MAX PLANCK INSTITUTE
FOR SECURITY AND PRIVACY

X-Wing

Manuel Barbosa, Deirdre Connolly, João Diogo Duarte, Aaron Kaiser, **Peter Schwabe**, Karolin Varner, Bas Westerbaan

December 20, 2024

Key agreement since 1976

New Directions in Cryptography

Invited Paper

WHITFIELD DIFFIE AND MARTIN E. HELLMAN, MEMBER, IEEE

***Abstract*—Two kinds of contemporary developments in cryptography are examined. Widening applications of teleprocessing have given rise to a need for new types of cryptographic systems, which minimize the need for secure key distribution channels and supply the equivalent of a written signature. This paper suggests ways to solve these currently open problems. It also discusses how the theories of communication and computation are beginning to provide the tools to solve cryptographic problems of long standing.**

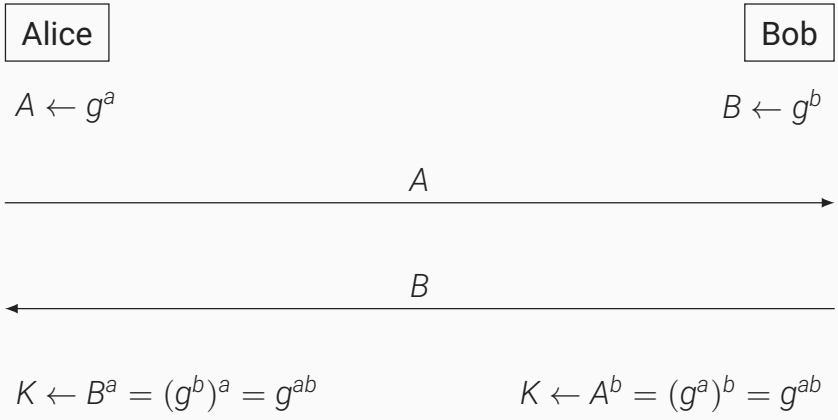
I. Introduction

WE STAND TODAY on the brink of a revolution in cryptography. The development of cheap digital hardware has freed it from the design limitations of mechanical computing and brought the cost of high grade cryptographic devices down to where they can be used in such commercial applications as remote cash dispensers and computer terminals. In turn, such applications create a need for new types of cryptographic systems which minimize the necessity of secure key distribution channels and supply the equivalent of a written signature. At the same time, theoretical developments in information theory and computer science show promise of providing provably secure cryptosystems, changing this ancient art into a science.

The best known cryptographic problem is that of privacy: preventing the unauthorized extraction of information from communications over an insecure channel. In order to use cryptography to insure privacy, however, it is currently necessary for the communicating parties to share a key which is known to no one else. This is done by sending the key in advance over some secure channel such as private courier or registered mail. A private conversation between two people with no prior acquaintance is a common occurrence in business, however, and it is unrealistic to expect initial business contacts to be postponed long enough for keys to be transmitted by some physical means. The cost and delay imposed by this key distribution problem is a major barrier to the transfer of business communications to large teleprocessing networks.

Section III proposes two approaches to transmitting keying information over public (i.e., insecure) channels without compromising the security of the system. In a *public key cryptosystem* enciphering and deciphering are governed by distinct keys, E and D, such that computing D from E is computationally infeasible (e.g., requiring 10^{100} instructions). The enciphering key E can thus be publicly disclosed without compromising the deciphering key D. Each user of the network can, therefore, place his enciphering key in a public directory. This enables any user of the system to send a message to any other user enci-

Key agreement since 1976

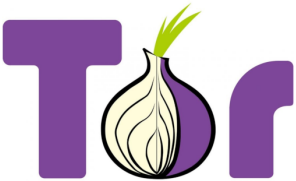


DH today


Nitrokey
secure your digital life


WIREGUARD
FAST, MODERN, SECURE VPN TUNNEL


zoom


Tor

DH today

- Use elliptic-curve version (ECDH)
- Most protocols: X25519 (Bernstein, 2016)

DH today

- Use elliptic-curve version (ECDH)
- Most protocols: X25519 (Bernstein, 2016)
- DH not as a “protocol” but as a building block
- In TLS: use signatures for authentication
- In WireGuard, Noise, Signal: DH also for authentication

Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*

Peter W. Shor†

Abstract

A digital computer is generally believed to be an efficient universal computing device; that is, it is believed able to simulate any physical computing device with an increase in computation time by at most a polynomial factor. This may not be true when quantum mechanics is taken into consideration. This paper considers factoring integers and finding discrete logarithms, two problems which are generally thought to be hard on a classical computer and which have been used as the basis of several proposed cryptosystems. Efficient randomized algorithms are given for these two problems on a hypothetical quantum computer. These algorithms take a number of steps polynomial in the input size, e.g., the number of digits of the integer to be factored.

Going post-quantum: NIST PQC

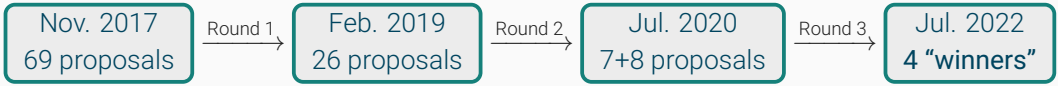
Count of Problem Category	Column Labels		
Row Labels	Key Exchange	Signature	Grand Total
?	1		1
Braids	1	1	2
Chebychev	1		1
Codes	19	5	24
Finite Automata	1	1	2
Hash		4	4
Hypercomplex Numbers	1		1
Isogeny	1		1
Lattice	24	4	28
Mult. Var	6	7	13
Rand. walk	1		1
RSA	1	1	2
Grand Total	**57**	**23**	**80**

4 31 27

Overview tweeted by Jacob Alperin-Sheriff on Dec 4, 2017.

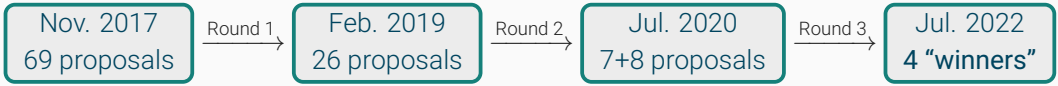
NIST PQC – how it went

NIST PQC



NIST PQC – how it went

NIST PQC



*“The public-key encryption and key-establishment algorithm that will be standardized is **CRYSTALS-KYBER**. The digital signatures that will be standardized are CRYSTALS-Dilithium, FALCON, and SPHINCS$^+$. While there are multiple signature algorithms selected, NIST recommends CRYSTALS-Dilithium as the primary algorithm to be implemented”*

—NIST IR 8413-upd1

The one-slide summary of ML-KEM

Lattice-based encryption K-PKE

- Arithmetic in $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^n + 1)$ with $q = 3329, n = 256$
- Computations of the form $As + e$ with $A \in \mathcal{R}_q^{k \times k}$ and $s, e \in \mathcal{R}_q^3$
- Security reduction from variant of Module-Learning-with-Errors (MLWE)

The one-slide summary of ML-KEM

Lattice-based encryption K-PKE

- Arithmetic in $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^n + 1)$ with $q = 3329, n = 256$
- Computations of the form $As + e$ with $A \in \mathcal{R}_q^{k \times k}$ and $s, e \in \mathcal{R}_q^3$
- Security reduction from variant of Module-Learning-with-Errors (MLWE)

Fujisaki-Okamoto Transform

- Required to achieve active (IND-CCA) security
- Enforce honestly generated ciphertexts
- Encapsulation generates all randomness as PRF(H(m))
- Decapsulation re-encrypts and compares ciphertexts

The one-slide summary of ML-KEM

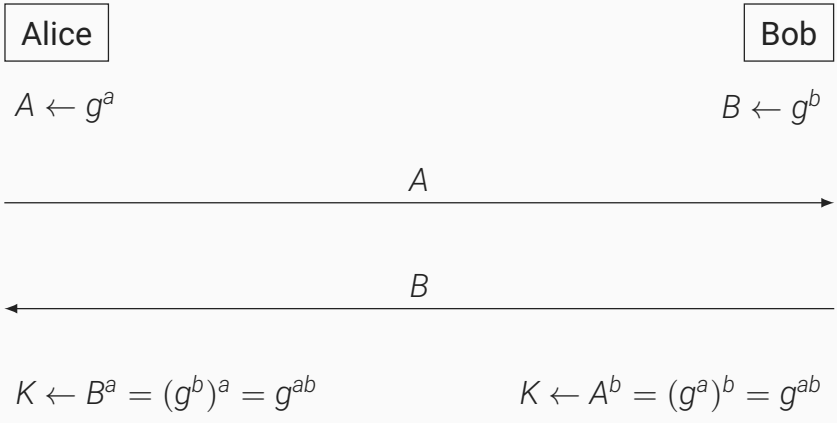
ML-KEM parameter sets

- ML-KEM-512
  - $k = 2$
  - NIST level 1 ($\approx$ AES-128)
  - PK: 800 B, CT: 768 B
- ML-KEM-768 (“recommended”)
  - $k = 3$
  - NIST level 3 ($\approx$ AES-192)
  - PK: 1184 B, CT: 1088 B
- ML-KEM-1024
  - $k = 4$
  - NIST level 5 ($\approx$ AES-256)
  - PK: 1568 B, CT: 1568 B

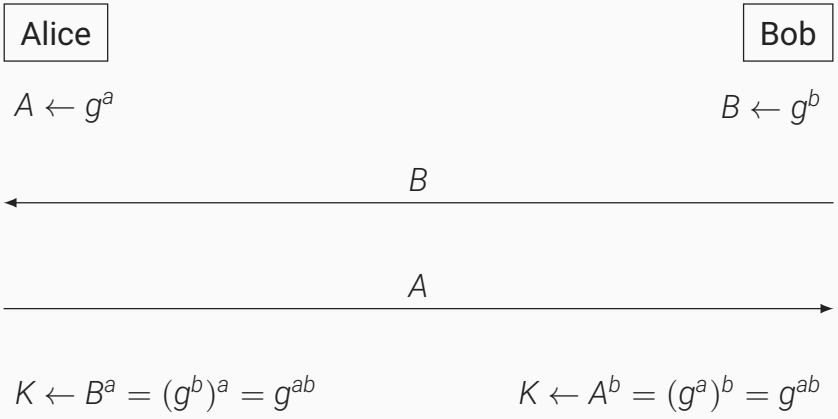
Now, switch all (EC)DH to ML-KEM and. . .

Mission accomplished – The world is safe again!

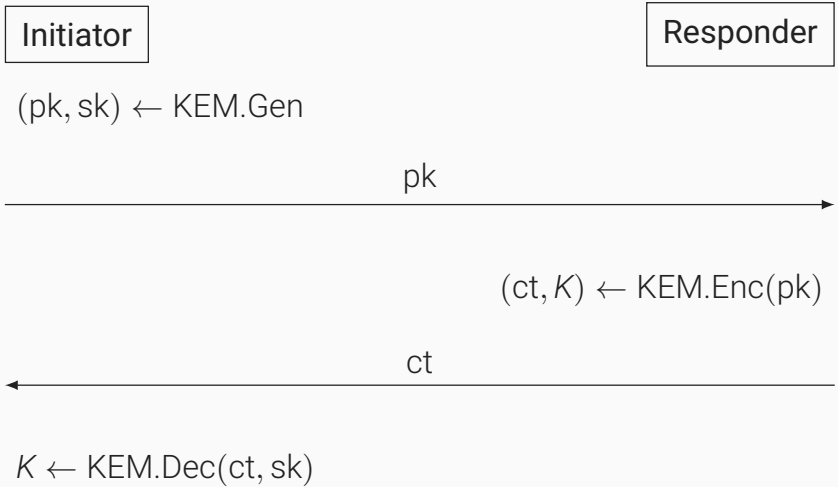
DH vs. KEMs


Alice
Bob
$A \leftarrow g^a$
$B \leftarrow g^b$
A
B
$K \leftarrow B^a = (g^b)^a = g^{ab}$
$K \leftarrow A^b = (g^a)^b = g^{ab}$

DH vs. KEMs


Alice
Bob
$A \leftarrow g^a$
$B \leftarrow g^b$
B
A
$K \leftarrow B^a = (g^b)^a = g^{ab}$
$K \leftarrow A^b = (g^a)^b = g^{ab}$

DH vs. KEMs


Initiator
Responder
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KEM.Gen}$
pk
$(\mathsf{ct}, K) \leftarrow \mathsf{KEM.Enc}(\mathsf{pk})$
ct
$K \leftarrow \mathsf{KEM.Dec}(\mathsf{ct}, \mathsf{sk})$

“Post-quantum schemes should only be used in combination with classical schemes (“hybrid”) if possible.”

—Recommendations by the BSI

https://www.bsi.bund.de/EN/Themen/Unternehmen-und-Organisationen/Informationen-und-Empfehlungen/Quantentechnologien-und-Post-Quanten-Kryptografie/quantentechnologien-und-post-quanten-kryptografie_node.html

Motivation for hybrid deployments

Don’t make systems less secure in the attempt to make them more secure against future quantum attackers!

Motivation for hybrid deployments

Don’t make systems less secure in the attempt to make them more secure against future quantum attackers!

- Cryptanalysis of PQ schemes is not as stable as for ECC
  - SIKE. . . (was deployed, **hybrid**, by Google and Cloudflare)
  - Late breaks of GeMSS and Rainbow

Motivation for hybrid deployments

Don’t make systems less secure in the attempt to make them more secure against future quantum attackers!

- Cryptanalysis of PQ schemes is not as stable as for ECC
  - SIKE. . . (was deployed, **hybrid**, by Google and Cloudflare)
  - Late breaks of GeMSS and Rainbow
- Implementation security of PQ schemes is not as mature as for ECC
  - SCA protection for ECC based on rich algebraic structure
  - For lattices: mostly masking + shuffling
  - Continued successful SCA against *protected* implementations
  - Compilers screwing with code in new ways (“Kyberslash”)

Isn’t hybrid to expensive?

Computational complexity

- Today’s systems use ECC
- ML-KEM is about as costly as ECC
- Hybrid costs about 2× slowdown

Isn’t hybrid to expensive?

Computational complexity

- Today’s systems use ECC
- ML-KEM is about as costly as ECC
- Hybrid costs about 2× slowdown
- Argument needs some more care with HW acceleration
- Anyway already have ECC
- Anyway will need PQC

Isn’t hybrid to expensive?

Computational complexity

- Today’s systems use ECC
- ML-KEM is about as costly as ECC
- Hybrid costs about 2× slowdown
- Argument needs some more care with HW acceleration
- Anyway already have ECC
- Anyway will need PQC

Sizes

- PQC cryptographic objects are much bigger than for ECC
- X25519 PK: 32 B
- Additing 32 Bytes to 1KB makes a small difference

Hybrid and the NIST competition

"NIST recognizes that some users may wish to deploy systems that use "hybrid modes," which combine post-quantum cryptographic algorithms with existing cryptographic algorithms (which may not be post-quantum). These "hybrid modes" are outside of the scope of this document, which is focused on post-quantum cryptographic algorithms only.

—NIST PQC Call for Proposals, 2016

Hybrid and the NIST competition

"NIST recognizes that some users may wish to deploy systems that use "hybrid modes," which combine post-quantum cryptographic algorithms with existing cryptographic algorithms (which may not be post-quantum). These "hybrid modes" are outside of the scope of this document, which is focused on post-quantum cryptographic algorithms only.

—NIST PQC Call for Proposals, 2016

Consequences

- Reduce complexity and probably discussions

Hybrid and the NIST competition

"NIST recognizes that some users may wish to deploy systems that use "hybrid modes," which combine post-quantum cryptographic algorithms with existing cryptographic algorithms (which may not be post-quantum). These "hybrid modes" are outside of the scope of this document, which is focused on post-quantum cryptographic algorithms only.

—NIST PQC Call for Proposals, 2016

Consequences

- Reduce complexity and probably discussions
- Non-mandatory hybrid deployment lead to other discussions:
  - Long discussions if Kyber512 meets level-1 security
  - No question if Kyber512+X25519 meets level-1 security

Hybrid and the NIST competition

"NIST recognizes that some users may wish to deploy systems that use "hybrid modes," which combine post-quantum cryptographic algorithms with existing cryptographic algorithms (which may not be post-quantum). These "hybrid modes" are outside of the scope of this document, which is focused on post-quantum cryptographic algorithms only.

—NIST PQC Call for Proposals, 2016

Consequences

- Reduce complexity and probably discussions
- Non-mandatory hybrid deployment lead to other discussions:
  - Long discussions if Kyber512 meets level-1 security
  - No question if Kyber512+X25519 meets level-1 security
- For targeted hybrid deployment, designs could have (and would have!) made other choices

Three approaches to hybrid KEMs

Protocol-level

- **+** Potential for optimal performance
- **+/-** Flexible choice of KEMs
- **-** High (per-protocol) analysis effort

Three approaches to hybrid KEMs

Protocol-level

- **+** Potential for optimal performance
- **+/-** Flexible choice of KEMs
- **-** High (per-protocol) analysis effort

Generic combiner

- **+** Low analysis effort (analyze *once*)
- **+/-** Flexible choice of KEMs
- **-** Computational overhead for being generic

Three approaches to hybrid KEMs

Protocol-level

- **+** Potential for optimal performance
- **+/-** Flexible choice of KEMs
- **-** High (per-protocol) analysis effort

Generic combiner

- **+** Low analysis effort (analyze *once*)
- **+/-** Flexible choice of KEMs
- **-** Computational overhead for being generic

Hybrid KEM

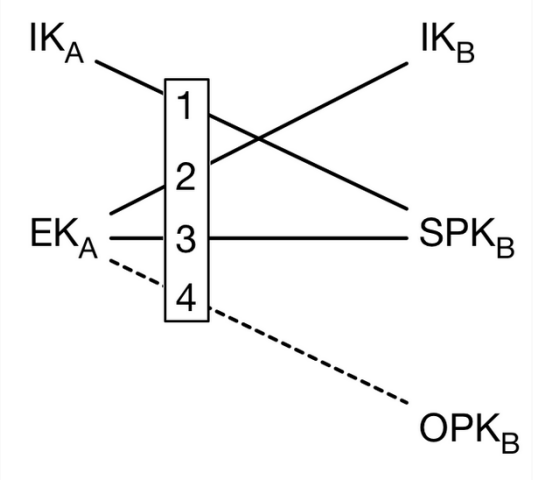
- **+** Low analysis effort (analyze *once*)
- **+** Close-to optimal performance
- **+/-** Cryptographically opinionated

The protocol-level approach

- CECPQ1, CECPQ2, PQXDH, PQ3, . . .

The protocol-level approach

- CECPQ1, CECPQ2, PQXDH, PQ3, . . .
- Let's look at X3DH → PQXDH

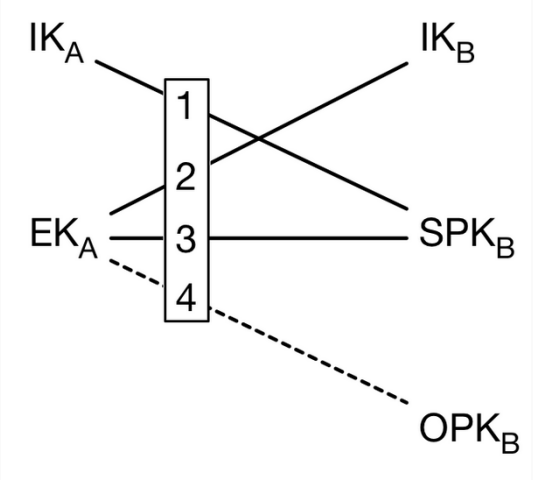


https://signal.org/docs/specifications/x3dh/

The protocol-level approach

- CECPQ1, CECPQ2, PQXDH, PQ3, ...
- Let's look at X3DH $\rightarrow$ PQXDH
- X3DH derives key as

$$sk = \mathrm{KDF}(\mathrm{DH}_1|\mathrm{DH}_2|\mathrm{DH}_3|\mathrm{DH}_4)$$



https://signal.org/docs/specifications/x3dh/

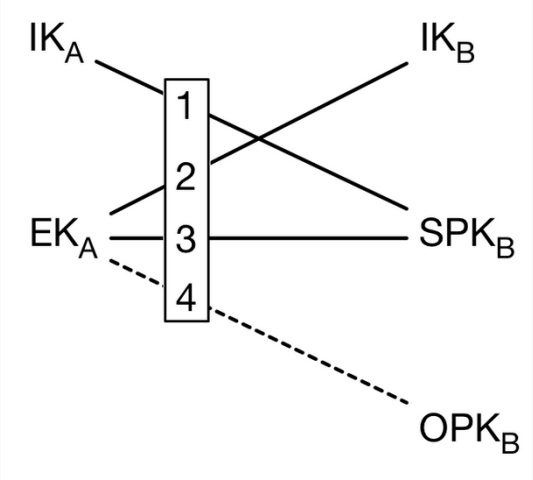
The protocol-level approach

- CECPQ1, CECPQ2, PQXDH, PQ3, ...
- Let's look at X3DH $\rightarrow$ PQXDH
- X3DH derives key as

$$sk = \mathrm{KDF}(\mathrm{DH}_1|\mathrm{DH}_2|\mathrm{DH}_3|\mathrm{DH}_4)$$

- PQXDH:
  - Additionally obtain SS from PQ KEM
  - Compute final shared key as

$$sk = \mathrm{KDF}(\mathrm{DH}_1|\mathrm{DH}_2|\mathrm{DH}_3|\mathrm{DH}_4|\mathrm{SS})$$



https://signal.org/docs/specifications/x3dh/

PQXDH security

- X3DH security:
  - Confidentiality
  - Mutual authentication
  - Forward secrecy
  - Deniability
- PQXDH: Additionally protect against harvest-now-decrypt-later (HNDL)

PQXDH security

Analysis by Bhargavan, Jacomme, Kiefer, Schmidt

https://cryspen.com/post/pqxdh/

Key confusion attack

- Attacker (malicious server) swaps KEM and DH keys
- KEM encapsulations to DH public key
- Resulting shared secret likely not secure
- Vice-versa, can downgrade DH security!

PQXDH security

Analysis by Bhargavan, Jacomme, Kiefer, Schmidt

https://cryspen.com/post/pqxdh/

Key confusion attack

- Attacker (malicious server) swaps KEM and DH keys
- KEM encapsulations to DH public key
- Resulting shared secret likely not secure
- Vice-versa, can downgrade DH security!
- Requires same length for KEM PK and DH PK

PQXDH security

Analysis by Bhargavan, Jacomme, Kiefer, Schmidt

https://cryspen.com/post/pqxdh/

KEM re-encapsulation attack

- Compromise one KEM private key
- MitM future exchanges of that user *even when they use a different KEM key!*
- Requires encapsulator to “control shared secret”

PQXDH security

Analysis by Bhargavan, Jacomme, Kiefer, Schmidt

https://cryspen.com/post/pqxdh/

KEM re-encapsulation attack

- Compromise one KEM private key
- MitM future exchanges of that user *even when they use a different KEM key!*
- Requires encapsulator to “control shared secret”
- Does not work with ML-KEM
- Requires PK-binding property not implied by IND-CCA!

The combiner approach

- Reminder: Two KEMs KEM_1 and KEM_2
- Want IND-CCA security, as long as *one* has IND-CCA security

The combiner approach

- Reminder: Two KEMs KEM_1 and KEM_2
- Want IND-CCA security, as long as *one* has IND-CCA security
- Simple idea: $\text{H}(ss_1, ss_2)$

The combiner approach

- Reminder: Two KEMs KEM_1 and KEM_2
- Want IND-CCA security, as long as *one* has IND-CCA security
- Simple idea: $\text{H}(ss_1, ss_2)$
- Problem: *Not* a robust IND-CCA KEM combiner

The combiner approach

- Reminder: Two KEMs KEM_1 and KEM_2
- Want IND-CCA security, as long as *one* has IND-CCA security
- Simple idea: $\text{H}(ss_1, ss_2)$
- Problem: *Not* a robust IND-CCA KEM combiner
- Attack:
  - Assume KEM_1 is broken
  - Assume given ct_1 it's easy to compute ct_1', s.t. $\text{KEM}_1.\text{Decaps}(ct_1) = \text{KEM}_1.\text{Decaps}(ct_1')$

The combiner approach

- Reminder: Two KEMs KEM_1 and KEM_2
- Want IND-CCA security, as long as *one* has IND-CCA security
- Simple idea: $\text{H}(ss_1, ss_2)$
- Problem: *Not* a robust IND-CCA KEM combiner
- Attack:
  - Assume KEM_1 is broken
  - Assume given ct_1 it's easy to compute ct_1', s.t. $\text{KEM}_1.\text{Decaps}(ct_1) = \text{KEM}_1.\text{Decaps}(ct_1')$
  - Challenge $(ct_1 || ct_2), K_c$
  - Adversary queries $K' = \text{Decaps}(ct_1' || ct_2)$, check if $K = K'$
- Solution: Use $\text{H}(ss_1, ss_2, ct_1, ct_2)$:

  Giacon, Heuer, Poettering. *KEM Combiners.* PKC 2018

The combiner approach

- Reminder: Two KEMs KEM_1 and KEM_2
- Want IND-CCA security, as long as *one* has IND-CCA security
- Simple idea: $\text{H}(ss_1, ss_2)$
- Problem: *Not* a robust IND-CCA KEM combiner
- Attack:
  - Assume KEM_1 is broken
  - Assume given ct_1 it's easy to compute ct_1', s.t. $\text{KEM}_1.\text{Decaps}(ct_1) = \text{KEM}_1.\text{Decaps}(ct_1')$
  - Challenge $(ct_1 || ct_2), K_c$
  - Adversary queries $K' = \text{Decaps}(ct_1' || ct_2)$, check if $K = K'$
- Solution: Use $\text{H}(ss_1, ss_2, ct_1, ct_2)$:

  Giacon, Heuer, Poettering. *KEM Combiners.* PKC 2018
- But, hang on, ECDH is not an IND-CCA KEM

The combiner approach

- Reminder: Two KEMs KEM_1 and KEM_2
- Want IND-CCA security, as long as *one* has IND-CCA security
- Simple idea: $\text{H}(ss_1, ss_2)$
- Problem: *Not* a robust IND-CCA KEM combiner
- Attack:
  - Assume KEM_1 is broken
  - Assume given ct_1 it's easy to compute ct_1', s.t. $\text{KEM}_1.\text{Decaps}(ct_1) = \text{KEM}_1.\text{Decaps}(ct_1')$
  - Challenge $(ct_1 || ct_2), K_c$
  - Adversary queries $K' = \text{Decaps}(ct_1' || ct_2)$, check if $K = K'$
- Solution: Use $\text{H}(ss_1, ss_2, ct_1, ct_2)$:
  Giacon, Heuer, Poettering. *KEM Combiners.* PKC 2018
- But, hang on, ECDH is not an IND-CCA KEM
- Solution: DHKEM (RFC 9180: Hybrid Public Key Encryption)

Motivation for X-Wing

When I get asked what KEM to use, I don't want to answer

"use Kyber768, but of course you should go for a hybrid solution together with some ECDH; look at this standard for a generic combiner, but if your protocol hashes full transcripts into the session key, it might be OK to not hash in the long ciphertext and gain some performance, except there is no formal proof of that."

Motivation for X-Wing

When I get asked what KEM to use, I don't want to answer

"use Kyber768, but of course you should go for a hybrid solution together with some ECDH; look at this standard for a generic combiner, but if your protocol hashes full transcripts into the session key, it might be OK to not hash in the long ciphertext and gain some performance, except there is no formal proof of that."

I want to answer

"use X-Wing."

X-Wing: the original idea

- Take the best™ ECDH: X25519

X-Wing: the original idea

- Take the best™ ECDH: X25519
- Take the best™ PQ-KEM: ML-KEM-768

X-Wing: the original idea

- Take the best™ ECDH: X25519
- Take the best™ PQ-KEM: ML-KEM-768
- Build KEM from ECDH

X-Wing: the original idea

- Take the best™ ECDH: X25519
- Take the best™ PQ-KEM: ML-KEM-768
- Build KEM from ECDH
- Use generic combiner

X-Wing: the original idea

- Take the best™ ECDH: X25519
- Take the best™ PQ-KEM: ML-KEM-768
- Build KEM from ECDH
- Use generic combiner
- Fix hash-function H = SHAKE-256

X-Wing: the original idea

- Take the best™ ECDH: X25519
- Take the best™ PQ-KEM: ML-KEM-768
- Build KEM from ECDH
- Use generic combiner
- Fix hash-function H = SHAKE-256
- Give the whole thing a cool name

X-Wing: the original idea

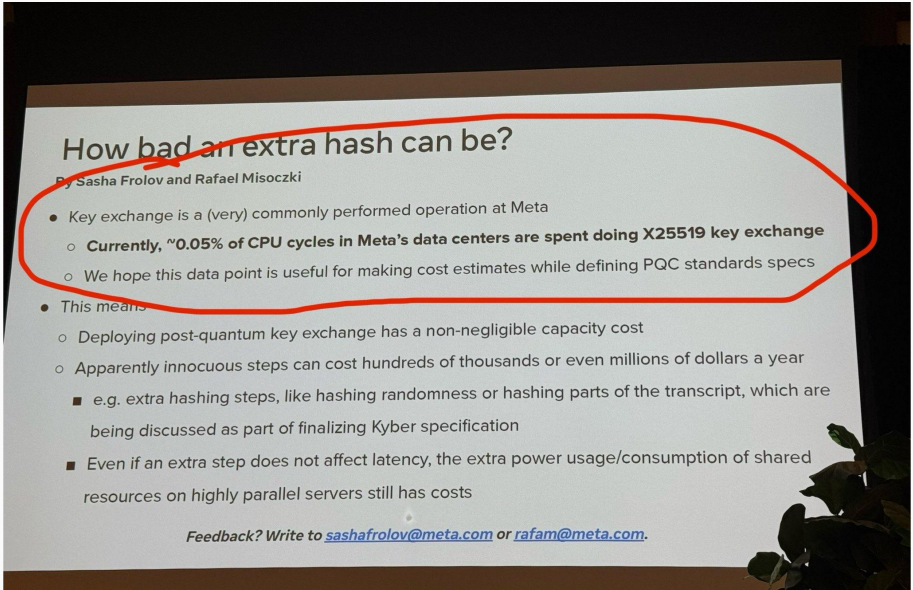
- Take the best™ ECDH: X25519
- Take the best™ PQ-KEM: ML-KEM-768
- Build KEM from ECDH
- Use generic combiner
- Fix hash-function H = SHAKE-256
- Give the whole thing a cool name
- Implement, advertise, done.

Get rid of the ciphertext hash!

Me: *"Would you use X-Wing in TLS?"*

Sophie Schmieg: *"Get rid of the ciphertext hash and we might."*

What's the deal of hashing 1 KB of data?



What's the deal of hashing 1 KB of data?

"Lastly, there is also a performance angle. If saving a single hash in TLS saves compute time worth millions of dollars/CO2 emissions/energy, then it's probably worth our collective time to review this single protocol by itself and remove unneeded hash function calls."

—Sophie Schmieg, Feb 21, 2024.

What if

- KEM is not broken in an *arbitrary* way

What if

- KEM is not broken in an *arbitrary* way
- but “only” broken to allow private-key recovery

What if

- KEM is not broken in an *arbitrary* way
- but “only” broken to allow private-key recovery

C2PRI security notion

$$\mathsf{Adv}^{\mathsf{C2PRI}}_{\mathsf{KEM},\mathcal{A}} = \Pr\left[\mathsf{Decaps}(c,\mathsf{sk}) = k^* \wedge c \neq c^* \;\middle|\; \begin{matrix}(\mathsf{sk},\mathsf{pk}) \leftarrow\!\$ \mathsf{Keygen}(\,)\\ (k^*,c^*) \leftarrow\!\$ \mathsf{Encaps}(\mathsf{pk})\\ c \leftarrow \mathcal{A}(\mathsf{sk},\mathsf{pk},k^*,c^*)\end{matrix}\right].$$

What if

- KEM is not broken in an *arbitrary* way
- but “only” broken to allow private-key recovery

C2PRI security notion

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

- Attacker needs to produce 2nd ciphertext preimage
- Attacker *is given the private key*

What if

- KEM is not broken in an *arbitrary* way
- but “only” broken to allow private-key recovery

C2PRI security notion

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

- Attacker needs to produce 2nd ciphertext preimage
- Attacker *is given the private key*
- Prove that ML-KEM is C2PRI secure
- Intuition: ML-KEM involves a bunch of hashes
- Proof models these hashes as random oracles

Putting it together: QSF

QSF (Quantum Superiority Fighter)

- KEM
- Nominal group G

Putting it together: QSF

Algorithm Keygen()
$(\mathsf{sk}_1, \mathsf{pk}_1) \leftarrow\$ \mathsf{KEM}.\mathsf{Keygen}()$
$\mathsf{sk}_2 \leftarrow\$ \varepsilon_h$
$\mathsf{pk}_2 \leftarrow \exp(g, \mathsf{sk}_2)$
$\mathsf{pk} \leftarrow (\mathsf{pk}_1, \mathsf{pk}_2)$
$\mathsf{sk} \leftarrow (\mathsf{sk}_1, \mathsf{sk}_2, \mathsf{pk}_2)$
return $(\mathsf{sk}, \mathsf{pk})$

Algorithm Encaps(pk)
$(\mathsf{pk}_1, \mathsf{pk}_2) \leftarrow \mathsf{pk}$
$k_1, c_1 \leftarrow\$ \mathsf{KEM}.\mathsf{Encaps}(\mathsf{pk}_1)$
$\mathsf{sk}_e \leftarrow\$ \varepsilon_h$
$c_2 \leftarrow \exp(g, \mathsf{sk}_e)$
$k_2 \leftarrow \exp(\mathsf{pk}_2, \mathsf{sk}_e)$
$k \leftarrow H(\mathsf{label}|k_1|k_2|c_2|\mathsf{pk}_2)$
$c \leftarrow (c_1, c_2)$
return (k, c)

Algorithm Decaps(c, sk)
$(\mathsf{sk}_1, \mathsf{sk}_2, \mathsf{pk}_2) \leftarrow \mathsf{sk}$
$(c_1, c_2) \leftarrow c$
$k_1 \leftarrow \mathsf{KEM}.\mathsf{Decaps}(c_1, \mathsf{sk}_1)$
$k_2 \leftarrow \exp(c_2, \mathsf{sk}_2)$
if $k_1 = \perp$ **then**
    return $\perp$
end if
$k \leftarrow H(\mathsf{label}|k_1|k_2|c_2|\mathsf{pk}_2)$
return k

Putting it together: QSF

QSF provides IND-CCA security, if

- SDH is hard in *G* and KEM is C2PRI secure (ROM), **or**
- KEM is an IND-CCA-secure KEM (standard model)

The final design

X-Wing private key (2464 bytes):

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

X-Wing public key (1216 bytes):

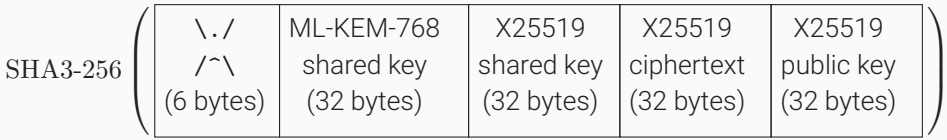
ML-KEM-768 public key (1184 bytes)	X25519 public key (32 bytes)

X-Wing ciphertext (1120 bytes):

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

The final design

X-Wing shared key (32 bytes):



The final design

Filippo Valsorda
@filippo@abyssdomain.expert

There's everything to love in

"X-Wing: The Hybrid KEM You've Been Looking For"
eprint.iacr.org/2024/039

- concrete choices!
- strong proofs
- easy to implement
- good performance
- "quantum superiority fighter"

\./
/^\

@durumcrustulum can I haz CCTV test vectors? <3

“Final” you say?

The label

Q: *“Could you please put the label at the end of the hash input?”*

A: Yes.

“Final” you say?

ML-KEM private-key format

- FIPS 203 allows two formats for private keys:
  - “Expanded format” (also includes PK and H(PK))
  - Seed format (64-byte randomness to generate keypair)
  - Different security with regards to MAL-BIND-K-CT and MAL-BIND-K-PK notions; see Cremers, Dax, Medinger. *Keeping Up with the KEMs: Stronger Security Notions for KEMs and automated analysis of KEM-based protocols.*
    Schmieg. *Unbindable Kemmy Schmidt: ML-KEM is neither MAL-BIND-K-CT nor MAL-BIND-K-PK.*

“Final” you say?

ML-KEM private-key format

- FIPS 203 allows two formats for private keys:
  - “Expanded format” (also includes PK and H(PK))
  - Seed format (64-byte randomness to generate keypair)
  - Different security with regards to MAL-BIND-K-CT and MAL-BIND-K-PK notions; see Cremers, Dax, Medinger. *Keeping Up with the KEMs: Stronger Security Notions for KEMs and automated analysis of KEM-based protocols.*
    Schmieg. *Unbindable Kemmy Schmidt: ML-KEM is neither MAL-BIND-K-CT nor MAL-BIND-K-PK.*
- In most scenarios this is a local choice
- In some scenarios it’s not, most notably HPKE

“Final” you say?

ML-KEM private-key format

- FIPS 203 allows two formats for private keys:
  - “Expanded format” (also includes PK and H(PK))
  - Seed format (64-byte randomness to generate keypair)
  - Different security with regards to MAL-BIND-K-CT and MAL-BIND-K-PK notions; see Cremers, Dax, Medinger. *Keeping Up with the KEMs: Stronger Security Notions for KEMs and automated analysis of KEM-based protocols.*
    Schmieg. *Unbindable Kemmy Schmidt: ML-KEM is neither MAL-BIND-K-CT nor MAL-BIND-K-PK.*
- In most scenarios this is a local choice
- In some scenarios it’s not, most notably HPKE
- Choice in X-Wing IETF draft:
  - Private key is 32-byte seed
  - Expand with SHAKE-256 to 32-byte X25519 private key and 64-byte ML-KEM private key in seed format

Learn more

https://x-wi.ng