Classical verification of quantum computational advantage

Gregory D. Kahanamoku-Meyer
November 10, 2021

Theory collaborators:
Norman Yao (Berkeley Physics)
Umesh Vazirani (Berkeley CS)
Soonwon Choi (MIT Physics)

arXiv:1912.05547
arXiv:2104.00687
Quantum computational advantage

Recent experimental demonstrations:

Random circuit sampling
[Arute et al., Nature '19]

Gaussian boson sampling
[Zhong et al., Science '20]
Quantum computational advantage

Recent experimental demonstrations:

Random circuit sampling  
[Arute et al., Nature '19]

Gaussian boson sampling  
[Zhong et al., Science '20]

Largest experiments → “impossible” to classically simulate
Quantum computational advantage

Recent experimental demonstrations:

Random circuit sampling [Arute et al., Nature '19]

Gaussian boson sampling [Zhong et al., Science '20]

Largest experiments → “impossible” to classically simulate

“... [Rule] out alternative [classical] hypotheses that might be plausible in this experiment” [Zhong et al.]
Quantum computational advantage

Recent experimental demonstrations:

Random circuit sampling
[Arute et al., Nature '19]

Gaussian boson sampling
[Zhong et al., Science '20]

Largest experiments → “impossible” to classically simulate

“... [Rule] out alternative [classical] hypotheses that might be plausible in this experiment” [Zhong et al.]

Quantum is the only reasonable explanation for observed behavior
Efficiently-verifiable test that only quantum computers can pass.
"Black-box" proofs of quantumness

Efficiently-verifiable test that only quantum computers can pass.

For polynomially-bounded classical verifier:

Completeness
∃ BQP prover s.t. Verifier accepts w.p. > $2/3$

Soundness
∀ BPP provers, Verifier accepts w.p. < $1/3$
“Black-box” proofs of quantumness

Efficiently-verifiable test that only quantum computers can pass.

For polynomially-bounded classical verifier:

- **Completeness**: \( \exists \) BQP prover s.t. Verifier accepts w.p. > \( \frac{2}{3} \)
- **Soundness**: \( \forall \) BPP provers, Verifier accepts w.p. < \( \frac{1}{3} \)

Fully classical verifier (and comms.),
"Black-box" proofs of quantumness

Efficiently-verifiable test that only quantum computers can pass.

For polynomially-bounded classical verifier:

Completeness

∃ BQP prover s.t. Verifier accepts w.p. > 2/3

Soundness

∀ BPP provers, Verifier accepts w.p. < 1/3

Fully classical verifier (and comms.), single black-box prover,
“Black-box” proofs of quantumness

Efficiently-verifiable test that only quantum computers can pass.

For polynomially-bounded classical verifier:

- **Completeness**
  - ∃ BQP prover s.t. Verifier accepts w.p. > 2/3

- **Soundness**
  - ∀ BPP provers, Verifier accepts w.p. < 1/3

Fully classical verifier (and comms.), single black-box prover, superpolynomial computational separation
“Black-box” proofs of quantumness

Efficiently-verifiable test that only quantum computers can pass.

Local: powerfully refute the extended Church-Turing thesis
“Black-box” proofs of quantumness

Efficiently-verifiable test that only quantum computers can pass.

Local: powerfully refute the extended Church-Turing thesis

Remote: validate an untrusted quantum cloud service
NISQ verifiable quantum advantage

Trivial solution: Shor’s algorithm
NISQ verifiable quantum advantage

Trivial solution: Shor’s algorithm... but we want to do near-term!
NISQ verifiable quantum advantage

Trivial solution: Shor’s algorithm... but we want to do near-term!

**Sampling problems**
e.g. random circuits, Boson sampling, ...  
✓ NISQ feasible  
✗ Efficiently verifiable

**Number theory problems**
e.g. factoring, discrete logarithm, ...  
✗ NISQ feasible  
✓ Efficiently verifiable

add structure  
make less costly

???
✓ NISQ feasible  
✓ Efficiently verifiable
Adding structure to sampling problems

Idea: some *property* of samples that we can check?
Adding structure to sampling problems

Idea: some *property* of samples that we can check?
Generically: seems difficult to make work.

The point of random circuits is that they *don’t* have structure!
Adding structure to sampling problems

Idea: some property of samples that we can check?
Generically: seems difficult to make work.

The point of random circuits is that they don’t have structure!

IQP circuits [Shepherd and Bremner, ’08]:

- Hide a secret string s in the quantum circuit
- Set up circuit so it is biased to generate samples x with $x^T \cdot s = 0.$
Consider a matrix $P \in \{0, 1\}^{k \times n}$ and “action” $\theta$. 

Example: 

$$H = X_0 X_1 X_3 + X_1 X_2 X_4 X_5 + \cdots$$

Distribution of sampling result $X$: 

$$\text{Pr}[X = x] = \left| \langle x | e^{-iH \theta} | 0 \rangle \right|^2$$
Consider a matrix $P \in \{0, 1\}^{k \times n}$ and "action" $\theta$.

Let $H = \sum_i \prod_j X_j^{P_{ij}}$.

Example:

$$H = X_0X_1X_3 + X_1X_2X_4X_5 + \cdots$$  \hspace{1cm} (1)
Consider a matrix \( P \in \{0, 1\}^{k \times n} \) and “action” \( \theta \).

Let \( H = \sum_i \prod_j X_j^{P_{ij}} \).

Example:

\[
H = X_0X_1X_3 + X_1X_2X_4X_5 + \cdots
\]  

Distribution of sampling result \( X \):

\[
\Pr[X = x] = \left| \left\langle x \left| e^{-iH\theta} \right| 0 \right\rangle \right|^2
\]
Consider a matrix $P \in \{0, 1\}^{k \times n}$ and “action” $\theta$.

Let $H = \sum_i \prod_j X_j^{P_{ij}}$.

Example:

$$H = X_0X_1X_3 + X_1X_2X_4X_5 + \cdots \quad (1)$$

Distribution of sampling result $X$:

$$\text{Pr}[X = x] = \left| \left\langle x \left| e^{-iH\theta} \right| 0 \right\rangle \right|^2 \quad (2)$$

Bremner, Jozsa, Shepherd ’11: classically sampling worst-case IQP circuits would collapse polynomial heirarchy

Bremner, Montanaro, Shepherd ’16: average case is likely hard as well
Let $\theta = \pi/8$, and $s$ (secret) and $P$ have the form:

$$P = \begin{bmatrix} G \\ R \end{bmatrix}$$

$G^T$ is generator of Quadratic Residue code, $R$ random.
Let $\theta = \pi/8$, and $s$ (secret) and $P$ have the form:

$$
P = \begin{bmatrix} G \\ R \end{bmatrix} \quad \text{Ps} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}
$$

$G^T$ is generator of Quadratic Residue code, $R$ random.

$$
\Pr[X^T \cdot s = 0] = \mathbb{E}_x \left[ \cos^2 \left( \frac{\pi}{8} (1 - 2\text{wt}(Gx)) \right) \right]
$$
Let $\theta = \pi/8$, and $s$ (secret) and $P$ have the form:

$$P = \begin{bmatrix} G \\ R \end{bmatrix}, \quad Ps = \begin{bmatrix} 1 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

$G^\top$ is generator of Quadratic Residue code, $R$ random.

$$\Pr[X^\top \cdot s = 0] = \mathbb{E}_x \left[ \cos^2 \left( \frac{\pi}{8} (1 - 2 \text{wt}(Gx)) \right) \right]$$

QR code: codewords have $\text{wt}(c)$ mod 4 $\in \{0, -1\}$
Let $\theta = \pi/8$, and $s$ (secret) and $P$ have the form:

$$
P = \begin{bmatrix}
G \\
R
\end{bmatrix}
$$

$G^\top$ is generator of Quadratic Residue code, $R$ random.

The probability that $X^\top \cdot s = 0$ is:

$$
\Pr[X^\top \cdot s = 0] = \cos^2 \left(\frac{\pi}{8}\right) \approx 0.85
$$

QR code: codewords have $\text{wt}(c) \mod 4 \in \{0, -1\}$
IQP: Hiding $s$

Quantum: $\Pr[X^T \cdot s = 0] \approx 0.85$

Best classical: $\Pr[Y^T \cdot s = 0] = ?$

$P = \begin{bmatrix} G \\ R \end{bmatrix}$  \hspace{1cm} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$

$Ps = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$
IQP: Hiding $s$

Quantum: $\Pr[X^\top \cdot s = 0] \approx 0.85$
Best classical: $\Pr[Y^\top \cdot s = 0] = ?$

$$P = \begin{bmatrix} G \\ R \end{bmatrix} \quad Ps = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \text{permute rows, Gauss-Jordan columns} \quad P's' = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Scrambling preserves quantum success rate.
IQP: Hiding $s$

Quantum: $\Pr[X^\top \cdot s = 0] \approx 0.85$
Best classical: $\Pr[Y^\top \cdot s = 0] = ?$

$P = \begin{bmatrix} G \\ R \end{bmatrix} \quad Ps = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad P's' = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$

Scrambling preserves quantum success rate.

**Conjecture [SB ’08]:** Scrambling $P$ cryptographically hides $G$ (and equivalently $s$)
IQP: Classical strategy

Quantum: $\Pr[X^\top \cdot s = 0] \approx 0.85$

Best classical: $\Pr[Y^\top \cdot s = 0] \approx 0.5$

Assuming $s$ hidden, can classical do better than 0.5? Try to take advantage properties of embedded code.
IQP: Classical strategy

Quantum: $\Pr[X^T \cdot s = 0] \approx 0.85$
Best classical: $\Pr[Y^T \cdot s = 0] \approx 0.5$

Assuming $s$ hidden, can classical do better than 0.5? Try to take advantage properties of embedded code.

Consider choosing random $d \leftarrow \{0, 1\}^n$, and letting

$$y = \sum_{p \in \text{rows}(P)} p$$

$$p \cdot d = 1$$
IQP: Classical strategy

Quantum: $\Pr[X^T \cdot s = 0] \approx 0.85$

Best classical: $\Pr[Y^T \cdot s = 0] \approx 0.5$

Assuming $s$ hidden, can classical do better than 0.5? **Try to take advantage properties of embedded code.**

Consider choosing random $d \leftarrow \{0, 1\}^n$, and letting

$$y = \sum_{p \in \text{rows}(P)} p$$

Then:

$$y \cdot s = \sum_{p \in \text{rows}(P)} p \cdot s \pmod{2}$$
IQP: Classical strategy

Quantum: \( \Pr[X^T \cdot s = 0] \approx 0.85 \)
Best classical: \( \Pr[Y^T \cdot s = 0] \approx 0.5 \)

Assuming \( s \) hidden, can classical do better than 0.5? Try to take advantage properties of embedded code.

Consider choosing random \( d \$ \{0, 1\}^n \), and letting

\[
y = \sum_{p \in \text{rows}(P), p \cdot d = 1} p
\]

Then:

\[
y \cdot s = \sum_{p \in \text{rows}(P), p \cdot d = p \cdot s = 1} 1 \quad (\text{mod } 2)
\]
IQP: Classical strategy

Quantum: \( \Pr[X^\top \cdot s = 0] \approx 0.85 \)
Best classical: \( \Pr[Y^\top \cdot s = 0] \approx 0.5 \)

Assuming \( s \) hidden, can classical do better than 0.5? **Try to take advantage properties of embedded code.**

Consider choosing random \( d \leftarrow \{0, 1\}^n \), and letting

\[
y = \sum_{p \in \text{rows}(P) \atop p \cdot d=1} p
\]

Then:

\[
y \cdot s = \sum_{p \in \text{rows}(P) \atop p \cdot s=1} p \cdot d \pmod{2}
\]
IQP: Classical strategy

Quantum: $\Pr[X^T \cdot s = 0] \approx 0.85$
Best classical: $\Pr[Y^T \cdot s = 0] \approx 0.5$

Assuming $s$ hidden, can classical do better than 0.5? **Try to take advantage properties of embedded code.**

Consider choosing random $d \leftarrow \{0, 1\}^n$, and letting

$$y = \sum_{p \in \text{rows}(P) \atop p \cdot d = 1} p$$

Then:

$$y \cdot s = \text{wt}(Gd) \pmod{2}$$

QR code codewords are 50% even parity, 50% odd parity.
IQP: Classical strategy [SB ’08]

Quantum: $\Pr[X^\top \cdot s = 0] \approx 0.85$
Classical: $\Pr[Y^\top \cdot s = 0] \approx 0.5$

Consider choosing random $d, e \leftarrow \{0, 1\}^n$, and letting

$$y = \sum_{\substack{p \in \text{rows}(P) \\text{ s.t.} \\ p \cdot d = p \cdot e = 1}} p$$
IQP: Classical strategy [SB ’08]

Quantum: $\Pr[X^T \cdot s = 0] \approx 0.85$
Classical: $\Pr[Y^T \cdot s = 0] \approx 0.5$

Consider choosing random $d, e \leftarrow \{0, 1\}^n$, and letting

$$ y = \sum_{p \in \text{rows}(P)} p$$

Then:
IQP: Classical strategy [SB ’08]

Quantum: $\Pr[X^T \cdot s = 0] \approx 0.85$

Classical: $\Pr[Y^T \cdot s = 0] \approx 0.5$

Consider choosing random $d, e \xleftarrow{\$} \{0, 1\}^n$, and letting

$$y = \sum_{\substack{p \in \text{rows}(P) \\ \text{such that} \\ p \cdot d = p \cdot e = 1}} p$$

Then:

$$y \cdot s = \sum_{\substack{p \in \text{rows}(P) \\ \text{such that} \\ p \cdot d = p \cdot e = 1}} p \cdot s \pmod{2}$$
IQP: Classical strategy [SB ’08]

Quantum: $\Pr[X^T \cdot s = 0] \approx 0.85$
Classical: $\Pr[Y^T \cdot s = 0] \approx 0.5$

Consider choosing random $d, e \xleftarrow{\$} \{0, 1\}^n$, and letting

$$y = \sum_{p \in \text{rows}(P) \atop p \cdot d = p \cdot e = 1} p$$

Then:

$$y \cdot s = \sum_{p \in \text{rows}(P) \atop p \cdot s = 1} (p \cdot d)(p \cdot e) \pmod{2}$$

Fact: $(G_d \cdot G_e) = 1$ iff $G_d$ and $G_e$ both have odd parity.
IQP: Classical strategy [SB ’08]

Quantum: \( \Pr[X^\top \cdot s = 0] \approx 0.85 \)

Classical: \( \Pr[Y^\top \cdot s = 0] \equiv 0.5 \)

Consider choosing random \( d, e \leftarrow \{0, 1\}^n \), and letting

\[
y = \sum_{p \in \text{rows}(P)} p \quad \text{such that } p \cdot d = p \cdot e = 1
\]

Then:

\[
y \cdot s = (Gd) \cdot (Ge) \pmod{2}
\]

Fact: \( (Gd) \cdot (Ge) = 1 \) iff \( Gd, Ge \) both have odd parity.
IQP: Classical strategy [SB '08]

Quantum: $\Pr[X^T \cdot s = 0] \approx 0.85$
Classical: $\Pr[Y^T \cdot s = 0] = 0.75$

Consider choosing random $d, e \leftarrow \{0, 1\}^n$, and letting

$$y = \sum_{p \in \text{rows}(P) \atop p \cdot d = p \cdot e = 1} p$$

Then:

$$y \cdot s = (Gd) \cdot (Ge) \pmod{2}$$

Fact: $(Gd) \cdot (Ge) = 1$ iff $Gd$, $Ge$ both have odd parity.

Thus $y \cdot s = 0$ with probability $3/4$!
Key: Correlate samples to attack the key $s$. 

Let $y_i$ form rows of a matrix $M$, such that $M \cdot s = 0$. Can solve for $s$ if $M$ has high rank. Empirically it does!
IQP: Can we do better classically? [GDKM ’19 arXiv:1912.05547]

Key: Correlate samples to attack the key $s$

Consider choosing one random $d \xleftarrow{\$} \{0, 1\}^n$, held constant over many different $e_i \xleftarrow{\$} \{0, 1\}^n$

$$y_i = \sum_{p \in \text{rows}(P)} p$$

$$y_i \cdot s = 1 \text{ iff } Gd, Ge_i \text{ both have odd parity.}$$
Key: Correlate samples to attack the key $s$

Consider choosing one random $d \overset{\$}{\leftarrow} \{0,1\}^n$, held constant over many different $e_i \overset{\$}{\leftarrow} \{0,1\}^n$

$$y_i = \sum_{p \in \text{rows}(P)} p$$

$$p \cdot d = p \cdot e_i = 1$$

$y_i \cdot s = 1$ iff $Gd, Ge_i$ both have odd parity.

$Gd$ has even parity $\Rightarrow$ all $y_i \cdot s = 0$
Key: Correlate samples to attack the key $s$

Consider choosing one random $d \leftarrow \{0, 1\}^n$, held constant over many different $e_i \leftarrow \{0, 1\}^n$

$$y_i = \sum_{p \in \text{rows}(P)} p$$

$$y_i \cdot s = 1 \text{ iff } Gd, Ge_i \text{ both have odd parity.}$$

$Gd$ has even parity $\Rightarrow$ all $y_i \cdot s = 0$

Let $y_i$ form rows of a matrix $M$, such that $Ms = 0$
IQP: Can we do better classically? [GDKM ’19 arXiv:1912.05547]

Key: Correlate samples to attack the key $s$

Consider choosing one random $d \leftarrow \{0, 1\}^n$, held constant over many different $e_i \leftarrow \{0, 1\}^n$

$$y_i = \sum_{p \in \text{rows}(P)} p$$

$$y_i \cdot s = 1 \text{ iff } Gd, Ge_i \text{ both have odd parity.}$$

$Gd$ has even parity $\Rightarrow$ all $y_i \cdot s = 0$

Let $y_i$ form rows of a matrix $M$, such that $Ms = 0$

Can solve for $s$! ... If $M$ has high rank.
Key: Correlate samples to attack the key $s$

Consider choosing one random $d \leftarrow \{0, 1\}^n$, held constant over many different $e_i \leftarrow \{0, 1\}^n$

$$y_i = \sum_{p \in \text{rows}(P)} p \quad \text{subject to} \quad p \cdot d = p \cdot e_i = 1$$

$y_i \cdot s = 1$ iff $Gd$, $Ge_i$ both have odd parity.

$Gd$ has even parity $\Rightarrow$ all $y_i \cdot s = 0$

Let $y_i$ form rows of a matrix $M$, such that $Ms = 0$

Can solve for $s$! ... If $M$ has high rank. Empirically it does!
IQP: can it be fixed?

• Attack relies on properties of QR code
IQP: can it be fixed?

- Attack relies on properties of QR code
- Could pick a different $G$ for which this attack would not succeed?
IQP: can it be fixed?

- Attack relies on properties of QR code
- Could pick a different $G$ for which this attack would not succeed?
- Ultimately, would like to rely on standard cryptographic assumptions...
NISQ verifiable quantum advantage

Sampling problems
- e.g. random circuits, Boson sampling, ...
  - ✓ NISQ feasible
  - ✗ Efficiently verifiable

Number theory problems
- e.g. factoring, discrete logarithm, ...
  - ✗ NISQ feasible
  - ✓ Efficiently verifiable

add structure

make less costly

???
- ✓ NISQ feasible
- ✓ Efficiently verifiable
Interactive proofs of quantumness

Round 1: Prover commits to a specific quantum state

Round 2+: Verifier asks for measurement in specific basis

Can be extended to verify arbitrary quantum computations! (arXiv:1804.01082)
Interactive proofs of quantumness

Round 1: Prover commits to a specific quantum state

Round 2+: Verifier asks for measurement in specific basis

By randomizing choice of basis and repeating interaction, can ensure prover would respond correctly in any basis


Can be extended to verify arbitrary quantum computations! (arXiv:1804.01082)
State commitment (round 1): trapdoor claw-free functions

How does the prover commit to a state?

Consider a 2-to-1 collision-resistant (claw-free) function $f$. 

Evaluate $f$ on uniform superposition $f \leftarrow \sum_{x} |x\rangle |f(x)\rangle$

Measure 2nd register as $y$

Store $y$ as commitment

Prover has committed to the state $(|x_0\rangle + |x_1\rangle) |y\rangle$
How does the prover commit to a state?

Consider a 2-to-1 collision-resistant (claw-free) function $f$.

Evaluate $f$ on uniform superposition
$$\sum_x |x\rangle |f(x)\rangle$$

Measure 2\textsuperscript{nd} register as $y$

Pick 2-to-1 function $f$

Store $y$ as commitment

Prover has committed to the state $(|x_0\rangle + |x_1\rangle) |y\rangle$
LWE protocol

Prover

Evaluate $f$ on uniform superposition: $\sum_x |x\rangle |f(x)\rangle$

Measure 2\textsuperscript{nd} register as $y$

Verifier

Pick trapdoor claw-free function $f$

Compute $x_0, x_1$ from $y$ using trapdoor

Brakerski, Christiano, Mahadev, Vidick, Vazirani ’18 (arXiv:1804.00640)
LWE protocol

**Prover**

Evaluate $f$ on uniform superposition: $\sum_x |x\rangle |f(x)\rangle$

Measure 2\textsuperscript{nd} register as $y$

Measure qubits of $|x_0\rangle + |x_1\rangle$ in given basis

**Verifier**

Pick trapdoor claw-free function $f$

Compute $x_0, x_1$ from $y$ using trapdoor

Pick standard or Hadamard basis

Validate result against $x_0, x_1$

---

Brakerski, Christiano, Mahadev, Vidick, Vazirani ’18 (arXiv:1804.00640)
LWE protocol

Prover

Evaluate $f$ on uniform superposition: $\sum_x |x\rangle |f(x)\rangle$
Measure 2\textsuperscript{nd} register as $y$

Measure qubits of $|x_0\rangle + |x_1\rangle$ in given basis

Verifier

Pick trapdoor claw-free function $f$

Compute $x_0, x_1$ from $y$ using trapdoor

Pick standard or Hadamard basis

Validate result against $x_0, x_1$

Subtlety: claw-free does not imply hardness of generating measurement outcomes!

Brakerski, Christiano, Mahadev, Vidick, Vazirani ’18 (arXiv:1804.00640)
LWE protocol

**Prover**
- Evaluate $f$ on uniform superposition: $\sum_x |x\rangle |f(x)\rangle$
- Measure 2$^{nd}$ register as $y$
- Measure qubits of $|x_0\rangle + |x_1\rangle$ in given basis

**Verifier**
- Pick trapdoor claw-free function $f$
- Compute $x_0, x_1$ from $y$ using trapdoor
- Pick standard or Hadamard basis
- Validate result against $x_0, x_1$

Subtlety: claw-free does *not* imply hardness of generating measurement outcomes!
Learning-with-Errors TCF has *adaptive hardcore bit*

Brakerski, Christiano, Mahadev, Vidick, Vazirani ’18 (arXiv:1804.00640)
## Trapdoor claw-free functions

<table>
<thead>
<tr>
<th>TCF</th>
<th>Trapdoor</th>
<th>Claw-free</th>
<th>Adaptive hard-core bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>LWE [1]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>$x^2 \mod N$ [3]</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Ring-LWE [2]</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Diffie-Hellman [3]</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
</tbody>
</table>


Trapdoor claw-free functions

<table>
<thead>
<tr>
<th>TCF</th>
<th>Trapdoor</th>
<th>Claw-free</th>
<th>Adaptive hard-core bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>LWE [1]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>$x^2 \mod N$ [3]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Ring-LWE [2]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Diffie-Hellman [3]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

BKVV ’20 [2]: Non-interactive protocol without adaptive hardcore bit, in random oracle model

Trapdoor claw-free functions

<table>
<thead>
<tr>
<th>TCF</th>
<th>Trapdoor</th>
<th>Claw-free</th>
<th>Adaptive hard-core bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>LWE [1]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>$x^2 \mod N$ [3]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Ring-LWE [2]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Diffie-Hellman [3]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

BKVV ’20 [2]: Non-interactive protocol without adaptive hardcore bit, in random oracle model

Can we remove AHCB in the standard model of cryptography?

LWE protocol

Evaluate $f$ on uniform superposition: $\sum_x |x\rangle |f(x)\rangle$

Measure 2\textsuperscript{nd} register as $y$

Measure qubits of $|x_0\rangle + |x_1\rangle$ in given basis

\[ f \]

\[ y \]

\[ \text{basis} \]

\[ \text{result} \]

Pick trapdoor claw-free function $f$

Compute $x_0, x_1$ from $y$ using trapdoor

Pick standard or Hadamard basis

Validate result against $x_0, x_1$

Replace Hadamard basis measurement with “1-player CHSH”

Brakerski, Christiano, Mahadev, Vidick, Vazirani ’18 (arXiv:1804.00640)
Interactive measurement: computational Bell test

Replace Hadamard basis measurement with two-step process: “condense” $x_0, x_1$ into a single qubit, and then do a “Bell test.”

\[
|0⟩\begin{cases} |0⟩ & |0⟩ |x_0 · r⟩ + |1⟩ |x_1 · r⟩ \end{cases} \quad \text{Measure all but ancilla in Hadamard basis}
\]

Pick random bitstring $r$

GDKM, Choi, Vazirani, Yao ’21 (arXiv:2104.00687)
Interactive measurement: computational Bell test

Replace Hadamard basis measurement with two-step process: “condense” $x_0, x_1$ into a single qubit, and then do a “Bell test.”

Now single-qubit state: $|0\rangle$ or $|1\rangle$ if $x_0 \cdot r = x_1 \cdot r$, otherwise $|+\rangle$ or $|-\rangle$. 

GDKM, Choi, Vazirani, Yao '21 (arXiv:2104.00687)
Interactive measurement: computational Bell test

Replace Hadamard basis measurement with two-step process: “condense” $x_0, x_1$ into a single qubit, and then do a “Bell test.”

$|\psi\rangle$ $\quad$ $\quad$ $10100111100$

$\vdots$ $\quad$ $\vdots$

$|x_0\rangle |x_0 \cdot r\rangle + |x_1\rangle |x_1 \cdot r\rangle$ $\quad$ $\quad$ $11010110011$

Measure all but ancilla in Hadamard basis $\quad$ $\quad$ $11101100100$

$\quad$ $\quad$ $10011000011$

$\quad$ $\quad$ $\vdots$

Pick random bitstring $r$

Now single-qubit state: $|0\rangle$ or $|1\rangle$ if $x_0 \cdot r = x_1 \cdot r$, otherwise $|+\rangle$ or $|-\rangle$.

Polarization hidden via:

Cryptographic secret (here) $\leftrightarrow$ Non-communication (Bell test)

GDKM, Choi, Vazirani, Yao ’21 (arXiv:2104.00687)
Interactive measurement: computational Bell test

Replace Hadamard basis measurement with two-step process: “condense” $x_0, x_1$ into a single qubit, and then do a “Bell test.”

\[ |x_0 \rangle |x_0 \cdot r \rangle + |x_1 \rangle |x_1 \cdot r \rangle \]

Measure all but ancilla in Hadamard basis

\[ r \]

Pick random bitstring $r$

\[ d \]

Pick $(Z + X)$ or $(Z - X)$ basis

\[ \text{basis} \]

Measure qubit in basis

\[ \text{result} \]

Validate against $r, x_0, x_1, d$

GDKM, Choi, Vazirani, Yao ’21 (arXiv:2104.00687)
Computational Bell test: classical bound

Run protocol many times, collect statistics.

\( p_s \): Success rate for standard basis measurement.

\( p_{\text{CHSH}} \): Success rate when performing CHSH-type measurement.

Under assumption of claw-free function:

Classical bound:

\[
    p_s + 4p_{\text{CHSH}} - 4 < \text{negl}(n)
\]

Ideal quantum:

\[
    p_s = 1, \quad p_{\text{CHSH}} = \cos^2\left(\frac{\pi}{8}\right)
\]

\[
    p_s + 4p_{\text{CHSH}} - 4 = \sqrt{2} - 1 \approx 0.414
\]

Note: Let \( p_s = 1 \). Then for \( p_{\text{CHSH}} \):

Classical bound \( 75\% \), ideal quantum \( \sim 85\% \). Same as regular CHSH!

GDKM, Choi, Vazirani, Yao '21 (arXiv:2104.00687)
Computational Bell test: classical bound

Run protocol many times, collect statistics.

\( p_s \): Success rate for standard basis measurement.

\( p_{\text{CHSH}} \): Success rate when performing CHSH-type measurement.

Under assumption of claw-free function:

**Classical bound:**  
\[ p_s + 4p_{\text{CHSH}} - 4 < \text{negl}(n) \]
Computational Bell test: classical bound

Run protocol many times, collect statistics.

\( p_s \): Success rate for standard basis measurement.

\( p_{\text{CHSH}} \): Success rate when performing CHSH-type measurement.

Under assumption of claw-free function:

\[
\text{Classical bound: } p_s + 4p_{\text{CHSH}} - 4 < \text{negl}(n)
\]

\[
\text{Ideal quantum: } p_s = 1, p_{\text{CHSH}} = \cos^2(\pi/8)
\]
Computational Bell test: classical bound

Run protocol many times, collect statistics.

- $p_s$: Success rate for standard basis measurement.
- $p_{\text{CHSH}}$: Success rate when performing CHSH-type measurement.

Under assumption of claw-free function:

**Classical bound:** $p_s + 4p_{\text{CHSH}} - 4 < \text{negl}(n)$

**Ideal quantum:** $p_s = 1, p_{\text{CHSH}} = \cos^2(\pi/8)$

\[ p_s + 4p_{\text{CHSH}} - 4 = \sqrt{2} - 1 \approx 0.414 \]
Computational Bell test: classical bound

Run protocol many times, collect statistics.

$p_s$: Success rate for standard basis measurement.

$p_{\text{CHSH}}$: Success rate when performing CHSH-type measurement.

Under assumption of claw-free function:

\[
\text{Classical bound: } p_s + 4p_{\text{CHSH}} - 4 < \text{negl}(n)
\]

\[
\text{Ideal quantum: } p_s = 1, p_{\text{CHSH}} = \cos^2(\pi/8)
\]

\[
p_s + 4p_{\text{CHSH}} - 4 = \sqrt{2} - 1 \approx 0.414
\]

\[
\text{Note: Let } p_s = 1. \text{ Then for } p_{\text{CHSH}}:
\]

Classical bound 75\%, ideal quantum \( \sim \) 85\%. Same as regular CHSH!

GDKM, Choi, Vazirani, Yao ’21 (arXiv:2104.00687)
Challenges for implementation

- Partial measurement
Challenges for implementation

• Partial measurement
  • Required for multi-round classical interaction
Challenges for implementation

- Partial measurement
  - Required for multi-round classical interaction
- Fidelity requirement
Challenges for implementation

- Partial measurement
  - Required for multi-round classical interaction
- Fidelity requirement
  - High fidelity needed to pass classical bound
Challenges for implementation

- Partial measurement
  - Required for multi-round classical interaction
- Fidelity requirement
  - High fidelity needed to pass classical bound
- Circuit sizes
Challenges for implementation

- Partial measurement
  - Required for multi-round classical interaction
- Fidelity requirement
  - High fidelity needed to pass classical bound
- Circuit sizes
  - Need to implement public-key crypto. on a superposition
Partial measurements in the lab

Trapped Ion Quantum Information lab at U. Maryland

Working on demonstration of protocols in trapped ions!

Prof. Christopher Monroe
Dr. Daiwei Zhu
Dr. Crystal Noel

and others!
Partial measurements in the lab

Trapped Ion Quantum Information lab at U. Maryland

Working on demonstration of protocols in trapped ions!

Partial measurement:
Partial measurements in the lab

Trapped Ion Quantum Information lab at U. Maryland

Working on demonstration of protocols in trapped ions!

Partial measurement:
Partial measurements in the lab

Trapped Ion Quantum Information lab at U. Maryland

Working on demonstration of protocols in trapped ions!

Partial measurement:
Partial measurements in the lab

Trapped Ion Quantum Information lab at U. Maryland

Working on demonstration of protocols in trapped ions!

Partial measurement:
Partial measurements in the lab

Trapped Ion Quantum Information lab at U. Maryland

Working on demonstration of protocols in trapped ions!

Partial measurement:
Partial measurements in the lab

Trapped Ion Quantum Information lab at U. Maryland

Working on demonstration of protocols in trapped ions!

Partial measurement:
Partial measurements in the lab

Trapped Ion Quantum Information lab at U. Maryland

Working on demonstration of protocols in trapped ions!

Partial measurement:
Partial measurements in the lab

Trapped Ion Quantum Information lab at U. Maryland

Working on demonstration of protocols in trapped ions!

Partial measurement:
Trapped Ion Quantum Information lab at U. Maryland

Working on demonstration of protocols in trapped ions!

Partial measurement:
Partial measurements in the lab

Trapped Ion Quantum Information lab at U. Maryland

Working on demonstration of protocols in trapped ions!

Partial measurement:
Partial measurements in the lab

Trapped Ion Quantum Information lab at U. Maryland

Working on demonstration of protocols in trapped ions!

Partial measurement:
How to deal with high fidelity requirement? Need \( \sim \) 83% fidelity in general to pass.
How to deal with high fidelity requirement? Need \(~ 83\%\) fidelity in general to pass.

Can show: a prover holding \((|x_0\rangle + |x_1\rangle) |y\rangle\) with \(\epsilon\) phase coherence passes!
How to deal with high fidelity requirement? Need \( \sim 83\% \) fidelity in general to pass.

Can show: a prover holding \((|x_0\rangle + |x_1\rangle) |y\rangle\) with \(\epsilon\) phase coherence passes!

When we generate \(\sum_x |x\rangle |f(x)\rangle\), add redundancy to \(f(x)\), for bit flip error detection!
Technique: postselection

How to deal with high fidelity requirement? Need $\sim 83\%$ fidelity in general to pass.

Numerical results for $x^2 \mod N$ with $\log N = 512$ bits.
Here: make transformation $x^2 \mod N \Rightarrow (kx)^2 \mod k^2 N$
Most demanding step in all these protocols: evaluating TCF

\[ U_f |x\rangle |0^\otimes n\rangle = |x\rangle |f(x)\rangle \]
Improving circuit sizes

Most demanding step in all these protocols: evaluating TCF

\[ U_f |x\rangle |0^\otimes n\rangle = |x\rangle |f(x)\rangle \]

Getting rid of adaptive hardcore bit helps!

\( x^2 \mod N \) and \textbf{Ring-LWE} have classical circuits as fast as \( O(n \log n) \)...
Most demanding step in all these protocols: evaluating TCF

\[ \mathcal{U}_f |x\rangle |0^\otimes n\rangle = |x\rangle |f(x)\rangle \]

Getting rid of adaptive hardcore bit helps!

\( x^2 \mod N \) and **Ring-LWE** have classical circuits as fast as \( \mathcal{O}(n \log n) \)... but they are recursive and hard to make reversible.
Improving circuit sizes

Most demanding step in all these protocols: evaluating TCF

\[ U_f |x⟩ |0^n⟩ = |x⟩ |f(x)⟩ \]

Getting rid of adaptive hardcore bit helps!

\( x^2 \mod N \) and **Ring-LWE** have classical circuits as fast as \( O(n \log n) \)...

but they are recursive and hard to make reversible.

Protocol allows us to make circuits irreversible!
Technique: taking out the garbage

**Goal:** $U_f |x\rangle |0^\otimes n\rangle = |x\rangle |f(x)\rangle$

When converting classical circuits to quantum:

**Garbage bits:** extra entangled outputs due to unitarity

Classical AND

Quantum AND (Toffoli)
Technique: taking out the garbage

**Goal:** $U_f |x\rangle |0^\otimes n\rangle = |x\rangle |f(x)\rangle$

When converting classical circuits to quantum:

**Garbage bits:** extra entangled outputs due to unitarity

Let $U'_f$ be a unitary generating garbage bits $g_f(x)$:

- $|x\rangle \equiv |x\rangle$
- $|0\rangle \equiv U'_f |g_f(x)\rangle$
- $|0\rangle \equiv |f(x)\rangle$
Technique: taking out the garbage

**Goal:** $U_f |x\rangle |0^n\rangle = |x\rangle |f(x)\rangle$

When converting classical circuits to quantum:

**Garbage bits:** extra entangled outputs due to unitarity

Let $U'_f$ be a unitary generating garbage bits $g_f(x)$:

\[
\begin{align*}
|x\rangle & \quad \Rightarrow \quad |x\rangle \\
|0\rangle & \quad \Rightarrow \quad |g_f(x)\rangle \\
|0\rangle & \quad \Rightarrow \quad |f(x)\rangle
\end{align*}
\]

\[
\begin{align*}
|x\rangle & \quad \Rightarrow \quad |x\rangle \\
|0\rangle & \quad \Rightarrow \quad |0\rangle \\
|0\rangle & \quad \Rightarrow \quad |f(x)\rangle
\end{align*}
\]

Let $U'_f$ be a unitary generating garbage bits $g_f(x)$:
Technique: taking out the garbage

**Goal:** $U_f |x\rangle |0^\otimes n\rangle = |x\rangle |f(x)\rangle$

When converting classical circuits to quantum:

**Garbage bits:** extra entangled outputs due to unitarity

Let $U'_f$ be a unitary generating garbage bits $g_f(x)$:

Lots of time and space overhead!
Technique: taking out the garbage

Goal: $U_f |x\rangle |0^\otimes n\rangle = |x\rangle |f(x)\rangle$

When converting classical circuits to quantum:

**Garbage bits:** extra entangled outputs due to unitarity

Let $U'_f$ be a unitary generating garbage bits $g_f(x)$:

Can we “measure them away” instead?
Technique: taking out the garbage

Measure garbage bits $g_f(x)$ in Hadamard basis, get some string $h$. End up with state:

$$\sum_x (-1)^{h \cdot g_f(x)} |x\rangle |f(x)\rangle$$
Technique: taking out the garbage

Measure garbage bits $g_f(x)$ in Hadamard basis, get some string $h$. End up with state:

$$\sum_x (-1)^{h \cdot g_f(x)} |x\rangle |f(x)\rangle$$

In general useless: unique phase $(-1)^{h \cdot g_f(x)}$ on every term.
Technique: taking out the garbage

Measure garbage bits $g_f(x)$ in Hadamard basis, get some string $h$. End up with state:

$$\sum_x (-1)^{h \cdot g_f(x)} |x\rangle |f(x)\rangle$$

In general useless: unique phase $(-1)^{h \cdot g_f(x)}$ on every term. But after collapsing onto a single output:

$$\left[ (-1)^{h \cdot g_f(x_0)} |x_0\rangle + (-1)^{h \cdot g_f(x_1)} |x_1\rangle \right] |y\rangle$$

Verifier can efficiently compute $g_f(x)$ for these two terms!

Can directly convert classical circuits to quantum! 1024-bit $x^2 \mod N$ costs only $10^6$ Toffoli gates.
Technique: taking out the garbage

Measure garbage bits $g_f(x)$ in Hadamard basis, get some string $h$. End up with state:

$$\sum_x (-1)^{h \cdot g_f(x)} |x\rangle |f(x)\rangle$$

In general useless: unique phase $(-1)^{h \cdot g_f(x)}$ on every term. But after collapsing onto a single output:

$$[(-1)^{h \cdot g_f(x_0)} |x_0\rangle + (-1)^{h \cdot g_f(x_1)} |x_1\rangle] |y\rangle$$

Verifier can efficiently compute $g_f(\cdot)$ for these two terms!
Technique: taking out the garbage

Measure garbage bits $g_f(x)$ in Hadamard basis, get some string $h$. End up with state:

$$
\sum_x (-1)^{h \cdot g_f(x)} |x\rangle |f(x)\rangle
$$

In general useless: unique phase $(-1)^{h \cdot g_f(x)}$ on every term. But after collapsing onto a single output:

$$
[(-1)^{h \cdot g_f(x_0)} |x_0\rangle + (-1)^{h \cdot g_f(x_1)} |x_1\rangle] |y\rangle
$$

Verifier can efficiently compute $g_f(\cdot)$ for these two terms!

Can directly convert classical circuits to quantum!
Technique: taking out the garbage

Measure garbage bits $g_f(x)$ in Hadamard basis, get some string $h$. End up with state:

$$\sum_x (-1)^{h \cdot g_f(x)} |x\rangle |f(x)\rangle$$

In general useless: unique phase $(-1)^{h \cdot g_f(x)}$ on every term. But after collapsing onto a single output:

$$[(-1)^{h \cdot g_f(x_0)} |x_0\rangle + (-1)^{h \cdot g_f(x_1)} |x_1\rangle] |y\rangle$$

Verifier can efficiently compute $g_f(\cdot)$ for these two terms!

Can directly convert classical circuits to quantum!

1024-bit $x^2 \mod N$ costs only $10^6$ Toffoli gates.
Paths forward

Bottleneck: Evaluating TCF on quantum superposition
Paths forward

**Bottleneck:** Evaluating TCF on quantum superposition

“In the box” ideas (not necessarily bad):

- Find more efficient TCFs
Bottleneck: Evaluating TCF on quantum superposition

“In the box” ideas (not necessarily bad):

- Find more efficient TCFs
- Better quantum circuits for TCFs
Paths forward

**Bottleneck:** Evaluating TCF on quantum superposition

**“In the box” ideas** (not necessarily bad):

- Find more efficient TCFs
- Better quantum circuits for TCFs
- ... public-key cryptography is just slow
Paths forward

**Bottleneck:** Evaluating TCF on quantum superposition

“In the box” ideas (not necessarily bad):

- Find more efficient TCFs
- Better quantum circuits for TCFs
- ... public-key cryptography is just slow
Paths forward

**Bottleneck:** Evaluating TCF on quantum superposition

**“In the box” ideas** (not necessarily bad):

- Find more efficient TCFs
- Better quantum circuits for TCFs
- ... public-key cryptography is just slow

**“Box-adjacent” ideas:**

- Explore other protocols (fix IQP and make it fast?)
Paths forward

**Bottleneck:** Evaluating TCF on quantum superposition

**“In the box” ideas** (not necessarily bad):

- Find more efficient TCFs
- Better quantum circuits for TCFs
- ... public-key cryptography is just slow

**“Box-adjacent” ideas:**

- Explore other protocols (fix IQP and make it fast?)
- Remove trapdoor—hash-based cryptography?
Bottleneck: Evaluating TCF on quantum superposition

“In the box” ideas (not necessarily bad):

- Find more efficient TCFs
- Better quantum circuits for TCFs
- ... public-key cryptography is just slow

“Box-adjacent” ideas:

- Explore other protocols (fix IQP and make it fast?)
- Remove trapdoor—hash-based cryptography?
Paths forward

**Bottleneck:** Evaluating TCF on quantum superposition

“In the box” ideas (not necessarily bad):

- Find more efficient TCFs
- Better quantum circuits for TCFs
- ... public-key cryptography is just slow

“Box-adjacent” ideas:

- Explore other protocols (fix IQP and make it fast?)
- Remove trapdoor—hash-based cryptography?

Way outside the box?
Backup!
## TCF constructions

<table>
<thead>
<tr>
<th>TCF</th>
<th>A.H.C.B.</th>
<th>Gate count</th>
<th>n for hardness</th>
</tr>
</thead>
<tbody>
<tr>
<td>LWE [1]</td>
<td>✓</td>
<td>$O(n^2 \log^2 n)$</td>
<td>$10^4$</td>
</tr>
<tr>
<td>Ring-LWE [2]</td>
<td>✗</td>
<td>$O(n \log^2 n)$</td>
<td>$10^3$</td>
</tr>
<tr>
<td>$x^2 \mod N$ [3]</td>
<td>✗</td>
<td>$O(n \log n)$</td>
<td>$10^3$</td>
</tr>
<tr>
<td>DDH [3]</td>
<td>✗</td>
<td>$O(n^3 \log^2 n)$</td>
<td>$10^2$</td>
</tr>
</tbody>
</table>

A.H.C.B. = "adaptive hard core bit"

---

**TCF constructions**

<table>
<thead>
<tr>
<th>TCF</th>
<th>A.H.C.B.</th>
<th>Gate count</th>
<th>$n$ for hardness</th>
</tr>
</thead>
<tbody>
<tr>
<td>LWE [1]</td>
<td>✓</td>
<td>$O(n^2 \log^2 n)$</td>
<td>$10^4$</td>
</tr>
<tr>
<td>Ring-LWE [2]</td>
<td>×</td>
<td>$O(n \log^2 n)$</td>
<td>$10^3$</td>
</tr>
<tr>
<td>$x^2 \mod N$ [3]</td>
<td>×</td>
<td>$O(n \log n)$</td>
<td>$10^3$</td>
</tr>
<tr>
<td>DDH [3]</td>
<td>×</td>
<td>$O(n^3 \log^2 n)$</td>
<td>$10^2$</td>
</tr>
</tbody>
</table>

A.H.C.B. = "adaptive hard core bit"

**Remarks:**

- Removing adaptive hardcore bit requirement helps!

---

## TCF constructions

<table>
<thead>
<tr>
<th>TCF</th>
<th>A.H.C.B.</th>
<th>Gate count</th>
<th>$n$ for hardness</th>
</tr>
</thead>
<tbody>
<tr>
<td>LWE [1]</td>
<td>✓</td>
<td>$O(n^2 \log^2 n)$</td>
<td>$10^4$</td>
</tr>
<tr>
<td>Ring-LWE [2]</td>
<td>x</td>
<td>$O(n \log^2 n)$</td>
<td>$10^3$</td>
</tr>
<tr>
<td>$x^2 \mod N$ [3]</td>
<td>x</td>
<td>$O(n \log n)$</td>
<td>$10^3$</td>
</tr>
<tr>
<td>DDH [3]</td>
<td>x</td>
<td>$O(n^3 \log^2 n)$</td>
<td>$10^2$</td>
</tr>
</tbody>
</table>

A.H.C.B. = “adaptive hard core bit”

**Remarks:**

- Removing adaptive hardcore bit requirement helps!
- Can’t just plug in $n$—constant factors

$y = x^2 \mod N$ with $N = pq$

Each $y$ has 4 roots $(x_0, x_1, -x_0, -x_1)$. 
$x^2 \mod N$

$y = x^2 \mod N$ with $N = pq$

Each $y$ has 4 roots $(x_0, x_1, -x_0, -x_1)$. Set domain to $[0, N/2]$ to make it 2-to-1
Each $y$ has 4 roots $(x_0, x_1, -x_0, -x_1)$. Set domain to $[0, N/2]$ to make it 2-to-1

- Finding a claw as hard as factoring $N$
$y = x^2 \mod N$ with $N = pq$

Each $y$ has 4 roots $(x_0, x_1, -x_0, -x_1)$. Set domain to $[0, N/2]$ to make it 2-to-1

- Finding a claw as hard as factoring $N$
- Features:
  - Simple to implement, asymptotically fast algorithms
  - Classical hardness in practice extremely well studied
$y = x^2 \mod N$ with $N = pq$

Each $y$ has 4 roots $(x_0, x_1, -x_0, -x_1)$. Set domain to $[0, N/2]$ to make it 2-to-1

- Finding a claw as hard as factoring $N$
- Features:
  - Simple to implement, asymptotically fast algorithms
  - Classical hardness in practice extremely well studied
- $O(n \log n \log \log n)$ Schonhage-Strassen multiplication seems out of reach, but
$y = x^2 \mod N$ with $N = pq$

Each $y$ has 4 roots $(x_0, x_1, -x_0, -x_1)$. Set domain to $[0, N/2]$ to make it 2-to-1.

- Finding a claw as hard as factoring $N$
- Features:
  - Simple to implement, asymptotically fast algorithms
  - Classical hardness in practice extremely well studied
- $O(n \log n \log \log n)$ Schonhage-Strassen multiplication seems out of reach, but
- $O(n^{1.58})$ Karatsuba mult. beats naive $O(n^2)$ alg. at $n \sim 100$
  (much earlier than in the classical case!)
$x^2 \mod N$

\[ y = x^2 \mod N \text{ with } N = pq \]

Each $y$ has 4 roots ($x_0, x_1, -x_0, -x_1$). Set domain to $[0, N/2]$ to make it 2-to-1

- Finding a claw as hard as factoring $N$
- Features:
  - Simple to implement, asymptotically fast algorithms
  - Classical hardness in practice extremely well studied
- $O(n \log n \log \log n)$ Schonnage-Strassen multiplication seems out of reach, but
- $O(n^{1.58})$ Karatsuba mult. beats naive $O(n^2)$ alg. at $n \sim 100$
  (much earlier than in the classical case!)

Q. advantage in $10^6$ Toffoli gates
Trapdoor from Decisional Diffie-Hellman (DDH)

Trapdoor functions from DDH [1, 2]: linear algebra in the exponent
Trapdoor functions from DDH [1, 2]: linear algebra in the exponent

Gen($1^\lambda$)

1. Choose group $\mathbb{G}$ of order $q \sim \mathcal{O}(2^\lambda)$, and generator $g$


Trapdoor functions from DDH [1, 2]: linear algebra in the exponent

Gen(1^λ)

1. Choose group $\mathbb{G}$ of order $q \sim \mathcal{O}(2^λ)$, and generator $g$
2. Choose random invertible $M \in \mathbb{Z}_q^{k \times k}$ for $k > \log q$

Trapdoor from Decisional Diffie-Hellman (DDH)

Trapdoor functions from DDH [1, 2]: **linear algebra in the exponent**

**Gen(1^λ)**

1. Choose group $G$ of order $q \sim \mathcal{O}(2^\lambda)$, and generator $g$
2. Choose random invertible $M \in \mathbb{Z}_q^{k \times k}$ for $k > \log q$
3. Compute $g^M = (g^{M_{ij}}) \in G^{k \times k}$


Trapdoor from Decisional Diffie-Hellman (DDH)

Trapdoor functions from DDH [1, 2]: linear algebra in the exponent

Gen(1\(^\lambda\))

1. Choose group \( G \) of order \( q \sim \mathcal{O}(2^\lambda) \), and generator \( g \)
2. Choose random invertible \( M \in \mathbb{Z}_q^{k \times k} \) for \( k > \log q \)
3. Compute \( g^M = (g^{M_{ij}}) \in G^{k \times k} \)
4. Return \( pk = (g^M), sk = (g, M) \)


Trapdoor from Decisional Diffie-Hellman (DDH)

Trapdoor functions from DDH [1, 2]: linear algebra in the exponent

\[ pk = (g^M), \; sk = (g, M). \text{ On input } x \in \{0, 1\}^k: \]

\[ f(x) = g^{Mx}, \text{ Inversion: } f^{-1}(f(x), M) = g^{M^{-1}Mx} = g^x. \]

Easy to find \( x \) from \( g^x \) by brute force.

Security proof: Given \( g^M \), DDH hides rank of \( M \). Inversion would imply algorithm to determine if \( M \) is full rank.

Trapdoor from Decisional Diffie-Hellman (DDH)

Trapdoor functions from DDH [1, 2]: linear algebra in the exponent

\[ \text{pk} = (g^M), \text{sk} = (g, M). \text{ On input } x \in \{0, 1\}^k: \]

**Evaluation:** \[ f(x) = g^{Mx} \]
Trapdoor from Decisional Diffie-Hellman (DDH)

Trapdoor functions from DDH [1, 2]: linear algebra in the exponent

\[ pk = (g^M), \ sk = (g, M). \] On input \( x \in \{0, 1\}^k \):

**Evaluation:** \( f(x) = g^{Mx} \)

**Inversion:** \( f^{-1}(f(x), M) = g^{M^{-1}Mx} = g^x \)

Easy to find \( x \) from \( g^x \) by brute force
Trapdoor functions from DDH [1, 2]: linear algebra in the exponent

$$pk = (g^M), \ sk = (g, M)$$.

On input $$x \in \{0, 1\}^k$$:

**Evaluation:** $$f(x) = g^{Mx}$$

**Inversion:** $$f^{-1}(f(x), M) = g^{M^{-1}Mx} = g^x$$

Easy to find $$x$$ from $$g^x$$ by brute force

**Security proof:** Given $$g^M$$, DDH hides rank of $$M$$. Inversion would imply algorithm to determine if $$M$$ is full rank.


TCF from DDH

Gen($1^{\lambda}$)

1. Choose group $G$ of order $q \sim \mathcal{O}(2^{\lambda})$, and generator $g$
TCF from DDH

Gen($1^\lambda$)

1. Choose group $\mathbb{G}$ of order $q \sim \mathcal{O}(2^\lambda)$, and generator $g$
2. Choose random invertible $M \in \mathbb{Z}_{q}^{k \times k}$ for $k > \log q$
TCF from DDH

Gen(1^\lambda)

1. Choose group $\mathbb{G}$ of order $q \sim \mathcal{O}(2^\lambda)$, and generator $g$
2. Choose random invertible $M \in \mathbb{Z}_q^{k \times k}$ for $k > \log q$
3. Compute $g^M = (g^{M_{ij}}) \in \mathbb{G}^{k \times k}$
Gen(1^\lambda)

1. Choose group $\mathbb{G}$ of order $q \sim \mathcal{O}(2^\lambda)$, and generator $g$
2. Choose random invertible $M \in \mathbb{Z}_q^{k \times k}$ for $k > \log q$
3. Compute $g^M = (g^{M_{ij}}) \in \mathbb{G}^{k \times k}$
4. Choose $s \in \{0, 1\}^k$
TCF from DDH

\[ \text{Gen}(1^\lambda) \]

1. Choose group \( G \) of order \( q \sim \mathcal{O}(2^\lambda) \), and generator \( g \)
2. Choose random invertible \( M \in \mathbb{Z}_q^{k \times k} \) for \( k > \log q \)
3. Compute \( g^M = (g^{M_{ij}}) \in G^{k \times k} \)
4. Choose \( s \in \{0, 1\}^k \)
5. Return \( pk = (g^M, g^{Ms}), sk = (g, M, s) \)
TCF from DDH

Gen(1^λ)

1. Choose group $\mathbb{G}$ of order $q \sim \mathcal{O}(2^λ)$, and generator $g$
2. Choose random invertible $M \in \mathbb{Z}^{k \times k}_q$ for $k > \log q$
3. Compute $g^M = (g^{M_{ij}}) \in \mathbb{G}^{k \times k}$
4. Choose $s \in \{0, 1\}^k$
5. Return $pk = (g^M, g^{Ms}), sk = (g, M, s)$
TCF from DDH

Gen($1^\lambda$)

1. Choose group $\mathbb{G}$ of order $q \sim \mathcal{O}(2^\lambda)$, and generator $g$
2. Choose random invertible $M \in \mathbb{Z}_q^{k \times k}$ for $k > \log q$
3. Compute $g^M = (g^{M_{ij}}) \in \mathbb{G}^{k \times k}$
4. Choose $s \in \{0, 1\}^k$
5. Return $pk = (g^M, g^{Ms})$, $sk = (g, M, s)$

Evaluation:

Let $d \sim \mathcal{O}(k^2)$. Define two functions $f_b : \mathbb{Z}_d^k \rightarrow \mathbb{G}^k$:

$$f_0(x) = g^{Mx} \quad f_1(x) = g^{Mx} g^{Ms} = g^{M(x+s)}$$
TCF from DDH

Gen($1^\lambda$)

1. Choose group $\mathbb{G}$ of order $q \sim \mathcal{O}(2^\lambda)$, and generator $g$
2. Choose random invertible $M \in \mathbb{Z}_q^{k \times k}$ for $k > \log q$
3. Compute $g^M = (g^{M_{ij}}) \in \mathbb{G}^{k \times k}$
4. Choose $s \in \{0, 1\}^k$
5. Return $pk = (g^M, g^{Ms}), sk = (g, M, s)$

**Evaluation:**

Let $d \sim \mathcal{O}(k^2)$. Define two functions $f_b : \mathbb{Z}_d^k \rightarrow \mathbb{G}^k$:

$$f_0(x) = g^{Mx} \quad f_1(x) = g^{Mx}g^{Ms} = g^{M(x+s)}$$

**Inversion:** $f^{-1}(f_0(x), M) = g^{M^{-1}Mx} = g^x$ (poly-time brute force)

GDKM, Choi, Vazirani, Yao ’21 (arXiv:2104.00687)
TCF from DDH: does it help?

- Via elliptic curves, can significantly reduce space requirement
TCF from DDH: does it help?

• Via elliptic curves, can significantly reduce space requirement
• But quantum circuit for group operation is complicated
TCF from DDH: does it help?

- Via elliptic curves, can significantly reduce space requirement
- But quantum circuit for group operation is *complicated*
- Need to perform as many group operations as Shor’s algorithm!
• Via elliptic curves, can significantly reduce space requirement
• But quantum circuit for group operation is complicated
• Need to perform as many group operations as Shor’s algorithm!
• Reversible Euclidean algorithm is hard, maybe irreversible optimization can help?
The CHSH game (Bell test)

Two-player cooperative game.

If anyone receives tails, want $A = B$. If both get heads, want $A \neq B$. 
The CHSH game (Bell test)

Two-player cooperative game.

If anyone receives tails, want $A = B$. If both get heads, want $A \neq B$.

Two players sharing a Bell pair:
The CHSH game (Bell test)

Two-player cooperative game.

If anyone receives tails, want $A = B$. If both get heads, want $A \neq B$.

Two players sharing a Bell pair:
The CHSH game (Bell test)

Two-player cooperative game.

If anyone receives tails, want $A = B$. If both get heads, want $A \neq B$.

Two players sharing a Bell pair:
The CHSH game (Bell test)

Two-player cooperative game.

Player 1

Referee

Player 2

If anyone receives tails, want $A = B$. If both get heads, want $A \neq B$.

Two players sharing a Bell pair:

Quantum: $\cos^2(\pi/8) \approx 85$

Classical: 75%
Full protocol

Prover (quantum)

Round 1
2. Generate state $\sum_{x=0}^{N/2} |x\rangle_y |f_i(x)\rangle_y$
3. Measure $y$ register, yielding bitstring $y$
   State is now $(|x_0\rangle + |x_1\rangle)_x |y\rangle_y$.
   $y$ register can be discarded

If preimage requested:
6a. Projectively measure $x$ register, yielding $x$

Otherwise, continue:

Round 2
7b. Add one ancilla $b$; use CNOTs to compute
   $|r \cdot x_0\rangle_b |x_0\rangle_x + |r \cdot x_1\rangle_b |x_1\rangle_x$ where
   $r \cdot x$ is bitwise inner product
8b. Measure $x$ register in Hadamard basis,
   yielding a string $d$.
   Discard $x$, state is now
   $|\psi\rangle_b \in \{|0\rangle, |1\rangle, |\rangle, |\rangle, |-\rangle\}$

Round 3
11b. Measure ancilla $b$ in the rotated basis
   $\left\{ \cos \left( \frac{\pi}{4} \right) \left| 0 \right\rangle + \sin \left( \frac{\pi}{4} \right) \left| 1 \right\rangle \right\}$, yielding a bit $b$

Verifier (classical)

1. Sample $(f_i, t) \leftarrow \text{Gen}(1^n)$

4. Using trapdoor $t$ compute $x_0$ and $x_1$

5. Randomly choose to request a preimage or continue

7a. If $x \in \{x_0, x_1\}$ return Accept

6b. Choose random bitstring $r$

9b. Using $r, x_0, x_1, d$, determine $|\psi\rangle_b$

10b. Choose random $m \in \{\frac{\pi}{4}, -\frac{\pi}{4}\}$

11b. If $b$ was likely given $|\psi\rangle_b$ return Accept