Performance of Quantum-Safe Isogenies on ARM Processors

Reza Azarderakhsh
Florida Atlantic University and PQSecure Technologies
Why Post-Quantum Cryptography? Why Now?

- The history of Integrated Circuits (IC)
  - 1958: First integrated circuit (1cm², 2 transistors)
  - 1971: Moore's Law is born (2,300 transistors)
  - 2014: IBM P8 Processor, 16 cores (650mm², > 4.2 billion transistors)

- Quantum Computers

  - 2015: 4-Qbit
  - 2016: 5-Qbit
  - 2016: 8-Qbit
  - 2017: 16-Qbit
  - 2018: 72-Qbit

- Photon-based Quantum Computers are under development!
Quantum Computers

Quantum Computers will be able to solve many problems in different areas:

- Weather prediction
- Medical and healthcare
- Machine learning
- What else?
Quantum Computers

Quantum Computers will be able to solve many problems in different areas:

- Weather prediction
- Medical and healthcare
- Machine learning
- What else?

Current PKC is also constructed on hard problems!

- **RSA**: Discrete Logarithm Problem (DLP)
- **ECC**: Elliptic Curve Discrete Logarithm Problem (ECDLP)

Shor’s quantum algorithm can solve these problems in **Polynomial-time 😞**
Primary PQC Candidates

Code-Based: McEliece

Hash-Based: Lamport – Merkle Signature

Lattice-Based: NTRU – LWE – RLWE

Multivariate: Rainbow Signatures

Isogeny-Based: SIDH – SIKE
Diffie-Hellman Key-exchange

Secret Key
Public Key

Public Key

Public Key

Shared Key

Secret Key
Public Key
Classical ECC vs. Post-Quantum Isogeny Cryptography

Classical elliptic curve cryptography
Classical ECC vs. Post-Quantum Isogeny Cryptography

Classical elliptic curve cryptography

\[ P + Q = R \]
Classical ECC vs. Post-Quantum Isogeny Cryptography

Post-Quantum Isogeny-based cryptography

\[ E \xrightarrow{\phi} E' = \phi(E) \]
Classical ECC vs. Post-Quantum Isogeny Cryptography

Post-Quantum Isogeny-based cryptography

\[ P \rightarrow Q \rightarrow \phi(P) \rightarrow \phi(Q) \]

\[ E \rightarrow \phi(E) \rightarrow E' = \phi(E) \]
Supersingular Isogeny-based Cryptography History

- The first suggestions to use isogenies in crypto by Couveignes in 1997.
- Supersingular isogeny hash function by Charles, Lauter and Goren in 2005.
- Isogeny-based public-key cryptosystems by Rostovtsev and Stolbunov in 2006.
- The biggest impetus by Jao, De Feo (SIDH) in 2011.
- Supersingular Isogeny Key Encapsulation (SIKE) by Jao et al. (NIST PQC 2017)
Isogeny-based Cryptography Underlying Problem

Consider two supersingular elliptic curves defined over a large prime extension field:

- $E_1$ and $E_2$ over $F_p^2$, where $p$ is a large prime.
- There exists some isogeny $\phi : E_1 \rightarrow E_2$ with a fixed, smooth degree which maps $E_1$ to $E_2$. 
Isogeny-based Cryptography Underlying Problem

Consider two supersingular elliptic curves defined over a large prime extension field:

- $E_1$ and $E_2$ over $F_p^2$, where $p$ is a large prime.
- There exists some isogeny $\phi : E_1 \rightarrow E_2$ with a fixed, smooth degree which maps $E_1$ to $E_2$.

Supersingular Isogeny Problem:

- Given $P, Q \in E_1$ and $\phi(P), \phi(Q) \in E_2$, retrieve the isogeny map $\phi$!
Isogeny-based Cryptography Underlying Problem

Consider two supersingular elliptic curves defined over a large prime extension field:

- $E_1$ and $E_2$ over $F_p^2$, where $p$ is a large prime.
- There exists some isogeny $\phi: E_1 \rightarrow E_2$ with a fixed, smooth degree which maps $E_1$ to $E_2$

Supersingular Isogeny Problem:

- Given $P, Q \in E_1$ and $\phi(P), \phi(Q) \in E_2$, retrieve the isogeny map $\phi$!
- The best known classical attack is based on “Meet in the Middle Attack”: $O(p^{1/4})$
- The best known quantum attack is based on “Claw’s algorithm”: $O(p^{1/6})$
Alice and Bob Isogeny Walks of Different Degree Isogenies
Alice and Bob Isogeny Walks of Different Degree Isogenies
Alice and Bob Isogeny Walks of Different Degree

Isogenies
Alice and Bob Isogeny Walks of Different Degree
Alice and Bob Isogeny Walks of Different Degree Isogenies

\[ j(E_{BA9}) = j(E_{AB12}) \]
Supersingular Isogeny Diffie-Hellman (SIDH) Key-exchange

Public Parameters

\[E, p\]
\[P_A, Q_A \in E\]
\[P_B, Q_B \in E\]

\[PK_A = [E_A, \phi_A(P_B), \phi_A(Q_B)]\]

\[PK_B = [E_B, \phi_B(P_A), \phi_B(Q_A)]\]
Supersingular Isogeny Cryptography Pros and Cons

Pros

• Very small public/private key size.
• Data-structure and implementation similar to ECC.
• Different security assumption.
• No possibility of decryption error.
• No error distribution, rejection sampling, etc.
• Conservative security analysis on generic attacks.

Cons

• Youngest PQC candidate.
• Slower compared to some other candidates.
• Security concerns when reuse keys.
• Needs more investigation on security and performance.
Small Keys Makes it Suitable for Embedded Devices

Communication bandwidth of some NIST PQC candidates KEM:

<table>
<thead>
<tr>
<th>Candidate</th>
<th>Primitive</th>
<th>Public Key (bytes)</th>
<th>Secret Key (bytes)</th>
<th>Ciphertext (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NewHope1024</td>
<td>RLWE</td>
<td>1824</td>
<td>3680</td>
<td>2208</td>
</tr>
<tr>
<td>Saber</td>
<td>Mod-LWR</td>
<td>992</td>
<td>2304</td>
<td>1088</td>
</tr>
<tr>
<td>NTRU-HRSS17</td>
<td>LWE</td>
<td>1138</td>
<td>1418</td>
<td>1278</td>
</tr>
<tr>
<td>Kyber-768</td>
<td>LWE</td>
<td>1088</td>
<td>2400</td>
<td>1152</td>
</tr>
<tr>
<td>NTRU Prime</td>
<td>RLWE</td>
<td>1218</td>
<td>1600</td>
<td>1047</td>
</tr>
<tr>
<td>SIKEp751</td>
<td>SI</td>
<td>564</td>
<td>644</td>
<td>596</td>
</tr>
</tbody>
</table>
Isogeny-based Cryptography Implementation

- Cryptography protocols deal with big integers $\rightarrow$ field arithmetic
- From top to bottom, the number of operations increases
- Optimization on the lowest level operations
Supersingular Isogeny Cryptography on ARM

• Different Families of Processors:
  • ARMv7-M → 32-bit Low-Power (Performance is challenging)
  • ARMv7-A → 32-bit High-Performance with NEON Instruction set
  • ARMv8-A → 64-bit High-Performance with Adv. SIMD instruction set
Using divide-and-conquer strategy for big integer multiplication:

- Replace one large $n$-bit multiplication with $3 \times \frac{n}{2}$-bit multiplications.
  \[
  A \cdot B = A_h \cdot B_h \cdot 2^n + [(A_h + A_l)(B_h + B_l) - A_h \cdot B_h - A_l \cdot B_l] \cdot 2^{\frac{n}{2}} + A_l \cdot B_l.
  \]
- Replace $\frac{n}{2}$-bit multiplications further with $3 \times \frac{n}{4}$-bit multiplications.
  \[
  A_h \cdot B_h = A_{hh} \cdot B_{hh} \cdot 2^{\frac{n}{2}} + [(A_{hh} + A_{hl})(B_{hh} + B_{hl}) - A_{hh} \cdot B_{hh} - A_{hl} \cdot B_{hl}] \cdot 2^{\frac{n}{4}} + A_{hl} \cdot B_{hl}.
  \]
  \[
  A_l \cdot B_l = A_{lh} \cdot B_{lh} \cdot 2^{\frac{n}{2}} + [(A_{lh} + A_{ll})(B_{lh} + B_{ll}) - A_{lh} \cdot B_{lh} - A_{ll} \cdot B_{ll}] \cdot 2^{\frac{n}{4}} + A_{ll} \cdot B_{ll}.
  \]
Supersingular Isogeny Cryptography on ARMv7-A

- Exploit SIMD capabilities of ARMv7-A cores for arithmetic implementation:
  - NEON assembly implementation
    - SIMD multiplication instructions reduce the total number of multiplications significantly!
    - For instance, see $128 \times 128$-multiplication using A32 and NEON below:

  \[
  \begin{array}{c}
  a_3 \quad a_2 \quad a_1 \quad a_0 \\
  b_3 \quad b_2 \quad b_1 \quad b_0 \\
  \end{array}
  \]

  \[
  \begin{array}{c}
  \text{MULL}(a_0, b_0) \quad \text{MULL}(a_1, b_0) \quad \text{MULL}(a_2, b_0) \quad \text{MULL}(a_3, b_0) \\
  \text{MULL}(a_0, b_1) \quad \text{MULL}(a_1, b_1) \quad \text{MULL}(a_2, b_1) \quad \text{MULL}(a_3, b_1) \\
  \text{MULL}(a_0, b_2) \quad \text{MULL}(a_1, b_2) \quad \text{MULL}(a_2, b_2) \quad \text{MULL}(a_3, b_2) \\
  \text{MULL}(a_0, b_3) \quad \text{MULL}(a_1, b_3) \quad \text{MULL}(a_2, b_3) \quad \text{MULL}(a_3, b_3) \\
  \end{array}
  \]

  \[
  \begin{array}{c}
  a_3 \quad a_2 \quad a_1 \quad a_0 \\
  b_3 \quad b_2 \quad b_1 \quad b_0 \\
  \end{array}
  \]

  \[
  \begin{array}{c}
  \text{VMULL}(a_0, a_1, b_0) \quad \text{VMULL}(a_0, a_2, b_0) \\
  \text{VMULL}(a_0, a_1, b_1) \quad \text{VMULL}(a_0, a_2, b_1) \\
  \text{VMULL}(a_0, a_1, b_2) \quad \text{VMULL}(a_0, a_2, b_2) \\
  \text{VMULL}(a_0, a_1, b_3) \quad \text{VMULL}(a_0, a_2, b_3) \\
  \end{array}
  \]

- 16× MULL instruction in A32 vs. 8× VMULL instruction in NEON!
Supersingular Isogeny Cryptography on ARMv8-A

• Exploit ASIMD capabilities of ARMv8-A cores for arithmetic implementation:
  • Pure A64 64-bit assembly gives better performance than pure ASIMD! (NO SIMD?! 😞)
  • We designed a tailored Mixed (A64+ASIMD) code for multiplier on ARMv8-A processors! 😊
Supersingular Isogeny Cryptography on ARMv8-A

- Exploit ASIMD capabilities of ARMv8-A cores for arithmetic implementation:
  - Pure A64 64-bit assembly gives better performance than pure ASIMD! *(NO SIMD?! 😞)*
  - We designed a tailored Mixed (A64+ASIMD) code for multiplier on ARMv8-A processors! 😊
Supersingular Isogeny Cryptography on ARMv8-A

- Exploit ASIMD capabilities of ARMv8-A cores for arithmetic implementation:
  - Pure A64 64-bit assembly gives better performance than pure ASIMD! (NO SIMD?! 😞)
  - We designed a tailored Mixed (A64+ASIMD) code for multiplier on ARMv8-A processors! 😊

```
Aₜₙ. Bₜₙ

Aₜₙₐₚₚₜₜₙₚₚₚₙₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚₚportion of 287x120 to 302x302]
## Performance Report on Various ARM Cores

<table>
<thead>
<tr>
<th>Work</th>
<th>Language</th>
<th>Processor</th>
<th>Scheme</th>
<th>Field Size</th>
<th>PQ Security level</th>
<th>Total Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>JAMJ17 (TDSC’17)</td>
<td>Pure C</td>
<td>Cortex-A57</td>
<td>SIDH</td>
<td>751</td>
<td>125</td>
<td>1,864</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A64 ASM/C</td>
<td></td>
<td></td>
<td></td>
<td>331</td>
</tr>
<tr>
<td>JACJM18 (TDSC’18)</td>
<td>Pure C</td>
<td>Cortex-A72</td>
<td>SIKE</td>
<td>751</td>
<td>125</td>
<td>1,495</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A64+ASIMD ASM/C</td>
<td></td>
<td></td>
<td></td>
<td>271</td>
</tr>
<tr>
<td>JAM18 (SPACE’18)</td>
<td>Pure C</td>
<td>Cortex-A8</td>
<td>SIKE</td>
<td>751</td>
<td>125</td>
<td>12,374</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NEON ASM/C</td>
<td></td>
<td></td>
<td></td>
<td>1,571</td>
</tr>
<tr>
<td></td>
<td>Pure C</td>
<td>Cortex-A15</td>
<td></td>
<td></td>
<td></td>
<td>3,891</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NEON ASM/C</td>
<td></td>
<td></td>
<td></td>
<td>526</td>
</tr>
</tbody>
</table>
Conclusion

• Quantum computers with their computational power will solve many NP problems.
• They can easily break all the current PKC protocols.
• We need to be prepared for this threat.
• NIST has already started the PQC standardization procedure.
• Different proposals have been submitted.
• SIKE is the only primitive which is constructed on the popular elliptic curves.
• SIKE offers the smallest keys and communication size.
• It is suitable for embedded devices and IoT.
“SIKE TEAM”