Kuperberg's Collimation Sieve vs. CSIDH

Chris Peikert

University of Michigan

Quantum Cryptanalysis of Post-Quantum Cryptography
Simons Institute
24 February 2020

He Gives C-Sieves on the CSIDH

Chris Peikert

University of Michigan

Quantum Cryptanalysis of Post-Quantum Cryptography
Simons Institute
24 February 2020

Conclusions

1. Proposed CSIDH parameters have relatively little quantum security beyond the cost of quantum evaluation (on a uniform superposition).

Conclusions

1. Proposed CSIDH parameters have relatively little quantum security beyond the cost of quantum evaluation (on a uniform superposition).

2. CSIDH-512 key recovery costs, e.g., only $\approx 2^{16}$ evaluations using $\approx 2^{40}$ bits of quantum-accessible RAM ($+$ small other resources).

Conclusions

1. Proposed CSIDH parameters have relatively little quantum security beyond the cost of quantum evaluation (on a uniform superposition).

2. CSIDH-512 key recovery costs, e.g., only $\approx 2^{16}$ evaluations using $\approx 2^{40}$ bits of quantum-accessible RAM (+ small other resources).

3. Assuming evaluation costs not much more than for the 'best case':

Conclusions

1. Proposed CSIDH parameters have relatively little quantum security beyond the cost of quantum evaluation (on a uniform superposition).

2. CSIDH-512 key recovery costs, e.g., only $\approx 2^{16}$ evaluations using $\approx 2^{40}$ bits of quantum-accessible RAM (+ small other resources).

3. Assuming evaluation costs not much more than for the 'best case':

   CSIDH-512 breakable with $\approx 2^{60}$ T-gates

Conclusions

1. Proposed CSIDH parameters have relatively little quantum security beyond the cost of quantum evaluation (on a uniform superposition).

2. CSIDH-512 key recovery costs, e.g., only $\approx 2^{16}$ evaluations using $\approx 2^{40}$ bits of quantum-accessible RAM ($+$ small other resources).

3. Assuming evaluation costs not much more than for the 'best case':

   CSIDH-512 breakable with $\approx 2^{60}$ T-gates, so falls well short of its claimed NIST level 1 p-q security. ($\geq 2^{170}$/MAXDEPTH)

Conclusions

1. Proposed CSIDH parameters have relatively little quantum security beyond the cost of quantum evaluation (on a uniform superposition).

2. CSIDH-512 key recovery costs, e.g., only $\approx 2^{16}$ evaluations using $\approx 2^{40}$ bits of quantum-accessible RAM ($+$ small other resources).

3. Assuming evaluation costs not much more than for the 'best case':

   CSIDH-512 breakable with $\approx 2^{60}$ T-gates, so falls well short of its claimed NIST level 1 p-q security. ($\geq 2^{170}$/MAXDEPTH)

   CSIDH-1024 breakable with $\approx 2^{72}$ T-gates and $\approx 2^{44}$ bits QRACM

Conclusions

1. Proposed CSIDH parameters have relatively little quantum security beyond the cost of quantum evaluation (on a uniform superposition).

2. CSIDH-512 key recovery costs, e.g., only $\approx 2^{16}$ evaluations using $\approx 2^{40}$ bits of quantum-accessible RAM ($+$ small other resources).

3. Assuming evaluation costs not much more than for the 'best case':

   CSIDH-512 breakable with $\approx 2^{60}$ T-gates, so falls well short of its claimed NIST level 1 p-q security. ($\geq 2^{170}$/MAXDEPTH)

   CSIDH-1024 breakable with $\approx 2^{72}$ T-gates and $\approx 2^{44}$ bits QRACM, so it also falls short of level 1.

Conclusions

1. Proposed CSIDH parameters have relatively little quantum security beyond the cost of quantum evaluation (on a uniform superposition).

2. CSIDH-512 key recovery costs, e.g., only $\approx 2^{16}$ evaluations using $\approx 2^{40}$ bits of quantum-accessible RAM (+ small other resources).

3. Assuming evaluation costs not much more than for the 'best case':

   CSIDH-512 breakable with $\approx 2^{60}$ T-gates, so falls well short of its claimed NIST level 1 p-q security. ($\geq 2^{170}$/MAXDEPTH)

   CSIDH-1024 breakable with $\approx 2^{72}$ T-gates and $\approx 2^{44}$ bits QRACM, so it also falls short of level 1.

   CSIDH-1792

Conclusions

1. Proposed CSIDH parameters have relatively little quantum security beyond the cost of quantum evaluation (on a uniform superposition).

2. CSIDH-512 key recovery costs, e.g., only $\approx 2^{16}$ evaluations using $\approx 2^{40}$ bits of quantum-accessible RAM (+ small other resources).

3. Assuming evaluation costs not much more than for the 'best case':

   CSIDH-512 breakable with $\approx 2^{60}$ T-gates, so falls well short of its claimed NIST level 1 p-q security. ($\geq 2^{170}$/MAXDEPTH)

   CSIDH-1024 breakable with $\approx 2^{72}$ T-gates and $\approx 2^{44}$ bits QRACM, so it also falls short of level 1.

   CSIDH-1792 breakable with $\approx 2^{84}$ T-gates and $\approx 2^{48}$ bits QRACM

Conclusions

1. Proposed CSIDH parameters have relatively little quantum security beyond the cost of quantum evaluation (on a uniform superposition).

2. CSIDH-512 key recovery costs, e.g., only $\approx 2^{16}$ evaluations using $\approx 2^{40}$ bits of quantum-accessible RAM (+ small other resources).

3. Assuming evaluation costs not much more than for the 'best case':

   CSIDH-512 breakable with $\approx 2^{60}$ T-gates, so falls well short of its claimed NIST level 1 p-q security. ($\geq 2^{170}$/MAXDEPTH)

   CSIDH-1024 breakable with $\approx 2^{72}$ T-gates and $\approx 2^{44}$ bits QRACM, so it also falls short of level 1.

   CSIDH-1792 breakable with $\approx 2^{84}$ T-gates and $\approx 2^{48}$ bits QRACM, so it also doesn't reach level 1

   possibly except for high end of MAXDEPTH range.

CSIDH ('sea-side') [CastryckLangeMartindalePannyRenes'18]

- Isogeny-based 'post-quantum commutative group action' following [Couveignes'97,RostovtsevStolbunov'06]: abelian group G, set Z, action

$$\star \colon G \times Z \to Z$$

CSIDH ('sea-side') [CastryckLangeMartindalePannyRenes'18]

- Isogeny-based 'post-quantum commutative group action' following [Couveignes'97,RostovtsevStolbunov'06]: abelian group G, set Z, action

$$\star\colon G \times Z \to Z$$

(Other isogeny-based crypto like SIDH [JF'11,...]: nonabelian, no group action.)

CSIDH ('sea-side') [CastryckLangeMartindalePannyRenes'18]

- Isogeny-based 'post-quantum commutative group action' following [Couveignes'97,RostovtsevStolbunov'06]: abelian group G, set Z, action

$$\star\colon G \times Z \to Z$$

(Other isogeny-based crypto like SIDH [JF'11,...]: nonabelian, no group action.)

DiffieHellman-style noninteractive key exchange with public param $z \in Z$:

Alice: secret $a \in G$, public $p_A = a \star z \in Z$

Bob: secret $b \in G$, public $p_B = b \star z \in Z$

Shared key: $a \star p_B = b \star p_A = (a + b) \star z$, by commutativity

CSIDH ('sea-side') [CastryckLangeMartindalePannyRenes'18]

- Isogeny-based 'post-quantum commutative group action' following [Couveignes'97,RostovtsevStolbunov'06]: abelian group G, set Z, action

$$\star\colon G \times Z \to Z$$

(Other isogeny-based crypto like SIDH [JF'11,...]: nonabelian, no group action.)

DiffieHellman-style noninteractive key exchange with public param $z \in Z$:

Alice: secret $a \in G$, public $p_A = a \star z \in Z$

Bob: secret $b \in G$, public $p_B = b \star z \in Z$

Shared key: $a \star p_B = b \star p_A = (a + b) \star z$, by commutativity

- Efficient! 64-byte keys, 80ms key exchange for claimed NIST level 1 quantum security: as hard as AES-128 key search

CSIDH ('sea-side') [CastryckLangeMartindalePannyRenes'18]

- Isogeny-based 'post-quantum commutative group action' following [Couveignes'97,RostovtsevStolbunov'06]: abelian group G, set Z, action

$$\star\colon G \times Z \to Z$$

(Other isogeny-based crypto like SIDH [JF'11,...]: nonabelian, no group action.)

DiffieHellman-style noninteractive key exchange with public param $z \in Z$:

Alice: secret $a \in G$, public $p_A = a \star z \in Z$

Bob: secret $b \in G$, public $p_B = b \star z \in Z$

Shared key: $a \star p_B = b \star p_A = (a + b) \star z$, by commutativity

- Efficient! 64-byte keys, 80ms key exchange for claimed NIST level 1 quantum security: as hard as AES-128 key search

- Signatures [Stolbunov'12,DeFeoGalbraith'19,BeullensKleinjungVercauteren'19]: pk + sig = 1468 bytes at same claimed security level

Attacking the CSIDH, Quantumly

- Secret-key recovery: given $z, a \star z \in Z$, find $a \in G$ (or equivalent)

Attacking the CSIDH, Quantumly

- Secret-key recovery: given $z, a \star z \in Z$, find $a \in G$ (or equivalent)
  Reduces to Hidden-Shift Problem (HShP) on G [ChildsJaoSoukharev'10]

Attacking the CSIDH, Quantumly

- Secret-key recovery: given $z, a \star z \in Z$, find $a \in G$ (or equivalent)
  Reduces to Hidden-Shift Problem (HShP) on G [ChildsJaoSoukharev'10]

Quantum HShP Algorithm Ingredients [Kuperberg'03,...]

1. Oracle outputs random 'labeled' quantum states, by evaluating $\star$ on a uniform superposition over G.

Attacking the CSIDH, Quantumly

- Secret-key recovery: given $z, a \star z \in Z$, find $a \in G$ (or equivalent)
  Reduces to Hidden-Shift Problem (HShP) on G [ChildsJaoSoukharev'10]

Quantum HShP Algorithm Ingredients [Kuperberg'03,...]

1. Oracle outputs random 'labeled' quantum states, by evaluating $\star$ on a uniform superposition over G.
2. Sieve combines labeled states to generate 'more favorable' ones.

Attacking the CSIDH, Quantumly

- Secret-key recovery: given $z, a \star z \in Z$, find $a \in G$ (or equivalent)
  Reduces to Hidden-Shift Problem (HShP) on G [ChildsJaoSoukharev'10]

Quantum HShP Algorithm Ingredients [Kuperberg'03, ...]

1. Oracle outputs random 'labeled' quantum states, by evaluating $\star$ on a uniform superposition over G.
2. Sieve combines labeled states to generate 'more favorable' ones.
3. Measurement of 'very favorable' state recovers bit(s) of hidden shift.

Attacking the CSIDH, Quantumly

- Secret-key recovery: given $z, a \star z \in Z$, find $a \in G$ (or equivalent)
  Reduces to Hidden-Shift Problem (HShP) on G [ChildsJaoSoukharev'10]

Quantum HShP Algorithm Ingredients [Kuperberg'03,...]

1. Oracle outputs random 'labeled' quantum states, by evaluating $\star$ on a uniform superposition over G.
2. Sieve combines labeled states to generate 'more favorable' ones.
3. Measurement of 'very favorable' state recovers bit(s) of hidden shift.

Sieve Algorithms

[Kuperberg'03] $2^{O(\sqrt{n})}$ oracle queries and qubits $(n = \log|G|)$

Attacking the CSIDH, Quantumly

- Secret-key recovery: given $z, a \star z \in Z$, find $a \in G$ (or equivalent)
  Reduces to Hidden-Shift Problem (HShP) on G [ChildsJaoSoukharev'10]

Quantum HShP Algorithm Ingredients [Kuperberg'03,...]

1. Oracle outputs random 'labeled' quantum states, by evaluating $\star$ on a uniform superposition over G.
2. Sieve combines labeled states to generate 'more favorable' ones.
3. Measurement of 'very favorable' state recovers bit(s) of hidden shift.

Sieve Algorithms

[Kuperberg'03] $2^{O(\sqrt{n})}$ oracle queries and qubits $(n = \log|G|)$

[Regev'04] $2^{O(\sqrt{n \log n})}$ oracle queries, only poly(n) qubits

Attacking the CSIDH, Quantumly

- Secret-key recovery: given $z, a \star z \in Z$, find $a \in G$ (or equivalent)
  Reduces to Hidden-Shift Problem (HShP) on G [ChildsJaoSoukharev'10]

Quantum HShP Algorithm Ingredients [Kuperberg'03,...]

1. Oracle outputs random 'labeled' quantum states, by evaluating $\star$ on a uniform superposition over G.
2. Sieve combines labeled states to generate 'more favorable' ones.
3. Measurement of 'very favorable' state recovers bit(s) of hidden shift.

Sieve Algorithms

[Kuperberg'03] $2^{O(\sqrt{n})}$ oracle queries and qubits $(n = \log|G|)$

[Regev'04] $2^{O(\sqrt{n \log n})}$ oracle queries, only poly(n) qubits

[Kuperberg'11] $2^{O(\sqrt{n})}$ oracle queries and bits of quantum-accessible RAM.

Attacking the CSIDH, Quantumly

- Secret-key recovery: given $z, a \star z \in Z$, find $a \in G$ (or equivalent)
  Reduces to Hidden-Shift Problem (HShP) on G [ChildsJaoSoukharev'10]

Quantum HShP Algorithm Ingredients [Kuperberg'03,...]

1. Oracle outputs random 'labeled' quantum states, by evaluating $\star$ on a uniform superposition over G.
2. Sieve combines labeled states to generate 'more favorable' ones.
3. Measurement of 'very favorable' state recovers bit(s) of hidden shift.

Sieve Algorithms

[Kuperberg'03] $2^{O(\sqrt{n})}$ oracle queries and qubits $(n = \log|G|)$

[Regev'04] $2^{O(\sqrt{n \log n})}$ oracle queries, only poly(n) qubits

[Kuperberg'11] $2^{O(\sqrt{n})}$ oracle queries and bits of quantum-accessible RAM.
'Collimation sieve' subsumes prior two, offers more trade-offs.
E.g., $\log(\text{queries}) \cdot \log(\text{QRACM}) \gtrsim n$.

Prior Security Estimates for CSIDH-512

- Oracle costs $\leq 2^{43.3}$ T-gates ($+$ much cheaper linear gates) for 'best case,' somewhat non-uniform superposition [BLMP'19]

Prior Security Estimates for CSIDH-512

- Oracle costs $\leq 2^{43.3}$ T-gates ($+$ much cheaper linear gates) for 'best case,' somewhat non-uniform superposition [BLMP'19]

  Good reason to expect similar cost for uniform superposition [BKV'19]

Prior Security Estimates for CSIDH-512

- Oracle costs $\leq 2^{43.3}$ T-gates ($+$ much cheaper linear gates) for 'best case,' somewhat non-uniform superposition [BLMP'19]

  Good reason to expect similar cost for uniform superposition [BKV'19]

- Sieve costs:

Work	Algorithm	Oracle queries	Sieve memory
CSIDH paper [CLMPR'18]	[Regev'04]	2^{62}	poly(n)

Prior Security Estimates for CSIDH-512

- Oracle costs $\leq 2^{43.3}$ T-gates ($+$ much cheaper linear gates) for 'best case,' somewhat non-uniform superposition [BLMP'19]

  Good reason to expect similar cost for uniform superposition [BKV'19]
- Sieve costs:

Work	Algorithm	Oracle queries	Sieve memory
CSIDH paper [CLMPR'18]	[Regev'04]	2^{62}	poly(n)
[BonnetainSchrottenloher'18]	[Kuperberg'03]	$2^{32.5}$	2^{31} qubits

Prior Security Estimates for CSIDH-512

- Oracle costs $\leq 2^{43.3}$ T-gates ($+$ much cheaper linear gates) for 'best case,' somewhat non-uniform superposition [BLMP'19]

  Good reason to expect similar cost for uniform superposition [BKV'19]

- Sieve costs:

Work	Algorithm	Oracle queries	Sieve memory
CSIDH paper [CLMPR'18]	[Regev'04]	2^{62}	poly(n)
[BonnetainSchrottenloher'18]	[Kuperberg'03]	$2^{32.5}$	2^{31} qubits
None prior!	[Kuperberg'11]	??	??

Our Contributions

- We generalize and practically improve Kuperberg's c-sieve, and analyze its concrete complexity on proposed CSIDH parameters:

Our Contributions

- We generalize and practically improve Kuperberg's c-sieve, and analyze its concrete complexity on proposed CSIDH parameters:
  - ⋆ Handle arbitrary group orders (generalizing from two-power/smooth)
  - ⋆ Recover several secret bits from each sieve run
  - ⋆ Control (classical) memory and time complexities better
  - ⋆ Run simulations up to the exact CSIDH-512 order $|G| \approx 2^{257.1}$

Our Contributions

- We generalize and practically improve Kuperberg's c-sieve, and analyze its concrete complexity on proposed CSIDH parameters:
  - ⋆ Handle arbitrary group orders (generalizing from two-power/smooth)
  - ⋆ Recover several secret bits from each sieve run
  - ⋆ Control (classical) memory and time complexities better
  - ⋆ Run simulations up to the exact CSIDH-512 order $|G| \approx 2^{257.1}$

Work	Algorithm	Oracle queries	Sieve memory
[CLMPR'18]	[Regev'04]	2^{62}	poly(n)
[BS'18]	[Kuperberg'03]	$2^{32.5}$	2^{31} qubits
This work	[Kuperberg'11]	$2^{18.7}$	2^{32} bits QRACM
		$2^{15.7}$	2^{40} bits QRACM
		$2^{14.1}$	2^{48} bits QRACM

Our Contributions

- We generalize and practically improve Kuperberg's c-sieve, and analyze its concrete complexity on proposed CSIDH parameters:
  - ⋆ Handle arbitrary group orders (generalizing from two-power/smooth)
  - ⋆ Recover several secret bits from each sieve run
  - ⋆ Control (classical) memory and time complexities better
  - ⋆ Run simulations up to the exact CSIDH-512 order $|G| \approx 2^{257.1}$

Work	Algorithm	Oracle queries	Sieve memory
[CLMPR'18]	[Regev'04]	2^{62}	poly(n)
[BS'18]	[Kuperberg'03]	$2^{32.5}$	2^{31} qubits
This work	[Kuperberg'11]	$2^{18.7}$	2^{32} bits QRACM
		$2^{15.7}$	2^{40} bits QRACM
		$2^{14.1}$	2^{48} bits QRACM

*Independently, Bonnetain and Schrottenloher gave a complementary, theoretical c-sieve analysis, arriving at similar conclusions.

Hidden Shifts and CRS-Style Crypto

Hidden-Shift Problem on Group $(G, +)$

- Given injective $f_0, f_1 \colon G \to Z$ such that $f_1(x) = f_0(x + s)$ for some 'secret' $s \in G$, find s.

Hidden Shifts and CRS-Style Crypto

Hidden-Shift Problem on Group $(G, +)$

- Given injective $f_0, f_1 \colon G \to Z$ such that $f_1(x) = f_0(x + s)$ for some 'secret' $s \in G$, find s.

Attacking CRS via HShP [ChildsJaoSoukharev'10]

- Fix a commutative group action $\star \colon G \times Z \to Z$.

Hidden Shifts and CRS-Style Crypto

Hidden-Shift Problem on Group $(G, +)$

- Given injective $f_0, f_1 \colon G \to Z$ such that $f_1(x) = f_0(x + s)$ for some 'secret' $s \in G$, find s.

Attacking CRS via HShP [ChildsJaoSoukharev'10]

- Fix a commutative group action $\star \colon G \times Z \to Z$.
- For base value $z_0 \in Z$ and public key $z_1 = s \star z_0$, define

$$\begin{aligned} f_b \colon G &\to Z \\ g &\mapsto g \star z_b. \end{aligned}$$

Hidden Shifts and CRS-Style Crypto

Hidden-Shift Problem on Group $(G, +)$

- Given injective $f_0, f_1 \colon G \to Z$ such that $f_1(x) = f_0(x + s)$ for some 'secret' $s \in G$, find s.

Attacking CRS via HShP [ChildsJaoSoukharev'10]

- Fix a commutative group action $\star \colon G \times Z \to Z$.
- For base value $z_0 \in Z$ and public key $z_1 = s \star z_0$, define

$$\begin{aligned} f_b \colon G &\to Z \\ g &\mapsto g \star z_b. \end{aligned}$$

Then f_b is injective because $\star$ is free and transitive, and

$$f_1(x) = x \star z_1 = x \star (s \star z_0) = (x + s) \star z_0 = f_0(x + s).$$

Hidden Shifts and CRS-Style Crypto

Hidden-Shift Problem on Group $(G, +)$

- Given injective $f_0, f_1 \colon G \to Z$ such that $f_1(x) = f_0(x + s)$ for some 'secret' $s \in G$, find s.

Attacking CRS via HShP [ChildsJaoSoukharev'10]

- Fix a commutative group action $\star \colon G \times Z \to Z$.
- For base value $z_0 \in Z$ and public key $z_1 = s \star z_0$, define

$$\begin{aligned} f_b \colon G &\to Z \\ g &\mapsto g \star z_b. \end{aligned}$$

  Then f_b is injective because $\star$ is free and transitive, and

$$f_1(x) = x \star z_1 = x \star (s \star z_0) = (x + s) \star z_0 = f_0(x + s).$$

- So, solving HShP for this f_0, f_1 recovers the secret key s.

Overview of ‘High Bits’ Collimation Sieve

- Solves HShP on a finite cyclic group $\mathbb{Z}_N$ of known order N.

Overview of 'High Bits' Collimation Sieve

- Solves HShP on a finite cyclic group $\mathbb{Z}_N$ of known order N.
- Works with (pure) quantum states called phase vectors, each having a vector of integer (phase) multipliers.

Overview of ‘High Bits’ Collimation Sieve

- Solves HShP on a finite cyclic group $\mathbb{Z}_N$ of known order N.
- Works with (pure) quantum states called phase vectors, each having a vector of integer (phase) multipliers.

  Given: ‘fresh’ phase vectors with huge (random) multipliers in $[N]$, of any desired feasible length L.

Overview of 'High Bits' Collimation Sieve

- Solves HShP on a finite cyclic group $\mathbb{Z}_N$ of known order N.
- Works with (pure) quantum states called phase vectors, each having a vector of integer (phase) multipliers.

  Given: 'fresh' phase vectors with huge (random) multipliers in $[N]$, of any desired feasible length L.

  Goal: construct a 'very nice' length-L phase vector having small (random) multipliers in $[S] = \{0, 1, \ldots, S-1\}$, for $S \lesssim L$.

Overview of 'High Bits' Collimation Sieve

- Solves HShP on a finite cyclic group $\mathbb{Z}_N$ of known order N.
- Works with (pure) quantum states called phase vectors, each having a vector of integer (phase) multipliers.

Given: 'fresh' phase vectors with huge (random) multipliers in $[N]$, of any desired feasible length L.

Goal: construct a 'very nice' length-L phase vector having small (random) multipliers in $[S] = \{0, 1, \ldots, S-1\}$, for $S \lesssim L$.

From this we can extract secret bit(s) using QFT.

Overview of 'High Bits' Collimation Sieve

- Solves HShP on a finite cyclic group $\mathbb{Z}_N$ of known order N.
- Works with (pure) quantum states called phase vectors, each having a vector of integer (phase) multipliers.

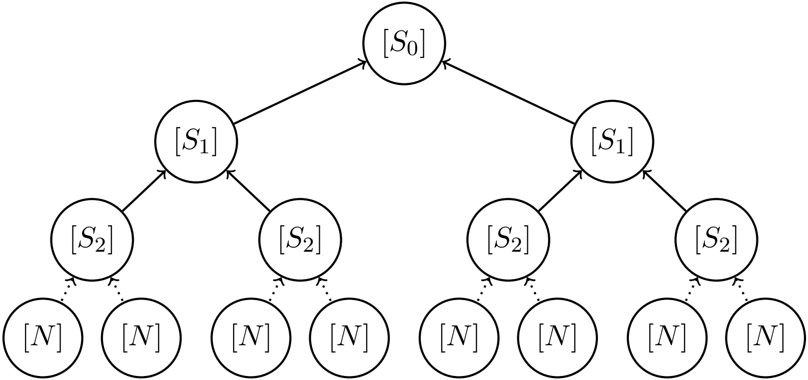
Given: 'fresh' phase vectors with huge (random) multipliers in $[N]$, of any desired feasible length L.

Goal: construct a 'very nice' length-L phase vector having small (random) multipliers in $[S] = \{0, 1, \ldots, S-1\}$, for $S \lesssim L$.

From this we can extract secret bit(s) using QFT.

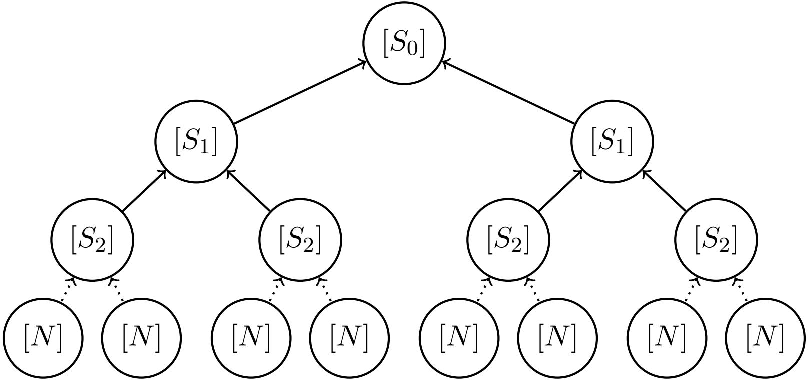
How: make progressively 'nicer' phase vectors with multipliers in successively smaller intervals, by collimating vectors.

Collimation Sieve Structure



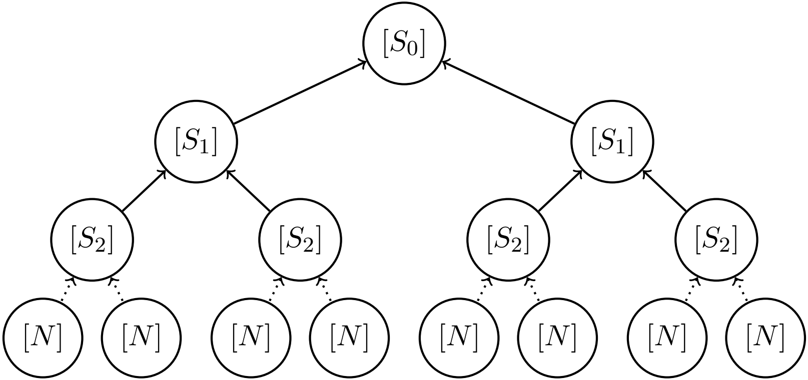
- Fix interval sizes $L \approx S_0 < S_1 < \cdots < S_d = N$, for $S_{i+1}/S_i \approx L$. Depth $d \approx \log_L(N) - 1 = \log(N)/\log(L) - 1$.

Collimation Sieve Structure



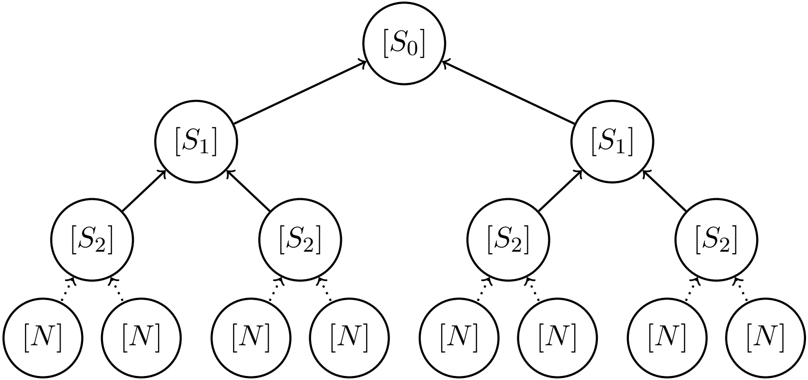
- Fix interval sizes $L \approx S_0 < S_1 < \cdots < S_d = N$, for $S_{i+1}/S_i \approx L$. Depth $d \approx \log_L(N) - 1 = \log(N)/\log(L) - 1$.
- Leaf nodes get 'fresh' length-L phase vectors on $[N]$.

Collimation Sieve Structure



- Fix interval sizes $L \approx S_0 < S_1 < \cdots < S_d = N$, for $S_{i+1}/S_i \approx L$. Depth $d \approx \log_L(N) - 1 = \log(N)/\log(L) - 1$.
- Leaf nodes get 'fresh' length-L phase vectors on $[N]$.
- Each internal node collimates its children, narrowing range by $\approx L$.

Collimation Sieve Structure



- Fix interval sizes $L \approx S_0 < S_1 < \cdots < S_d = N$, for $S_{i+1}/S_i \approx L$. Depth $d \approx \log_L(N) - 1 = \log(N)/\log(L) - 1$.
- Leaf nodes get 'fresh' length-L phase vectors on $[N]$.
- Each internal node collimates its children, narrowing range by $\approx L$.
- Key insight: more QRACM $\Longrightarrow$ larger L, lower depth, fewer vectors

Phase Vectors

- For $s \in \mathbb{Z}_N$, a phase vector of length L is a pure quantum state

$$|\psi\rangle \propto \sum_{j \in [L]} \chi(b(j) \cdot s/N)|j\rangle \ , \quad \chi(x) = \exp(2\pi i \cdot x)$$

where the (known) $b(j) \in [N]$ are its phase multipliers.

Phase Vectors

- For $s \in \mathbb{Z}_N$, a phase vector of length L is a pure quantum state

$$|\psi\rangle \propto \sum_{j \in [L]} \chi(b(j) \cdot s/N)|j\rangle \ , \quad \chi(x) = \exp(2\pi i \cdot x)$$

where the (known) $b(j) \in [N]$ are its phase multipliers.

- E.g., we get qubit $|\psi\rangle \propto |0\rangle + \chi(b' \cdot s/N)|1\rangle$ for uniform $b' \in [N]$ by querying the hidden-shift oracle. So $L = 2$, $b(0) = 0$, and $b(1) = b'$.

Phase Vectors

- For $s \in \mathbb{Z}_N$, a phase vector of length L is a pure quantum state

$$|\psi\rangle \propto \sum_{j \in [L]} \chi(b(j) \cdot s/N)|j\rangle \ , \quad \chi(x) = \exp(2\pi i \cdot x)$$

  where the (known) $b(j) \in [N]$ are its phase multipliers.

- E.g., we get qubit $|\psi\rangle \propto |0\rangle + \chi(b' \cdot s/N)|1\rangle$ for uniform $b' \in [N]$ by querying the hidden-shift oracle. So $L = 2$, $b(0) = 0$, and $b(1) = b'$.

- In general, we store the phase multipliers in a sorted list. So a phase vector requires $\tilde{O}(L)$ bits but only $\log L$ qubits.

Phase Vectors

- For $s \in \mathbb{Z}_N$, a phase vector of length L is a pure quantum state

$$|\psi\rangle \propto \sum_{j \in [L]} \chi(b(j) \cdot s/N)|j\rangle \ , \quad \chi(x) = \exp(2\pi i \cdot x)$$

  where the (known) $b(j) \in [N]$ are its phase multipliers.

- E.g., we get qubit $|\psi\rangle \propto |0\rangle + \chi(b' \cdot s/N)|1\rangle$ for uniform $b' \in [N]$ by querying the hidden-shift oracle. So $L = 2$, $b(0) = 0$, and $b(1) = b'$.

- In general, we store the phase multipliers in a sorted list.
  So a phase vector requires $\tilde{O}(L)$ bits but only $\log L$ qubits.

- This is the source of the exponential improvement in quantum space versus Kuperberg's first sieve.

Combining Phase Vectors

- Given phase vectors $|\psi_1\rangle, |\psi_2\rangle$ of lengths L_1, L_2 with multiplier functions b_1, b_2, tensoring them yields a state

$$|\psi'\rangle = |\psi_1, \psi_2\rangle \propto \sum_{j_1 \in [L_1]} \sum_{j_2 \in [L_2]} \chi(b_1(j_1) \cdot s/N) \cdot \chi(b_2(j_2) \cdot s/N) |j_1, j_2\rangle$$
$$= \sum_{\vec{j} \in L} \chi(b'(\vec{j}) \cdot s/N) |\vec{j}\rangle$$

where $b'(\vec{j}) = b_1(j_1) + b_2(j_2)$ and $L = [L_1] \times [L_2] \cong [L_1 L_2]$.

Combining Phase Vectors

- Given phase vectors $|\psi_1\rangle, |\psi_2\rangle$ of lengths L_1, L_2 with multiplier functions b_1, b_2, tensoring them yields a state

$$|\psi'\rangle = |\psi_1, \psi_2\rangle \propto \sum_{j_1 \in [L_1]} \sum_{j_2 \in [L_2]} \chi(b_1(j_1) \cdot s/N) \cdot \chi(b_2(j_2) \cdot s/N) |j_1, j_2\rangle$$
$$= \sum_{\vec{j} \in L} \chi(b'(\vec{j}) \cdot s/N) |\vec{j}\rangle$$

  where $b'(\vec{j}) = b_1(j_1) + b_2(j_2)$ and $L = [L_1] \times [L_2] \cong [L_1 L_2]$.

- E.g., ℓ 'fresh' labeled qubits from the oracle yield a length-2^ℓ phase vector whose multipliers are the (mod-N) subset-sums of the labels.

Combining Phase Vectors

- Given phase vectors $|\psi_1\rangle, |\psi_2\rangle$ of lengths L_1, L_2 with multiplier functions b_1, b_2, tensoring them yields a state

$$|\psi'\rangle = |\psi_1, \psi_2\rangle \propto \sum_{j_1 \in [L_1]} \sum_{j_2 \in [L_2]} \chi(b_1(j_1) \cdot s/N) \cdot \chi(b_2(j_2) \cdot s/N) |j_1, j_2\rangle$$
$$= \sum_{\vec{j} \in L} \chi(b'(\vec{j}) \cdot s/N) |\vec{j}\rangle$$

  where $b'(\vec{j}) = b_1(j_1) + b_2(j_2)$ and $L = [L_1] \times [L_2] \cong [L_1 L_2]$.

- E.g., ℓ 'fresh' labeled qubits from the oracle yield a length-2^ℓ phase vector whose multipliers are the (mod-N) subset-sums of the labels.

  This yields a 'fresh' length-L phase vector on $[N]$, in $\log L$ queries.

Combining Phase Vectors

- Given phase vectors $|\psi_1\rangle, |\psi_2\rangle$ of lengths L_1, L_2 with multiplier functions b_1, b_2, tensoring them yields a state

$$|\psi'\rangle = |\psi_1, \psi_2\rangle \propto \sum_{j_1 \in [L_1]} \sum_{j_2 \in [L_2]} \chi(b_1(j_1) \cdot s/N) \cdot \chi(b_2(j_2) \cdot s/N) |j_1, j_2\rangle$$
$$= \sum_{\vec{j} \in L} \chi(b'(\vec{j}) \cdot s/N) |\vec{j}\rangle$$

  where $b'(\vec{j}) = b_1(j_1) + b_2(j_2)$ and $L = [L_1] \times [L_2] \cong [L_1 L_2]$.

- E.g., ℓ 'fresh' labeled qubits from the oracle yield a length-2^ℓ phase vector whose multipliers are the (mod-N) subset-sums of the labels.

  This yields a 'fresh' length-L phase vector on $[N]$, in $\log L$ queries.

- A more interesting combination procedure: collimation...

Collimation Procedure

Given: two phase vectors $|\psi_i\rangle$ of length $L_i \approx L$ on $[S']$

Goal: one phase vector $|\psi\rangle$ of length $\approx L$ on $[S]$, for $S \approx S'/L$

Collimation Procedure

Given: two phase vectors $|\psi_i\rangle$ of length $L_i \approx L$ on $[S']$

Goal: one phase vector $|\psi\rangle$ of length $\approx L$ on $[S]$, for $S \approx S'/L$

How:

1. Form the phase vector $|\psi'\rangle = |\psi_1, \psi_2\rangle$ with index set $[L_1] \times [L_2]$ and multipliers $b'(\vec{\jmath}) = b_1(j_1) + b_2(j_2)$.

Collimation Procedure

Given: two phase vectors $|\psi_i\rangle$ of length $L_i \approx L$ on $[S']$

Goal: one phase vector $|\psi\rangle$ of length $\approx L$ on $[S]$, for $S \approx S'/L$

How:

1. Form the phase vector $|\psi'\rangle = |\psi_1, \psi_2\rangle$ with index set $[L_1] \times [L_2]$ and multipliers $b'(\vec{\jmath}) = b_1(j_1) + b_2(j_2)$.
2. Measure $|\psi'\rangle$ according to $q = \lfloor b'(\vec{\jmath})/S \rfloor$.
   All 'surviving' multipliers are in $[S]$, up to global phase.

Collimation Procedure

Given: two phase vectors $|\psi_i\rangle$ of length $L_i \approx L$ on $[S']$

Goal: one phase vector $|\psi\rangle$ of length $\approx L$ on $[S]$, for $S \approx S'/L$

How:

1. Form the phase vector $|\psi'\rangle = |\psi_1, \psi_2\rangle$ with index set $[L_1] \times [L_2]$ and multipliers $b'(\vec{\jmath}) = b_1(j_1) + b_2(j_2)$.
2. Measure $|\psi'\rangle$ according to $q = \lfloor b'(\vec{\jmath})/S \rfloor$. All 'surviving' multipliers are in $[S]$, up to global phase.
3. Compute the subset $J \subseteq [L_1] \times [L_2]$ of $\vec{\jmath}$ that satisfy the above, reindex J to $[|J|]$, and output the resulting $|\psi\rangle$.

Collimation Procedure

Given: two phase vectors $|\psi_i\rangle$ of length $L_i \approx L$ on $[S']$

Goal: one phase vector $|\psi\rangle$ of length $\approx L$ on $[S]$, for $S \approx S'/L$

How:

1. Form the phase vector $|\psi'\rangle = |\psi_1, \psi_2\rangle$ with index set $[L_1] \times [L_2]$ and multipliers $b'(\vec{j}) = b_1(j_1) + b_2(j_2)$.
2. Measure $|\psi'\rangle$ according to $q = \lfloor b'(\vec{j})/S \rfloor$.
   All 'surviving' multipliers are in $[S]$, up to global phase.
3. Compute the subset $J \subseteq [L_1] \times [L_2]$ of $\vec{j}$ that satisfy the above, reindex J to $[|J|]$, and output the resulting $|\psi\rangle$.

Analysis

- Phase vector $|\psi'\rangle$ has length $L_1 L_2 \approx L^2$, and the multipliers $b'(\vec{j})$ are well distributed in $[2S']$.

Collimation Procedure

Given: two phase vectors $|\psi_i\rangle$ of length $L_i \approx L$ on $[S']$

Goal: one phase vector $|\psi\rangle$ of length $\approx L$ on $[S]$, for $S \approx S'/L$

How:

1. Form the phase vector $|\psi'\rangle = |\psi_1, \psi_2\rangle$ with index set $[L_1] \times [L_2]$ and multipliers $b'(\vec{\jmath}) = b_1(j_1) + b_2(j_2)$.
2. Measure $|\psi'\rangle$ according to $q = \lfloor b'(\vec{\jmath})/S \rfloor$.
   All 'surviving' multipliers are in $[S]$, up to global phase.
3. Compute the subset $J \subseteq [L_1] \times [L_2]$ of $\vec{\jmath}$ that satisfy the above, reindex J to $[|J|]$, and output the resulting $|\psi\rangle$.

Analysis

- Phase vector $|\psi'\rangle$ has length $L_1 L_2 \approx L^2$, and the multipliers $b'(\vec{\jmath})$ are well distributed in $[2S']$.
- So, most size-S subintervals have $\approx L^2 \cdot S/(2S') \approx L$ multipliers.

(In practice, need some tricks to control the variance.)

Collimation Procedure

Given: two phase vectors $|\psi_i\rangle$ of length $L_i \approx L$ on $[S']$

Goal: one phase vector $|\psi\rangle$ of length $\approx L$ on $[S]$, for $S \approx S'/L$

How:

1. Form the phase vector $|\psi'\rangle = |\psi_1, \psi_2\rangle$ with index set $[L_1] \times [L_2]$ and multipliers $b'(\vec{\jmath}) = b_1(j_1) + b_2(j_2)$.
2. Measure $|\psi'\rangle$ according to $q = \lfloor b'(\vec{\jmath})/S \rfloor$. All 'surviving' multipliers are in $[S]$, up to global phase.
3. Compute the subset $J \subseteq [L_1] \times [L_2]$ of $\vec{\jmath}$ that satisfy the above, reindex J to $[|J|]$, and output the resulting $|\psi\rangle$.

Analysis

- Phase vector $|\psi'\rangle$ has length $L_1 L_2 \approx L^2$, and the multipliers $b'(\vec{\jmath})$ are well distributed in $[2S']$.
- So, most size-S subintervals have $\approx L^2 \cdot S/(2S') \approx L$ multipliers.

  (In practice, need some tricks to control the variance.)
- Step 3 requires $O(1)$ QRACM$[L]$ lookups and $\tilde{O}(L)$ classical work.

Post-Processing: Regularization and Measurement

- Collimation sieve yields a phase vector $|\psi\rangle$ on $[S]$ of length $L \approx S$.

Post-Processing: Regularization and Measurement

- Collimation sieve yields a phase vector $|\psi\rangle$ on $[S]$ of length $L \approx S$.
- Suppose $L = S$ and $b\colon [S] \to [S]$ is a bijection.

Post-Processing: Regularization and Measurement

- Collimation sieve yields a phase vector $|\psi\rangle$ on $[S]$ of length $L \approx S$.
- Suppose $L = S$ and $b\colon [S] \to [S]$ is a bijection. Can reindex $|\psi\rangle$ as

$$|\psi\rangle \propto \sum_{j \in [S]} \chi(j \cdot s/N)|j\rangle.$$

Post-Processing: Regularization and Measurement

- Collimation sieve yields a phase vector $|\psi\rangle$ on $[S]$ of length $L \approx S$.
- Suppose $L = S$ and $b \colon [S] \to [S]$ is a bijection. Can reindex $|\psi\rangle$ as

$$|\psi\rangle \propto \sum_{j \in [S]} \chi(j \cdot s/N) |j\rangle.$$

Its QFT_S is essentially the point function at $s \cdot S/N$. Measuring yields the $\log S$ most-significant bits of s, with large probability.

Post-Processing: Regularization and Measurement

- Collimation sieve yields a phase vector $|\psi\rangle$ on $[S]$ of length $L \approx S$.
- Suppose $L = S$ and $b\colon [S] \to [S]$ is a bijection. Can reindex $|\psi\rangle$ as

$$|\psi\rangle \propto \sum_{j \in [S]} \chi(j \cdot s/N)|j\rangle.$$

  Its QFT_S is essentially the point function at $s \cdot S/N$. Measuring yields the $\log S$ most-significant bits of s, with large probability.

- If $b\colon [L] \to [S]$ is not a bijection, measure to make it densely injective onto some $X \subseteq [S]$. Can then reindex as

$$|\tilde{\psi}\rangle \propto \sum_{j \in X} \chi(j \cdot s/N)|j\rangle.$$

Post-Processing: Regularization and Measurement

- Collimation sieve yields a phase vector $|\psi\rangle$ on $[S]$ of length $L \approx S$.
- Suppose $L = S$ and $b\colon [S] \to [S]$ is a bijection. Can reindex $|\psi\rangle$ as

$$|\psi\rangle \propto \sum_{j \in [S]} \chi(j \cdot s/N)|j\rangle.$$

  Its QFT_S is essentially the point function at $s \cdot S/N$. Measuring yields the $\log S$ most-significant bits of s, with large probability.

- If $b\colon [L] \to [S]$ is not a bijection, measure to make it densely injective onto some $X \subseteq [S]$. Can then reindex as

$$|\tilde{\psi}\rangle \propto \sum_{j \in X} \chi(j \cdot s/N)|j\rangle.$$

  This is a densely subsampled Fourier transform of a point function. Measuring its QFT yields almost $\log S$ bits of s.

Practical Issues

Issue 1: Lengths of collimated phase vectors are quite variable.
Too short and too long are both problems.

Practical Issues

Issue 1: Lengths of collimated phase vectors are quite variable.
Too short and too long are both problems.

Solution: Request lengths adaptively, and discard too-short vectors.

Practical Issues

Issue 1: Lengths of collimated phase vectors are quite variable.
Too short and too long are both problems.

Solution: Request lengths adaptively, and discard too-short vectors.
(Discarding 2% saves $\geq 2^{10}$ factor in longest vector.)

Practical Issues

Issue 1: Lengths of collimated phase vectors are quite variable.
Too short and too long are both problems.

Solution: Request lengths adaptively, and discard too-short vectors.
(Discarding 2% saves $\geq 2^{10}$ factor in longest vector.)

Issue 2: Measuring sieve output on $[S]$ yields $\approx \log S$ MSBs of secret.

Practical Issues

Issue 1: Lengths of collimated phase vectors are quite variable.
Too short and too long are both problems.

Solution: Request lengths adaptively, and discard too-short vectors.
(Discarding 2% saves $\geq 2^{10}$ factor in longest vector.)

Issue 2: Measuring sieve output on $[S]$ yields $\approx \log S$ MSBs of secret.

Solution: Sieve to 'scaled intervals' $S^i \cdot [S]$ for $i = 0, \ldots, \log_S(N) - 1$,
tensor results and measure to get entire secret.

Open Questions

- **Key Question:** what is the complexity of the requisite CSIDH oracle?

Open Questions

- **Key Question:** what is the complexity of the requisite CSIDH oracle?
- Existing estimates [BLMP'19] are for 'best conceivable' distributions; we need uniform distribution.

Open Questions

- **Key Question:** what is the complexity of the requisite CSIDH oracle?
- Existing estimates [BLMP'19] are for 'best conceivable' distributions; we need uniform distribution. Or do we?

Open Questions

- **Key Question:** what is the complexity of the requisite CSIDH oracle?
- Existing estimates [BLMP'19] are for 'best conceivable' distributions; we need uniform distribution. Or do we?
- We have many short relations in class group [BKV'19], enabling fast reduction of uniform distribution to exponent vectors with similar norm statistics as 'best conceivable'. Overall cost? Depth?

Open Questions

- **Key Question:** what is the complexity of the requisite CSIDH oracle?
- Existing estimates [BLMP'19] are for 'best conceivable' distributions; we need uniform distribution. Or do we?
- We have many short relations in class group [BKV'19], enabling fast reduction of uniform distribution to exponent vectors with similar norm statistics as 'best conceivable'. Overall cost? Depth?
- More direct constructions of quantum CSIDH circuits?

Open Questions

- **Key Question:** what is the complexity of the requisite CSIDH oracle?
- Existing estimates [BLMP'19] are for 'best conceivable' distributions; we need uniform distribution. Or do we?
- We have many short relations in class group [BKV'19], enabling fast reduction of uniform distribution to exponent vectors with similar norm statistics as 'best conceivable'. Overall cost? Depth?
- More direct constructions of quantum CSIDH circuits?
- Amortize the oracle computations? E.g., to get initial phase vectors?

Open Questions

- **Key Question:** what is the complexity of the requisite CSIDH oracle?
- Existing estimates [BLMP'19] are for 'best conceivable' distributions; we need uniform distribution. Or do we?
- We have many short relations in class group [BKV'19], enabling fast reduction of uniform distribution to exponent vectors with similar norm statistics as 'best conceivable'. Overall cost? Depth?
- More direct constructions of quantum CSIDH circuits?
- Amortize the oracle computations? E.g., to get initial phase vectors?
- **Question 2:** break CSIDH using partial information about secret?

Open Questions

- **Key Question:** what is the complexity of the requisite CSIDH oracle?
- Existing estimates [BLMP'19] are for 'best conceivable' distributions; we need uniform distribution. Or do we?
- We have many short relations in class group [BKV'19], enabling fast reduction of uniform distribution to exponent vectors with similar norm statistics as 'best conceivable'. Overall cost? Depth?
- More direct constructions of quantum CSIDH circuits?
- Amortize the oracle computations? E.g., to get initial phase vectors?
- **Question 2:** break CSIDH using partial information about secret?

Conclusions

1. Proposed CSIDH parameters have relatively little quantum security beyond the cost of quantum evaluation (on a uniform superposition).

2. CSIDH-512 key recovery costs, e.g., only $\approx 2^{16}$ evaluations using $\approx 2^{40}$ bits of quantum-accessible RAM (+ small other resources).

3. Assuming evaluation costs not much more than for the 'best case': CSIDH-512, -1024, and maybe even -1792 do not reach NIST level 1 quantum security.

Paper: ePrint 2019/725

Code: `https://github.com/cpeikert/CollimationSieve`