Quantum Pseudorandomness Cannot be Shrunk in a Black-Box Way

Samuel Bouaziz--Ermann
Joint work with Garazi Muguruza

LIP6, Sorbonne Université, CNRS
arXiv:2402.13324

May 17, 2024

Classical Pseudorandomness

Classical Pseudorandomness

Randomness in cryptography

Randomness is essential to build cryptographic primitives.
We want to sample from the **uniform distribution** $\mathcal{U}_\ell$.
Problem: how to generate it **efficiently**?

Classical Pseudorandomness

Randomness in cryptography

Randomness is essential to build cryptographic primitives.
We want to sample from the **uniform distribution** $\mathcal{U}_\ell$.
Problem: how to generate it **efficiently**?

Pseudorandom Number Generator

A function $F : \{0,1\}^n \to \{0,1\}^\ell$ is a Pseudorandom Number Generator (PRNG) if:

Classical Pseudorandomness

Randomness in cryptography

Randomness is essential to build cryptographic primitives.
We want to sample from the **uniform distribution** $\mathcal{U}_\ell$.
Problem: how to generate it **efficiently**?

Pseudorandom Number Generator

A function $F : \{0,1\}^n \to \{0,1\}^\ell$ is a Pseudorandom Number Generator (PRNG) if:

1. $F(x)$ can be computed efficiently.

Classical Pseudorandomness

Randomness in cryptography

Randomness is essential to build cryptographic primitives.
We want to sample from the **uniform distribution** $\mathcal{U}_\ell$.
Problem: how to generate it **efficiently**?

Pseudorandom Number Generator

A function $F : \{0,1\}^n \to \{0,1\}^\ell$ is a Pseudorandom Number Generator (PRNG) if:

1. $F(x)$ can be computed efficiently.
2. $F(x) \approx \mathcal{U}_\ell$, when $x \leftarrow \mathcal{U}_n$.

Classical Pseudorandomness

Randomness in cryptography

Randomness is essential to build cryptographic primitives.
We want to sample from the **uniform distribution** $\mathcal{U}_\ell$.
Problem: how to generate it **efficiently**?

Pseudorandom Number Generator

A function $F : \{0,1\}^n \rightarrow \{0,1\}^\ell$ is a Pseudorandom Number Generator (PRNG) if:

1. $F(x)$ can be computed efficiently.
2. $F(x) \approx \mathcal{U}_\ell$, when $x \leftarrow \mathcal{U}_n$.
3. $\ell > n$.

Classical Pseudorandomness

Randomness in cryptography

Randomness is essential to build cryptographic primitives.
We want to sample from the **uniform distribution** $\mathcal{U}_\ell$.
Problem: how to generate it **efficiently**?

Pseudorandom Number Generator

A function $F : \{0,1\}^n \to \{0,1\}^\ell$ is a Pseudorandom Number Generator (PRNG) if:

1. $F(x)$ can be computed efficiently.
2. $F(x) \approx \mathcal{U}_\ell$, when $x \leftarrow \mathcal{U}_n$.
3. $\ell > n$.

One-Way Functions

A function $F : \{0,1\}^n \to \{0,1\}^n$ is a One-Way Function (OWF) if:

1. $F(x)$ can be computed efficiently.
2. Given $y = F(x)$, it is hard to compute x.

Other cryptography primitives

One-Way Functions

A function $F : \{0,1\}^n \rightarrow \{0,1\}^n$ is a One-Way Function (OWF) if:

1. $F(x)$ can be computed efficiently.
2. Given $y = F(x)$, it is hard to compute x.

Public Key Encryption

A Public Key Encryption (PKE) scheme allows two users to communicate over an untrusted authenticated channel.

Other cryptography primitives

One-Way Functions

A function $F : \{0,1\}^n \to \{0,1\}^n$ is a One-Way Function (OWF) if:

1. $F(x)$ can be computed efficiently.
2. Given $y = F(x)$, it is hard to compute x.

Public Key Encryption

A Public Key Encryption (PKE) scheme allows two users to communicate over an untrusted authenticated channel.

Alice		Bob
$(\mathsf{sk}, \mathsf{pk}) = \mathbf{Kgen}()$	$\xrightarrow{\mathsf{pk}}$	$c = \mathbf{Enc}(m, \mathsf{pk})$
$m = \mathbf{Dec}(c, \mathsf{sk})$	$\xleftarrow{c}$	

One-Way Functions

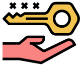
A function $F : \{0,1\}^n \rightarrow \{0,1\}^n$ is a One-Way Function (OWF) if:

1. $F(x)$ can be computed efficiently.
2. Given $y = F(x)$, it is hard to compute x.

Public Key Encryption

A Public Key Encryption (PKE) scheme allows two users to communicate over an untrusted authenticated channel.

Alice		Bob
$(\mathsf{sk}, \mathsf{pk}) = \mathbf{Kgen}()$	$\xrightarrow{\mathsf{pk}}$	$c = \mathbf{Enc}(m, \mathsf{pk})$
$m = \mathbf{Dec}(c, \mathsf{sk})$	$\xleftarrow{c}$	



The relevance of One-Way Functions

- Most advanced cryptographic schemes require one-way functions.
- For example, a hash function has to be a one-way function.
- It is the weakest assumption to do classical cryptography.

The relevance of One-Way Functions

- Most advanced cryptographic schemes require one-way functions.
- For example, a hash function has to be a one-way function.
- It is the weakest assumption to do classical cryptography.

Theorem

One-way functions and Pseudorandom Number Generators are equivalent, in a black-box way.

Black-box proof

Black-box proof

Black-box constructions

They are the most natural class of constructions.

Black-box proof

Black-box constructions

They are the most natural class of constructions.
A black-box construction of A from B means that:

- The construction of A from B does not use the "code" of B.

Black-box proof

Black-box constructions

They are the most natural class of constructions.

A black-box construction of A from B means that:

- The construction of A from B does not use the “code” of B.
- If an adversary breaks A, then an adversary breaks primitive B, without using the “code” of A.

Black-box proof

Black-box constructions

They are the most natural class of constructions.
A black-box construction of A from B means that:

- The construction of A from B does not use the “code” of B.
- If an adversary breaks A, then an adversary breaks primitive B, without using the “code” of A.

Black-box constructions *relativize*, meaning that for any oracle O such that B exists (relative to O), then A exists (relative to O).

Black-box proof

Black-box constructions

They are the most natural class of constructions.
A black-box construction of A from B means that:

- The construction of A from B does not use the "code" of B.
- If an adversary breaks A, then an adversary breaks primitive B, without using the "code" of A.

Black-box constructions *relativize*, meaning that for any oracle O such that B exists (relative to O), then A exists (relative to O).
Example: Relative to random oracles, OWF exists. Thus, so does PRNG.

Black-box proof

Black-box constructions

They are the most natural class of constructions.
A black-box construction of A from B means that:

- The construction of A from B does not use the "code" of B.
- If an adversary breaks A, then an adversary breaks primitive B, without using the "code" of A.

Black-box constructions *relativize*, meaning that for any oracle O such that B exists (relative to O), then A exists (relative to O).
Example: Relative to random oracles, OWF exists. Thus, so does PRNG.

Black-box impossibility results

A black-box impossibility result of A from B consist of exhibiting an oracle O such that, relative to O, B exists but not A.

Some results about classical cryptography

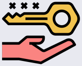
Theorem

$\exists$  $\Rightarrow P \neq NP$

Theorem

$\exists$  $\Rightarrow P \neq NP$

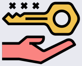
Theorem

$\exists$  $\Rightarrow \exists$ 

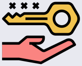
Theorem

$\exists$  $\Rightarrow P \neq NP$

Theorem

$\exists$  $\Rightarrow \exists$ 

Theorem

$\exists$  $\not\Rightarrow \exists$ 

Worlds of cryptography

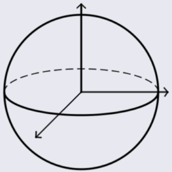
Different worlds where we might live in (Imp'95):

- 😰 **Algorithmica** P=NP
- 😵‍💫 **Heuristica** NP problems are easy on average but hard on the worst case
- 😟 **Pessiland** P$\neq$NP but $\nexists$ one-way function.
- 😌 **Minicrypt** One-Way Functions exist!
- 🤩 **Cryptomania** Public Key Encryption exists!

Quantum Randomness

We can also consider quantum randomness.

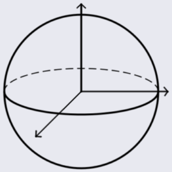
The equivalent to the uniform distribution is the *Haar measure* μ_{2^n}.



Quantum Pseudorandomness

Quantum Randomness

We can also consider quantum randomness.
The equivalent to the uniform distribution is the *Haar measure* μ_{2^n}.



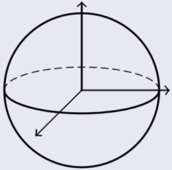
Quantum Pseudorandom States Generators

A function $F : \{0,1\}^\lambda \to \left(\mathbb{C}^2\right)^{\otimes n}$ is a Pseudorandom Quantum State generator (PRS) if:

Quantum Pseudorandomness

Quantum Randomness

We can also consider quantum randomness.
The equivalent to the uniform distribution is the *Haar measure* μ_{2^n}.



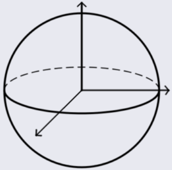
Quantum Pseudorandom States Generators

A function $F : \{0,1\}^\lambda \to \left(\mathbb{C}^2\right)^{\otimes n}$ is a Pseudorandom Quantum State generator (PRS) if:

1. $F(k)$ can be computed efficiently.

Quantum Randomness

We can also consider quantum randomness.
The equivalent to the uniform distribution is the *Haar measure* μ_{2^n}.



Quantum Pseudorandom States Generators

A function $F : \{0,1\}^\lambda \to \left(\mathbb{C}^2\right)^{\otimes n}$ is a Pseudorandom Quantum State generator (PRS) if:

1. $F(k)$ can be computed efficiently.
2. $F(k) \approx \mu_{2^n}$, when $k \leftarrow \mathcal{U}_\lambda$.

Formal definition of PRSs

Definition (Pseudorandom quantum states [JSL18])

A keyed family of n-qubit quantum states $\{|\varphi_k\rangle\}_{k\in\{0,1\}^\lambda}$ is *pseudorandom* if the following two conditions hold:

Formal definition of PRSs

Definition (Pseudorandom quantum states [JSL18])

A keyed family of n-qubit quantum states $\{|\varphi_k\rangle\}_{k\in\{0,1\}^\lambda}$ is *pseudorandom* if the following two conditions hold:

1. **Efficient generation**. There is a QPT algorithm G such that:

$$G_\lambda(k) = |\varphi_k\rangle\langle\varphi_k| \,.$$

Formal definition of PRSs

Definition (Pseudorandom quantum states [JSL18])

A keyed family of n-qubit quantum states $\{|\varphi_k\rangle\}_{k\in\{0,1\}^\lambda}$ is *pseudorandom* if the following two conditions hold:

1. **Efficient generation**. There is a QPT algorithm G such that:

$$G_\lambda(k) = |\varphi_k\rangle\langle\varphi_k| \, .$$

2. **Pseudorandomness**. For any QPT adversary $\mathcal{A}$ and all polynomials $t(\cdot)$, we have:

$$\left| \Pr_{k\leftarrow\{0,1\}^\lambda}\left[\mathcal{A}\left(|\varphi_k\rangle^{\otimes t(\lambda)}\right)=1\right] - \Pr_{|\nu\rangle\leftarrow\mu_{2^n}}\left[\mathcal{A}\left(|\nu\rangle^{\otimes t(\lambda)}\right)=1\right] \right| \leq \mathsf{negl}(\lambda).$$

Formal definition of PRSs

Definition (Pseudorandom quantum states [JSL18])

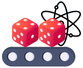
A keyed family of n-qubit quantum states $\{|\varphi_k\rangle\}_{k\in\{0,1\}^\lambda}$ is *pseudorandom* if the following two conditions hold:

1. **Efficient generation**. There is a QPT algorithm G such that:

$$G_\lambda(k) = |\varphi_k\rangle\langle\varphi_k| \,.$$

2. **Pseudorandomness**. For any QPT adversary $\mathcal{A}$ and all polynomials $t(\cdot)$, we have:

$$\left| \Pr_{k\leftarrow\{0,1\}^\lambda}\left[\mathcal{A}\left(|\varphi_k\rangle^{\otimes t(\lambda)}\right)=1\right] - \Pr_{|\nu\rangle\leftarrow\mu_{2n}}\left[\mathcal{A}\left(|\nu\rangle^{\otimes t(\lambda)}\right)=1\right]\right| \leq \mathrm{negl}(\lambda).$$

If $n \approx \lambda$, it is a **long-PRS**, or just PRS. 

Definition (Pseudorandom quantum states [JSL18])

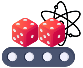
A keyed family of n-qubit quantum states $\{|\varphi_k\rangle\}_{k\in\{0,1\}^\lambda}$ is *pseudorandom* if the following two conditions hold:

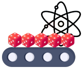
1. **Efficient generation**. There is a QPT algorithm G such that:

$$G_\lambda(k) = |\varphi_k\rangle\langle\varphi_k| \,.$$

2. **Pseudorandomness**. For any QPT adversary $\mathcal{A}$ and all polynomials $t(\cdot)$, we have:

$$\left| \Pr_{k\leftarrow\{0,1\}^\lambda}\left[\mathcal{A}\left(|\varphi_k\rangle^{\otimes t(\lambda)}\right) = 1\right] - \Pr_{|\nu\rangle\leftarrow\mu_{2^n}}\left[\mathcal{A}\left(|\nu\rangle^{\otimes t(\lambda)}\right) = 1\right] \right| \leq \mathsf{negl}(\lambda).$$

If $n \approx \lambda$, it is a **long-PRS**, or just PRS. 

If $n \approx \log\lambda$, it is a **short-PRS**. 

Worlds of quantum cryptography

Worlds relative to which quantum computation is possible.

- MiniQcrypt: Quantum resistant One-Way Functions exist!

Worlds of quantum cryptography

Worlds relative to which quantum computation is possible.

- MiniQcrypt: Quantum resistant One-Way Functions exist!
- MicroCrypt: PRSs exist!

  oblivious transfer, multi party computation, public key encryption with quantum keys, quantum one-time digital signatures, pseudo one-time pad encryption schemes, statistically binding and computationally hiding commitments and quantum computational zero knowledge proofs, bit commitments...

Worlds of quantum cryptography

Worlds relative to which quantum computation is possible.

- MiniQcrypt: Quantum resistant One-Way Functions exist!
- MicroCrypt: PRSs exist!

  oblivious transfer, multi party computation, public key encryption with quantum keys, quantum one-time digital signatures, pseudo one-time pad encryption schemes, statistically binding and computationally hiding commitments and quantum computational zero knowledge proofs, bit commitments...
- Another world: short-PRSs exist!

  bit commitments, pseudodeterministic one-way functions, pseudodeterministic pseudorandom number generators, pseudodeterministic signatures...

Worlds of quantum cryptography

Worlds relative to which quantum computation is possible.

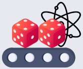
- MiniQcrypt: Quantum resistant One-Way Functions exist!
- MicroCrypt: PRSs exist!

  oblivious transfer, multi party computation, public key encryption with quantum keys, quantum one-time digital signatures, pseudo one-time pad encryption schemes, statistically binding and computationally hiding commitments and quantum computational zero knowledge proofs, bit commitments...
- Another world: short-PRSs exist!

  bit commitments, pseudodeterministic one-way functions, pseudodeterministic pseudorandom number generators, pseudodeterministic signatures...
- Cryptomania: Public Key Encryption exists! (resistant to quantum attacks)

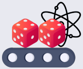
Relation between quantum primitives

Theorem ([JLS18])

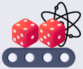
 $\Longrightarrow$ 

Relation between quantum primitives

Theorem ([JLS18])

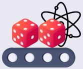
 $\Rightarrow$ 

Theorem ([Kre21])

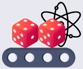
 $\not\Rightarrow$ 

Relation between quantum primitives

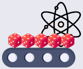
Theorem ([JLS18])

 $\Longrightarrow$ 

Theorem ([Kre21])

 $\not\Longrightarrow$ 

Theorem ([BBO+24,ALY24])

 $\Longrightarrow$ 

Relation between quantum primitives

Theorem ([JLS18])

Theorem ([Kre21])

Theorem ([BBO+24,ALY24])

Idea: Use Tomography, with cost $O(2^d) = \mathrm{poly}(n)$

Different type of PRSs

What about the size of PRS?

Different type of PRSs

What about the size of PRS?

Claim

The output length of a PRNG do not matter, as they are all equivalent to each other.

Claim

The relationship between long-PRS and short-PRS is unclear

Different type of PRSs

What about the size of PRS?

Claim

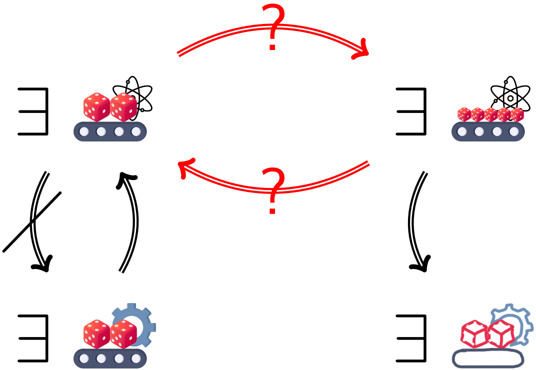
The output length of a PRNG do not matter, as they are all equivalent to each other.

Claim

The relationship between long-PRS and short-PRS is unclear

This work: long-PRSs do not imply short-PRSs.

Relations between primitives



Idea of the proof

Kretschmer's oracle

There exists an oracle $\mathcal{U}$, relative to which:

- PRSs exist.
- PromiseBQP $=$ PromiseQMA. (no OWF)

Idea of the proof

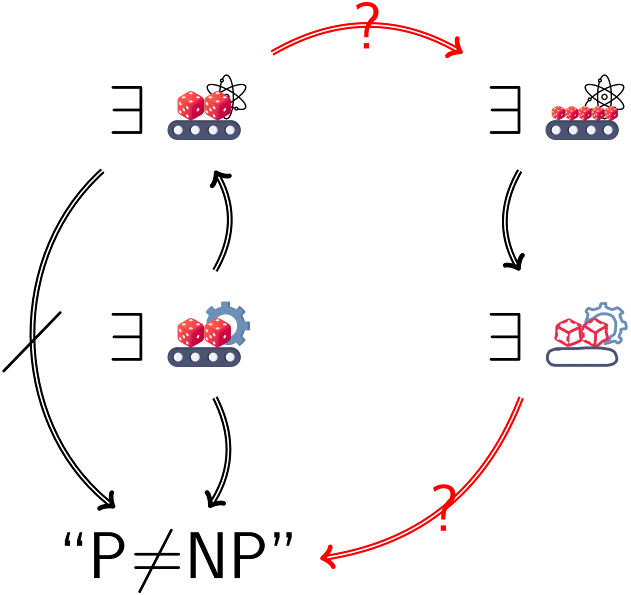
Kretschmer's oracle

There exists an oracle $\mathcal{U}$, relative to which:

- PRSs exist.
- PromiseBQP $=$ PromiseQMA. (no OWF)

It suffices to show that PD-OWF imply
PromiseBQP $\neq$ PromiseQMA (in a black-box way)!

Relations between primitives



Quantum Pseudo-deterministic One-Way Functions

Definition

A QPT algorithm $F : \{0,1\}^{m(\lambda)} \to \{0,1\}^{\ell(\lambda)}$ is a *quantum pseudo-deterministic one-way function* if:

Quantum Pseudo-deterministic One-Way Functions

Definition

A QPT algorithm $F : \{0,1\}^{m(\lambda)} \to \{0,1\}^{\ell(\lambda)}$ is a *quantum pseudo-deterministic one-way function* if:

- **Pseudodeterminism**. There exits a set $\mathcal{K}_\lambda$, for some $c > 0$ and all $\lambda \in \mathbb{N}$:
    1. $\Pr\left[x \in \mathcal{K}_\lambda \,\middle|\, x \leftarrow \{0,1\}^{m(\lambda)}\right] \geq 1 - O(\lambda^{-c})$.

Definition

A QPT algorithm $F : \{0,1\}^{m(\lambda)} \to \{0,1\}^{\ell(\lambda)}$ is a *quantum pseudo-deterministic one-way function* if:

- **Pseudodeterminism**. There exits a set $\mathcal{K}_\lambda$, for some $c > 0$ and all $\lambda \in \mathbb{N}$:
  1. $\Pr\left[x \in \mathcal{K}_\lambda \,\middle|\, x \leftarrow \{0,1\}^{m(\lambda)}\right] \geq 1 - O(\lambda^{-c})$.
  2. For any $x \in \mathcal{K}_\lambda$:

$$\max_{y \in \{0,1\}^{\ell(\lambda)}} \Pr\left[y = F(x)\right] \geq 1 - \mathsf{negl}(\lambda). \tag{1}$$

Definition

A QPT algorithm $F : \{0,1\}^{m(\lambda)} \to \{0,1\}^{\ell(\lambda)}$ is a *quantum pseudo-deterministic one-way function* if:

- **Pseudodeterminism**. There exits a set $\mathcal{K}_\lambda$, for some $c > 0$ and all $\lambda \in \mathbb{N}$:
  1. $\Pr\left[x \in \mathcal{K}_\lambda \,\middle|\, x \leftarrow \{0,1\}^{m(\lambda)}\right] \geq 1 - O(\lambda^{-c})$.
  2. For any $x \in \mathcal{K}_\lambda$:

$$\max_{y \in \{0,1\}^{\ell(\lambda)}} \Pr\left[y = F(x)\right] \geq 1 - \mathrm{negl}(\lambda). \tag{1}$$

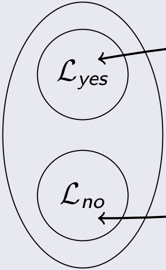
- **Security**. For every QPT inverter $\mathcal{A}$:

$$\Pr_{x \leftarrow \{0,1\}^{m(\lambda)}}\left[F\left(\mathcal{A}(F(x))\right) = F(x)\right] \leq \mathrm{negl}(\lambda). \tag{2}$$

Quantum complexity classes

Definition (PromiseBQP)

$\mathcal{L} = (\mathcal{L}_{yes}, \mathcal{L}_{no}) \in$ PromiseBQP if $\exists$ a QTP $\mathcal{A}$ such that:

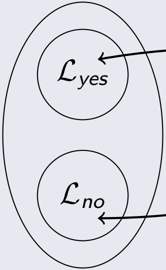


$x \in \mathcal{L}_{yes}$, $\Pr[\mathcal{A}(x) = 1] \geq 2/3$

$x \in \mathcal{L}_{no}$, $\Pr[\mathcal{A}(x) = 0] \geq 2/3$

Definition (PromiseQMA)

$\mathcal{L} = (\mathcal{L}_{yes}, \mathcal{L}_{no}) \in$ PromiseQMA if $\exists$ a QTP $\mathcal{A}$ such that:



$x \in \mathcal{L}_{yes}$, $\exists \left|\phi\right\rangle, \Pr\left[\mathcal{A}(x, \left|\phi\right\rangle) = 1\right] \geq 2/3$

$x \in \mathcal{L}_{no}$, $\forall \left|\phi\right\rangle, \Pr\left[\mathcal{A}(x, \left|\phi\right\rangle) = 0\right] \geq 2/3$

Sketch proof

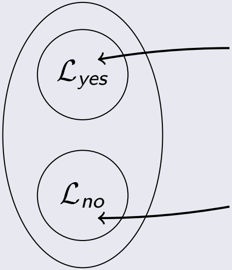
Assume we have a PD-OWF F. The goal is to exhibit a language $\mathcal{L}$ such that $\mathcal{L} \in$ PromiseQMA, and $\mathcal{L} \notin$ PromiseBQP.

Sketch proof

Assume we have a PD-OWF F. The goal is to exhibit a language $\mathcal{L}$ such that $\mathcal{L} \in$ PromiseQMA, and $\mathcal{L} \notin$ PromiseBQP.

Definition (The language)

We define $\mathcal{L} = (\mathcal{L}_{yes}, \mathcal{L}_{no})$ as:



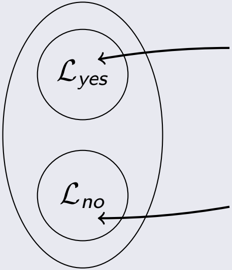
$$\mathcal{L}_{yes} = \left\{ \begin{array}{c} y \text{ that have a pre-image} \\ \text{relative to } F \end{array} \right\}$$

$$\mathcal{L}_{no} = \left\{ \begin{array}{c} y \text{ that do not have a pre-image} \\ \text{relative to } F \end{array} \right\}$$

Sketch proof

Definition (The language)

We define $\mathcal{L} = (\mathcal{L}_{yes}, \mathcal{L}_{no})$ as:



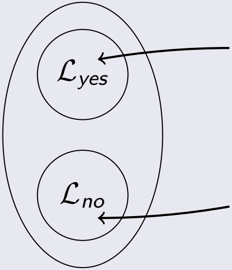
$$\mathcal{L}_{yes} = \left\{ \begin{array}{c} y \text{ such that } \exists x \\ \Pr\left[F(x) = y\right] \geq 1 - \mathrm{negl}(n) \end{array} \right\}$$

$$\mathcal{L}_{no} = \left\{ \begin{array}{c} y \text{ such that } \forall x \\ \Pr\left[F(x) = y\right] \leq 1 - \frac{1}{\mathrm{poly}(n)} \end{array} \right\}$$

Sketch proof

Definition (The language)

We define $\mathcal{L} = (\mathcal{L}_{yes}, \mathcal{L}_{no})$ as:



$$\mathcal{L}_{yes} = \left\{ \begin{array}{c} y \text{ such that } \exists x \\ \Pr\left[F(x) = y\right] \geq 1 - \mathrm{negl}(n) \end{array} \right\}$$

$$\mathcal{L}_{no} = \left\{ \begin{array}{c} y \text{ such that } \forall x \\ \Pr\left[F(x) = y\right] \leq 1 - \frac{1}{\mathrm{poly}(n)} \end{array} \right\}$$

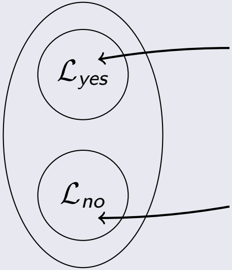
$\mathcal{L} \in$ PromiseQMA

$Ver(x, y)$ runs $F(x)$ many times and checks that $F(x) = y$ every time.

Sketch proof

Definition (The language)

We define $\mathcal{L} = (\mathcal{L}_{yes}, \mathcal{L}_{no})$ as:



$$\mathcal{L}_{yes} = \left\{ \begin{array}{c} y \text{ such that } \exists x \\ \Pr\left[F(x) = y\right] \geq 1 - \mathrm{negl}(n) \end{array} \right\}$$

$$\mathcal{L}_{no} = \left\{ \begin{array}{c} y \text{ such that } \forall x \\ \Pr\left[F(x) = y\right] \leq 1 - \frac{1}{\mathrm{poly}(n)} \end{array} \right\}$$

$\mathcal{L} \in$ PromiseQMA

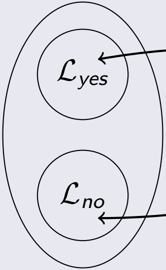
$Ver(x, y)$ runs $F(x)$ many times and checks that $F(x) = y$ every time.

$\mathcal{L} \notin$ PromiseBQP

Otherwise, it would break the security definition of PD-OWF.

Definition (The language)

We define $\mathcal{L} = (\mathcal{L}_{yes}, \mathcal{L}_{no})$ as:



$$\mathcal{L}_{yes} = \left\{ \begin{array}{c} (y, x') \text{ such that } \exists x, x' \prec x \\ \Pr\left[F(x) = y\right] \geq 1 - \mathrm{negl}(n) \end{array} \right\}$$

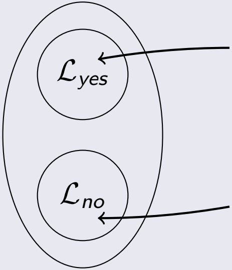
$$x' \prec x \Leftrightarrow x = \underbrace{0101}_{x'}010001$$

$$\mathcal{L}_{no} = \left\{ \begin{array}{c} (y, x') \text{ such that } \forall x, x' \not\prec x \text{ or} \\ \Pr\left[F(x) = y\right] \leq 1 - \frac{1}{\mathrm{poly}(n)} \end{array} \right\}$$

Sketch proof

Definition (The language)

We define $\mathcal{L} = (\mathcal{L}_{yes}, \mathcal{L}_{no})$ as:



$$\mathcal{L}_{yes} = \left\{ \begin{array}{c} (y, x') \text{ such that } \exists x, x' \prec x \\ \Pr\left[F(x) = y\right] \geq 1 - \mathrm{negl}(n) \end{array} \right\}$$

$$x' \prec x \Leftrightarrow x = \underbrace{0101}_{x'}010001$$

$$\mathcal{L}_{no} = \left\{ \begin{array}{c} (y, x') \text{ such that } \forall x, x' \not\prec x \text{ or} \\ \Pr\left[F(x) = y\right] \leq 1 - \frac{1}{\mathrm{poly}(n)} \end{array} \right\}$$

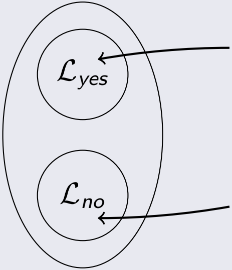
$\mathcal{L} \in$ PromiseQMA

$Ver(x, (x', y))$ checks that $x' \prec x$ and runs $F(x)$ many times and checks that $F(x) = y$ every time.

Sketch proof

Definition (The language)

We define $\mathcal{L} = (\mathcal{L}_{yes}, \mathcal{L}_{no})$ as:



$$\mathcal{L}_{yes} = \left\{ \begin{array}{c} (y, x') \text{ such that } \exists x, x' \prec x \\ \Pr[F(x) = y] \geq 1 - \mathrm{negl}(n) \end{array} \right\}$$

$$x' \prec x \Leftrightarrow x = \underbrace{0101}_{x'}010001$$

$$\mathcal{L}_{no} = \left\{ \begin{array}{c} (y, x') \text{ such that } \forall x, x' \not\prec x \text{ or} \\ \Pr[F(x) = y] \leq 1 - \frac{1}{\mathrm{poly}(n)} \end{array} \right\}$$

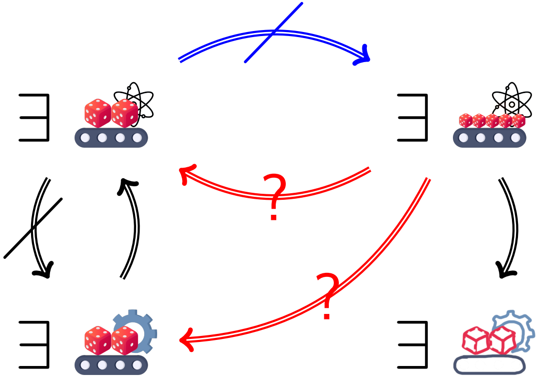
$\mathcal{L} \in$ PromiseQMA

$Ver(x, (x', y))$ checks that $x' \prec x$ and runs $F(x)$ many times and checks that $F(x) = y$ every time.

$\mathcal{L} \notin$ PromiseBQP

Otherwise, it would break the security definition of PD-OWF: for some y, we can learn a pre-image bit by bit.

Conclusion



Pictures of the presentation are adapted from icons from flaticon.com

Conclusion

- We showed that PRSs do not imply short-PRSs.

Conclusion

- We showed that PRSs do not imply short-PRSs.
- Whether short-PRSs imply PRSs or not is still open. However, poly-time short-PRSs imply one-time long-PRSs.

Conclusion

- We showed that PRSs do not imply short-PRSs.
- Whether short-PRSs imply PRSs or not is still open. However, poly-time short-PRSs imply one-time long-PRSs.
- There are many open questions regarding the relationship between quantum cryptographic primitives (EFI, OWSG, PRU...)

Conclusion

- We showed that PRSs do not imply short-PRSs.
- Whether short-PRSs imply PRSs or not is still open. However, poly-time short-PRSs imply one-time long-PRSs.
- There are many open questions regarding the relationship between quantum cryptographic primitives (EFI, OWSG, PRU...)
- Ongoing work: separating Quantum Computation Classical Communication (QCCC) primitives from PRSs, such as KE and PKE with classical keys.

Conclusion

- We showed that PRSs do not imply short-PRSs.
- Whether short-PRSs imply PRSs or not is still open. However, poly-time short-PRSs imply one-time long-PRSs.
- There are many open questions regarding the relationship between quantum cryptographic primitives (EFI, OWSG, PRU...)
- Ongoing work: separating Quantum Computation Classical Communication (QCCC) primitives from PRSs, such as KE and PKE with classical keys.

Thank you for your attention!

Bibliography