Quantum Period Finding against Symmetric Primitives

Xavier Bonnetain

January 26, 2021

Outline

Quantum Computing

Quantum computing

Classical computing

x	Input
$\downarrow$	
$a, b, \ldots$	Intermediate values
$\downarrow$	
y	Final result

Quantum computing

$\lvert x \rangle$	Input
$\downarrow$	
$\lvert a, b, \ldots \rangle$	Intermediate state
$\downarrow$	
$\lvert y \rangle \longrightarrow y$	Final measurement

Differences

- More possibilities $\lvert 0 \rangle$, $\lvert 1 \rangle$, $\lvert 0 \rangle - \lvert 1 \rangle \ldots$
- Reversible computing
- New operators $H : \lvert b \rangle \mapsto \frac{1}{\sqrt{2}} \left(\lvert 0 \rangle + (-1)^b \lvert 1 \rangle \right)$

Amplitude Amplification

Unstructured Search problem

- $f : \{0,1\}^n \to \{0,1\}$, with M inputs x such that $f(x) = 1$
- Goal : find any x such that $f(x) = 1$, given oracle access to f.

Classical resolution

Brute force search, in $\Theta(2^n/M)$ samples.

Quantum resolution

Amplitude amplification, in $\Theta\left(\sqrt{2^n/M}\right)$ quantum queries

Ex: A single-target key search on AES-128 requires 2^{82} quantum operations.

Simon's algorithm

Simon's problem

Simon's problem

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

Classical resolution

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

Quantum resolution

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

Simon's algorithm [Sim94]

Quantum circuit

- *Start from* $|0\rangle |0\rangle$

Simon's algorithm [Sim94]

Quantum circuit

- Start from $|0\rangle\,|0\rangle$
- *Apply H:* $\frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle\,|0\rangle$

Simon's algorithm [Sim94]

Quantum circuit

- Start from $|0\rangle\,|0\rangle$
- Apply H: $\frac{1}{2^{n/2}}\sum_{x=0}^{2^n-1}|x\rangle\,|0\rangle$
- *Apply* O_f: $\frac{1}{2^{n/2}}\sum_{x=0}^{2^n-1}|x\rangle\,|f(x)\rangle$

Simon's algorithm [Sim94]

Quantum circuit

- Start from $|0\rangle|0\rangle$
- Apply H: $\frac{1}{2^{n/2}}\sum_{x=0}^{2^n-1}|x\rangle|0\rangle$
- Apply O_f: $\frac{1}{2^{n/2}}\sum_{x=0}^{2^n-1}|x\rangle|f(x)\rangle$
- *Measure the second register: get* $f(x_0)$ *and project to* $\frac{1}{\sqrt{2}}(|x_0\rangle + |x_0 \oplus s\rangle)$

Simon's algorithm [Sim94]

Quantum circuit

- Start from $|0\rangle|0\rangle$
- Apply H: $\frac{1}{2^{n/2}}\sum_{x=0}^{2^n-1}|x\rangle|0\rangle$
- Apply O_f: $\frac{1}{2^{n/2}}\sum_{x=0}^{2^n-1}|x\rangle|f(x)\rangle$
- Measure the second register: get $f(x_0)$ and project to $\frac{1}{\sqrt{2}}(|x_0\rangle+|x_0\oplus s\rangle)$
- *Reapply H:* $\frac{1}{2^{n/2}}\sum_{y=0}^{2^n-1}(-1)^{x_0\cdot y}|y\rangle+(-1)^{(x_0\oplus s)\cdot y}|y\rangle$

Simon's algorithm [Sim94]

Quantum circuit

- Start from $|0\rangle |0\rangle$
- Apply H: $\frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle |0\rangle$
- Apply O_f: $\frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle |f(x)\rangle$
- Measure the second register: get $f(x_0)$ and project to $\frac{1}{\sqrt{2}}(|x_0\rangle + |x_0 \oplus s\rangle)$
- Reapply H: $\frac{1}{2^{n/2}} \sum_{y=0}^{2^n-1} (-1)^{x_0 \cdot y} |y\rangle + (-1)^{(x_0 \oplus s) \cdot y} |y\rangle$
- *The state is* $\frac{1}{2^{n/2}} \sum_{y=0}^{2^n-1} (-1)^{x_0 \cdot y} (1 + (-1)^{s \cdot y}) |y\rangle$

The y_0 we measure must satisfy $1 + (-1)^{s \cdot y_0} \neq 0 \Rightarrow y_0 \cdot s = 0$.

Simon's algorithm [Sim94]

Simon's problem

- $f : \{0,1\}^n \to \{0,1\}^n$, $s \in \{0,1\}^n$
- $\forall x, y, f(y) = f(x) \Leftrightarrow x \oplus y \in \{0, s\}$
- 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
- Requires $n + 2$ queries on average

Simon-based cryptanalysis

General idea

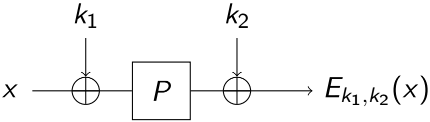
Create a periodic function from a cipher, whose period is a secret.

Characteristics

- Polynomial time, only $\mathcal{O}(n)$ queries
- Require quantum queries

The Even-Mansour Cipher

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

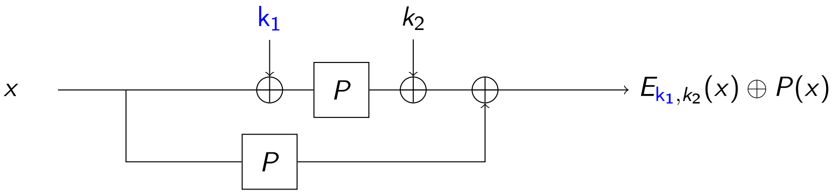


$$E_{k_1,k_2}(x) = k_2 \oplus P(x \oplus k_1)$$

Classical security

Any attack needs Time $\times$ Data $\geq 2^n$

Quantum attack [KM12]

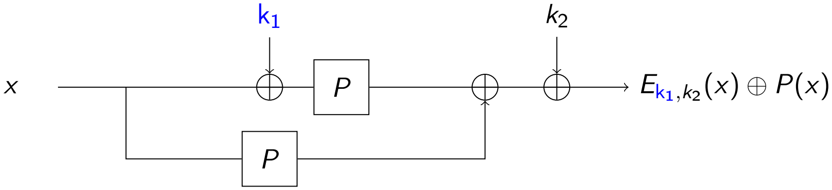


Quantum attack

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

Even-Mansour is broken in polynomial time, with quantum query access.

Quantum attack [KM12]

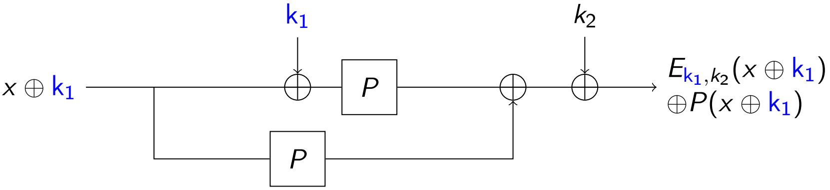


Quantum attack

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

Even-Mansour is broken in polynomial time, with quantum query access.

Quantum attack [KM12]

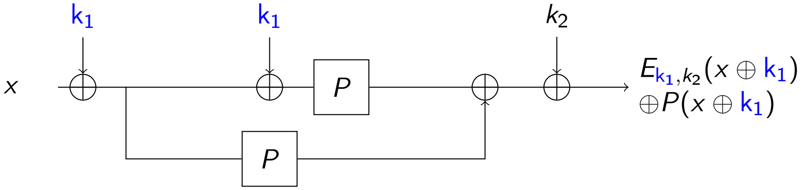


Quantum attack

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

Even-Mansour is broken in polynomial time, with quantum query access.

Quantum attack [KM12]

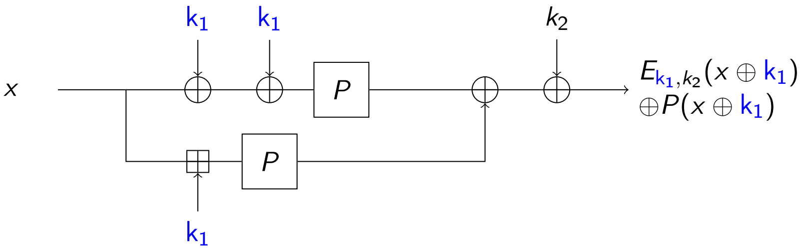


Quantum attack

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

Even-Mansour is broken in polynomial time, with quantum query access.

Quantum attack [KM12]

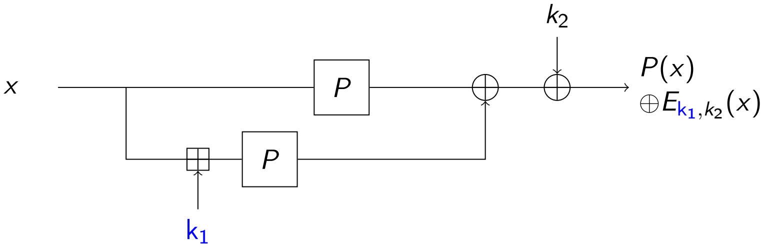


Quantum attack

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

Even-Mansour is broken in polynomial time, with quantum query access.

Quantum attack [KM12]



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.

A technical issue [Bon20]

Periodic function

- $f(x) = E_{k_1,k_2}(x) \oplus P(x)$
- We may have $f(x) = f(y)$ and $x \neq y \oplus k_1$

A technical issue [Bon20]

Periodic function

- $f(x) = E_{k_1,k_2}(x) \oplus P(x)$
- We may have $f(x) = f(y)$ and $x \neq y \oplus k_1$

- Soundness is not affected

A technical issue [Bon20]

Periodic function

- $f(x) = E_{k_1,k_2}(x) \oplus P(x)$
- We may have $f(x) = f(y)$ and $x \neq y \oplus k_1$

- Soundness is not affected
- Biaises appear in the sampled values.

A technical issue [Bon20]

Periodic function

- $f(x) = E_{k_1,k_2}(x) \oplus P(x)$
- We may have $f(x) = f(y)$ and $x \neq y \oplus k_1$

- Soundness is not affected
- Biaises appear in the sampled values.
- Worst case: $f(x) = \begin{cases} 1 & \text{if } x \in \{0, s\} \\ 0 & \text{otherwise} \end{cases}$

A technical issue [Bon20]

Periodic function

- $f(x) = E_{k_1,k_2}(x) \oplus P(x)$
- We may have $f(x) = f(y)$ and $x \neq y \oplus k_1$

- Soundness is not affected
- Biaises appear in the sampled values.
- Worst case: $f(x) = \begin{cases} 1 & \text{if } x \in \{0, s\} \\ 0 & \text{otherwise} \end{cases}$
- For almost all functions, requires only $n + 3$ queries on average

Simon-based cryptanalysis

- Distinguishers on Feistel constructions
- Multiple quantum slide attacks
- AEZ
- Multiple modes of operation
- Quantum related-key attacks
- . . .

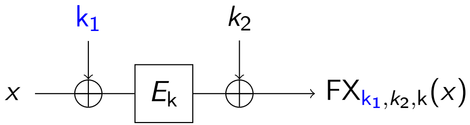
Simon-based cryptanalysis

- Distinguishers on Feistel constructions
- Multiple quantum slide attacks
- AEZ
- Multiple modes of operation
- Quantum related-key attacks
- ...

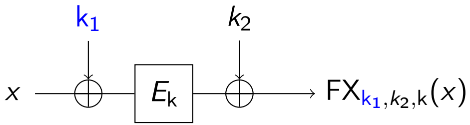
Require quantum queries

The Offline Simon's Algorithm

Example: FX construction



Example: FX construction



Quantum attack: "Grover-meet-Simon" [LM17]

- Quantum search for k
- Checking: Kuwakado and Morii's attack works $\iff$ the guess of k is correct

Total time is $\underbrace{poly(n)}_{\text{Simon's algo}} \times \underbrace{2^{|k|/2}}_{\text{Grover's iterates}}$.

Our remark on FX [BHNSS19]

The function:

$$f_z(x) = \mathrm{FX}_{\mathsf{k}_1, k_2, \mathsf{k}}(x) \oplus E_z(x)$$

has $f_z(x \oplus \mathsf{k}_1) = f_z(x)$ **if** $z = \mathsf{k}$ **(the good one)**.
f_z is a sum:

$$f_z(x) = \underbrace{\mathrm{FX}_{\mathsf{k}_1, k_2, \mathsf{k}}(x)}_{\text{Independent of z: online function } f} \oplus \underbrace{E_z(x)}_{\text{Grover search space: offline function } g}$$

For one query to f_z

- Do one quantum query to $\mathrm{FX}_{\mathsf{k}_1, k_2, \mathsf{k}}(x)$ (fixed!)
- Add $E_z(x)$ (only depends on public information)

A new test algorithm

1. Begin with $\mathcal{O}(n)$ states of the form $\sum_{x \in \{0,1\}^n} |x\rangle |f(x)\rangle$
2. Make queries to g and build: $\sum_{x \in \{0,1\}^n} |x\rangle |(f \oplus g)(x)\rangle$
3. With Simon's algorithm, obtain a single output bit: whether $f \oplus g$ has a period or not
4. **Revert the computations**, query g again, put the "sample states" back to

$$\sum_{x \in \{0,1\}^n} |x\rangle |f(x)\rangle$$

This emulates a reversible quantum circuit that tests for the periodicity of $f \oplus g$, *with only preprocessed queries to* f.

Our Q2 attack on FX

The queries to $\mathrm{FX}_{k_1,k_2,k}(x)$ are made beforehand.

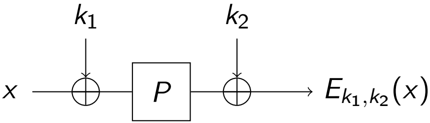
Test function

- Fetch the sample states $\sum_{x\in\{0,1\}^n} |x\rangle \left|\mathrm{FX}_{k_1,k_2,k}(x)\right\rangle$
- Create the Simon states $\sum_{x\in\{0,1\}^n} |x\rangle \left|\mathrm{FX}_{k_1,k_2,k}(x) \oplus E_z(x)\right\rangle$
- Test if there is a period
- **Revert** the operations and get back the sample states

Quantum search cost

- Time unchanged
- Queries reduced from $\mathcal{O}\left(n2^{|k|/2}\right)$ to $\mathcal{O}\left(n\right)$
- Needs $\mathcal{O}\left(n^2\right)$ Qubits

Back to the Even-Mansour cipher

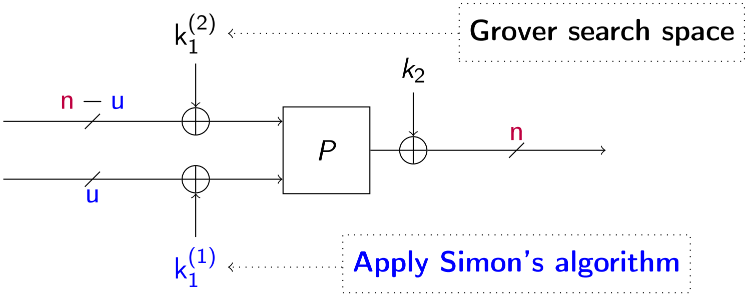


Producing the sample states with Q1 queries is possible. . . in time 2^n, with the whole codebook.

$\implies$ not an attack.

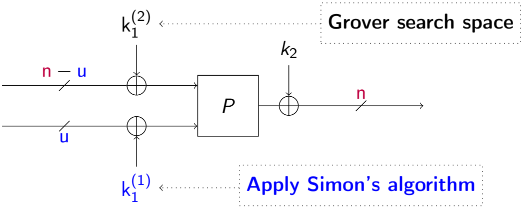
Q1 attack on Even-Mansour

We separate k_1 in two parts.



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

Q1 attack on Even-Mansour (ctd.)



$f(x) = E_{k_1,k_2}(x \| 0^{n-u}) \oplus P(x \| k_1^{(2)})$ has period $k_1^{(1)}$

1. Produce the sample states $\sum_x |x\rangle |E_{k_1,k_2}(x\|0^{n-u})\rangle$
2. Search the good $k_1^{(2)}$ ($n - u$ bits)

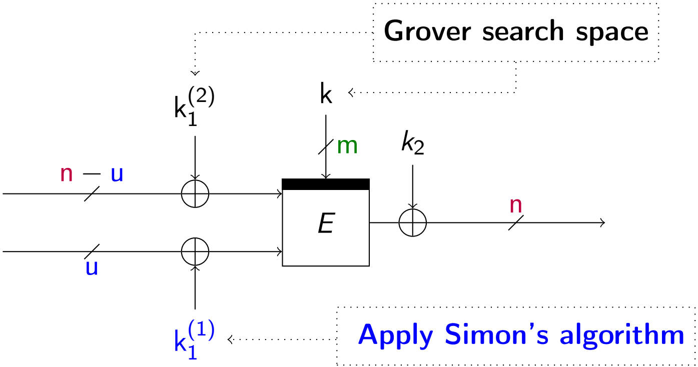
Data: 2^u

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

Time: $2^u + 2^{(n-u)/2}$

Balances when Data = Time = $2^{n/3}$

Q1 attack on the FX construction



We do the same, with more guesses in Grover's algorithm:
Data = Time = $2^{(n+m)/3}$.

Summary

The offline approach

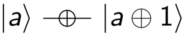
We reuse the quantum queries for each iteration of Simon's algorithm when the periodic function allows it.

Consequences

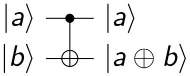
- Drastically reduces the number of quantum queries.
- Allows to convert a Q2 attack into a Q1 attack.
- Provides the best known Q1 attacks

Ciphers and Circuits

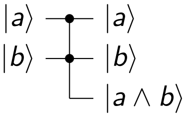
Quantum Operations/Gates



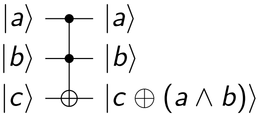
(a) Pauli X gate, or NOT gate.



(b) CNOT gate



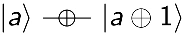
(c) AND gate



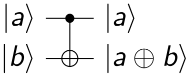
(d) Toffoli gate

} These are decomposed onto a standard set of gates: "Clifford + T"

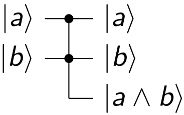
Quantum Operations/Gates



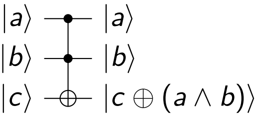
(a) Pauli X gate, or NOT gate.



(b) CNOT gate



(c) AND gate

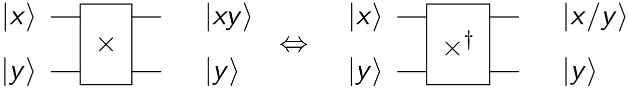


(d) Toffoli gate

} Only these require T gates

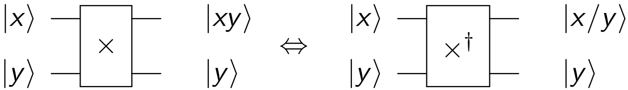
In-place vs. Out-of-place

In-place:



In-place vs. Out-of-place

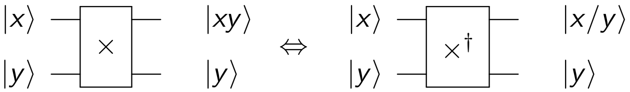
In-place:



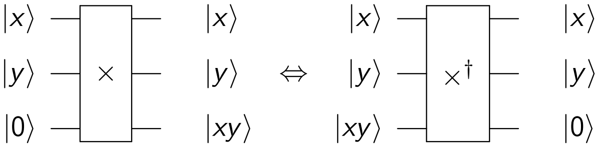
Conclusion: In-place multiplication as expensive as division

In-place vs. Out-of-place

In-place:



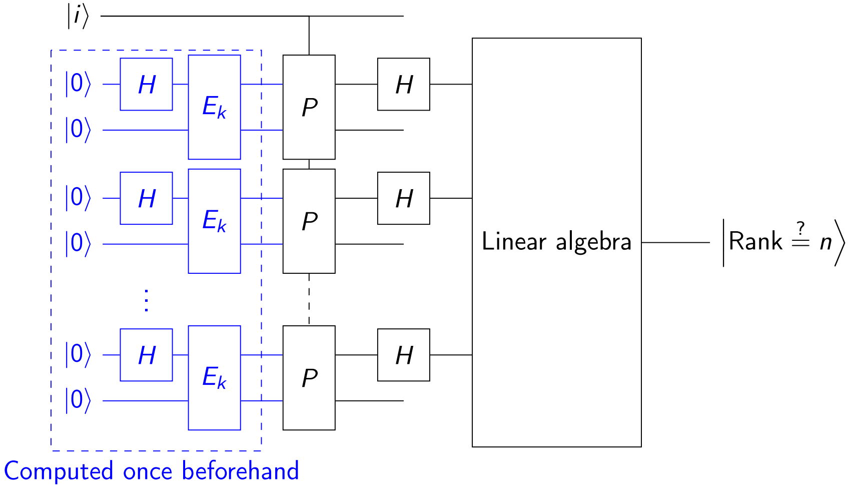
Out-of-place:



Q#

- We wrote the linear algebra and block ciphers 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`

Shape of the circuit



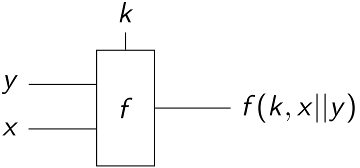
Linear Algebra [BJ20]

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: $O((m+n)\lg n)$
- Gates: $mn^2 + mn$ Toffoli gates
- Qubits: mn as input, plus $m + \frac{n(3n-1)}{2}$ extra qubits

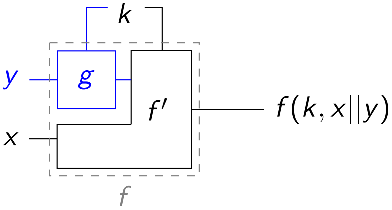
Optimization: Reduce input

The Simon-function oracle looks like:



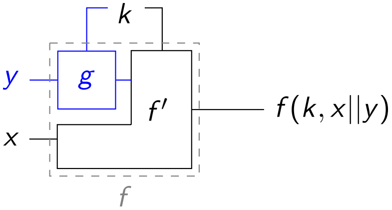
Optimization: Reduce input

The Simon-function oracle looks like:



Optimization: Reduce input

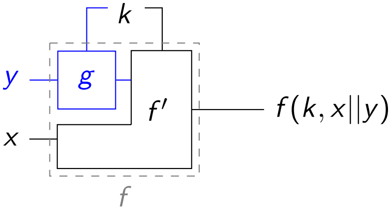
The Simon-function oracle looks like:



- Precompute g once for all ciphers

Optimization: Reduce input

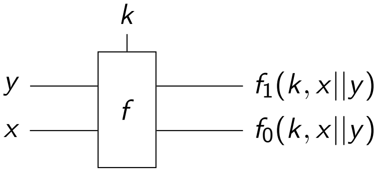
The Simon-function oracle looks like:



- Precompute g once for all ciphers
- Even better: g is a permutation, so don't compute it at all

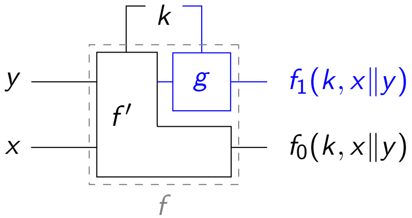
Optimization: Reduce Output

We only need 11 bits of output:



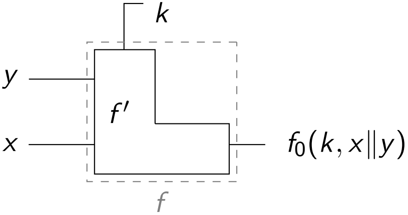
Optimization: Reduce Output

We only need 11 bits of output:



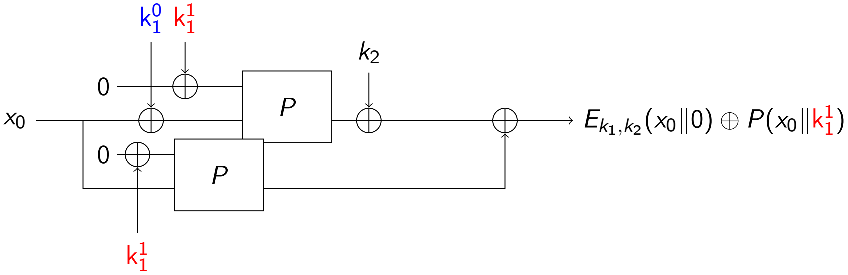
Optimization: Reduce Output

We only need 11 bits of output:



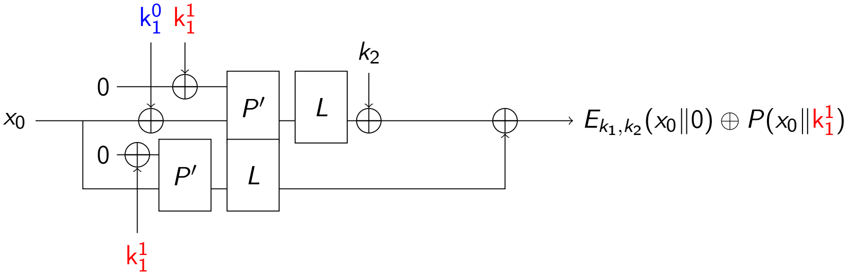
Optimizations at the end

Suppose $P = L \circ P'$ for a linear function L:



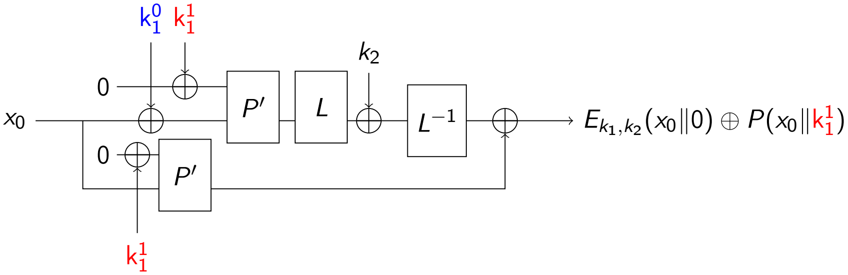
Optimizations at the end

Suppose $P = L \circ P'$ for a linear function L:



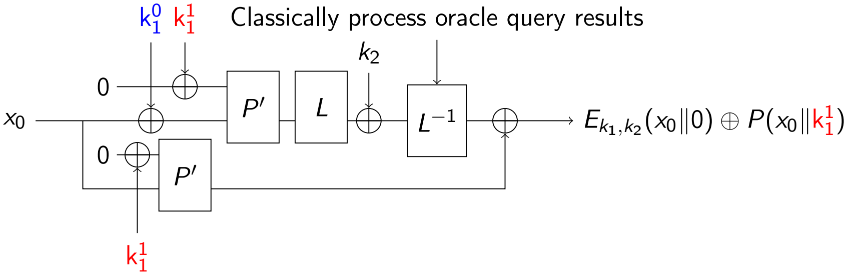
Optimizations at the end

Suppose $P = L \circ P'$ for a linear function L:

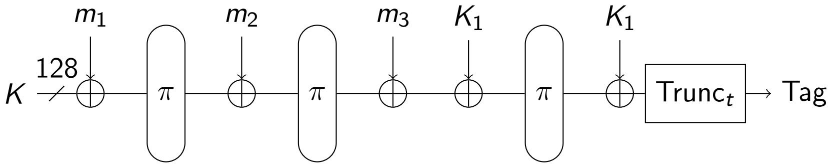


Optimizations at the end

Suppose $P = L \circ P'$ for a linear function L:



Primitive: Chaskey



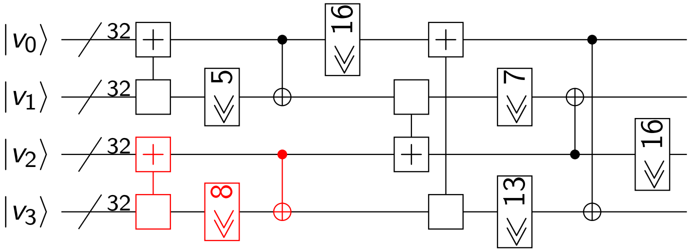
- Lightweight MAC, ISO standard
- At most 2^{48} message blocks with the same key.

ARX construction

- Addition: Easily in-place; cheap circuits are well-studied
- Rotation: Done "in-software" by re-labelling qubits
- Xor: Just CNOT gates

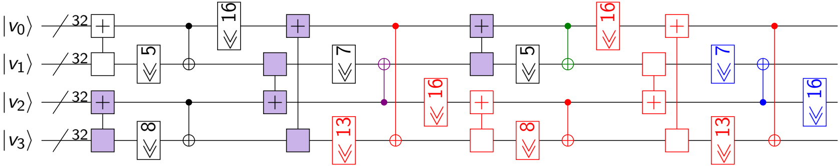
Chaskey Circuits

First round:



Chaskey Circuits

Last 2 rounds:



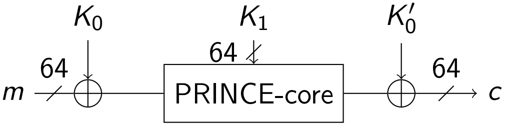
Red: entirely removed

Blue: Linear post-processing

Green: Done when copying out

Purple: Only least significant 16 bits

Primitive: PRINCE



Block cipher, used to encrypt memory in microcontrollers

Components of Prince-core

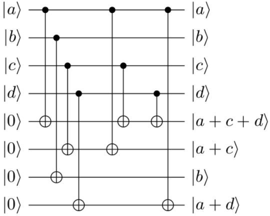
- Linear layer
- Constant additions
- Non-linear S-box (function in $\{0,1\}^4 \to \{0,1\}^4$)

PRINCE: Linear Layer

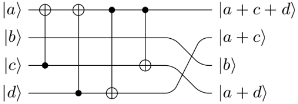
We follow [JNRV20] and use a PLU decomposition:

$$M = \begin{pmatrix} 1&0&1&1 \\ 1&0&1&0 \\ 0&1&0&0 \\ 1&0&0&1 \end{pmatrix} = \begin{pmatrix} 1&0&0&0 \\ 0&0&0&1 \\ 0&1&0&0 \\ 0&0&1&0 \end{pmatrix} \cdot \begin{pmatrix} 1&0&0&0 \\ 0&1&0&0 \\ 1&0&1&0 \\ 1&0&0&1 \end{pmatrix} \cdot \begin{pmatrix} 1&0&1&1 \\ 0&1&0&0 \\ 0&0&1&0 \\ 0&0&0&1 \end{pmatrix} = P \cdot L \cdot U$$

(a) Invertible linear transformation M and its PLU decomposition.



(b) Naive circuit computing M.



(c) In-place implementation of M.

PRINCE: S-box

We use an expression from secure hardware implementations [BKN18]:

$$S(x) = A_1 \circ Q_{294} \circ A_2 \circ Q_{294} \circ A_3 \circ Q_{294} \circ A_4$$

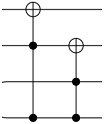
- A_i: Affine (use PLU decomposition)
- Q_{294}: Quadratic function

PRINCE: S-box

We use an expression from secure hardware implementations [BKN18]:

$$S(x) = A_1 \circ Q_{294} \circ A_2 \circ Q_{294} \circ A_3 \circ Q_{294} \circ A_4$$

- A_i: Affine (use PLU decomposition)
- Q_{294}: Quadratic function

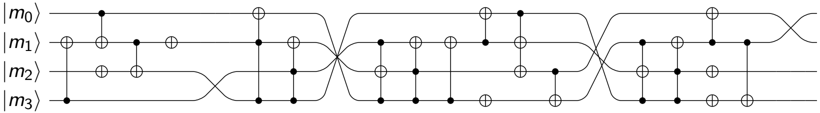


PRINCE: S-box

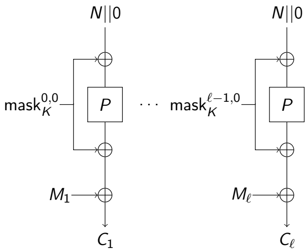
We use an expression from secure hardware implementations [BKN18]:

$$S(x) = A_1 \circ Q_{294} \circ A_2 \circ Q_{294} \circ A_3 \circ Q_{294} \circ A_4$$

- A_i: Affine (use PLU decomposition)
- Q_{294}: Quadratic function

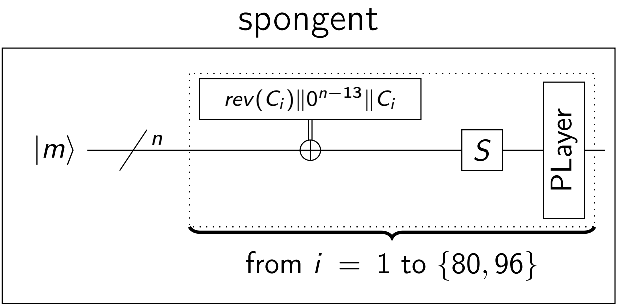


Elephant

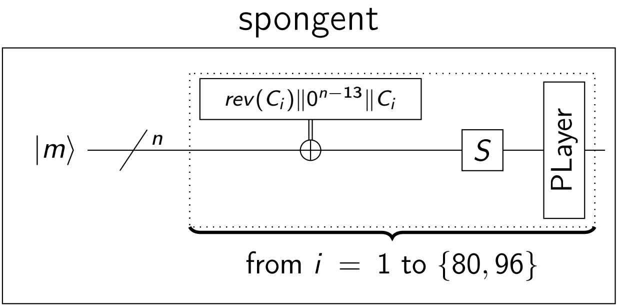


- Authenticated encryption, NIST LWC candidate
- 128 bits of key, 3 state sizes: 160, 176, 200
- Data limitation, respectively $2^{47}, 2^{47}, 2^{69}$ blocks.
- 160 and 176 use the SpongeNT permutation
- 200 uses Keccak

Elephant: SpongeNT

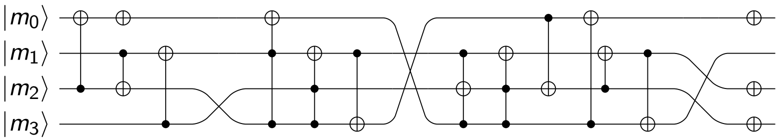


Elephant: SpongeNT



- Round constants Just X gates
- S: Use secure hardware decomposition
- PLayer: Decompose into swaps

Elephant: SpongeNT S-Box



Elephant: Keccak

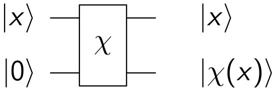
Keccak is a composition of 5 functions:

$$\underbrace{\iota}_{\text{round constant}} \circ \underbrace{\chi}_{\text{non-linear}} \circ \underbrace{\pi \circ \rho \circ \theta}_{\text{use PLU decomposition}}$$

Elephant: Keccak

Keccak is a composition of 5 functions:

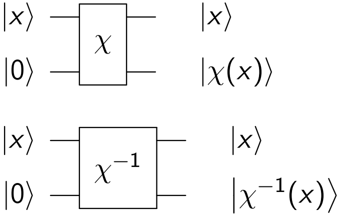
$$\underbrace{\iota}_{\text{round constant}} \circ \underbrace{\chi}_{\text{non-linear}} \circ \underbrace{\pi \circ \rho \circ \theta}_{\text{use PLU decomposition}}$$



Elephant: Keccak

Keccak is a composition of 5 functions:

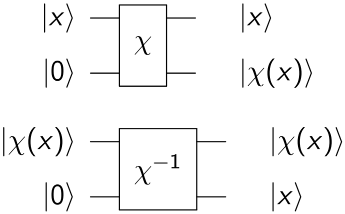
$$\underbrace{\iota}_{\text{round constant}} \circ \underbrace{\chi}_{\text{non-linear}} \circ \underbrace{\pi \circ \rho \circ \theta}_{\text{use PLU decomposition}}$$



Elephant: Keccak

Keccak is a composition of 5 functions:

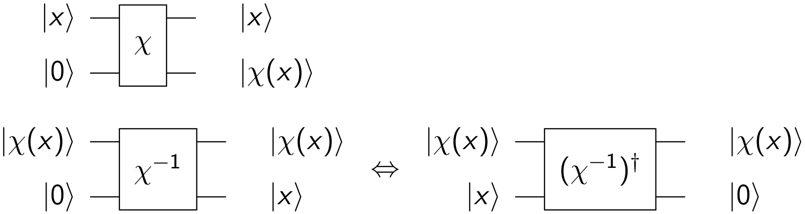
$$\underbrace{\iota}_{\text{round constant}} \circ \underbrace{\chi}_{\text{non-linear}} \circ \underbrace{\pi \circ \rho \circ \theta}_{\text{use PLU decomposition}}$$



Elephant: Keccak

Keccak is a composition of 5 functions:

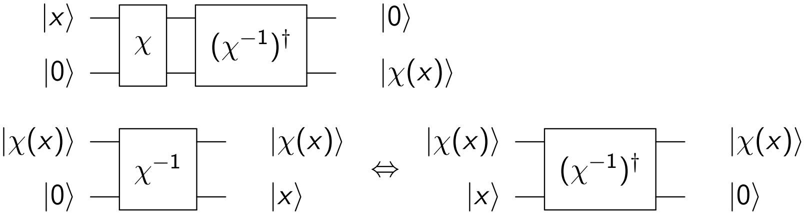
$$\underbrace{\iota}_{\text{round constant}} \circ \underbrace{\chi}_{\text{non-linear}} \circ \underbrace{\pi \circ \rho \circ \theta}_{\text{use PLU decomposition}}$$



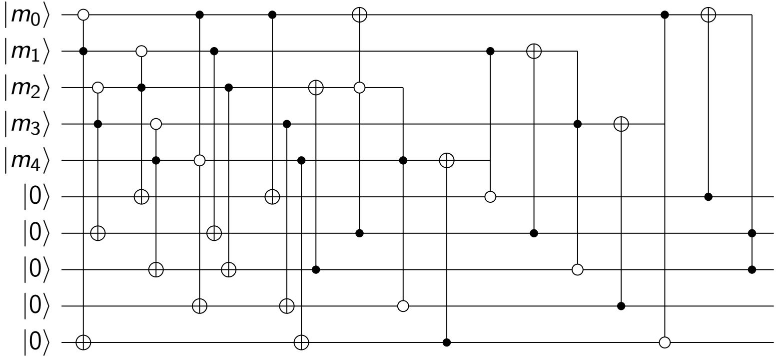
Elephant: Keccak

Keccak is a composition of 5 functions:

$$\underbrace{\iota}_{\text{round constant}} \circ \underbrace{\chi}_{\text{non-linear}} \circ \underbrace{\pi \circ \rho \circ \theta}_{\text{use PLU decomposition}}$$



Elephant: Keccak χ function



(based on optimized classical Keccak χ and χ^{-1} implementations)

Overall Cipher Costs

Cipher	Block Size	Operations				Depth		Qubits
		CNOT	1QC	T	M	T	All	
Chaskey-8	128	$1.81 \cdot 2^{14}$	$1.14 \cdot 2^{13}$	$1.63 \cdot 2^{12}$	$1.75 \cdot 2^{10}$	$1.68 \cdot 2^{10}$	$1.37 \cdot 2^{14}$	160
Chaskey-12	128	$1.46 \cdot 2^{15}$	$1.82 \cdot 2^{13}$	$1.31 \cdot 2^{13}$	$1.38 \cdot 2^{11}$	$1.36 \cdot 2^{11}$	$1.11 \cdot 2^{15}$	160
PRINCE	64	$1.22 \cdot 2^{15}$	$1.60 \cdot 2^{12}$	$1.68 \cdot 2^{13}$	0	$1.41 \cdot 2^{9}$	$1.64 \cdot 2^{11}$	128
Elephant	160	$1.71 \cdot 2^{18}$	$1.17 \cdot 2^{16}$	$1.34 \cdot 2^{17}$	0	$1.56 \cdot 2^{11}$	$1.29 \cdot 2^{14}$	160
	176	$1.05 \cdot 2^{19}$	$1.45 \cdot 2^{16}$	$1.66 \cdot 2^{17}$	0	$1.76 \cdot 2^{11}$	$1.68 \cdot 2^{14}$	176
	200	$1.07 \cdot 2^{19}$	$1.08 \cdot 2^{16}$	$1.13 \cdot 2^{15}$	$1.72 \cdot 2^{12}$	$1.34 \cdot 2^{8}$	$1.29 \cdot 2^{17}$	400

"1QC" are single-qubit Clifford operations and "M" are measurements.

Conclusion

Overall Results

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

All figures in log base 2 except bitlength.

Mitigation 1: Limit Queries

The cost of the attack decreases with the number of queries (up to $\widetilde{\mathcal{O}}\left(2^{n/3}\right)$). If we limit queries:

Target	Bitlength	Offline Queries	Operations All	Operations T	Depth All	Depth T	Qubits	Query Limit
Chaskey-8	128	48	64.9	64.4	56.0	53.9	14.5	limited
Chaskey-12	128	48	65.1	64.5	56.4	54.1	14.5	
Chaskey-8	128	50	64.3	64.0	55.5	54.4	14.5	unlimited
Chaskey-12	128	51	64.5	64.2	55.9	55.2	14.5	

All figures in log base 2 except bitlength.

Mitigation 1: Limit Queries

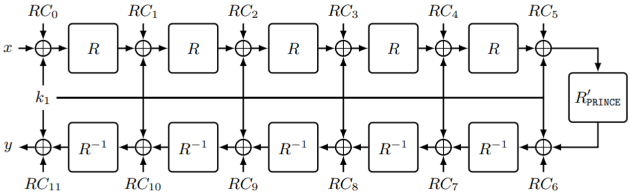
The cost of the attack decreases with the number of queries (up to $\widetilde{\mathcal{O}}\left(2^{n/3}\right)$). If we limit queries:

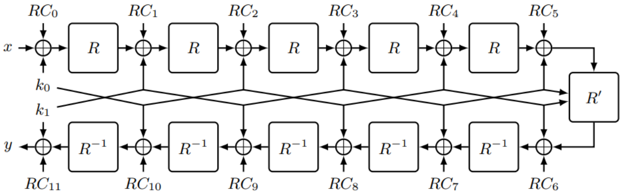
Target	Bitlength	Offline Queries	Operations All	Operations T	Depth All	Depth T	Qubits	Query Limit
Elephant	160	47	84.1	82.5	72.6	70.4	14.8	limited
	176	47	92.5	90.9	80.8	78.5	15.1	
	200	69	93.6	91.7	83.7	79.3	16.4	
Elephant	160	63	76.9	76.3	67.3	67.1	14.8	unlimited
	176	68	82.6	81.7	72.4	72.1	15.1	
	200	76	90.7	89.7	81.1	80.1	16.4	

All figures in log base 2 except bitlength.

Mitigation 2: Change the cipher

PRINCEv2 uses a different construction and is immune:





Mitigation 3: Larger State Sizes

Target	Bitlength	Offline Queries	Operations All	Operations T	Depth All	Depth T	Qubits	Attack
Elephant	160	47	84.1	82.5	72.6	70.4	14.8	Offline Simon
	176	47	92.5	90.9	80.8	78.5	15.1	
	200	69	93.6	91.7	83.7	79.3	16.4	
Elephant	160	0	85.1	83.1	80.2	77.3	9.6	Exhaustive quantum key search
	176	0	85.4	83.4	80.4	77.5	9.8	
	200	0	85.1	81.0	83.0	74.0	10.0	

All figures in log base 2 except bitlength.

Mitigation 3: Larger State Sizes

Target	Bitlength	Offline Queries	Operations All	Operations T	Depth All	Depth T	Qubits	Attack
Elephant	160	47	84.1	82.5	72.6	70.4	14.8	Offline Simon
	176	47	92.5	90.9	80.8	78.5	15.1	
	200	69	93.6	91.7	83.7	79.3	16.4	
Elephant	160	0	85.1	83.1	80.2	77.3	9.6	Exhaustive quantum key search
	176	0	85.4	83.4	80.4	77.5	9.8	
	200	0	85.1	81.0	83.0	74.0	10.0	

All figures in log base 2 except bitlength.

- Elephant needs an increase to *both* key and state size to increase quantum security.

Conclusion: Thanks for listening!

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

All figures in log base 2 except bitlength.