Multi-target Preimage search using parallel Grover

Gustavo Banegas[1] and Daniel J. Bernstein[1,2]

TU/e Technische Universiteit Eindhoven University of Technology

ECRYPT-NET Meeting
October 11th, 2017

[1]Department of Mathematics and Computer Science
Technische Universiteit Eindhoven
gustavo@cryptme.in

[2]Department of Computer Science
University of Illinois at Chicago
djb@cr.yp.to

Introduction

Preimage

Let H be a function that $H : \{0,1\}^b \to \{0,1\}^b$. Preimage search is given an output y, find a x such that $H(x) = y$.

Introduction

Preimage

Let H be a function that $H : \{0,1\}^b \to \{0,1\}^b$. Preimage search is given an output y, find a x such that $H(x) = y$.
It is desirable that given an output it should be computationally infeasible to find any input that maps to that output.

Introduction

Brute-force search for one preimage

Let H be a function that $H : \{0,1\}^b \rightarrow \{0,1\}^b$.
The brute force is to check every input x given an output y. The time complexity will be 2^b guesses using classical computers.
If we apply Grover's algorithm, using a quantum computer, the complexity decreases to $2^{b/2}$ guesses.

Introduction

Brute-force search for multi target preimages

Let H be a function that $H : \{0,1\}^b \to \{0,1\}^b$.
Now, we have a set of output y's, i.e., $Y = \{y_1, y_2, \ldots, y_t\}$ and we want to find one y_i and we verify every input x with a set of output Y.
If we **ignore several costs**, the complexity decreases to $2^b/t$ guesses in a classical computer.
If we apply Grover's algorithm, using a quantum computer, the complexity decreases to $2^{b/2}/t^{1/2}$ guesses.

Introduction

Costs for comparison

One big cost for preimage search in both cases is the comparisons.

Introduction

Costs for comparison

One big cost for preimage search in both cases is the comparisons.

- Classical computer:
  - Single target: 2^b
  - Multi target: $t \cdot (2^b)/t$

Introduction

Costs for comparison

One big cost for preimage search in both cases is the comparisons.

- Classical computer:
  - Single target: 2^b
  - Multi target: $t \cdot (2^b)/t$
- Quantum computer:
  - Single target: $2^{b/2}$
  - Multi target: $t \cdot (2^{b/2})/t^{1/2}$

Introduction

Parallel multi-target image attack for AES:

Introduction

Parallel multi-target image attack for AES:

van Oorschot–Wiener “parallel rho method”

Introduction

Parallel multi-target image attack for AES:

van Oorschot–Wiener "parallel rho method"

- It uses a mesh of p small processors.

Introduction

Parallel multi-target image attack for AES:

van Oorschot–Wiener "parallel rho method"

- It uses a mesh of p small processors.
- Each processor runs $2^{128}/pt$ fast steps, to find one of t independent AES keys $k_1, \ldots, k_t$, using a fixed plain text, e.g, AES(0).

Introduction

Parallel multi-target image attack for AES:

van Oorschot–Wiener "parallel rho method"

- It uses a mesh of p small processors.
- Each processor runs $2^{128}/pt$ fast steps, to find one of t independent AES keys $k_1, \ldots, k_t$, using a fixed plain text, e.g, AES(0).

However, it is pre-quantum.

Introduction

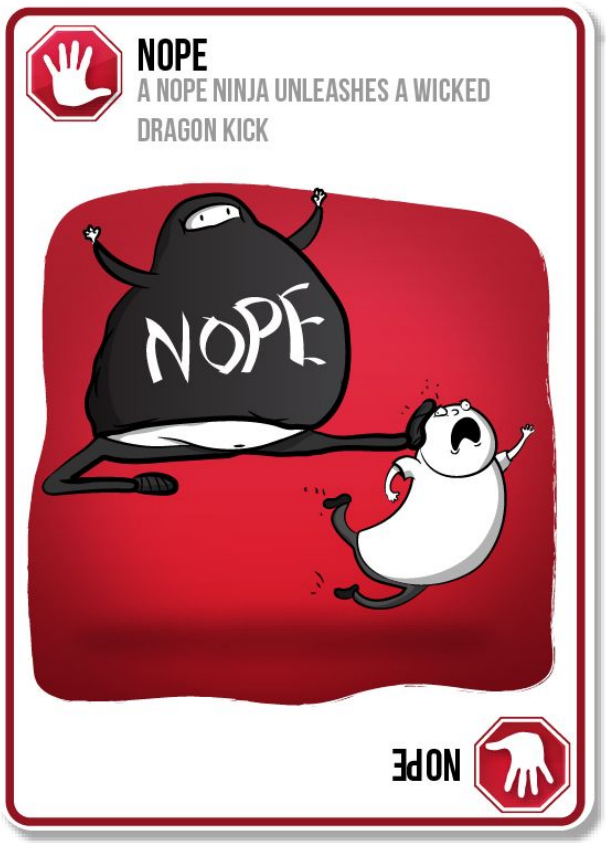
Is AES−128 secure against quantum attacks?

NIST has claimed that AES-128 is secure enough.

Introduction

Is AES−128 secure against quantum attacks?

NIST has claimed that AES-128 is secure enough.



Introduction - Parallel rho method

Distinguished Point

Consider $H : \{0,1\}^b \to \{0,1\}^b$
Take x an input of H, $x' = H(x)$.
Thereafter, take x' and apply H again, $x'' = H(x')$.
It is possible to do it n times and we denote as $H^n(x)$.

Introduction - Parallel rho method

Distinguished Point

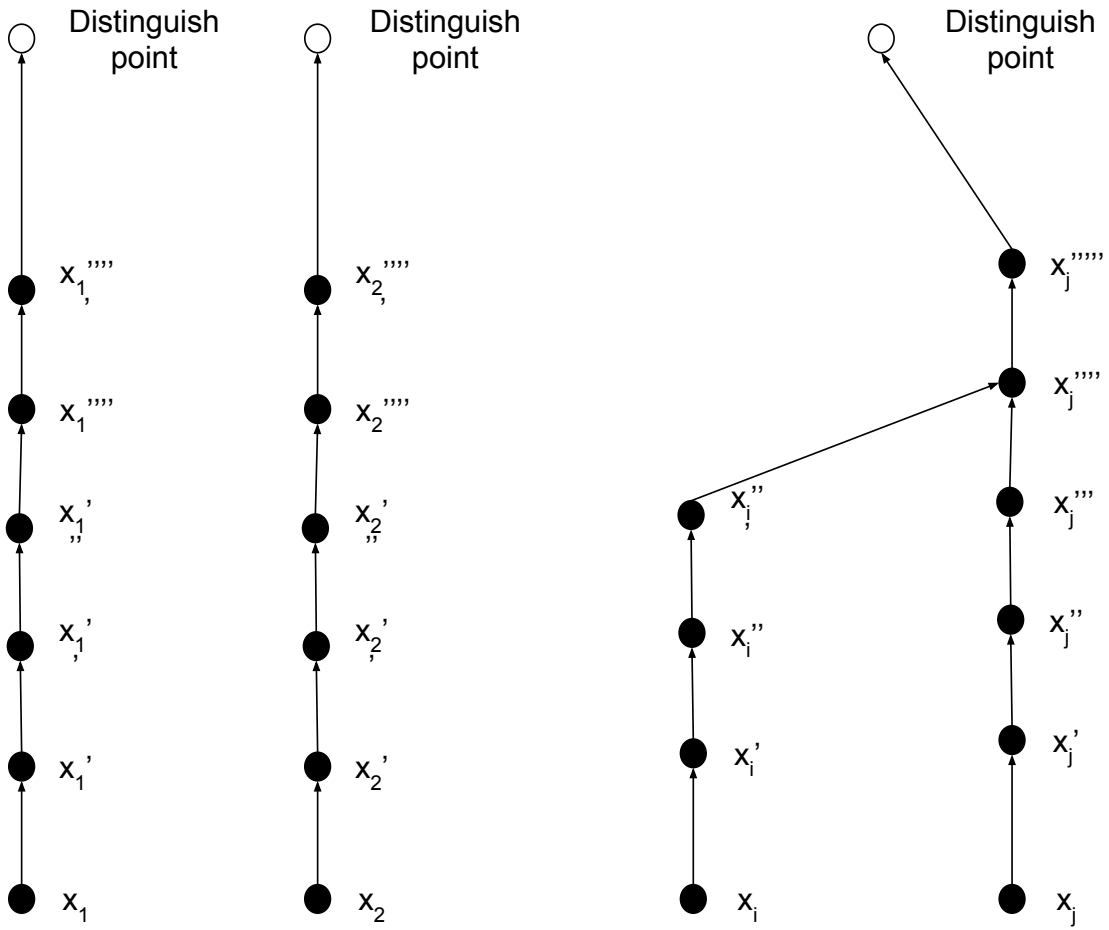
Consider $H : \{0,1\}^b \to \{0,1\}^b$

We want that our distinguished point satisfied $d = b/2$ and we denote as:

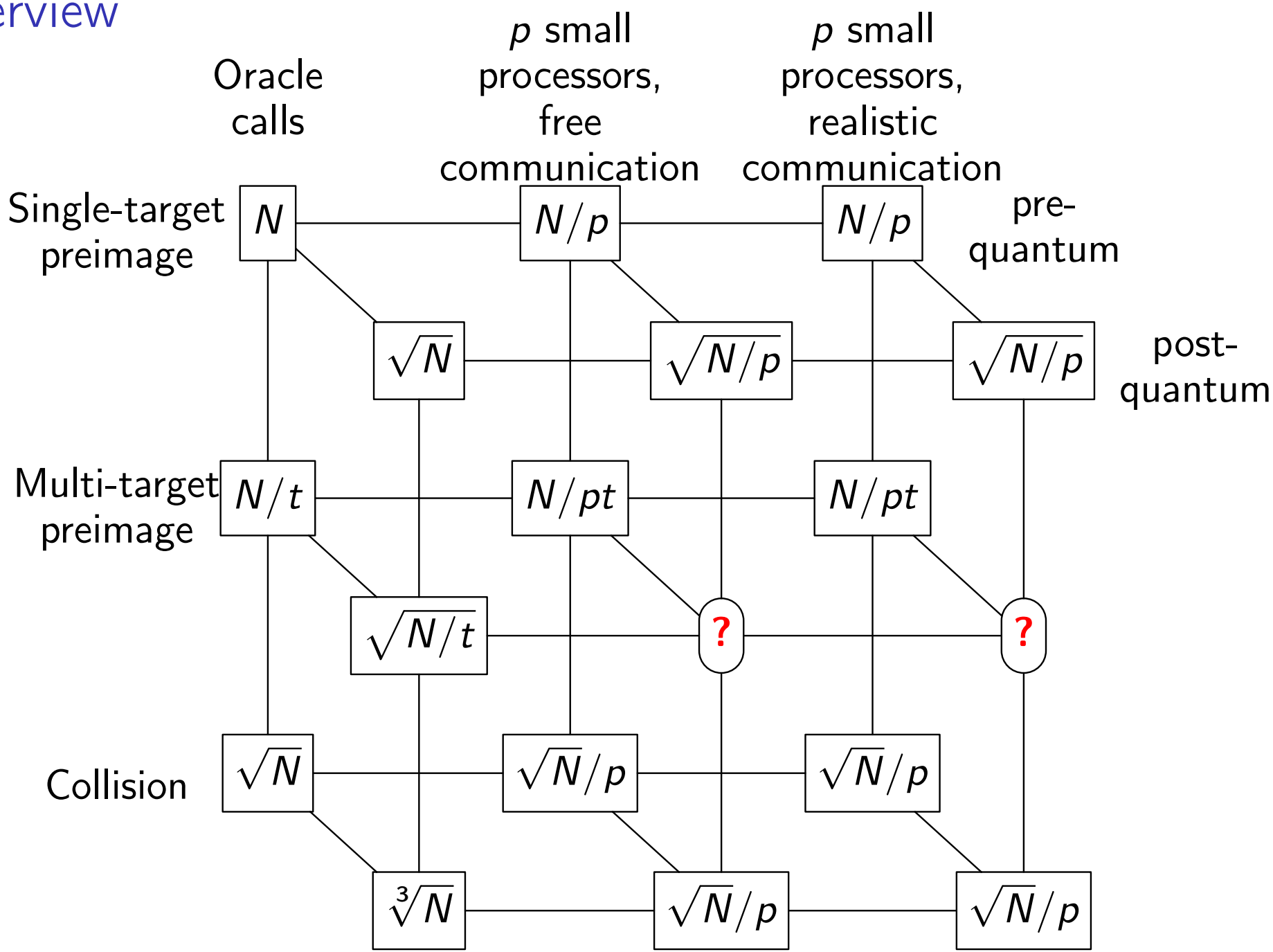
$$H_d(x) = \underbrace{0 \ldots 0}_{d \text{ zeros}} \{0,1\}^{b/2}$$

Introduction - Parallel rho method

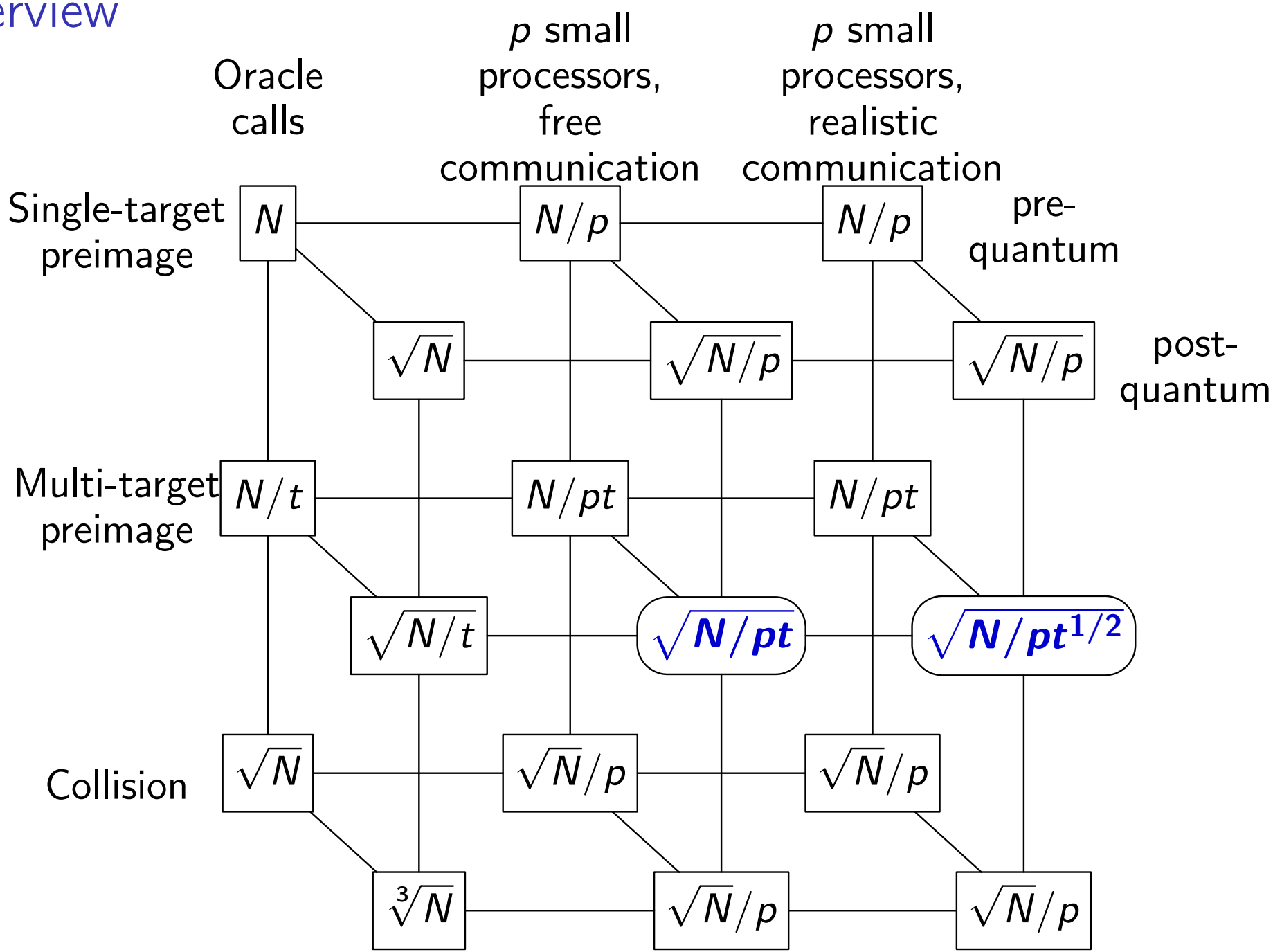
Distinguished Point



Overview


Oracle calls
p small processors, free communication
p small processors, realistic communication
Single-target preimage
Multi-target preimage
Collision
pre-quantum
post-quantum
N
N/p
N/p
√N
√(N/p)
√(N/p)
N/t
N/pt
N/pt
√(N/t)
?
?
√N
√(N/p)
√(N/p)
∛N
√(N/p)
√(N/p)

Overview


Oracle calls
p small processors, free communication
p small processors, realistic communication
pre-quantum
post-quantum
Single-target preimage
Multi-target preimage
Collision
N
N/p
N/p
$\sqrt{N}$
$\sqrt{N/p}$
$\sqrt{N/p}$
N/t
N/pt
N/pt
$\sqrt{N/t}$
$\sqrt{N/pt}$
$\sqrt{N/pt^{1/2}}$
$\sqrt{N}$
$\sqrt{N/p}$
$\sqrt{N/p}$
$\sqrt[3]{N}$
$\sqrt{N/p}$
$\sqrt{N/p}$

Distinguished point in quantum setting

Distinguished point in quantum computers

- The operations in quantum computer must be **reversible**;

Distinguished point in quantum setting

Distinguished point in quantum computers

- The operations in quantum computer must be **reversible**;
- It is not possible to design a "simple circuit" for distinguished point;

Distinguished point in quantum setting

Distinguished point in quantum computers

- The operations in quantum computer must be **reversible**;
- It is not possible to design a "simple circuit" for distinguished point;

- The sorting needs to be reversible too.

Distinguished point in quantum setting

Using classical computers

Example to compute $H^3(x)$:

Distinguished point in quantum setting

Using classical computers

Example to compute $H^3(x)$:

time 0:	x	0	0

Distinguished point in quantum setting

Using classical computers

Example to compute $H^3(x)$:

time 0:	x	0	0
time 1:	x	0	$H(x)$

Distinguished point in quantum setting

Using classical computers

Example to compute $H^3(x)$:

time 0:	x	0	0
time 1:	x	0	$H(x)$
time 2:	x	0	$H^2(x)$

Distinguished point in quantum setting

Using classical computers

Example to compute $H^3(x)$:

time 0:	x	0	0
time 1:	x	0	$H(x)$
time 2:	x	0	$H^2(x)$
time 3:	x	$H^3(x)$	$H^2(x)$

Distinguished point in quantum setting

Trade-off from Bennett–Tompa

Example to compute $H^3(x)$:

Distinguished point in quantum setting

Trade-off from Bennett–Tompa

Example to compute $H^3(x)$:

time 0:	x	0	0	0

Distinguished point in quantum setting

Trade-off from Bennett–Tompa

Example to compute $H^3(x)$:

time 0:	x	0	0	0
time 1:	x	0	$H(x)$	0

Distinguished point in quantum setting

Trade-off from Bennett–Tompa

Example to compute $H^3(x)$:

time 0:	x	0	0	0
time 1:	x	0	$H(x)$	0
time 2:	x	0	$H(x)$	$H^2(x)$

Distinguished point in quantum setting

Trade-off from Bennett–Tompa

Example to compute $H^3(x)$:

time 0:	x	0	0	0
time 1:	x	0	$H(x)$	0
time 2:	x	0	$H(x)$	$H^2(x)$
time 3:	x	$H^3(x)$	$H(x)$	$H^2(x)$

Distinguished point in quantum setting

Trade-off from Bennett–Tompa

Example to compute $H^3(x)$:

time 0:	x	0	0	0
time 1:	x	0	$H(x)$	0
time 2:	x	0	$H(x)$	$H^2(x)$
time 3:	x	$H^3(x)$	$H(x)$	$H^2(x)$
time 4:	x	$H^3(x)$	$H(x)$	0

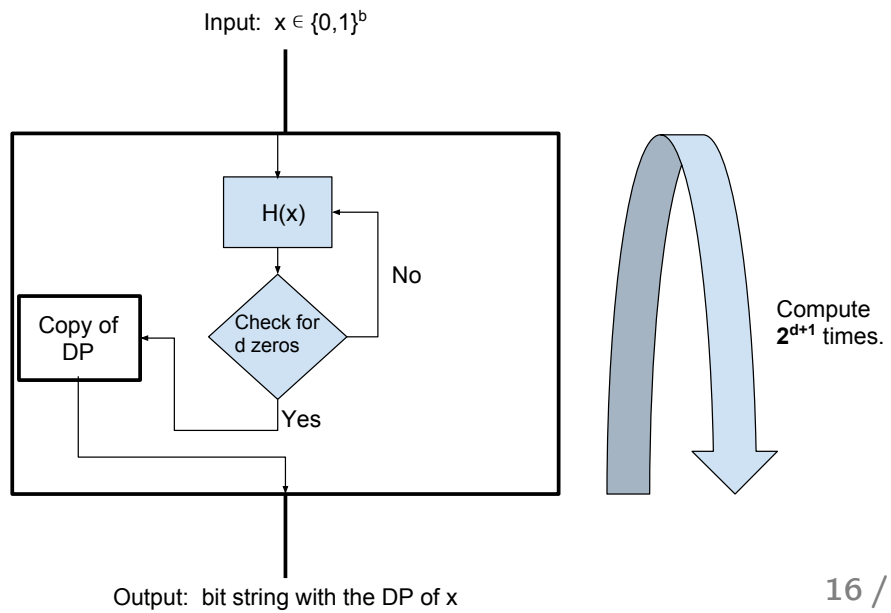
Distinguished point in quantum setting

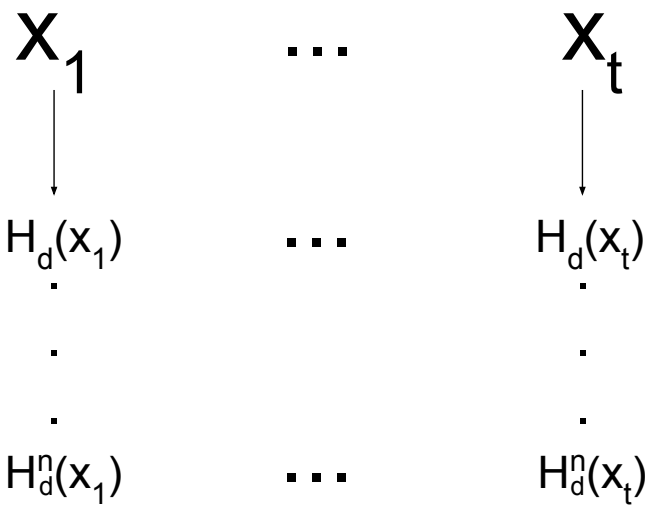
Trade-off from Bennett–Tompa

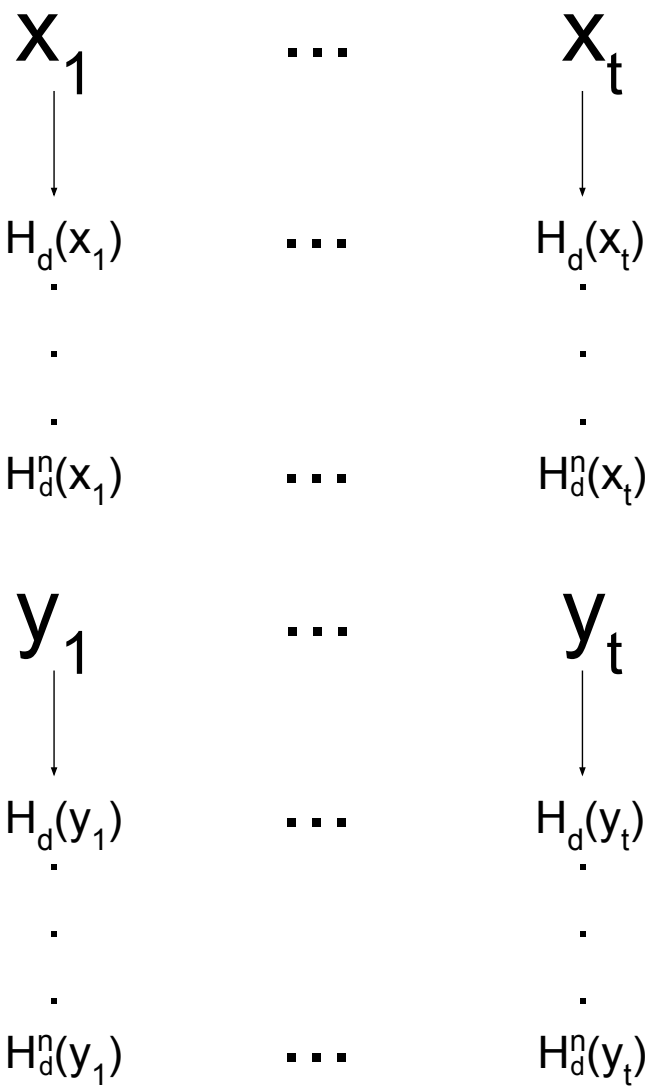
Example to compute $H^3(x)$:

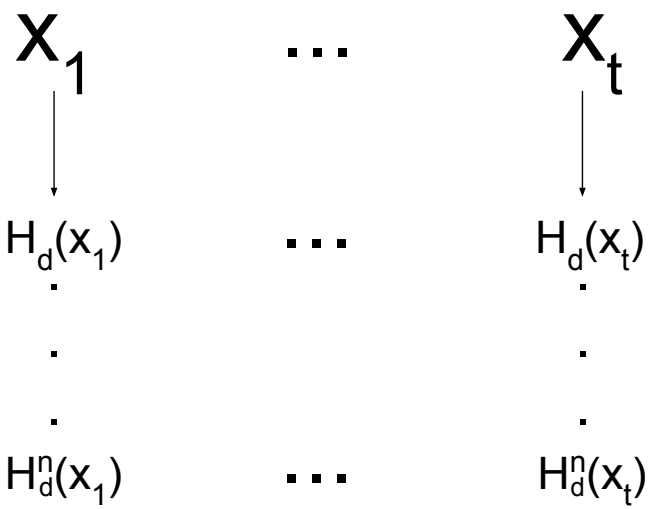
time 0:	x	0	0	0
time 1:	x	0	$H(x)$	0
time 2:	x	0	$H(x)$	$H^2(x)$
time 3:	x	$H^3(x)$	$H(x)$	$H^2(x)$
time 4:	x	$H^3(x)$	$H(x)$	0
time 5:	x	$H^3(x)$	0	0

$H_d^n(x) =$




x_1 ... x_t
$H_d(x_1)$ ... $H_d(x_t)$
$H_d^n(x_1)$ ... $H_d^n(x_t)$


x_1 ... x_t
$H_d(x_1)$ ... $H_d(x_t)$
$H_d^n(x_1)$ ... $H_d^n(x_t)$
y_1 ... y_t
$H_d(y_1)$ ... $H_d(y_t)$
$H_d^n(y_1)$ ... $H_d^n(y_t)$



$$H_d^n(y_i) \stackrel{?}{=} H_d^n(x_j)$$

If this condition is true then we need to run classically:

$$H^{n_k}(x_i) = y_j$$

Reversibility

Reversibility of Distinguished point

- Bennett-Tompa technique to build a reversible circuit for H_d^n;
- It is possible to achieve $a + O(b \log_2 n)$ ancillas and gate depth $O(gn^{1+\epsilon})$.

[3] Efficient distributed quantum computing
Beals, Robert and Brierley, Stephen and Gray, Oliver and Harrow, Aram W. and Kutin, Samuel and Linden, Noah and Shepherd, Dan and Stather, Mark

Reversibility

Reversibility of Distinguished point

- Bennett-Tompa technique to build a reversible circuit for H_d^n;
- It is possible to achieve $a + O(b \log_2 n)$ ancillas and gate depth $O(gn^{1+\epsilon})$.

Reversibility of sorting on a mesh network

- Using the sorting strategy from "Efficient distributed quantum computing"[3];
- We used odd-even mergesort;
- It is possible to perform the sorting of t elements using $O(t(b + (\log t)^2))$ ancillas and $O(t^{1/2}(\log t)^2)$ steps.

---

[3] Efficient distributed quantum computing
Beals, Robert and Brierley, Stephen and Gray, Oliver and Harrow, Aram W. and Kutin, Samuel and Linden, Noah and Shepherd, Dan and Stather, Mark

Finding t-images

Fix images $y_1, \ldots, y_t$. We build a reversible circuit that performs the following operations:

- Input a vector $(x_1, \ldots, x_t)$.

Finding t-images

Fix images $y_1, \ldots, y_t$. We build a reversible circuit that performs the following operations:

- Input a vector $(x_1, \ldots, x_t)$.
- Compute, in parallel, the chain ends for $x_1, \ldots, x_t$: i.e., $H_d^n(x_1), \ldots, H_d^n(x_t)$.

Finding t-images

Fix images $y_1, \ldots, y_t$. We build a reversible circuit that performs the following operations:

- Input a vector $(x_1, \ldots, x_t)$.
- Compute, in parallel, the chain ends for $x_1, \ldots, x_t$: i.e., $H_d^n(x_1), \ldots, H_d^n(x_t)$.
- Precompute the chain ends for $y_1, \ldots, y_t$.

Finding t-images

Fix images $y_1, \ldots, y_t$. We build a reversible circuit that performs the following operations:

- Input a vector $(x_1, \ldots, x_t)$.
- Compute, in parallel, the chain ends for $x_1, \ldots, x_t$: i.e., $H_d^n(x_1), \ldots, H_d^n(x_t)$.
- Precompute the chain ends for $y_1, \ldots, y_t$.
- Sort the chain ends for $x_1, \ldots, x_t$ and the chain ends for $y_1, \ldots, y_t$.

Finding t-images

Fix images $y_1, \ldots, y_t$. We build a reversible circuit that performs the following operations:

- Input a vector $(x_1, \ldots, x_t)$.
- Compute, in parallel, the chain ends for $x_1, \ldots, x_t$: i.e., $H_d^n(x_1), \ldots, H_d^n(x_t)$.
- Precompute the chain ends for $y_1, \ldots, y_t$.
- Sort the chain ends for $x_1, \ldots, x_t$ and the chain ends for $y_1, \ldots, y_t$.
- If there is a collision, say a collision between the chain end for x_i and the chain end for y_j: recompute the chain for x_i, checking each chain element to see whether it is a preimage for y_j.

Finding t-images

Fix images $y_1, \ldots, y_t$. We build a reversible circuit that performs the following operations:

- Input a vector $(x_1, \ldots, x_t)$.
- Compute, in parallel, the chain ends for $x_1, \ldots, x_t$: i.e., $H_d^n(x_1), \ldots, H_d^n(x_t)$.
- Precompute the chain ends for $y_1, \ldots, y_t$.
- Sort the chain ends for $x_1, \ldots, x_t$ and the chain ends for $y_1, \ldots, y_t$.
- If there is a collision, say a collision between the chain end for x_i and the chain end for y_j: recompute the chain for x_i, checking each chain element to see whether it is a preimage for y_j.
- Output 0 if a preimage was found, otherwise 1.

Example

- Imagine AES-128;

Example

- Imagine AES-128;
- Consider $t = 2^{40}$ and $p = 2^{40}$, for this example.

Example

- Imagine AES-128;
- Consider $t = 2^{40}$ and $p = 2^{40}$, for this example.
- The probability to find is roughly $t^{5/2}/N$; For our example: $(2^{40})^{5/2}/2^{128} \approx 2^{-28}$.

Example

- Imagine AES-128;
- Consider $t = 2^{40}$ and $p = 2^{40}$, for this example.
- The probability to find is roughly $t^{5/2}/N$; For our example: $(2^{40})^{5/2}/2^{128} \approx 2^{-28}$.
- Each processor is going to use $\sqrt{N/pt^{3/2}}$ iterations;
- $\sqrt{2^{128}/2^{40}(2^{40})^{3/2}} \approx \sqrt{2^{128}/2^{100}}$

Example

- Imagine AES-128;
- Consider $t = 2^{40}$ and $p = 2^{40}$, for this example.
- The probability to find is roughly $t^{5/2}/N$; For our example: $(2^{40})^{5/2}/2^{128} \approx 2^{-28}$.
- Each processor is going to use $\sqrt{N/pt^{3/2}}$ iterations;
- $\sqrt{2^{128}/2^{40}(2^{40})^{3/2}} \approx \sqrt{2^{128}/2^{100}}$
  $= \sqrt{2^{28}} = 2^{14}$ iterations.

Conclusion

Conclusion:

- Circuit uses $O(a + tb + t(\log t)^2)$ ancillas;
- Depth of $O(\sqrt{N/pt^{1/2}}(gt^{\epsilon/2} + (\log t)^2 \log b))$;
- Approximately $\sqrt{N/pt^{3/2}}$ iterations.
- Create the circuit using quantum simulator for AES;
  - We already implemented using libquantum; One round of AES with $11,100$ gates;

[4] (libquantum instead of LiQ$Ui\,|\rangle$);

[4] Applying Grover's algorithm to AES: quantum resource estimates Grassl, Markus and Langenberg, Brandon and Roetteler, Martin and Steinwandt, Rainer

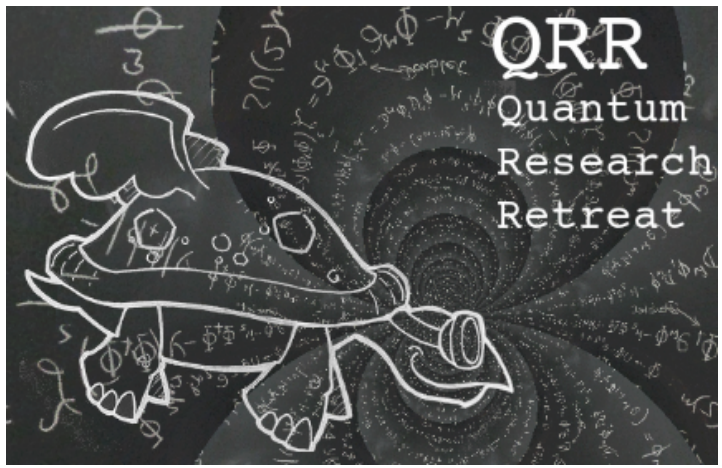
Outreach

- This work was at SAC 2017;
- We gave a talk at CWG (Crypto working group) reaching the Dutch community;
- We gave a talk at Quantum Cryptanalysis Seminar in Dagstuhl (Reaching the scientific community);
- Ei/ψ - Security in times of surveillance (General Public);

What's next?

- Check for the real number of qubits/gates giving an implementation;
- Change libquantum for "Big Integer";
- Implement the work from "Quantum resources estimates for ECC"[5];
- Finish the side channel attacks on ECC (work with Riscure);

- Quantum Research Retreat (QRR) in Eindhoven (mid December, https://cryptme.in/events/);



[5] Quantum resource estimates for computing elliptic curve discrete logarithms
Martin Roetteler, Michael Naehrig, Krysta M. Svore, Kristin Lauter

Questions

Thank you for your attention.
Questions?

gustavo@cryptme.in