Concrete quantum cryptanalysis of binary elliptic curves

Gustavo Banegas, Daniel J. Bernstein, Iggy van Hoof and Tanja Lange

2021, September 13
Google Claims ‘Quantum Supremacy,’ Marking a Major Milestone in Computing

Quantum computing leaps ahead in 2019 with new power and speed

- Shor (1994): Sufficiently large quantum computers break DLP and RSA
- How big do these quantum computers need to be?
- Previous recent work: RSA & prime field ECDH.
- This work: binary ECDH.
- Results: $7n + \lfloor \log_2(n) \rfloor + 9$ qubits, $48n^3 + O(n^{\log_2(3)+1})$ TOF gates.
Most expensive step in Shor is adding precomputed points \([2^i]P\).

We treat rest of Shor as blackbox.

Point addition uses operations in \(\mathbb{F}_{2^n}\):
- Addition
- Multiplication
- Division

How do we build quantum circuits?
Quantum Gates

- Quantum bits: qubits.
- Today quantum computing 101: no purely quantum gates.
- Classical reversible gates:

  **NOT:**
  \[ a \oplus 1 - a \]

  **SWAP:**
  \[ a \leftrightarrow b \]

  **CNOT:**
  \[ a \oplus b \]

  **TOF:**
  \[ a \cdot b \]

\[ c \oplus (a \cdot b) \]
Quantum circuits

- We can now make circuits.
- Number of qubits is most important.
- Need a measure of quality:
  - Gate count?
  - TOF gates?
  - Depth?
  - Physical implementation?
Basic arithmetic: Addition in $\mathbb{F}_{2^n}$

- Binary addition with a constant: NOT gates (same as classically).
- Binary addition of 2 variables:
  - bitwise XOR $\rightarrow$ CNOT.
  - Reversible: 2 inputs $f, g$; 2 outputs $f \oplus g, g$.
  - $n$ CNOTs.
Basic arithmetic: Multiplication by $x$ in $\mathbb{F}_{2^n}$

- **Field**: use $\mathbb{F}_{2^n} \cong \mathbb{F}_2[x]/m(x)$ for an irreducible polynomial $m(x) \in \mathbb{F}_2[x]$ of degree $n$.
- Times $x$ without reduction is free with SWAP.
- Modular reduction in 1 CNOT for trinomial $m(x)$.
- Modular reduction in 3 CNOTs for pentanomial $m(x)$.
- Do in reverse for division by $x$.

- Figure: Multiplication by $x$ modulo $x^4 + x + 1$ with $g_0 + \cdots + g_3x^3$ as the input and $h_0 + \cdots + h_3x^3 = x \cdot g \mod x^4 + x + 1 = g_3 + (g_0 + g_3)x + g_1x^2 + g_2x^3$ as the output.
Basic arithmetic: Multiplication by constant & Squaring in $\mathbb{F}_{2^n}$

- Multiplication by a constant is a linear map.
- Turn linear map into a series of CNOTs using LUP decomposition.
- Do the same with squaring, linear map; $(a + b)^2 = a^2 + b^2$ in $\mathbb{F}_{2^n}$.
- Alternatively, adding the squaring result to a second polynomial also with only CNOTs.
Earlier work (van Hoof, Quantum Information and Computation 2020):

Quantum Karatsuba multiplication in $\mathbb{F}_{2^n}$.

No ancillary qubits needed, only $3n$ space.

Previous work used extra qubits.

Optimal TOF gate count for Karatsuba: $n^{\log_2 3}$ TOF gates.
Most expensive step: division or inversion.

We compare 2 methods:

- Extended Euclidean algorithm.
- Fermat’s little theorem.
Normal Euclidean algorithm has variable number of steps.

Based on constant time classical inversion (Bernstein & Yang, CHES 2019).

\[ |\delta\rangle \xrightarrow{|\delta|} \]

\[ |\text{sign}\rangle \]

\[ |f\rangle \xrightarrow{n+1} \]

\[ |g\rangle \xrightarrow{n+1} \]

\[ g_0[\ell] = 0 \]

\[ a = 0 \]

\[ |r\rangle \xrightarrow{n+1} \]

\[ |v\rangle \xrightarrow{n+1} \]

\[ \frac{1}{\chi} \]

\[ +1 \]

\[ \frac{1}{\chi} \]

\[ \times \]

\[ \times \]

\[ \times \]

\[ \times \]

\[ \times \]

\[ \times \]

\[ \times \]

Figure: Step \( \ell \) of Algorithm 1. \(|\delta| = \lfloor \log(n) \rfloor + 1.\)
Fermat’s little theorem: $x^p \equiv x \mod p \rightarrow x^{p-2} \equiv x^{-1} \mod p$.

Binary FLT: $x^{2n-2} \equiv x^{-1} \mod m(x)$.

Itoh-Tsujii inversion optimizes this.

Large number of squarings, low number of multiplications.

Number of multiplications depends on HW of $n - 1$.

Every multiplication costs $n^{\log_2 3}$ TOF gates.

Squaring costs only CNOT gates.
FLT-based inversion circuit

Figure: Step 1-3 of Algorithm 2 for \( n = 10 \). \( S \) is the squaring circuit and \( M \) is multiplication.
XGCD vs FLT

- Extended Euclidean algorithm uses more TOF gates.
- Fermat’s little theorem uses more qubits and CNOT gates.
- Example: \( n = 233 \):

<table>
<thead>
<tr>
<th>inversion method</th>
<th>TOF gates</th>
<th>qubits</th>
</tr>
</thead>
<tbody>
<tr>
<td>XGCD</td>
<td>827,977</td>
<td>1,646</td>
</tr>
<tr>
<td>FLT</td>
<td>132,783</td>
<td>3,029</td>
</tr>
</tbody>
</table>
Point addition

- On the curve we need to add multiples of $P$ to quantum $P_1 = (x_1, y_1)$.
  - If qubit $q_i = 1$ output $(x_3, y_3) = P_1 + P_2$ with pre-computed $P_2 = (x_2, y_2) = [2^i]P$.
  - Else: output $P_1$.
- Binary addition in affine coordinates: 2 S, 2 M and 2 D in $F_{2^n}$.
- 2nd division returns ancillary qubits to all-0.
- Special cases:
  - Addition with $O$.
  - Addition when $x_1 = x_2$. 

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$\frac{1}{n} + x_2$</th>
<th>$+ a + x_2$</th>
<th>$+ x_2$</th>
<th>$x_3$ or $x_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q$</td>
<td></td>
<td></td>
<td></td>
<td>$q$</td>
</tr>
<tr>
<td>$y_1$</td>
<td>$\frac{1}{n} + y_2$</td>
<td>$M$</td>
<td>$S$</td>
<td>$y_3$ or $y_1$</td>
</tr>
<tr>
<td>$0$</td>
<td>$\frac{1}{n}$</td>
<td>$D$</td>
<td></td>
<td>$0$</td>
</tr>
</tbody>
</table>
Can we use classical precomputation?

Classical computation is very cheap.

Classically precompute all \(2^\ell(a_0P + a_12P + \cdots + a_{\ell-1}2^{\ell-1}P)\) with \(a_i \in \{0, 1\}\) and handle \(\ell\) bits in one quantum addition.

Need qRAM lookups.

Example: \(n = 233:\)

<table>
<thead>
<tr>
<th>Window size</th>
<th>Point additions</th>
<th>TOF gates</th>
<th>Lookups</th>
<th>Pre-computed points</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>468</td>
<td>781 M</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>68</td>
<td>113 M</td>
<td>408</td>
<td>8,704</td>
</tr>
<tr>
<td>16</td>
<td>30</td>
<td>52 M</td>
<td>180</td>
<td>1,966,080</td>
</tr>
<tr>
<td>32</td>
<td>16</td>
<td>27 M</td>
<td>96</td>
<td>68,719,476,736</td>
</tr>
</tbody>
</table>
• Division is the most expensive step.
• Results without windowing:

<table>
<thead>
<tr>
<th>$n$</th>
<th>qubits</th>
<th>Point addition TOF gates</th>
<th>Total TOF gates</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>68</td>
<td>7,360</td>
<td>132,480</td>
</tr>
<tr>
<td>16</td>
<td>125</td>
<td>21,016</td>
<td>714,544</td>
</tr>
<tr>
<td>127</td>
<td>904</td>
<td>559,141</td>
<td>143,140,096</td>
</tr>
<tr>
<td>163</td>
<td>1,157</td>
<td>893,585</td>
<td>293,095,880</td>
</tr>
<tr>
<td>233</td>
<td>1,647</td>
<td>1,669,299</td>
<td>781,231,932</td>
</tr>
<tr>
<td>283</td>
<td>1,998</td>
<td>2,427,369</td>
<td>1,378,745,592</td>
</tr>
<tr>
<td>571</td>
<td>4,015</td>
<td>8,987,401</td>
<td>10,281,586,744</td>
</tr>
</tbody>
</table>
Summary: Windowing

- Need approximation of cost of qRAM lookup.
- Previous work: $2(2^\ell - 1)$ TOF gates per lookup (Babbush, Gidney, Berry, Wiebe, McClean, Paler, Folwer and Neven, Physical Review, 2018).
- Results with windowing:

<table>
<thead>
<tr>
<th>$n$</th>
<th>$\ell$</th>
<th>TOF gates</th>
<th>Lookups</th>
<th>Total TOF gates</th>
<th>pre-computed points</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>7</td>
<td>29,344</td>
<td>24</td>
<td>35,440</td>
<td>512</td>
</tr>
<tr>
<td>16</td>
<td>8</td>
<td>125,808</td>
<td>36</td>
<td>144,168</td>
<td>1,536</td>
</tr>
<tr>
<td>127</td>
<td>13</td>
<td>11,733,960</td>
<td>120</td>
<td>13,699,800</td>
<td>163,840</td>
</tr>
<tr>
<td>163</td>
<td>13</td>
<td>24,113,592</td>
<td>156</td>
<td>26,669,184</td>
<td>212,992</td>
</tr>
<tr>
<td>233</td>
<td>14</td>
<td>58,401,000</td>
<td>204</td>
<td>65,085,264</td>
<td>557,056</td>
</tr>
<tr>
<td>283</td>
<td>14</td>
<td>101,913,840</td>
<td>252</td>
<td>110,170,872</td>
<td>688,128</td>
</tr>
<tr>
<td>571</td>
<td>16</td>
<td>655,955,224</td>
<td>432</td>
<td>712,577,464</td>
<td>4,718,592</td>
</tr>
</tbody>
</table>
Comparison to other work

- Division and multiplication numbers look good for binary fields.
- General results:
  - $7n + \lfloor \log_2(n) \rfloor + 9$ qubits, mostly ancillary qubits for division.
  - $48n^3 + 8n^{\log_2(3)+1} + 352n^2 \log_2(n) + 512n^2 + O(n^{\log_2(3)})$ TOF gates.
- Prime field: similar results (Roetteler, Naehrig, Svore and Lauter, Asiacrpt 2017 & Hāner, Jaques, Naehrig, Roetteler and Soeken, PQCrypto 2020).
  - Lots of speedup over the prime field case: addition, multiplication, division all cheaper in $\mathbb{F}_{2^n}$.
- Projective binary coordinates: not optimized for space (Amento, Roetteler, Steinwandt, Quantum Information and Computation 2013)
  - Significantly worse space.
  - Lower TOF gate count due to fewer divisions.