

# 키공유 프로토콜과 PQ/T 하이브리드

손용하

성신여자대학교 융합보안공학과

2026.02.24 @ KpqC Winter Camp

키 공유 프로토콜

(Authenticated Key Exchange; AKE)

# Motivation: 안전한 통신 채널 구축

---

Alice



Internet

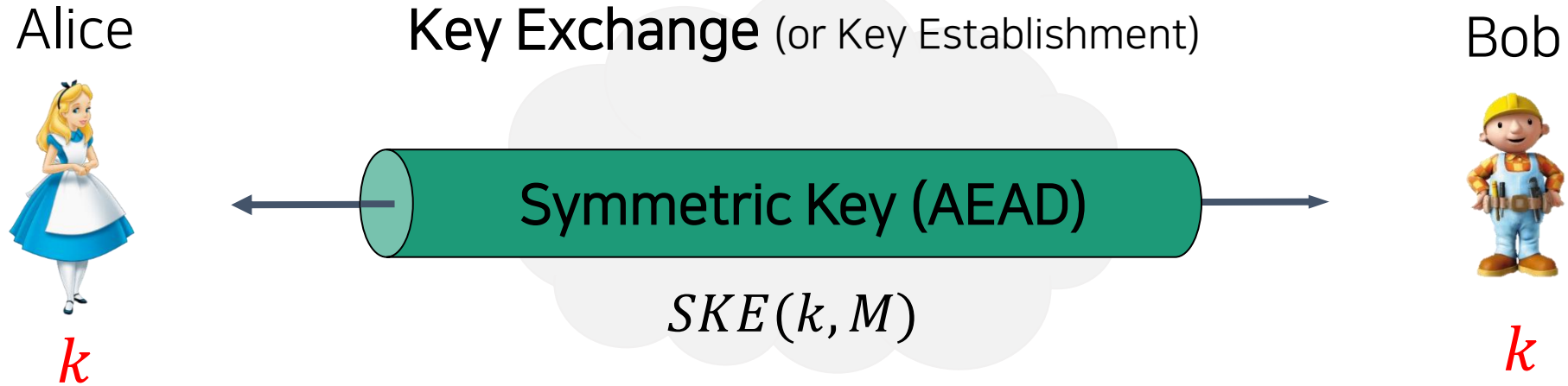


Bob



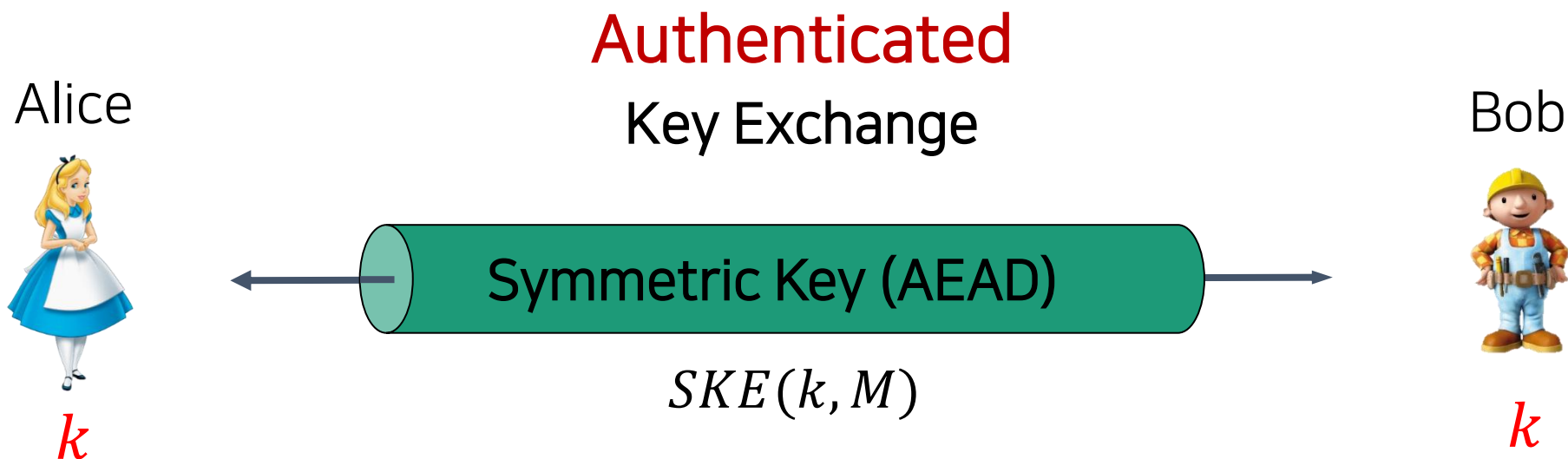
Adversary

# Motivation: 안전한 통신 채널 구축



- 키 교환을 먼저 수행하고, 실제 데이터는 (고속) 대칭키 암호화
  - 구체적으로는 기밀성과 무결성을 위해 AEAD(e.g. AES-GCM)를 사용
- 대표적인 예: Transport Layer Security (TLS) protocol for HTTPS

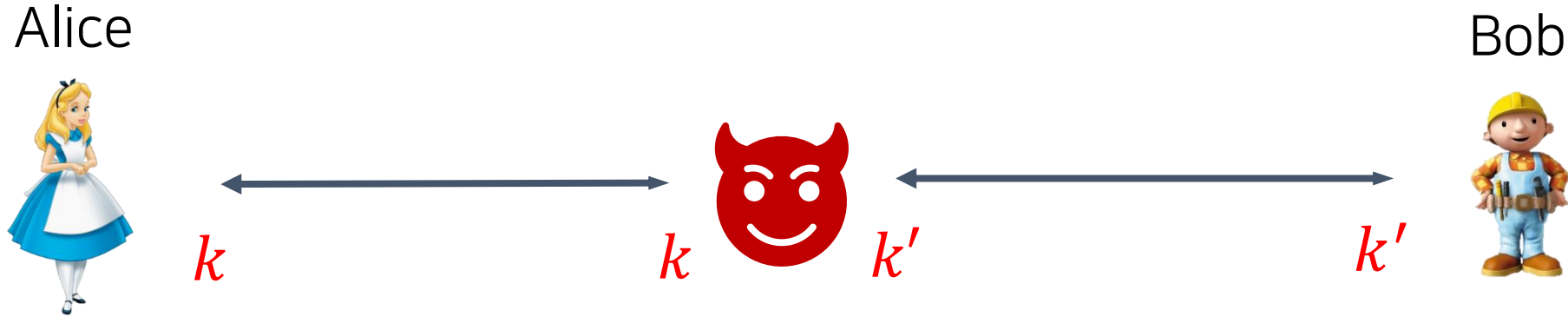
# Authenticated Key Exchange



Q. 키 교환은 KEM으로 바로 해결?

- No. 내가 통신을 주고받는 사람이 누구인지 확인하는 장치가 없음

# Authenticated Key Exchange



Q. 키 교환은 KEM으로 바로 해결?

- No. 내가 통신을 주고받는 사람이 누구인지 확인하는 장치가 없음
- Man-in-the-middle (Mitm) 공격 가능

A. KEM 적용 전에 Digital Signature 를 활용하여 송/수신자의 신원을 확인

# Authenticated Key Exchange

---

Alice



$\sigma = \text{Sign}(sk_B, \text{"안녕하세요, 저 Bob입니다."})$

$vk_B$

Bob



인척 하는  
공격자

$(vk_B, sk_B)$

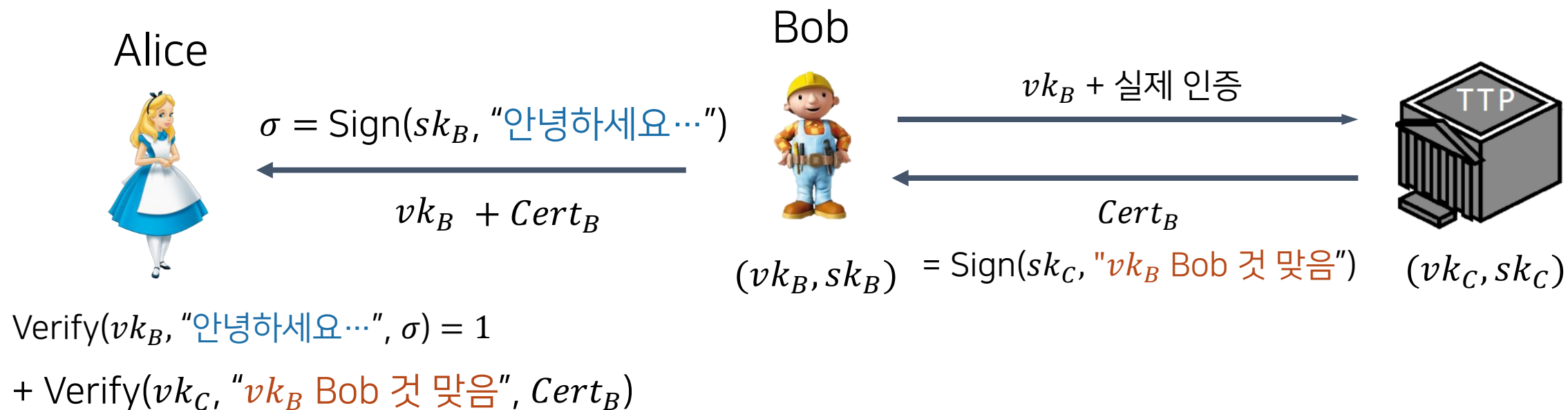
$\text{Verify}(vk_B, \text{"안녕하세요..."}, \sigma) = 1 ?$

---

A. KEM 적용 전에 Digital Signature 를 활용하여 송/수신자의 신원을 확인

- But 위와 같은 단순 적용은 아무것도 확인해주지 않음

# Authenticated Key Exchange



A. KEM 적용 전에 Digital Signature 를 활용하여 송/수신자의 신원을 확인

- Bob은 인증기관에게  $vk_B$ 에 대한 인증서  $Cert_B$ 를 발급받음 ( $\approx$  인증기관의 서명)
- Alice는  $\sigma, Cert_B$  모두 검증 ( $Cert_B$  검증은 인증기관의 공개키  $vk_C$ 로)



# Authenticated Key Exchange

## Authenticated Key Exchange

Alice



Bob



$vk_B, Cert_B, \sigma = \text{Sign}(sk_B, \text{"I'm Bob..."})$

$Cert_B$  &  $\sigma$  검증

- $sk_B$
- $vk_B$
- $Cert_B$

$Encap(pk_B)$

$k$

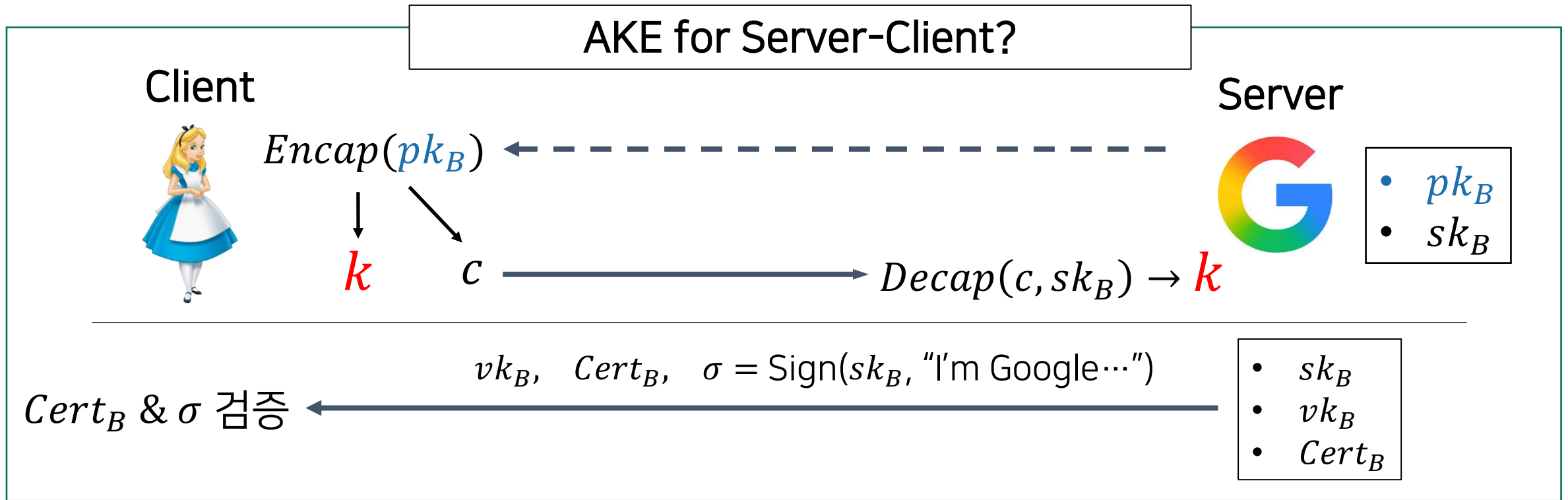
$c$

- $pk_B$
  - $sk_B$
- KEM  
key pair

$Decap(c, sk_B) \rightarrow k$

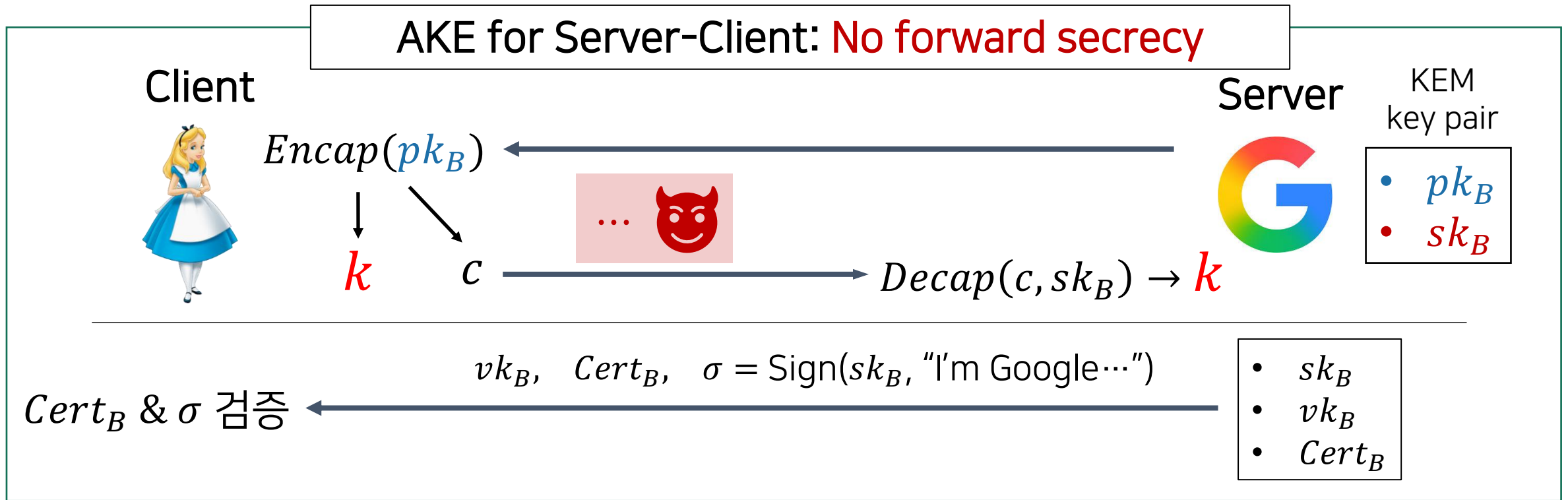
신원을 확인한 이후 KEM을 활용하여 키 공유  
(Bob도 Alice의 신원을 확인하는 과정 필요. 그림에서는 생략)

# AKE in the Server - Client Scenario



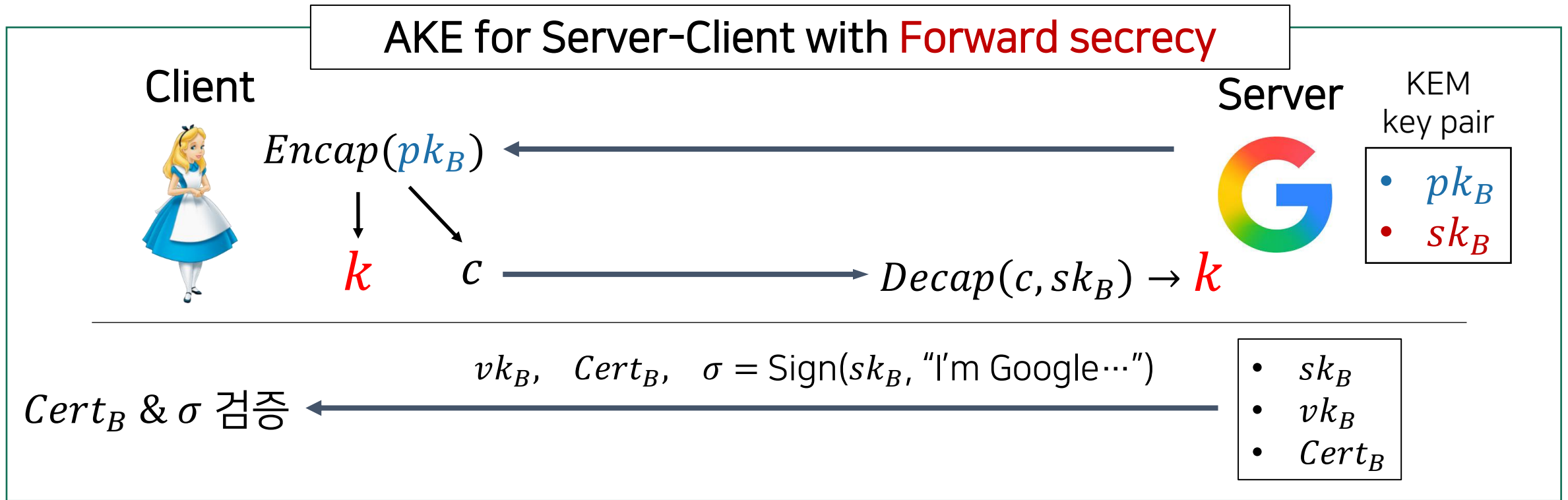
- Server는 Client를 기다리고, Client가 먼저 통신 요청을 보내는 경우가 많다.
  - 서버의 공개된  $pk_B$ 를 가져와서  $c$ 를 보내며 통신 요청 시작,
  - 이후 서버는 본인의 신원을 증명하기 위해 (MitM 방지) 응답하는 것이 자연스러움

# AKE in the Server - Client Scenario



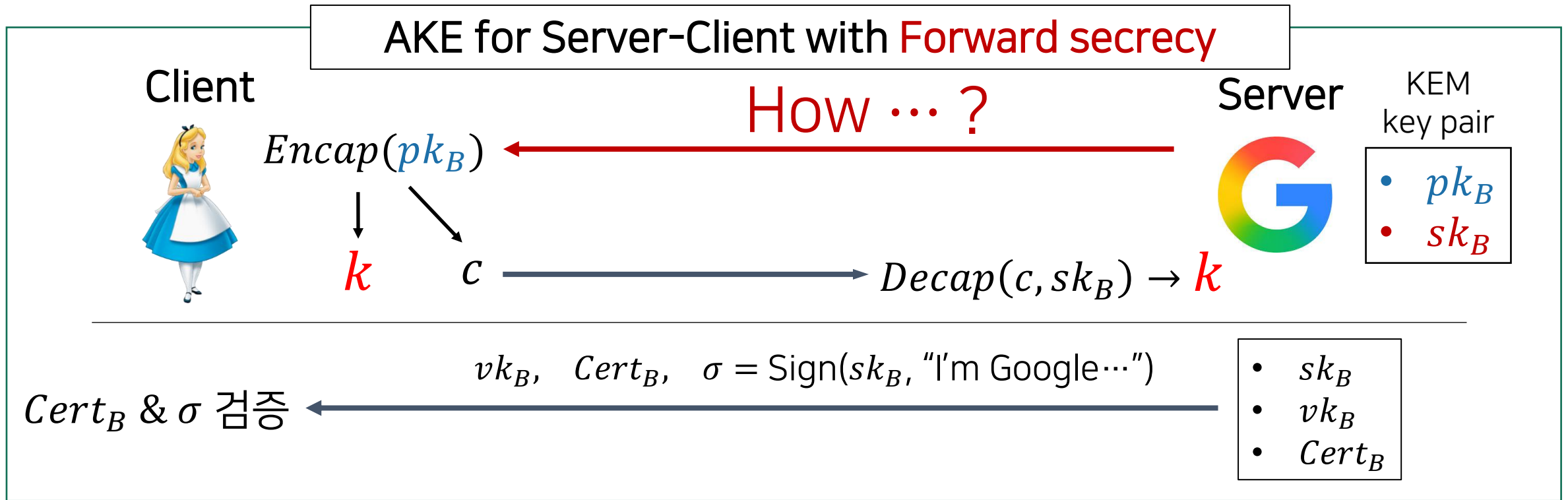
- 서버로 향하는 packet을 모두 모아두고 있는 공격자 🐼
- 추후 서버의 취약점을 활용하여  $sk_B$ 를 얻어낼 수 있다면
- 모아둔 모든 packet을 복호화 가능 = **Forward Secrecy**를 만족하지 못한다

# AKE in the Server - Client Scenario



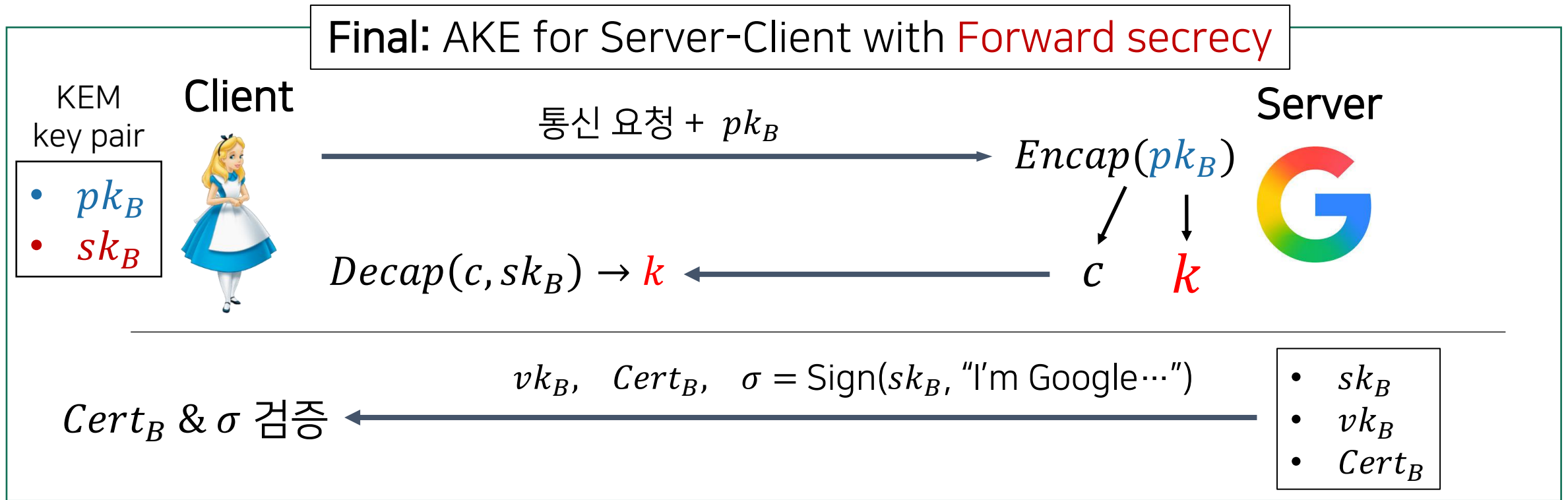
- 문제의 근원은 **고정된**  $(pk_B, sk_B)$ 
  - 매 번 KEM key pair를 새로 생성해주고
  - 사용한 Key Pair는 해당 Client와 연결을 끊을 때 삭제하면 해결!

# AKE in the Server - Client Scenario



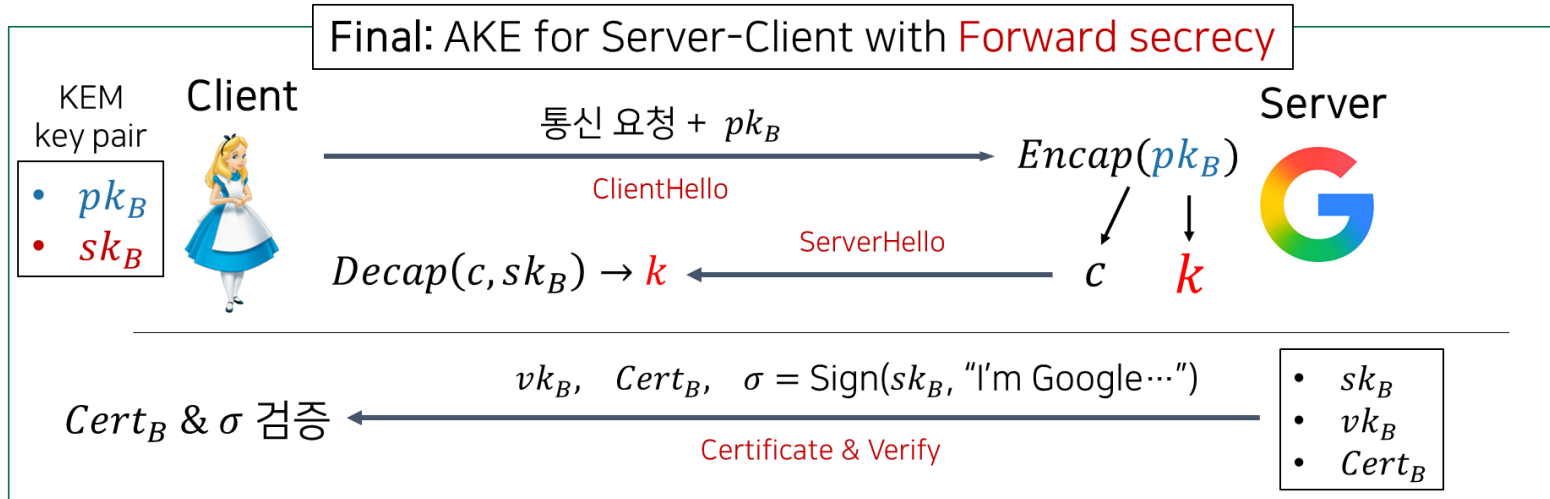
- 그런데, 이러면 Client가 먼저 접근해서  $pk_B$ 를 가져오는 방향이 맞지 않다:
- Client가 (아무 정보 없이) "통신 요청"을 보내면 Server는 Key pair 생성 ...
  - 1회의 통신이 추가로 필요

# AKE in the Server - Client Scenario



- KEM에서 역할을 바꿔서 해결 ☺
  - Client가 KEM key pair 생성하고 통신 요청과 동시에  $pk_B$  전송
  - 그 후 Server가 Encap - Client가 Decap

# AKE in the Server - Client Scenario

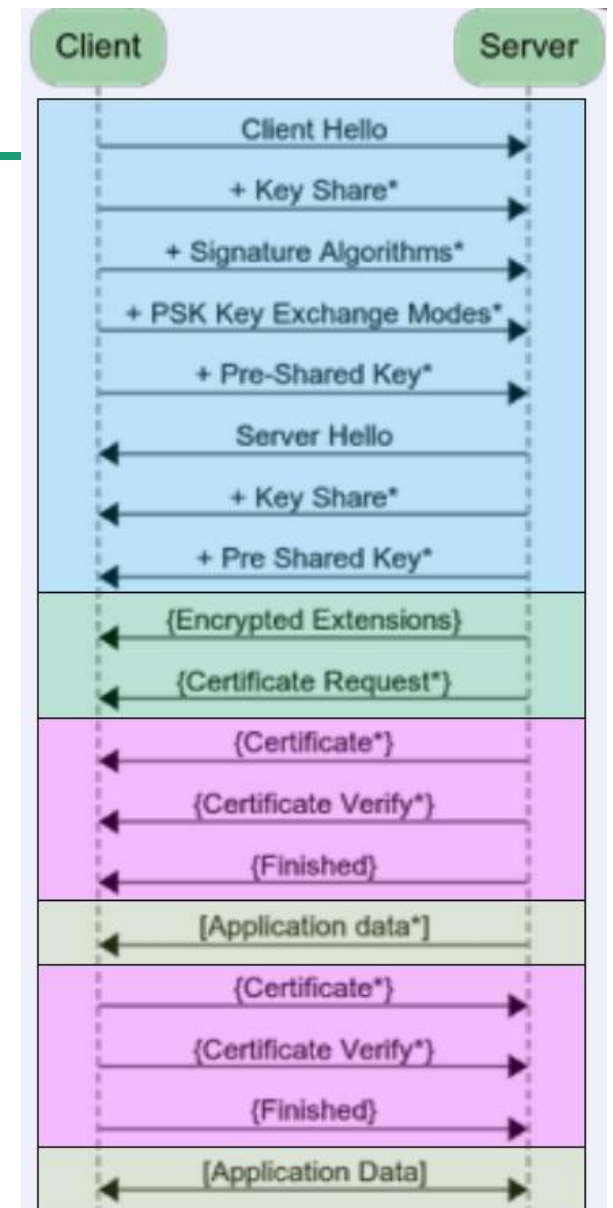


**Note 1.** Client의 신원도 확인할 필요가 있다면,

- Client의 인증서를 보내고 검증하는 단계만 추가하면 OK

**Note 2.** 위 설명은 KEM 기반으로 쓰여있긴 하지만,

- 해당 부분을 DH Key Exchange로 바꿔적은 것이 **TLS 1.3**



**TLS 1.3 Protocol**  
(Full handshake)

## Things to remember ...

Do not design AKE protocol yourself ...

# Just use latest version of TLS

현재 TLS 1.3이 최신입니다.

Dan Boneh  
דן בונה



Boneh in 2007

Born	1969 (age 56 - 57) <a href="#">Israel</a>
Alma mater	<a href="#">Princeton University</a> (PhD)
Known for	<a href="#">Pairing-based cryptography</a> <a href="#">ID-based encryption</a>
Awards	<a href="#">Gödel Prize</a> <a href="#">Selfridge Prize</a> <a href="#">ACM Prize in Computing</a> <a href="#">Sloan Research Fellowship</a>



Post-Quantum / Traditional  
Hybrid KEM

# Background: Traditional & Post-Quantum

- Traditional (Classical) Crypto
  - (양자 컴퓨팅에 안전하지 않지만) 현재 많은 시스템에서 사용되고 있으며
  - 고전 컴퓨터에 대한 안전성은 비교적 오랜 기간 검증되어 안정화됨
- Post-Quantum Crypto
  - 이론적인 안전성은 어느정도 합의가 된 듯 하지만,
  - 실제 (Real-World) 안전성 (e.g. Side-Channel, Concrete Attack)은 아직 안정되지 않음
  - 기반 난제가 **이론적으로** 공격당할 가능성도 무시할 수 없음

## Round four

On July 5, 2022, NIST announced four candidates for PQC Standardization Round 4.

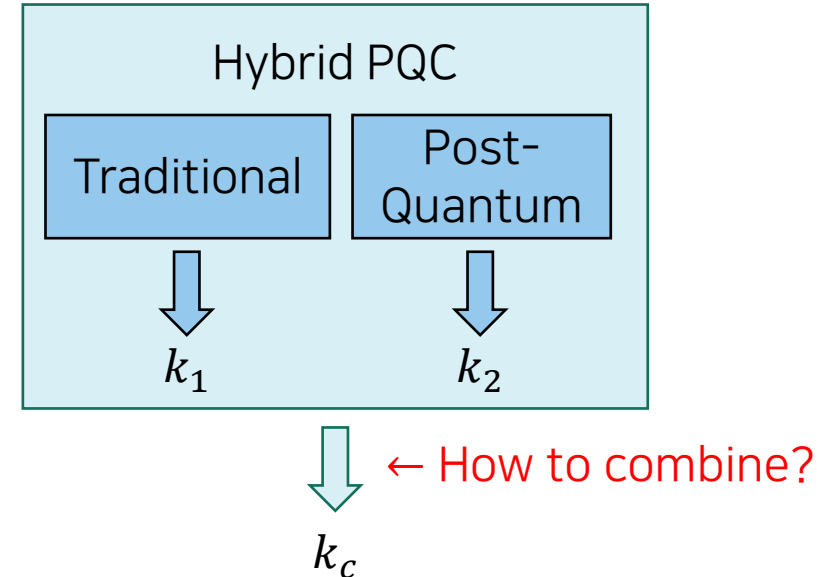
Type	PKE/KEM
Code-based	<ul style="list-style-type: none"><li>• <a href="#">BIKE</a></li><li>• <a href="#">Classic McEliece</a></li><li>• <a href="#">HQC</a></li></ul>
Supersingular elliptic curve isogeny	<ul style="list-style-type: none"><li>• <a href="#">SIKE</a> (<u>Broken August 5, 2022</u>)</li></ul>

# Post-Quantum / Traditional Hybrid

- 기존의 공개키 암호와 양자내성암호를 동시에 사용하여  
두 암호 중 하나만 안전해도 전체가 안전하도록 설계하는 컨셉
  - 양자컴퓨터가 기존 공개키 암호를 깨는 시점에서는 양자 내성을 가짐
  - 혹시 양자내성암호가 깨지더라도, 최소한 기존 공개키 암호의 안전성을 가짐

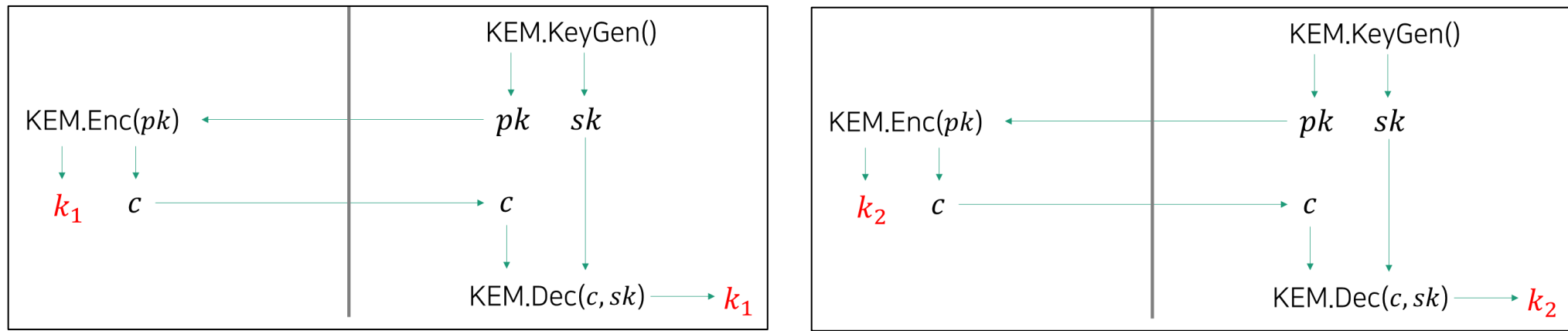
구체적으로 **어떻게** 동시에 사용하는지,  
즉, 어떻게 Combine하는지

KEM/DSA 각각 한 가지씩 알아보겠습니다.



# Hybrid KEM Combiner

- PQ / T KEM을 각각 수행하여  $k_1, k_2$  만들고 ...
- Goal: PQ or T 둘 중 하나라도 안전하면 Combination 결과물도 안전하도록  $k_1, k_2$ 를 조합

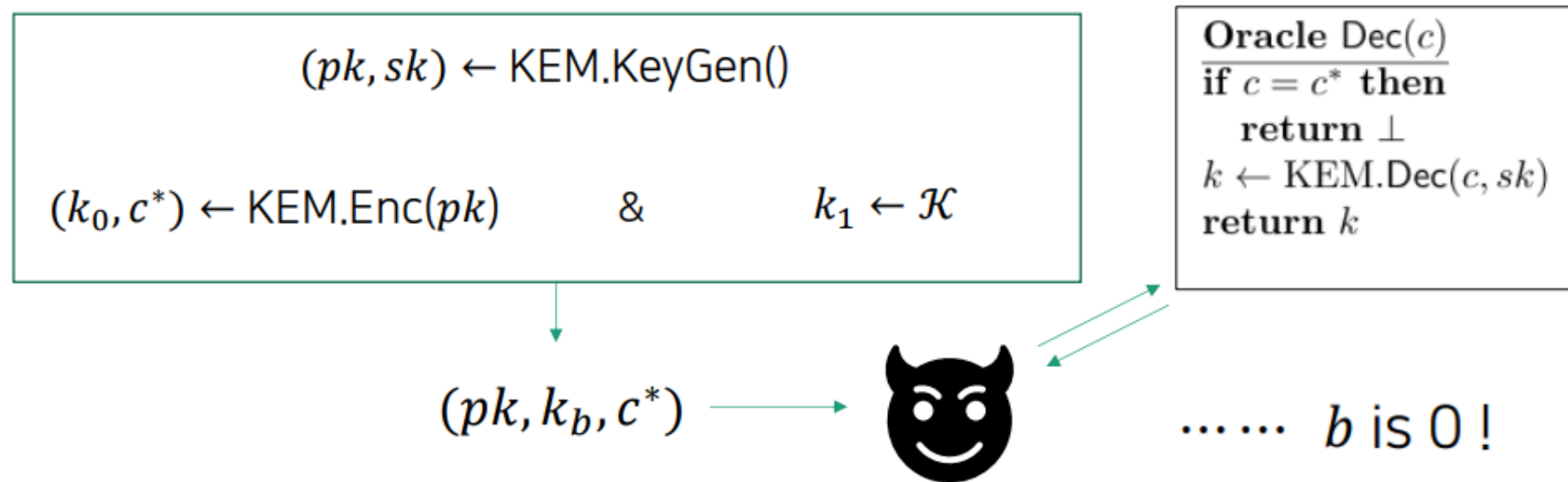


**Takeaway 1.** 가장 쉽게 생각할 수 있는 방법  $k_c := H(k_1 || k_2)$

IND-CCA secure하지 않음

# Recall: IND-CCA Security of KEM

- $(pk, c)$  및 Decryption Oracle을 가지고 있는 공격자가,  
 $c$ 에 담겨있는  $k_0$  v.s. key space  $\mathcal{K}$ 에서 무작위로 선택된  $k_1$  을 구별할 수 없다.



원하는 Decryption이 가능한 경우에도  
공개키  $pk$ , 암호문  $c$ 로부터 내부  $k$ 에 대한 정보를 알 수 없다는 뜻

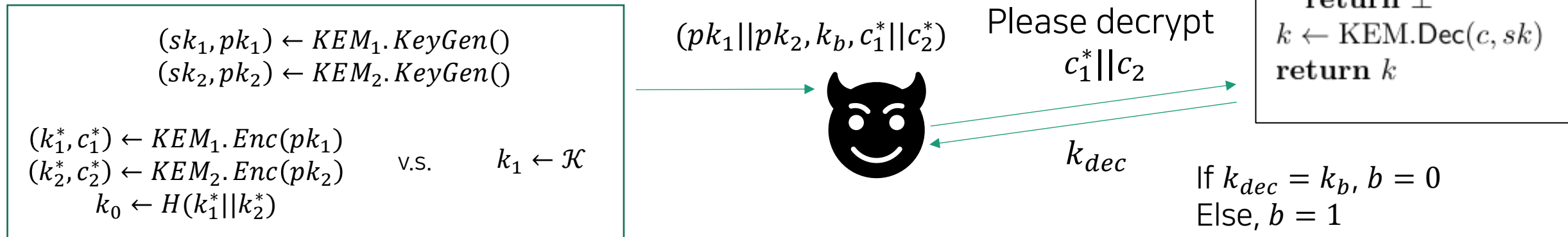
# Hybrid KEM Combiner

- PQ / T KEM을 각각 수행하여  $k_1, k_2$  만들고 ...
- Goal: PQ or T 둘 중 하나라도 안전하면 Combination 결과물도 안전하도록  $k_1, k_2$ 를 조합
  - **Takeaway 1.**  $k_c := H(k_1 || k_2)$  는 IND-CCA secure 하지 않다.

∴ (1번째 KEM이 IND-CCA더라도) 2번째 KEM에 대해

$c_2 \neq c_2^*$  이면서  $Dec_2(c_2) = k_2$  인  $c_2$  를 찾을 수 있다면,  $(c_1^* || c_2)$  를 Dec Oracle에 질의할 수 있음

⇒ Dec Oracle의 대답으로 구별 가능



# Hybrid KEM Combiner – 2018 [GHP18]

- PQ / T KEM을 각각 수행하여  $k_1, k_2$  만들고 ...
- Goal: PQ or T 둘 중 하나라도 안전하면 Combination 결과물도 안전하도록  $k_1, k_2$ 를 조합
  - **Takeaway 1.**  $k_c := H(k_1 || k_2)$  는 IND-CCA secure 하지 않다.
- 해결방안:  $k_c$  생성할 때 각 KEM의 암호문  $c_i$ 도 같이 넣는다.

$$k_c := H(k_1 || k_2 || c_1 || c_2)$$

( $\because c_2 \neq c_2^*$  이면서  $Dec_2(c_2) = k_2$  인  $c_2$  를 찾더라도 다른  $k_c$  가 생성)

Theorem

$H$ 가 Random Oracle이고, 구성 KEM 중 하나라도 IND-CCA이면

$H(k_1 || k_2 || c_1 || c_2)$  조합은 IND-CCA

# Hybrid KEM Combiner – 2025: X-Wing

\* [BCD+25], X-Wing: The Hybrid KEM You've Been Looking For, CiC

- [GHP18]  $k_c := H(k_1 \| k_2 \| c_1 \| c_2)$  를 ML-KEM / X25519의 특성을 활용하여 최적화
  1. ML-KEM의 암호문  $c_1$ 을 제외할 수 있음을 증명
  2. X25519의 공개키  $pk_2$ 도 key에 포함해야 함을 증명

**Takeaway 2. [X-Wing]  $k_c := H(k_1 \| k_2 \| c_2 \| pk_2)$**

- 사실 이것 자체가 주는 실질적 이득은 그렇게 크지 않지만 ...
  - [GHP18]의 Hash 입력:  $6 + 32 + 32 + 1088 + 32 = 1222$  bytes
  - [X-Wing]의 Hash 입력:  $6 + 32 + 32 + 32 + 32 = 132$  bytes
    - SHA3-256 1 block에서 처리 가능하여 키 결합에 필요한 연산이 약간 빨라짐 (10% 정도 개선)
- IETF 표준화가 잘 진행되고 있는 방향입니다.



# Hybrid KEM Combiner – 2025: X-Wing

---

- [GHP18]  $k_c := H(k_1 \| k_2 \| c_1 \| c_2)$  를 ML-KEM / X25519의 특성을 활용하여 최적화
  1. ML-KEM의 암호문  $c_1$ 을 제외할 수 있음을 증명
  2. X25519의 공개키  $pk_2$ 도 key에 포함해야 함을 증명

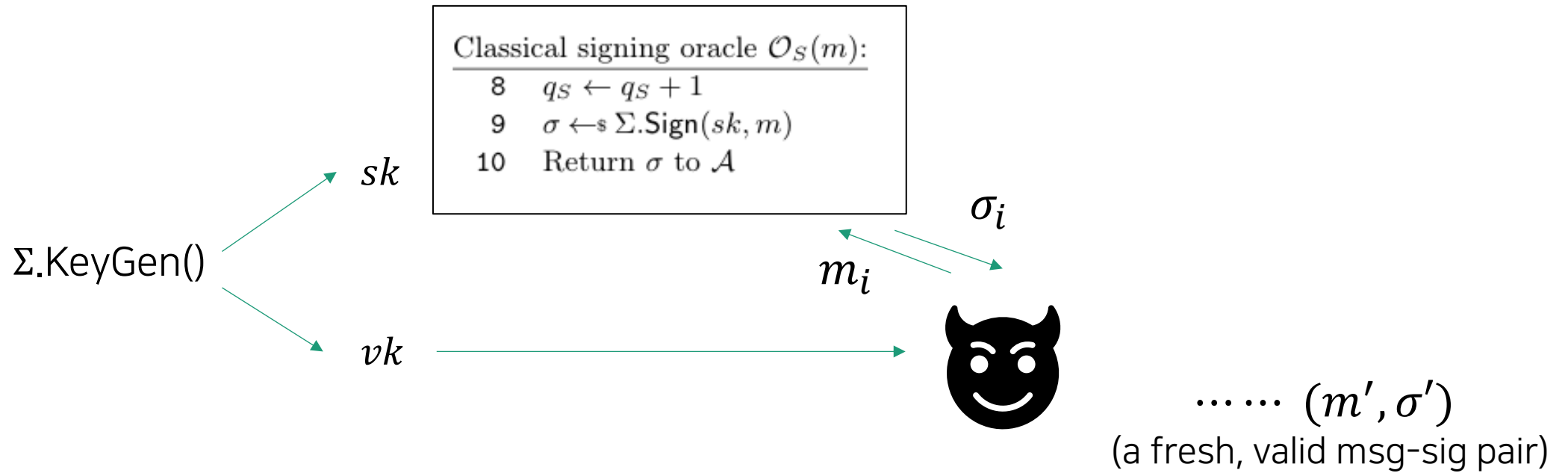
**Takeaway 2. [X-Wing]  $k_c := H(k_1 \| k_2 \| c_2 \| pk_2)$**

- 이후 추가적으로 연구된 사항:
  - ML-KEM → HQC, FrodoKEM, McEliece, NTRU, NTRU+, SMAUG-T 사용 OK
  - X25519 → 임의의 prime-order group 상의 DH Key Exchange 사용 OK
    - Ex) NIST P-256 커브

# Post-Quantum / Traditional Hybrid Digital Signature

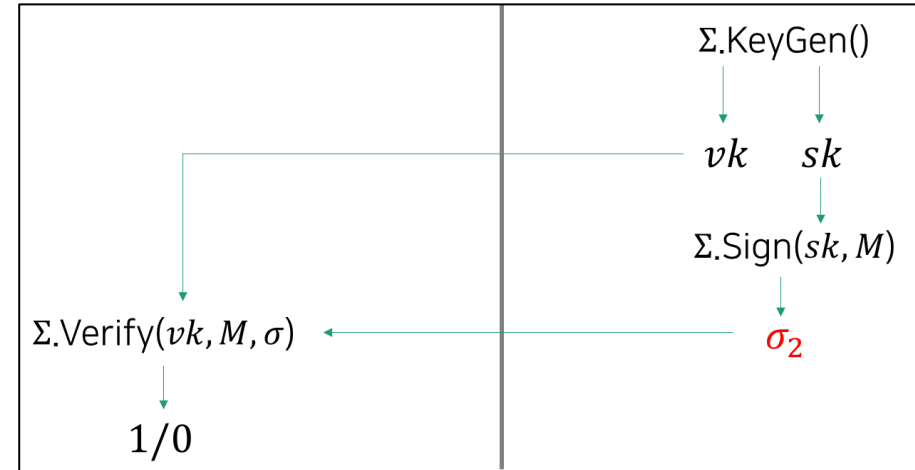
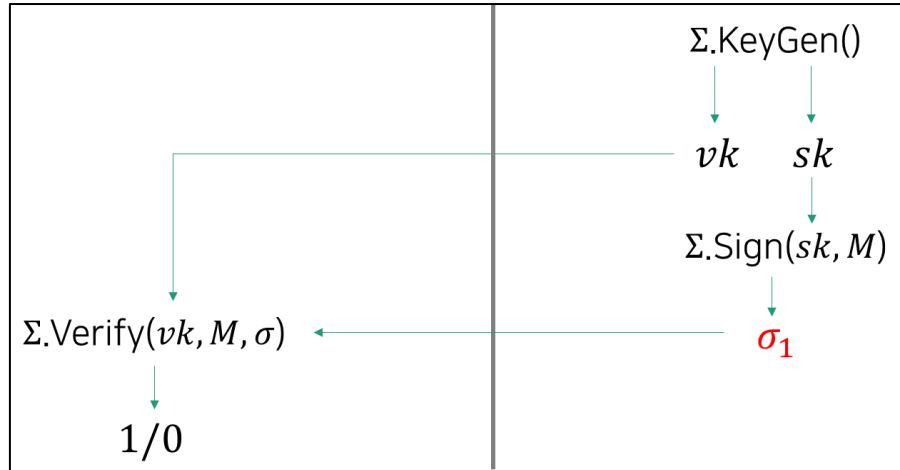
# Recall: EUF-CMA Security of DSA

- Signing Oracle이 주어진 상황에서,  
질의하지 않은 메시지에 대한 서명을 만들어낼 수 있으면 공격 성공
- EUF-CMA: 이러한 공격이 어려움



# Hybrid Digital Signature

- PQ / T Sig를 각각 수행하여  $\sigma_1, \sigma_2$  만들고 ...
- Goal: PQ or T 둘 중 하나라도 안전하면 Combination 결과물도 안전하도록  $\sigma_1, \sigma_2$  를 조합



가장 쉽게 생각할 수 있는 방법  $\sigma_c := \sigma_1 || \sigma_2$  는

EUF-CMA Secure 하다 (...!)

\* [BHMS17], Transitioning to a Quantum-Resistant Public Key Infrastructure, PQCrypto'17

# Hybrid Digital Signature with Labeling

---

- 한 가지 Issue:  $\sigma_c := \sigma_1 || \sigma_2$  로 Hybrid 서명을 만들면,
  - 각 Sign의 Valid 서명  $(m, \sigma_1) / (m, \sigma_2)$  을 곧바로 얻을 수 있음
  - = Hybrid 시스템의 서명만 있어도 Legacy 시스템 / PQ 시스템에 써먹을 수 있음
- **Takeaway 3. Concatenate + Hybrid Labeling**
  - “Hybrid” 와 같은 label을 메시지 앞에 붙이면, Hybrid 시스템에만 유효한 서명이 된다.
  - $\sigma_1, \sigma_2$ 는 “Hybrid|| $m$ ”라는 메시지에 대한 서명. 즉,  $m$ 에 대한 서명이 아님
- 또 한 가지의 (절차적) Issue:
  - TLS에 사용되는 전자 서명은 (학술적 정의와 다르게) 가변길이 메시지를 허용하지 않음
  - Labeling을 하기 위해서는 규격이나 표준을 바꿔야 함
    - 역시 IETF 표준화가 진행되고 있는 방향입니다.

Thanks 😊

Any Questions?