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 coins \( \leftarrow G(m) \)
- Optional: also use message confirmation 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 $\text{Encrypt}(pk, \cdot)$ 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 $\delta$, it must be querying some $x \in S$

Simulator can extract $x$ with probability $\epsilon$ depending on $\delta$; depth $d$

<table>
<thead>
<tr>
<th>O2H variant</th>
<th>Oracles differ</th>
<th>Sim can simulate</th>
<th>Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original [Unr15]</td>
<td>Arbitrary</td>
<td>$G$ or $H$</td>
<td>$\delta \leq 2d\sqrt{\epsilon}$</td>
</tr>
<tr>
<td>Semi-classical [AHU19]</td>
<td>Arbitrary</td>
<td>$(G$ or $H$) and $S$</td>
<td>$\delta \leq 2\sqrt{d\epsilon}$</td>
</tr>
<tr>
<td>Double-sided</td>
<td>One place</td>
<td>$G$ and $H$</td>
<td>$\delta \leq 2\sqrt{\epsilon}$</td>
</tr>
</tbody>
</table>
Modular reduction outline

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

- **rPKE**
  - IND-CPA
  - δ-correct

- **dPKE = T(rPKE, G)**
  - OW-CPA
  - Hard to find failures

- **KEM = $U^L(dPKE)$**
  - IND-CCA

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

Like [JZCWM18] + new O2H
Modular reduction outline

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

$\mathbb{U} \not\perp$ IND-CCA KEM

Explicit: $U \perp$

$\mathbb{U} \not\perp$ IND-CCA KEM

Implicit: $U \perp$

$\mathbb{U} \perp$ IND-CCA KEM

Explicit: $U \perp$

with msg conf

$\mathbb{U} \perp$ IND-CCA KEM

Implicit: $U \perp$

$\mathbb{U} \perp$ IND-CCA KEM

Explicit: $U \perp$

$k \leftarrow H(m)$ 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 $\rightarrow$ IND-CCA KEM

Encaps$(pk)$:

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

Decaps$((sk, pk, prf k), c)$:

\[ \text{If } c = c^* : \text{ return } \bot \]
\[ m' \leftarrow \text{Decrypt}(sk, c) \]
\[ \text{If Encrypt}(pk, m') = c: \]
\[ \text{return } k' \leftarrow H(m) \]
\[ \text{Else: return } k' \leftarrow \text{PRF}(prf k, c) \]

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\((pk)\):
\[
\begin{align*}
R & \quad m \leftarrow \text{message space} \\
& \quad c \leftarrow \text{Encrypt}(pk, m) \\
& \quad k \leftarrow H(m)
\end{align*}
\]

Decaps\(((sk, pk, prfk), c)\):
\[
\begin{align*}
& \quad \text{If } c = c^* : \text{return } \bot \\
& \quad m' \leftarrow \text{Decrypt}(sk, c) \\
& \quad \text{If } \text{Encrypt}(pk, m') = c:
& \quad \quad \text{return } k' \leftarrow H(m) \\
& \quad \text{Else: return } k' \leftarrow R(c)
\end{align*}
\]

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

Encaps($pk$):
\[ m \leftarrow \text{message space} \]
\[ c \leftarrow \text{Encrypt}(pk, m) \]
\[ k \leftarrow R(c) \]

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

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

Encaps($pk$):

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

Decaps((sk, pk, prf k), c):

\[ \text{If } c = c^* : \text{return } \bot \]
\[ \text{Else: return } k' \leftarrow R(c) \]

1. Adv is given $c^* \leftarrow \text{Encrypt}(pk, m^*)$ and either $k^* \leftarrow H(m^*)$ or random
2. Change PRF($prf k, c$) $\rightarrow R(c)$
3. Forward $H(m) \rightarrow R(\text{Encrypt}(pk, m))$
4. Now Decaps oracle is easy
OW-CPA dPKE → IND-CCA KEM

Encaps($pk$):
\[ m \leftarrow \mathbb{R} \text{ message space} \]
\[ c \leftarrow \text{Encrypt}(pk, m) \]
\[ k \leftarrow R(c) \]

Decaps((sk, pk, prf k), c):
\[ \text{If } c = c^*: \text{return } \perp \]
\[ \text{Else: return } k' \leftarrow R(c) \]

1. Adv is given $c^* \leftarrow \text{Encrypt}(pk, m^*)$ and either $k^* \leftarrow H(m^*)$ or random
2. Change \( \text{PRF}(prf k, c) \rightarrow R(c) \)
3. Forward $H(m) \rightarrow R(\text{Encrypt}(pk, m))$
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 $d \cdot \text{Adv}_A^{\text{IND-CPA}}$

Get rid of injectivity requirements

Find failing message instead of ciphertext

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

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


References


