Freie Universität Berlin

Classical vs. quantum learning of discrete distributions

Dominik Hangleiter

QSI Seminar, September 10, 2020

Thanks to . . .

Ryan Sweke

Jean-Pierre Seifert

Jens Eisert

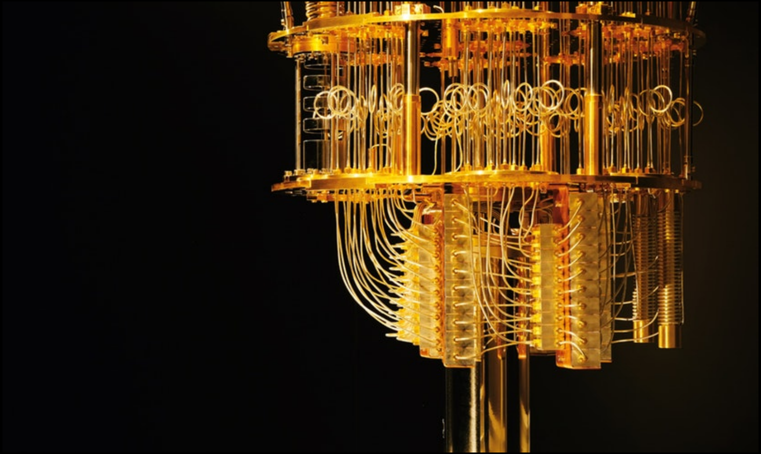
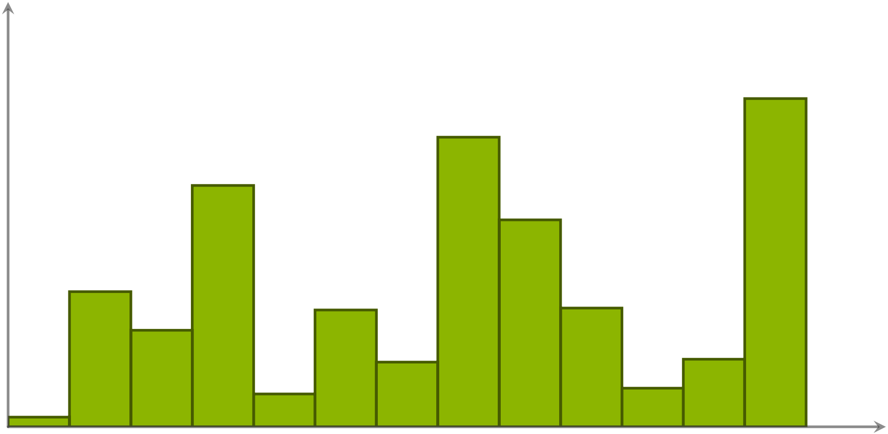


Image Ref.: Graham Carlow/IBM

vs.



Machine learning

Image Ref.: Wikipedia – Katze (CC0)

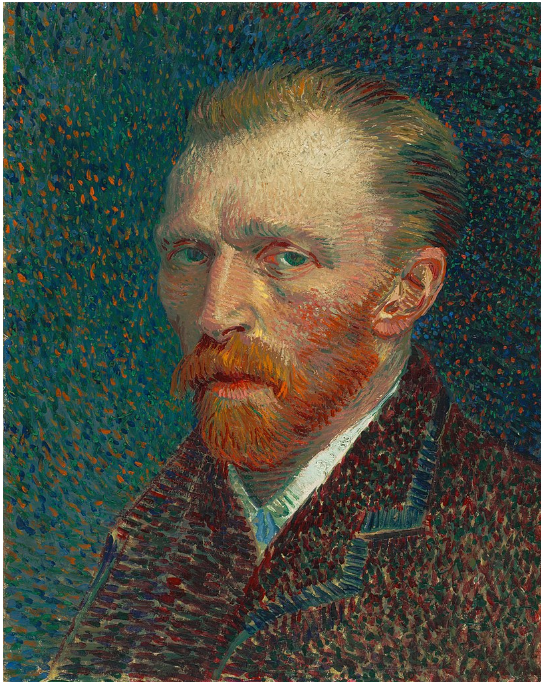
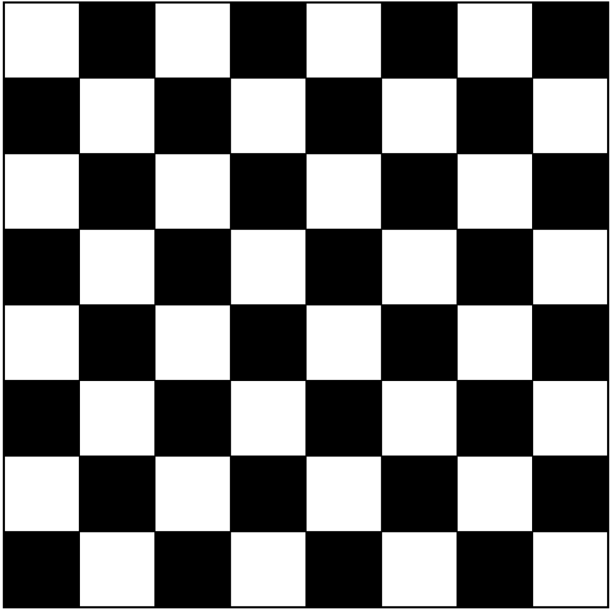


Image Ref.: Wikipedia – Vincent Van Gogh (CC0)



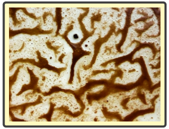
Machine learning and distribution learning

Supervised learning

Unsupervised learning

Reinforcement learning

oracle



learner

Machine learning and distribution learning

Supervised learning

Unsupervised learning

Reinforcement learning

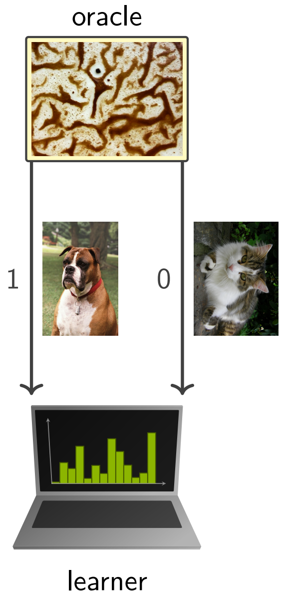


Image Refs.: Wikipedia – Hund, Katze, Van Gogh, Kaffeesatz (CC0), QC: Graham Carlow/IBM

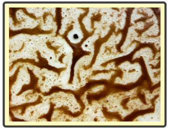
Machine learning and distribution learning

Supervised learning

Unsupervised learning

Reinforcement learning

oracle



Distribution on $\{\text{images}\} \times \{0, 1\}$.

?

learner

Machine learning and distribution learning

Supervised learning

Unsupervised learning

Reinforcement learning

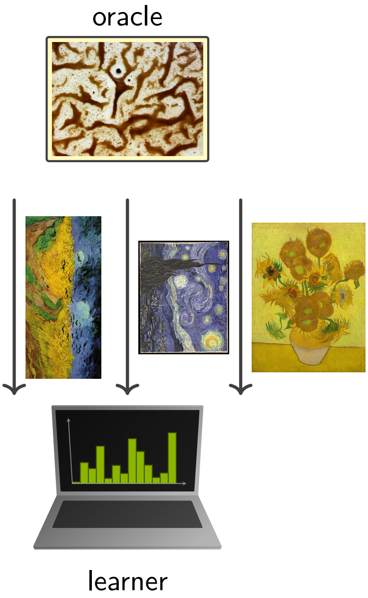
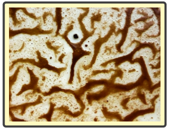


Image Refs.: Wikipedia – Hund, Katze, Van Gogh, Kaffeesatz (CC0), QC: Graham Carlow/IBM

Machine learning and distribution learning

oracle



Supervised learning

Unsupervised learning

Reinforcement learning

Distribution on {images}

learner

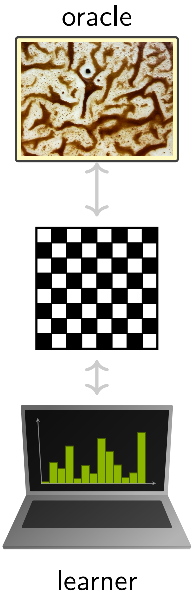
Image Refs.: Wikipedia – Hund, Katze, Van Gogh, Kaffeesatz (CC0), QC: Graham Carlow/IBM

Machine learning and distribution learning

Supervised learning

Unsupervised learning

Reinforcement learning



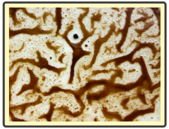
Machine learning and distribution learning

Supervised learning

Unsupervised learning

Reinforcement learning

oracle



Distribution on moves, conditioned on environ. configs.

learner

Machine learning and distribution learning

Supervised learning

Unsupervised learning

Reinforcement learning

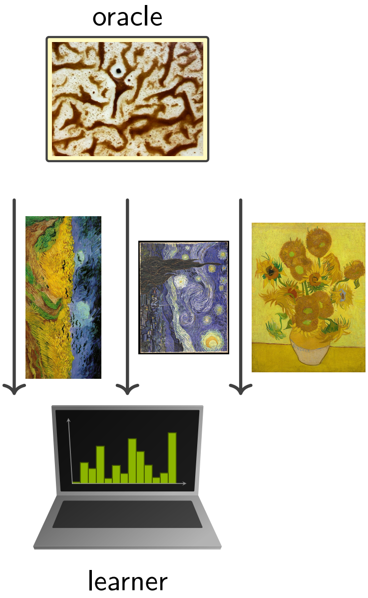


Image Refs.: Wikipedia – Hund, Katze, Van Gogh, Kaffeesatz (CC0), QC: Graham Carlow/IBM

Machine learning and distribution learning

Supervised learning

Unsupervised learning

Reinforcement learning

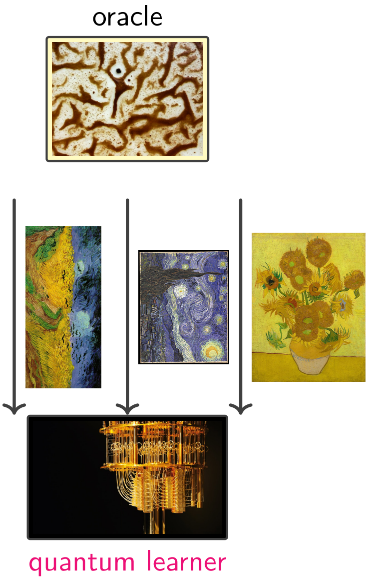
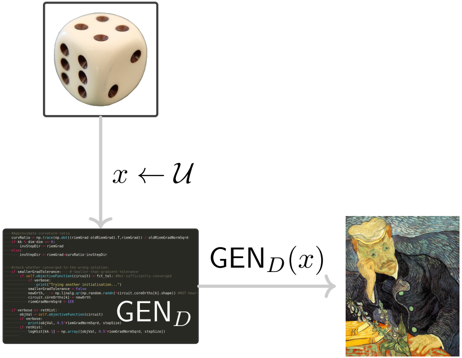


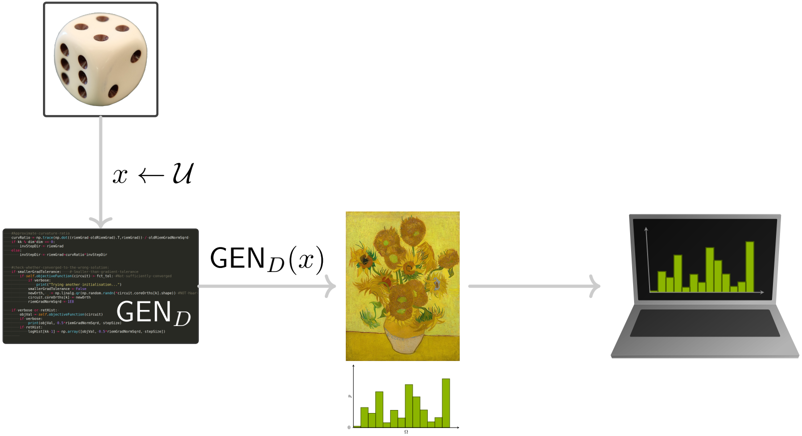
Image Refs.: Wikipedia – Hund, Katze, Van Gogh, Kaffeesatz (CC0), QC: Graham Carlow/IBM

Unsupervised learning: generator vs. density learning



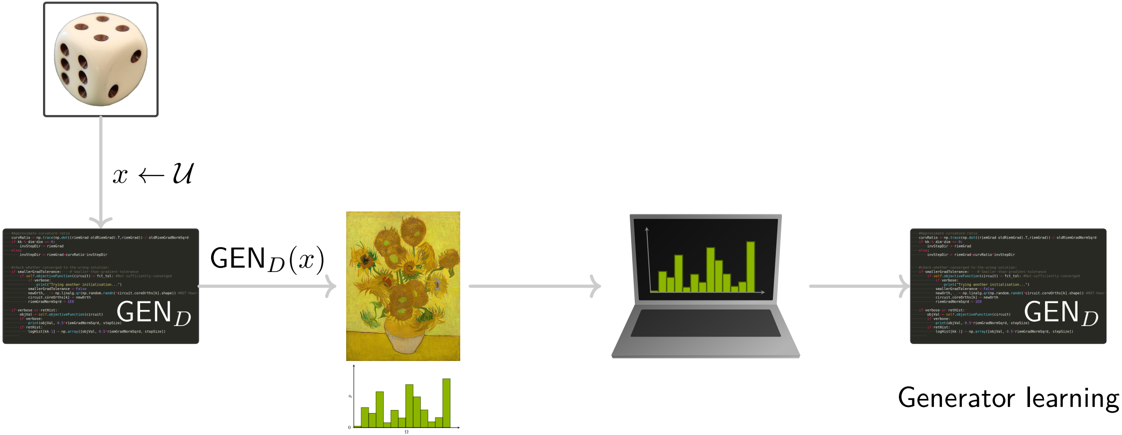
Unsupervised learning: generator vs. density learning

Unsupervised learning: generator vs. density learning



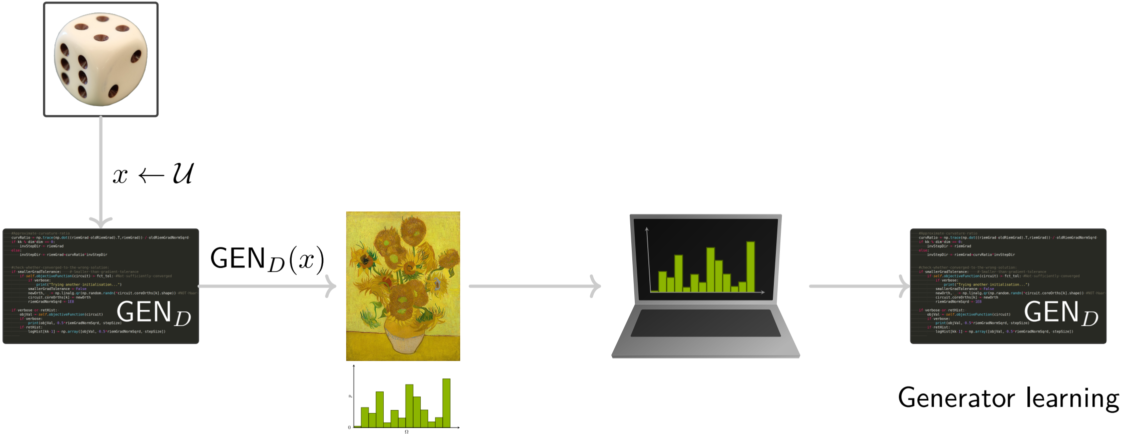
Unsupervised learning: generator vs. density learning

Unsupervised learning: generator vs. density learning



Generator learning

Unsupervised learning: generator vs. density learning



Task: Learn a generator GEN_D of a distribution D.

Classical vs. quantum generative modelling

Question: Quantum generator-learning advantage?

Are there distributions which are

- not efficiently classsically generator-learnable, but
- efficiently quantum generator-learnable?

The details matter – the case of function learning

Learning Boolean functions $f : \{0,1\}^n \to \{0,1\}$.

Arunachalam and de Wolf: A survey of quantum learning theory. SIGACT news (2017).
Arunachalam and de Wolf: Optimal quantum sample complexity of learning algorithms. CCC'17.
Bshouty and Jackson: Learning DNF over the uniform distribution using a quantum example oracle. SIAM J. Comp (1999).

The details matter – the case of function learning

Learning Boolean functions $f : \{0,1\}^n \to \{0,1\}$.

exact learning

membership queries

query complexity

no speedup

PAC learning

random example queries

sample complexity

at most poly speedup

agnostic learning

computational complexity

quantum example queries

super-poly speedup

Arunachalam and de Wolf: A survey of quantum learning theory. SIGACT news (2017).
Arunachalam and de Wolf: Optimal quantum sample complexity of learning algorithms. CCC'17.
Bshouty and Jackson: Learning DNF over the uniform distribution using a quantum example oracle. SIAM J. Comp (1999).

The details matter – the case of function learning

Learning Boolean functions $f : \{0,1\}^n \to \{0,1\}$.

exact learning

PAC learning

agnostic learning

learning framework

membership queries

random example queries

quantum example queries

query complexity

sample complexity

computational complexity

no speedup

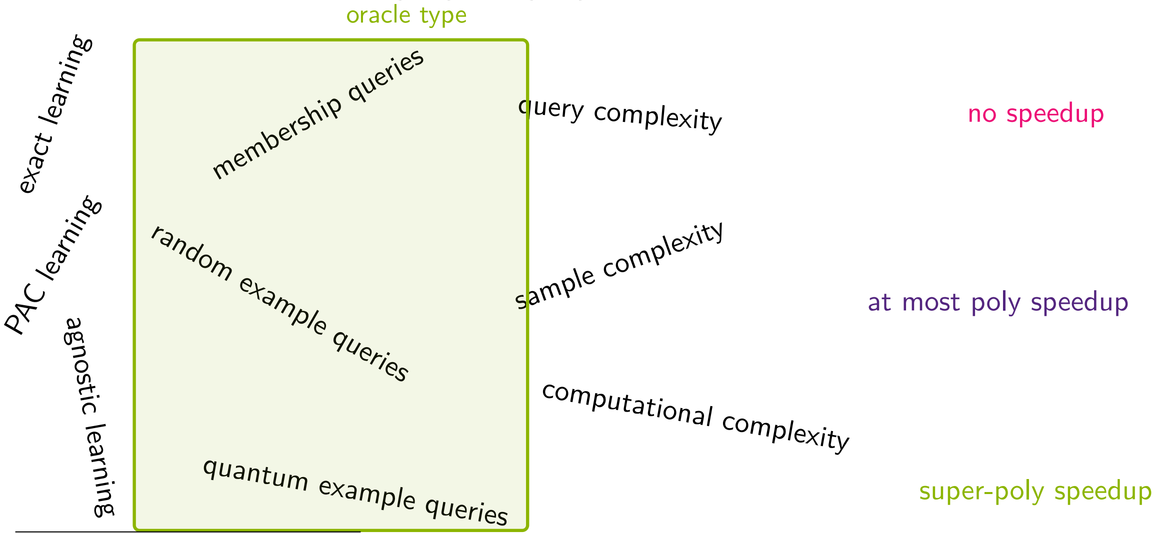
at most poly speedup

super-poly speedup

Arunachalam and de Wolf: A survey of quantum learning theory. SIGACT news (2017).
Arunachalam and de Wolf: Optimal quantum sample complexity of learning algorithms. CCC'17.
Bshouty and Jackson: Learning DNF over the uniform distribution using a quantum example oracle. SIAM J. Comp (1999).

The details matter – the case of function learning

Learning Boolean functions $f : \{0,1\}^n \to \{0,1\}$.



Arunachalam and de Wolf: A survey of quantum learning theory. SIGACT news (2017).
Arunachalam and de Wolf: Optimal quantum sample complexity of learning algorithms. CCC'17.
Bshouty and Jackson: Learning DNF over the uniform distribution using a quantum example oracle. SIAM J. Comp (1999).

The details matter – the case of function learning

Learning Boolean functions $f : \{0,1\}^n \to \{0,1\}$.

exact learning

membership queries

PAC learning

random example queries

agnostic learning

quantum example queries

complexity

query complexity

sample complexity

computational complexity

no speedup

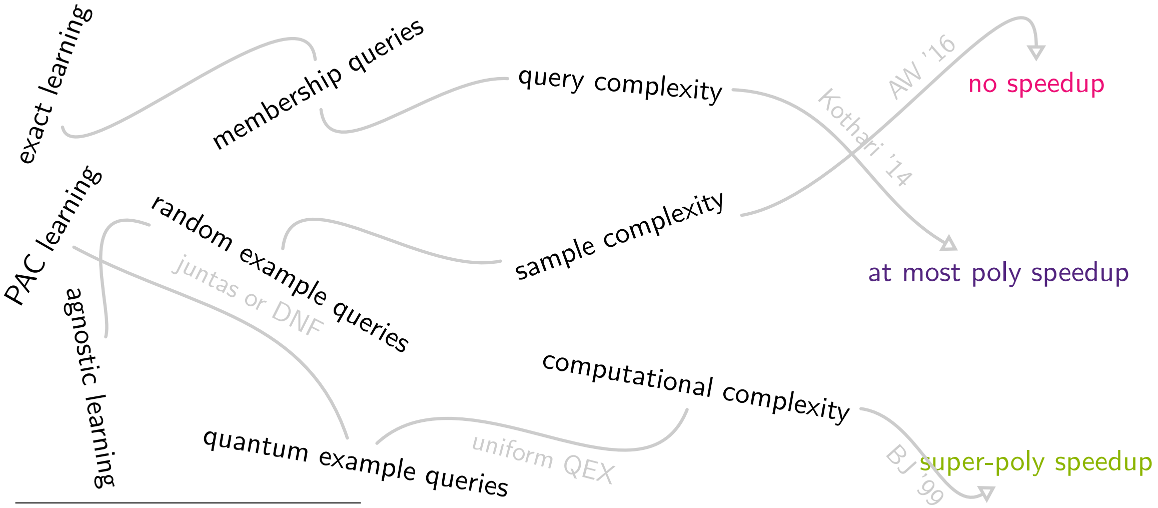
at most poly speedup

super-poly speedup

Arunachalam and de Wolf: A survey of quantum learning theory. SIGACT news (2017).
Arunachalam and de Wolf: Optimal quantum sample complexity of learning algorithms. CCC'17.
Bshouty and Jackson: Learning DNF over the uniform distribution using a quantum example oracle. SIAM J. Comp (1999).

The details matter – the case of function learning

Learning Boolean functions $f : \{0,1\}^n \to \{0,1\}$.



Arunachalam and de Wolf: A survey of quantum learning theory. SIGACT news (2017).
Arunachalam and de Wolf: Optimal quantum sample complexity of learning algorithms. CCC'17.
Bshouty and Jackson: Learning DNF over the uniform distribution using a quantum example oracle. SIAM J. Comp (1999).

Specify the question!

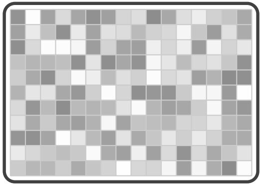
Classes of distributions

Sample space Ω_n. Think $\{0,1\}^n$.

Classes of distributions

Sample space Ω_n. Think $\{0,1\}^n$.

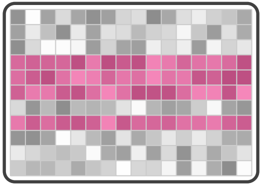
Distributions $\mathcal{D}_n$ over Ω_n.



Classes of distributions

Sample space Ω_n. Think $\{0,1\}^n$.

Distributions $\mathcal{D}_n$ over Ω_n.



Distribution (concept) class $\mathcal{C}_n \subset \mathcal{D}_n$

Classes of distributions

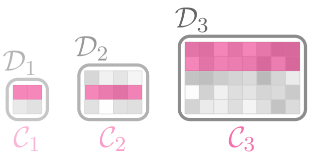
Sample space Ω_n. Think $\{0,1\}^n$.

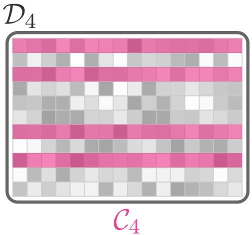
Distributions $\mathcal{D}_n$ over Ω_n.

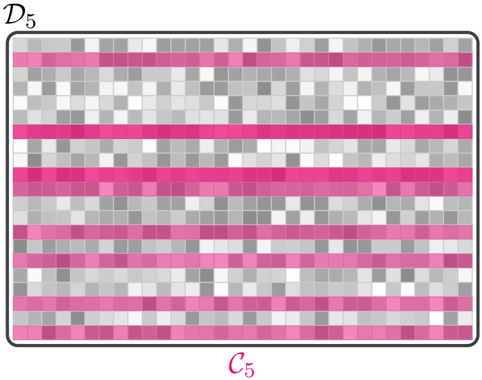
Distribution (concept) class $\mathcal{C}_n \subset \mathcal{D}_n$

Efficiently generate problem instances!

$\mathcal{IG}(1^n) \to D \in \mathcal{C}_n$







Classical vs. quantum learning (1)

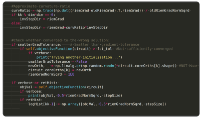
Question: Quantum generator-learning advantage?

Is there a distribution (concept) class which is

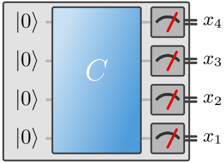
- not efficiently classsically generator-learnable, but
- efficiently quantum generator-learnable?

Generator-learning: classical vs. quantum

Generators



GEN_D

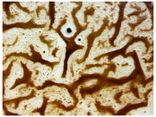


QGEN_D

Oracle access

SAMPLE(D)

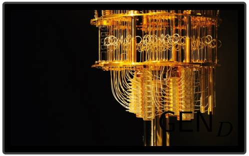
x $\leftarrow$ D



QSAMPLE(D)

$\sum_x \sqrt{D(x)}|x\rangle$

Learning algorithm

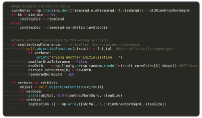


Excellent discussion: Bshouty and Jackson, SIAM J. Comput. (1998)

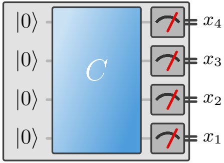
Figure Refs. Dice, coffee grounds: Wikipedia (CC0), QC: Graham Carlow/IBM

Generator-learning: classical vs. quantum

Generators



GEN_D

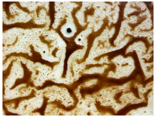


QGEN_D

Oracle access

SAMPLE(D)

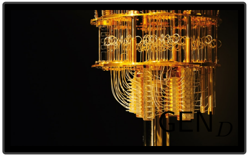
$x \leftarrow D$



QSAMPLE(D)

$\sum_x \sqrt{D(x)}|x\rangle$

Learning algorithm

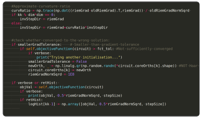


Excellent discussion: Bshouty and Jackson, SIAM J. Comput. (1998)

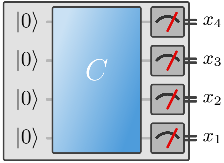
Figure Refs. Dice, coffee grounds: Wikipedia (CC0), QC: Graham Carlow/IBM

Generator-learning: classical vs. quantum

Generators

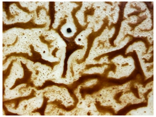
GEN_D



QGEN_D

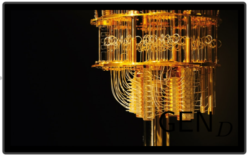
Oracle access

SAMPLE(D)
x ← D



QSAMPLE(D)
$\sum_x \sqrt{D(x)}|x\rangle$

Learning algorithm

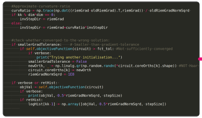



Excellent discussion: Bshouty and Jackson, SIAM J. Comput. (1998)

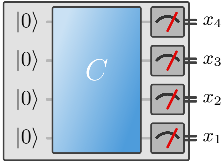
Figure Refs. Dice, coffee grounds: Wikipedia (CC0), QC: Graham Carlow/IBM

Generator-learning: classical vs. quantum

Generators



GEN_D

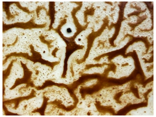


QGEN_D

Oracle access

SAMPLE(D)

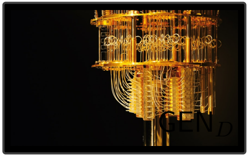
$x \leftarrow D$



QSAMPLE(D)

$\sum_x \sqrt{D(x)}|x\rangle$

Learning algorithm

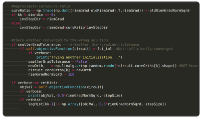


Excellent discussion: Bshouty and Jackson, SIAM J. Comput. (1998)

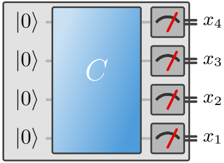
Figure Refs. Dice, coffee grounds: Wikipedia (CC0), QC: Graham Carlow/IBM

Generator-learning: classical vs. quantum

Generators



GEN_D

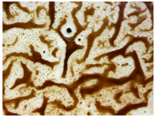


QGEN_D

Oracle access

SAMPLE(D)

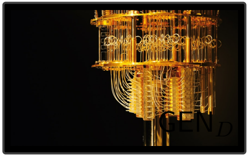
$x \leftarrow D$



QSAMPLE(D)

$\sum_x \sqrt{D(x)}|x\rangle$

Learning algorithm

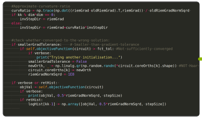


Excellent discussion: Bshouty and Jackson, SIAM J. Comput. (1998)

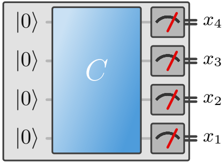
Figure Refs. Dice, coffee grounds: Wikipedia (CC0), QC: Graham Carlow/IBM

Generator-learning: classical vs. quantum

Generators

GEN_D

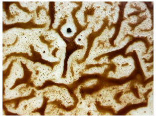


QGEN_D

Oracle access

SAMPLE(D)

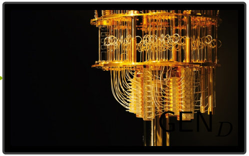
x ← D



QSAMPLE(D)

$\sum_x \sqrt{D(x)}|x\rangle$

Learning algorithm



Excellent discussion: Bshouty and Jackson, SIAM J. Comput. (1998)

Figure Refs. Dice, coffee grounds: Wikipedia (CC0), QC: Graham Carlow/IBM

Classical vs. quantum learning (2)

Question: Quantum generator-learning advantage?

Is there a class of efficiently classically generated discrete distributions which is

- not efficiently classsical generator-learnable, but
- efficiently quantum generator-learnable

w.r.t. the SAMPLE oracle?

PAC generator-learning distribution classes

PAC learning of distribution classes

A distribution class $\mathcal{C}$ is PAC learnable w.r.t. distance d, if there is an algorithm $\mathcal{A}$ which for every $D \in \mathcal{C}$ and every $\epsilon, \delta > 0$, given access to an oracle $O(D)$, outputs

- with probability at least $1 - \delta$ (Probably)

a generator $\mathrm{GEN}_{D'}$ of a distribution D' such that

- $d(D, D') < \epsilon$. (Approximately Correct)

PAC generator-learning distribution classes

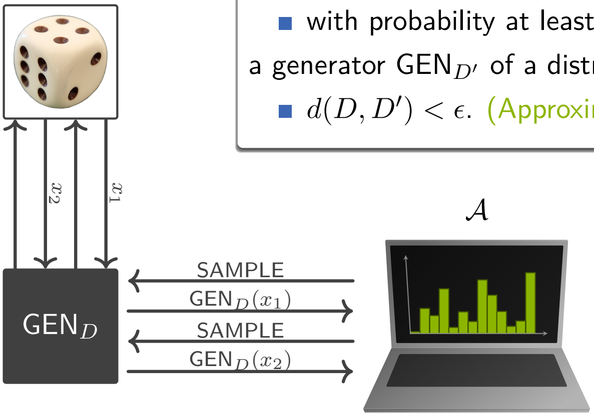
PAC learning of distribution classes

A distribution class $\mathcal{C}$ is PAC learnable w.r.t. distance d, if there is an algorithm $\mathcal{A}$ which for every $D \in \mathcal{C}$ and every $\epsilon, \delta > 0$, given access to an oracle $O(D)$, outputs

- with probability at least $1 - \delta$ (Probably)

a generator $\mathrm{GEN}_{D'}$ of a distribution D' such that

- $d(D, D') < \epsilon$. (Approximately Correct)



PAC generator-learning distribution classes

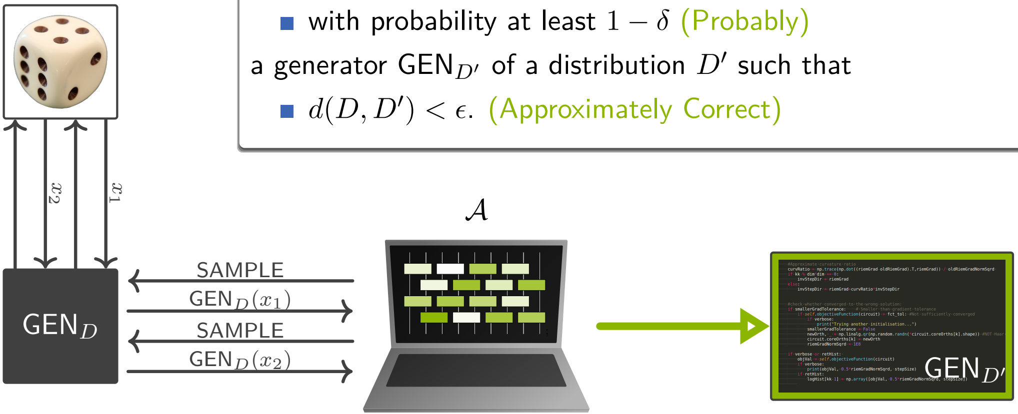
PAC learning of distribution classes

A distribution class $\mathcal{C}$ is PAC learnable w.r.t. distance d, if there is an algorithm $\mathcal{A}$ which for every $D \in \mathcal{C}$ and every $\epsilon, \delta > 0$, given access to an oracle $O(D)$, outputs

- with probability at least $1 - \delta$ (Probably)

a generator $\mathrm{GEN}_{D'}$ of a distribution D' such that

- $d(D, D') < \epsilon$. (Approximately Correct)



PAC generator-learning distribution classes

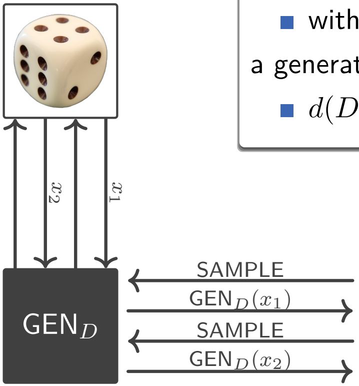
PAC learning of distribution classes

A distribution class $\mathcal{C}$ is efficiently PAC learnable w.r.t. distance d, if there is an algorithm $\mathcal{A}$ which for every $D \in \mathcal{C}$ and every $\epsilon, \delta > 0$, given access to an oracle $O(D)$, outputs in time $\mathrm{poly}(|D|, 1/\epsilon, 1/\delta)$

- with probability at least $1 - \delta$ (Probably)

a generator $\mathrm{GEN}_{D'}$ of a distribution D' such that

- $d(D, D') < \epsilon$. (Approximately Correct)



PAC generator-learning distribution classes

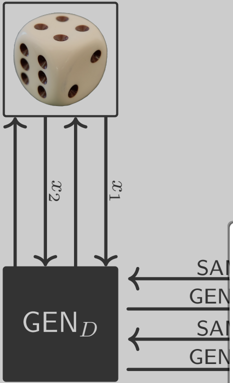
PAC learning of distribution classes

A distribution class $\mathcal{C}$ is efficiently PAC learnable w.r.t. distance d, if there is an algorithm $\mathcal{A}$ which for every $D \in \mathcal{C}$ and every $\epsilon, \delta > 0$, given access to an oracle $O(D)$, outputs in time $\mathrm{poly}(|D|, 1/\epsilon, 1/\delta)$

- with probability at least $1-\delta$ (Probably)

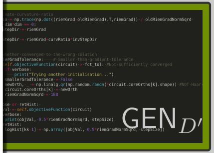
a generator $\mathrm{GEN}_{D'}$ of a distribution D' such that

- $d(D, D') < \epsilon$. (Approximately Correct)



Distance measures: KL divergence

$$d_{\mathsf{KL}}(D, D') = \sum_x D(x) \log\left(\frac{D(x)}{D'(x)}\right)$$



Finally ...

A quantum vs. classical separation for distribution learning

Question: Quantum generator-learning advantage?

Is there a class of efficiently classically generated discrete distributions which is

- not efficiently classsical PAC generator-learnable, but
- efficiently quantum PAC generator-learnable

w.r.t. the SAMPLE oracle and the KL divergence?

Coyle et al., NPJ Quant. Inf. (2020).

A quantum vs. classical separation for distribution learning

Question: Quantum generator-learning advantage?

Is there a class of efficiently classically generated discrete distributions which is

- not efficiently classsical PAC generator-learnable, but
- efficiently quantum PAC generator-learnable

w.r.t. the SAMPLE oracle and the KL divergence?

Theorem: **Y E S*** **!**

*under the decisional Diffie-Hellman assumption for the group family of quadratic residues

Coyle et al., NPJ Quant. Inf. (2020).

Proof sketch

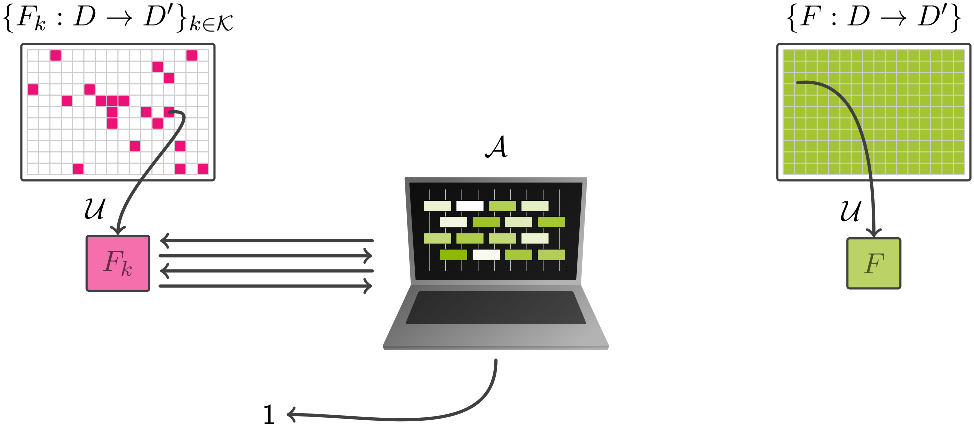
Proof sketch

1. Classical hardness
2. Quantum easiness

Distributions that are hard to learn classically (1)

Pseudorandom function (PRF)

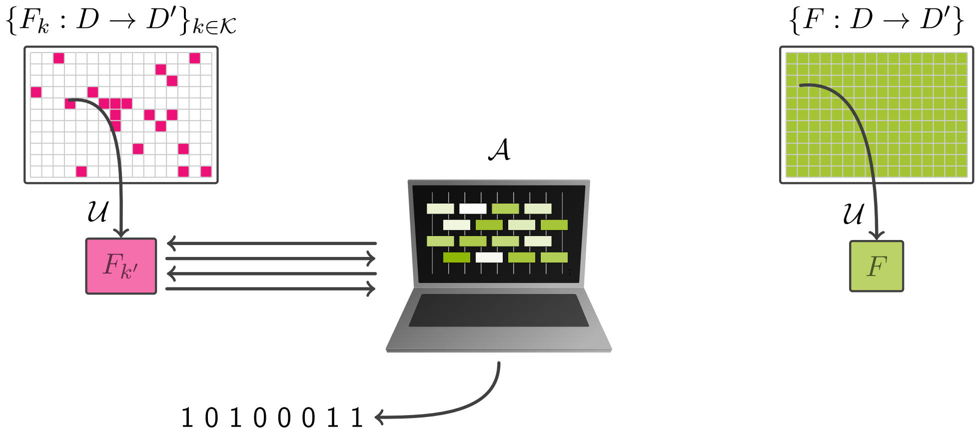
A collection of keyed functions $\{F_k : D \to D'\}_{k\in\mathcal{K}}$ that cannot be distinguished from uniformly random functions in $\{F : D \to D'\}$ by any polynomial-time algorithm $\mathcal{A}$.



Distributions that are hard to learn classically (1)

Pseudorandom function (PRF)

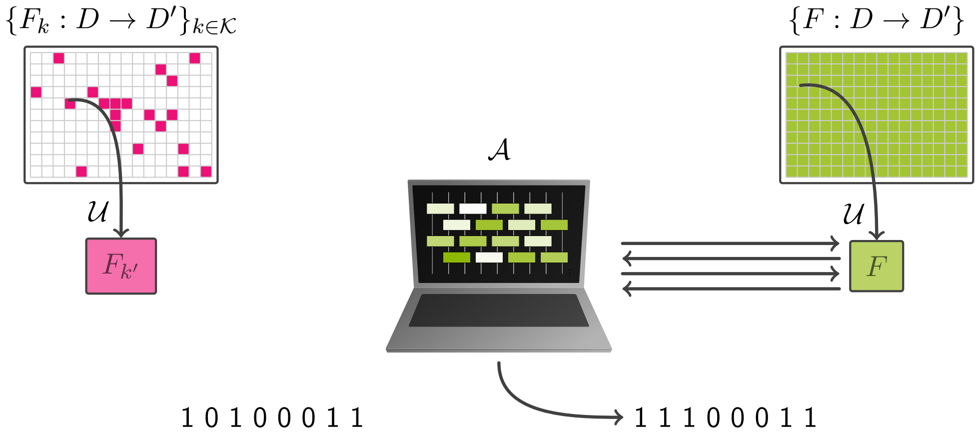
A collection of keyed functions $\{F_k : D \to D'\}_{k \in \mathcal{K}}$ that cannot be distinguished from uniformly random functions in $\{F : D \to D'\}$ by any polynomial-time algorithm $\mathcal{A}$.



Distributions that are hard to learn classically (1)

Pseudorandom function (PRF)

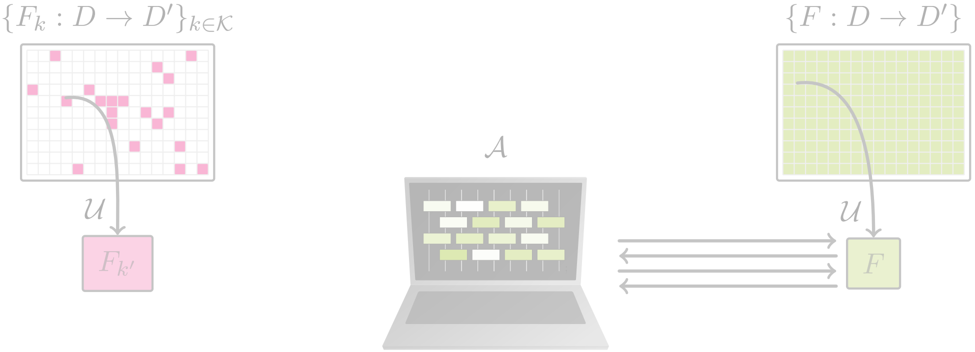
A collection of keyed functions $\{F_k : D \to D'\}_{k\in\mathcal{K}}$ that cannot be distinguished from uniformly random functions in $\{F : D \to D'\}$ by any polynomial-time algorithm $\mathcal{A}$.



Distributions that are hard to learn classically (1)

Pseudorandom function (PRF)

A collection of keyed functions $\{F_k : D \to D'\}_{k\in\mathcal{K}}$ that cannot be distinguished from uniformly random functions in $\{F : D \to D'\}$ by any polynomial-time algorithm $\mathcal{A}$.



$$\left| \Pr_{k\leftarrow U(\mathcal{K})} [\mathcal{A}^{O(F_k)} = 1] - \Pr_{F\leftarrow U(\mathcal{F})} [\mathcal{A}^{O(F)} = 1] \right| < \text{negl.?}$$

problem size problem size

Distributions that are hard to learn classically (1)

Pseudorandom function (PRF)

A collection of keyed functions $\{F_k : D \to D'\}_{k \in \mathcal{K}}$ that cannot be distinguished from uniformly random functions in $\{F : D \to D'\}$ by any polynomial-time algorithm $\mathcal{A}$.

$\{F_k : D \to D'\}_{k \in \mathcal{K}}$

Oracles & Security

Classical algorithms $\mathcal{A}$

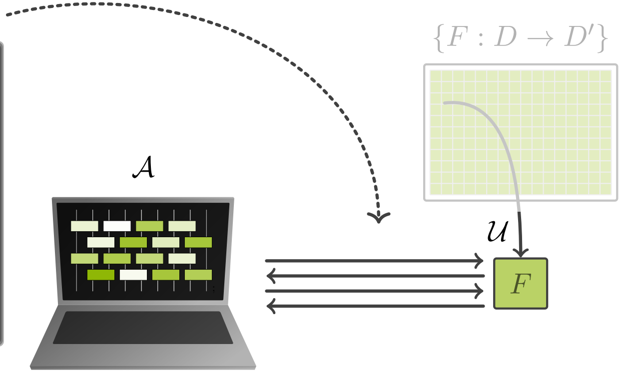
random — weak-secure

membership — classical-secure

Quantum algorithms $\mathcal{A}$

random — standard-secure

qu. memb. — quantum-secure



$$\left| \Pr_{\substack{k \leftarrow U(\mathcal{K}) \\ \text{problem size}}} \left[\mathcal{A}^{O(F_k)} = 1\right] - \Pr_{\substack{F \leftarrow U(\mathcal{F}) \\ \text{problem size}}} \left[\mathcal{A}^{O(F)} = 1\right] \right| < \text{negl.?}$$

Distributions that are hard to learn classically (1)

Pseudorandom function (PRF)

A collection of keyed functions $\{F_k : D \to D'\}_{k \in \mathcal{K}}$ that cannot be distinguished from uniformly random functions in $\{F : D \to D'\}$ by any polynomial-time algorithm $\mathcal{A}$.

$\{F_k : D \to D'\}_{k \in \mathcal{K}}$

Oracles & Security

Classical algorithms $\mathcal{A}$

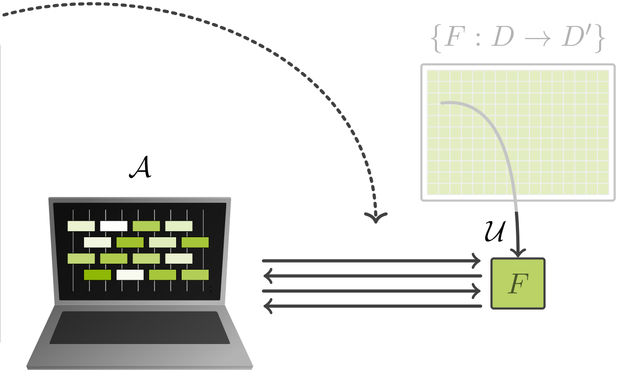
random   weak-secure

membership   classical-secure

Quantum algorithms $\mathcal{A}$

random   standard-secure

qu. memb.   quantum-secure



$$\left| \Pr_{\substack{k \leftarrow U(\mathcal{K})}} \left[\mathcal{A}^{O(F_k)} = 1\right] - \Pr_{\substack{F \leftarrow U(\mathcal{F})}} \left[\mathcal{A}^{O(F)} = 1\right] \right| < \text{negl.?}$$

problem size   problem size

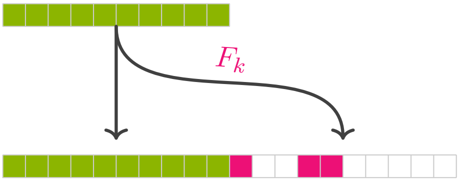
Distributions that are hard to learn classically (2)

Theorem (Kearns et al., '94)

Given a classical-secure PRF $\{F_k\}_k$, the distribution class $\{D_k\}_k$ defined by the "Kearns generator"

$$\mathit{KGEN}_k(x) = x||F_k(x)$$

cannot be efficiently classically generator-learned.



Proof idea

If such a learning algorithm $\tilde{\mathcal{A}}$ exists,
then we can use this algorithm to construct an efficient adversary $\mathcal{A}$ for the PRF!

Kearns et al.: *On the learnability of discrete distributions* STOC (1995).

Digging deeper: How to construct PRFs

∃ one-way function

↓

∃ pseudorandom generator

↓

∃ pseudorandom function

↓

∃ hard-to-learn distribution class

Digging deeper: How to construct PRFs

∃ one-way function

↓

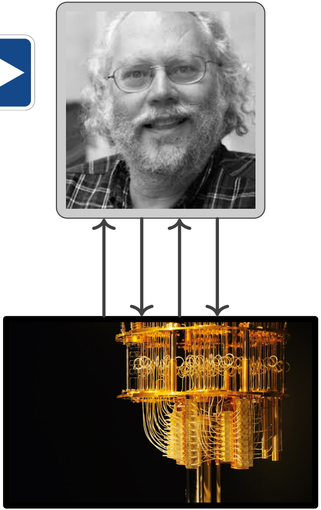
∃ pseudorandom generator

↓

∃ pseudorandom function

↓

∃ hard-to-learn distribution class



Digging deeper: How to construct PRFs

∃ one-way function

adversary?

∃ pseudorandom generator

adversary?

∃ pseudorandom function

adversary?

∃ hard-to-learn distribution class

Digging deeper: How to construct PRFs

$\exists$ one-way function

↓

$\exists$ pseudorandom generator

↓

$\exists$ pseudorandom function

↓

$\exists$ hard-to-learn distribution class

Digging deeper: How to construct PRFs

∃ one-way function

↓

∃ pseudorandom generator

↓

∃ pseudorandom function

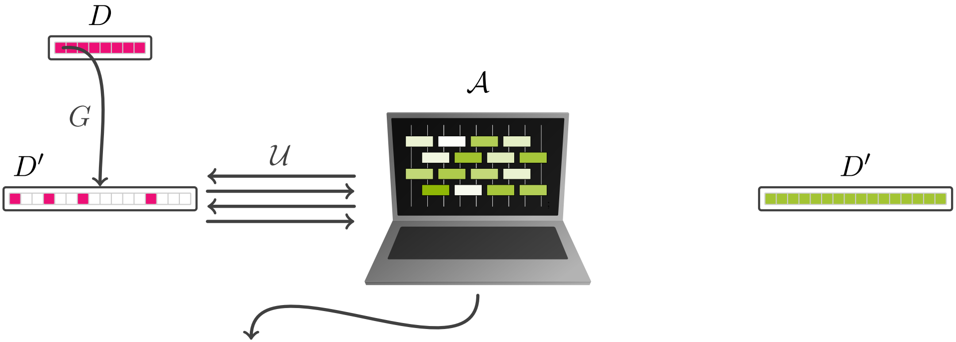
↓

∃ hard-to-learn distribution class

Constructing PRFs from PRGs

What is a pseudorandom generator?

An efficiently computable function $G : D \to D'$ is called a pseudorandom generator if $G(x), x \leftarrow \mathcal{U}(D)$ cannot be efficiently distinguished from a uniformly random $y \leftarrow \mathcal{U}(D')$

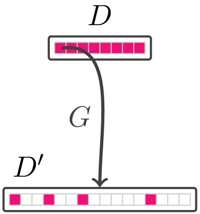


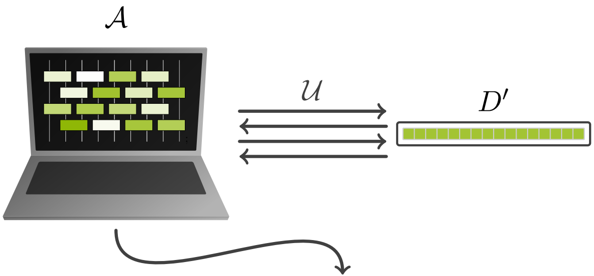
$$\left| \Pr_{\underbrace{x \leftarrow \mathcal{U}(D)}_{\text{problem size}}} [\mathcal{A}(G(x)) = 1] - \Pr_{\underbrace{y \leftarrow \mathcal{U}(D')}_{\text{problem size}}} [\mathcal{A}(y) = 1] \right| < \text{negl.?}$$

Constructing PRFs from PRGs

What is a pseudorandom generator?

An efficiently computable function $G : D \to D'$ is called a pseudorandom generator if $G(x), x \leftarrow \mathcal{U}(D)$ cannot be efficiently distinguished from a uniformly random $y \leftarrow \mathcal{U}(D')$



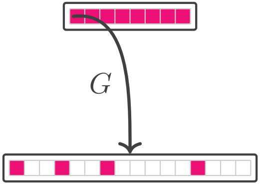


$$\Big| \underbrace{\Pr_{x \leftarrow \mathcal{U}(D)}}_{\text{problem size}} [\mathcal{A}(G(x)) = 1] - \underbrace{\Pr_{y \leftarrow \mathcal{U}(D')}}_{\text{problem size}} [\mathcal{A}(y) = 1] \Big| < \text{negl.?}$$

Constructing PRFs from PRGs

Input: length-doubling PRG:

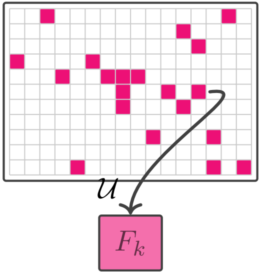
$$G : D \to D \times D$$
$$x \mapsto G(x) =: G^0(x) || G^1(x).$$



Goal: Construct a PRF

$$F : D \times \{0,1\}^n \to D,$$

such that for $k \in \mathcal{K} \equiv D$, F_k looks random.



Goldreich, Goldwasser, Micali: *How to construct random functions.* J. ACM (1986).

Constructing PRFs from PRGs

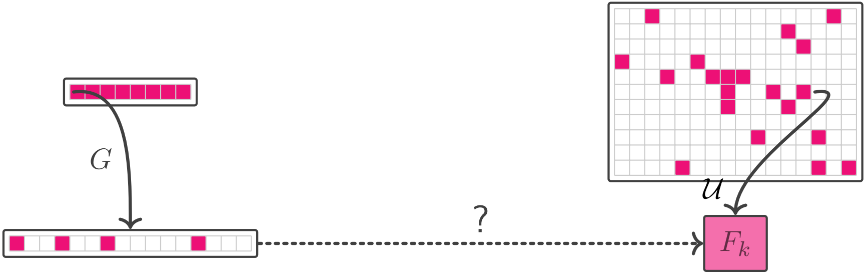
Input: length-doubling PRG:

$$G : D \to D \times D$$
$$x \mapsto G(x) =: G^0(x) || G^1(x).$$

Goal: Construct a PRF

$$F : D \times \{0,1\}^n \to D,$$

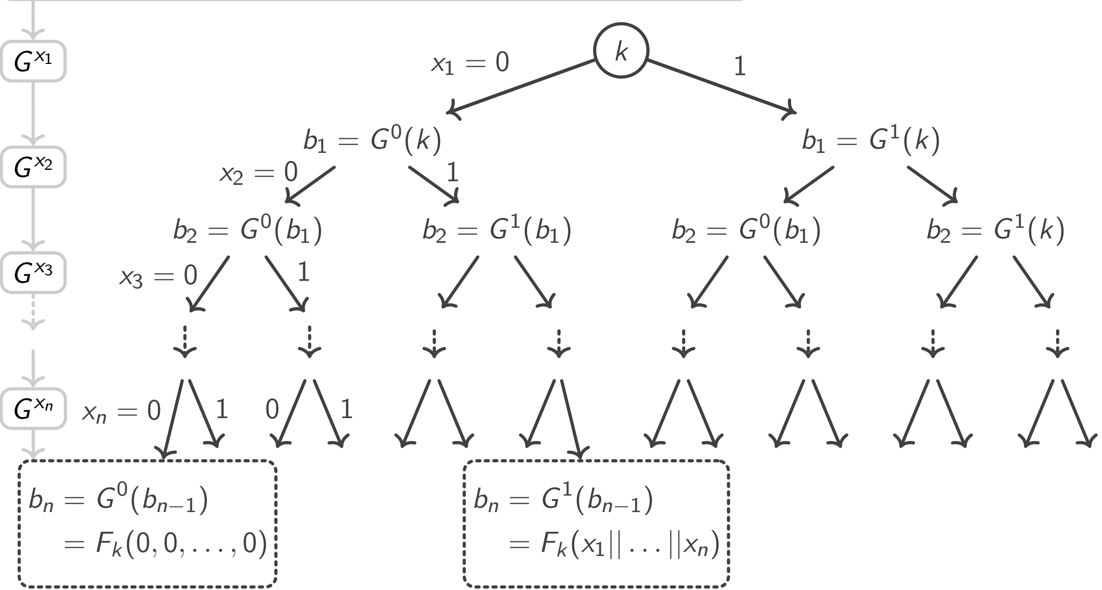
such that for $k \in \mathcal{K} \equiv D$, F_k looks random.



Goldreich, Goldwasser, Micali: *How to construct random functions.* J. ACM (1986).

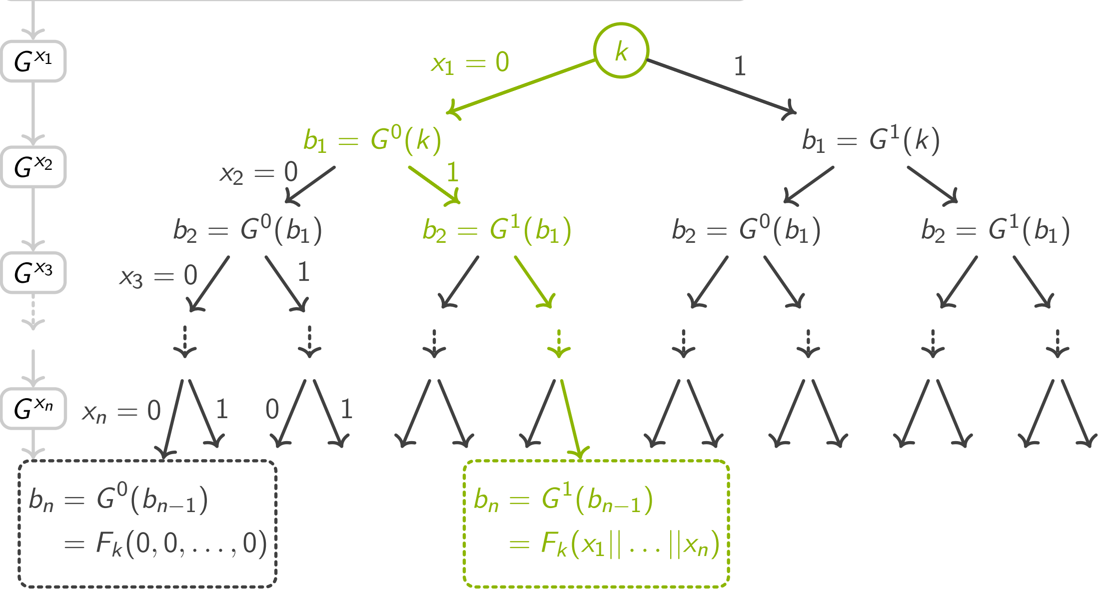
Constructing PRFs from PRGs: The GGM construction

Input: $G = G^0 || G^1$, key $k \in \mathcal{K}$, input string $x = x_1 || \ldots || x_n$

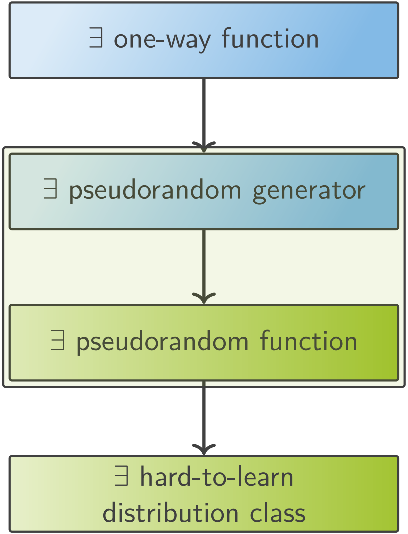


Constructing PRFs from PRGs: The GGM construction

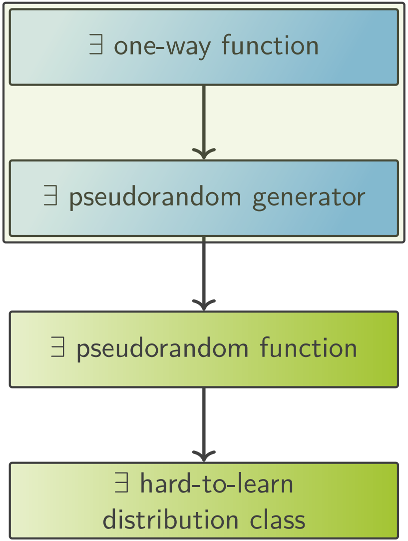
Input: $G = G^0 || G^1$, key $k \in \mathcal{K}$, input string $x = x_1 || \ldots || x_n$



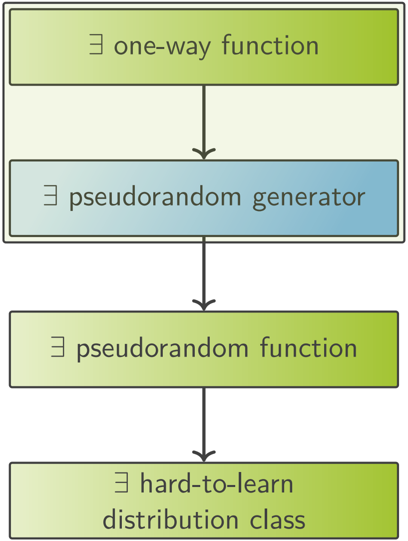
Digging deeper: How to construct PRGs



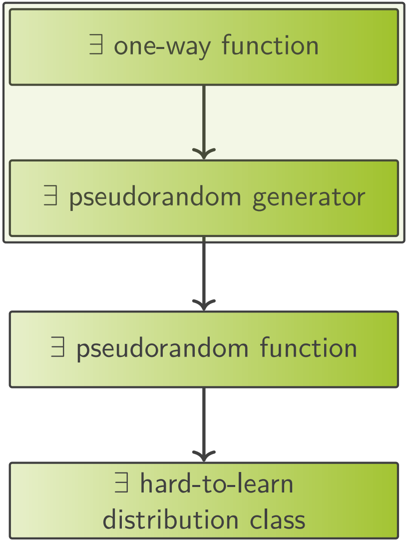
Digging deeper: How to construct PRGs



Digging deeper: How to construct PRGs



Digging deeper: How to construct PRGs



Constructing PRGs from one-way-functions: Discrete logarithm and DDH

Modular Exponentiation: p prime, g generator of $\mathbb{Z}_p^*$

$$\begin{aligned}\mathsf{modexp}_{g,p} : \mathbb{N} &\to \mathbb{Z}_p \\ x &\mapsto g^x \bmod p\end{aligned}$$

Discrete logarithm

Given $y = g^x \bmod p$

$$\mathsf{dlog}_{g,p}(y) = x$$

Constructing PRGs from one-way-functions: Discrete logarithm and DDH

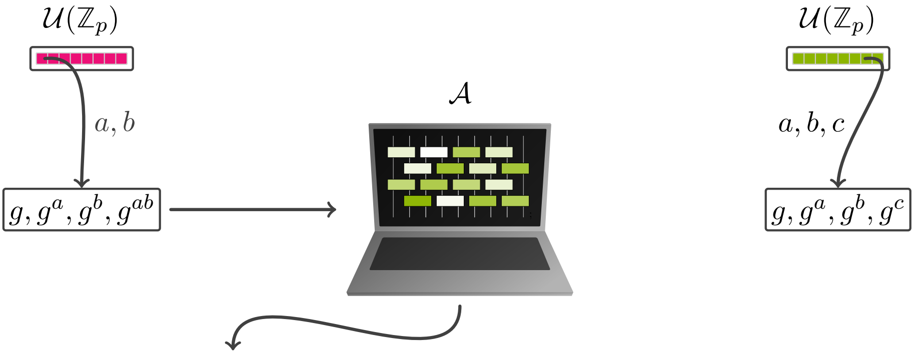
Modular Exponentiation: p prime, g generator of $\mathbb{Z}_p^*$

$$\begin{aligned} \mathrm{modexp}_{g,p} : \mathbb{N} &\to \mathbb{Z}_p \\ x &\mapsto g^x \bmod p \end{aligned}$$

Discrete logarithm

Given $y = g^x \bmod p$

$$\mathrm{dlog}_{g,p}(y) = x$$



$$\left| \underbrace{\Pr_{a,b \leftarrow \mathcal{U}(\mathbb{Z}_p)}}_{\text{problem size}} [\mathcal{A}(g, g^a, g^b, g^{ab}) = 1] - \underbrace{\Pr_{a,b,c \leftarrow \mathcal{U}(\mathbb{Z}_p)}}_{\text{problem size}} [\mathcal{A}(g, g^a, g^b, g^c) = 1] \right| < \text{negl.?}$$

Constructing PRGs from one-way-functions: Discrete logarithm and DDH

Modular Exponentiation: p prime, g generator of $\mathbb{Z}_p^*$

$$\mathsf{modexp}_{g,p} : \mathbb{N} \to \mathbb{Z}_p$$
$$x \mapsto g^x \bmod p$$

Discrete logarithm

Given $y = g^x \bmod p$

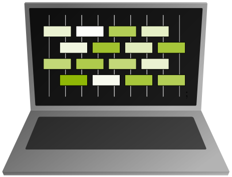
$$\mathsf{dlog}_{g,p}(y) = x$$

$\mathcal{U}(\mathbb{Z}_p)$

a, b

g, g^a, g^b, g^{ab}

$\mathcal{A}$



$\mathcal{U}(\mathbb{Z}_p)$

a, b, c

g, g^a, g^b, g^c

$$\left| \underset{\text{problem size}}{\Pr_{a,b \leftarrow \mathcal{U}(\mathbb{Z}_p)}} [\mathcal{A}(g, g^a, g^b, g^{ab}) = 1] - \underset{\text{problem size}}{\Pr_{a,b,c \leftarrow \mathcal{U}(\mathbb{Z}_p)}} [\mathcal{A}(g, g^a, g^b, g^c) = 1] \right| < \text{negl.?}$$

The DDH assumption for quadratic residues

The DDH assumption is not believed to hold for all $\mathbb{Z}_p^*$.

e.g., if $p-1$ has small prime factors DDH is false.

Blum, Micali: FOCS '82.

Boneh: The decisional Diffie-Hellman problem. '98

The DDH assumption for quadratic residues

The DDH assumption is not believed to hold for all $\mathbb{Z}_p^*$.

e.g., if $p-1$ has small prime factors DDH is false.

Quadratic residues for safe primes

$p = 2q+1$, q, p prime.

$$\mathrm{QR}_p = \{y \in \mathbb{Z}_p^* : \exists x \in \mathbb{Z}_p^* \text{ s.t. } x^2 = y \bmod p\} \simeq \mathbb{Z}_q$$

Blum, Micali: FOCS '82.

Boneh: The decisional Diffie-Hellman problem. '98

The DDH assumption for quadratic residues

The DDH assumption is not believed to hold for all $\mathbb{Z}_p^*$.

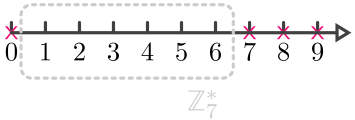
e.g., if $p-1$ has small prime factors DDH is false.

Quadratic residues for safe primes

$p = 2q+1$, q, p prime.

$$\mathrm{QR}_p = \{y \in \mathbb{Z}_p^* : \exists x \in \mathbb{Z}_p^* \text{ s.t. } x^2 = y \bmod p\} \simeq \mathbb{Z}_q$$

Example: $p = 7 = 2 \cdot 3 + 1$



Blum, Micali: FOCS '82.

Boneh: The decisional Diffie-Hellman problem. '98

The DDH assumption for quadratic residues

The DDH assumption is not believed to hold for all $\mathbb{Z}_p^*$.

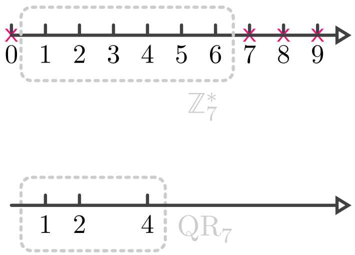
e.g., if $p-1$ has small prime factors DDH is false.

Quadratic residues for safe primes

$p = 2q+1$, q, p prime.

$$\mathrm{QR}_p = \{y \in \mathbb{Z}_p^* : \exists x \in \mathbb{Z}_p^* \text{ s.t. } x^2 = y \bmod p\} \simeq \mathbb{Z}_q$$

Example: $p = 7 = 2 \cdot 3 + 1$



Blum, Micali: FOCS '82.

Boneh: The decisional Diffie-Hellman problem. '98

The DDH assumption for quadratic residues

The DDH assumption is not believed to hold for all $\mathbb{Z}_p^*$.

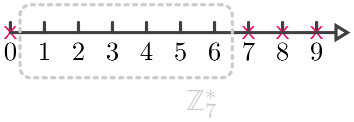
e.g., if $p-1$ has small prime factors DDH is false.

Quadratic residues for safe primes

$p = 2q+1$, q, p prime.

$$\mathrm{QR}_p = \{y \in \mathbb{Z}_p^* : \exists x \in \mathbb{Z}_p^* \text{ s.t. } x^2 = y \bmod p\} \simeq \mathbb{Z}_q$$

Example: $p = 7 = 2 \cdot 3 + 1$



e.g. $3^2 \bmod 7 = 2$

Blum, Micali: FOCS '82.

Boneh: The decisional Diffie-Hellman problem. '98

The DDH assumption for quadratic residues

The DDH assumption is not believed to hold for all $\mathbb{Z}_p^*$.

e.g., if $p-1$ has small prime factors DDH is false.

Quadratic residues for safe primes

$p = 2q+1$, q, p prime.

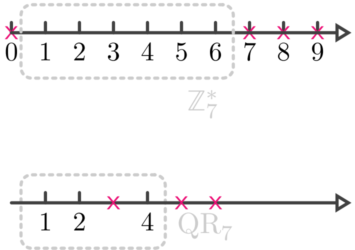
$$\mathrm{QR}_p = \{y \in \mathbb{Z}_p^* : \exists x \in \mathbb{Z}_p^* \text{ s.t. } x^2 = y \bmod p\} \simeq \mathbb{Z}_q$$

Efficient bijection $\mathrm{QR}_p \leftrightarrow \mathbb{Z}_q^*$

$$f_p : \mathrm{QR}_p \to \mathbb{Z}_q^*$$

$$x \mapsto f_p(x) := \begin{cases} x & x \leq q \\ p-x & x > q \end{cases}$$

Example: $p = 7 = 2 \cdot 3 + 1$



e.g. $3^2 \bmod 7 = 2$

Blum, Micali: FOCS '82.

Boneh: The decisional Diffie-Hellman problem. '98

The DDH assumption for quadratic residues

The DDH assumption is not believed to hold for all $\mathbb{Z}_p^*$.

e.g., if $p-1$ has small prime factors DDH is false.

Quadratic residues for safe primes

$p = 2q+1$, q, p prime.

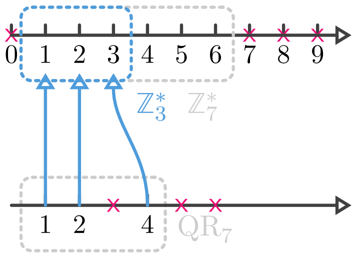
$$\mathrm{QR}_p = \{y \in \mathbb{Z}_p^* : \exists x \in \mathbb{Z}_p^* \text{ s.t. } x^2 = y \text{ mod } p\} \simeq \mathbb{Z}_q$$

Efficient bijection $\mathrm{QR}_p \leftrightarrow \mathbb{Z}_q^*$

$$f_p : \mathrm{QR}_p \to \mathbb{Z}_q^*$$

$$x \mapsto f_p(x) := \begin{cases} x & x \le q \\ p - x & x > q \end{cases}$$

Example: $p = 7 = 2 \cdot 3 + 1$



e.g. 3^2 mod $7 = 2$

Blum, Micali: FOCS '82.

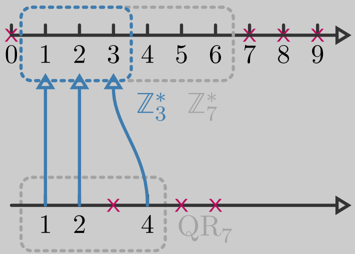
Boneh: The decisional Diffie-Hellman problem. '98

The DDH assumption for quadratic residues

Important properties

1. believed to satisfy the DDH property (Boneh '98)
2. A problem instance can be efficiently generated.
   - membership in QR_p can be checked efficiently.
   - safe primes of a given length can be efficiently generated.
   - almost all elements are generators.

Example: $p = 7 = 2 \cdot 3 + 1$



e.g. $3^2 \bmod 7 = 2$

Efficient bijection $\mathrm{QR}_p \leftrightarrow \mathbb{Z}_q^*$

$$f_p : \mathrm{QR}_p \to \mathbb{Z}_q^*$$

$$x \mapsto f_p(x) := \begin{cases} x & x \leq q \\ p - x & x > q \end{cases}$$

Blum, Micali: FOCS '82.

Boneh: The decisional Diffie-Hellman problem. '98

The DDH assumption for quadratic residues

Important properties

1. believed to satisfy the DDH property (Boneh '98)
2. A problem instance can be efficiently generated.
   - membership in QR_p can be checked efficiently.
   - safe primes of a given length can be efficiently generated.
   - almost all elements are generators.

A pseudorandom generator from QR_p

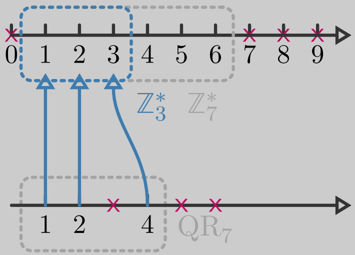
$$\tilde{G}_{(p,g,g^a)} : \mathbb{Z}_q \to \mathrm{QR}_p \times \mathrm{QR}_p$$

$$b \mapsto g^b \bmod p || g^{ab} \bmod p$$

$$= G^0_{(p,g,g^a)}(b) || G^1_{(p,g,g^a)}(b)$$

$$G_{(p,g,g^a)} = f_p \circ \tilde{G}_{(p,g,g^a)} : \mathbb{Z}_q \to \mathbb{Z}_q \times \mathbb{Z}_q$$

Example: $p = 7 = 2 \cdot 3 + 1$



e.g. $3^2 \bmod 7 = 2$

The full construction

1. Make the DDH assumption for the group family of quadratic residues

$$\mathsf{modexp}_{p,g}(x) = g^x \bmod p.$$

2. Define the pseudorandom generator

$$G_{(p,g,g^a)} : \mathbb{Z}_q \to \mathbb{Z}_q \times \mathbb{Z}_q \text{ with } a \in \mathbb{Z}_q$$
$$b \mapsto f_p(g^b \bmod p) || f_p(g^{ab} \bmod p) \equiv G^0_{(p,g,g^a)}(b) || G^1_{(p,g,g^a)}(b).$$

3. Define the pseudorandom function

$$F_{(p,g,g^a),b} : \mathbb{Z}_q \times \{0,1\}^n \to \mathbb{Z}_q \text{ with } b \leftarrow \mathcal{U}(\mathbb{Z}_q)$$

via the GGM construction using $G^0_{(p,g,g^a)}, G^1_{(p,g,g^a)}$.

4. Define the distribution class $\{D_{(p,g,g^a),b}\}_b$ on $\{0,1\}^{2n+m}$ via the (modified) Kearns generator

$$\mathsf{GEN}(x) = x || \mathsf{BIN}_n(F_{(p,g,g^a),b}(x)) || \mathsf{BIN}_m(p, g, g^a) \text{ with } x \leftarrow \mathcal{U}(\{0,1\}^n).$$

The full construction

1. Make the DDH assumption for the group family of quadratic residues

$$\mathsf{modexp}_{p,g}(x) = g^x \bmod p.$$

2. Define the pseudorandom generator

$$\begin{aligned} G_{(p,g,g^a)} &: \mathbb{Z}_q \to \mathbb{Z}_q \times \mathbb{Z}_q \text{ with } a \in \mathbb{Z}_q \\ &b \mapsto f_p(g^b \bmod p) || f_p(g^{ab} \bmod p) \equiv G^0_{(p,g,g^a)}(b) || G^1_{(p,g,g^a)}(b). \end{aligned}$$

3. Define the pseudorandom function

$$F_{(p,g,g^a),b} : \mathbb{Z}_q \times \{0,1\}^n \to \mathbb{Z}_q \text{ with } b \leftarrow \mathcal{U}(\mathbb{Z}_q)$$

via the GGM construction using $G^0_{(p,g,g^a)}, G^1_{(p,g,g^a)}$.

4. Define the distribution class $\{D_{(p,g,g^a),b}\}_b$ on $\{0,1\}^{2n+m}$ via the (modified) Kearns generator

$$\mathsf{GEN}(x) = x || \mathsf{BIN}_n(F_{(p,g,g^a),b}(x)) || \mathsf{BIN}_m(p, g, g^a) \text{ with } x \leftarrow \mathcal{U}(\{0,1\}^n).$$

The full construction

1. Make the DDH assumption for the group family of quadratic residues

$$\mathsf{modexp}_{p,g}(x) = g^x \text{ mod } p.$$

2. Define the pseudorandom generator

$$G_{(p,g,g^a)} : \mathbb{Z}_q \to \mathbb{Z}_q \times \mathbb{Z}_q \text{ with } a \in \mathbb{Z}_q$$
$$b \mapsto f_p(g^b \text{ mod } p) || f_p(g^{ab} \text{ mod } p) \equiv G^0_{(p,g,g^a)}(b) || G^1_{(p,g,g^a)}(b).$$

3. Define the pseudorandom function

$$F_{(p,g,g^a),b} : \mathbb{Z}_q \times \{0,1\}^n \to \mathbb{Z}_q \text{ with } b \leftarrow \mathcal{U}(\mathbb{Z}_q)$$

via the GGM construction using $G^0_{(p,g,g^a)}, G^1_{(p,g,g^a)}$.

4. Define the distribution class $\{D_{(p,g,g^a),b}\}_b$ on $\{0,1\}^{2n+m}$ via the (modified) Kearns generator

$$\mathsf{GEN}(x) = x || \mathsf{BIN}_n(F_{(p,g,g^a),b}(x)) || \mathsf{BIN}_m(p,g,g^a) \text{ with } x \leftarrow \mathcal{U}(\{0,1\}^n).$$

The full construction

1. Make the DDH assumption for the group family of quadratic residues

$$\mathsf{modexp}_{p,g}(x) = g^x \bmod p.$$

2. Define the pseudorandom generator

$$G_{(p,g,g^a)} : \mathbb{Z}_q \to \mathbb{Z}_q \times \mathbb{Z}_q \text{ with } a \in \mathbb{Z}_q$$
$$b \mapsto f_p(g^b \bmod p) || f_p(g^{ab} \bmod p) \equiv G^0_{(p,g,g^a)}(b) || G^1_{(p,g,g^a)}(b).$$

3. Define the pseudorandom function

$$F_{(p,g,g^a),b} : \mathbb{Z}_q \times \{0,1\}^n \to \mathbb{Z}_q \text{ with } b \leftarrow \mathcal{U}(\mathbb{Z}_q)$$

via the GGM construction using $G^0_{(p,g,g^a)}, G^1_{(p,g,g^a)}$.

4. Define the distribution class $\{D_{(p,g,g^a),b}\}_b$ on $\{0,1\}^{2n+m}$ via the (modified) Kearns generator

$$\mathsf{GEN}(x) = x || \mathsf{BIN}_n(F_{(p,g,g^a),b}(x)) || \mathsf{BIN}_m(p, g, g^a) \text{ with } x \leftarrow \mathcal{U}(\{0,1\}^n).$$

Cracking hard-to-learn distribution classes with a quantum computer

$\exists$ one-way function

↓

$\exists$ pseudorandom generator

↓

$\exists$ pseudorandom function

↓

$\exists$ hard-to-learn distribution class

Cracking hard-to-learn distribution classes with a quantum computer

∃ one-way function

quantum adversary

∃ pseudorandom generator

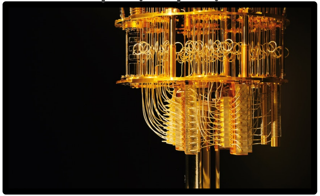
quantum adversary

∃ pseudorandom function

quantum adversary

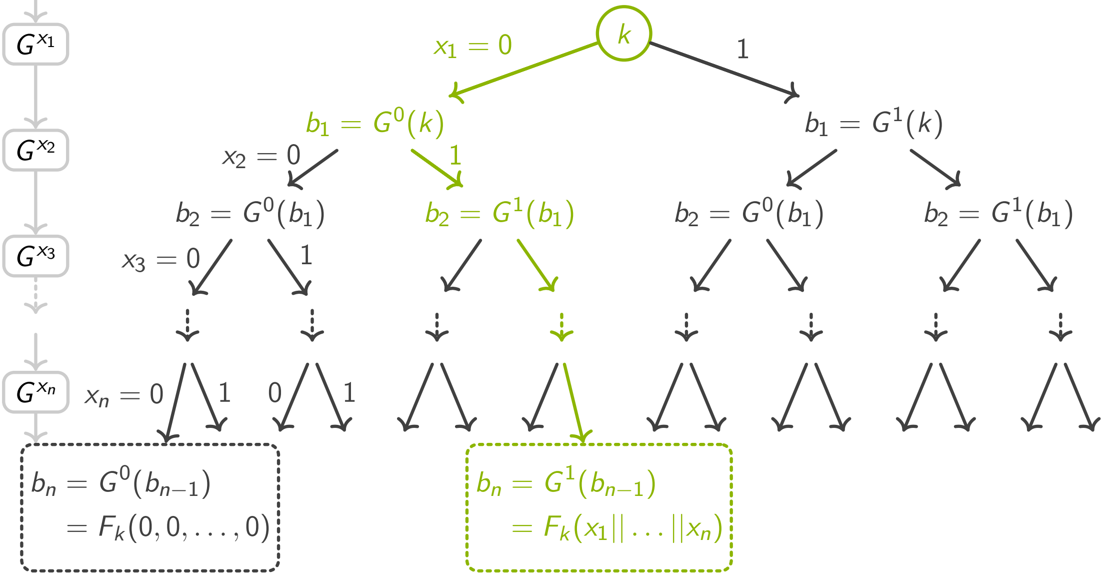
∃ hard-to-learn distribution class

quantum adversary

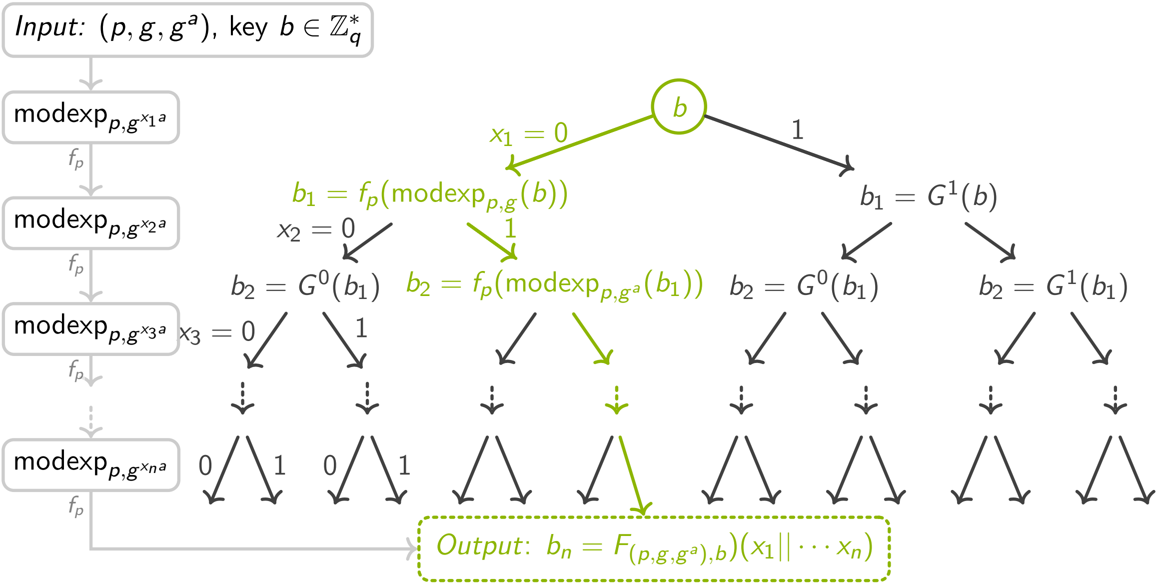


Cracking the GGM tree

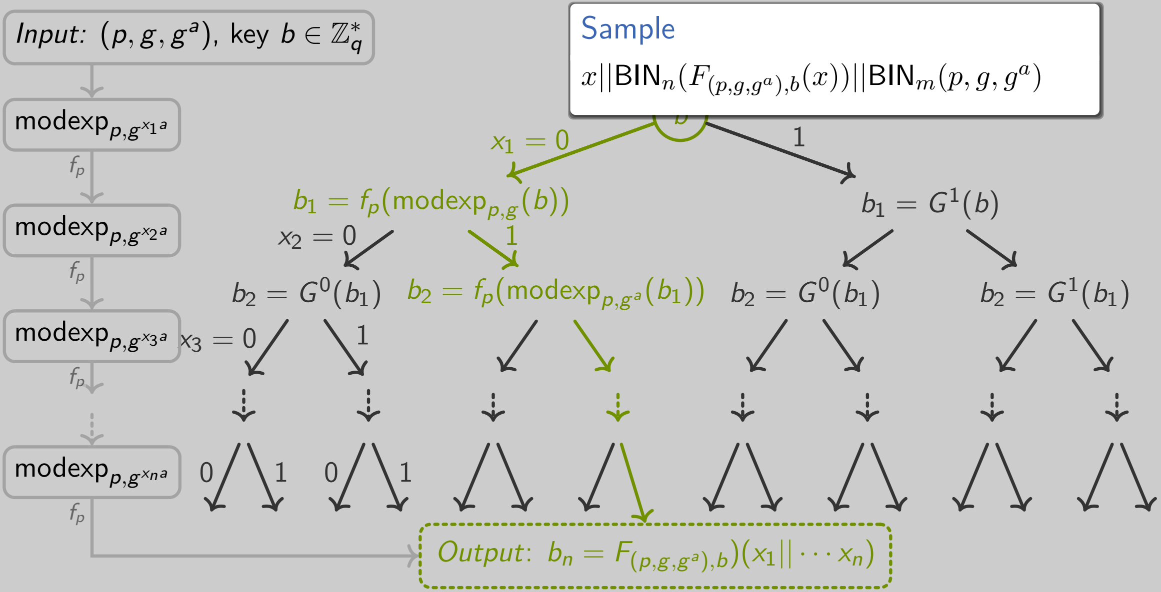
Input: $G = G^0 || G^1$, key $k \in \mathcal{K}$, input string $x = x_1 || \ldots || x_n$



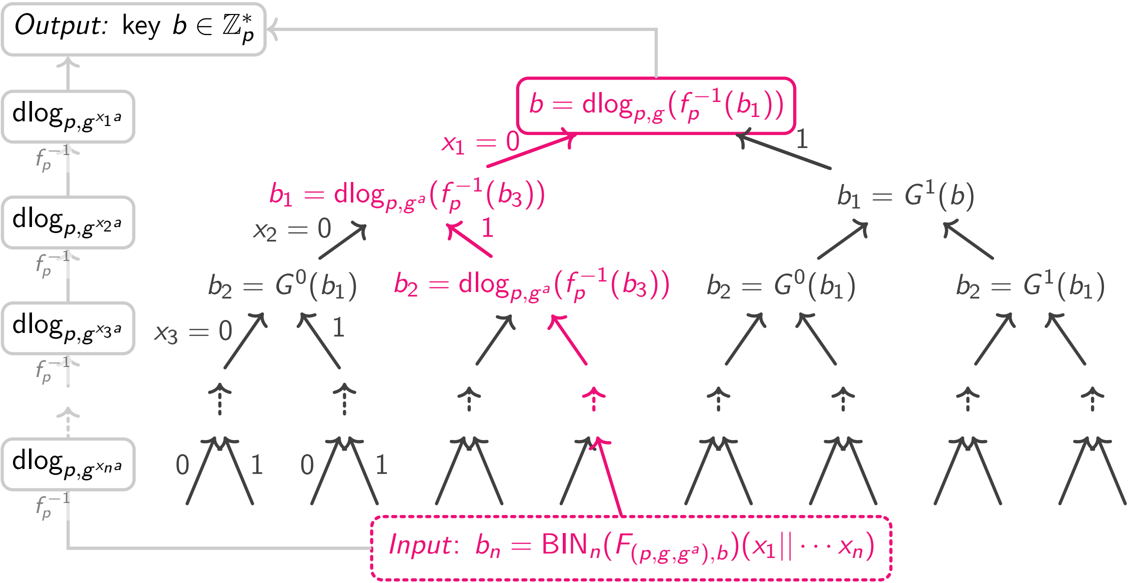
Cracking the GGM tree



Cracking the GGM tree

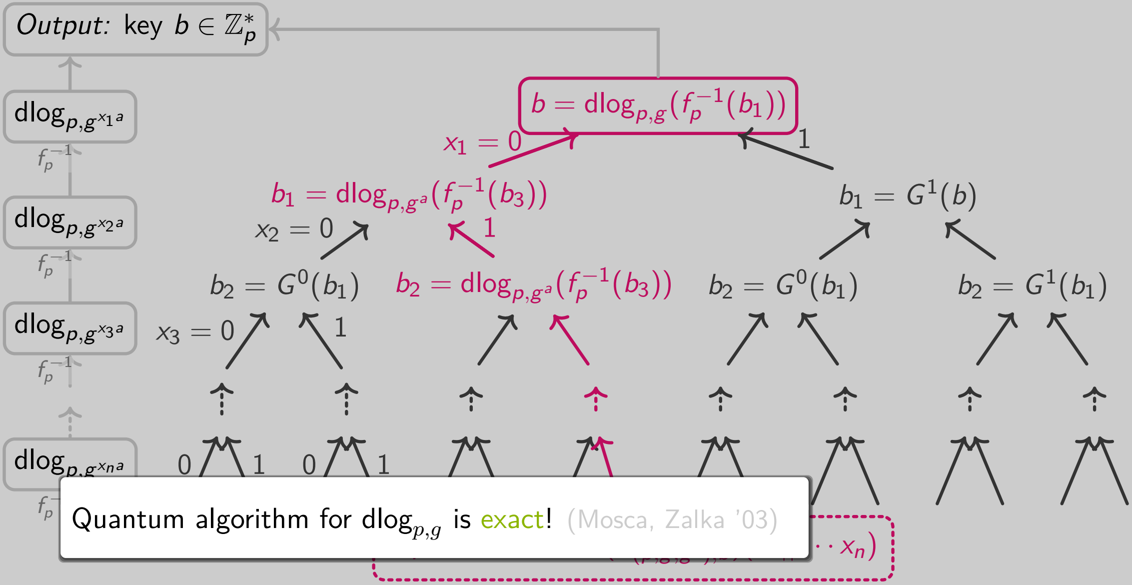


Cracking the GGM tree



Mosca, Zalka: Exact QFTs and dLogs. arXiv:quant-ph/0301093 (2003)

Cracking the GGM tree



Quantum algorithm for $\mathrm{dlog}_{p,g}$ is exact! (Mosca, Zalka '03)

Mosca, Zalka: Exact QFTs and dLogs. arXiv:quant-ph/0301093 (2003)

Cracking the GGM tree

Output: key $b \in \mathbb{Z}_p^*$

$\mathrm{dlog}_{p,g^{x_1}a}$

f_p^{-1}

$\mathrm{dlog}_{p,g^{x_2}a}$

f_p^{-1}

$\mathrm{dlog}_{p,g^{x_3}a}$

f_p^{-1}

$\mathrm{dlog}_{p,g^{x_n}a}$

f_p^{-1}

b_1 $= G^1(b)$

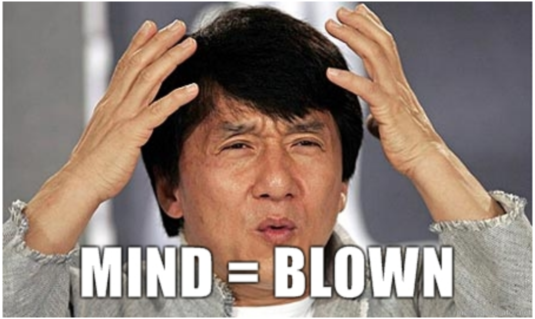
x_2

$b_2 = G$ $\quad b_2 = G^1(b_1)$

$x_3 = 0$

0 1 0 1

$\cdots x_n)$



The distribution $D_{(p,g,g^a),b}$ can be exactly PAC generator-learned from a single sample!

Quantum algorithm for $\mathrm{dlog}_{p,g}$ is exact! (Mosca, Zalka '03)

Mosca, Zalka: Exact QFTs and dLogs. arXiv:quant-ph/0301093 (2003)

Figure Ref: The internet.

A quantum vs. classical separation for distribution learning

Question: Quantum generator-learning advantage?

Is there a class of efficiently classically generated discrete distributions which is

- not efficiently classsical PAC generator-learnable, but
- efficiently quantum PAC generator-learnable

w.r.t. the SAMPLE oracle and the KL divergence?

Theorem: **Y E S*** **!**

*under the decisional Diffie-Hellman assumption for the group family of quadratic residues

Coyle et al., NPJ Quant. Inf. (2020).

Wrap up

Discussion

PROs

- Our result shows that discrete distributions admit structure that can be exploited by quantum computers.

CONs

- The result is not a practical result and (a bit) artificial.
  - a single sample always suffices for learning.
  - the learning algorithm is always exact.

OUTlook

- Really, we would like to show a quantum advantage for a relevant problem, for example, learning 'mixtures of Gaussians'.
- Weaken the crypto assumptions – e.g., are weak PRFs sufficient for hardness?

Discussion

PROs

- Our result shows that discrete distributions admit structure that can be exploited by quantum computers.

CONs

- The result is not a practical result and (a bit) artificial.
  - a single sample always suffices for learning.
  - the learning algorithm is always exact.

OUTlook

- Really, we would like to show a quantum advantage for a relevant problem, for example, learning ‘mixtures of Gaussians’.
- Weaken the crypto assumptions – e.g., are weak PRFs sufficient for hardness?

Discussion

arXiv:2007.14451

PROs

- Our result shows that discrete distributions admit structure that can be exploited by quantum computers.

CONs

- The result is not a practical result and (a bit) artificial.
  - a single sample always suffices for learning.
  - the learning algorithm is always exact.

OUTlook

- Really, we would like to show a quantum advantage for a relevant problem, for example, learning 'mixtures of Gaussians'.
- Weaken the crypto assumptions – e.g., are weak PRFs sufficient for hardness?

THANK YOU!