A FUSION ALGORITHM FOR SOLVING THE HIDDEN SHIFT PROBLEM IN FINITE ABELIAN GROUPS

Wouter Castryck, Ann Dooms, **Carlo Emerencia**, **Alexander Lemmens**

06/07/21

VRIJE UNIVERSITEIT BRUSSEL
KU LEUVEN

CONTENTS

1. The Hidden Shift Problem
    1. Formulation
    2. Importance in Cryptography
    3. Previous Algorithms

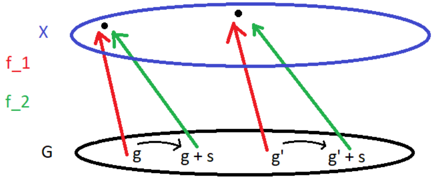
2. The Fusion Algorithm
    1. Torsion part
    2. Collimation Part
    3. Combined Algorithm

THE HIDDEN SHIFT PROBLEM

THE HIDDEN SHIFT PROBLEM

Definition

Let $(G, +)$ be an abelian group, X be a set and $f_1, f_2 : G \to X$ be injective functions for which there is a $s \in G$ such that $f_1(g) = f_2(g + s)$ for all $g \in G$. Find s.



EXAMPLE: SIMON´S PROBLEM

EXAMPLE: SIMON´S PROBLEM

- $G = \mathbb{Z}_2^m$

EXAMPLE: SIMON'S PROBLEM

- $G = \mathbb{Z}_2^m$
- $f_i : G \to X, f_1(x) = f_2(x + s)$

EXAMPLE: SIMON'S PROBLEM

- $G = \mathbb{Z}_2^m$
- $f_i : G \to X, f_1(x) = f_2(x+s)$
- Polynomial-time solution

EXAMPLE: SIMON'S PROBLEM

- $G = \mathbb{Z}_2^m$
- $f_i : G \to X$, $f_1(x) = f_2(x + s)$
- Polynomial-time solution
- $\chi_v(w) = (-1)^{\langle v,w \rangle}$

EXAMPLE: SIMON´S PROBLEM

- $G = \mathbb{Z}_2^m$
- $f_i : G \to X, f_1(x) = f_2(x+s)$
- Polynomial-time solution
- $\chi_v(w) = (-1)^{\langle v,w \rangle}$
- $\frac{1}{\sqrt{2}}(|0\rangle + \chi_v(s)|1\rangle) \overset{H}{\mapsto} \frac{1}{2}((1+\chi_v(s))|0\rangle + (1-\chi_v(s))|1\rangle)$

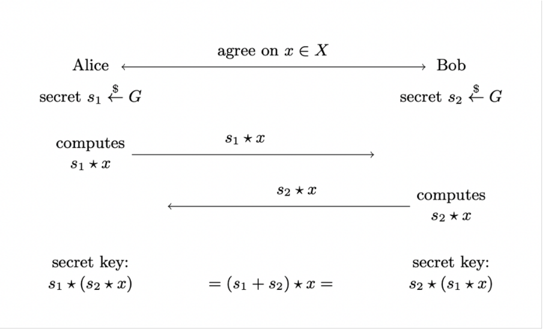
EXAMPLE: SIMON'S PROBLEM

- $G = \mathbb{Z}_2^m$
- $f_i : G \to X, f_1(x) = f_2(x+s)$
- Polynomial-time solution
- $\chi_v(w) = (-1)^{\langle v,w \rangle}$
- $\frac{1}{\sqrt{2}}(|0\rangle + \chi_v(s)|1\rangle) \overset{H}{\mapsto} \frac{1}{2}((1+\chi_v(s))|0\rangle + (1-\chi_v(s))|1\rangle)$
- System of equations $\to$ recover s

DIFFIE HELLMAN KEY EXCHANGE

DIFFIE HELLMAN KEY EXCHANGE

Abelian group G acts on set X



HARD HOMOGENEOUS SPACES

HARD HOMOGENEOUS SPACES

- G abelian acts strictly transitively on set X

HARD HOMOGENEOUS SPACES

- G abelian acts strictly transitively on set X
  - $0 \star x = x$

HARD HOMOGENEOUS SPACES

- G abelian acts strictly transitively on set X
  - $0 \star x = x$
  - $(s_1 + s_2) \star x = s_1 \star (s_2 \star x)$

HARD HOMOGENEOUS SPACES

- G abelian acts strictly transitively on set X
  - $0 \star x = x$
  - $(s_1 + s_2) \star x = s_1 \star (s_2 \star x)$
- **Vectorisation Problem**: Given $x, s \star x$, extract s.

HARD HOMOGENEOUS SPACES

- G abelian acts strictly transitively on set X
  - $0 \star x = x$
  - $(s_1 + s_2) \star x = s_1 \star (s_2 \star x)$
- **Vectorisation Problem**: Given $x, s \star x$, extract s.
- **Parallelization Problem**: Given $x, s_1 \star x, s_2 \star x$, compute $(s_1 + s_2) \star x$

HARD HOMOGENEOUS SPACES

- G abelian acts strictly transitively on set X
  - $0 \star x = x$
  - $(s_1 + s_2) \star x = s_1 \star (s_2 \star x)$
- **Vectorisation Problem**: Given $x, s \star x$, extract s.
- **Parallelization Problem**: Given $x, s_1 \star x, s_2 \star x$, compute $(s_1 + s_2) \star x$
- As Hidden Shift Instance:

HARD HOMOGENEOUS SPACES

- G abelian acts strictly transitively on set X
  - $0 \star x = x$
  - $(s_1 + s_2) \star x = s_1 \star (s_2 \star x)$
- **Vectorisation Problem**: Given $x, s \star x$, extract s.
- **Parallelization Problem**: Given $x, s_1 \star x, s_2 \star x$, compute $(s_1 + s_2) \star x$
- As Hidden Shift Instance:
  - $f_1 : G \to X : a \mapsto a \star (s \star x)$

HARD HOMOGENEOUS SPACES

- G abelian acts strictly transitively on set X
  - $0 \star x = x$
  - $(s_1 + s_2) \star x = s_1 \star (s_2 \star x)$
- **Vectorisation Problem**: Given $x, s \star x$, extract s.
- **Parallelization Problem**: Given $x, s_1 \star x, s_2 \star x$, compute $(s_1 + s_2) \star x$
- As Hidden Shift Instance:
  - $f_1 : G \to X : a \mapsto a \star (s \star x)$
  - $f_2 : G \to X : a \mapsto a \star x$

CSIDH

CSIDH

- $G = \mathsf{Cl}(\mathscr{O})$

CSIDH

- $G = \mathsf{Cl}(\mathscr{O})$
- X set of supersingular elliptic curves $E/\mathbb{F}_p$ (up to isomorphism) such that $\mathrm{End}_{\mathbb{F}_p}(E) \cong \mathscr{O}$.

CSIDH

- $G = \mathrm{Cl}(\mathscr{O})$
- X set of supersingular elliptic curves $E/\mathbb{F}_p$ (up to isomorphism) such that $\mathrm{End}_{\mathbb{F}_p}(E) \cong \mathscr{O}$.
- G acts on set X via isogeny

STANDARD METHOD

- Apply f_1, f_2 to a superposition of all group elements
- measure to obtain

$$\frac{1}{\sqrt{2}}(|g\rangle + |g+s\rangle)$$

- Perform Abelian Quantum Fourier transform to obtain

$$\frac{1}{\sqrt{2}}(|0\rangle + \chi(s)|1\rangle)$$

- $\chi : G \longrightarrow \mathbb{C}^\times$ is a character

PREVIOUS ALGORITHMS

PREVIOUS ALGORITHMS

- All based on creating length 2 phase vectors: $\varphi(\chi) = \frac{1}{\sqrt{2}}(|0\rangle + \chi(s)|1\rangle)$

PREVIOUS ALGORITHMS

- All based on creating length 2 phase vectors: $\varphi(\chi) = \frac{1}{\sqrt{2}}(|0\rangle + \chi(s)|1\rangle)$
  - Kuperberg: combine "bad" phase vectors to make "better" ones

PREVIOUS ALGORITHMS

- All based on creating length 2 phase vectors: $\varphi(\chi) = \frac{1}{\sqrt{2}}(|0\rangle + \chi(s)|1\rangle)$
  - Kuperberg: combine "bad" phase vectors to make "better" ones
  - Regev: improvement combining more phase vectors and using more general measurements, requiring less memory

PREVIOUS ALGORITHMS

- All based on creating length 2 phase vectors: $\varphi(\chi) = \frac{1}{\sqrt{2}}(|0\rangle + \chi(s)|1\rangle)$
  - Kuperberg: combine "bad" phase vectors to make "better" ones
  - Regev: improvement combining more phase vectors and using more general measurements, requiring less memory
  - Kuperberg: collimation sieve using QRACM

PREVIOUS ALGORITHMS

- All based on creating length 2 phase vectors: $\varphi(\chi) = \frac{1}{\sqrt{2}}(|0\rangle + \chi(s)|1\rangle)$
  - Kuperberg: combine “bad” phase vectors to make “better” ones
  - Regev: improvement combining more phase vectors and using more general measurements, requiring less memory
  - Kuperberg: collimation sieve using QRACM
  - Peikert: adaptation of collimation sieve on most significant bit

TORSION ALGORITHM

TORSION ALGORITHM

- G finite abelian with $p2^tG = 0$

TORSION ALGORITHM

- G finite abelian with $p2^tG = 0$
- Example: $G = \mathbb{Z}_3^2 \times \mathbb{Z}_4 \times \mathbb{Z}_2$

TORSION ALGORITHM

- G finite abelian with $p2^tG = 0$
- Example: $G = \mathbb{Z}_3^2 \times \mathbb{Z}_4 \times \mathbb{Z}_2$
- $\chi_{(a_1,a_2,a_3,a_4)}(x_1, x_2, x_3, x_4) = e^{2\pi \boldsymbol{i}(\frac{a_1x_1+a_2x_2}{3}+\frac{a_3x_3}{4}+\frac{a_4x_4}{2})}$

TORSION ALGORITHM

- G finite abelian with $p2^tG = 0$
- Example: $G = \mathbb{Z}_3^2 \times \mathbb{Z}_4 \times \mathbb{Z}_2$
- $\chi_{(a_1,a_2,a_3,a_4)}(x_1, x_2, x_3, x_4) = e^{2\pi \boldsymbol{i}(\frac{a_1x_1+a_2x_2}{3}+\frac{a_3x_3}{4}+\frac{a_4x_4}{2})}$
- Goal: combine characters to get $(a_1, a_2, 0, 0)$

TORSION ALGORITHM

TORSION ALGORITHM

- Let $s = (s_1, s_2, s_3, s_4), \chi = \chi_{(1,2,0,0)}$

TORSION ALGORITHM

- Let $s = (s_1, s_2, s_3, s_4), \chi = \chi_{(1,2,0,0)}$
- We have $\frac{1}{\sqrt{2}}(|0\rangle + \chi(s)|1\rangle)$

TORSION ALGORITHM

- Let $s = (s_1, s_2, s_3, s_4), \chi = \chi_{(1,2,0,0)}$
- We have $\frac{1}{\sqrt{2}}(|0\rangle + \chi(s)|1\rangle)$
- Apply Hadamard: $\frac{1}{2}((1 + \chi(s))|0\rangle + (1 - \chi(s))|1\rangle)$

TORSION ALGORITHM

- Let $s = (s_1, s_2, s_3, s_4), \chi = \chi_{(1,2,0,0)}$
- We have $\frac{1}{\sqrt{2}}(|0\rangle + \chi(s)|1\rangle)$
- Apply Hadamard: $\frac{1}{2}((1 + \chi(s))|0\rangle + (1 - \chi(s))|1\rangle)$
- If we measure 1: $\chi(s) \neq 1$

TORSION ALGORITHM

- Let $s = (s_1, s_2, s_3, s_4), \chi = \chi_{(1,2,0,0)}$
- We have $\frac{1}{\sqrt{2}}(|0\rangle + \chi(s)|1\rangle)$
- Apply Hadamard: $\frac{1}{2}((1 + \chi(s))|0\rangle + (1 - \chi(s))|1\rangle)$
- If we measure 1: $\chi(s) \neq 1 \Rightarrow (s_1 + 2s_2)^2 \equiv 1$ (mod 3)

TORSION ALGORITHM

- Let $s = (s_1, s_2, s_3, s_4), \chi = \chi_{(1,2,0,0)}$
- We have $\frac{1}{\sqrt{2}}(|0\rangle + \chi(s)|1\rangle)$
- Apply Hadamard: $\frac{1}{2}((1 + \chi(s))|0\rangle + (1 - \chi(s))|1\rangle)$
- If we measure 1: $\chi(s) \neq 1 \Rightarrow (s_1 + 2s_2)^2 \equiv 1$ (mod 3)
- After enough equations, we can recover s_1, s_2

TORSION ALGORITHM

TORSION ALGORITHM

- Reduction to $\mathbb{Z}_4 \times \mathbb{Z}_2$ now possible

TORSION ALGORITHM

- Reduction to $\mathbb{Z}_4 \times \mathbb{Z}_2$ now possible
- $\chi_{(a_1,a_2)}(x_1, x_2) = e^{2\pi \boldsymbol{i}(\frac{a_1 x_1}{4} + \frac{a_2 x_2}{2})}$

TORSION ALGORITHM

- Reduction to $\mathbb{Z}_4 \times \mathbb{Z}_2$ now possible
- $\chi_{(a_1,a_2)}(x_1, x_2) = e^{2\pi \boldsymbol{i}(\frac{a_1 x_1}{4} + \frac{a_2 x_2}{2})}$
- Goal: $\chi_a^2 = 1$ (i.e. if $a \in \{(2,0),(0,1),(2,1)\}$)

TORSION ALGORITHM

- Reduction to $\mathbb{Z}_4 \times \mathbb{Z}_2$ now possible
- $\chi_{(a_1,a_2)}(x_1, x_2) = e^{2\pi i(\frac{a_1 x_1}{4} + \frac{a_2 x_2}{2})}$
- Goal: $\chi_a^2 = 1$ (i.e. if $a \in \{(2,0), (0,1), (2,1)\}$)
- Use method Simon's problem to recover s_3, s_4

OUR COLLIMATION

OUR COLLIMATION

- long phase vectors $\Psi = \frac{1}{\sqrt{L}}(\chi_1(s)|\chi_1\rangle + \ldots + \chi_L(s)|\chi_L\rangle)$.

OUR COLLIMATION

- long phase vectors $\Psi = \frac{1}{\sqrt{L}}(\chi_1(s)|\chi_1\rangle + \ldots + \chi_L(s)|\chi_L\rangle)$.
- $\chi_j : G \longrightarrow \mathbb{C}^\times$ are characters

OUR COLLIMATION

- long phase vectors $\Psi = \frac{1}{\sqrt{L}}(\chi_1(s)|\chi_1\rangle + \ldots + \chi_L(s)|\chi_L\rangle)$.
- $\chi_j : G \longrightarrow \mathbb{C}^\times$ are characters
- Force the characters into smaller and smaller regions of $G^\vee$.

OUR COLLIMATION

- long phase vectors $\Psi = \frac{1}{\sqrt{L}}(\chi_1(s)|\chi_1\rangle + \ldots + \chi_L(s)|\chi_L\rangle)$.
- $\chi_j : G \longrightarrow \mathbb{C}^\times$ are characters
- Force the characters into smaller and smaller regions of $G^\vee$.
- Create phase vector of density > 1 by collimation

OUR COLLIMATION

- long phase vectors $\Psi = \frac{1}{\sqrt{L}}(\chi_1(s)|\chi_1\rangle + \ldots + \chi_L(s)|\chi_L\rangle)$.
- $\chi_j : G \longrightarrow \mathbb{C}^\times$ are characters
- Force the characters into smaller and smaller regions of $G^\vee$.
- Create phase vector of density > 1 by collimation
- thin out to obtain regular phase vector supported on a subgroup H of $G^\vee$.
- Apply Fourier transform to retrieve s mod $\ker H$.
- Reduces hidden shift problem to $\ker H$.

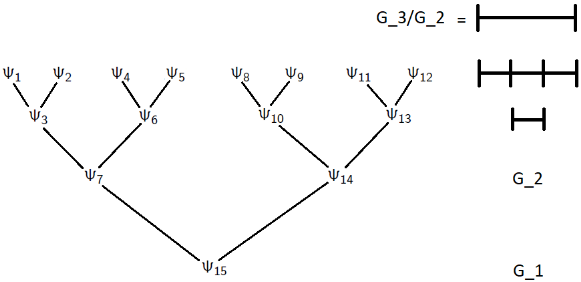
COLLIMATION EXAMPLE

COLLIMATION EXAMPLE

subgroup chain $\{1\} = G_0 \leq G_1 \leq G_2 \leq G_3 = G^\vee$

COLLIMATION EXAMPLE

subgroup chain $\{1\} = G_0 \leq G_1 \leq G_2 \leq G_3 = G^\vee$



COMBINED ALGORITHM

COMBINED ALGORITHM

- G finite abelian with subgroup chain $\{1\} = G_0 \leq G_1 = G^\vee[p2^t] \leq \cdots \leq G_M = G^\vee$

COMBINED ALGORITHM

- G finite abelian with subgroup chain $\{1\} = G_0 \leq G_1 = G^\vee[p2^t] \leq \cdots \leq G_M = G^\vee$
- Apply collimation until ψ is obtained with support in G_1

COMBINED ALGORITHM

- G finite abelian with subgroup chain $\{1\} = G_0 \leq G_1 = G^\vee[p2^t] \leq \cdots \leq G_M = G^\vee$
- Apply collimation until ψ is obtained with support in G_1
- Apply **Torsion Algorithm** to retrieve s mod $p2^tG$

COMBINED ALGORITHM

- G finite abelian with subgroup chain $\{1\} = G_0 \leq G_1 = G^{\vee}[p2^t] \leq \cdots \leq G_M = G^{\vee}$
- Apply collimation until ψ is obtained with support in G_1
- Apply **Torsion Algorithm** to retrieve s mod $p2^tG$
- Apply **Collimation Algorithm** to retrieve s entirely

CONCLUSION

CONCLUSION

- Combination of polynomial-time Friedl et al. with sub-exponential Kuperberg/Peikert

CONCLUSION

- Combination of polynomial-time Friedl et al. with sub-exponential Kuperberg/Peikert
- Works for any Abelian group G with complexity poly$(\log(|G|)) \cdot 2^{O(\sqrt{\log(|p2^t G|)})}$

CONCLUSION

- Combination of polynomial-time Friedl et al. with sub-exponential Kuperberg/Peikert
- Works for any Abelian group G with complexity poly$(\log(|G|)) \cdot 2^{O(\sqrt{\log(|p2^tG|)})}$
- Consequences for cryptosystems based on hard homogeneous spaces

CONCLUSION

- Combination of polynomial-time Friedl et al. with sub-exponential Kuperberg/Peikert
- Works for any Abelian group G with complexity poly$(\log(|G|)) \cdot 2^{O(\sqrt{\log(|p2^tG|)})}$
- Consequences for cryptosystems based on hard homogeneous spaces
  - Can the complexity be improved?

CONCLUSION

- Combination of polynomial-time Friedl et al. with sub-exponential Kuperberg/Peikert
- Works for any Abelian group G with complexity poly$(\log(|G|)) \cdot 2^{O(\sqrt{\log(|p2^t G|)})}$
- Consequences for cryptosystems based on hard homogeneous spaces
  - Can the complexity be improved?
  - Can this be generalized to other torsion, while being kept memory-friendly?

THE END

Thank you for your attention