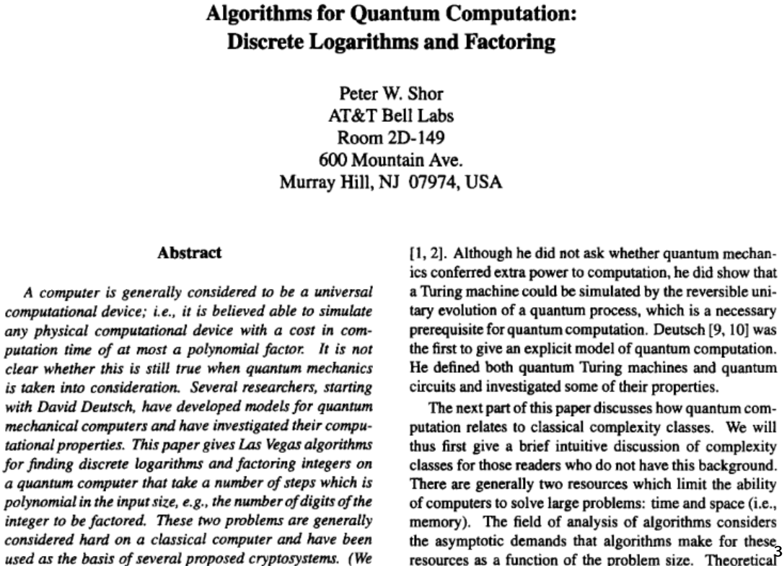
Quantum resource estimation to break cryptographic schemes

Gustavo Banegas[1]

ACCESS Seminar
May 18, 2020

[1]INRIA & LIX - École polytechnique, France
gustavo@cryptme.in

Outline


Algorithms for Quantum Computation: Discrete Logarithms and Factoring

Peter W. Shor
AT&T Bell Labs
Room 2D-149
600 Mountain Ave.
Murray Hill, NJ 07974, USA

Abstract

A computer is generally considered to be a universal computational device; i.e., it is believed able to simulate any physical computational device with a cost in computation time of at most a polynomial factor. It is not clear whether this is still true when quantum mechanics is taken into consideration. Several researchers, starting with David Deutsch, have developed models for quantum mechanical computers and have investigated their computational properties. This paper gives Las Vegas algorithms for finding discrete logarithms and factoring integers on a quantum computer that take a number of steps which is polynomial in the input size, e.g., the number of digits of the integer to be factored. These two problems are generally considered hard on a classical computer and have been used as the basis of several proposed cryptosystems. (We

[1, 2]. Although he did not ask whether quantum mechanics conferred extra power to computation, he did show that a Turing machine could be simulated by the reversible unitary evolution of a quantum process, which is a necessary prerequisite for quantum computation. Deutsch [9, 10] was the first to give an explicit model of quantum computation. He defined both quantum Turing machines and quantum circuits and investigated some of their properties.

The next part of this paper discusses how quantum computation relates to classical complexity classes. We will thus first give a brief intuitive discussion of complexity classes for those readers who do not have this background. There are generally two resources which limit the ability of computers to solve large problems: time and space (i.e., memory). The field of analysis of algorithms considers the asymptotic demands that algorithms make for these resources as a function of the problem size. Theoretical

Cryptoapocalypse

A fast quantum mechanical algorithm for database search

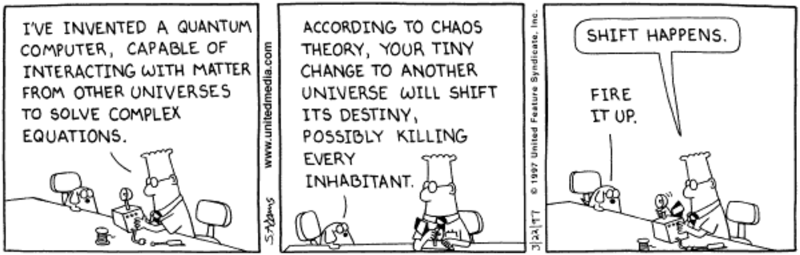
Lov K. Grover
3C-404A, AT&T Bell Labs
600 Mountain Avenue
Murray Hill NJ 07974
lkg@mhcnet.att.com

Summary

An unsorted database contains N records, of which just one satisfies a particular property. The problem is to identify that one record. Any classical algorithm, deterministic or probabilistic, will clearly take $O(N)$ steps since on the average it will have to examine a large fraction of the N records. Quantum mechanical systems can do several operations simultaneously due to their wave like properties. This paper gives an $O(\sqrt{N})$ step quantum mechanical algorithm for identifying that record. It is within a constant factor of the fastest possible quantum mechanical algorithm.

This paper applies quantum computing to a mundane problem in information processing and presents an algorithm that is significantly faster than any classical algorithm can be. The problem is this: there is an unsorted database containing N items out of which just one item satisfies a given condition - that one item has to be retrieved. Once an item is examined, it is possible to tell whether or not it satisfies the condition in one step. However, there does not exist any sorting on the database that would aid its selection. The most efficient classical algorithm for this is to examine the items in the database one by one. If an item satisfies the required condition stop; if it does not, keep track of this item so that it is not examined again. It is easily seen

In other words..



Introduction to Quantum Computing

How a quantum computer works?

- It performs computations based on **probabilities** of an object's state before it is measured;

Introduction to Quantum Computing

How a quantum computer works?

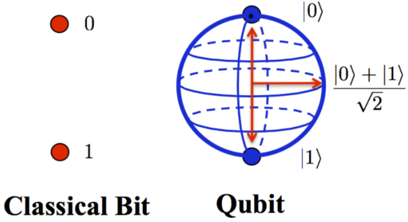
- It performs computations based on **probabilities** of an object's state before it is measured;
- We can change the probabilities of a **state**;

Quantum Computation - qubits

Qubit vs Classical bit

Quantum Computation - qubits

Qubit vs Classical bit



$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$\alpha |0\rangle + \beta |1\rangle ,$$

$$|\alpha|^2 + |\beta|^2 = 1$$

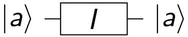
.

Measuring quantum state

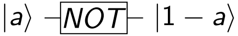
Measuring collapses the state.

Quantum gates

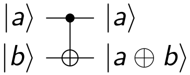
Identity gate:



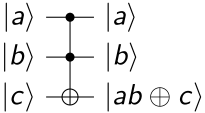
NOT gate:



CNOT gate:



Toffoli gate:



Quantum gates

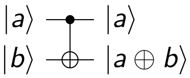
Identity gate:

$$|a\rangle \; - \boxed{\;I\;} - \; |a\rangle$$

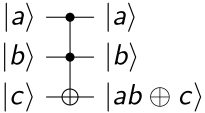
NOT gate:

$$|a\rangle \; - \boxed{NOT} - \; |1 - a\rangle$$

CNOT gate:



Toffoli gate:



Hadamard Gate:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

$$|b\rangle \; - \boxed{H} - \; \frac{(|0\rangle + (-1)^b |1\rangle)}{\sqrt{2}}$$

$$|b\rangle \; - \boxed{H} - \boxed{H} - \; |b\rangle$$

n-Qubit system

Definition

$|\psi\rangle \in \mathbb{C}^2$ such that $|| \, |\psi\rangle \, || = 1$,

$$|\psi\rangle = \sum_{x \in \{0,1\}^n} \alpha_x |x\rangle$$

where

$$\sum_{x \in \{0,1\}^n} |\alpha_x|^2 = 1.$$

Example 2-qubit system

- 4 basis states: $|0\rangle \otimes |0\rangle$, $|0\rangle \otimes |1\rangle$,$|1\rangle \otimes |0\rangle$, $|1\rangle \otimes |1\rangle$.
- It is common to use just: $|0\rangle \, |1\rangle$,$|10\rangle$

Quantum computation and reversibility

Reversibility

Quantum evolution is unitary (or any operation that changes the state needs to be unitary);
Unitary means:

$$UU^\dagger = U^\dagger U = I$$

Quantum computation and reversibility

Reversibility

Quantum evolution is unitary (or any operation that changes the state needs to be unitary);
Unitary means:

$$UU^\dagger = U^\dagger U = I$$

A unitary transformation taking basis states to basis states must be a permutation.
if $U\left|x\right\rangle = \left|u\right\rangle$ and $U\left|y\right\rangle = \left|u\right\rangle$, then $\left|x\right\rangle = U^{-1}\left|u\right\rangle = \left|y\right\rangle$.
Therefore quantum mechanics imposes the constraint that classically it must be reversible computation.

Computing functions

Using classical computers

Example to compute $H(H(H(H(x)))) = H^4(x)$:

Computing functions

Using classical computers

Example to compute $H(H(H(H(x)))) = H^4(x)$:

Time	$R0$	$R1$	$R2$

Computing functions

Using classical computers

Example to compute $H(H(H(H(x)))) = H^4(x)$:

Time	$R0$	$R1$	$R2$
time 0:	x	0	0

Computing functions

Using classical computers

Example to compute $H(H(H(H(x)))) = H^4(x)$:

Time	$R0$	$R1$	$R2$
time 0:	x	0	0
time 1:	x	0	$H(x)$

Computing functions

Using classical computers

Example to compute $H(H(H(H(x)))) = H^4(x)$:

Time	$R0$	$R1$	$R2$
time 0:	x	0	0
time 1:	x	0	$H(x)$
time 2:	x	0	$H^2(x)$

Computing functions

Using classical computers

Example to compute $H(H(H(H(x)))) = H^4(x)$:

$Time$	$R0$	$R1$	$R2$
time 0:	x	0	0
time 1:	x	0	$H(x)$
time 2:	x	0	$H^2(x)$
time 3:	x	0	$H^3(x)$

Computing functions

Using classical computers

Example to compute $H(H(H(H(x)))) = H^4(x)$:

Time	$R0$	$R1$	$R2$
time 0:	x	0	0
time 1:	x	0	$H(x)$
time 2:	x	0	$H^2(x)$
time 3:	x	0	$H^3(x)$
time 4:	x	$H^4(x)$	$H^3(x)$

Computing functions

Using classical computers

Example to compute $H(H(H(H(x)))) = H^4(x)$:

Time	$R0$	$R1$	$R2$
time 0:	x	0	0
time 1:	x	0	$H(x)$
time 2:	x	0	$H^2(x)$
time 3:	x	0	$H^3(x)$
time 4:	x	$H^4(x)$	$H^3(x)$

Total: 5 Computations of H and only 3 registers.

Computing functions

Make it reversible with Bennett–Tompa's method

Example to compute $H^4(x)$:

Computing functions

Make it reversible with Bennett–Tompa's method

Example to compute $H^4(x)$:

$Time$	$R0$	$R1$	$R2$	$R3$	$R4$

Computing functions

Make it reversible with Bennett–Tompa's method

Example to compute $H^4(x)$:

Time	$R0$	$R1$	$R2$	$R3$	$R4$
time 0:	x	0	0	0	0

Computing functions

Make it reversible with Bennett–Tompa's method

Example to compute $H^4(x)$:

Time	$R0$	$R1$	$R2$	$R3$	$R4$
time 0:	x	0	0	0	0
time 1:	x	0	$H(x)$	0	0

Computing functions

Make it reversible with Bennett–Tompa's method

Example to compute $H^4(x)$:

Time	$R0$	$R1$	$R2$	$R3$	$R4$
time 0:	x	0	0	0	0
time 1:	x	0	$H(x)$	0	0
time 2:	x	0	$H(x)$	$H^2(x)$	0

Computing functions

Make it reversible with Bennett–Tompa's method

Example to compute $H^4(x)$:

Time	$R0$	$R1$	$R2$	$R3$	$R4$
time 0:	x	0	0	0	0
time 1:	x	0	$H(x)$	0	0
time 2:	x	0	$H(x)$	$H^2(x)$	0
time 3:	x	0	$H(x)$	$H^2(x)$	$H^3(x)$

Computing functions

Make it reversible with Bennett–Tompa's method

Example to compute $H^4(x)$:

Time	$R0$	$R1$	$R2$	$R3$	$R4$
time 0:	x	0	0	0	0
time 1:	x	0	$H(x)$	0	0
time 2:	x	0	$H(x)$	$H^2(x)$	0
time 3:	x	0	$H(x)$	$H^2(x)$	$H^3(x)$
time 4:	x	$H^4(x)$	$H(x)$	$H^2(x)$	$H^3(x)$

Computing functions

Make it reversible with Bennett–Tompa's method

Example to compute $H^4(x)$:

Time	$R0$	$R1$	$R2$	$R3$	$R4$
time 0:	x	0	0	0	0
time 1:	x	0	$H(x)$	0	0
time 2:	x	0	$H(x)$	$H^2(x)$	0
time 3:	x	0	$H(x)$	$H^2(x)$	$H^3(x)$
time 4:	x	$H^4(x)$	$H(x)$	$H^2(x)$	$H^3(x)$
time 5:	x	$H^4(x)$	$H(x)$	$H^2(x)$	0

Computing functions

Make it reversible with Bennett–Tompa's method

Example to compute $H^4(x)$:

Time	$R0$	$R1$	$R2$	$R3$	$R4$
time 0:	x	0	0	0	0
time 1:	x	0	$H(x)$	0	0
time 2:	x	0	$H(x)$	$H^2(x)$	0
time 3:	x	0	$H(x)$	$H^2(x)$	$H^3(x)$
time 4:	x	$H^4(x)$	$H(x)$	$H^2(x)$	$H^3(x)$
time 5:	x	$H^4(x)$	$H(x)$	$H^2(x)$	0
time 6:	x	$H^4(x)$	$H(x)$	0	0

Computing functions

Make it reversible with Bennett–Tompa's method

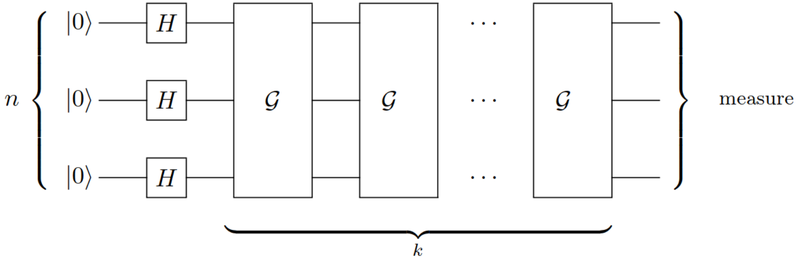
Example to compute $H^4(x)$:

Time	$R0$	$R1$	$R2$	$R3$	$R4$
time 0:	x	0	0	0	0
time 1:	x	0	$H(x)$	0	0
time 2:	x	0	$H(x)$	$H^2(x)$	0
time 3:	x	0	$H(x)$	$H^2(x)$	$H^3(x)$
time 4:	x	$H^4(x)$	$H(x)$	$H^2(x)$	$H^3(x)$
time 5:	x	$H^4(x)$	$H(x)$	$H^2(x)$	0
time 6:	x	$H^4(x)$	$H(x)$	0	0
time 7:	x	$H^4(x)$	0	0	0

Total: 8 Computations of H and 5 registers.

Grover's Algorithm

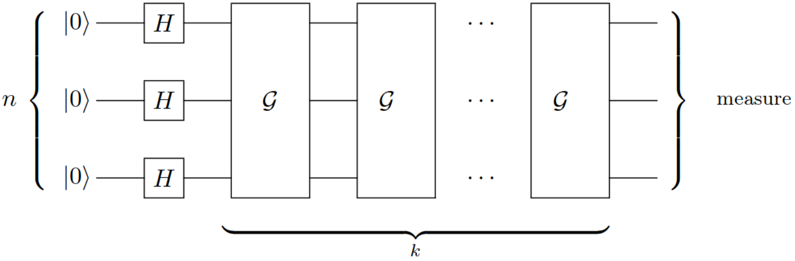
Grover's algorithm in a nutshell



- Needs $k = O(\sqrt{N})$ queries in database of size $N = 2^n$ elements;
- $\mathcal{G}$ is the "Grover" step, and it consists of evaluating a search function.

Grover's Algorithm

Grover's algorithm in a nutshell



- Needs $k = O(\sqrt{N})$ queries in database of size $N = 2^n$ elements;
- $\mathcal{G}$ is the "Grover" step, and it consists of evaluating a search function.

Grover's Algorithm

Grover's algorithm in a nutshell

Given a function $f : \{0,1\}^* \rightarrow \{0,1\}$, we can define Grover's algorithm as:

Grover's Algorithm

Grover's algorithm in a nutshell

Given a function $f : \{0,1\}^* \to \{0,1\}$, we can define Grover's algorithm as:

Grover(f):

1. Start with $|\phi_0\rangle = |1^n\rangle$
2. Apply $\mathbf{H}^{\otimes n}$
3. Repeat $O\left(\sqrt{2^n}\right)$ times
4. Quantum evaluation of f with oracle $\mathcal{O}_f$
5. Amplification;
6. Measure $x = |\phi\rangle$ and return $f(x) = 1$.

The function f can be a preimage search of a hash function or a key search.

Quantum AES

AES in quantum gates

- All the operations can only be build using quantum gates;

Quantum AES

AES in quantum gates

- All the operations can only be build using quantum gates;
- It needs to be reversible;

Quantum AES

AES in quantum gates

- All the operations can only be build using quantum gates;
- It needs to be reversible;
- Lower depth and low amount of qubits.

Quantum AES

AES in quantum gates

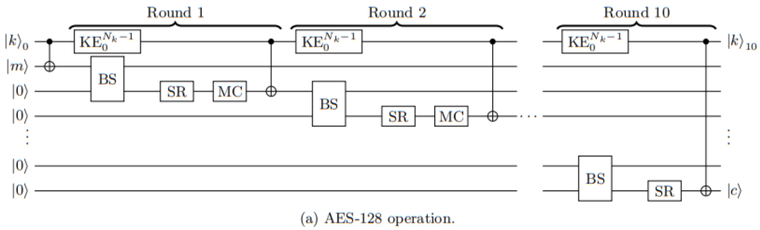


Figure: Complete AES-128. Figure from JNRV[2]

[2]Samuel Jaques, Michael Naehrig, Martin Roetteler and Fernando Virdia. Implementing Grover oracles for quantum key search on AES and LowMC. In EUROCRYPT 2020, 2020.

Quantum AES

AES in quantum gates

Even small details can decrease number of operations.

[3] Markus Grassl, Brandon Langenberg, Martin Roetteler, and Rainer Steinwandt. Applying Grovers algorithm to AES: quantum resource estimates. In PQCRYPTO 16. Springer, 2016.

Quantum AES

AES in quantum gates

Even small details can decrease number of operations.
For example: the SubByte function, looking only for the multiplicative inverse. The authors in GLRS[3] compute as:

$$\alpha^{-1} = \alpha^{254} = ((\alpha \cdot \alpha^2) \cdot (\alpha \cdot \alpha^2)^4 \cdot (\alpha \cdot \alpha^2)^{16} \cdot \alpha^{64})^2.$$

[3] Markus Grassl, Brandon Langenberg, Martin Roetteler, and Rainer Steinwandt. Applying Grovers algorithm to AES: quantum resource estimates. In PQCRYPTO 16. Springer, 2016.

Quantum AES

Reversible computation of $\alpha^{-1} = \alpha^{254}$ proposed in GLRS

(*)- multiplication between two values occurs, (^)- squaring or multi-squaring occurs, (^^) when an out-of-place squaring or multi-squaring.

Step \ Qubits position	0…7	8…15	16…23	24…31	32…39
0	α				
1 ^^	α	α^2			
2 *	α	α^2	α^3		
3 ^^	α	α^2	α^3	α^{12}	
4 *	α	α^2	α^3	α^{12}	α^{15}
5 *	α	α^2		α^{12}	α^{15}
6 ^	α	α^2		α^{48}	α^{15}
7 *	α	α^2	α^{63}	α^{48}	α^{15}
8 ^^	α		α^{63}	α^{48}	α^{15}
9 ^	α^{64}		α^{63}	α^{48}	α^{15}
10 *	α^{64}	α^{127}	α^{63}	α^{48}	α^{15}
11 ^	α^{64}	α^{254}	α^{63}	α^{48}	α^{15}
12 ^	α	α^{254}	α^{63}	α^{48}	α^{15}
13 *	α	α^{254}		α^{48}	α^{15}
14 ^^	α	α^{254}	α^3	α^{48}	α^{15}
15 ^	α	α^{254}	α^3	α^{12}	α^{15}
16 *	α	α^{254}	α^3	α^{12}	
17 ^^	α	α^{254}	α^3		
18 ^^	α	α^{254}	α^3	α^2	
19 *	α	α^{254}		α^2	
20 ^^	α	α^{254}			

Quantum AES

Reversible computation of $\alpha^{-1} = \alpha^{254}$ proposed in GLRS

(*)- multiplication between two values occurs, (^)- squaring or multi-squaring occurs, (^^) when an out-of-place squaring or multi-squaring.

Step \ Qubits position	$0\ldots7$	$8\ldots15$	$16\ldots23$	$24\ldots31$	$32\ldots39$
0	α				
1 ^^	α	α^2			
2 *	α	α^2	α^3		
3 ^^	α	α^2	α^3	α^{12}	
4 *	α	α^2	α^3	α^{12}	α^{15}
5 *	α	α^2		α^{12}	α^{15}
6 ^	α	α^2		α^{48}	α^{15}
7 *	α	α^2	α^{63}	α^{48}	α^{15}
8 ^^	α		α^{63}	α^{48}	α^{15}
9 ^	α^{64}		α^{63}	α^{48}	α^{15}
10 *	α^{64}	α^{127}	α^{63}	α^{48}	α^{15}
11 ^	α^{64}	α^{254}	α^{63}	α^{48}	α^{15}
12 ^	α	α^{254}	α^{63}	α^{48}	α^{15}
13 *	α	α^{254}		α^{48}	α^{15}
14 ^^	α	α^{254}	α^3	α^{48}	α^{15}
15 ^	α	α^{254}	α^3	α^{12}	α^{15}
16 *	α	α^{254}	α^3	α^{12}	
17 ^^	α	α^{254}	α^3		
18 ^^	α	α^{254}	α^3	α^2	
19 *	α	α^{254}		α^2	
20 ^^	α	α^{254}			

8 multiplications and 29 squarings.

Quantum AES

Better way to compute $\alpha^{-1} = \alpha^{254}$

(*)- multiplication between two values occurs, (^)- squaring or multi-squaring occurs, (^^) when an out-of-place squaring or multi-squaring.

Step \ Qubits position	$0\ldots7$	$8\ldots15$	$16\ldots23$	$24\ldots31$	$32\ldots39$
0	α				
1 ^^	α	α^2			
2 *	α	α^2	α^3		
3 ^^	α	α^2	α^3	α^{12}	
4 ^^	α		α^3	α^{12}	
5 *	α	α^{15}	α^3	α^{12}	
6 ^	α	α^{60}	α^3	α^{12}	
7 ^^	α	α^{60}	α^3		
8 *	α	α^{60}	α^3	α^{63}	
9 ^	α	α^{60}	α^3	α^{126}	
10 *	α	α^{60}	α^3	α^{126}	α^{127}
11 ^	α	α^{60}	α^3	α^{126}	α^{254}
12 ^	α	α^{60}	α^3	α^{63}	α^{254}
13 *	α	α^{60}	α^3		α^{254}
14 ^^	α	α^{60}	α^3	α^{12}	α^{254}
15 ^	α	α^{15}	α^3	α^{12}	α^{254}
16*	α		α^3	α^{12}	α^{254}
17 ^^	α	α^2	α^3	α^{12}	α^{254}
18 ^^	α	α^2	α^3		α^{254}
19 *	α	α^2			α^{254}
20 ^^	α				α^{254}

Quantum AES

Better way to compute $\alpha^{-1} = \alpha^{254}$

(*)- multiplication between two values occurs, (^)- squaring or multi-squaring occurs, (^^) when an out-of-place squaring or multi-squaring.

Step \ Qubits position	$0\dots7$	$8\dots15$	$16\dots23$	$24\dots31$	$32\dots39$
0	α				
1 ^^	α	α^2			
2 *	α	α^2	α^3		
3 ^^	α	α^2	α^3	α^{12}	
4 ^^	α		α^3	α^{12}	
5 *	α	α^{15}	α^3	α^{12}	
6 ^	α	α^{60}	α^3	α^{12}	
7 ^^	α	α^{60}	α^3		
8 *	α	α^{60}	α^3	α^{63}	
9 ^	α	α^{60}	α^3	α^{126}	
10 *	α	α^{60}	α^3	α^{126}	α^{127}
11 ^	α	α^{60}	α^3	α^{126}	α^{254}
12 ^	α	α^{60}	α^3	α^{63}	α^{254}
13 *	α	α^{60}	α^3		α^{254}
14 ^^	α	α^{60}	α^3	α^{12}	α^{254}
15 ^	α	α^{15}	α^3	α^{12}	α^{254}
16*	α		α^3	α^{12}	α^{254}
17 ^^	α	α^2	α^3	α^{12}	α^{254}
18 ^^	α	α^2	α^3		α^{254}
19 *	α	α^2			α^{254}
20 ^^	α				α^{254}

7 multiplications and 19 squarings.

Quantum AES

AES in quantum gates

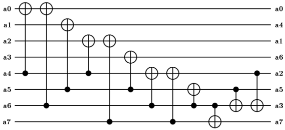


Figure: Squaring in $F_2[x]/x^8 + x^4 + x^3 + x + 1$

Quantum AES

AES in quantum gates

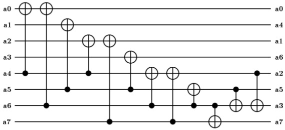


Figure: Squaring in $F_2[x]/x^8 + x^4 + x^3 + x + 1$

This circuit requires 12 CNOTS

Quantum AES

AES in quantum gates

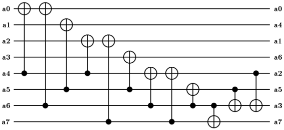


Figure: Squaring in $F_2[x]/x^8 + x^4 + x^3 + x + 1$

This circuit requires 12 CNOTS

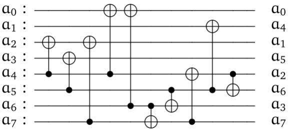


Figure: Improved Squaring in $F_2[x]/x^8 + x^4 + x^3 + x + 1$.

Quantum AES

AES in quantum gates

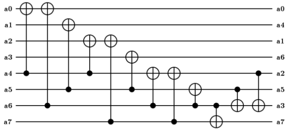


Figure: Squaring in $F_2[x]/x^8 + x^4 + x^3 + x + 1$

This circuit requires 12 CNOTS

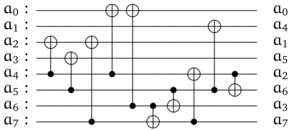


Figure: Improved Squaring in $F_2[x]/x^8 + x^4 + x^3 + x + 1$.

This circuit requires 10 CNOTS

Grover's algorithm for breaking AES

Quantum Resources

Table: Number of gates for running Grover's algorithm on AES-128

	Cliff +CNOT	T gates
GLRS[4]	$1.55 \cdot 2^{86}$	$1.19 \cdot 2^{86}$
LPS[5]	$1.46 \cdot 2^{82}$	$1.47 \cdot 2^{81}$
JNRV[6]	$1.13 \cdot 2^{82}$	$1.32 \cdot 2^{78}$

[4] Markus Grassl, Brandon Langenberg, Martin Roetteler, and Rainer Steinwandt. Applying Grovers algorithm to AES: quantum resource estimates. In PQCRYPTO 16. Springer, 2016.

[5] Brandon Langenberg, Hai Pham, and Rainer Steinwandt. Reducing the cost of implementing AES as a quantum circuit. Cryptology ePrint Archive, Report 2019/854, 2019.

[6] Samuel Jaques, Michael Naehrig, Martin Roetteler and Fernando Virdia. Implementing Grover oracles for quantum key search on AES and LowMC. In EUROCRYPT 2020, 2020.

Shor's algorithm

In summary Shor's algorithm has two parts:

- A reduction of the factoring problem to the problem of **order-finding**, which can be done on a classical computer;

Shor's algorithm

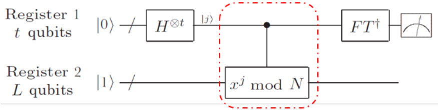
In summary Shor's algorithm has two parts:

- A reduction of the factoring problem to the problem of **order-finding**, which can be done on a classical computer;
- A quantum algorithm to solve the **order-finding** problem.

Shor's algorithm

A toy example can be when we have $N = 15$. Let's see how Shor's algorithm works:

1. Select an arbitrary number, such as $a = 2$ (< 15)
2. $gcd(a, N) = gcd(2, 15) = 1$
3. Find the period of function $f(x) = a^x \mod N$, which satisfies $f(x + r) = f(x)$;
4. Get $r = 4$ through the circuit below;
5. $gcd(a^{\frac{r}{2}} + 1, N) = gcd(5, 15) = 3$;
6. $gcd(a^{\frac{r}{2}} - 1, N) = gcd(3, 15) = 5$;
7. For $N = 15$, the two decomposed prime numbers are 3 and 5.



Ressource Estimation

Break RSA (Integer Factoring)

From Gidney & Ekerå(2019)[7] uses "$3n + 0.002 n lg(n)$ logical qubits, $0.3n^3 + 0.0005n^3 lg(n)$ Toffolis, and $500n^2 + n^2 lg(n)$ measurement depth to factor n-bit RSA integers"

RSA Bits	Qubits	Toffoli + T Gates (billions)
1024	3092	0.4
2048	6189	2.7
3072	9287	9.9

[7] Craig Gidney and Martin Ekerå. How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits. arXiv preprint quant-ph/1904.09749, 2019. https://arxiv.org/abs/1905.09749

Introduction to Binary ECC

Basic overview

- Binary elliptic curves are elliptic curves defined over a binary field $\mathbb{F}_{2^n}$;

Introduction to Binary ECC

Basic overview

- Binary elliptic curves are elliptic curves defined over a binary field $\mathbb{F}_{2^n}$;
- We use polynomial representation and the operations are in $\mathbb{F}_2$ since $\mathbb{F}_{2^n} \cong \mathbb{F}_2[z]/(m(z))$, where $m(z)$ is an irreducible polynomial of degree n;

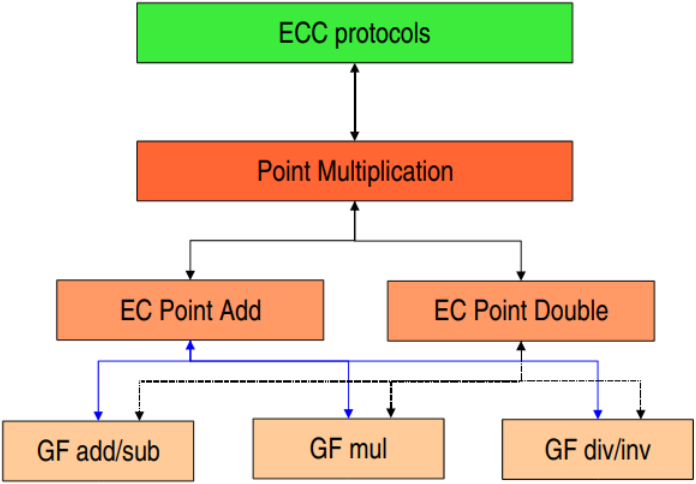
Introduction to Binary ECC

Basic overview

- Binary elliptic curves are elliptic curves defined over a binary field $\mathbb{F}_{2^n}$;
- We use polynomial representation and the operations are in $\mathbb{F}_2$ since $\mathbb{F}_{2^n} \cong \mathbb{F}_2[z]/(m(z))$, where $m(z)$ is an irreducible polynomial of degree n;
- All computations are done mod $m(z)$.

Introduction to Binary ECC

Basic overview of operations



Introduction to Binary ECC

Hardness of ECC

Public Parameters: Point $\mathbf{P} \in E_p$	
Alice	**Bob**
$\alpha \xleftarrow{\$} \mathbb{F}_p$	$\beta \xleftarrow{\$} \mathbb{F}_p$

Introduction to Binary ECC

Hardness of ECC

Public Parameters: Point $\mathbf{P} \in E_p$	
Alice	**Bob**
$\alpha \xleftarrow{\$} \mathbb{F}_p$	$\beta \xleftarrow{\$} \mathbb{F}_p$
Computes:	Computes:
$\mathbf{P}_\alpha = \alpha\mathbf{P}$	$\mathbf{P}_\beta = \beta\mathbf{P}$

Introduction to Binary ECC

Hardness of ECC

Public Parameters: Point $\mathbf{P} \in E_p$		
Alice		**Bob**
$\alpha \xleftarrow{\$} \mathbb{F}_p$		$\beta \xleftarrow{\$} \mathbb{F}_p$
Computes:		Computes:
$\mathbf{P}_\alpha = \alpha\mathbf{P}$		$\mathbf{P}_\beta = \beta\mathbf{P}$
	$\xrightarrow{\mathbf{P}_\alpha}$	
	$\xleftarrow{\mathbf{P}_\beta}$	

Introduction to Binary ECC

Hardness of ECC

Public Parameters: Point $\mathbf{P} \in E_p$

Alice		**Bob**
$\alpha \xleftarrow{\$} \mathbb{F}_p$		$\beta \xleftarrow{\$} \mathbb{F}_p$
Computes:		Computes:
$\mathbf{P}_\alpha = \alpha\mathbf{P}$		$\mathbf{P}_\beta = \beta\mathbf{P}$
	$\xrightarrow{\mathbf{P}_\alpha}$	
	$\xleftarrow{\mathbf{P}_\beta}$	
Computes:		Computes:
$\mathbf{P}_{\alpha\beta} = \alpha\mathbf{P}_\beta$		$\mathbf{P}_{\alpha\beta} = \beta\mathbf{P}_\alpha$

Public Parameters: Point $\mathbf{P} \in E_p$

Alice		**Bob**
$\alpha \xleftarrow{\$} \mathbb{F}_p$		$\beta \xleftarrow{\$} \mathbb{F}_p$
Computes:		Computes:
$\mathbf{P}_\alpha = \alpha\mathbf{P}$		$\mathbf{P}_\beta = \beta\mathbf{P}$
	$\xrightarrow{\mathbf{P}_\alpha}$	
	$\xleftarrow{\mathbf{P}_\beta}$	
Computes:		Computes:
$\mathbf{P}_{\alpha\beta} = \alpha\mathbf{P}_\beta$		$\mathbf{P}_{\alpha\beta} = \beta\mathbf{P}_\alpha$

Alice and Bob have the shared point $\mathbf{P}_{\alpha\beta} = [\alpha \cdot \beta]\mathbf{P}$

Shor's circuit for Binary Elliptic Curves

- Implementation of a quantum circuit for the inversion using GCD and FLT (Fermat's little theorem);

[8] Iggy van Hoof. Space-efficient quantum multiplication of polynomials for binary finite fields with sub-quadratic Toffoli gate count.Quantum Information & Computation, pages 721–735, 2020.https://arxiv.org/abs/1910.02849.

Shor's circuit for Binary Elliptic Curves

- Implementation of a quantum circuit for the inversion using GCD and FLT (Fermat's little theorem);
  - We use for multiplication Karatsuba from Iggy's paper[8];

[8]Iggy van Hoof. Space-efficient quantum multiplication of polynomials for binary finite fields with sub-quadratic Toffoli gate count.Quantum Information & Computation, pages 721–735, 2020.https://arxiv.org/abs/1910.02849.

Shor's circuit for Binary Elliptic Curves

- Implementation of a quantum circuit for the inversion using GCD and FLT (Fermat's little theorem);
  - We use for multiplication Karatsuba from Iggy's paper[8];
  - The GCD-based inversion performed better in number of qubits and gates.

[8] Iggy van Hoof. Space-efficient quantum multiplication of polynomials for binary finite fields with sub-quadratic Toffoli gate count.Quantum Information & Computation, pages 721–735, 2020.https://arxiv.org/abs/1910.02849.

Shor's circuit for Binary Elliptic Curves

- Implementation of a quantum circuit for the inversion using GCD and FLT (Fermat's little theorem);
  - We use for multiplication Karatsuba from Iggy's paper[8];
  - The GCD-based inversion performed better in number of qubits and gates.
- Implementation of quantum Point addition and Point "doubling";

[8] Iggy van Hoof. Space-efficient quantum multiplication of polynomials for binary finite fields with sub-quadratic Toffoli gate count.Quantum Information & Computation, pages 721–735, 2020.https://arxiv.org/abs/1910.02849.

Shor's circuit for Binary Elliptic Curves

- Implementation of a quantum circuit for the inversion using GCD and FLT (Fermat's little theorem);
  - We use for multiplication Karatsuba from Iggy's paper[8];
  - The GCD-based inversion performed better in number of qubits and gates.
- Implementation of quantum Point addition and Point "doubling";
- Present the a quantum version of "window" addition;

[8] Iggy van Hoof. Space-efficient quantum multiplication of polynomials for binary finite fields with sub-quadratic Toffoli gate count.Quantum Information & Computation, pages 721–735, 2020.https://arxiv.org/abs/1910.02849.

Shor's circuit for Binary Elliptic Curves

- Implementation of a quantum circuit for the inversion using GCD and FLT (Fermat's little theorem);
  - We use for multiplication Karatsuba from Iggy's paper[8];
  - The GCD-based inversion performed better in number of qubits and gates.
- Implementation of quantum Point addition and Point "doubling";
- Present the a quantum version of "window" addition;
- Q# implementation of Karatsuba and other functions;

[8] Iggy van Hoof. Space-efficient quantum multiplication of polynomials for binary finite fields with sub-quadratic Toffoli gate count.Quantum Information & Computation, pages 721–735, 2020.https://arxiv.org/abs/1910.02849.

Shor's circuit for Binary Elliptic Curves

- Implementation of a quantum circuit for the inversion using GCD and FLT (Fermat's little theorem);
  - We use for multiplication Karatsuba from Iggy's paper[8];
  - The GCD-based inversion performed better in number of qubits and gates.
- Implementation of quantum Point addition and Point "doubling";
- Present the a quantum version of "window" addition;
- Q# implementation of Karatsuba and other functions;
- All the details will be present at CHES 2021.

[8] Iggy van Hoof. Space-efficient quantum multiplication of polynomials for binary finite fields with sub-quadratic Toffoli gate count.Quantum Information & Computation, pages 721–735, 2020.https://arxiv.org/abs/1910.02849.

Break Binary ECC (DLP)

From Banegas, Bernstein, van Hoof and Lange(2021)[9], we require the following amount of ressources:

		Single step			Total
n	qubits	TOF gates	CNOT gates	depth upper bound	TOF gates
163	1,157	893,585	827,379	1,262,035	293,095,880
233	1,647	1,669,299	1,614,947	2,405,889	781,231,932
283	1,998	2,427,369	2,358,734	3,503,510	1,378,745,592
571	4,015	8,987,401	9,080,190	13,237,682	10,281,586,744

[9] Banegas, G., Bernstein, D. J., van Hoof, I., Lange, T. Concrete quantum cryptanalysis of binary elliptic curves. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2021(1)

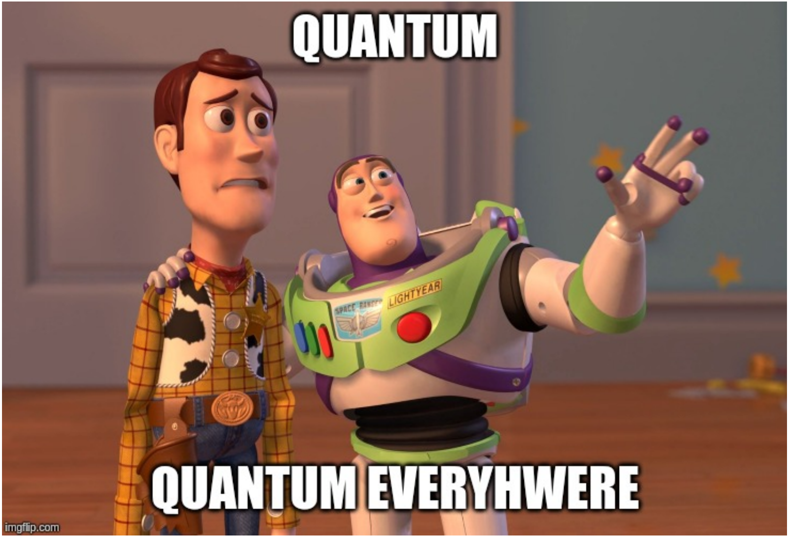
Break Binary ECC (DLP)

From Banegas, Bernstein, van Hoof and Lange(2021)[9], we require the following amount of ressources:

n	qubits	Single step TOF gates	Single step CNOT gates	Single step depth upper bound	Total TOF gates
163	1,157	893,585	827,379	1,262,035	293,095,880
233	1,647	1,669,299	1,614,947	2,405,889	781,231,932
283	1,998	2,427,369	2,358,734	3,503,510	1,378,745,592
571	4,015	8,987,401	9,080,190	13,237,682	10,281,586,744

In fact, $7n + \lfloor \log(n) \rfloor + 9$ qubits,
$48n^3 + 8n^{\log(3)+1} + 352n^2 \log(n) + 512n^2 + O(n^{\log(3)})$ Toffoli gates and $O(n^3)$ CNOT gates.

[9]Banegas, G., Bernstein, D. J., van Hoof, I., Lange, T. Concrete quantum cryptanalysis of binary elliptic curves. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2021(1)


QUANTUM
QUANTUM EVERYHWERE
imgflip.com

Questions

Thank you for your attention.
Questions?
gustavo@cryptme.in