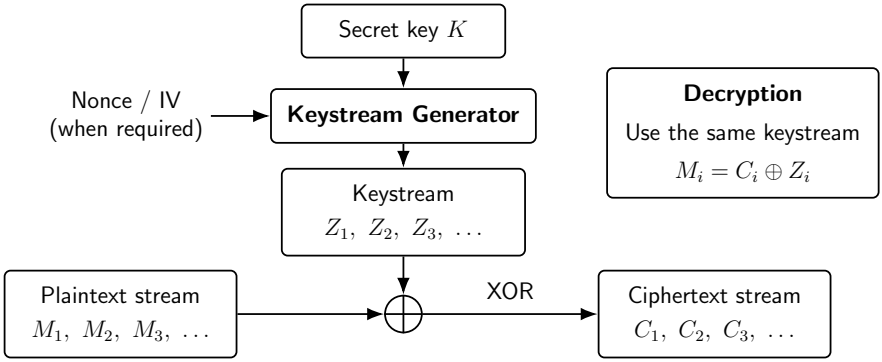
Keystream Z

생성된 Keystream을 Plaintext Stream과 결합하여 Ciphertext Stream을 만든다.

핵심

Stream Cipher는 Secret Key로부터 생성된 Keystream을 이용하여 Data를 연속적으로 암호화한다.

Stream Cipher: Encryption with a Keystream



$$C_i = M_i \oplus Z_i$$

Process successive bits or bytes as the stream advances.
$\oplus$ denotes bitwise XOR.

5-1. Encryption and Decryption with XOR

대표적인 Stream Cipher 구조에서는 Plaintext와 Keystream을 XOR하여 암호화한다.

Encryption:

$$C = M \oplus Z$$

Receiver가 동일한 Keystream Z를 생성할 수 있다면

$$M = C \oplus Z$$

로 Decryption할 수 있다.

이는 XOR의 다음 성질 때문이다.

$$A \oplus B \oplus B = A$$

실제로,

$$C \oplus Z = (M \oplus Z) \oplus Z = M$$

이다.

핵심

Sender와 Receiver가 동일한 Keystream을 생성할 수 있다면 XOR를 이용하여 Encryption과 Decryption을 수행할 수 있다.

5-2. Block Cipher vs. Stream Cipher

Block Cipher vs. Stream Cipher

	Block Cipher	Stream Cipher
Processing Unit	Fixed-Size Block	Bit / Byte Stream
Basic Operation	Block 단위로 Cipher 적용	Keystream과 Plaintext 결합
Output	Ciphertext Blocks	Ciphertext Stream
Typical Structure	$C_i = E_K(P_i)$	$C_i = M_i \oplus Z_i$
Examples	AES, DES, 3DES	Stream Encryption Algorithms

두 방식 모두 Secret Key를 사용하는 Symmetric Encryption이다.

차이는 Data를 처리하는 기본 단위와 Encryption 구조에 있다.

핵심

Block Cipher와 Stream Cipher는 모두 Symmetric Encryption이지만, Data를 처리하는 구조가 다르다.

6. What We Learned

이번 Section에서는 Symmetric Encryption의 기본 구조를 살펴보았다.

- Symmetric Encryption은 Sender와 Receiver가 동일한 Secret Key를 사용한다.
- 안전한 Symmetric Encryption을 위해서는 Strong Encryption Algorithm과 Secure Secret Key가 필요하다.
- Secret Key를 안전하게 공유하는 Key Distribution Problem이 존재한다.
- Cryptanalysis는 Cipher의 특성을 이용하고, Brute-Force Attack은 Key Space를 탐색한다.
- Block Cipher는 Data를 Fixed-Size Block 단위로 처리한다.
- DES와 3DES를 거쳐 AES가 대표적인 현대 Symmetric Block Cipher가 되었다.
- ECB와 같이 Cipher를 사용하는 방식이 잘못되면 Plaintext의 Pattern이 노출될 수 있다.
- Stream Cipher는 Keystream을 생성하여 Plaintext Stream과 결합한다.

핵심

Symmetric Encryption의 Security는 강한 Cipher뿐 아니라 Secret Key의 관리와 Cipher를 사용하는 방식에도 의존한다.

Chapter 2. Cryptographic Tools

2.2 Message Authentication and Hash Functions

Integrity and Authentication

Learning Objectives

이번 Section에서는 Message Authentication과 Cryptographic Hash Function의 기본 개념을 학습한다.

Learning Goals

- Confidentiality와 Message Authentication의 차이를 이해한다.
- Message Authentication Code (MAC)의 역할을 이해한다.
- Hash Function과 MAC의 차이를 이해한다.
- Secure Hash Function의 주요 Security Property를 이해한다.
- Preimage, Second Preimage, Collision Resistance를 구분할 수 있다.
- Birthday Paradox와 Collision Security의 관계를 이해한다.
- SHA 계열의 기본적인 발전 흐름을 이해한다.

1. Why Message Authentication?

앞에서는 Encryption을 이용하여 Data의 Confidentiality를 보호하였다.

Attacker reads the message $\longrightarrow$ Encryption

하지만 공격자가 Message를 읽는 것뿐 아니라 변경할 수도 있다.

Original Message

$\Downarrow$

Attacker modifies the message

$\Downarrow$

Modified Message

따라서 다음과 같은 질문도 필요하다.

- Message가 중간에 변경되지 않았는가?
- Message가 실제 Sender로부터 왔는가?

핵심

Encryption은 주로 Confidentiality를 제공하고, Message Authentication은 Integrity와 Source Authentication을 확인한다.

1-1. What Should Be Verified?

Message Authentication에서는 Received Message가 Authentic한지 확인한다.

주요 확인 대상은 다음과 같다.

- **Integrity**
  Message의 내용이 변경되지 않았는가?
- **Source Authentication**
  Message가 주장된 Sender로부터 왔는가?
- **Timeliness**
  오래된 Message가 지연되거나 Replay된 것은 아닌가?
- **Sequence**
  Message의 순서가 변경되지 않았는가?

핵심

Message Authentication은 Message의 내용뿐 아니라 Source, Time, Sequence와 관련된 문제도 다룰 수 있다.

1-2. Is Encryption Alone Enough?

Encryption된 Message가 정상적으로 Decryption된다고 해서 Message가 변조되지 않았다고 보장할 수는 없다.

ECB Mode를 생각해보자.

$$C_1 = E_K(P_1), \qquad C_2 = E_K(P_2), \qquad C_3 = E_K(P_3)$$

공격자가 Ciphertext Block의 순서를 바꾸면

$$C_1, C_2, C_3 \quad \longrightarrow \quad C_3, C_1, C_2$$

Receiver는 여전히 각 Block을 정상적으로 Decryption할 수 있다.

$$P_3, P_1, P_2$$

하지만 전체 Message의 의미는 변경될 수 있다.

중요

Successful Decryption이 Message Integrity를 보장하는 것은 아니다.

2. Message Authentication Code (MAC)

Message Authentication을 위한 대표적인 방법이 Message Authentication Code (MAC)이다.

Sender와 Receiver가 Secret Key K를 공유한다고 하자.

Sender는 Message M에 대해

$$T = MAC_K(M)$$

을 계산한다.

그리고 다음을 전송한다.

$$M \parallel T$$

여기서 T는 Message에 대한 Authentication Tag이다.

핵심

MAC은 Message와 Secret Key를 이용하여 Message를 검증하기 위한 Authentication Tag를 생성한다.

2-1. Verifying a MAC

Receiver도 동일한 Secret Key K를 가지고 있다.

Received Message M에 대해

$$T' = MAC_K(M)$$

을 다시 계산한다.

그리고 Received Tag T와 비교한다.

$$T' \stackrel{?}{=} T$$

$$\text{Match} \longrightarrow \text{Accept}$$

$$\text{Not Match} \longrightarrow \text{Reject}$$

공격자가 Message를 M'으로 변경하면 올바른 Secret Key 없이

$$MAC_K(M')$$

을 생성할 수 없어야 한다.

핵심

MAC Verification을 통해 Message Integrity와 Source Authentication을 확인할 수 있다.

2-2. Authentication vs. Confidentiality

MAC 자체는 Message를 암호화하지 않는다.

$$M \parallel MAC_K(M)$$

을 전송하면 공격자는 M을 읽을 수 있다.

Different Security Goals

	Confidentiality	Integrity / Authentication
Encryption	Yes	Not necessarily
MAC	No	Yes

Confidentiality와 Authentication이 모두 필요하다면 Encryption과 Authentication을 함께 사용할 수 있다.

핵심

Encryption과 Message Authentication은 서로 다른 Security Goal을 제공한다.

3. What is a Hash Function?

Hash Function은 Variable-Length Message를 입력받아 Fixed-Length Hash Value를 생성한다.

$$h = H(M)$$

Message M

$\Downarrow$

Hash Function H

$\Downarrow$

Hash Value h

Hash Value는 다음과 같이 부르기도 한다.

- Hash
- Message Digest
- Fingerprint

핵심

Hash Function은 크기가 다양한 Data를 고정된 크기의 Hash Value로 변환한다.

3-1. Hash Function vs. MAC

Hash Function과 MAC의 중요한 차이는 Secret Key의 사용 여부이다.

Hash:

$$h = H(M)$$

MAC:

$$T = MAC_K(M)$$

Hash vs. MAC

	Hash	MAC
Input	Message	Message + Secret Key
Secret Key	No	Yes
Fixed-Size Output	Yes	Yes
Authentication	By itself, No	Yes

핵심

Hash Function 자체에는 Secret Key가 없기 때문에 Hash만으로 Sender를 Authentication할 수는 없다.

3-2. Why Is Hash Alone Not Authentication?

Sender가 다음을 전송한다고 생각해보자.

$$M \parallel H(M)$$

공격자가 Message를

$$M \rightarrow M'$$

으로 변경할 수 있다.

Hash Function은 공개되어 있으므로 공격자도 새로운 Hash를 계산할 수 있다.

$$H(M')$$

그리고

$$M' \parallel H(M')$$

을 Receiver에게 전송할 수 있다.

Receiver는 Hash만으로 이러한 공격을 구분할 수 없다.

중요

Hash Function은 Message의 Fingerprint를 만들지만, Hash만으로는 Message Authentication을 제공하지 않는다.

3-3. Authentication Using a Hash Function

Hash Function에 Secret Key를 결합하면 Message Authentication에 사용할 수 있다.

교재의 기본적인 예를 생각하면,

$$T = H(K \parallel M \parallel K)$$

와 같은 형태를 사용할 수 있다.

Sender와 Receiver만 Secret Key K를 알고 있으므로 공격자는 Message를 변경한 후 올바른 Tag를 생성하기 어려워야 한다.

실제 표준적인 방법으로는 HMAC이 널리 사용된다.

핵심

Secret Key와 Hash Function을 결합하여 Message Authentication Code를 구성할 수 있다.

4. Secure Hash Functions

Cryptographic Hash Function은 단순히 Fixed-Length Output을 만드는 것만으로 충분하지 않다.

대표적인 Security Property는 다음과 같다.

1. **Preimage Resistance**
   Hash Value로부터 원래 Input을 찾기 어려워야 한다.
2. **Second Preimage Resistance**
   주어진 Message와 같은 Hash를 가지는 다른 Message를 찾기 어려워야 한다.
3. **Collision Resistance**
   같은 Hash를 가지는 어떤 두 Message를 찾는 것도 어려워야 한다.

핵심

Secure Hash Function은 One-Way Property와 Collision에 대한 저항성을 가져야 한다.

4-1. Preimage Resistance

Hash Value h가 주어졌다고 하자.

공격자의 목표는

$$H(x) = h$$

를 만족하는 Input x를 찾는 것이다.

$$x$$
$$\Downarrow$$
$$H(x)$$

Forward Direction은 쉽게 계산할 수 있지만,

$$h \to x$$

와 같은 Reverse Direction은 현실적으로 어려워야 한다.

이를 **Preimage Resistance** 또는 **One-Way Property**라고 한다.

핵심

특정 Hash Value가 주어졌을 때 그 Hash를 만드는 Input을 찾기 어려워야 한다.

4-2. Second Preimage Resistance

이번에는 특정 Message x가 주어진다.

공격자의 목표는

$$y \neq x$$

이면서

$$H(y) = H(x)$$

를 만족하는 다른 Message y를 찾는 것이다.

즉,

$$x \quad \longrightarrow \quad H(x)$$

와 동일한 Hash를 가지는 다른 Message를 찾아서는 안 된다.

핵심

주어진 Message와 동일한 Hash를 가지는 다른 Message를 찾기 어려워야 한다.

4-3. Collision Resistance

Collision Attack에서는 특정 Message나 Hash Value가 미리 주어지지 않는다.

공격자는 아무 두 Message x, y를 찾아서

$$x \neq y$$

이면서

$$H(x) = H(y)$$

가 되도록 하면 된다.

즉 공격자는

$$(x, y)$$

두 Input을 모두 자유롭게 선택할 수 있다.

이 점이 Preimage Attack과 Collision Attack의 중요한 차이이다.

핵심

Collision Resistance는 같은 Hash를 가지는 어떤 두 Message라도 찾기 어려워야 한다는 Property이다.

5. Security Strength of Hash Functions

n-bit Hash Function은 최대

$$2^n$$

개의 서로 다른 Hash Value를 가질 수 있다.

하지만 공격의 종류에 따라 필요한 계산량은 서로 다르다.

Ideal n-bit Hash Function

Attack	Approximate Work
Preimage	2^n
Second Preimage	2^n
Collision	$2^{n/2}$

Collision Attack은 특정 Hash Value를 찾는 것이 아니라, 같은 Hash를 가지는 아무 두 Input을 찾으면 된다.

핵심

Hash Output Length가 n bit라고 해서 모든 공격에 대해 n-bit Security를 제공하는 것은 아니다.

5-1. Why Does Collision Require $2^{n/2}$?

Collision에서는 특정 Hash Value가 목표가 아니다.

공격자가 q개의 서로 다른 Message를 만들었다고 하자.

$$M_1, M_2, \ldots, M_q$$

각 Message의 Hash를 계산하면

$$H(M_1), H(M_2), \ldots, H(M_q)$$

를 얻는다.

이 Hash들 사이에서 비교할 수 있는 Pair의 수는

$$\binom{q}{2} = \frac{q(q-1)}{2} \approx \frac{q^2}{2}$$

이다.

즉 Message의 수가 증가하면 비교 가능한 Pair의 수는 약 q^2으로 증가한다.

핵심

Collision은 하나의 정해진 Hash를 맞추는 문제가 아니라, 많은 Hash 중 같은 값을 가지는 Pair를 찾는 문제이다.

5-2. The Birthday Paradox

Birthday Paradox도 같은 원리로 이해할 수 있다.

365개의 가능한 Birthday가 있다고 하자.

특정 사람과 같은 Birthday를 찾는 것과,

Any two people have the same birthday

를 찾는 것은 서로 다른 문제이다.

23명이 모이면 가능한 Pair의 수는

$$\binom{23}{2} = 253$$

개이다.

따라서 23명만 있어도 같은 Birthday를 가진 Pair가 존재할 확률은 약 50%가 된다.

핵심

사람의 수는 적어도 사람 사이의 Pair는 훨씬 빠르게 증가한다. 이것이 Birthday Paradox의 핵심이다.

5-3. Birthday Attack on Hash Functions

n-bit Hash Function에는

$$2^n$$

개의 가능한 Hash Value가 있다.

q개의 Message를 Hash하면 비교 가능한 Hash Pair는 대략

$$q^2$$

개가 된다.

Collision이 나타날 정도의 규모를 생각하면

$$q^2 \approx 2^n$$

이므로,

$$q \approx \sqrt{2^n}$$

이다.
따라서

$$q \approx 2^{n/2}$$

가 된다.

핵심

Birthday Attack 때문에 이상적인 n-bit Hash Function의 Collision Security는 약 $n/2$ bit이다.

5-4. Example: 128-bit Hash

128-bit Hash Function을 생각해보자.

가능한 Hash Value의 수는

$$2^{128}$$

개이다.

특정 Hash Value에 대한 Preimage를 찾으려면 이상적으로 약

$$2^{128}$$

수준의 계산이 필요하다.

하지만 Collision은 Birthday Attack 때문에

$$2^{128/2} = 2^{64}$$

수준의 계산으로 찾을 수 있다.

128-bit Hash

Attack	Approximate Work
Preimage	2^{128}
Collision	2^{64}

5-5. What Does SHA-256 Mean?

SHA-256의 “256”은 Security Level을 직접 의미하는 것이 아니다.

SHA-256은 Variable-Length Input으로부터 **256-bit Hash Value**를 생성한다.

$$M \xrightarrow{\text{SHA-256}} h$$

$$|h| = 256 \text{ bits}$$

이상적인 256-bit Hash Function이라면

- Preimage Attack: approximately 2^{256}
- Second Preimage Attack: approximately 2^{256}
- Collision Attack: approximately 2^{128}

중요

SHA-256의 “256”은 Hash Output Length이다. 어떤 공격을 고려하는지에 따라 Effective Security Strength는 달라질 수 있다.

6. Secure Hash Algorithms

대표적인 Cryptographic Hash Function으로 Secure Hash Algorithm (SHA) 계열이 있다.

SHA Family

Algorithm	Hash Output Length
SHA-1	160 bits
SHA-256	256 bits
SHA-384	384 bits
SHA-512	512 bits

SHA-256, SHA-384, SHA-512 등은 SHA-2 Family에 속한다.

SHA-3는 SHA-1 및 SHA-2와 다른 구조를 사용하는 Hash Function Family이다.

핵심

SHA Algorithm 이름의 숫자는 일반적으로 생성되는 Hash Value의 길이를 나타낸다.

6-1. SHA-1, SHA-2, and SHA-3

SHA 계열은 시간이 지나면서 새로운 Algorithm으로 발전해왔다.

SHA-1

⇓

SHA-2

⇓

SHA-3

- **SHA-1**
  160-bit Hash Value를 생성한다.
- **SHA-2**
  SHA-256, SHA-384, SHA-512 등의 Algorithm을 포함한다.
- **SHA-3**
  SHA-1과 SHA-2와는 다른 내부 구조를 사용하는 별도의 Hash Function Family이다.

핵심

Secure Hash Algorithm도 Security 요구와 Cryptanalysis의 발전에 따라 새로운 표준이 개발되어 왔다.

7. Other Applications of Hash Functions

Cryptographic Hash Function은 Message Authentication 이외에도 다양한 Security Mechanism에서 사용된다.

대표적인 예는 다음과 같다.

- **Password Verification**
  Password 자체 대신 Password로부터 계산한 Hash Value를 이용한다.
- **File Integrity Checking**
  파일의 Hash Value를 저장해 두고, 나중에 다시 계산하여 파일이 변경되었는지 확인한다.
- **Digital Signatures**
  전체 Message 대신 Message Digest를 이용하여 효율적으로 Digital Signature를 생성할 수 있다.

핵심

Hash Function은 Data의 Compact Fingerprint를 제공하기 때문에 다양한 Security Mechanism의 기본 도구로 사용된다.

7-1. Hash Functions and Passwords

Password P 자체를 저장하는 대신 Hash Value를 저장한다고 생각해보자.

Registration:

$$P \xrightarrow{H} H(P)$$

Stored:

$$H(P)$$

Login 시 사용자가 입력한 Password P'에 대해

$$H(P')$$

를 계산하고 Stored Hash와 비교한다.

$$H(P') \stackrel{?}{=} H(P)$$

핵심

Password Verification에서는 입력된 Password가 동일한 Hash를 만드는지 확인하여 Password를 검증할 수 있다.

7-2. Hash Functions and File Integrity

파일이 변경되었는지 확인하기 위해 Hash Function을 사용할 수 있다.

원래 File F에 대해

$$h = H(F)$$

를 계산하여 안전하게 저장한다.

나중에 File F'에 대해

$$h' = H(F')$$

를 다시 계산한다.

그리고

$$h' \stackrel{?}{=} h$$

를 비교한다.

Hash Value가 다르면 File이 변경되었다는 것을 알 수 있다.

핵심

Hash Function은 File의 Fingerprint를 생성하여 Data Integrity를 확인하는 데 사용할 수 있다.

8. Encryption, MAC, and Hash

지금까지 배운 Cryptographic Tool을 비교해보자.

Different Cryptographic Tools

	Encryption	MAC	Hash
Secret Key	Yes	Yes	No
Confidentiality	Yes	No	No
Integrity	Not necessarily	Yes	Can detect changes
Source Authentication	Not necessarily	Yes	No
Typical Output	Ciphertext	Authentication Tag	Message Digest

각 Cryptographic Tool은 서로 다른 Security Goal을 위해 사용된다.

핵심

Encryption, MAC, Hash는 비슷해 보일 수 있지만 목적과 Security Property가 서로 다르다.

9. What We Learned

이번 Section에서는 Message Authentication과 Hash Function을 살펴보았다.

- Encryption은 주로 Confidentiality를 제공한다.
- Message Authentication은 Integrity와 Source Authentication을 확인한다.
- MAC은 Message와 Secret Key를 이용하여 Authentication Tag를 생성한다.
- Hash Function은 Variable-Length Input을 Fixed-Length Hash Value로 변환한다.
- Hash Function 자체에는 Secret Key가 없으므로 Hash만으로 Source Authentication을 제공하지 않는다.
- Secure Hash Function에는 Preimage, Second Preimage, Collision Resistance가 중요하다.
- n-bit Hash의 Preimage Security는 이상적으로 약 2^n이지만, Collision Security는 Birthday Attack 때문에 약 $2^{n/2}$이다.
- SHA-256의 256은 Security Level이 아니라 Hash Output Length를 의미한다.

핵심

Cryptographic Tool을 사용할 때는 Algorithm 자체뿐 아니라 어떤 Security Goal과 Attack을 고려하는지 이해해야 한다.

Thank You

Questions?