Quantum LLL
with an Application to Mersenne Number Cryptosystems

Marcel Tiepelt\textsuperscript{1} Alan Szepieniec\textsuperscript{2}

\textsuperscript{1}Karlsruhe Institute of Technology
\textsuperscript{2}Nervos Foundation

LatinCrypt 2019
Santiago de Chile, Oct. 2-4
Overview

- Quantum circuit representation of LLL
  - for (textbook) rational numbers
  - for floating-point approximation

- Resource estimates of (sub)circuits, in Toffoli-gates

- Focus on qubits count
Why quantum translation of LLL?

Consider LLL as a subroutine, e.g., SVP oracle in cryptanalysis

- Assume 256 bits of classical security, for $O(2^{256})$ expected number of oracle calls

Quantumly: 128 bits of security, Groverization promises improvement to $O(2^{128})$ → Requires efficient translation of LLL into quantum setting!

But: translation of (text-book) LLL results in large overhead w.r.t. the number of qubits!

Does Grover with a QLLL give us the desired improvement?
Why quantum translation of LLL?

Consider LLL as a subroutine, e.g., SVP oracle in cryptanalysis

- Assume 256 bits of classical security, for $O(2^{256})$ expected number of oracle calls

- Quantumly: 128 bits of security, Groverization promises improvement to $O(2^{128})$
  \[\rightarrow\] Requires efficient translation of LLL into quantum setting!
Why quantum translation of LLL?

Consider LLL as a subroutine, e.g., SVP oracle in cryptanalysis

- Assume 256 bits of classical security, for $O(2^{256})$ expected number of oracle calls

- Quantumly: 128 bits of security, Groverization promises improvement to $O(2^{128})$
  \[\rightarrow\] Requires efficient translation of LLL into quantum setting!

- **But**: translation of (text-book) LLL results in large overhead w.r.t. the number of qubits!
Why quantum translation of LLL?

Consider LLL as a subroutine, e.g., SVP oracle in cryptanalysis

- Assume 256 bits of classical security, for $O(2^{256})$ expected number of oracle calls

- Quantumly: 128 bits of security, Groverization promises improvement to $O(2^{128})$
  \[ \rightarrow \text{Requires efficient translation of LLL into quantum setting!} \]

- **But**: translation of (text-book) LLL results in large overhead w.r.t. the number of qubits!

Does Grover with a QLLL give us the desired improvement?
(Classical) LLL

1: **Input:** Basis $B = (b_1, b_2, ..., b_r)$
2: **Output:** Reduced Basis $\hat{B}$
3: $B^*, M \leftarrow \text{GSO}(B)$
4: $k \leftarrow 2$
5: **while** $k \leq r$ **do**
6:    Size-reduce($b_k, b_{k-1}$)
7:    **if** Lovász condition holds on $b_k, b_{k-1}$ **then**
8:        Size-reduce($b_k, \{b_j\}_{0 \leq j \leq k-1}$), update $M$
9:        $k++$
10:    **else**
11:        swap $b_k, b_{k-1}$, update $M$
12:        $k := \max(2, k - 1)$
13:    **end if**
14: **end while**
Variants


“Best” variant: $L^2$ Nguyen and Stehlé [3]

(many more)
Quantum LLL Setup

Registers

$|B\rangle$ Basis representing a superposition of integer lattices

$|M(i)\rangle$ transformation $M$ in iteration $i$ s.t.: $B = MB^*$

$|K\rangle, |cntl\rangle$ counters, controls
Quantum LLL Setup

Registers

$|B\rangle$ Basis representing a superposition of integer lattices

$|M^{(i)}\rangle$ transformation $M$ in iteration $i$ s.t.: $B = MB^*$

$|K\rangle, |cntl\rangle$ counters, controls

Operations

*Arithmetic* in $\mathbb{Q}$ or $\mathbb{R}$, *vector operations* in $\mathbb{Z}$

*misc* compare, round, $\max(x, y)$, ...
Quantum LLL Setup

Registers

$|B\rangle$ Basis representing a superposition of integer lattices

$|M^{(i)}\rangle$ transformation $M$ in iteration $i$ s.t.: $B = MB^*$

$|K\rangle, |cntl\rangle$ counters, controls

Operations

Arithmetic in $\mathbb{Q}$ or $\mathbb{R}$, vector operations in $\mathbb{Z}$

misc compare, round, max($x, y$), ...

Notations

function $f(X)$

uncompute (run circuit backwards) $(f(X))^{-1}$
Quantum LLL

|L⟩ | QGSO |
|B⟩ | size-reduce: |bk⟩, |bk⟩-1⟩ |
|M⟩ | Lovász |
|Lov⟩ |
|K⟩ | 0 ≤ |K⟩ ≤ r |
|ctl1⟩ |
|ctl2⟩ |
|J⟩ |

\[
\text{rank}(\mathcal{L}) \text{ cycles}
\]

\[
\text{bound}(K) \text{ cycles}
\]

\[
\text{branch: size-reduce}
\]

\[
\text{branch: swap}
\]

\[
0 \leq |J⟩ \leq |K⟩ - 2
\]

\[
(0 \leq |J⟩ \leq |K⟩ - 2)^{-1}
\]

\[
0 \leq |J⟩ \leq |K⟩ - 2
\]

\[
(0 \leq |J⟩ \leq |K⟩ - 2)^{-1}
\]

\[
\max(2, |K⟩ - 1)
\]

|L⟩ |
|B⟩ |
|M⟩ |
|Lov⟩ |
|K⟩ |
|ctl1⟩ |
|ctl2⟩ |
|J⟩ |
Quantum LLL

\[ |L\rangle \quad |B\rangle \quad |M\rangle \quad |Lov\rangle \quad |K\rangle \quad |ctl_1\rangle \quad |ctl_2\rangle \quad |J\rangle \]

**QGSO**
- size-reduce: \( |b_K\rangle, |b_{K-1}\rangle \)
- Lovász

branch: size-reduce
- branch: swap

\[ 0 \leq |K| \leq r \]

\[ 0 \leq |J| \leq |K| - 2 \]

\[ 0 \leq |J| \leq |K| - 2 \]

\[ 0 \leq |J| \leq |K| - 2 \]

\[ 0 \leq |J| \leq |K| - 2 \]

\[ \max(2, |K| - 1) \]

\[ \text{rank}(\mathcal{L}) \text{ cycles} \]

\[ \text{bound}(K) \text{ cycles} \]
Pitfall I: unbounded loops

Classical
- Apply operation until loop terminates
Pitfall I: unbounded loops

Quantum

- Apply as often as necessary, but not *too* often

Classical

- Apply operation until loop terminates
Pitfall I: unbounded loops

Quantum

- Apply as often as necessary, but not \textit{too} often

Classical

- Apply operation until loop terminates

\begin{equation}
\text{Loop } k := 2; \text{ while}(k \leq r);
\end{equation}

\begin{align*}
|K\rangle &\geq 2 \\
|K\rangle &\leq r \\
|K\rangle &\leq r^{-1} \\
((|K\rangle \geq 2)^{-1} &\pm 1 \\
\text{bound}(K) \text{ cycles}
\end{align*}
Pitfall I: unbounded loops

Quantum

- Apply as often as necessary, but not *too* often

Classical

- Apply operation until loop terminates

Loop $k := 2; \text{while}(k \leq r);$;

$|K\rangle$  
$|\text{cntl}_1\rangle$  
$|\text{cntl}_2\rangle$  
$|\psi\rangle$

Apply Task

bound($K$) cycles

Quantum: worst-case running time for all (unbounded) loops
Pitfall Part II: size-reduction cleanup

Size reduction: $b_i \xrightarrow{\text{reduce by } b_j} \hat{b}_i$
Update $M$ s.t. $\hat{B} = MB^*$

Classical
Pitfall Part II: size-reduction cleanup

Size reduction: $b_i \xrightarrow{reduce \ by \ b_j} \hat{b}_i$
Update $M$ s.t. $\hat{B} = M\hat{B}^*$

Classical

\[
\left\lfloor m_{ij} \right\rfloor \leftarrow \text{round}(m_{ij}) \\
\hat{b}_i \leftarrow b_i - \left\lfloor m_{ij} \right\rfloor b_j \\
\hat{m}_{ij} \leftarrow m_{ij} - \left\lfloor m_{ij} \right\rfloor \\
\text{free}(\left\lfloor m_{ij} \right\rfloor), \text{free}(b_i), \text{free}(m_{ij})
\]
Pitfall Part II: size-reduction cleanup

Size reduction: \( b_i \xrightarrow{\text{reduce by } b_j} \hat{b}_i \)

Update \( M \) s.t. \( \hat{B} = M\hat{B}^* \)

**Classical**

\[
\begin{align*}
\lceil m_{ij} \rceil & \leftarrow \text{round}(m_{ij}) \\
\hat{b}_i & \leftarrow b_i - \lceil m_{ij} \rceil b_j \\
\hat{m}_{ij} & \leftarrow m_{ij} - \lceil m_{ij} \rceil \\
\text{free}(\lceil m_{ij} \rceil), \text{free}(b_i), \text{free}(m_{ij})
\end{align*}
\]

\( m_{ij}, b_i \) can not be recomputed from \( \hat{m}_{ij}, \hat{b}_{ij} \)

\( \Rightarrow \) information about *larger* basis is lost
Pitfall Part II: size-reduction cleanup

Quantum

\[ |m_{ij}\rangle, |b_i\rangle \text{ can not be recomputed from } |\hat{m}_{ij}\rangle, |\hat{b}_{ij}\rangle \]
\[ \Rightarrow |b_i\rangle, |m_{ij}\rangle \text{ or } |\lfloor m_{ij}\rfloor\rangle \text{ need to be preserved for reversibility} \]
Pitfall Part II: size-reduction cleanup

Quantum

\[ m_{ij} \rightarrow \left\lceil m_{ij} \right\rceil \rightarrow \hat{m}_{ij} \]

\[ |m_{ij}\rangle, |b_i\rangle \text{ can not be recomputed from } |\hat{m}_{ij}\rangle, |\hat{b}_{ij}\rangle \]
\[ \Rightarrow |b_i\rangle, |m_{ij}\rangle \text{ or } |\left\lceil m_{ij} \right\rceil\rangle \text{ need to be preserved for reversibility} \]

Quantum: need fresh memory in every size-reduction

(similar issues arises from divisions/ preserving the remainder for fp-numbers)
Impact?

Size reduction is conditionally applied to all vectors of $|M(i)\rangle$

Reversible size-reduction:

$|M(i)\rangle|B\rangle|0\rangle \Rightarrow |M(i)\rangle|B\rangle|M(i+1)\rangle$

Size reduction is conditionally applied to all vectors of $|M(i)\rangle$.

Reversible size-reduction:

$|M(i)\rangle|B\rangle|0\rangle \Rightarrow |M(i)\rangle|B\rangle|M(i+1)\rangle$
Impact?

Size reduction is conditionally applied to all vectors of $|M^{(i)}\rangle$

Reversible size-reduction:

$|M^{(i)}\rangle |B\rangle |0\rangle \Rightarrow |M^{(i)}\rangle |B\rangle |M^{(i+1)}\rangle$

How many qubits does this require?

$\text{sizeOf}(M)$ qubits for each reduction

$\text{bound}(K)$ many iterations

$\rightarrow \text{bound}(K) \times \text{sizeOf}(M)$

Bad if $\text{bound}(K)$ is large
Impact?

Size reduction is conditionally applied to all vectors of $|M^{(i)}\rangle$

Reversible size-reduction:

$|M^{(i)}\rangle|B\rangle|0\rangle \Rightarrow |M^{(i)}\rangle|B\rangle|M^{(i+1)}\rangle$

How many qubits does this require?

- $\text{sizeOf}(M)$ qubits for each reduction
- $\text{bound}(K)$ many iterations
- $\Rightarrow \text{bound}(K) \times \text{sizeOf}(M)$
Impact?

Size reduction is conditionally applied to all vectors of $|M^{(i)}\rangle$

Reversible size-reduction:

$|M^{(i)}\rangle|B\rangle|0\rangle \Rightarrow |M^{(i)}\rangle|B\rangle|M^{(i+1)}\rangle$

How many qubits does this require?

- sizeOf(M) qubits for each reduction
- bound(K) many iterations
  $\rightarrow$ bound(K) $\times$ sizeOf(M)

Bad if bound(K) is large

Marcel Tiepelt, Alan Szepieniec – Quantum LLL
Can we do better?

\[ |M^{(0)}\rangle|0\rangle...|0\rangle \]

size-reduce

\[ |M^{(0)}\rangle|M^{(1)}\rangle|0\rangle...|0\rangle \]

... \[ |M^{(0)}\rangle|M^{(1)}\rangle...|M^{(j-2)}\rangle|M^{(j-1)}\rangle|0\rangle \]

size-reduce

\[ |M^{(0)}\rangle|M^{(1)}\rangle...|M^{(j-2)}\rangle|M^{(j-1)}\rangle|M^{(j)}\rangle \]
Can we do better?

\[ |\mathcal{M}(0)\rangle|0\rangle \ldots |0\rangle \]

size-reduce

\[ |\mathcal{M}(0)\rangle|\mathcal{M}(1)\rangle|0\rangle \ldots |0\rangle \]

\[ \vdots \]

\[ |\mathcal{M}(0)\rangle|\mathcal{M}(1)\rangle \ldots |\mathcal{M}(j-2)\rangle|\mathcal{M}(j-1)\rangle|0\rangle \]

size-reduce

\[ |\mathcal{M}(0)\rangle|\mathcal{M}(1)\rangle \ldots |\mathcal{M}(j-2)\rangle|\mathcal{M}(j-1)\rangle|\mathcal{M}(j)\rangle \]

\[ |\mathcal{M}(0)\rangle|\mathcal{M}(1)\rangle \ldots |0\rangle \ldots |0\rangle |\mathcal{M}(j)\rangle \]

\[ (size-reduce)^{-1} \]

\[ |\mathcal{M}(0)\rangle|\mathcal{M}(1)\rangle \ldots |\mathcal{M}(j-2)\rangle|0\rangle |\mathcal{M}(j)\rangle \]

\[ (size-reduce)^{-1} \]

Requires at most: \( j \times \text{sizeOf}(\mathcal{M}) \) qubits
Can we do better?

\[ |M^{(0)}\rangle |0\rangle \cdots |0\rangle \]

size-reduce

\[ |M^{(0)}\rangle |M^{(1)}\rangle |0\rangle \cdots |0\rangle \]

\[ |M^{(0)}\rangle |M^{(1)}\rangle \cdots |M^{(j-2)}\rangle |M^{(j-1)}\rangle |0\rangle \]

size-reduce

\[ |M^{(0)}\rangle |M^{(1)}\rangle \cdots |M^{(j-2)}\rangle |M^{(j-1)}\rangle |M^{(j)}\rangle \]

→ Requires at most: \( j \times \text{sizeOf}(M) \) qubits
Impact?

\[ |M^{(0)}\rangle \]
Impact?

\[ |M^{(0)}\rangle \rightarrow |M^{(0)}\rangle |M^{(j)}\rangle \]
Impact?

\[ |M^{(0)}\rangle \]
\[ \rightarrow |M^{(0)}\rangle |M(j)\rangle \]
\[ \rightarrow \ldots \]
\[ \rightarrow |M^{(0)}\rangle |M(j)\rangle \ldots |M^{(\text{bound}(K))}\rangle \]

(Optimal for \( j = \sqrt{\text{bound}(K)} \))
Impact?

\[ |M^{(0)}\rangle \rightarrow |M^{(0)}\rangle|M^{(j)}\rangle \rightarrow \ldots \rightarrow |M^{(0)}\rangle|M^{(j)}\rangle\ldots|M^{(\text{bound}(K))}\rangle \]

(Optimal for \( j = \sqrt{\text{bound}(K)} \))

**Trade-off:**
(Maximal) number of qubits: \( \sqrt{\text{bound}(K)} \times \text{sizeOf}(M) \)
For \# additional iterations: \( \text{bound}(K) \)
Resource Estimate

- Given basis $B := (b_1, b_2, ..., b_r)$, $b_i \in \mathbb{Z}^d$
- (qu)bit-length $n$ in $b_i$
- $bound(K) := r^2 \log \hat{B}$, $\hat{B} :=$ bounds norm of initial basis
Resource Estimate

- Given basis $B := (b_1, b_2, \ldots, b_r), b_i \in \mathbb{Z}^d$
- (qu)bit-length $n$ in $b_i$
- $\text{bound}(K) := r^2 \log \hat{B}, \hat{B} := \text{bounds norm of initial basis}$

<table>
<thead>
<tr>
<th>#Toffoli</th>
<th>#Qubits</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O \left( 2 \log \hat{B}(r^3 d + r^4) \left( \frac{n^2}{\log n} + 2n \right) \right)$</td>
<td>$\max(d, r) \cdot n$</td>
</tr>
</tbody>
</table>
Resource Estimate

- Given basis $B := (b_1, b_2, ..., b_r)$, $b_i \in \mathbb{Z}^d$
- (qu)bit-length $n$ in $b_i$
- $bound(K) := r^2 \log \hat{B}$, $\hat{B} :=$ bounds norm of initial basis

<table>
<thead>
<tr>
<th>Method</th>
<th>#Toffoli</th>
<th>#Qubits</th>
</tr>
</thead>
<tbody>
<tr>
<td>QLLL</td>
<td>$O \left( 2 \log \hat{B}(r^3 d + r^4) \left( \frac{n^2}{\log n} + 2n \right) \right)$</td>
<td>$\max(d, r) \cdot n$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>#Qubits_M</th>
</tr>
</thead>
<tbody>
<tr>
<td>text-book</td>
<td>$O \left( r^3 d \log \hat{B}(\log \hat{B})^{\frac{1}{2}} \right)$</td>
</tr>
<tr>
<td>Schnorr</td>
<td>$O \left( r^2 d \log \hat{B}(\log \hat{B})^{\frac{1}{2}} \right)$</td>
</tr>
<tr>
<td>$L^2$</td>
<td>$O \left( r(\log \hat{B})^{\frac{1}{2}} (1.6d + o(d)) \right)$</td>
</tr>
</tbody>
</table>
Problem

- Given $a, b \leftarrow \mathbb{Z}_p$ with low Hamming weight, $G \leftarrow \mathbb{Z}_p$
- Given $pk := aG + b = H \mod p$, Find $a, b$
**Application: Groverization of Attack on Mersenne number cryptosystems**

**Problem**

- Given $a, b \leftarrow \mathbb{Z}_p$ with low Hamming weight, $G \leftarrow \mathbb{Z}_p$
- Given $pk := aG + b = H \mod p$, Find $a, b$

(Best) approach due to Beunardeau et al. [1] applies lattice reduction after partitioning sparse $a, b$, such that each partition represents small number
Resource Estimate of Grover Oracle

Instantiation for 256-bits of security with $n = 756839$ the QLLL oracle requires:

- #Toffoli: $\approx 2^{85}$
- #Qubits: $\approx 2^{52}$
- Schnorr: $\approx 2^{65}$
- $L_2$: $\approx 2^{55}$
Resource Estimate of Grover Oracle

Instantiation for 256-bits of security with $n = 756839$ the QLLL oracle requires:

<table>
<thead>
<tr>
<th>Method</th>
<th>#Toffoli</th>
<th>#Qubits</th>
</tr>
</thead>
<tbody>
<tr>
<td>text-book</td>
<td>$\approx 2^{85}$</td>
<td>$\approx 2^{52}$</td>
</tr>
<tr>
<td>Schnorr</td>
<td>$\approx 2^{65}$</td>
<td>$\approx 2^{44}$</td>
</tr>
<tr>
<td>$L^2$</td>
<td>$\approx 2^{55}$</td>
<td>$\approx 2^{33}$</td>
</tr>
</tbody>
</table>
Conclusions

Quantum vs. Classical

- Apply size-reduction **and** swap conditionally
- Average is worst-case, domain knowledge gives significant improvements!
- Split LLL reduction to improve qubit overhead
  \[ O \left( r^3 d \log \hat{B} (\log \hat{B})^{\frac{1}{2}} \right) \]

- Apply either size-reduction **or** swap
- Bad worst-case, good (empirical) average time