KEM 결합의 암호학적 안전성

손 용 하
성신여자대학교 융합보안공학과

2025.07.16 KpqC 연구단 워크숍

Background: Hybrid Mode

- Classical (Traditional) Crypto
  - Quantum Computing에 안전하지 않지만
  - 많은 Legacy 시스템에서 사용되고 있으며
  - Classical Computer에 대한 안전성은 비교적 Stable
- Post Quantum Crypto
  - 이론적인 안전성은 어느정도 Consensus가 있는 듯 하지만,
  - 실질적인 안전성(e.g. Side-Channel, Concrete Attack)은 비교적 Unstable
  - (Ring-variants 등에 대한) 밝혀지지 않은 취약점이 존재할 가능성도 무시할 수 없음

Background: Hybrid Mode

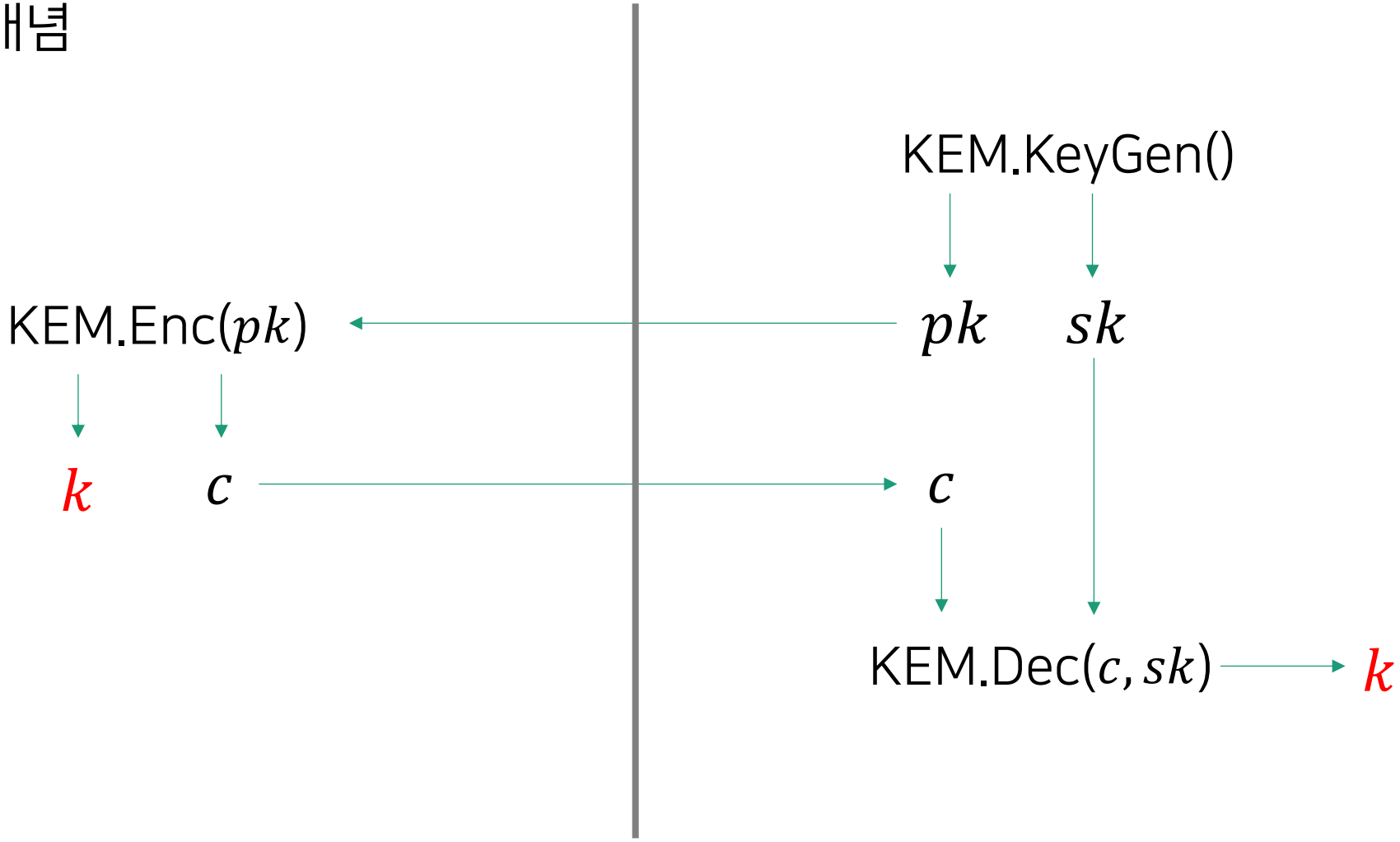
- 약간의 비효율을 감수하고서라도, Classical 과 PQC 를 동시에 사용할 것을 제안
  - CNSA 2.0 (NSA에서 발행하는 암호화 알고리즘 모음) 에서 하이브리드를 사용할 것을 강력히 권장
  - TLS 1.3 에 Hybrid 키 교환 및 Hybrid 서명을 지원하는 표준 (활발히) 작성 중

Questions

- 구체적으로 어떻게 동시에 사용하는가? = Construction
- 어떤 안전성을 요구하는가? = Security

Brief Recaps

- KEM 기본 개념



Brief Recaps

- IND-CPA Security of KEM
  - 공개키 pk, 암호문 c로부터 내부 k 에 대한 정보를 알 수 없다.
  - (pk, c) 가 주어진 상황에서,
  - 실제로 c에 대응하는 k_0 v.s. $\mathcal{K}$ 에서 무작위로 선택된 k_1 가 구별 불가능

$(pk, sk) \leftarrow \text{KEM.KeyGen}()$

$(k_0, c^*) \leftarrow \text{KEM.Enc}(pk)$ & $k_1 \leftarrow \mathcal{K}$

(pk, k_b, c^*) →  …… b is 0 !

Brief Recaps

- IND-CCA Security of KEM
  - 공개키 pk, 암호문 c로부터 내부 k 에 대한 정보를 알 수 없다.
  - (pk, c) 그리고 Decryption Oracle이 주어진 상황에서,
  - 실제로 c에 대응하는 k_0 v.s. $\mathcal{K}$ 에서 무작위로 선택된 k_1 가 구별 불가능

$(pk, sk) \leftarrow \text{KEM.KeyGen}()$

$(k_0, c^*) \leftarrow \text{KEM.Enc}(pk)$ & $k_1 \leftarrow \mathcal{K}$

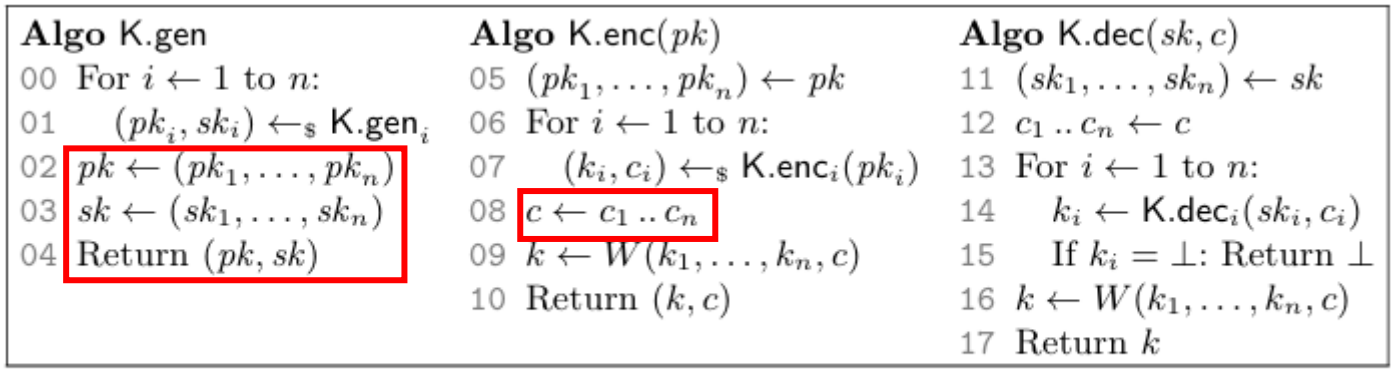
Oracle $\text{Dec}(c)$
if $c = c^*$ **then**
  return $\perp$
$k \leftarrow \text{KEM.Dec}(c, sk)$
return k

(pk, k_b, c^*)

…… b is 0 !

KEM Combiners [GHP18, PKC]

- "CCA" KEM들의 구체적인 결합과 안전성 분석
  - WANT: 단 1개의 component라도 CCA이 만족되면, 결과물도 CCA !
    - Recall: Classical + PQC 결합에서, 하나라도 안전하면 전체의 안전성이 보장되어야 함
  - pk_i, sk_i 및 c_i들은 단순히 Concat (Parallel combine)
    - 이 부분은 지금까지도 특별히 개선된 사항은 없는 것 같음



Algo K.gen
00 For $i \leftarrow 1$ to n:
01 $(pk_i, sk_i) \leftarrow_{\$} \mathsf{K.gen}_i$
02 $pk \leftarrow (pk_1, \ldots, pk_n)$
03 $sk \leftarrow (sk_1, \ldots, sk_n)$
04 Return (pk, sk)

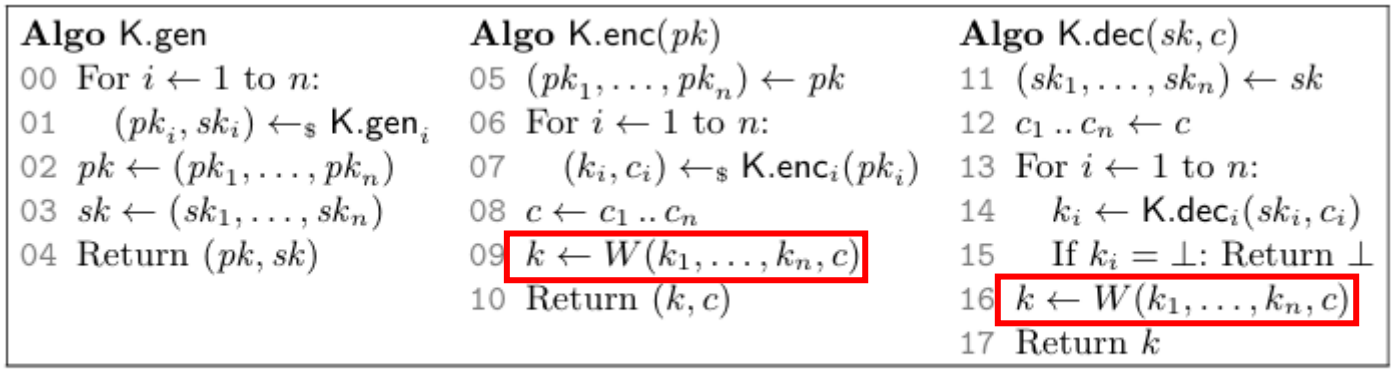
Algo K.enc(pk)
05 $(pk_1, \ldots, pk_n) \leftarrow pk$
06 For $i \leftarrow 1$ to n:
07 $(k_i, c_i) \leftarrow_{\$} \mathsf{K.enc}_i(pk_i)$
08 $c \leftarrow c_1 \,..\, c_n$
09 $k \leftarrow W(k_1, \ldots, k_n, c)$
10 Return (k, c)

Algo K.dec(sk, c)
11 $(sk_1, \ldots, sk_n) \leftarrow sk$
12 $c_1 \,..\, c_n \leftarrow c$
13 For $i \leftarrow 1$ to n:
14 $k_i \leftarrow \mathsf{K.dec}_i(sk_i, c_i)$
15 If $k_i = \perp$: Return $\perp$
16 $k \leftarrow W(k_1, \ldots, k_n, c)$
17 Return k

Fig. 4. Parallel KEM combiner, defined with respect to some core function W.

KEM Combiners [GHP18, PKC]

- "CCA" KEM들의 구체적인 결합과 안전성 분석
  - WANT: 단 1개의 component라도 CCA이 만족되면, 결과물도 CCA !
  - **핵심 질문**: 각 KEM의 결과물인 k_i를 어떻게 조합할 것인가?



Algo K.gen
00 For $i \leftarrow 1$ to n:
01 $(pk_i, sk_i) \leftarrow_{\$} \mathsf{K.gen}_i$
02 $pk \leftarrow (pk_1, \ldots, pk_n)$
03 $sk \leftarrow (sk_1, \ldots, sk_n)$
04 Return (pk, sk)

Algo K.enc(pk)
05 $(pk_1, \ldots, pk_n) \leftarrow pk$
06 For $i \leftarrow 1$ to n:
07 $(k_i, c_i) \leftarrow_{\$} \mathsf{K.enc}_i(pk_i)$
08 $c \leftarrow c_1 \,..\, c_n$
09 $k \leftarrow W(k_1, \ldots, k_n, c)$
10 Return (k, c)

Algo K.dec(sk, c)
11 $(sk_1, \ldots, sk_n) \leftarrow sk$
12 $c_1 \,..\, c_n \leftarrow c$
13 For $i \leftarrow 1$ to n:
14 $k_i \leftarrow \mathsf{K.dec}_i(sk_i, c_i)$
15 If $k_i = \perp$: Return $\perp$
16 $k \leftarrow W(k_1, \ldots, k_n, c)$
17 Return k

Fig. 4. Parallel KEM combiner, defined with respect to some core function W.

KEM Combiner [GHP18, PKC]

WANT: 단 1개의 component라도 CCA이면, 결과물도 CCA

핵심 질문: 각 KEM 의 결과물인 k_i 를 어떻게 조합할 것인지?

Takeaway 1 : $k_c := H(k_1||k_2)$ 는 IND-CCA secure 하지 않다. (H가 RO 이더라도)

$\because$ (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

$\Rightarrow$ Dec Oracle의 대답으로 쉽게 구별 가능

$(sk_1, pk_1) \leftarrow KEM_1.KeyGen()$
$(sk_2, pk_2) \leftarrow KEM_2.KeyGen()$

$(k_1^*, c_1^*) \leftarrow KEM_1.Enc(pk_1)$
$(k_2^*, c_2^*) \leftarrow KEM_2.Enc(pk_2)$
$k_0 \leftarrow H(k_1^*||k_2^*)$

v.s. $k_1 \leftarrow \mathcal{K}$

$(pk_1||pk_2, k_b, c_1^*||c_2^*)$

Please decrypt $c_1^*||c_2$

k_{dec}

Oracle Dec(c)
if $c = c^*$ **then**
  return $\perp$
$k \leftarrow$ KEM.Dec(c, sk)
return k

If $k_{dec} = k_b$, $b = 0$
Else, $b = 1$

KEM Combiner [GHP18, PKC]

WANT: 단 1개의 component라도 CCA이면, 결과물도 CCA

핵심 질문: 각 KEM 의 결과물인 k_i 를 어떻게 조합할 것인지?

- Direct remedy: k_c 생성할 때 각 KEM의 암호문 c_i도 같이 넣자

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

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

Takeaway 2

Thm. H가 Random Oracle이고, 구성 KEM 중 하나라도 IND-CCA이면

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

KEM Combiner [GHP18, PKC]

WANT: 단 1개의 component라도 CCA이면, 결과물도 CCA

핵심 질문: 각 KEM 의 결과물인 k_i 를 어떻게 조합할 것인지?

이외에도 가정을 약화시키면서 좀 더 복잡한 Construction들도 제안,

그러나 현재 Real World 적용에 고려되는 Construction은 아닌 듯.

$$k_c := F\left(\pi_{k_2} \circ \pi_{k_1}(0) || c_1 || c_2\right)$$

For some block cipher π

- F: PRF / π: Ideal Cipher 이면 IND-CCA
- F: RO / π: PRP 이면 IND-CCA

$$k_c := F(k_1 || c_2) \oplus F(k_2 || c_1)$$

- F가 PRF이면 IND-CCA
  - F: Random Oracle보다 약한 가정

X-Wing: 2024/039

- KEM Combiners의 방법을 기반으로 하여, ML-KEM / X25519 의 특성을 활용한 최적화

IACR Communications in Cryptology
ISSN 3006-5496, Vol. 1, No. 1, 22 pages.

https://doi.org/10.62056/a3qj89n4e

Check for updates

X-Wing

The Hybrid KEM You've Been Looking For

Manuel Barbosa[1,2,3], Deirdre Connolly[6], João Diogo Duarte[1,2], Aaron Kaiser[3], Peter Schwabe[3,4], Karolin Varner[3,5] and Bas Westerbaan[7]

CRYSTALS-KYBER

Algorithm Specifications And Supporting Documentation

(version 3.02)

Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, Damien Stehlé

August 4, 2021

Core Observation of X-Wing

- $k_c = H(k_1 || k_2)$는 같은 Key를 생성하는 다른 Ciphertext를 찾을 수 있다면 안전하지 않았음
  - Ciphertext Second-Preimage Resistance (C2PRI)

    "(sk, pk, k, c) 가 주어지더라도, $Dec(c', sk) = k$ 인 또 다른 c' 를 찾는 것이 어려움"

Definition 7 (C2PRI). We define the advantage function of an adversary $\mathcal{A}$ against KEM second preimage resistance as:

$$\mathrm{Adv}^{\mathrm{C2PRI}}_{\mathrm{KEM},\mathcal{A}} = \Pr\left[\mathsf{Dec}(c, sk) = k^* \wedge c \neq c^* \;\middle|\; \begin{array}{l}(sk,pk)\leftarrow\$\mathsf{KeyGen}()\\(k^*,c^*)\leftarrow\$\mathsf{Enc}(pk)\\c\leftarrow\mathcal{A}(sk,pk,k^*,c^*)\end{array}\right].$$

Note: sk가 없으면, IND-CCA로부터 유도되는 성질임

- **Theorem (Informal)** : IND-CCA KEM_1, C2PRI KEM_2를 $H(k_1 || k_2)$로 조합하면 IND-CCA

Theorem (Informal) : ML-KEM은 C2PRI 를 만족한다

Why ML-KEM is C2PRI?

- **Theorem (Informal)** : ML-KEM은 C2PRI 를 만족한다.
  - Lattice-based 여서 그런 건 전혀 아니고, FO-transformation을 사용해서 그런 것
  - 더 정확히는, 최종 Key (K or $\overline{K}$)를 도출할 때 Random Oracle G 와 J 를 사용하기 때문
    - 따라서 일반적으로 FO-transformation을 사용하는 모든 CCA KEM으로 확장 가능

ML-KEM-768.Dec$(sk \in \{0,1\}^{768k+96}, c \in \{0,1\}^{256(d_u k+d_v)})$

$sk_{PKE} \leftarrow sk[0 : 384k]$
$pk_{PKE} \leftarrow sk[384k : 768k + 32]$
$h \leftarrow sk[768k + 32 : 768k + 64]$
$z \leftarrow sk[768k + 64 : 768k + 96]$
$m' \leftarrow \text{PKE.Dec}(sk_{PKE}, c)$
$(K, r') \leftarrow G(m'\|h)$
$\overline{K} \leftarrow J(z\|c, 32)$
$c' \leftarrow \text{PKE.Enc}(pk_{PKE}, m', r')$
if $c \neq c'$ **then**
  $K \leftarrow \overline{K}$
return K

Figure 11: ML-KEM-768 decapsulation [NIS23]

X-Wing

- **Cororally**: DH KEX가 IND-CCA인 상황에선, C2PRI KEM와 $H(k_1||k_2)$로 조합하면 IND-CCA
- KEM_1이 IND-CCA, KEM_2가 안전하지 않은 경우를 커버한 것.

- KEM_1이 안전하지 않고, KEM_2 가 IND-CCA인 경우 증명도 필요
- 즉, DH KEX 가 안전하지 않아도 IND-CCA KEM 을 $H(k_X||k_M)$로 조합하면 IND-CCA ?
  - 하지만 아쉽게도 이것은 False ☹

    ∵ DH KEX가 C2PRI를 만족하지 않아서 c_1은 생략할 수 없다.

Takeaway 3 (X-Wing): DH KEX, ML-KEM 은 다음과 같이 조합해도 OK

$$k_c := H(k_X||k_M||c_X)$$

X-Wing 최종본

X-Wing private key (2464 bytes):

ML-KEM-768 private key (2400 bytes)	X25519 private key (32 bytes)

X-Wing public key (1216 bytes):

ML-KEM-768 public key (1184 bytes)	X25519 public key (32 bytes)

X-Wing ciphertext (1120 bytes):

ML-KEM-768 ciphertext (1088 bytes)	X25519 ciphertext (32 bytes)

X-Wing shared key (32 bytes):

SHA3-256 (

\./ /^\ (6 bytes)	ML-KEM-768 shared key (32 bytes)	X25519 shared key (32 bytes)	X25519 ciphertext (32 bytes)

)

The initial X-Wing label is encoded as 6-byte ASCII string "\.//^\".

1. Multi-target failure attack 을 방어하기 위해, pk_X를 추가

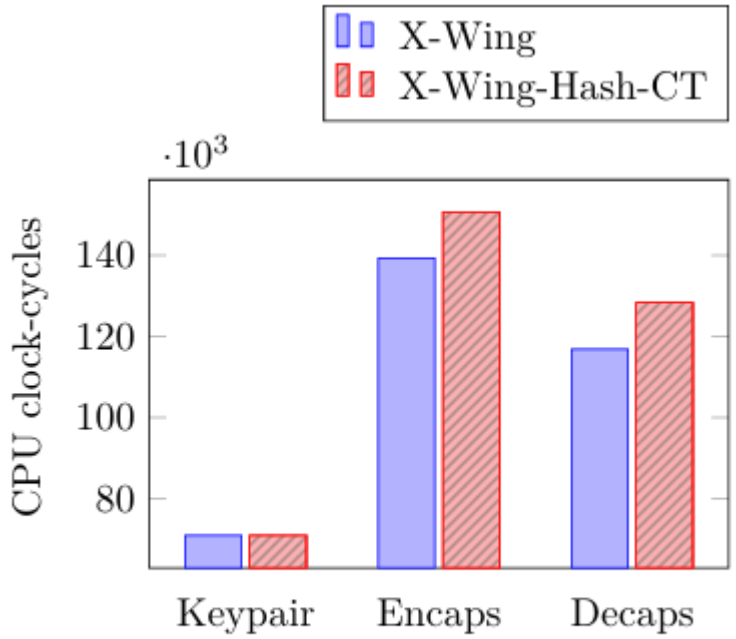
$$k_c := H(k_M||k_X||c_X||pk_X)$$

- 해당 공격은 IND-CCA notion으로 capture 되는 것은 아닌 것 같습니다.
  - pk_X가 없어도 증명 가능(한 것으로 보임)
- 다만, pk_X 없애봐야 성능에 영향이 매우 미미 …
  - SHA3-256은 256-block으로 동작

2. Dec마다 $pk_X = g^{sk}$ 계산하지 않도록 Private Key에 pk_X 별도로 저장(하는 듯 함)
3. 굳이 ML-KEM-768를 쓰는 이유?
   - Previous Work들이 보수적인 이유로 이것을 택함

X-Wing: 성능

- X-Wing-Hash-CT (ML-KEM-768 의 Ciphertext도 추가해서 Hash하는 방식) 과 비교
  - **Hash 입력이 짧아진 만큼의 성능 개선**



(a) Benchmarks for X-Wing and X-Wing-Hash-CT AVX2 implementations.

	KeyGen	Encaps	Decaps
X-Wing	70942	139374	116930
X-Wing-Hash-CT	70976	150708	128336
Ratio	-	92%	91%
Kyber	52732	67624	53156

CPU clock cycles (AVX2)

Side-Note. TLS 1.3 의 Hybrid KEM

- TLS 1.3 에서의 Hybrid KEM은 $k_c = H(k_1 || k_2)$으로 진행 중

하지만,

- TLS 1.3에서는 주고 받은 모든 메시지를

  Derived Key로 hash하여 검증하는 부분이 있음

  - “주고 받은 모든 메시지”에는 KEM 암호문들도 포함

    ⇒ KEM 암호문 검증을 하는 것으로 볼 수 있음

- 안전성이 증명된 방식인지는 모르겠습니다.

Workgroup:	Network Working Group
Internet-Draft:	draft-ietf-tls-hybrid-design-12
Published:	14 January 2025
Intended Status:	Informational
Expires:	18 July 2025

Authors:	D. Stebila	S. Fluhrer	S. Gueron
	University of Waterloo	*Cisco Systems*	*U. Haifa & Meta*

Hybrid key exchange in TLS 1.3

...

3.3. Shared secret calculation

Here we also take a simple "concatenation approach": the two shared secrets are concatenated together and used as the shared secret in the existing TLS 1.3 key schedule. Again, we do not add any additional structure (length fields) in the concatenation procedure: for both the traditional groups and post quantum KEMs, the shared secret output length is fixed for a specific elliptic curve or parameter set.¶

In other words, if the `NamedGroup` is `MyECDHMyPQKEM`, the shared secret is calculated as

```
concatenated_shared_secret = MyECDH.shared_secret || MyPQKEM.shared_secret
```

Thank You !