Shor vs. RSA

Tanja Lange

Eindhoven University of Technology

SAC – Post-quantum cryptography

Schoolbook RSA encryption

1977 Rivest, Shamir, Adleman. Do not use Schoolbook RSA in practice!

Schoolbook RSA encryption

1977 Rivest, Shamir, Adleman. Do not use Schoolbook RSA in practice!

KeyGen:

1. Pick large primes $p, q; p \neq q$.
2. Compute $n = p \cdot q$, $\varphi(n) = (p-1)(q-1)$.
3. Pick $1 < e < n$ with $\gcd(e, \varphi(n)) = 1$.
4. Compute $d \equiv e^{-1} \bmod \varphi(n)$.
5. Output public key (n, e), private key (n, d).

Schoolbook RSA encryption

1977 Rivest, Shamir, Adleman. Do not use Schoolbook RSA in practice!

KeyGen:

1. Pick large primes p, q; $p \neq q$.
2. Compute $n = p \cdot q$, $\varphi(n) = (p-1)(q-1)$.
3. Pick $1 < e < n$ with $\gcd(e, \varphi(n)) = 1$.
4. Compute $d \equiv e^{-1} \bmod \varphi(n)$.
5. Output public key (n, e), private key (n, d).

Enc message $0 \leq m < n$:

1. Compute $c \equiv m^e \bmod n$.
2. Output c.

Dec ciphertext $0 \leq c < n$:

1. Compute $m' \equiv c^d \bmod n$.
2. Output m'.

Schoolbook RSA encryption

1977 Rivest, Shamir, Adleman. Do not use Schoolbook RSA in practice!

KeyGen:

1. Pick large primes p, q; $p \neq q$.
2. Compute $n = p \cdot q$, $\varphi(n) = (p-1)(q-1)$.
3. Pick $1 < e < n$ with $\gcd(e, \varphi(n)) = 1$.
4. Compute $d \equiv e^{-1} \bmod \varphi(n)$.
5. Output public key (n, e), private key (n, d).

Enc message $0 \leq m < n$:

1. Compute $c \equiv m^e \bmod n$.
2. Output c.

Dec ciphertext $0 \leq c < n$:

1. Compute $m' \equiv c^d \bmod n$.
2. Output m'.

Some k exists with $ed = 1 + k\varphi(n)$

Use Fermat's little theorem.

This works:

$$m' \equiv c^d \equiv (m^e)^d \equiv m^{ed} = m^{1+k\varphi(n)} \equiv m \cdot (m^{\varphi(n)})^k \equiv m \cdot 1 \equiv m \bmod n.$$

Shor's algorithm as a black box

- In 1994 Shor showed that quantum computers can efficiently compute the period of a function.
- He showed how to use this to solve factoring and discrete logarithms.

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

[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.

How to break RSA by finding a period?

Let $n = p \cdot q$ with p, q prime (and odd).

- Pick a with $\gcd(a, n) = 1$ (but else we have found a factor of n).
- Ask Shor for period of function

$$f_a : x \mapsto a^x \bmod n.$$

This requires a circuit for f_a for x in superposition.

How to break RSA by finding a period?

Let $n = p \cdot q$ with p, q prime (and odd).

- Pick a with $\gcd(a, n) = 1$ (but else we have found a factor of n).
- Ask Shor for period of function

$$f_a : x \mapsto a^x \bmod n.$$

  This requires a circuit for f_a for x in superposition.
- Obtain s with $f_a(x + s) = f_a(x)$ for all x.
  Note: s may be a multiple of the period of f_a.

$$a^{x+s} \equiv a^x \bmod n$$

How to break RSA by finding a period?

Let $n = p \cdot q$ with p, q prime (and odd).

- Pick a with $\gcd(a, n) = 1$ (but else we have found a factor of n).
- Ask Shor for period of function

$$f_a : x \mapsto a^x \bmod n.$$

  This requires a circuit for f_a for x in superposition.

- Obtain s with $f_a(x + s) = f_a(x)$ for all x.
  Note: s may be a multiple of the period of f_a.

$$a^{x+s} \equiv a^x \bmod n \text{ thus } a^s \equiv 1 \bmod n$$

  and we have found the order of a mod n (or a multiple thereof).

How to break RSA by finding a period?

Let $n = p \cdot q$ with p, q prime (and odd).

- Pick a with $\gcd(a, n) = 1$ (but else we have found a factor of n).
- Ask Shor for period of function

  $$f_a : x \mapsto a^x \bmod n.$$

  This requires a circuit for f_a for x in superposition.
- Obtain s with $f_a(x + s) = f_a(x)$ for all x.
  Note: s may be a multiple of the period of f_a.

  $$a^{x+s} \equiv a^x \bmod n \text{ thus } a^s \equiv 1 \bmod n$$

  and we have found the order of a mod n (or a multiple thereof).
- If s is odd, try again with a new choice of a.
- Else put $s = 2^r \cdot t$, with t odd, compute $a^t \bmod n$.
  - If $a^t \equiv \pm 1 \bmod n$ try again with a new choice of a.
  - Square the previous result.
    - If this gives -1, try again with a new choice of a.
    - If this gives 1, compute the gcd of the previous result minus 1 with n.
    - Else repeat this step.

Explanation using CRT

- There are 2 square roots of 1 in $\mathbb{F}_p$, namely ± 1.
- If $a^c \equiv 1 \bmod p$ and $a^c \equiv -1 \bmod q$ then

$$\gcd(a^c - 1, n) = p.$$

Explanation using CRT

- There are 2 square roots of 1 in $\mathbb{F}_p$, namely ± 1.
- If $a^c \equiv 1 \bmod p$ and $a^c \equiv -1 \bmod q$ then

$$\gcd(a^c - 1, n) = p.$$

- We can notice this situation by observing

$$a^{2c} \equiv 1 \bmod n \text{ and } a^c \not\equiv \pm 1 \bmod n.$$

Explanation using CRT

- There are 2 square roots of 1 in $\mathbb{F}_p$, namely ± 1.
- If $a^c \equiv 1 \bmod p$ and $a^c \equiv -1 \bmod q$ then

$$\gcd(a^c - 1, n) = p.$$

- We can notice this situation by observing

$$a^{2c} \equiv 1 \bmod n \text{ and } a^c \not\equiv \pm 1 \bmod n.$$

- We cannot find such a c by chance.
  That's where Shor's algorithm comes in:

Explanation using CRT

- There are 2 square roots of 1 in $\mathbb{F}_p$, namely ± 1.
- If $a^c \equiv 1 \bmod p$ and $a^c \equiv -1 \bmod q$ then

$$\gcd(a^c - 1, n) = p.$$

- We can notice this situation by observing

$$a^{2c} \equiv 1 \bmod n \text{ and } a^c \not\equiv \pm 1 \bmod n.$$

- We cannot find such a c by chance.
  That's where Shor's algorithm comes in:

$$a^s = a^{2^r t} \equiv 1 \bmod n$$

  means that we can look at r candidates for c.
- If s is odd (no candidates) or we encounter -1 then (a, s) does not factor n and we try again.
- Improvement: pick a with Jacobi symbol $(a|n) = -1$, so s is even.