Quantum Period Finding against Symmetric Primitives in Practice

Xavier Bonnetain    Samuel Jaques
“Practical” quantum computing

An issue with quantum attacks
- We can’t run them
- We often only have asymptotics

What we do
We propose complete quantum circuits for the offline Simon’s cryptanalysis

Aims
- Get a better understanding of the attack
- Allow comparison with other quantum algorithms
- Study the limitations of the attack
We wrote the components of the attack in Q#, a quantum programming language
Simulates and tests X, CNOT, Toffoli, And, up to thousands of qubits
Counts resource use with some rudimentary optimization
The library is available:
https://github.com/sam-jaques/offline-quantum-period-finding
<table>
<thead>
<tr>
<th>Simon’s algorithm</th>
<th>The Offline Simon’s Algorithm</th>
<th>Quantum circuits</th>
<th>Conclusion</th>
</tr>
</thead>
</table>

**Simon’s algorithm**

Xavier Bonnetain, Samuel Jaques

Quantum Period Finding against Symmetric Primitives in Practice
Simon’s problem

- $f : \{0, 1\}^n \rightarrow \{0, 1\}^n$
- $s \in \{0, 1\}^n$
- $\forall x, y, f(y) = f(x) \iff x \oplus y \in \{0, s\}$
- $f$ hides the period $s$
- Goal: find $s$, given oracle access to $f$.

Classical resolution

Find a collision, in $\Omega(2^{n/2})$ samples.

Quantum resolution

Simon’s algorithm, in $\mathcal{O}(n)$ quantum queries, $\mathcal{O}(n^3)$ classical operations
Simon’s problem

- \( f : \{0, 1\}^n \rightarrow \{0, 1\}^n, \ s \in \{0, 1\}^n \)
- \( \forall x, y, f(y) = f(x) \Leftrightarrow x \oplus y \in \{0, s\} \)
- Goal: find \( s \), given oracle access to \( f \).

Simon’s algorithm

- Superposition queries \( \sum_x |x\rangle |f(x)\rangle \)
- Sample \( y: \ s \cdot y = 0 \)
- Repeat \( O(n) \) times and solve the system
The Even-Mansour Cipher

Built from a random permutation $P : \{0, 1\}^n \rightarrow \{0, 1\}^n$.

$$E_{k_1, k_2}(x) = k_2 \oplus P(x \oplus k_1)$$

Classical security

Any attack needs $\text{Time} \times \text{Data} \geq 2^n$
Quantum attack [KM12]

Quantum attack

\[ f(x) = E_{k_1,k_2}(x) \oplus P(x) \text{ satisfies } f(x \oplus k_1) = f(x) \, . \]

Even-Mansour is broken in polynomial time, with quantum query access.
Quantum attack [KM12]

$$f(x) = E_{k_1,k_2}(x) \oplus P(x)$$ satisfies $$f(x \oplus k_1) = f(x)$$.

Even-Mansour is broken in polynomial time, with quantum query access.
Quantum attack [KM12]

Quantum attack

\[ f(x) = E_{k_1, k_2}(x) \oplus P(x) \text{ satisfies } f(x \oplus k_1) = f(x). \]

Even-Mansour is broken in polynomial time, with quantum query access.
Quantum attack [KM12]

Quantum attack

\[ f(x) = E_{k_1, k_2}(x) \oplus P(x) \text{ satisfies } f(x \oplus k_1) = f(x). \]

Even-Mansour is broken in polynomial time, with quantum query access.
Quantum attack [KM12]

Quantum attack

\[ f(x) = E_{k_1,k_2}(x) \oplus P(x) \] satisfies \[ f(x \oplus k_1) = f(x) \].

Even-Mansour is broken in polynomial time, with quantum query access.
Quantum attack [KM12]

$$f(x) = E_{k_1,k_2}(x) \oplus P(x)$$ satisfies $$f(x \oplus k_1) = f(x)$$.

Even-Mansour is broken in polynomial time, with quantum query access.
The Offline Simon’s Algorithm
Removing the quantum queries

Producing the sample states with Q1 queries is possible... in time $2^n$, with the whole codebook.

$\rightarrow$ not an attack.
Q1 attack on Even-Mansour

We separate $k_1$ in two parts.

$$(\underbrace{\underbrace{k_1^{(1)}}_{x} + \underbrace{\underbrace{u}_{n - u}}_{0}}_{k_1^{(2)}})$$

$$k_2$$

$n$

$p$
Q1 attack on Even-Mansour

We separate $k_1$ in two parts.

\[
\begin{align*}
0 & \quad \text{n} - u \\
x & \quad u \\
k_1^{(1)} & \\
k_1^{(2)} & \\
k_2 & \quad \text{n}
\end{align*}
\]

Define $f(x) = E_{k_1,k_2}(x\|0^n - u) \oplus P(x\|k_1^{(2)})$. 
Q1 attack on Even-Mansour

We separate $k_1$ in two parts.

Define $f(x) = E_{k_1,k_2}(x\|0^{n-u}) \oplus P(x\|k_1^{(2)})$.

Data: $2^u$

Memory: $\mathcal{O}(nu)$

Time: $\mathcal{O}(2^u + 2^{(n-u)/2})$
Quantum circuits
Shape of the circuit

| i ⟩

| 0 ⟩ → H → E_k

| 0 ⟩ → H → E_k

| 0 ⟩ → H → E_k

| 0 ⟩ → H → E_k

| 0 ⟩ → H → E_k

Linear algebra → | Rank ⩾ n ⟩

Computation once beforehand

Simon's algorithm

Quantum circuits
Concrete query estimates [Bon21]

For an Even-Mansour with an $n$-bit state, it is enough to have:

- $n+9$ queries to the periodic function
- 11 bits of output for the periodic function
Attack-specific optimizations

**Attack properties**
- Part of the input is fixed
- We only need 11 bits of output
- only relevant property is the periodicity

**Optimizations**
- Compute only once the shared part
- Remove useless parts of the last rounds
- Remove the linear functions of the last rounds
Primitive: Chaskey

- Lightweight MAC, ISO standard
- At most $2^{48}$ message blocks with the same key.

ARX construction
- Addition: Easily in-place; cheap circuits are well-studied
- Rotation: Done “in-software” by re-labelling qubits
- Xor: Just CNOT gates
Chaskey Circuits

First round:

\[
|v_0\rangle \quad \begin{array}{c}
\downarrow 32 \\
+ \\
\downarrow 15 \\
\uparrow 7 \\
\downarrow 13 \\
\uparrow 16
\end{array}
\]

\[
|v_1\rangle \quad \begin{array}{c}
\downarrow 32 \\
\downarrow 7 \\
\uparrow 16
\end{array}
\]

\[
|v_2\rangle \quad \begin{array}{c}
\downarrow 32 \\
\downarrow 15 \\
\downarrow 7 \\
\downarrow 13 \\
\uparrow 16
\end{array}
\]

\[
|v_3\rangle \quad \begin{array}{c}
\downarrow 32 \\
\downarrow 15 \\
\downarrow 7 \\
\downarrow 13 \\
\uparrow 16
\end{array}
\]
Chaskey Circuits

First round:

\[
\begin{align*}
|v_0\rangle & \quad \oplus \quad 32 \\
|v_1\rangle & \quad \oplus \quad 32 \\
|v_2\rangle & \quad \oplus \quad 32 \\
|v_3\rangle & \quad \oplus \quad 32
\end{align*}
\]
Chaskey Circuits

Last 2 rounds:

\[ |v_0\rangle + |v_1\rangle + |v_2\rangle + |v_3\rangle \]

Xavier Bonnetain, Samuel Jaques
Chaskey Circuits

Last 2 rounds:

|v₀⟩ + 32 + 16
|v₁⟩ + 32 5
|v₂⟩ + 32 + 7
|v₃⟩ + 32

Xavier Bonnetain, Samuel Jaques
Linear Algebra

Circuit to find the rank of an $m \times n$ binary matrix, with $m > n$:

- Compute a triangular basis and reduce the input vectors in-place.
- Depth: $O((m + n) \log n)$
- Gates: $mn^2 + mn$ Toffoli gates
- Qubits: $mn$ as input, plus $m + \frac{n(3n-1)}{2}$ extra qubits
## Chaskey Summary

<table>
<thead>
<tr>
<th>Target</th>
<th>Bitlength</th>
<th>Offline Queries</th>
<th>Operations All</th>
<th>Operations T</th>
<th>Depth All</th>
<th>Depth T</th>
<th>Qubits</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chaskey-8</td>
<td>128</td>
<td>48</td>
<td>64.9</td>
<td>64.4</td>
<td>56.0</td>
<td>53.9</td>
<td>14.5</td>
<td>limited queries</td>
</tr>
<tr>
<td>Chaskey-12</td>
<td>128</td>
<td>48</td>
<td>65.1</td>
<td>64.5</td>
<td>56.4</td>
<td>54.1</td>
<td>14.5</td>
<td></td>
</tr>
<tr>
<td>Chaskey-8</td>
<td>128</td>
<td>50</td>
<td>64.3</td>
<td>64.0</td>
<td>55.5</td>
<td>54.4</td>
<td>14.5</td>
<td>unlimited queries</td>
</tr>
<tr>
<td>Chaskey-12</td>
<td>128</td>
<td>51</td>
<td>64.5</td>
<td>64.2</td>
<td>55.9</td>
<td>55.2</td>
<td>14.5</td>
<td></td>
</tr>
<tr>
<td>Chaskey-8</td>
<td>128</td>
<td>1</td>
<td>80.3</td>
<td>77.5</td>
<td>79.0</td>
<td>75.4</td>
<td>8.6</td>
<td>Exhaustive search</td>
</tr>
<tr>
<td>Chaskey-12</td>
<td>128</td>
<td>1</td>
<td>80.8</td>
<td>78.0</td>
<td>79.6</td>
<td>75.9</td>
<td>8.6</td>
<td></td>
</tr>
</tbody>
</table>

All figures in log base 2 except bitlength.
Conclusion
Summary

- The attack is competitive against exhaustive search
- Requires large amount of data
- Reasonable amount of qubits ($\sim 10000$)
- Not a near-term quantum attack
- Requires a specific structure
Limitations of the attack

Optimizations are limited with an increased number of rounds:

- \( \sim 25\% \) gain with Chaskey-8 (8 rounds), \( \sim 1\% \) gain with Elephant (80 rounds)
Limitations of the attack

Optimizations are limited with an increased number of rounds:
- \( \sim 25\% \) gain with Chaskey-8 (8 rounds), \( \sim 1\% \) gain with Elephant (80 rounds)

Many approaches can limit the impact of the attack:
- Data limitation (ex. Chaskey, Elephant)
- Large state size (ex. Elephant)
- Avoid the required structure (ex. PRINCEv2)
**Thanks for listening!**

<table>
<thead>
<tr>
<th>Target</th>
<th>Bitlength</th>
<th>Offline Queries</th>
<th>Operations All</th>
<th>Operations T</th>
<th>Depth All</th>
<th>Depth T</th>
<th>Qubits</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSA</td>
<td>2048</td>
<td>–</td>
<td>–</td>
<td>31</td>
<td>31</td>
<td>–</td>
<td>12.6</td>
<td>[GE19]</td>
</tr>
<tr>
<td>Chaskey-8</td>
<td>128</td>
<td>48</td>
<td>64.9</td>
<td>64.4</td>
<td>56.0</td>
<td>53.9</td>
<td>14.5</td>
<td></td>
</tr>
<tr>
<td>Chaskey-12</td>
<td>128</td>
<td>48</td>
<td>65.1</td>
<td>64.5</td>
<td>56.4</td>
<td>54.1</td>
<td>14.5</td>
<td></td>
</tr>
<tr>
<td>PRINCE</td>
<td>64</td>
<td>48</td>
<td>65.0</td>
<td>64.5</td>
<td>55.2</td>
<td>53.8</td>
<td>14.0</td>
<td>ours</td>
</tr>
<tr>
<td>Elephant</td>
<td>160</td>
<td>47</td>
<td>84.1</td>
<td>82.5</td>
<td>72.6</td>
<td>70.4</td>
<td>14.8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>176</td>
<td>47</td>
<td>92.5</td>
<td>90.9</td>
<td>80.8</td>
<td>78.5</td>
<td>15.1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>69</td>
<td>93.6</td>
<td>91.7</td>
<td>83.7</td>
<td>79.3</td>
<td>16.4</td>
<td></td>
</tr>
<tr>
<td>AES</td>
<td>128</td>
<td>1</td>
<td>82.3</td>
<td>80.4</td>
<td>74.7</td>
<td>71.6</td>
<td>10.7</td>
<td>[DP20]</td>
</tr>
</tbody>
</table>

All figures in log base 2 except bitlength.