Rambus

Tighter proofs of CCA security in the QROM

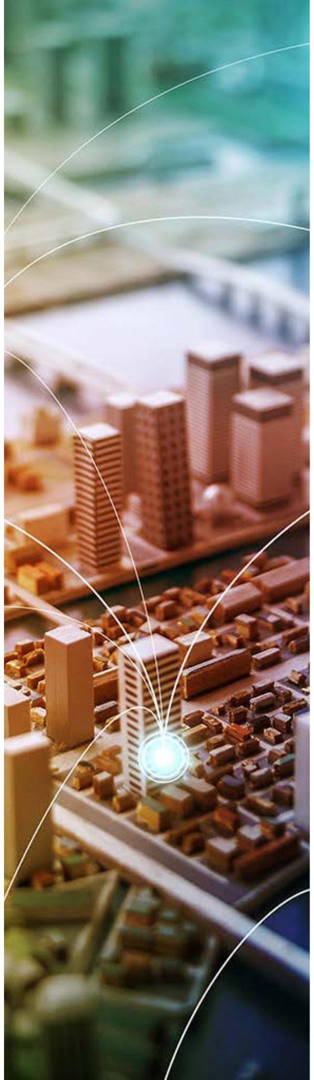
Nina Bindel

Mike Hamburg, presenting

Andreas Hülsing

Edoardo Persichetti

August 23, 2019



Outline

17 different PKE/KEM families in NIST PQC round 2

Core mathematical problem with hashing as glue. Eg:

- Start with passively-secure rPKE or dPKE
- Turn into KEM by encrypting random m; then $k \leftarrow H(m, c)$

CCA security requires variant of Fujisaki-Okamoto transform [FO99]:

- If rPKE, derandomize by setting $\mathbf{coins} \leftarrow G(m)$
- Optional: also use message confirmation $\mathbf{tag} \leftarrow H'(m)$
- Recipient checks that m was encrypted properly; if not, reject
  - Explicit rejection: $k \leftarrow \perp$
  - Implicit rejection: $k \leftarrow H(prfkey, c)$

Contributions of this paper

Modular proof that certain KEMs are almost as secure as underlying PKE

- Either implicit rejection, or explicit + message confirmation

Consider reaction attacks against PKE with nonzero failure probability

- Tightly: adversary must submit a failing ciphertext, without knowledge of sk, to gain advantage

Limitations:

- QROM proof, not standard model
- Some steps aren't tight
- Requires dPKE $\mathrm{Encrypt}(pk,\cdot)$ injective whp
- Doesn't model multi-key attacks
  - Doesn't resolve $G(m)$ vs $G(pk, m)$

Related work

[HHK17]: original modular proofs of QROM security

Comprehensive but not very tight

[SXY18]: tighter results using implicit rejection

[JZCWM18, JZM19]: line of improved approaches, mostly using implicit rejection

[HKSU19]: approximately the same overall bound as this work

With/without injectivity requirements depending on version

Uses disjoint simulability (DS) security notion instead of OW-CPA

Classical vs Quantum Random Oracles

Random oracle model: pretend the hash H is a uniformly random function

- Adversary can't run H anymore, has to call an oracle
- Simulator can see the calls, choose the outputs
  - (They must still look uniformly, independently random)

Classical ROM

- Simulator can record all oracle queries
- Simulator can reprogram oracle adaptively

Quantum ROM

- Queries are quantum superpositions
- Much harder to record oracle queries (see [Zha19])
- Much harder to respond adaptively

Unruh’s one-way to hiding (O2H) technique

Suppose simulator changes oracle G to a slightly different oracle H

- G, H differ only on a small set S

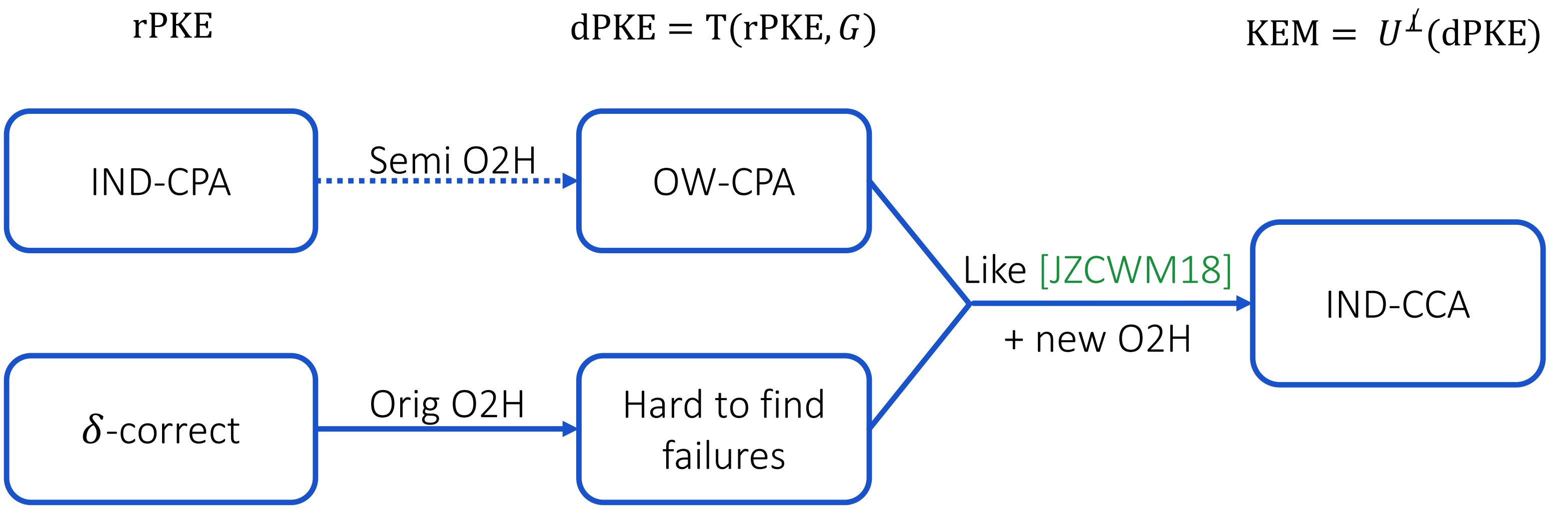
If adversary behaves differently w/p δ, it must be querying some $x \in S$

- Simulator can extract x with probability ϵ depending on δ; depth d

O2H variant	Oracles differ	Sim can simulate	Bound
Original [Unr15]	Arbitrary	G or H	$\delta \leq 2d\sqrt{\epsilon}$
Semi-classical [AHU19]	Arbitrary	(G or H) and S	$\delta \leq 2\sqrt{d\epsilon}$
Double-sided New!	One place	G and H	$\delta \leq 2\sqrt{\epsilon}$

Modular reduction outline

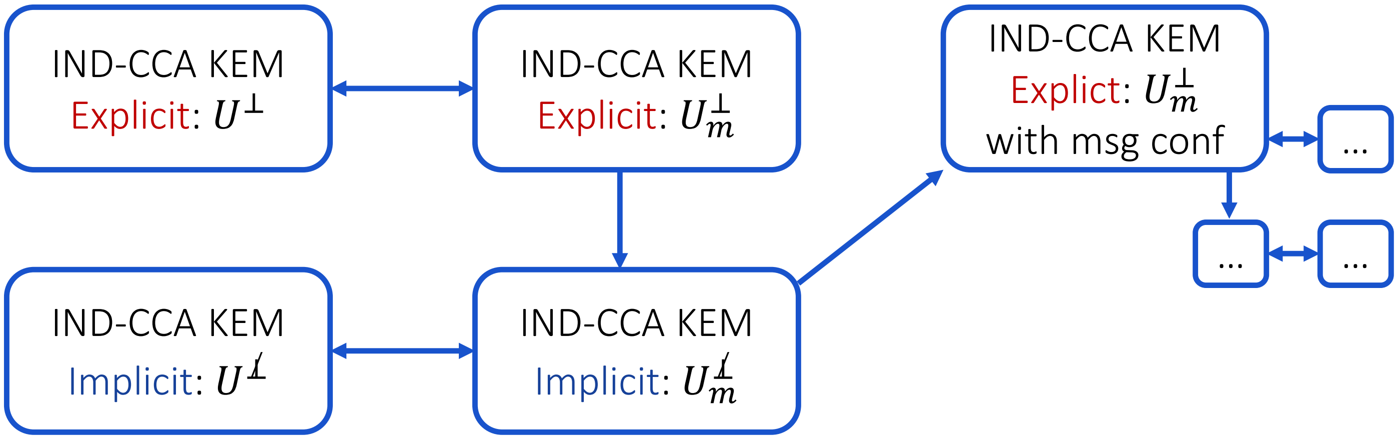
(Assuming $Enc(pk,\cdot)$ injective)



Could start with OW-CPA instead via orig O2H, at cost of factor of d tightness

Modular reduction outline

(Assuming $Enc(pk,\cdot)$ injective)



$k \leftarrow H(m)$ is as secure as $k \leftarrow H(m, c)$

... in single-target case in QROM!

Explicit rejection is secure with (short) message confirmation hash

OW-CPA dPKE → IND-CCA KEM

Encaps(pk):

$m \xleftarrow{R}$ message space

$c \leftarrow \text{Encrypt}(pk, m)$

$k \leftarrow H(m)$

Decaps($(sk, pk, prfk), c$):

If $c = c^*$: return $\perp$

$m' \leftarrow \text{Decrypt}(sk, c)$

If $\text{Encrypt}(pk, m') = c$:

return $k' \leftarrow H(m)$

Else: return $k' \leftarrow \text{PRF}(prfk, c)$

1. Adv is given $c^* \leftarrow \text{Encrypt}(pk, m^*)$ and either $k^* \leftarrow H(m^*)$ or random

OW-CPA dPKE → IND-CCA KEM

Encaps(pk):

$m \xleftarrow{R}$ message space
$c \leftarrow \text{Encrypt}(pk, m)$
$k \leftarrow H(m)$

Decaps($(sk, pk, prfk), c$):

If $c = c^*$: return $\perp$
$m' \leftarrow \text{Decrypt}(sk, c)$
If $\text{Encrypt}(pk, m') = c$:
  return $k' \leftarrow H(m)$
Else: return $k' \leftarrow R(c)$

1. Adv is given $c^* \leftarrow \text{Encrypt}(pk, m^*)$ and either $k^* \leftarrow H(m^*)$ or random
2. Change $\text{PRF}(prfk, c) \rightarrow R(c)$

OW-CPA dPKE → IND-CCA KEM

Encaps(pk):

$m \xleftarrow{R}$ message space
$c \leftarrow \text{Encrypt}(pk, m)$
$k \leftarrow R(c)$

Decaps($(sk, pk, prfk), c$):

If $c = c^*$: return $\perp$
$m' \leftarrow \text{Decrypt}(sk, c)$
If $\text{Encrypt}(pk, m') = c$:
  return $k' \leftarrow R(c)$
Else: return $k' \leftarrow R(c)$

1. Adv is given $c^* \leftarrow \text{Encrypt}(pk, m^*)$ and either $k^* \leftarrow H(m^*)$ or random
2. Change $\text{PRF}(prfk, c) \rightarrow R(c)$
3. Forward $H(m) \rightarrow R\big(\text{Encrypt}(pk, m)\big)$
   - Requires $\text{Encrypt}(pk, \cdot)$ injective
   - Independent of PRF changes (red $R(c)$) unless decryption failed

OW-CPA dPKE → IND-CCA KEM

Encaps(pk):

$m \xleftarrow{R}$ message space
$c \leftarrow \text{Encrypt}(pk, m)$
$k \leftarrow R(c)$

Decaps$((sk, pk, prfk), c)$:

If $c = c^*$: return $\perp$
Else: return $k' \leftarrow R(c)$

1. Adv is given $c^* \leftarrow \text{Encrypt}(pk, m^*)$ and either $k^* \leftarrow H(m^*)$ or random
2. Change $\text{PRF}(prfk, c) \rightarrow R(c)$
3. Forward $H(m) \rightarrow R(\text{Encrypt}(pk, m))$
4. Now **Decaps** oracle is easy

OW-CPA dPKE → IND-CCA KEM

Encaps(pk):

$m \xleftarrow{R}$ message space
$c \leftarrow \text{Encrypt}(pk, m)$
$k \leftarrow R(c)$

Decaps($(sk, pk, prfk), c$):

If $c = c^*$: return $\bot$
Else: return $k' \leftarrow R(c)$

1. Adv is given $c^* \leftarrow \text{Encrypt}(pk, m^*)$ and either $k^* \leftarrow H(m^*)$ or random
2. Change $\text{PRF}(prfk, c) \rightarrow R(c)$
3. Forward $H(m) \rightarrow R\big(\text{Encrypt}(pk, m)\big)$
4. Now **Decaps** oracle is easy
5. Problem is equivalent to distinguishing $(c^*, k^*, H[m^* \rightarrow k^*]) \leftrightarrow (c^*, k^*, H)$
   - Apply double-sided O2H: can recover m^*

Future goals

Tighter proof

- No square roots, possibly using [MW18] notion of IND
- No loss of tightness $d \cdot \mathrm{Adv}_{\mathrm{A}}^{\mathrm{IND-CPA}}$

Get rid of injectivity requirements

Find failing message instead of ciphertext

Multi-key security proof with $H(pk, \dots)$

Prove security of explicit rejection without keyconf

Acknowledgments

This work comes from Oxford 2019 PQC workshop

Thanks to Dan Bernstein, Edward Eaton, Kathrin Hövelmanns, and Mark Zhandry for helpful discussions and feedback.

References

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. *Secure integration of asymmetric and symmetric encryption schemes*. Crypto 1999.

[Unr15] Dominique Unruh. *Revocable quantum timed-release encryption*. JACM 2015.

[HHK17] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. *A Modular Analysis of the Fujisaki-Okamoto Transformation*. TCC 2017.

[JZCWM18] Haodong Jiang and Zhenfeng Zhang and Long Chen and Hong Wang and Zhi Ma. IND-CCA-secure Key Encapsulation Mechanism in the Quantum Random Oracle Model, Revisited. Crypto 2018.

[MW18] Daniele Micciancio and Michael Walter. *On the Bit security of cryptographic primitives*. Eurocrypt 2018.

References

[SXY18] Keita Xagawa and Takashi Yamakawa. *Tightly-secure key-encapsulation mechanism in the quantum random oracle model*. Eurocrypt 2018.

[HKSU18] Kathrin Hövelmanns, Eike Kiltz, Sven Schäge, and Dominique Unruh. *Generic Authenticated Key Exchange in the Quantum Random Oracle Model.* ePrint 2018.

[Zha19] Mark Zhandry. *How to Record Quantum Queries, and Applications to Quantum Indifferentiability*. Crypto 2019.

[AHU19] Andris Ambainis, Mike Hamburg, and Dominique Unruh. *Quantum security proofs using semi-classical oracles.* Crypto 2019.

[JZM19] Haodong Jiang and Zhenfeng Zhang and Zhi Ma. *Tighter security proofs for generic key encapsulation mechanism in the quantum random oracle model.* ePrint 2019.

Questions?

Rambus
Data · Faster · Safer