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

2.4 Digital Signatures and Key Management

Authentication, Certificates, and Digital Envelopes

1. Why Digital Signatures?

앞에서는 Public-Key Cryptography를 이용하여 Confidentiality와 Authentication을 제공하는 기본 아이디어를 살펴보았다.

이번에는 Bob이 Alice에게 Message를 보낸다고 하자.

Message를 숨기는 것은 중요하지 않지만, Alice는 다음을 확인하고 싶다.

- **Source Authentication**
  Message가 실제 Bob으로부터 왔는가?
- **Integrity**
  Message가 전송 도중 변경되지 않았는가?

이를 위한 대표적인 방법이 **Digital Signature**이다.

핵심

Digital Signature는 Sender의 Identity와 Message Integrity를 검증하기 위한 Cryptographic Mechanism이다.

1-1. Creating a Digital Signature

Bob이 Message M에 Digital Signature를 생성한다고 하자.

Step 1. Calculate Hash

먼저 Message의 Hash Value를 계산한다.

$$h = H(M)$$

Step 2. Generate Signature

Bob은 자신의 Private Key PR_B를 이용하여 Digital Signature를 생성한다.

$$S_B = \mathrm{Sign}_{PR_B}(h)$$

Step 3. Send Message and Signature

Bob은 다음을 Alice에게 전송한다.

$$\boxed{M \parallel S_B}$$

핵심

Digital Signature는 Message의 Hash와 Sender의 Private Key를 이용하여 생성한다.

1-2. Verifying a Digital Signature

Alice는 Bob으로부터 Message M과 Signature S_B를 받는다.

Step 1. Calculate Hash

Received Message의 Hash를 계산한다.

$$h_1 = H(M)$$

Step 2. Verify Signature

Bob의 Public Key PU_B를 이용하여 Signature를 검증한다.

$$\text{Verify}_{PU_B}(h_1, S_B)$$

$$\text{Valid Signature} \longrightarrow \text{Accept}$$

$$\text{Invalid Signature} \longrightarrow \text{Reject}$$

단, Bob의 Public Key가 실제 Bob의 것이라는 전제가 필요하다.

핵심

Private Key로 생성한 Signature를 Public Key로 검증하여 Authentication과 Integrity를 확인한다.

1-3. (참고) RSA Key Pair and Signature Generation

RSA를 이용하여 Digital Signature의 수학적 원리를 간단히 살펴보자.

Bob의 Key Pair는 다음과 같다.

$$PU_B = (n, e), \qquad PR_B = (n, d)$$

여기서 e와 d는 다음 관계를 만족한다.

$$ed \equiv 1 \pmod{\varphi(n)}$$

Signature Generation (Bob)

먼저 Message M의 Hash Value를 계산한다.

$$h = H(M)$$

Bob은 자신의 Private Key d를 이용하여 서명한다.

$$\boxed{S_B = h^d \bmod n}$$

핵심

RSA의 단순화된 모델에서는 Message의 Hash Value에 Private Key를 적용하여 Digital Signature를 생성한다.

1-4. (참고) RSA Signature Verification

Alice는 Bob으로부터 Message M과 Signature S_B를 받는다.

Step 1. Calculate Hash

Received Message의 Hash Value를 계산한다.

$$h_1 = H(M)$$

Step 2. Verify with Public Key

Bob의 Public Key (n, e)를 이용하여 계산한다.

$$h_2 = S_B^e \bmod n$$

Bob이 생성한 Signature는 $S_B = h^d \bmod n$이므로,

$$h_2 = (h^d)^e \bmod n = h$$

따라서 Alice는 다음을 확인한다.

$$\boxed{h_1 \stackrel{?}{=} h_2}$$

핵심

RSA에서는 Private Key와 Public Key의 수학적 관계를 이용하여 Signature를 검증한다.

※ RSA의 수학적 원리를 설명하기 위한 단순화 모델이다. 실제 RSA Signature는 RSA-PSS 등의 안전한 방식을 사용한다.

1-5. Digital Signature vs. Encryption

Digital Signature와 Public-Key Encryption은 서로 다른 Security Goal을 제공한다.

Different Security Goals

	Public-Key Encryption	Digital Signature
Primary Goal	Confidentiality	Authentication / Integrity
Sender Uses	Receiver's Public Key	Sender's Private Key
Receiver Uses	Receiver's Private Key	Sender's Public Key
Message Secrecy	Yes	No

Digital Signature는 Message를 숨기지 않는다.

$$M \parallel S_B$$

따라서 공격자가 Message를 읽을 수는 있지만, Bob의 Private Key 없이 유효한 Signature를 위조하기는 어려워야 한다.

핵심

Encryption은 Confidentiality를, Digital Signature는 Authentication과 Integrity를 위한 Mechanism이다.

2. Why Public-Key Certificates?

Digital Signature를 검증하려면 Sender의 Public Key가 필요하다.

그런데 중요한 질문이 남아 있다.

Is this really Bob's Public Key?

공격자 Eve가 자신의 Public Key를 보내면서 Bob의 Public Key라고 주장할 수도 있다.

이를 해결하기 위해 **Certificate Authority (CA)**를 사용한다.

CA는 Bob의 Identity와 Public Key의 관계를 확인하고 그 정보에 Digital Signature를 생성한다.

$$\boxed{\text{Bob's Identity} + PU_B + \text{CA Signature}}$$

핵심

Public-Key Certificate는 특정 Public Key가 누구의 것인지 검증할 수 있도록 하는 Mechanism이다.

2-1. Creating and Verifying a Certificate

1. Certificate Creation

CA는 Bob의 Identity와 Public Key를 포함하는 Certificate Data를 준비한다.

$$Cert_B = (ID_B, PU_B, \text{Validity}, \ldots)$$

CA는 Certificate Data의 Hash를 계산하고 자신의 Private Key로 서명한다.

$$h = H(Cert_B)$$

$$S_{CA} = \text{Sign}_{PR_{CA}}(h)$$

2. Certificate Verification

Alice는 신뢰하는 CA의 Public Key PU_{CA}로 Certificate의 Signature를 검증한다.

$$\text{Verify}_{PU_{CA}}(H(Cert_B), S_{CA})$$

Signature가 유효하고 Certificate의 다른 검증 조건도 만족하면, Alice는 Bob의 Public Key를 신뢰할 수 있다.

핵심

CA는 자신의 Private Key로 Certificate에 서명하고, 사용자는 CA의 Public Key로 이를 검증한다.

3. Digital Envelopes

Public-Key Encryption은 Key Distribution에 유용하지만, 대용량 Message 암호화에는 상대적으로 비효율적이다.

따라서 Symmetric Encryption과 Public-Key Encryption을 함께 사용할 수 있다.

Bob이 Alice에게 Message M을 보낸다고 하자.

Step 1. Generate a Random Symmetric Key

$$K_s \leftarrow \text{Random Key Generation}$$

Step 2. Encrypt the Message

$$C_M = E_{K_s}(M)$$

Step 3. Encrypt the Symmetric Key

Alice의 Public Key PU_A를 사용한다.

$$C_K = E_{PU_A}(K_s)$$

Step 4. Send the Digital Envelope

$$\boxed{C_M \parallel C_K}$$

핵심

Digital Envelope는 Symmetric Encryption으로 Message를 보호하고, Public-Key Encryption으로 Symmetric Key를 보호한다.

4. Digital Signatures and Key Management

이번 Section에서는 Public-Key Cryptography의 실제 활용을 살펴보았다.

Cryptographic Mechanisms

Mechanism	Main Purpose
Digital Signature	Authentication and Integrity
Public-Key Certificate	Verify Public-Key Ownership
Digital Envelope	Protect Messages Using Symmetric and Public-Key Encryption

- Digital Signature는 Sender의 Private Key로 생성한다.
- Certificate는 CA의 Signature를 통해 Public Key의 Identity를 확인하도록 한다.
- Digital Envelope는 Symmetric Encryption과 Public-Key Encryption의 장점을 결합한다.

핵심

실제 Security System은 하나의 Cryptographic Algorithm이 아니라, 여러 Cryptographic Tools를 결합하여 구성된다.

Chapter 2. Cryptographic Tools

2.5 Random and Pseudorandom Numbers

Randomness and Unpredictability

1. Why Do We Need Random Numbers?

Cryptographic Algorithms에서는 안전한 Random Number가 중요하다.

대표적인 활용 사례는 다음과 같다.

- **Key Generation**
  RSA Key Pair 및 Symmetric Secret Key를 생성한다.
- **Session Key / Digital Envelope**
  통신 세션 또는 일회성 Message Encryption에 사용할 Key를 생성한다.

예를 들어 AES-256의 Key Space는 2^{256}이지만, 실제 Key가 쉽게 예측된다면 안전하지 않다.

핵심

Cryptographic Security는 Algorithm뿐 아니라 Key와 Nonce를 생성하는 Randomness에도 의존한다.

2. Randomness vs. Unpredictability

Random Number의 중요한 두 가지 특성을 살펴보자.

1. Randomness (Statistical Properties)

- **Uniform Distribution**: 각 값이 동일한 확률로 나타난다.
- **Independence**: 이전 값이 다음 값의 확률에 영향을 주지 않는다.

예를 들어, 각 Bit가 Independent하고 Uniform하게 생성된다면,

$$X_i \overset{i.i.d.}{\sim} \text{Bernoulli}(1/2)$$

$$P(X_{n+1} = 0 \mid X_1, \ldots, X_n) = \frac{1}{2}$$

즉, 이전 Bit를 모두 알아도 다음 Bit를 예측할 수 없다.

2. Unpredictability (Security Property)

실제 Cryptographic System에서는 공격자가 이전 Output을 관찰하더라도 다음 값을 예측하기 어려워야 한다.

핵심

True i.i.d. Uniform Randomness는 예측 불가능성을 제공한다. 하지만 통계적으로 Random하게 보인다는 사실만으로 Cryptographic Unpredictability가 보장되지는 않는다.

2-1. (참고) Statistical Randomness vs. Predictability

다음과 같은 Deterministic Generator를 생각해보자.

$$X_1 \sim \text{Bernoulli}(1/2), \qquad X_{n+1} = 1 - X_n$$

생성되는 Sequence는 다음 둘 중 하나이다.

$$0, 1, 0, 1, 0, 1, \ldots$$

$$1, 0, 1, 0, 1, 0, \ldots$$

각 X_n은 Uniform Distribution을 따르지만,

$$P(X_{n+1} = 0 \mid X_n = 1) = 1$$

이므로 Independent하지 않다.

실제 PRNG도 Deterministic Algorithm으로 동작한다.

$$(X_1, X_2, \ldots) = G(\text{Seed})$$

통계적으로 Random하게 보이더라도, Seed가 노출되면 미래 Output을 예측할 수 있다.

핵심

Statistical Test를 통과하는 것과 Cryptographic Unpredictability를 보장하는 것은 다르다. 따라서 암호학에서는 안전한 PRNG와 Seed 관리가 중요하다.

3. True Random vs. Pseudorandom Numbers

Random Number Generator는 크게 두 가지로 구분할 수 있다.

1. True Random Number Generator (TRNG)

- Thermal Noise 등 물리적인 비결정적 현상을 이용한다.
- 물리적 Entropy를 측정하고 처리하여 Random Bits를 생성한다.

$$\text{Physical Noise} \longrightarrow \text{TRNG} \longrightarrow R$$

2. Pseudorandom Number Generator (PRNG)

- Seed와 Deterministic Algorithm을 이용한다.
- 같은 Seed를 사용하면 동일한 Sequence가 생성된다.

$$\text{Seed} \longrightarrow \text{PRNG} \longrightarrow R_1, R_2, \ldots$$

핵심

TRNG는 물리적 현상에 기반하고, PRNG는 Deterministic Algorithm으로 난수열을 생성한다.

3-1. (참고) How Does Thermal Noise Generate Random Bits?

전자회로의 저항에서는 전자의 열운동 때문에 작고 불규칙한 Voltage Fluctuation이 발생한다.

이를 **Thermal Noise**라고 한다.

Thermal Noise (Analog Voltage)

⇓

Amplification and Sampling

⇓

Digitization (0 / 1)

⇓

Randomness Extraction / Conditioning

⇓

Random Bits

예를 들어 측정한 전압을 기준값과 비교하여 Bit를 만들 수 있다. 단, 편향과 잡음 품질을 점검해야 한다.

핵심

TRNG는 물리적 Noise를 측정한 뒤 Digital Bit로 변환하고, 필요한 후처리를 수행한다.

3-2. (참고) How Does a PRNG Work?

PRNG는 초기값인 **Seed**와 정해진 Algorithm으로 다음 값을 계산한다.

간단한 예로 Linear Congruential Generator (LCG)를 보자.

$$X_{i+1} = (aX_i + c) \bmod m$$

$X_0 = 1$, $a = 5$, $c = 3$, $m = 16$으로 정하면,

$$X_1 = (5 \cdot 1 + 3) \bmod 16 = 8$$
$$X_2 = (5 \cdot 8 + 3) \bmod 16 = 11$$
$$X_3 = (5 \cdot 11 + 3) \bmod 16 = 10$$

동일한 Seed와 Parameters를 사용하면 항상 동일한 Sequence가 생성된다.

핵심

PRNG는 Deterministic하다. 이 LCG는 원리 설명용이며 Cryptographic Key 생성에는 부적합하다.

4. Cryptographically Secure PRNG

PRNG가 Deterministic하다고 해서 반드시 Cryptography에 사용할 수 없는 것은 아니다.

Cryptographically Secure PRNG (CSPRNG)는 공격자가 Output을 관찰하더라도 다음 값을 예측하기 어렵도록 설계된다.

Unpredictable Seed

⇓

CSPRNG

⇓

Cryptographically Secure Random Bits

- Seed가 노출되거나 예측 가능하면 Security가 약화될 수 있다.
- Statistical Test를 통과하는 것만으로 Security가 보장되지는 않는다.

핵심

암호학에서는 단순히 Random해 보이는 것보다 공격자에 대한 **Unpredictability**가 중요하다.

4-1. (참고) Example: AES-Based CSPRNG

실제 CSPRNG는 AES와 같은 Cryptographic Algorithm을 이용하여 구성할 수 있다.

예를 들어 Secret Key K와 Counter를 사용하여 다음과 같이 Random Bits를 생성한다고 하자.

$$R_1 = AES_K(1)$$

$$R_2 = AES_K(2)$$

$$R_3 = AES_K(3)$$

여기서 Counter 값 $1, 2, 3, \ldots$은 누구나 알 수 있는 값이다.

하지만 공격자가 Secret Key K를 모른다면 AES Output을 예측하기 어렵다.

Secret Key K + Counter $\longrightarrow$ **AES** $\longrightarrow$ **Pseudorandom Bits**

핵심

CSPRNG는 Deterministic Algorithm을 사용하더라도 Secret Key와 안전한 Cryptographic Algorithm을 통해 예측하기 어려운 Output을 생성할 수 있다.

※ AES-CTR 기반 난수 생성의 개념적 예시이다. 실제 CTR_DRBG는 내부 상태 갱신과 Reseeding 등을 포함한다.

Thank You

Questions?