QFactory from Learning With Errors

Lattice Coding and Crypto Meeting
8 May 2019

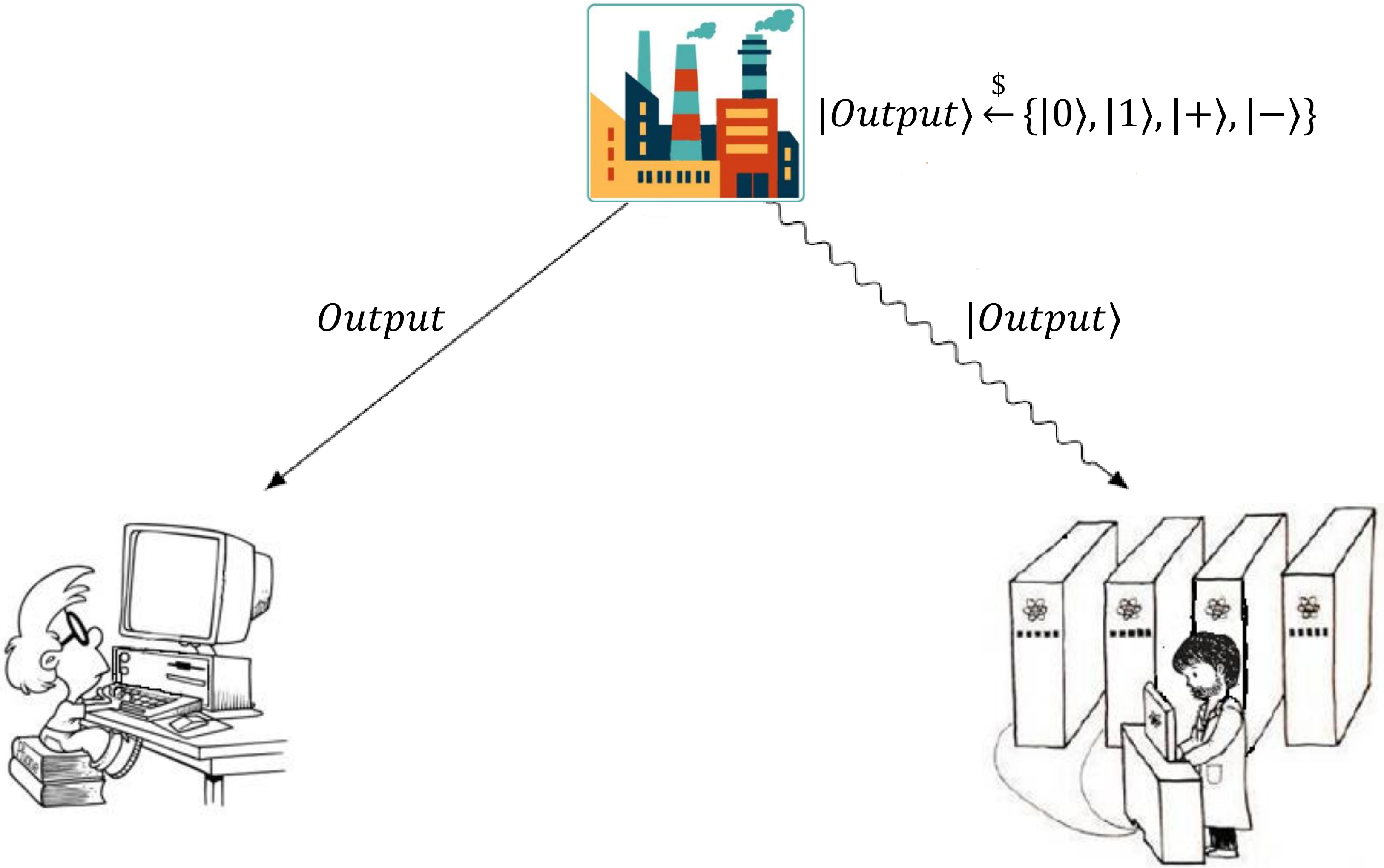
Alexandru Cojocaru
University of Edinburgh

Overview

- Part 1: Malicious QFactory
  - Functionality
  - Required assumptions
  - Protocol description
  - Security
  - Protocol Extensions (e.g. verification)
- Part 2: Functions implementation
  - QHBC QFactory functions
  - Malicious QFactory functions

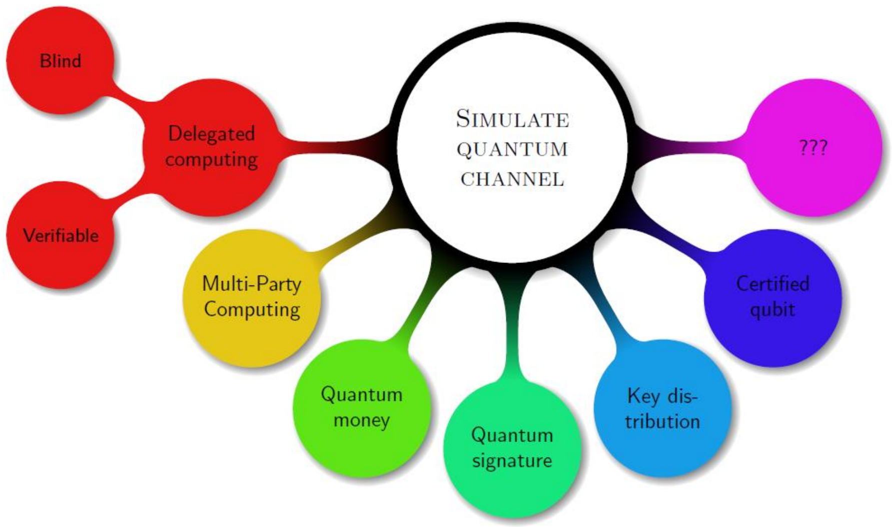
II. Classical delegation of secret qubits against Malicious Adversaries
or
Malicious 4-states QFactory

Malicious 4-states QFactory functionality



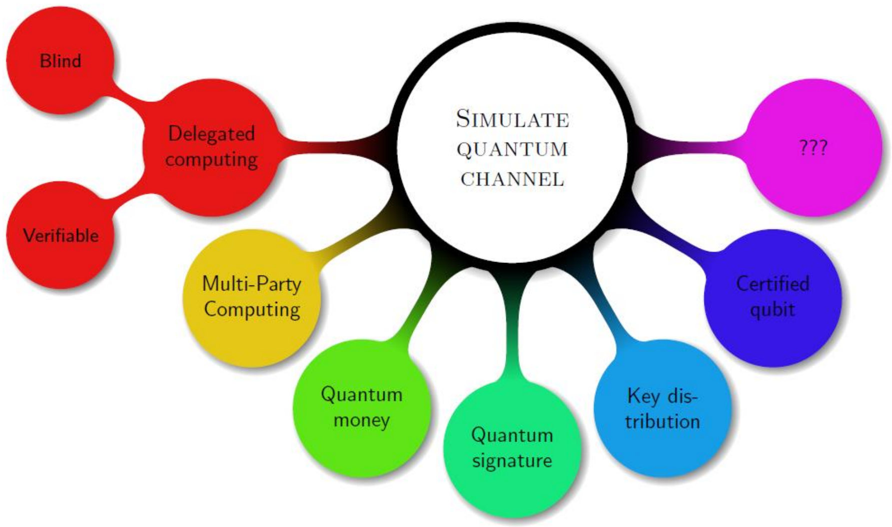
Motivation

There exist protocols for most of these applications where quantum communication only consists of the qubits $|0\rangle, |1\rangle, |+\rangle, |-\rangle$

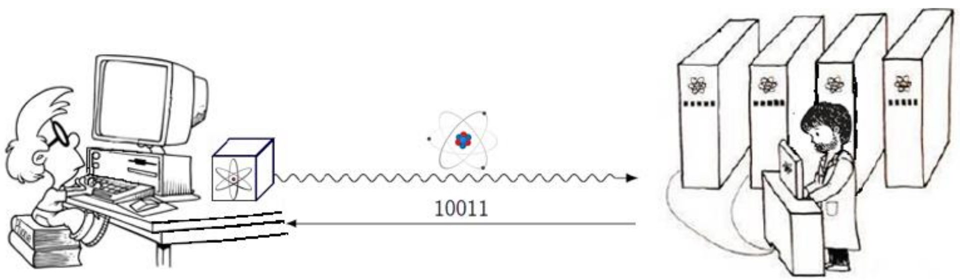


Motivation

There exist protocols for most of these applications where quantum communication only consists of the qubits $|0\rangle, |1\rangle, |+\rangle, |-\rangle$

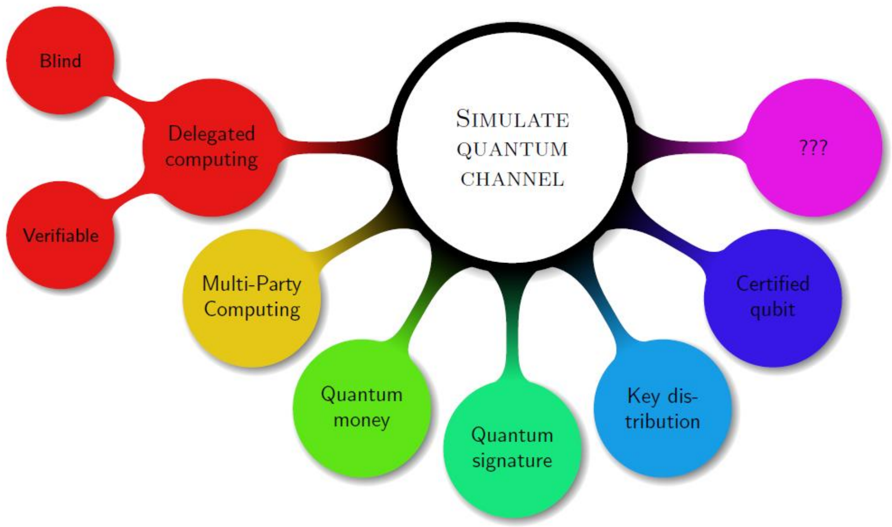


Functionality of Malicious 4-states QFactory $\Rightarrow$ classical delegation of quantum computation (against malicious adversaries)

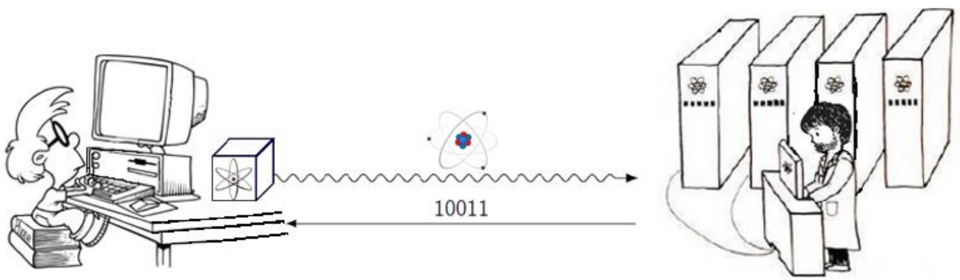


Motivation

There exist protocols for most of these applications where quantum communication only consists of the qubits $|0\rangle, |1\rangle, |+\rangle, |-\rangle$



Functionality of Malicious 4-states QFactory $\Rightarrow$ classical delegation of quantum computation (against malicious adversaries)
as long as the basis of qubits is hidden from any adversary



Malicious 4-states QFactory Required Assumptions

Functions $\{f_k\}$

One-way

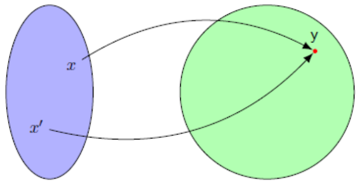
This function is hard to invert.

2-Regular

2 preimages for any element in $Im(f_k)$

Trapdoor

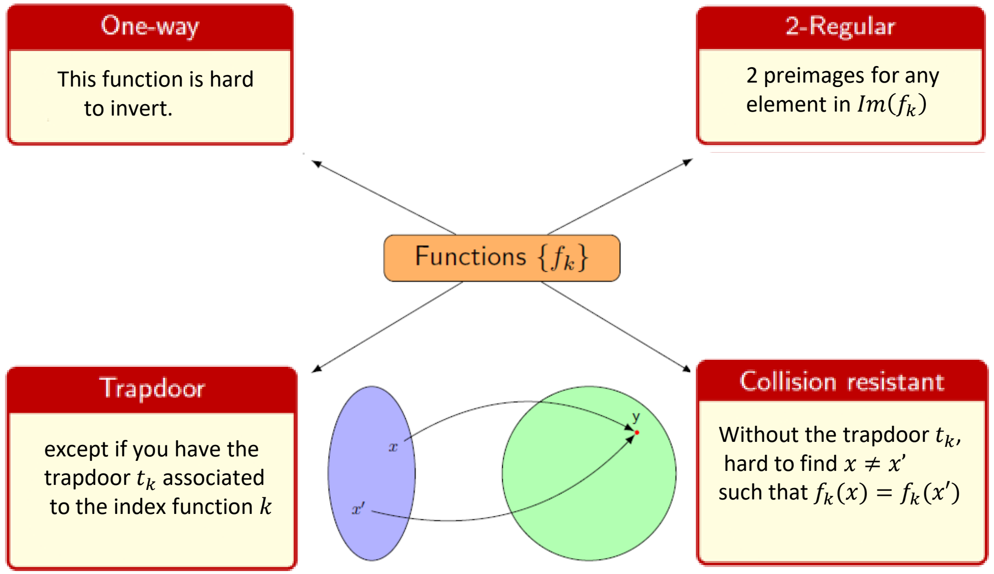
except if you have the trapdoor t_k associated to the index function k



Collision resistant

Without the trapdoor t_k, hard to find $x \neq x'$ such that $f_k(x) = f_k(x')$

Malicious 4-states QFactory Required Assumptions



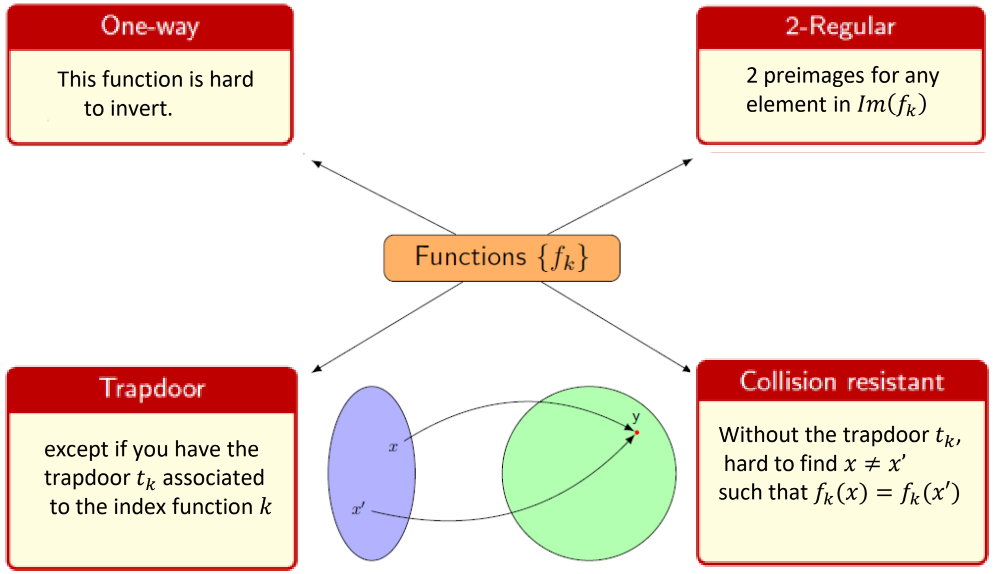
g_k: $D \rightarrow R$ injective, homomorphic, quantum-safe, trapdoor one-way;

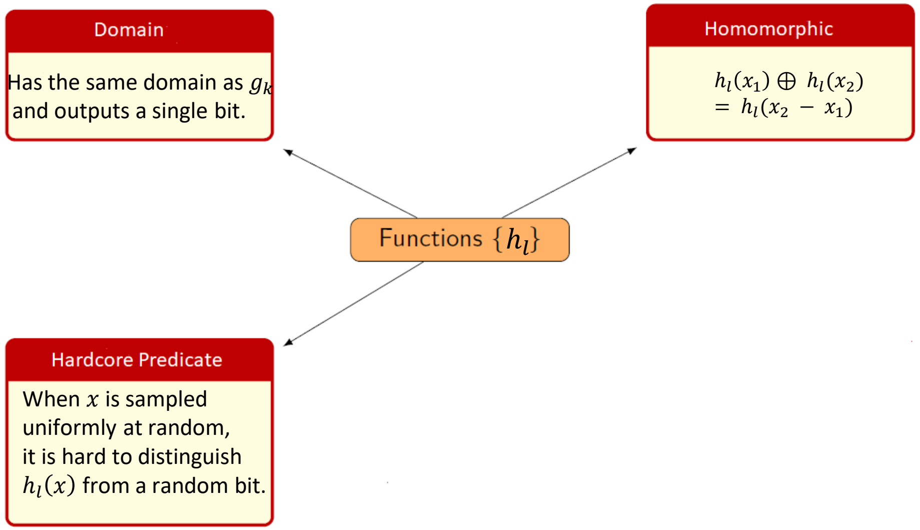
$$f_k : D \times \{0, 1\} \rightarrow R$$

$$f_k(x, c) = \begin{cases} g_k(x), & if\ c = 0 \\ g_k(x) \ \star g_k(x_0) = g_k(x + x_0), & if\ c = 1 \end{cases}$$

where x_0 is chosen by the Client at random from the domain of g_k

Malicious 4-states QFactory Required Assumptions





g_k: $D \to R$ injective, homomorphic, quantum-safe, trapdoor one-way;

$$f_k : D \times \{0,1\} \to R$$

$$f_k(x,c) = \begin{cases} g_k(x), & if\ c = 0 \\ g_k(x) \star g_k(x_0) = g_k(x + x_0), & if\ c = 1 \end{cases}$$

where x_0 is chosen by the Client at random from the domain of g_k

Malicious 4-states QFactory Protocol

$Choose\ (k, t_k)$
$Choose\ l$

Malicious 4-states QFactory Protocol

Choose (k, t_k)
Choose l

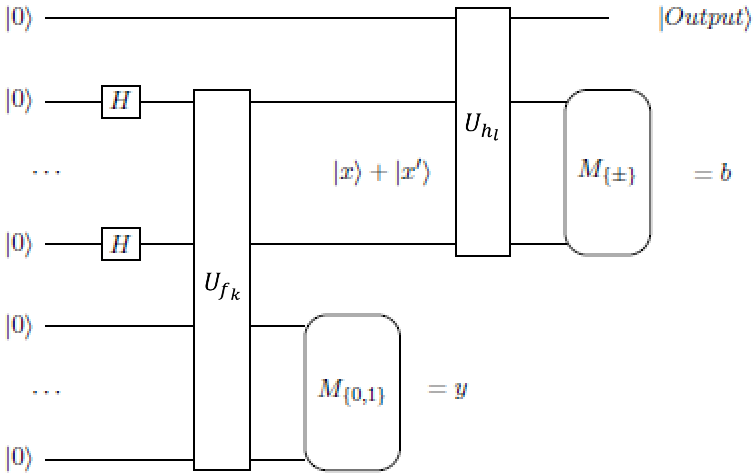
$$k, l \longrightarrow$$

Malicious 4-states QFactory Protocol

Choose (k, t_k)
Choose l

k, l →

Compute the circuit



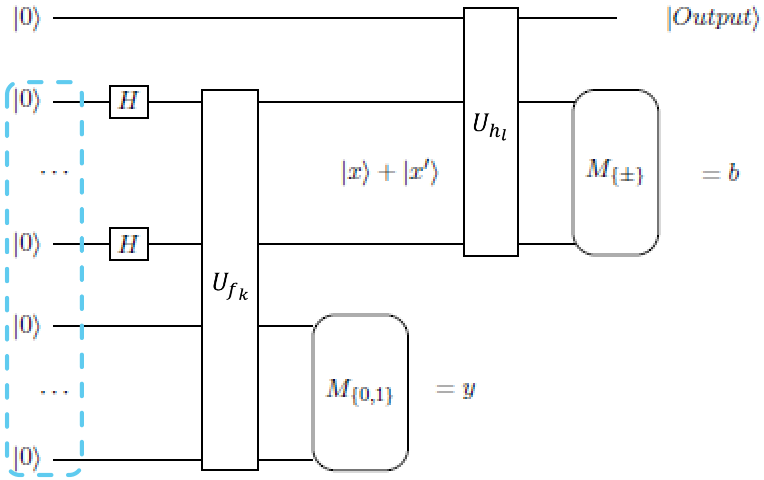
Malicious 4-states QFactory Protocol

$|0^n\rangle \, |0^m\rangle$

Choose (k, t_k)
Choose l

k, l →

Compute
the circuit



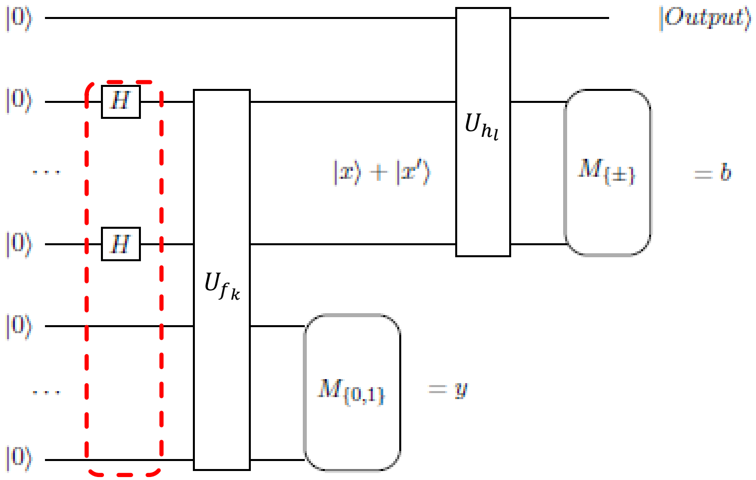
Malicious 4-states QFactory Protocol

$|0^n\rangle\,|0^m\rangle \rightarrow \sum_{x\in Dom(f_k)}|x\rangle|0^m\rangle$

Choose (k, t_k)
Choose l

k, l

Compute the circuit



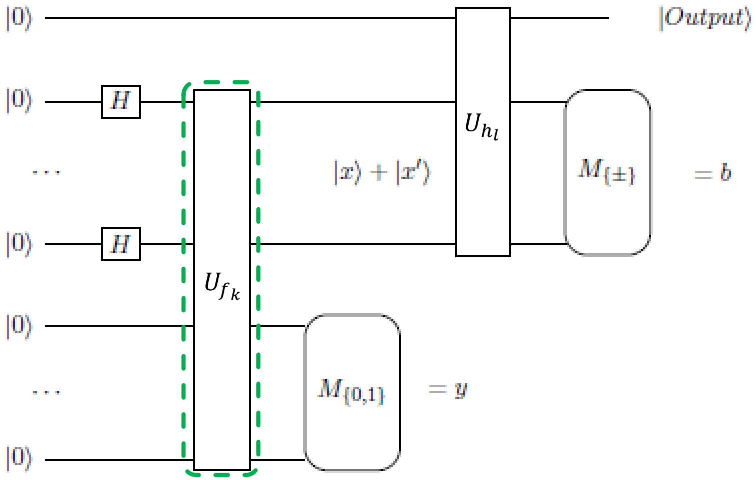
Malicious 4-states QFactory Protocol

$|0^n\rangle\,|0^m\rangle \rightarrow \sum_{x\in Dom(f_k)}|x\rangle|0^m\rangle \rightarrow \sum_{x\in Dom(f_k)}|x\rangle|f(x)\rangle$

Choose (k, t_k)
Choose l

k, l →

Compute the circuit



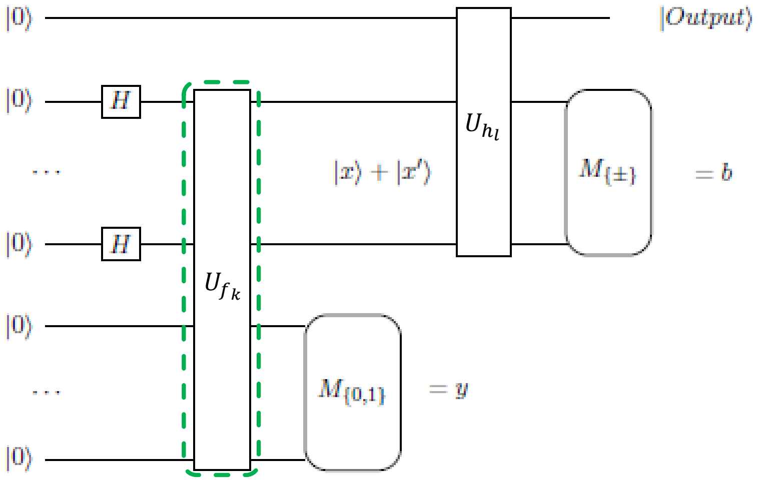
Malicious 4-states QFactory Protocol

$|0^n\rangle\,|0^m\rangle \rightarrow \sum_{x\in Dom(f_k)}|x\rangle|0^m\rangle \rightarrow \sum_{x\in Dom(f_k)}|x\rangle|f(x)\rangle = \sum_{y\in Im(f_k)}(|x\rangle + |x'\rangle)\otimes|y\rangle$

Choose (k, t_k)
Choose l

k, l

Compute the circuit



Malicious 4-states QFactory Protocol

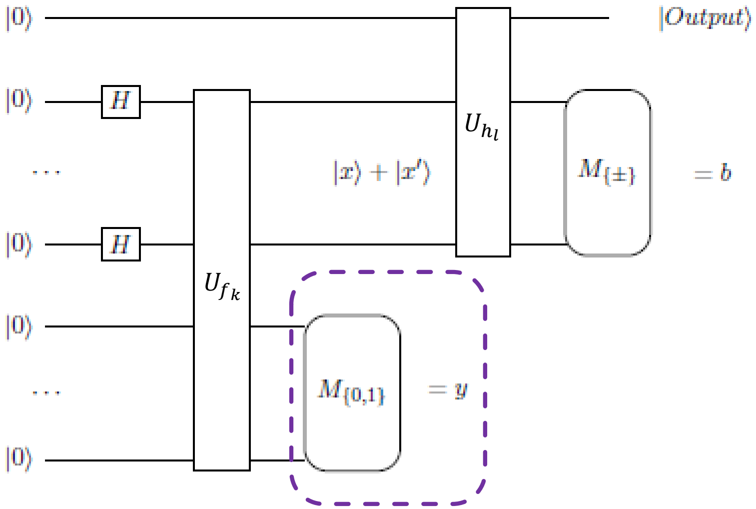
$$|0^n\rangle\,|0^m\rangle \rightarrow \sum_{x\in Dom(f_k)} |x\rangle|0^m\rangle \rightarrow \sum_{x\in Dom(f_k)} |x\rangle|f(x)\rangle = \sum_{y\in Im(f_k)} (|x\rangle + |x'\rangle)\otimes|y\rangle \rightarrow (|x\rangle + |x'\rangle)\otimes|y\rangle = (|z\rangle|0\rangle + |z'\rangle|1\rangle)\otimes|y\rangle$$

$$x = (z, 0) \quad x' = (z', 1)$$

Choose (k, t_k)
Choose l

$$\xrightarrow{\quad k,\, l \quad}$$

Compute the circuit



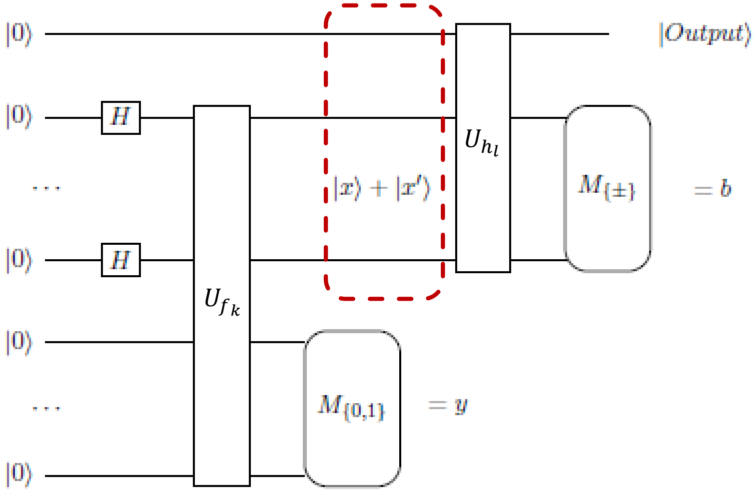
Malicious 4-states QFactory Protocol

$$|0^n\rangle\,|0^m\rangle \to \sum_{x\in Dom(f_k)} |x\rangle|0^m\rangle \to \sum_{x\in Dom(f_k)} |x\rangle|f(x)\rangle = \sum_{y\in Im(f_k)} (|x\rangle + |x'\rangle) \otimes |y\rangle \to (|x\rangle + |x'\rangle) \otimes |y\rangle = (|z\rangle|0\rangle + |z'\rangle|1\rangle) \otimes |y\rangle \to (|z\rangle|0\rangle|0\rangle + |z'\rangle|1\rangle|0\rangle)$$

Choose (k, t_k)
Choose l

k, l →

Compute the circuit



Malicious 4-states QFactory Protocol

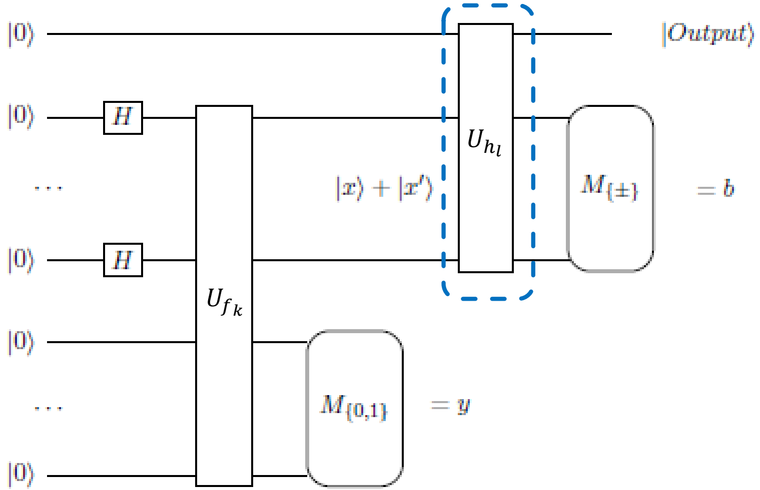
$$\sum_{x\in Dom(f_k)} |x\rangle|f(x)\rangle = \sum_{y\in Im(f_k)} (|x\rangle + |x'\rangle)\otimes|y\rangle \rightarrow (|x\rangle + |x'\rangle)\otimes|y\rangle = (|z\rangle|0\rangle + |z'\rangle|1\rangle)\otimes|y\rangle \rightarrow (|z\rangle|0\rangle|0\rangle + |z'\rangle|1\rangle|0\rangle) \rightarrow |z\rangle|0\rangle|h(z)\rangle + |z'\rangle|1\rangle|h(z')\rangle$$

Choose (k, t_k)
Choose l

k, l →

Compute the circuit

$$|z\rangle|c\rangle|0\rangle \xrightarrow{\widetilde{U_h}} |z\rangle\,|c\rangle\,|h(z)\,\rangle$$



Malicious 4-states QFactory Protocol

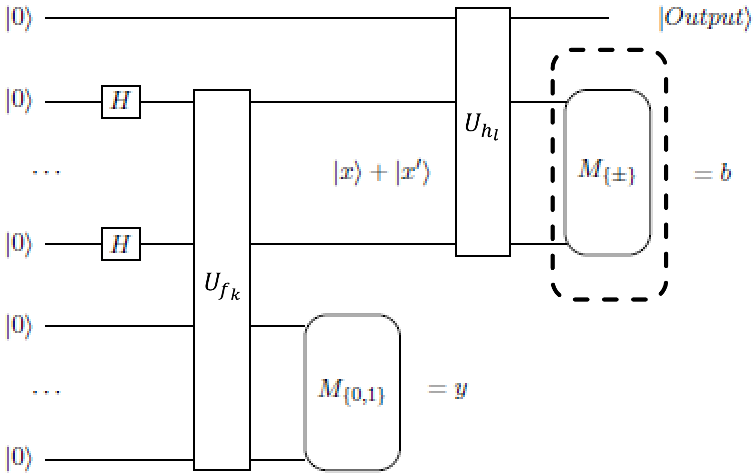
$$\sum_{x\in Dom(f_k)} |x\rangle|f(x)\rangle = \sum_{y\in Im(f_k)} (|x\rangle + |x'\rangle)\otimes|y\rangle \rightarrow (|x\rangle + |x'\rangle)\otimes|y\rangle = (|z\rangle|0\rangle + |z'\rangle|1\rangle)\otimes|y\rangle \rightarrow (|z\rangle|0\rangle|0\rangle + |z'\rangle|1\rangle|0\rangle) \rightarrow |z\rangle|0\rangle|h(z)\rangle + |z'\rangle|1\rangle|h(z')\rangle \Rightarrow |\boldsymbol{Output}\rangle$$

Choose (k, t_k)
Choose l

k, l

Compute the circuit

$$|z\rangle|c\rangle|0\rangle \xrightarrow{\widetilde{U_h}} |z\rangle\,|c\rangle\,|h(z)\rangle$$



Malicious 4-states QFactory Protocol

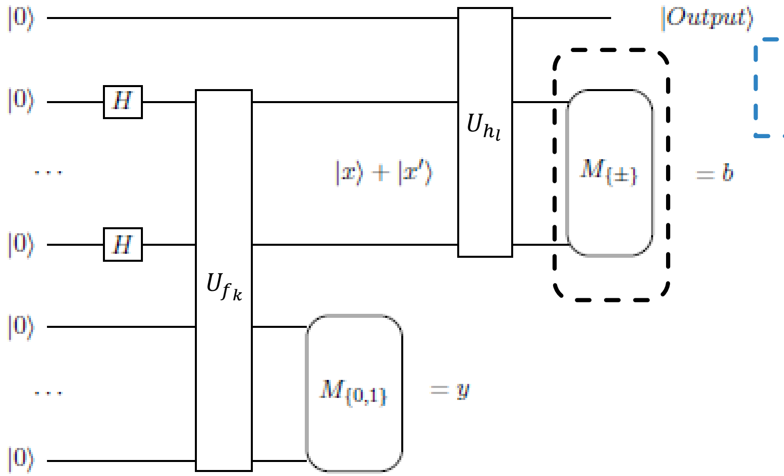
$$\sum_{x \in Dom(f_k)} |x\rangle|f(x)\rangle = \sum_{y \in Im(f_k)} (|x\rangle + |x'\rangle) \otimes |y\rangle \rightarrow (|x\rangle + |x'\rangle) \otimes |y\rangle = (|z\rangle|0\rangle + |z'\rangle|1\rangle) \otimes |y\rangle \rightarrow (|z\rangle|0\rangle|0\rangle + |z'\rangle|1\rangle|0\rangle) \rightarrow |z\rangle|0\rangle|h(z)\rangle + |z'\rangle|1\rangle|h(z')\rangle \Rightarrow |\boldsymbol{Output}\rangle$$

Choose (k, t_k)
Choose l

$$\xrightarrow{\quad k,\, l \quad}$$

Compute the circuit

$$|z\rangle|c\rangle|0\rangle \overset{\widetilde{U_h}}{\rightarrow} |z\rangle\,|c\rangle\,|h(z)\,\rangle$$



$|Output\rangle \in \{|0\rangle, |1\rangle, |+\rangle, |-\rangle\}$

Malicious 4-states QFactory Protocol

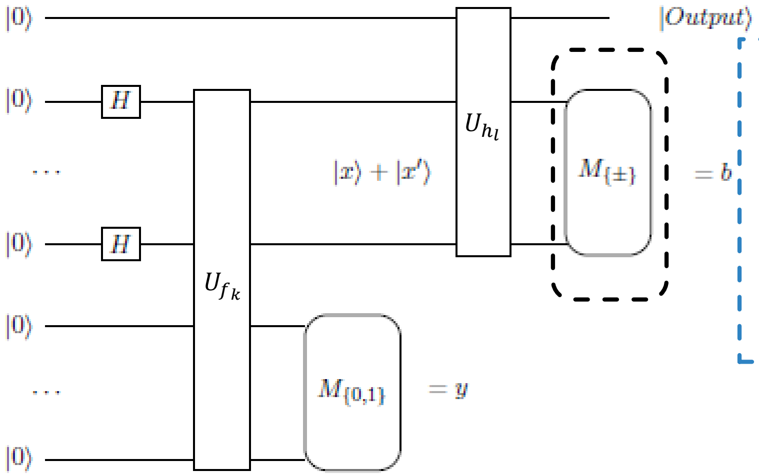
$$\sum_{x\in Dom(f_k)} |x\rangle|f(x)\rangle = \sum_{y\in Im(f_k)} (|x\rangle+|x'\rangle)\otimes|y\rangle \rightarrow (|x\rangle+|x'\rangle)\otimes|y\rangle = (|z\rangle|0\rangle+|z'\rangle|1\rangle)\otimes|y\rangle \rightarrow (|z\rangle|0\rangle|0\rangle+|z'\rangle|1\rangle|0\rangle) \rightarrow |z\rangle|0\rangle|h(z)\rangle+|z'\rangle|1\rangle|h(z')\rangle \Rightarrow |\boldsymbol{Output}\rangle$$

Choose (k, t_k)
Choose l

$$k, l \longrightarrow$$

Compute the circuit

$$|z\rangle|c\rangle|0\rangle \xrightarrow{\widetilde{U_h}} |z\rangle\,|c\rangle\,|h(z)\,\rangle$$



$|Output\rangle \in \{|0\rangle, |1\rangle, |+\rangle, |-\rangle\}$

$|Output\rangle = H^{B_1}X^{B_2}|0\rangle$

$B_1 = h(z) \oplus h(z')$

$B_2 = \{[\sum(x_i \oplus x_i') \cdot b_i] mod\ 2 \cdot B_1\} \oplus [h(z) \cdot (1 \oplus B_1)]$

$x = (z, 0)$
$x' = (z', 1)$

Malicious 4-states QFactory Protocol

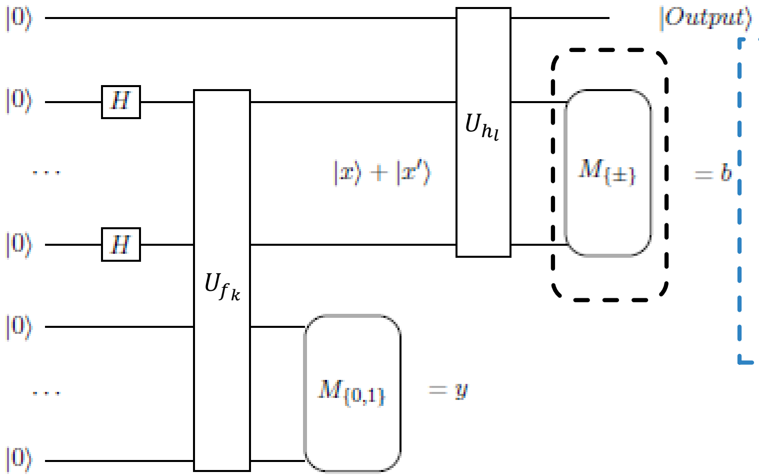
$$\sum_{x \in Dom(f_k)} |x\rangle|f(x)\rangle = \sum_{y \in Im(f_k)} (|x\rangle + |x'\rangle) \otimes |y\rangle \rightarrow (|x\rangle + |x'\rangle) \otimes |y\rangle = (|z\rangle|0\rangle + |z'\rangle|1\rangle) \otimes |y\rangle \rightarrow (|z\rangle|0\rangle|0\rangle + |z'\rangle|1\rangle|0\rangle) \rightarrow |z\rangle|0\rangle|h(z)\rangle + |z'\rangle|1\rangle|h(z')\rangle \Rightarrow |\boldsymbol{Output}\rangle$$

Choose (k, t_k)
Choose l

k, l

Compute the circuit

$$|z\rangle|c\rangle|0\rangle \xrightarrow{\widetilde{U_h}} |z\rangle\,|c\rangle\,|h(z)\,\rangle$$



$|Output\rangle \in \{|0\rangle, |1\rangle, |+\rangle, |-\rangle\}$

$|Output\rangle = H^{B_1} X^{B_2} |0\rangle$

$B_1 = h(z) \oplus h(z')$

$B_2 = \{[\sum (x_i \oplus x_i') \cdot b_i] mod\ 2 \cdot B_1\} \oplus [h(z) \cdot (1 \oplus B_1)]$

$$x = (z, 0)$$
$$x' = (z', 1)$$

Produces $|\boldsymbol{Output}\rangle$

Malicious 4-states QFactory Protocol

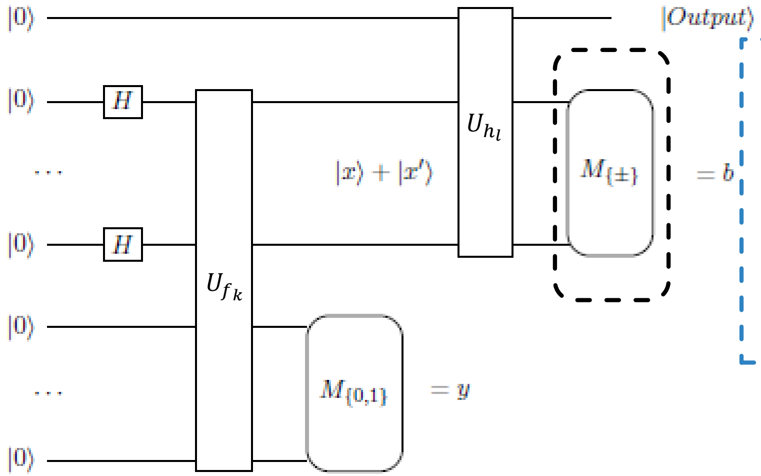
$$\sum_{x\in Dom(f_k)} |x\rangle|f(x)\rangle = \sum_{y\in Im(f_k)} (|x\rangle+|x'\rangle)\otimes|y\rangle \to (|x\rangle+|x'\rangle)\otimes|y\rangle = (|z\rangle|0\rangle+|z'\rangle|1\rangle)\otimes|y\rangle \to (|z\rangle|0\rangle|0\rangle+|z'\rangle|1\rangle|0\rangle) \to |z\rangle|0\rangle|h(z)\rangle+|z'\rangle|1\rangle|h(z')\rangle \Rightarrow |\boldsymbol{Output}\rangle$$

Choose (k, t_k)
Choose l

k, l →

Compute the circuit

$$|z\rangle|c\rangle|0\rangle \xrightarrow{\widetilde{U_h}} |z\rangle\,|c\rangle\,|h(z)\,\rangle$$



$|Output\rangle \in \{|0\rangle, |1\rangle, |+\rangle, |-\rangle\}$

$|Output\rangle = H^{B_1}X^{B_2}|0\rangle$

$B_1 = h(z) \oplus h(z')$

$B_2 = \{[\sum(x_i \oplus x_i') \cdot b_i] mod\ 2 \cdot B_1\} \oplus [h(z)\cdot(1 \oplus B_1)]$

$x = (z, 0)$
$x' = (z', 1)$

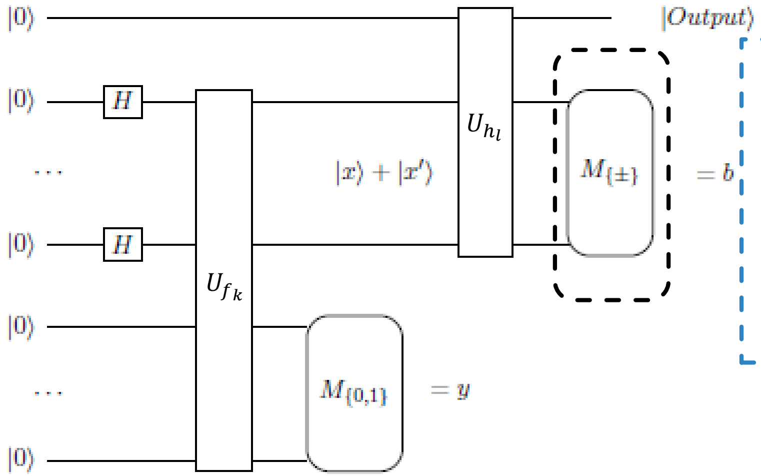
Produces $|\boldsymbol{Output}\rangle$

← y, b

Malicious 4-states QFactory Protocol

$$\sum_{x \in Dom(f_k)} |x\rangle|f(x)\rangle = \sum_{y \in Im(f_k)} (|x\rangle + |x'\rangle) \otimes |y\rangle \rightarrow (|x\rangle + |x'\rangle) \otimes |y\rangle = (|z\rangle|0\rangle + |z'\rangle|1\rangle) \otimes |y\rangle \rightarrow (|z\rangle|0\rangle|0\rangle + |z'\rangle|1\rangle|0\rangle) \rightarrow |z\rangle|0\rangle|h(z)\rangle + |z'\rangle|1\rangle|h(z')\rangle \Rightarrow |\boldsymbol{Output}\rangle$$

$$|z\rangle|c\rangle|0\rangle \overset{\widetilde{U_h}}{\rightarrow} |z\rangle\, |c\rangle\, |h(z)\,\rangle$$



$|Output\rangle \in \{|0\rangle, |1\rangle, |+\rangle, |-\rangle\}$

$|Output\rangle = H^{B_1} X^{B_2} |0\rangle$

$B_1 = h(z) \oplus h(z')$

$B_2 = \{[\sum (x_i \oplus x_i') \cdot b_i] mod\ 2 \cdot B_1\} \oplus [h(z) \cdot (1 \oplus B_1)]$

$x = (z, 0)$
$x' = (z', 1)$

Choose (k, t_k)
Choose l

k, l →

Compute the circuit

Produces $|Output\rangle$

← y, b

$(x, x') = Inv(t_k, y)$

Compute B_1, B_2

Malicious 4-states QFactory Protocol

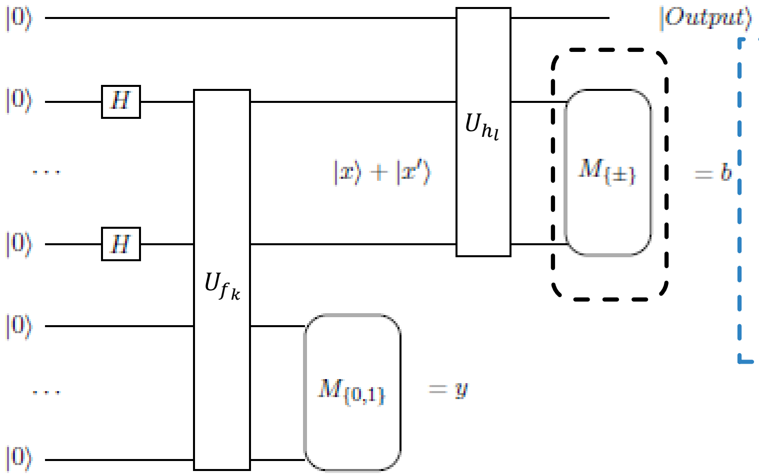
$$\sum_{x \in Dom(f_k)} |x\rangle|f(x)\rangle = \sum_{y \in Im(f_k)} (|x\rangle + |x'\rangle) \otimes |y\rangle \rightarrow (|x\rangle + |x'\rangle) \otimes |y\rangle = (|z\rangle|0\rangle + |z'\rangle|1\rangle) \otimes |y\rangle \rightarrow (|z\rangle|0\rangle|0\rangle + |z'\rangle|1\rangle|0\rangle) \rightarrow |z\rangle|0\rangle|h(z)\rangle + |z'\rangle|1\rangle|h(z')\rangle \Rightarrow |\boldsymbol{Output}\rangle$$

Choose (k, t_k)
Choose l

k, l

Compute the circuit

$$|z\rangle|c\rangle|0\rangle \xrightarrow{\widetilde{U_h}} |z\rangle\,|c\rangle\,|h(z)\,\rangle$$



$|Output\rangle \in \{|0\rangle, |1\rangle, |+\rangle, |-\rangle\}$

$|Output\rangle = H^{B_1}X^{B_2}|0\rangle$

$B_1 = h(z) \oplus h(z')$

$B_2 = \{[\sum(x_i \oplus x_i') \cdot b_i] mod\ 2 \cdot B_1\} \oplus [h(z) \cdot (1 \oplus B_1)]$

$x = (z, 0)$
$x' = (z', 1)$

Produces $|\boldsymbol{Output}\rangle$

y, b

$(x, x') = Inv(t_k, y)$

Compute B_1, B_2

Gets Output

Security (in the quantum malicious setting)

- $|Output\rangle = H^{B_1} X^{B_2} |0\rangle$
- B_1 = the basis bit of $|Output\rangle$
- If $B_1 = 0$ then $|Output\rangle \in \{|0\rangle, |1\rangle\}$ and if $B_1 = 1$ then $|Output\rangle \in \{|+\rangle, |-\rangle\}$

Security

- Blindness of the basis B_1 of $|Output\rangle$ against malicious adversaries.
- **Theorem**: No matter what Bob does, he cannot determine B_1.

- Server cannot do better than a random guess: B_1 is a **hard-core predicate** (wrt the function g);

Security (in the quantum malicious setting)

- B_1 is a hard-core predicate $\Longrightarrow$ **basis-blindness**

- The *basis-blindness* is the “maximum” security:

  - Even after an honest run we can at most guarantee basis blindness, but not full blindness about the output state:

    - $|Output\rangle \in \{|0\rangle, |1\rangle, |+\rangle, |-\rangle\}$
    - Then the Adversary can determine B_2 with probability at least $\frac{3}{4}$:
    - Makes a random guess $\widetilde{B_1}$ and then measures $|Output\rangle$ in the $\widetilde{B_1}$ basis, obtaining measurement outcome $\widetilde{B_2}$: if $\widetilde{B_1} = B_1$ then $\widetilde{B_2} = B_2$ with probability 1, otherwise $\widetilde{B_2} = B_2$ with probability $\frac{1}{2}$;

- Basis-blindness is proven to be sufficient for many secure computation protocols, e.g. *blind quantum computation* (UBQC protocol);

- Basis-blindness is required for classical verification of QFactory;
  $\Longrightarrow$ *classical verification of quantum computations*

Security (in the quantum malicious setting)

Recall:

$|Output\rangle \in \{|0\rangle, |1\rangle, |+\rangle, |-\rangle\}$

$$|Output\rangle = H^{B_1} X^{B_2} |0\rangle$$

$$B_1 = h(z) \oplus h(z')$$

$$B_2 = \{[\sum(x_i \oplus x_i') \cdot b_i] \bmod 2 \cdot B_1\} \oplus [h(z) \cdot (1 \oplus B_1)]$$

Security (in the quantum malicious setting)

Recall:

$|Output\rangle \in \{|0\rangle, |1\rangle, |+\rangle, |-\rangle\}$

$$|Output\rangle = H^{B_1} X^{B_2} |0\rangle$$

$$B_1 = h(z) \oplus h(z')$$

$$B_2 = \{[\sum (x_i \oplus x_i{}') \cdot b_i] \ mod\ 2 \cdot B_1\} \oplus [h(z) \cdot (1 \oplus B_1)]$$

B_1 = the basis bit of $|Output\rangle$

- $|Output\rangle \in \{|0\rangle, |1\rangle\} \Leftrightarrow B_1 = 0$
- $|Output\rangle \in \{|+\rangle, |-\rangle\} \Leftrightarrow B_1 = 1$

$\Rightarrow$ *Hiding* the basis equivalent to hiding $B_1 = h(z) \oplus h(z')$

Security (in the quantum malicious setting)

Recall:

$|Output\rangle \in \{|0\rangle, |1\rangle, |+\rangle, |-\rangle\}$

$$|Output\rangle = H^{B_1} X^{B_2} |0\rangle$$

$$B_1 = h(z) \oplus h(z')$$

$$B_2 = \{[\sum(x_i \oplus x_i') \cdot b_i] \, mod \; 2 \cdot B_1\} \oplus [h(z) \cdot (1 \oplus B_1)]$$

B_1 = the basis bit of $|Output\rangle$

- $|Output\rangle \in \{|0\rangle, |1\rangle\} \Leftrightarrow B_1 = 0$
- $|Output\rangle \in \{|+\rangle, |-\rangle\} \Leftrightarrow B_1 = 1$

$\Rightarrow$ *Hiding* the basis equivalent to hiding $B_1 = h(z) \oplus h(z')$

- Using the definition of f:

$$f(z, c) = g(z) + c \cdot g(z_0) \overset{homomorphic}{=} g(z + c \cdot z_0)$$

Security (in the quantum malicious setting)

Recall:

$|Output\rangle \in \{|0\rangle, |1\rangle, |+\rangle, |-\rangle\}$

$$|Output\rangle = H^{B_1} X^{B_2} |0\rangle$$

$$B_1 = h(z) \oplus h(z')$$

$$B_2 = \{[\sum (x_i \oplus x_i') \cdot b_i] \; mod \; 2 \cdot B_1\} \oplus [h(z) \cdot (1 \oplus B_1)]$$

B_1 = the basis bit of $|Output\rangle$

- $|Output\rangle \in \{|0\rangle, |1\rangle\} \Leftrightarrow B_1 = 0$
- $|Output\rangle \in \{|+\rangle, |-\rangle\} \Leftrightarrow B_1 = 1$

$\Rightarrow$ *Hiding* the basis equivalent to hiding $B_1 = h(z) \oplus h(z')$

- Using the definition of f:

$$f(z,c) = g(z) + c \cdot g(z_0) \overset{homomorphic}{=} g(z + c \cdot z_0)$$

- g is *injective*, the 2 preimages of f are:

$$x = (z,0) \; and \; x' = (z + z_0, 1) \Rightarrow z' = z + z_0$$

Security (in the quantum malicious setting)

Recall:

$|Output\rangle \in \{|0\rangle, |1\rangle, |+\rangle, |-\rangle\}$

$$|Output\rangle = H^{B_1} X^{B_2} |0\rangle$$

$$B_1 = h(z) \oplus h(z')$$

$$B_2 = \{[\sum (x_i \oplus x_i') \cdot b_i] \bmod 2 \cdot B_1\} \oplus [h(z) \cdot (1 \oplus B_1)]$$

B_1 = the basis bit of $|Output\rangle$

- $|Output\rangle \in \{|0\rangle, |1\rangle\} \Leftrightarrow B_1 = 0$
- $|Output\rangle \in \{|+\rangle, |-\rangle\} \Leftrightarrow B_1 = 1$

$\Rightarrow$ *Hiding* the basis equivalent to hiding $B_1 = h(z) \oplus h(z')$

- Using the definition of f:

$$f(z, c) = g(z) + c \cdot g(z_0) \overset{homomorphic}{=} g(z + c \cdot z_0)$$

- g is *injective*, the 2 preimages of f are:

$$x = (z, 0) \text{ and } x' = (z + z_0, 1) \Rightarrow z' = z + z_0$$

- h is *homomorphic*:

$$B_1 = h(z) \oplus h(z') = h(z' - z) = h(z_0)$$

Security (in the quantum malicious setting)

Recall:

$|Output\rangle \in \{|0\rangle, |1\rangle, |+\rangle, |-\rangle\}$

$$|Output\rangle = H^{B_1}X^{B_2}|0\rangle$$

$$B_1 = h(z) \oplus h(z')$$

$$B_2 = \{[\sum(x_i \oplus x_i') \cdot b_i] \bmod 2 \cdot B_1\} \oplus [h(z) \cdot (1 \oplus B_1)]$$

B_1 = the basis bit of $|Output\rangle$

- $|Output\rangle \in \{|0\rangle, |1\rangle\} \Leftrightarrow B_1 = 0$
- $|Output\rangle \in \{|+\rangle, |-\rangle\} \Leftrightarrow B_1 = 1$

$\Rightarrow$ *Hiding* the basis equivalent to hiding $B_1 = h(z) \oplus h(z')$

- Using the definition of f:

$$f(z,c) = g(z) + c \cdot g(z_0) \overset{homomorphic}{=} g(z + c \cdot z_0)$$

- g is *injective*, the 2 preimages of f are:

$$x = (z, 0) \; and \; x' = (z + z_0, 1) \Rightarrow z' = z + z_0$$

- h is *homomorphic:*

$$B_1 = h(z) \oplus h(z') = h(z' - z) = h(z_0)$$

- h is *hardcore predicate:*

$$B_1 = h(z_0) \; is \; hidden$$

Security (in the quantum malicious setting)

Overview

- The client picks at random z_0 and then sends $K' = \left(K, g_K(z_0)\right)$ to the Server (as the public description of f)
- As the basis of the output qubit is $B_1 = h(z_0)$, then the basis is basically fixed by the Client at the very beginning of the protocol.
- The output basis depends only on the Client's random choice of z_0 and is independent of the Server's communication.
- Then, no matter how the Server deviates and no matter what are the messages (y, b) sent by Server, to prove that the basis $B_1 = h(z_0)$ is completely hidden from the Server, is *sufficient* to use that h is a hardcore predicate.

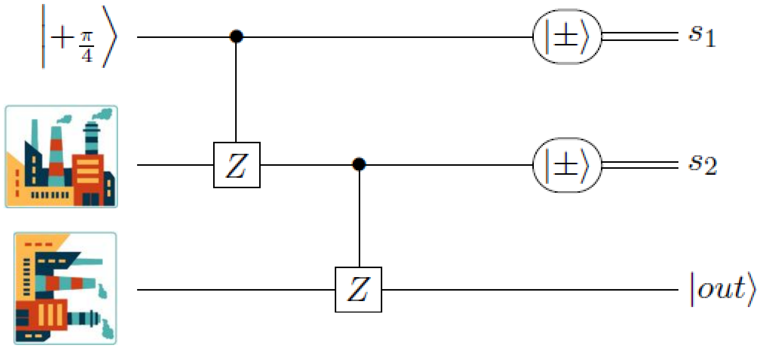
Extensions of QFactory

Malicious 8-states QFactory

- To use Malicious 4-states QFactory for applications where communication consists of $|+_\theta\rangle$, with $\theta \in \{0, \frac{\pi}{4}, \dots, \frac{7\pi}{4}\}$, we provide a gadget that achieves such a state from 2 outputs of Malicious 4-states QFactory.

Malicious 8-states QFactory

- To use Malicious 4-states QFactory for applications where communication consists of $|+_\theta\rangle$, with $\theta \in \{0, \frac{\pi}{4}, \dots, \frac{7\pi}{4}\}$, we provide a gadget that achieves such a state from 2 outputs of Malicious 4-states QFactory.

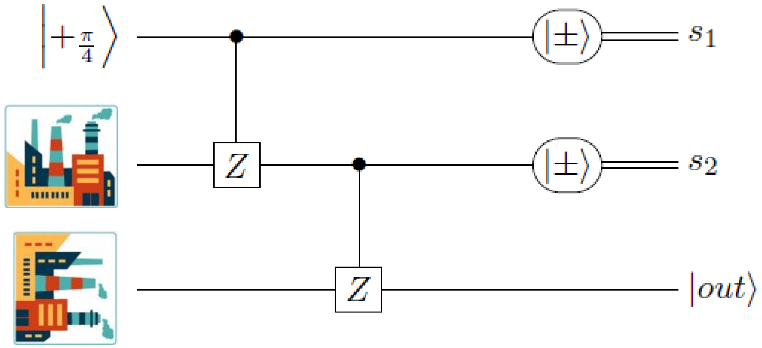


$$|out\rangle = R\left[L_1\pi + L_2\frac{\pi}{2} + L_3\frac{\pi}{4}\right]|+\rangle$$

$$L_3 = B_1$$
$$L_2 = B_1' \oplus [(B_2 \oplus s_2) \cdot B_1]$$
$$L_1 = B_2' \oplus B_2 \oplus [B_1 \cdot (s_1 \oplus s_2)]$$

Malicious 8-states QFactory

- To use Malicious 4-states QFactory for applications where communication consists of $|+_\theta\rangle$, with $\theta \in \{0, \frac{\pi}{4}, \dots, \frac{7\pi}{4}\}$, we provide a gadget that achieves such a state from 2 outputs of Malicious 4-states QFactory.



$$|out\rangle = R\left[L_1\pi + L_2\frac{\pi}{2} + L_3\frac{\pi}{4}\right]|+\rangle$$

$$L_3 = B_1$$
$$L_2 = B_1' \oplus [(B_2 \oplus s_2) \cdot B_1]$$
$$L_1 = B_2' \oplus B_2 \oplus [B_1 \cdot (s_1 \oplus s_2)]$$

- No information about the bases (L_2, L_3) of the new output state $|out\rangle$ is leaked:
  - We prove the basis blindness of the output of the gadget by a reduction to the

  basis-blindness of 1 of the 2 outputs of Malicious 4-states QFactory;

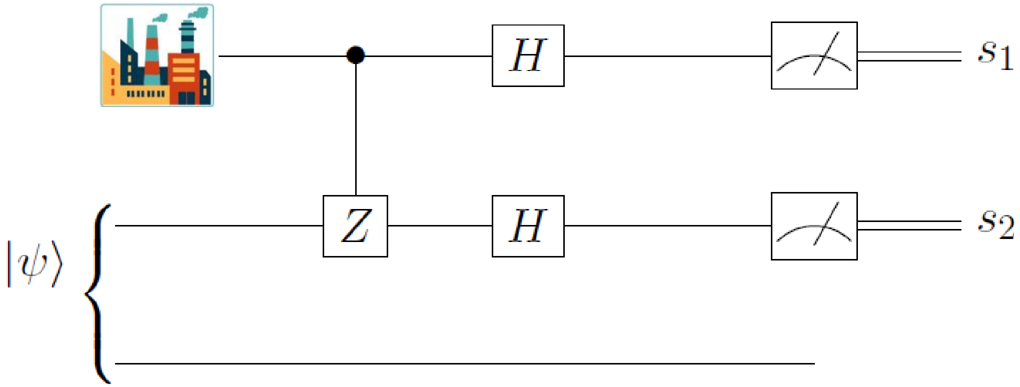
  If you could determine L_2 and L_3, then you would determine B_1 or $B_1{}'$.

Blind Measurements

- Perform a measurement on a first qubit of an arbitrary state $|\psi\rangle$ in such a way that the adversary is oblivious whether he is performing a measurement in 1 out of 2 possible basis (e.g. X or Z basis).
  - Useful for classical verification of quantum computations;
- Achieved using the following gadget:

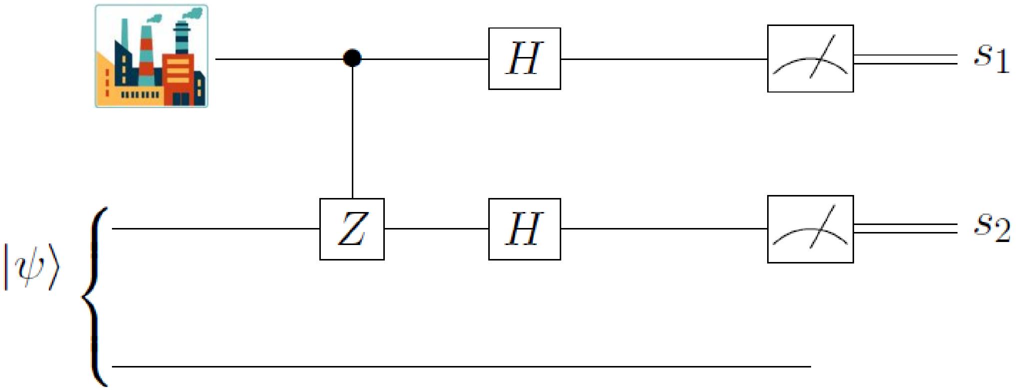
Blind Measurements

- Perform a measurement on a first qubit of an arbitrary state $|\psi\rangle$ in such a way that the adversary is oblivious whether he is performing a measurement in 1 out of 2 possible basis (e.g. X or Z basis).
  - Useful for classical verification of quantum computations (Mahadev FOCS18);
- Achieved using the following gadget:



Blind Measurements

- Perform a measurement on an arbitrary state $|\psi\rangle$ in such a way that the adversary is oblivious whether he is performing a measurement in 1 out of 2 possible basis (e.g. X or Z basis).
  - Useful for classical verification of quantum computations (Mahadev FOCS18);
- Achieved using the following gadget:



- No information about the basis of the measurement is leaked;
  - We prove the measurement blindness of the output of the gadget by a reduction to the basis-blindness of Malicious 4-states QFactory;

Classical verification of quantum computations

- Basis-blindness is not sufficient for verifiable blind quantum computation;
- To achieve verification, we combine Basis Blindness and *Self-Testing*;

Classical verification of quantum computations

- Basis-blindness is not sufficient for verifiable blind quantum computation;
- To achieve verification, we combine Basis Blindness and *Self-Testing*;
- Self-Testing
  - Given measurement statistics, classical parties are certain that some untrusted quantum states, that 2 **non-communicating** quantum parties share, are the states that the classical parties believe to have;
  - In our case, we replace the non-communication property with the basis-blindness condition;

Classical verification of quantum computations

$\{|0\rangle, |1\rangle, |+\rangle, |-\rangle\}$ $\longrightarrow$ $|+_\theta\rangle,\ \theta \in \{0, \frac{\pi}{4}, \ldots, \frac{7\pi}{4}\}$

4 states hidden bases

8 states hidden bases

Self-Testing

?

Verification

Classical verification of quantum computations

Verification Protocol

1. We repeat Malicious 8-states QFactory multiple times – independent runs;

Classical verification of quantum computations

Verification Protocol

1. We repeat Malicious 8-states QFactory multiple times – independent runs;
2. The Client chooses and announces a random fraction of the output qubits of these runs to use them for a test;

Classical verification of quantum computations

Verification Protocol

1. We repeat Malicious 8-states QFactory multiple times – independent runs;
2. The Client chooses and announces a random fraction of the output qubits of these runs to use them for a test;
3. The Server is instructed by the Client to measure the test qubits in random angles and sends the measurement results to the Client;

Classical verification of quantum computations

Verification Protocol

1. We repeat Malicious 8-states QFactory multiple times – independent runs;
2. The Client chooses and announces a random fraction of the output qubits of these runs to use them for a test;
3. The Server is instructed by the Client to measure the test qubits in random angles and sends the measurement results to the Client;
4. With the measurement results, the client knowing the basis of the test qubits and the measurement angles, he can check their statistics;

Classical verification of quantum computations

Verification Protocol

1. We repeat Malicious 8-states QFactory multiple times – independent runs;
2. The Client chooses and announces a random fraction of the output qubits of these runs to use them for a test;
3. The Server is instructed by the Client to measure the test qubits in random angles and sends the measurement results to the Client;
4. With the measurement results, the client knowing the basis of the test qubits and the measurement angles, he can check their statistics;
5. Since the server does not know the basis bits of these test states, he is unlikely to succeed in guessing the correct statistics unless he is honest.

QHBC QFactory
Function Construction

QHBC QFactory

Required Assumptions:

One-way

This function is hard to invert. . .

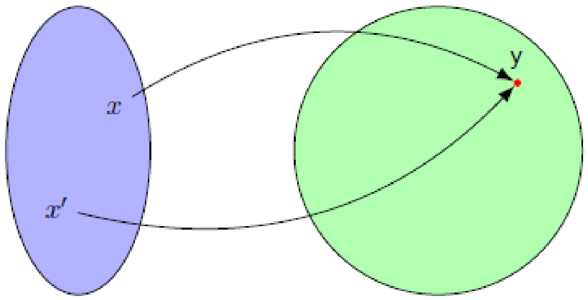
2-Regular

2 preimages for all elements in $Im(f_k)$

Functions $\{f_k\}$

Trapdoor

. . . except if you have the trapdoor t_k associated to the function index k.



Collision resistant

Without trapdoor t_k, hard to find $x \neq x'$ such that $f_k(x) = f_k(x')$

I. Function Constructions

- We propose 2 generic constructions, using:
  - A) A bijective, quantum-safe, trapdoor one-way function $g_k\colon D \to R$

$$f_{k'} : D \times \{0,1\} \to R$$

$$f_{k'}(x,c) = \begin{cases} g_{k_1}(x), & \text{if } c = 0 \\ g_{k_2}(x), & \text{if } c = 1 \end{cases}$$

$$(k_1, t_{k_1}) \leftarrow_{\$} Gen_{\mathcal{G}}(1^n)$$
$$(k_2, t_{k_2}) \leftarrow_{\$} Gen_{\mathcal{G}}(1^n)$$
$$k' := (k_1, k_2)$$
$$t'_k := (t_{k_1}, t_{k_2})$$

I. Function Constructions

- We propose 2 generic constructions, using:
  - A) A bijective, quantum-safe, trapdoor one-way function g_k: $D \to R$

$$f_{k'} : D \times \{0,1\} \to R$$

$$f_{k'}(x,c) = \begin{cases} g_{k_1}(x), & \text{if } c = 0 \\ g_{k_2}(x), & \text{if } c = 1 \end{cases}$$

$$(k_1, t_{k_1}) \leftarrow_\$ Gen_{\mathcal{G}}(1^n)$$
$$(k_2, t_{k_2}) \leftarrow_\$ Gen_{\mathcal{G}}(1^n)$$
$$k' := (k_1, k_2)$$
$$t'_k := (t_{k_1}, t_{k_2})$$

  - B) An injective, homomorphic, quantum-safe, trapdoor one-way function g_k: $D \to R$

$$f_{k'} : D \times \{0,1\} \to R$$

$$f_{k'}(x,c) = \begin{cases} g_k(x), & \text{if } c = 0 \\ g_k(x) \star g_k(x_0) = g_k(x + x_0) \quad , & \text{if } c = 1 \end{cases}$$

$$(k, t_k) \leftarrow_\$ Gen_{\mathcal{G}}(1^n)$$
$$x_0 \leftarrow_\$ D \setminus \{0\}$$
$$k' := (k, g_k(x_0))$$
$$t'_k := (t_k, x_0)$$

where x_0 is chosen by the Client at random from the domain of g_k

Injective, homomorphic, quantum-safe, trapdoor one-way function

Construction based on the Micciancio and Peikert trapdoor function – derived from the Learning With Errors problem:

$$g_K : \mathbb{Z}_q^n \times \chi^m \to \mathbb{Z}_q^m$$
$$g_K(s, e) = Ks \ + e \ mod \ q$$

$where \ K \leftarrow \mathbb{Z}_q^{m \times n}$ and $e_i \in \chi \ if \ |e_i| \leq \mu = \frac{q}{4}$

Homomorphic property

- $g_K(s,e) + g_K(s_0,e_0) \; mod \; q = (Ks + e + Ks_0 + e_0) \; mod \; q = g_K((s + s_0) \; mod \; q, e + e_0)$

Homomorphic property

- $g_K(s,e) + g_K(s_0,e_0) \; mod \; q = (Ks + e + Ks_0 + e_0) \; mod \; q = g_K\big((s + s_0) \; mod \; q, e + e_0\big)$
- Issue: domain of g_K imposes that each component of $e + e_0$ must be bounded by μ !
- Otherwise, we will just have 1 preimage;

Homomorphic property

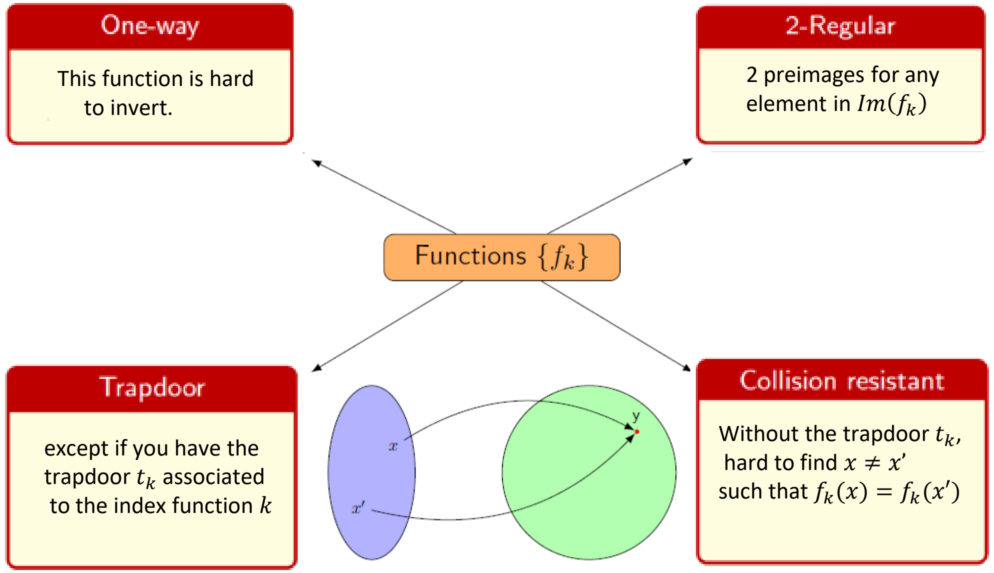
- $g_K(s,e) + g_K(s_0,e_0)\ mod\ q = (Ks + e + Ks_0 + e_0)\ mod\ q = g_K\big((s + s_0)\ mod\ q, e + e_0\big)$
- Issue: domain of g_K imposes that each component of $e + e_0$ must be bounded by μ !
- Otherwise, we will just have 1 preimage;
- To solve this:
  - We are sampling e_0 from a smaller set, such that when added with a random input e, the total noise $e + e_0$ is bounded by μ with high probability;
  - We showed that if e_0 is sampled such that it is bounded by $\mu' = \frac{\mu}{m}$, then $e + e_0$ lies in the domain of the function with constant probability $\Rightarrow$ f is 2-regular with constant probability
  - However, what we must show is that when e_0 is restricted to this smaller domain $g_K(s_0,e_0)$ is still hard to invert.

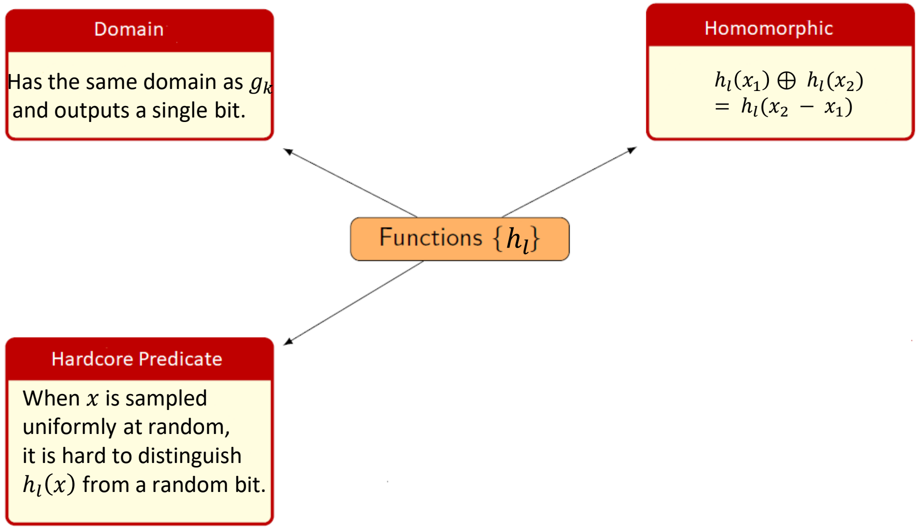
Homomorphic property

- $g_K(s,e) + g_K(s_0,e_0) \bmod q = (Ks + e + Ks_0 + e_0) \bmod q = g_K\big((s + s_0) \bmod q, e + e_0\big)$
- Issue: domain of g_K imposes that each component of $e + e_0$ must be bounded by μ !
- Otherwise, we will just have 1 preimage;
- To solve this:
  - We are sampling e_0 from a smaller set, such that when added with a random input e, the total noise $e + e_0$ is bounded by μ with high probability;
  - We showed that if e_0 is sampled such that it is bounded by $\mu' = \frac{\mu}{m}$, then $e + e_0$ lies in the domain of the function with constant probability $\Rightarrow$ f is 2-regular with constant probability
  - However, what we must show is that when e_0 is restricted to this smaller domain $g_K(s_0, e_0)$ is still hard to invert.
- Finally, we show there exists an explicit choice of parameters such that both g and the restriction of g to the domain of e_0 are one-way functions and such that all the other properties of g are preserved.

Malicious QFactory Function Construction

Malicious QFactory Required Assumptions





g_k: $D \rightarrow R$ injective, homomorphic, quantum-safe, trapdoor one-way;

$$f_k : D \times \{0,1\} \rightarrow R$$

$$f_k(x,c) = \begin{cases} g_k(x), & if\ c = 0 \\ g_k(x) \star g_k(x_0) = g_k(x + x_0), & if\ c = 1 \end{cases}$$

Malicious QFactory functions

- "QHBC" functions:

$$\bar{g}_K : \mathbb{Z}_q^n \times \chi^m \rightarrow \mathbb{Z}_q^m \qquad \bar{f}_{K'} : \mathbb{Z}_q^n \times \chi^m \times \{0,1\} \rightarrow \mathbb{Z}_q^m$$

$$K \xleftarrow{\$} \mathbb{Z}_q^{m \times n} \qquad K' = \left(K,\ \bar{g}_K(s_0,\ e_0)\right)$$

$$\bar{g}_K(s,e) = Ks + e \bmod q \qquad \bar{f}_{K'}(s,e,c) = \bar{g}_K(s,e) + c \cdot \bar{g}_K(s_0,\ e_0)$$

Malicious QFactory functions

- "QHBC" functions:

$$\bar{g}_K : \mathbb{Z}_q^n \times \chi^m \rightarrow \mathbb{Z}_q^m \qquad \bar{f}_{K'} : \mathbb{Z}_q^n \times \chi^m \times \{0,1\} \rightarrow \mathbb{Z}_q^m$$

$$K \overset{\$}{\leftarrow} \mathbb{Z}_q^{m \times n} \qquad K' = \left(K,\ \bar{g}_K(s_0,\ e_0)\right)$$

$$\bar{g}_K(s,e) = Ks + e \bmod q \qquad \bar{f}_{K'}(s,e,c) = \bar{g}_K(s,e) + c \cdot \bar{g}_K(s_0,\ e_0)$$

- "Malicious" functions:

$$g_K : \mathbb{Z}_q^n \times \chi^m \times \{0,1\} \rightarrow \mathbb{Z}_q^m \qquad f_{K'} : \mathbb{Z}_q^n \times \chi^m \times \{0,1\} \times \{0,1\} \rightarrow \mathbb{Z}_q^m$$

$$g_K(s,e,d) = \bar{g}_K(s,e) + d \cdot v \bmod q \qquad f_{K'}(s,e,d,c) = g_K(s,e,d) + c \cdot g_K(s_0,e_0,d_0)$$

where $v = \begin{pmatrix} \frac{q}{2} \\ 0 \\ \dots \\ 0 \end{pmatrix} \in \mathbb{Z}^m$.

Construction of the function h

- $g_K : \mathbb{Z}_q^n \times \chi^m \times \{0,1\} \rightarrow \mathbb{Z}_q^m$

$$g_K(s,e,d) = \bar{g}_K(s,e) + \boldsymbol{d} \cdot \boldsymbol{v} \; mod \; q = Ks + e + d \cdot \begin{pmatrix} \frac{q}{2} \\ 0 \\ \dots \\ 0 \end{pmatrix} mod \; q$$

Construction of the function h

- $g_K : \mathbb{Z}_q^n \times \chi^m \times \{0,1\} \to \mathbb{Z}_q^m$

$$g_K(s,e,d) = \bar{g}_K(s,e) + \boldsymbol{d} \cdot \boldsymbol{v} \; mod \; q = Ks + e + d \cdot \begin{pmatrix} \frac{q}{2} \\ 0 \\ \dots \\ 0 \end{pmatrix} mod \; q$$

- $h : \mathbb{Z}_q^n \times \chi^m \times \{0,1\} \to \{0,1\}$

$$h(s,e,d) = d$$

Construction of the function h

- $g_K : \mathbb{Z}_q^n \times \chi^m \times \{0,1\} \to \mathbb{Z}_q^m$

$$g_K(s,e,d) = \bar{g}_K(s,e) + d \cdot v \; mod \; q = Ks + e + d \cdot \begin{pmatrix} \frac{q}{2} \\ 0 \\ \ldots \\ 0 \end{pmatrix} mod \; q$$

- $h : \mathbb{Z}_q^n \times \chi^m \times \{0,1\} \to \{0,1\}$

$$h(s,e,d) = d$$

Properties of g

1. Homomorphic:

- $$g_K(s_1,e_1,d_1) + g_K(s_2,e_2,d_2) = \bar{g}_K(s_1,e_1) + d_1 \cdot v + \bar{g}_K(s_2,e_2) + d_2 \cdot v \; mod \; q = \bar{g}_K(s_1 + s_2 \; mod \; q, e_1 + e_2) + (d_1 + d_2) \cdot v \; mod \; q = \bar{g}_K(s_1 + s_2 \; mod \; q, e_1 + e_2, d_1 \oplus d_2)$$

Construction of the function h

- $g_K : \mathbb{Z}_q^n \times \chi^m \times \{0,1\} \to \mathbb{Z}_q^m$

$$g_K(s,e,d) = \bar{g}_K(s,e) + d \cdot v \; mod \; q = Ks + e + d \cdot \begin{pmatrix} \frac{q}{2} \\ 0 \\ \dots \\ 0 \end{pmatrix} mod \; q$$

- $h : \mathbb{Z}_q^n \times \chi^m \times \{0,1\} \to \{0,1\}$

$$h(s,e,d) = d$$

Properties of g

1. Homomorphic:

> $$g_K(s_1,e_1,d_1) + g_K(s_2,e_2,d_2) = \bar{g}_K(s_1,e_1) + d_1 \cdot v + \bar{g}_K(s_2,e_2) + d_2 \cdot v \;\; mod \; q =$$
> $$\bar{g}_K(s_1 + s_2 \; mod \; q \,, e_1 + e_2) + (d_1 + d_2) \cdot v \;\; mod \; q = \bar{g}_K(s_1 + s_2 \; mod \; q \,, e_1 + e_2, d_1 \oplus d_2)$$

2. One-way:

> $Reduction\ to\ the\ one-wayness\ of\ \bar{g}_K$:

$$To\ invert\ y = \bar{g}_K(s,e):$$
$$d \overset{\$}{\leftarrow} \{0,1\}$$
$$y' \leftarrow y + d \cdot v$$
$$(s',e',d') \leftarrow A_K(y')$$
$$return\ (s',e')$$

Construction of the function h

- $g_K : \mathbb{Z}_q^n \times \chi^m \times \{0,1\} \rightarrow \mathbb{Z}_q^m$

$$g_K(s,e,d) = \bar{g}_K(s,e) + d \cdot v \; mod\; q = Ks + e + d \cdot \begin{pmatrix} \frac{q}{2} \\ 0 \\ \dots \\ 0 \end{pmatrix} mod\; q$$

Properties of g

3. Injective:
   - $Suppose\ \exists\ (s_1, e_1, d_1), (s_2, e_2, d_2)\ s.t.\ g_K(s_1, e_1, d_1) = g_K(s_2, e_2, d_2)$
   - $\bar{g}_K(s_1, e_1) - \bar{g}_K(s_2, e_2) + (d_1 - d_2) \cdot v = 0\; mod\; q$
   - $If\ d_1 = d_2\ then\ \ \bar{g}_K(s_1, e_1) = \bar{g}_K(s_2, e_2) \Rightarrow s_1 = s_2\,, e_1 = e_2$ ✔

Construction of the function h

- $g_K : \mathbb{Z}_q^n \times \chi^m \times \{0,1\} \rightarrow \mathbb{Z}_q^m$

$$g_K(s,e,d) = \bar{g}_K(s,e) + \boldsymbol{d} \cdot v \; mod \; q = Ks + e + d \cdot \begin{pmatrix} \frac{q}{2} \\ 0 \\ \dots \\ 0 \end{pmatrix} mod \; q$$

Properties of g

3. Injective:

- $Suppose \; \exists \; (s_1, e_1, d_1), (s_2, e_2, d_2) \; s.t. \; g_K(s_1, e_1, d_1) = g_K(s_2, e_2, d_2)$
- $\bar{g}_K(s_1, e_1) - \bar{g}_K(s_2, e_2) + (d_1 - d_2) \cdot v = 0 \; mod \; q$
- $If \; d_1 = d_2 \; then \; \bar{g}_K(s_1, e_1) = \bar{g}_K(s_2, e_2) \Rightarrow s_1 = s_2, e_1 = e_2$ ✓
- $If \; d_1 \neq d_2 \Rightarrow \bar{g}_K(s_1, e_1) - \bar{g}_K(s_2, e_2) = v \Leftrightarrow K(s_1 - s_2) + (e_1 - e_2) = \begin{pmatrix} \frac{q}{2} \\ 0 \\ \dots \\ 0 \end{pmatrix} mod \; q \quad (*)$
- $K = \begin{pmatrix} K_1 \\ \bar{K} \end{pmatrix}, \; e_1 - e_2 = e = \begin{pmatrix} e' \\ \bar{e} \end{pmatrix} \quad \overset{(*)}{\Rightarrow} \quad \begin{cases} \langle K_1, s_1 - s_2 \rangle + e' = \frac{q}{2} & (1) \\ \bar{K}\,(s_1 - s_2) + \bar{e} = 0 & (2) \end{cases}$
- $But \; \bar{g}_{\bar{K}} \; is \; also \; injective \; (\bar{g} \; is \; injective \; \forall \; m = \Omega(n))$

$$\overset{(2)}{\Rightarrow} s_1 = s_2$$

$$\overset{(1)}{\Rightarrow} e' = \frac{q}{2}. \; But \; |e'| = |e_{1,1} - e_{2,1}| \leq |e_{1,1}| + |e_{2,1}| < \frac{q}{2}.$$

$Contradiction$

Construction of the function h

- $g_K : \mathbb{Z}_q^n \times \chi^m \times \{0,1\} \rightarrow \mathbb{Z}_q^m$

$$g_K(s,e,d) = \bar{g}_K(s,e) + d \cdot v \bmod q = Ks + e + d \cdot \begin{pmatrix} \frac{q}{2} \\ 0 \\ \dots \\ 0 \end{pmatrix} \bmod q$$

- $h : \mathbb{Z}_q^n \times \chi^m \times \{0,1\} \rightarrow \{0,1\}$

$$h(s,e,d) = d$$

Properties of h

1. *Homomorphic* $h(x_1) \oplus h(x_2) = h(x_2 - x_1)$
   - $h(s_1,e_1,d_1) \oplus h(s_2,e_2,d_2) = d_1 \oplus d_2 = h(s_2 - s_1 \bmod q, e_2 - e_1, d_2 \oplus d_1)$

Construction of the function h

- $g_K : \mathbb{Z}_q^n \times \chi^m \times \{0,1\} \to \mathbb{Z}_q^m$

$$g_K(s,e,d) = \bar{g}_K(s,e) + d \cdot v \bmod q = Ks + e + d \cdot \begin{pmatrix} \frac{q}{2} \\ 0 \\ \dots \\ 0 \end{pmatrix} \bmod q$$

- $h : \mathbb{Z}_q^n \times \chi^m \times \{0,1\} \to \{0,1\}$

$$h(s,e,d) = d$$

Properties of h

1. *Homomorphic* $h(x_1) \oplus h(x_2) = h(x_2 - x_1)$
   - $h(s_1,e_1,d_1) \oplus h(s_2,e_2,d_2) = d_1 \oplus d_2 = h(s_2 - s_1 \bmod q, e_2 - e_1, d_2 \oplus d_1)$

2. *Hardcore predicate* (*wrt* g):

- *Given* $(K, g_K(s,e,d))$ *is hard to guess* d
- *Hard to distinguish*: $D_1 = \{K, Ks + e\}$ *and* $D_2 = \{K, Ks + e + v\}$
- *From decisional LWE*: $D_1 \overset{c}{\approx} \{K, u\}, u \overset{u}{\leftarrow} \mathbb{Z}_q^m$
- v *is a fixed vector*: $D_2 \overset{c}{\approx} \{K, u\} \overset{c}{\approx} D_1$

Summary and Future work

- QFactory: simulates quantum channel from classical channel;
- Solve blind delegated quantum computations using ~~quantum client~~ → classical client;
- Protocol is secure in the malicious setting;
- Several extensions of the protocol can be achieved, including classical verification of quantum computations;

Summary and Future work

- QFactory: simulates quantum channel from classical channel;
- Solve blind delegated quantum computations using ~~quantum client~~ → classical client;
- Protocol is secure in the malicious setting;
- Several extensions of the protocol can be achieved, including classical verification of quantum computations;

Next:

- Improve the efficiency of the QFactory protocol, by looking at other post-quantum solutions;
- Prove the security of the QFactory module in the composable setting;
- Explore new possible applications (e.g. multiparty quantum computation).

1) **“On the possibility of classical client blind quantum computing”** **(Cojocaru, Colisson, Kashefi, Wallden)**

- https://arxiv.org/abs/1802.08759

2) **“QFactory: classically-instructed remote secret qubits preparation”** **(Cojocaru, Colisson, Kashefi, Wallden)**

- https://arxiv.org/abs/1904.06303

Thank you!

MP Trapdoor function

- $q = 2^k$
- $g^t = [2^0\ 2^1\ \dots 2^{k-1}] \in \mathbb{Z}_\mathrm{q}^\mathrm{k}$
- $G = I_n \otimes g^t \in \mathbb{Z}_\mathrm{q}^{\mathrm{n} \times nk}$

MP Trapdoor function

- I) Invert $\boxed{\bar{b} = g_{g^t}(s,e)} = s \cdot g^t + e^t,$

where $e \in \mathbb{Z}^k$, $s = s_{k-1}s_{k-2} \dots s_1 s_0 \in \mathbb{Z}_\mathrm{q}$, $s_i \in \{0,1\}$ and $e_i \in \left[-\frac{q}{4}, \frac{q}{4}\right]$

MP Trapdoor function

- I) Invert $\boxed{\bar{b} = g_{g^t}(s, e)} = s \cdot g^t + e^t$,

where $e \in \mathbb{Z}^k$, $s = s_{k-1}s_{k-2} \ldots s_1 s_0 \in \mathbb{Z}_\mathrm{q}$, $s_i \in \{0,1\}$ and $e_i \in \left[-\frac{q}{4}, \frac{q}{4}\right]$

- $\bar{b} = [2^0 \cdot s + e_0, 2^1 \cdot s + e_1, \ldots, 2^{k-1} \cdot s + e_{k-1}]$
- $\bar{b}_{k-1} = 2^{k-1} \cdot s + e_{k-1} = 2^{k-1} \cdot \left(s_0 + 2s_1 + \ldots + 2^{k-1}s_{k-1}\right) + e_{k-1}$
- $\bar{b}_{k-1} = 2^{k-1} \cdot s_0 + e_{k-1} \; mod \; q = \frac{q}{2} \cdot s_0 + e_{k-1} \; mod \; q$

MP Trapdoor function

- I) Invert $\boxed{\bar{b} = g_{g^t}(s,e)} = s \cdot g^t + e^t$,

where $e \in \mathbb{Z}^k$, $s = s_{k-1}s_{k-2} \dots s_1 s_0 \in \mathbb{Z}_\mathrm{q}$, $s_i \in \{0,1\}$ and $e_i \in \left[-\frac{q}{4}, \frac{q}{4}\right]$

- $\bar{b} = [2^0 \cdot s + e_0, 2^1 \cdot s + e_1, \dots, 2^{k-1} \cdot s + e_{k-1}]$
- $\bar{b}_{k-1} = 2^{k-1} \cdot s + e_{k-1} = 2^{k-1} \cdot \left(s_0 + 2s_1 + \dots + 2^{k-1}s_{k-1}\right) + e_{k-1}$
- $\bar{b}_{k-1} = 2^{k-1} \cdot s_0 + e_{k-1}\ mod\ q = \frac{q}{2} \cdot s_0 + e_{k-1}\ mod\ q$
- If $\bar{b}_{k-1}$ is closer to $\frac{q}{2}$ than to 0, then $s_0 = 1$, otherwise $s_0 = 0$.

MP Trapdoor function

- I) Invert $\boxed{\bar{b} = g_{g^t}(s,e)} = s \cdot g^t + e^t$,

where $e \in \mathbb{Z}^k$, $s = s_{k-1}s_{k-2}\dots s_1 s_0 \in \mathbb{Z}_q$, $s_i \in \{0,1\}$ and $e_i \in \left[-\frac{q}{4}, \frac{q}{4}\right]$

- $\bar{b} = [2^0 \cdot s + e_0, 2^1 \cdot s + e_1, \dots, 2^{k-1} \cdot s + e_{k-1}]$
- $\bar{b}_{k-1} = 2^{k-1} \cdot s + e_{k-1} = 2^{k-1} \cdot \left(s_0 + 2s_1 + \dots + 2^{k-1}s_{k-1}\right) + e_{k-1}$
- $\bar{b}_{k-1} = 2^{k-1} \cdot s_0 + e_{k-1} \bmod q = \frac{q}{2} \cdot s_0 + e_{k-1} \bmod q$
- If $\bar{b}_{k-1}$ is closer to $\frac{q}{2}$ than to 0, then $s_0 = 1$, otherwise $s_0 = 0$.
- $\bar{b}_{k-2} = 2^{k-2} \cdot \left(s_0 + 2s_1 + \cdots + 2^{k-1}s_{k-1}\right) + e_{k-2}$
- $\bar{b}_{k-2} = 2^{k-2}s_0 + 2^{k-1}s_1 + e_{k-2} \bmod q$
- $\bar{b}_{k-2} - 2^{k-2}s_0 = \frac{q}{2}\, s_1 + e_{k-2} \bmod q$
- If $\bar{b}_{k-2} - 2^{k-2}s_0$ is closer to $\frac{q}{2}$ than to 0, then $s_1 = 1$, otherwise $s_1 = 0$.
- And so on …

MP Trapdoor function

- II) Invert $\boxed{\bar{\bar{b}} = g_G(s,e)} = s^t \cdot G + e^t$

where $s = [s_0\ s_1\ \dots s_{n-1}] \in \mathbb{Z}_q^n$ and $e = [e_0 \dots\ e_{nk-1}] \in \mathbb{Z}^{nk}$

- $\bar{\bar{b}} = [s_0 \cdot g^t\ ,\ s_1 \cdot g^t, \dots, s_{n-1} \cdot g^t] + [e_0 \dots\ e_{nk-1}]$
- $\bar{\bar{b}} = \left[g_{g^t}\left(s_0, e^{(1)}\right), g_{g^t}\left(s_1, e^{(2)}\right),\ \dots, g_{g^t}\left(s_{n-1}, e^{(n)}\right)\right]$,
  - where $e^{(1)}$ are the first n elements of e, $e^{(2)}$ - the next n elements of e and so on;
- Then, we run Invert $g_{g^t}(s,e)$ n times for each component of $\bar{\bar{b}}$

MP Trapdoor function

- III) Generate Key & Trapdoor
- Idea: For an arbitrary index K, the trapdoor t_K is such that $K \cdot \begin{bmatrix} R \\ I \end{bmatrix} = G$

MP Trapdoor function

- III) Generate Key & Trapdoor
- Idea: For an arbitrary index K, the trapdoor t_K is such that $K \cdot \begin{bmatrix} R \\ I \end{bmatrix} = G$
- 1) $R \overset{\$}{\leftarrow} \mathbb{Z}^{(m-nk) \times nk}$
- 2) $T = \begin{bmatrix} I_{m-nk} & R \\ 0 & I_{nk} \end{bmatrix} \Longrightarrow T^{-1} = \begin{bmatrix} I_{m-nk} & -R \\ 0 & I_{nk} \end{bmatrix}$
- 3) $\bar{A} \overset{\$}{\leftarrow} \mathbb{Z}_q^{n \times (m-nk)}$
- 4) $A' = [\bar{A} \mid G] \in \mathbb{Z}_q^{n \times (nk+m-nk)}$
- 5) $K = A' \cdot T^{-1} \in \mathbb{Z}_q^{n \times m}$
- 6) $K = [\bar{A} \mid G] \cdot \begin{bmatrix} I_{m-nk} & -R \\ 0 & I_{nk} \end{bmatrix} = [\bar{A} \mid G - \bar{A}R]$
  - K is close to uniform as long as $[\bar{A} \mid \bar{A}R]$ is close to uniform;
- 7) $K \cdot \begin{bmatrix} R \\ I \end{bmatrix} = [\bar{A} \mid G - \bar{A}R] \cdot \begin{bmatrix} R \\ I \end{bmatrix} = \bar{A}R + G - \bar{A}R = G$
- Output $K, t_K = R$

MP Trapdoor function

- IV) Invert $(b \;=\; g_K(s,e), t_K)$
- $b = s^t \cdot K + e^t$
- $b' \leftarrow b \cdot \begin{bmatrix} t_K \\ I \end{bmatrix} = s^t \cdot K \cdot \begin{bmatrix} t_K \\ I \end{bmatrix} + \; e^t \cdot \begin{bmatrix} t_K \\ I \end{bmatrix} = s^t \cdot G + e^t \cdot \begin{bmatrix} t_K \\ I \end{bmatrix} = g_G\left(s, e^t \cdot \begin{bmatrix} t_K \\ I \end{bmatrix}\right)$
- Run $Invert_G(b') \implies s \;\;,\; e = b \; - s^t \cdot K$