MAX PLANCK INSTITUTE
FOR SECURITY AND PRIVACY

Formally verifying Kyber

Episode V: Machine-checked IND-CCA Security and Correctness of ML-KEM in EasyCrypt

José Bacelar Almeida, Santiago Arranz Olmos, Manuel Barbosa, Gilles Barthe, François Dupressoir, Benjamin Grégoire, Vincent Laporte, Jean-Christophe Léchenet, Cameron Low, Tiago Oliveira, Hugo Pacheco, **Miguel Quaresma**, Peter Schwabe, Pierre-Yves Strub

August 21, 2024

Summary

- Starting point: Episode IV

Summary

- Starting point: Episode IV
- Motivation for formally verifying cryptography

Summary

- Starting point: Episode IV
- Motivation for formally verifying cryptography
- From Kyber to the ML-KEM draft standard

Summary

- Starting point: Episode IV
- Motivation for formally verifying cryptography
- From Kyber to the ML-KEM draft standard
- ML-KEM correctness and security properties:

Summary

- Starting point: Episode IV
- Motivation for formally verifying cryptography
- From Kyber to the ML-KEM draft standard
- ML-KEM correctness and security properties:
    - Machine-checked proofs

Summary

- Starting point: Episode IV
- Motivation for formally verifying cryptography
- From Kyber to the ML-KEM draft standard
- ML-KEM correctness and security properties:
    - Machine-checked proofs
    - Optimized & verified implementations

Starting point: Episode IV

- **"Formally verifying Kyber Episode IV: Implementation Correctness"** [ABB+23]

Starting point: Episode IV

- **"Formally verifying Kyber Episode IV: Implementation Correctness"** [ABB+23]
  - Human auditable formal specification
  - Reference and vectorized/optimized implementations of Kyber-768
  - Computer-verified proofs of functional correctness using EasyCrypt

Starting point: Episode IV

- **"Formally verifying Kyber Episode IV: Implementation Correctness"** [ABB+23]
  - Human auditable formal specification
  - Reference and vectorized/optimized implementations of Kyber-768
  - Computer-verified proofs of functional correctness using EasyCrypt
- What is missing:

Starting point: Episode IV

- **"Formally verifying Kyber Episode IV: Implementation Correctness"** [ABB+23]
    - Human auditable formal specification
    - Reference and vectorized/optimized implementations of Kyber-768
    - Computer-verified proofs of functional correctness using EasyCrypt
- What is missing:
    - No analysis of cryptographic properties in EasyCrypt

Starting point: Episode IV

- **"Formally verifying Kyber Episode IV: Implementation Correctness"** [ABB+23]
  - Human auditable formal specification
  - Reference and vectorized/optimized implementations of Kyber-768
  - Computer-verified proofs of functional correctness using EasyCrypt
- What is missing:
  - No analysis of cryptographic properties in EasyCrypt
  - Targeted round 3 Kyber, not ML-KEM

Motivation: objectives

- Migrate Kyber implementations to ML-KEM

Motivation: objectives

- Migrate Kyber implementations to ML-KEM
  - Requires adapting proofs of functional correctness

Motivation: objectives

- Migrate Kyber implementations to ML-KEM
  - Requires adapting proofs of functional correctness
- Ensure our interpretation of FIPS 203 [Nat23] is secure
  - No compliance yet: ML-KEM was a draft and compliance requires certification
  - Consolidate changes introduced to Kyber (now ML-KEM) in a security proof (e.g. changes to the Fujisaki-Okamoto (FO) transform introduced by [HHK17])

Motivation: objectives

- Migrate Kyber implementations to ML-KEM
  - Requires adapting proofs of functional correctness
- Ensure our interpretation of FIPS 203 [Nat23] is secure
  - No compliance yet: ML-KEM was a draft and compliance requires certification
  - Consolidate changes introduced to Kyber (now ML-KEM) in a security proof (e.g. changes to the Fujisaki-Okamoto (FO) transform introduced by [HHK17])

Derive that our implementations are an IND-CCA secure KEM at the assembly level

Motivation: relevance

- Kyber 2017 [BDK+18]
    - Public key compression invalidated assumption in security proof
    - Tweaked FO transform [FO99, FO13] broke (security) proof in QROM

Motivation: relevance

- Early implementations of Kyber failed to perform implicit rejection

Motivation: relevance

- Early implementations of Kyber failed to perform implicit rejection
- Discrepancy between SABER's [DKR+20] implementation and specification of the FO transform [GMP22, Sec. 5.4]

Motivation: relevance

- Early implementations of Kyber failed to perform implicit rejection
- Discrepancy between SABER's [DKR$^+$20] implementation and specification of the FO transform [GMP22, Sec. 5.4]
- Overflow was found in the NTT implementations of Kyber targeting Cortex-M4: not triggered by test vectors

Motivation: relevance

- Early implementations of Kyber failed to perform implicit rejection
- Discrepancy between SABER's [DKR+20] implementation and specification of the FO transform [GMP22, Sec. 5.4]
- Overflow was found in the NTT implementations of Kyber targeting Cortex-M4: not triggered by test vectors
- KyberSlash: timing side-channel found when using `DIV` instruction [BBB+24]

Motivation: relevance

Goal: provide end to end formally verified implementations

Motivation: relevance

Goal: provide end to end formally verified implementations

- Security proofs
- Functional correctness proofs

Motivation: relevance

Goal: provide end to end formally verified implementations

- Security proofs
- Functional correctness proofs
- Link the two:
  - Ensure the functional specification matches the one in the security proof
  - Derive that our implementations are secure

Contributions

- Computer-verified proof of IND-CCA security of ML-KEM

Contributions

- Computer-verified proof of IND-CCA security of ML-KEM
  - Proof of IND-CCA security down to a variant of MLWE
  - Functional correctness proof of ML-KEM implementations in Jasmin [ABB+17, ABB+20]

Contributions

- Computer-verified proof of IND-CCA security of ML-KEM
    - Proof of IND-CCA security down to a variant of MLWE
    - Functional correctness proof of ML-KEM implementations in Jasmin [ABB+17, ABB+20]
- Proof of IND-CPA security with concrete bounds that consider low-level details:
    - compression
    - SHA3-based noise generation
    - SHA3-based rejection sampling

Contributions

- Computer-verified proof of IND-CCA security of ML-KEM
  - Proof of IND-CCA security down to a variant of MLWE
  - Functional correctness proof of ML-KEM implementations in Jasmin [ABB+17, ABB+20]
- Proof of IND-CPA security with concrete bounds that consider low-level details:
  - compression
  - SHA3-based noise generation
  - SHA3-based rejection sampling
- Formalization, in EasyCrypt, of the relevant FO transform variant introduced in [HHK17]

Contributions

- Computer-verified proof of IND-CCA security of ML-KEM
  - Proof of IND-CCA security down to a variant of MLWE
  - Functional correctness proof of ML-KEM implementations in Jasmin [ABB+17, ABB+20]
- Proof of IND-CPA security with concrete bounds that consider low-level details:
  - compression
  - SHA3-based noise generation
  - SHA3-based rejection sampling
- Formalization, in EasyCrypt, of the relevant FO transform variant introduced in [HHK17]

Concrete security bound for ML-KEM that considers low-level details

Kyber to ML-KEM - Kyber 2017

- Lattice-based PKE scheme based on Module-LWE [BDK+18]

Kyber to ML-KEM - Kyber 2017

- Lattice-based PKE scheme based on Module-LWE [BDK+18]
- Several optimizations: parameter choice + **public-key and ciphertext compression**

Kyber to ML-KEM - Kyber 2017

- Lattice-based PKE scheme based on Module-LWE [BDK+18]
- Several optimizations: parameter choice + **public-key and ciphertext compression**
- Correctness and security implications:
  - Correctness bound is hard to compute: only heuristic results
  - Public-key compression not contemplated in security proof
  - Tweaked FO transform: ciphertext hashing

Kyber to ML-KEM - Kyber Round 3

- KEM selected by NIST for standardization after round 3

Kyber to ML-KEM - Kyber Round 3

- KEM selected by NIST for standardization after round 3
- Differences from Kyber 2017 [BDK+18]:
  - Updated parameters (smaller prime q and noise parameter η)
  - No public key compression
  - Public keys transmitted in the NTT domain
  - Full specification of the scheme

Kyber to ML-KEM - Kyber Round 3

- KEM selected by NIST for standardization after round 3
- Differences from Kyber 2017 [BDK+18]:
  - Updated parameters (smaller prime q and noise parameter η)
  - No public key compression
  - Public keys transmitted in the NTT domain
  - Full specification of the scheme
- Tweaked FO transform still in use: ciphertext hashing

Algorithm 8 KYBER.CCAKEM.Enc(pk)

Input: Public key $pk \in \mathcal{B}^{12 \cdot k \cdot n/8+32}$
Output: Ciphertext $c \in \mathcal{B}^{d_u \cdot k \cdot n/8 + d_v \cdot n/8}$
Output: Shared key $K \in \mathcal{B}^*$

$m \leftarrow \mathcal{B}^{32}$
$m \leftarrow \mathrm{H}(m)$
$(\bar{K}, r) := \mathrm{G}(m \| \mathrm{H}(pk))$
$c :=$ KYBER.CPAPKE.Enc(pk, m, r)
$K :=$ KDF$(\bar{K} \| \mathrm{H}(c))$
return (c, K)

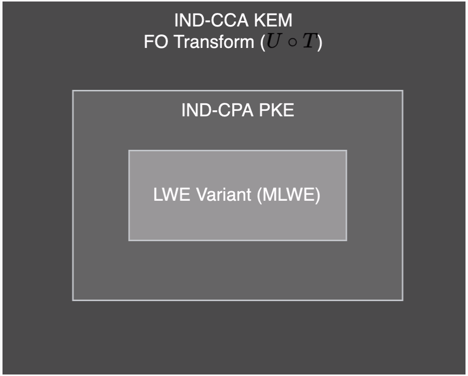
Algorithm 9 KYBER.CCAKEM.Dec(c, sk)

Input: Ciphertext $c \in \mathcal{B}^{d_u \cdot k \cdot n/8 + d_v \cdot n/8}$
Input: Secret key $sk \in \mathcal{B}^{24 \cdot k \cdot n/8+96}$
Output: Shared key $K \in \mathcal{B}^*$

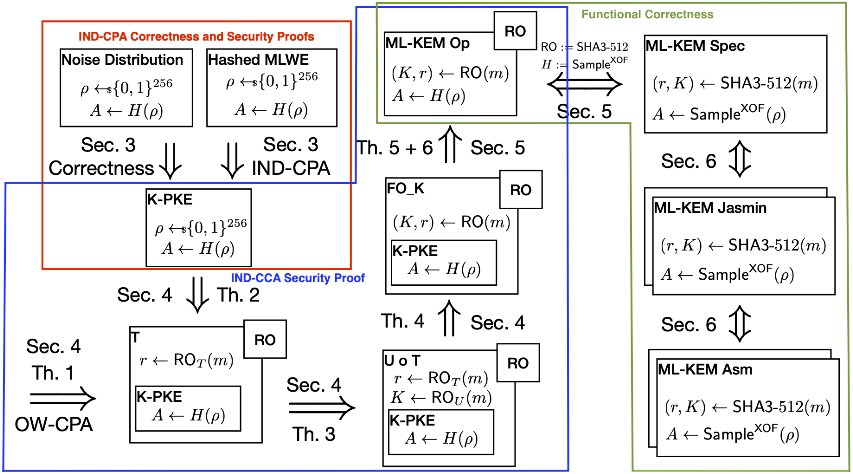
$pk := sk + 12 \cdot k \cdot n/8$
$h := sk + 24 \cdot k \cdot n/8 + 32 \in \mathcal{B}^{32}$
$z := sk + 24 \cdot k \cdot n/8 + 64$
$m' :=$ KYBER.CPAPKE.Dec(sk, c)
$(\bar{K}', r') := \mathrm{G}(m' \| h)$
$c' :=$ KYBER.CPAPKE.Enc(pk, m', r')
if $c = c'$ **then**
    return $K :=$ KDF$(\bar{K}' \| \mathrm{H}(c))$
else
    return $K :=$ KDF$(z \| \mathrm{H}(c))$
end if
return K

Kyber to ML-KEM - ML-KEM “draft”

- Formerly CRYSTALS-Kyber
- Typical design of post-quantum KEMs
- IND-CPA PKE scheme from variant of LWE
- IND-CCA KEM using FO transform [FO99, FO13, HHK17]

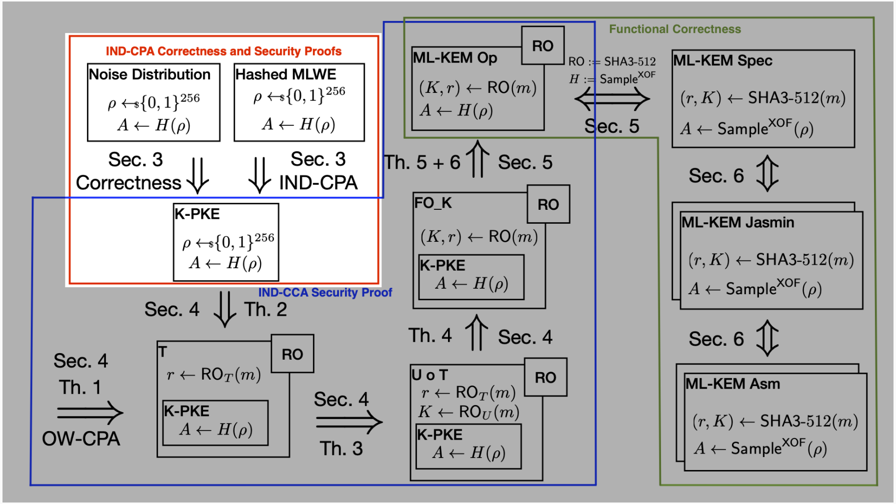


Overview



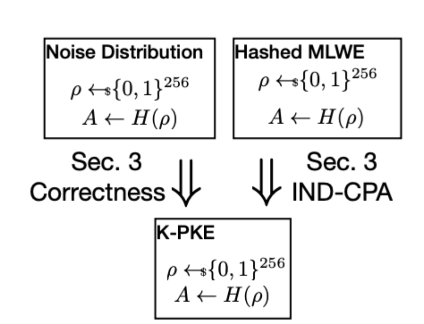
IND-CPA Correctness and Security Proofs
Noise Distribution
$\rho \leftarrow_{\$} \{0,1\}^{256}$
$A \leftarrow H(\rho)$
Hashed MLWE
$\rho \leftarrow_{\$} \{0,1\}^{256}$
$A \leftarrow H(\rho)$
Sec. 3
Correctness
Sec. 3
IND-CPA
K-PKE
$\rho \leftarrow_{\$} \{0,1\}^{256}$
$A \leftarrow H(\rho)$
IND-CCA Security Proof
Sec. 4
Th. 2
Sec. 4
Th. 1
OW-CPA
T
RO
$r \leftarrow \mathsf{RO}_T(m)$
K-PKE
$A \leftarrow H(\rho)$
Sec. 4
Th. 3
U o T
RO
$r \leftarrow \mathsf{RO}_T(m)$
$K \leftarrow \mathsf{RO}_U(m)$
K-PKE
$A \leftarrow H(\rho)$
Th. 4
Sec. 4
FO_K
RO
$(K, r) \leftarrow \mathsf{RO}(m)$
K-PKE
$A \leftarrow H(\rho)$
Th. 5 + 6
Sec. 5
ML-KEM Op
RO
$(K, r) \leftarrow \mathsf{RO}(m)$
$A \leftarrow H(\rho)$
RO := SHA3-512
H := Sample$^{\mathsf{XOF}}$
Sec. 5
Functional Correctness
ML-KEM Spec
$(r, K) \leftarrow$ SHA3-512(m)
$A \leftarrow$ Sample$^{\mathsf{XOF}}(\rho)$
Sec. 6
ML-KEM Jasmin
$(r, K) \leftarrow$ SHA3-512(m)
$A \leftarrow$ Sample$^{\mathsf{XOF}}(\rho)$
Sec. 6
ML-KEM Asm
$(r, K) \leftarrow$ SHA3-512(m)
$A \leftarrow$ Sample$^{\mathsf{XOF}}(\rho)$

Overview: IND-CPA construction



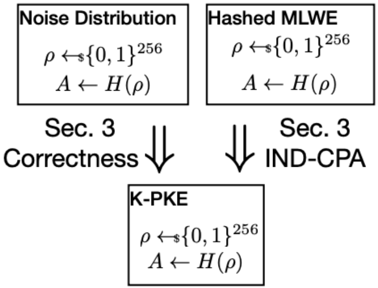
Overview: IND-CPA construction

- IND-CPA security and correctness proof



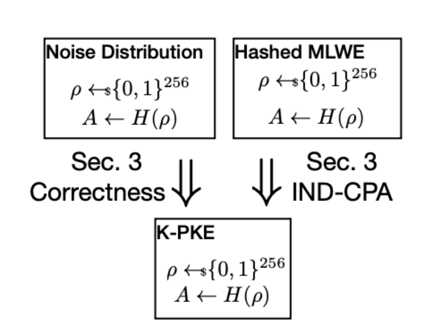
Overview: IND-CPA construction

- IND-CPA security and correctness proof
- K-PKE scheme underlies Kyber and ML-KEM

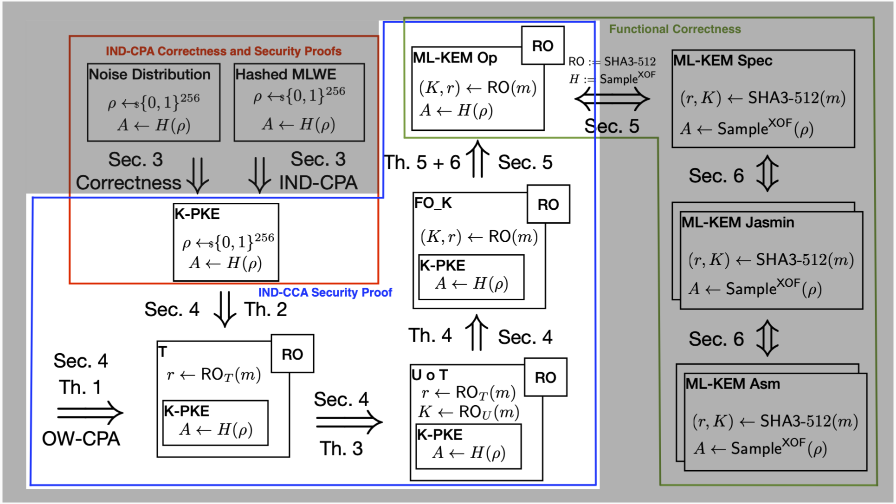


Overview: IND-CPA construction

- IND-CPA security and correctness proof
- K-PKE scheme underlies Kyber and ML-KEM
- Security proof under a variant of MLWE: Hashed MLWE
  - Replace sampling of matrix A with deterministic procedure H
- Correctness proof sets upper bound for a decryption failure

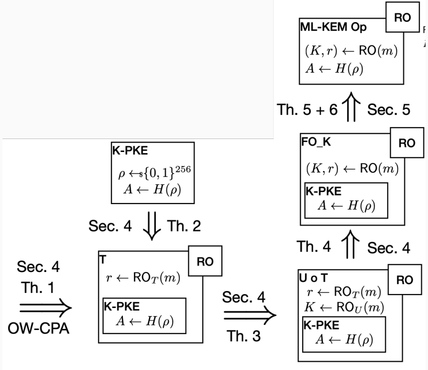


Overview: IND-CCA security

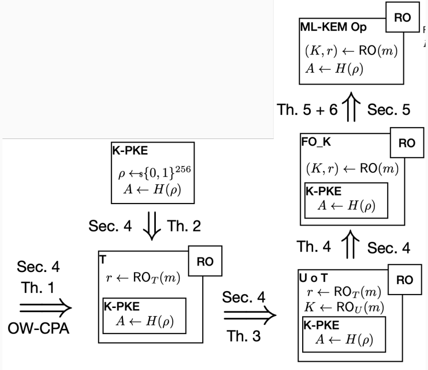


Overview: IND-CCA security

- Machine-checked security proofs for ML-KEM

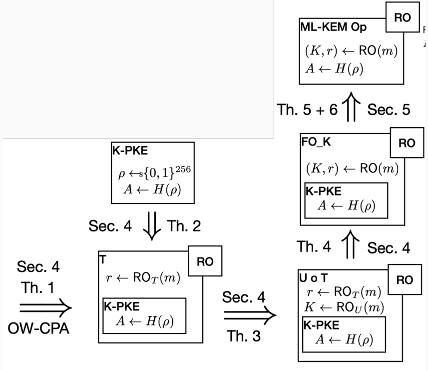


- Machine-checked security proofs for ML-KEM
- IND-CCA security ML-KEM$_{op}$ follows from instantiating FO_k transform:
    - Reuse K-PKE construction from earlier
    - Instantiate with concrete parameters (ML-KEM-768 in our case)

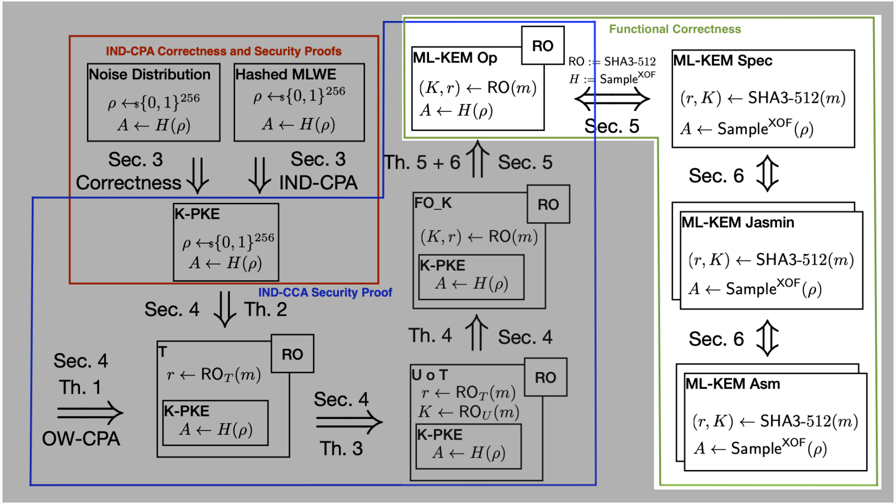


Overview: IND-CCA security

- Machine-checked security proofs for ML-KEM
- IND-CCA security ML-KEM$_{op}$ follows from instantiating FO_k transform:
  - Reuse K-PKE construction from earlier
  - Instantiate with concrete parameters (ML-KEM-768 in our case)
- IND-CCA security of FO_k derived from proof that shows security of the composition of T and U transforms

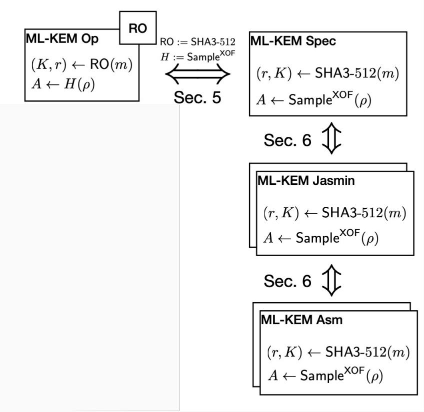


Overview: Implementation correctness



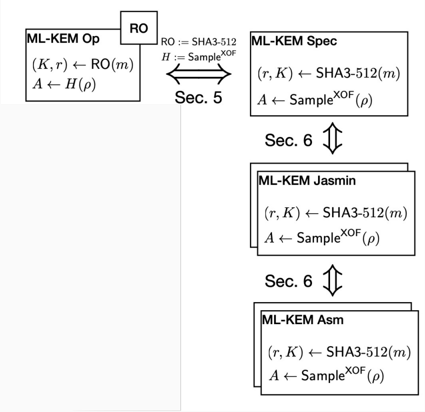
Overview: Implementation correctness

- Functional correctness proof of ML-KEM-768 Jasmin x86-64 implementations



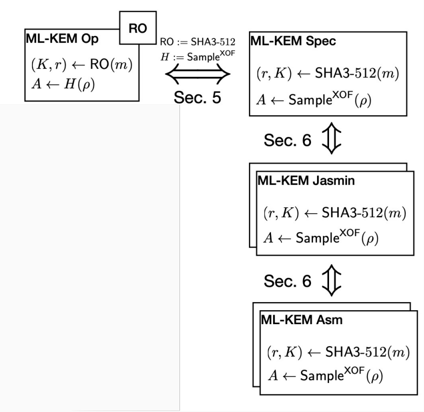
Overview: Implementation correctness

- Functional correctness proof of ML-KEM-768 Jasmin x86-64 implementations
- Jasmin compiler guarantees:
    - Semantic preservation of assembly implementations
    - Constant-time code via the type system

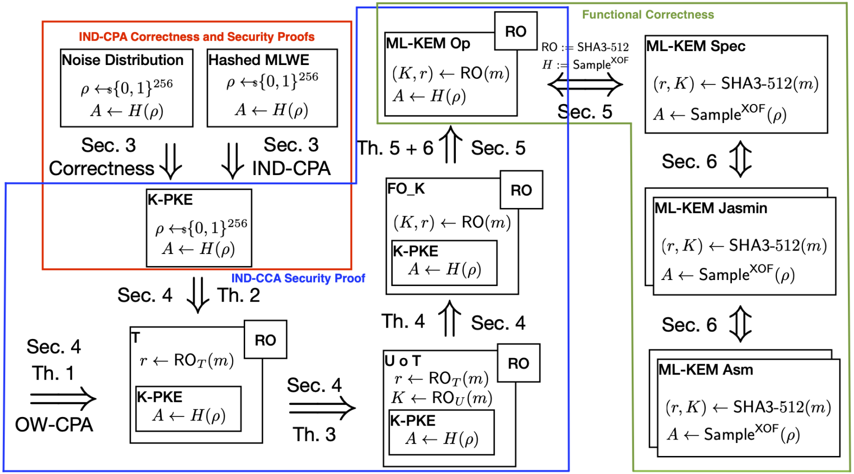


Overview: Implementation correctness

- Functional correctness proof of ML-KEM-768 Jasmin x86-64 implementations
- Jasmin compiler guarantees:
  - Semantic preservation of assembly implementations
  - Constant-time code via the type system
- Gap between security proof and assembly implementation:
  - Hash function (SHA3-512) is not a Random Oracle



Overview



IND-CPA Correctness and Security Proofs
Noise Distribution
$\rho \leftarrow_\$ \{0,1\}^{256}$
$A \leftarrow H(\rho)$
Hashed MLWE
$\rho \leftarrow_\$ \{0,1\}^{256}$
$A \leftarrow H(\rho)$
Sec. 3 Correctness
Sec. 3 IND-CPA
K-PKE
$\rho \leftarrow_\$ \{0,1\}^{256}$
$A \leftarrow H(\rho)$
IND-CCA Security Proof
Sec. 4
Th. 2
Sec. 4
Th. 1
OW-CPA
T
RO
$r \leftarrow \mathsf{RO}_T(m)$
K-PKE
$A \leftarrow H(\rho)$
Sec. 4
Th. 3
U o T
RO
$r \leftarrow \mathsf{RO}_T(m)$
$K \leftarrow \mathsf{RO}_U(m)$
K-PKE
$A \leftarrow H(\rho)$
Th. 4
Sec. 4
FO_K
RO
$(K, r) \leftarrow \mathsf{RO}(m)$
K-PKE
$A \leftarrow H(\rho)$
Th. 5 + 6
Sec. 5
ML-KEM Op
RO
$(K, r) \leftarrow \mathsf{RO}(m)$
$A \leftarrow H(\rho)$
RO := SHA3-512
H := Sample$^{\mathsf{XOF}}$
Sec. 5
Functional Correctness
ML-KEM Spec
$(r, K) \leftarrow$ SHA3-512(m)
$A \leftarrow$ Sample$^{\mathsf{XOF}}(\rho)$
Sec. 6
ML-KEM Jasmin
$(r, K) \leftarrow$ SHA3-512(m)
$A \leftarrow$ Sample$^{\mathsf{XOF}}(\rho)$
Sec. 6
ML-KEM Asm
$(r, K) \leftarrow$ SHA3-512(m)
$A \leftarrow$ Sample$^{\mathsf{XOF}}(\rho)$

EasyCrypt for machine-checked proofs

- Advantages:
  - Games, theorems and proofs match papers

EasyCrypt for machine-checked proofs

- Advantages:
    - Games, theorems and proofs match papers
    - Abstract proofs (e.g. FO transform) can be instantiated with concrete schemes/parameters

EasyCrypt for machine-checked proofs

- Advantages:
  - Games, theorems and proofs match papers
  - Abstract proofs (e.g. FO transform) can be instantiated with concrete schemes/parameters
  - Supports specifications, security proofs, implementations, functional correctness

EasyCrypt for machine-checked proofs

- Advantages:
  - Games, theorems and proofs match papers
  - Abstract proofs (e.g. FO transform) can be instantiated with concrete schemes/parameters
  - Supports specifications, security proofs, implementations, functional correctness
- Drawbacks:

EasyCrypt for machine-checked proofs

- Advantages:
  - Games, theorems and proofs match papers
  - Abstract proofs (e.g. FO transform) can be instantiated with concrete schemes/parameters
  - Supports specifications, security proofs, implementations, functional correctness
- Drawbacks:
  - Proofs are not automatic and require significant effort
  - Theorems can be hard to read

Caveats/Limitations

- Large Trusted Code Base (TCB):
  - EasyCrypt (not formally verified)
  - EasyCrypt proof statements and specifications[1]
  - SMT solvers
- Classical security proof only: no security proof against quantum adversaries

[1] Machine-readable standards could provide a solution to the latter (future work)

Future/Ongoing Work

- Update implementations to final FIPS-203

Future/Ongoing Work

- Update implementations to final FIPS-203
- ML-KEM implementations: more efficient code and support for more parameter sets and architectures (currently x86-64 only)

Future/Ongoing Work

- Update implementations to final FIPS-203
- ML-KEM implementations: more efficient code and support for more parameter sets and architectures (currently x86-64 only)
- Extend (security) proof to the QROM

Future/Ongoing Work

- Update implementations to final FIPS-203
- ML-KEM implementations: more efficient code and support for more parameter sets and architectures (currently x86-64 only)
- Extend (security) proof to the QROM
- Improve proof automation: integrate with other tools (e.g. Cryptoline [FLS+19, TFS+22])

Future/Ongoing Work

- Update implementations to final FIPS-203
- ML-KEM implementations: more efficient code and support for more parameter sets and architectures (currently x86-64 only)
- Extend (security) proof to the QROM
- Improve proof automation: integrate with other tools (e.g. Cryptoline [FLS+19, TFS+22])
- Formally verify other primitives: ML-DSA, SLH-DSA, FrodoKEM, etc

Future/Ongoing Work

- Update implementations to final FIPS-203
- ML-KEM implementations: more efficient code and support for more parameter sets and architectures (currently x86-64 only)
- Extend (security) proof to the QROM
- Improve proof automation: integrate with other tools (e.g. Cryptoline [FLS+19, TFS+22])
- Formally verify other primitives: ML-DSA, SLH-DSA, FrodoKEM, etc
- Industry adoption of formally verified implementations

More online

https://formosa-crypto.org

- **High-assurance Kyber:**
  - Episode IV: https://eprint.iacr.org/2023/215
  - Episode V: https://eprint.iacr.org/2024/843
- **EasyCrypt specifications**: https://github.com/formosa-crypto/crypto-specs
- **Libjade:** https://github.com/formosa-crypto/libjade

References

[ABB+17] José Bacelar Almeida, Manuel Barbosa, Gilles Barthe, Arthur Blot, Benjamin Grégoire, Vincent Laporte, Tiago Oliveira, Hugo Pacheco, Benedikt Schmidt, and Pierre-Yves Strub. Jasmin: High-assurance and high-speed cryptography. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017: 24th Conference on Computer and Communications Security*, pages 1807–1823, Dallas, TX, USA, October 31 – November 2, 2017. ACM Press.

[ABB+20] José Bacelar Almeida, Manuel Barbosa, Gilles Barthe, Benjamin Grégoire, Adrien Koutsos, Vincent Laporte, Tiago Oliveira, and Pierre-Yves Strub. The last mile: High-assurance and high-speed cryptographic implementations. In *2020 IEEE Symposium on Security and Privacy*, pages 965–982, San Francisco, CA, USA, May 18–21, 2020. IEEE Computer Society Press.

[ABB+23] José Bacelar Almeida, Manuel Barbosa, Gilles Barthe, Benjamin Grégoire, Vincent Laporte, Jean-Christophe Léchenet, Tiago Oliveira, Hugo Pacheco, Miguel Quaresma, Peter Schwabe, Antoine Séré, and Pierre-Yves Strub. Formally verifying kyber episode IV: Implementation correctness. Cryptology ePrint Archive, Paper 2023/215, 2023. https://eprint.iacr.org/2023/215.

[BBB+24] Daniel J. Bernstein, Karthikeyan Bhargavan, Shivam Bhasin, Anupam Chattopadhyay, Tee Kiah Chia, Matthias J. Kannwischer, Franziskus Kiefer, Thales Paiva, Prasanna Ravi, and Goutam Tamvada. KyberSlash: Exploiting secret-dependent division timings in kyber implementations. Cryptology ePrint Archive, Paper 2024/1049, 2024.

[BDK+18] Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, and Damien Stehlé. CRYSTALS – Kyber: a CCA-secure module-lattice-based KEM. In *2018 IEEE European Symposium on Security and Privacy, EuroS&P 2018*, pages 353–367. IEEE, 2018. https://eprint.iacr.org/2017/634.

[DKR+20] Jan-Pieter D'Anvers, Angshuman Karmakar, Sujoy Sinha Roy, Frederik Vercauteren, Jose Maria Bermudo Mera, Michiel Van Beirendonck, and Andrea Basso. SABER. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions.

[FLS+19] Yu-Fu Fu, Jiaxiang Liu, Xiaomu Shi, Ming-Hsien Tsai, Bow-Yaw Wang, and Bo-Yin Yang. Signed cryptographic program verification with typed CryptoLine. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019: 26th Conference on Computer and Communications Security*, pages 1591–1606, London, UK, November 11–15, 2019. ACM Press.

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael J. Wiener, editor, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–554, Santa Barbara, CA, USA, August 15–19, 1999. Springer, Berlin, Heidelberg, Germany.

[FO13] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. *Journal of Cryptology*, 26(1):80–101, January 2013.

[GMP22] Paul Grubbs, Varun Maram, and Kenneth G. Paterson. Anonymous, robust post-quantum public key encryption. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology – EUROCRYPT 2022, Part III*, volume 13277 of *Lecture Notes in Computer Science*, pages 402–432, Trondheim, Norway, May 30 – June 3, 2022. Springer, Cham, Switzerland.

[HHK17] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. A modular analysis of the Fujisaki-Okamoto transformation. Cryptology ePrint Archive, Report 2017/604, 2017.

[Nat23] National Institute of Standards and Technology. FIPS PUB 203 (Initial Public Draft) – module-lattice-based key-encapsulation mechanism standard, 2023. https://csrc.nist.gov/pubs/fips/203/ipd.

[TFS+22] Ming-Hsien Tsai, Yu-Fu Fu, Xiaomu Shi, Jiaxiang Liu, Bow-Yaw Wang, and Bo-Yin Yang. Automatic certified verification of cryptographic programs with COQCRYPTOLINE. Cryptology ePrint Archive, Report 2022/1116, 2022.