Permutation-based cryptography and post-quantum security

Dominique Unruh

RWTH Aachen
University of Tartu

Permutations in cryptography

<u>Unstructured permutations</u>

- Building block of many constructions
  - E.g., Sponge/SHA3, Even-Mansour
- Block ciphers
  - Family of permutations
  - As building block
  - As a goal
    e.g., 3/4-round Luby-Rackoff

Random functions / permutations

- Unstructured functions / permutations
  often modeled as
  random functions / permutations
- E.g., random oracle model, ideal cipher model, pseudo-random permutations, round function of Sponge

	Uses	Makes
Luby-Rackoff	Function (family)	Permutation (family)
Even-Mansour	Permutation	Permutation (family)
Sponge	Permutation	Function

How to prove things? (Classically)

- Consider random functions (heuristically or PRF)
- Random function = lazy sampled oracle

much easier!!!

Random function
$f \xleftarrow{\$} \text{Functions}$

Query(x):
return $f(x)$

$\equiv$

Lazy sampler
$f \leftarrow \text{Empty}$

Query(x):
$f(x) \xleftarrow{\$} \text{Bits}$ (cached)
return $f(x)$

Lazy sampling permutations

- Lazy sampling: works same for permutations

Random permutation
$\pi \leftarrow^{\$}$ Permutations

<u>Query(x):</u>
return $\pi(x)$

<u>Inv-Query(y):</u>
return $\pi^{-1}(y)$

$\approx$

Lazy sampler
$\pi \leftarrow$ Empty

<u>Query(x):</u>
$pi(x) \leftarrow^{\$}$ Bits
(cached)
return $\pi(x)$

<u>Inv-Query(y):</u>
$x \leftarrow^{\$}$ Bits
$pi(x) \leftarrow y$
return x

Only needed for invertible permutations

Postquantum security: Lazy sampling?

- Lazy sampling in post-quantum security?
- Adversary can do "superposition queries": Ask for

$$|H(1)\rangle + |H(2)\rangle + |H(3)\rangle + \cdots$$

  in single query
- Lazy sampler would need to sample whole oracle in first step
  → defeats the idea
- Measurements disturb state
  → can't "look" at query or make log

Long believed to be impossible!

Post-quantum security: Not all is well

- Can't just heuristically assume a "classical" random function
- Known:
  - Fiat-Shamir can become postquantum insecure (salvageable in many circumstances)
  - Fischlin transform also
  - 3-round Luby-Rackoff insecure
- Must model random functions/permutations with superposition queries!

[Ambainis, Rosmanis, U, Quantum attacks on classical proof systems, 2014]

[Kuwakado, Morii, Quantum distinguisher between the 3-round Feistel..., 2010]

Post-quantum security: Solutions?

Direct proofs (situation-specific technique, model quantum)

- Random function is one-way
- Random function ≈ random permutation (non-invertible)
- Random function is collision-resistant
- Etc.

Hard for complex / multi-round constructions (e.g., Sponge)

[Zhandry, A note on quantum collision and , 2023]

Post-quantum security: Solutions? (II)

General methods:

- Oneway-to-hiding (O2H)
  - "Cannot distinguish H and modified H, unless I query H where it was modified"
  - Also applies to permutations (not sure it's used)
- Compressed oracles
  - Lazy sampling in superposition
  - Tricky but powerful
  - Not for permutations (yet)

[U, Revocable quantum timed-release encryption, 2014]

[Zhandry, How to record quantum queries, 2019]

Compressed oracles (I)

- Want to simulate a random function
- Keep quantum register for each output
- Initially: In superposition between all outputs

$$|0\rangle + |1\rangle + |2\rangle + \cdots$$

- Querying collapses

$$|0\rangle + |2\rangle + |4\rangle + \cdots$$

[Zhandry, How to record quantum queries, 2019]

Compressed oracles (II)

- Added trick: New state $|\perp\rangle$ instead of $|0\rangle + |1\rangle + |2\rangle + \cdots$
- Query:

$$|\perp\rangle \quad \rightarrow \quad |0\rangle + |2\rangle + |4\rangle + \cdots$$

- Use as initial state for all outputs.
- (Many details omitted)

Compressed oracles (III)

Effect:

- Lazy sampling in superposition
- E.g., query $|0\rangle$, then query $|1\rangle + |2\rangle + |3\rangle$.
- Leads to:

$$|0 \mapsto x, 1 \mapsto y\rangle + |0 \mapsto x, 2 \mapsto y\rangle + |0 \mapsto x, 3 \mapsto y\rangle$$

- In each state, only 2 outputs have been sampled!

Compressed oracles (IV)

$$|0 \mapsto x, 1 \mapsto y\rangle + |0 \mapsto x, 2 \mapsto y\rangle + |0 \mapsto x, 3 \mapsto y\rangle$$

Powerful:

- Can track queries
- Efficient representation
- Can look at the list of queries
  - Important for simulators

→ Indifferentiability proofs for Merkle-Damgård and Sponge

[Zhandry, How to record quantum queries, 2019]

[Czajkowski, Majenz, Schaffner, Zur, Quantum lazy sampling ..., 2019]

Compressed oracles and permutations

- Can we use the compressed oracle for permutations?
- Yes and no…
- For non-invertible permutations:
  Trivial by "random permutation $\approx$ random function"
- For invertible permutations…

Compressed oracle for invertible permutations

- Earlier version of [Czajkowski, Majenz, Schaffner, Zur, 2019]: Constructs a C.O. variant for permutations, shows indifferentiability of Sponge. Broken!
- [Unruh, Compressed Permutation Oracles, 2021]: Constructs a C.O. variant for permutations, shows collision-resistance of Sponge. Broken!
- [Hosoyamada, Iwata, 4-Round Luby-Rackoff…, 2019]: Uses C.O. to that Luby-Rackoff implements a permutation. Broken!

There's a curse on the C.O. and permutations!

One more paper appeared last week…

Big questions

- Can we simulate *permutations* using the compressed oracles?

- Can we show the postquantum security of Sponge?
  (Using an invertible round function)

(Trying to) save the C.O. permutations

Defining the C.O. with permutations is easy:

- C.O. maintains a list of input/output pairs (in superposition)
- Oracle **QUERY** allows to get an output
- Define oracle **FLIP** that exchanges input/output pairs (in superposition)
- Result: A permutation that is lazily sampled, and can be queried both directions

[U, Towards Compressed Permutation Oracles, 2023]

(Trying to) save the C.O. permutations (II)

Problem:

QUERY/FLIP $\approx$ random permutation ???

- Proving this is elusive, conjectured true

Only result:

If you can find a construction using random oracles
that implements the QUERY/FLIP C.O. (Luby-Rackoff?)
then
QUERY/FLIP $\approx$ random permutation

[U, Towards Compressed Permutation Oracles, 2023]

(Trying to) save the C.O. permutations (III)

$$\mathrm{CPO} \approx f \circ \mathrm{CPO}$$ // symmetry

$$\approx f \circ C(H)$$ // by proof

$$\approx f$$ // *C(H)* perm, *f* random

$$\Rightarrow \mathrm{CPO} \approx f$$

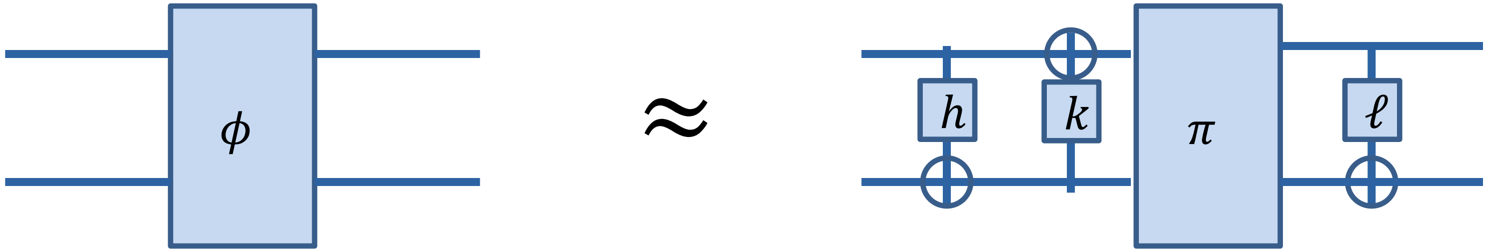
Proving Sponge

Can we prove Sponge secure? (E.g., indifferentiable.)

- If QUERY/FLIP works, probably yes. (Not checked)
- *Last week* on arXiv:
  Indifferentiability of Sponge using C.O.
- Sidesteps the permutation C.O.
- I have not yet managed to understand/check the detail,
  but I can give the basic idea

Proving Sponge (II)

Idea:



Then prove security using the rhs as round function, and treat permutation π as public.

Needs C.O. only for h, k, ℓ

[Alagic, Carolan, Majenz, Tokat, The Sponge is quantum indifferentiable, 2025]

Conclusions

- Permutations quite tricky in the post-quantum world
- Sponge maybe secure
  - Paper appeared last week (arXiv)
  - But I think we need to wait a bit till we know whether we can trust it

Funded by the European Union
European Research Council
Established by the European Commission
European Union
European Regional Development Fund