**No. 109/19, 7–20**
**ISSN 2657-6988 (online)**
**ISSN 2657-5841 (printed)**
**DOI: 10.26408/109.01**

**Submitted: 21.08.2018**
**Accepted:15.10.2018**
**Published: 30.03.2019**

# INTEGER FACTORIZATION – CRYPTOLOGY MEETS NUMBER THEORY

**Josef Pieprzyk**
CSIRO, Sydney, Australia,
Institute of Computer Science, Polish Academy of Sciences, Warsaw, Poland,
e-mail: josef.pieprzyk@csiro.au, ORCID 0000-0002-1917-6466

**Abstract:** Integer factorization is one of the oldest mathematical problems. Initially, the interest in factorization was motivated by curiosity about behaviour of prime numbers, which are the basic building blocks of all other integers. Early factorization algorithms were not very efficient. However, this dramatically has changed after the invention of the well-known RSA public-key cryptosystem. The reason for this was simple. Finding an efficient factoring algorithm is equivalent to breaking RSA.
The work overviews development of integer factoring algorithms. It starts from the classical sieve of Eratosthenes, covers the Fermat algorithm and explains the quadratic sieve, which is a good representative of modern factoring algorithms. The progress in factoring is illustrated by examples of RSA challenge moduli, which have been factorized by groups of mathematicians and cryptographers. Shor's quantum factorization algorithm with polynomial complexity is described and the impact on public-key encryption is discussed.

**Keywords:** Cryptography, Number Theory, Public-key Cryptography, Factorization, RSA Cryptosystems, Quantum Computing, Shor Algorithm.

## 1. INTRODUCTION

Factoring or decomposition of integers into their prime factors is one the oldest mathematical problem that has been under investigation over centuries and has attracted attention of many best mathematical minds. Eratosthenes (276–194 BC) was the first mathematician known to us who designed a simple algorithm for finding prime factors. It is called *sieve of Eratosthenes* and enumerates all primes smaller than a given integer $N$. Other eminent mathematicians who made various contributions to factoring are Fermat (1607–1665) and Euler (1707–1783). Application of mechanical calculators in early 20-th century and computers in its middle gave mathematicians tools for development of new and more efficient integer factorization algorithms. But even then factoring integers larger than 100-decimal digits long was beyond anyone's dream. A significant exceleration of theory and practice of factoring is due to development of the famous RSA public key encryption algorithm [Rivest, Shamir and Adleman 1978]. It turns out that RSA

security can be easily broken if an adversary can factor the public modulus. As a result, integer factorization (which is a part of Number Theory) has also become a part of Cryptography.

Modern algorithms are able to factor integers containing more than 200-decimal digits. Despite evident progress, we still do not have polynomial-time algorithms. The best ones have sub-exponential complexity. A breakthrough has come when Shor [1997] published his quantum factorization algorithm, which is polynomial-time. This breaks RSA assuming that we are able to build quantum computers (or at least quantum factorization devices). In the work we review integer factorization algorithm and concentrate on algorithms for factoring integers in a general form (as opposed to special-form integers).

## 2. CLASSICAL ALGORITHMS

### 2.1. Sieve of Eratosthenes

The original algorithm can be used to primality testing and factoring. The version given below finds factors of a given odd integer *N*. Note that for an even integer, it is easy to divide it by a sequence of 2's so we get an odd integer. The notation $i|N$ means that integer $i$ divides $N$ (without a remainder).

**Algorithm 1:** Factoring by Sieve of Eratosthenes

**Input** : An odd integer $N$
**Output**: A set $F$ of prime factors of $N$
*Initialize the set* $F = \emptyset$;
**for** $i = 3$ **to** $\lfloor\sqrt{N}\rfloor$ **do**
  **if** $i|N$ **then**
    $F := F \cup \{i\}$;
  **end**
**end**

The algorithm runs through $\frac{\lfloor\sqrt{N}\rfloor}{2}$ steps and it is easy to see that its complexity is $O(\sqrt{N})$ or equivalent $O(2^{n/2})$, where $n = \lfloor\log_2 N\rfloor$ is the number of bits needed to represent the integer *N*. Its high (exponential) complexity restricts its application to relatively short integers (say, no longer than 20 decimal digits).

## 2.2. Fermat Algorithm

The observation made by Fermat is that it is easy to find nontrivial factors if an integer $N$ can be represented as

$$N = x^2 - y^2 = (x - y)\,(x + y)$$

Note then $p = (x - y)$ and $q = (x + y)$ are nontrivial factors of $N$.

**Algorithm 2:** Fermat Algorithm

**Input** : An odd integer $N$
**Output**: A set $F$ of two factors of $N$

```
Initialize x = ⌈√N⌉;
while α = x² − N is NOT a square root do
    x := x + 1;
end
y = √α;
F := {(x − y), (x + y)};
```

The algorithm works best if $N$ has two factors of similar sizes. Let us have a closer look at complexity of the algorithm. Let us start from rather trivial observation. The factors found by the algorithm are

$$p = x + y \text{ and } q = x - y$$

and $p > q$. The above relations can be represented as follows:

$$x = \frac{p+q}{2} \quad \text{and} \quad y = \frac{p-q}{2}$$

Note the algorithm exits the while loop, when $x = \frac{p+q}{2}$ and finds the solution. Therefore the number of steps in the algorithm is the distance between the initial value of $x = \left\lceil \sqrt{N} \right\rceil$ and the final value $x = \frac{p+q}{2}$. The following sequence describes computational complexity of the algorithm

$$\mathbb{C}(p,q) = \frac{p+q}{2} - \sqrt{N} = \frac{p+q-2\sqrt{pq}}{2} = \frac{(\sqrt{p}-\sqrt{q})^2}{2} = \frac{(p-\sqrt{N})^2}{2p}$$

Clearly, it depends on how far away the factors $p$ and $q$ are from $\sqrt{N}$. Let us investigate the case for which $\mathbb{C}(p,q) = 1$, i.e. the algorithm needs one step only or

$$(p - \sqrt{N})^2 = 2p \;\longrightarrow\; p - \sqrt{2}\sqrt{p} - \sqrt{N} = 0$$

The quadratic equation has two solutions

$$\sqrt{p} = \frac{\sqrt{2} \pm \sqrt{2 + 4\sqrt{N}}}{2} \quad \text{this implies} \quad p = 1 \pm \sqrt{1 + 2\sqrt{N}} + \sqrt{N}$$

It means that the difference

$$|p - \sqrt{N}| = |1 \pm \sqrt{1 + 2\sqrt{N}}| = O(N^{1/4})$$

is small enough the Fermat algorithm works instantanously. On the other hand, if the factors are far away from $\sqrt{N}$ or they have only trivial factors (the integer $N$ is prime), then $\mathbb{C}(p, q) = O(N)$.

## 3. QUADRATIC SIEVE

The idea of quadratic sieve (QS) can be traced back to Kraitchik [Pomerance 1996]. The starting point is the Fermat Algorithm. The following list describes modifications and improvements.

- Instead of considering the relation $N = x^2 - y^2 = (x - y)(x + y)$, we can use a congruence
$$x^2 - y^2 = 0 \mod N.$$
- To find the above relation, we use function $Q(x) = x^2 - N$, where $x \in X = \{\lceil\sqrt{N}\rceil, \lceil\sqrt{N}\rceil + 1, \ldots, \lceil\sqrt{N}\rceil + \ell\}$. Note that selection of $x$ that is closest to $\lceil\sqrt{N}\rceil$ guarantees that $x^2 - N$ grows slowly so it is much smaller than $N$. Now we are looking for a collection of $x \in C \subset X$ such that
$$\prod_{x \in C} x^2 = \prod_{x \in C} Q(x) = y^2 \pmod N.$$
- The trick is to find $\prod_{x \in C} Q(x)$ so it is equal to $y^2$. As the integers $Q(x)$ are relatively short, we can try to factorise them using a factor base of the smallest consecutive primes. Assume that our factor base is
$$FB = \{2, 3, 5, 7, \ldots \alpha\},$$
where $\alpha$ is the largest prime in *FB*. Now we use the primes from *FB* to factorise $Q(x)$; $x \in X$. Denote $X' \subset X$ such that for each $x \in X'$, $Q(x)$ is fully factorised (i.e. all their factors are in *FB*). Finally, we choose a subset $C \subset X'$ such that
$$\prod_{x \in C} Q(x) = p_{k_1}^{e_{k_1}} \cdots p_{k_m}^{e_{k_m}} \pmod N$$

where all primes $p_{k_i} \in FB$ and all exponents $e_{k_i}$ are even $(i = 1, 2, \ldots, m)$. Consequently, we obtain

$$u = \prod_{x \in C} x \mod N \text{ and } v = p_{k_1}^{e_{k_1}/2} \cdots p_{k_m}^{e_{k_m}/2} \pmod N.$$

This is to say that our target quadratic relation is $u^2 = v^2 \mod N$.
The steps listed above lead us to the following algorithm.

**Algorithm 3:** Quadratic Sieve

**Input** : An odd integer $N$
**Output**: Two factors $p$ and $q$

*Initialize the factor base* $FB = \{2, 3, \ldots, \alpha\}$;
*Initialize* $X' = \emptyset$;
**for** $x = \lceil \sqrt{N} \rceil$ **to** $\lceil \sqrt{N} \rceil + \ell$ **do**
    *compute Q(x) and factor it using primes from FB*;
    **if** all factors of $Q(x)$ in $FB$ **then**
        $X' := X' \cup \{x\}$;
    **end**
**end**
*From* $X'$ *find a subset* $C$ *of* $x$ *so their product has all even exponents in its factorisation*;
**if** $C$ found **then**
    *compute* $u$ *and* $v$;
    $p = gcd(N, u - v)$ *and* $q = gcd(N, u + v)$;
    *return* $p$ *and* $q$;
**else**
    *return FAILURE*
**end**

## 4. CONTINUED FRACTION AND FACTORIZATION

It is not too difficult to notice that integers $Q(x)$ grow while $x_i = \sqrt{N} + i$ is getting bigger. Consider

$$Q(x_i) = (\sqrt{N} + i)^2 - N = i(2\sqrt{N} + i)$$

Assuming that $i \ll \sqrt{N}$ and $i = 1, 2, \ldots$, then integers $Q(x_i)$ grow linearly with $\sqrt{N}$. This implies that factorization of $Q(x_i)$ using the factor base *FB* becomes more and more time consuming. Lehmer and Powers [1931] suggested to replace the

sequence of $Q(x)$ by a sequence generated by a continued fraction expansion of $\sqrt{N}$. Let as denote

$$\sqrt{N} = [a_0, a_1, a_2, a_3, \dots] = a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{a_{3+\dots}}}}$$

The idea is to approximate $\sqrt{N}$ by consecutive continued fraction convergents, i.e.

$$\frac{p_k}{q_k} = [a_0, a_1, a_2, \dots, a_k],$$

where $k$ = 1,2, ... . This means that $N$ can be approximated by $\left(\frac{p_k}{q_k}\right)^2$. In other words we choose

$$Q(k) = p_k^2 - q_k^2 N \Longrightarrow Q(k) = p_k^2 \pmod{N}$$

The advantage of generation of $Q(k)$ over $Q(x)$ is that $|Q(k)| < 2\sqrt{N}$ for all $k$. In other words, $Q(k)$ does not grow with $k$ and its factorization using the *FB* takes a constant workload.

## 5. QS EXAMPLE

Let us illustrate steps of the algorithm using a simple numerical example [Pieprzyk, Hardjono and Seberry 2003]. Assume that we wish to find factors of $N = 4841$. First we generate a sequence of quadratic residues $Q(x)$. To keep $Q(x)$ as small as possible, we find $m = \lfloor \sqrt{N} \rfloor = 69$ and compute

$$Q(x) = (m + x)^2 - N \tag{1}$$

for $x = -8 \dots, -1, 0, 1, \dots, 8$. The sequence of $Q$s is as follows:

$$\begin{aligned}
&= -8 \to Q(x) = -1120 = (-1) \cdot 2^5 \cdot 5 \cdot 7 \\
&= -7 \to Q(x) = -997 = (-1) \cdot 997 \\
&= -6 \to Q(x) = -872 = (-1) \cdot 2^3 \cdot 109 \\
&= -5 \to Q(x) = -745 = (-1) \cdot 5 \cdot 149 \\
&= -4 \to Q(x) = -616 = (-1) \cdot 2^3 \cdot 7 \cdot 11 \\
&= -3 \to Q(x) = -485 = (-1) \cdot 5 \cdot 97 \\
&= -2 \to Q(x) = -352 = (-1) \cdot 2^5 \cdot 11 \\
&= -1 \to Q(x) = -217 = (-1) \cdot 7 \cdot 31
\end{aligned}$$

$$
\begin{aligned}
x = 0 &\rightarrow Q(x) = -80 = 2^4 \cdot 5 \\
x = 1 &\rightarrow Q(x) = 59 = 59 \\
x = 2 &\rightarrow Q(x) = 200 = 2^3 \cdot 5^2 \\
x = 3 &\rightarrow Q(x) = 343 = 7^3 \\
x = 4 &\rightarrow Q(x) = 488 = 2^3 \cdot 61 \\
x = 5 &\rightarrow Q(x) = 635 = 5 \cdot 127 \\
x = 6 &\rightarrow Q(x) = 784 = 2^4 \cdot 7^2 \\
x = 7 &\rightarrow Q(x) = 935 = 5 \cdot 11 \cdot 17 \\
x = 8 &\rightarrow Q(x) = 1088 = 2^6 \cdot 17
\end{aligned}
$$

A factor base is a collection of the smallest consecutive primes so $FB = \{-1, 2, 3, 5, 7, 11\}$. Note that $Q(-8)$, $Q(-4)$, $Q(-2)$, $Q(0)$, $Q(2)$, $Q(3)$, and $Q(6)$ have all their factors in the set *FB*. These are the required full factorizations. There are eight fully factored *Q*s and the number of elements in the set *FB* is six so there is a good chance to find a quadratic con-gruence $u^2 \equiv v^2 \pmod{N}$. For a fully factored $Q(x)$, we create a binary vector $F(x)$ of the length $\ell|FB|$ whose coordinates indicate the presence or absence of an odd factor from *FB*. Thus, for $Q(-8)$, the vector $F(-8 = [1, 1, 0, 1, 1, 0]$ as its factorization contains –1 and primes 2, 5 and 7. The collection of all vectors $F$ for fully factored *Q*s, is:

$$
\begin{aligned}
& & [-1, 2, 3, 5, 7, 11] \\
Q(-8) \rightarrow F(-8) &= [\ 1,\ 1, 0, 1, 1,\ 0] \\
Q(-4) \rightarrow F(-4) &= [\ 1,\ 1, 0, 0, 1,\ 1] \\
Q(-2) \rightarrow F(-2) &= [\ 1,\ 1, 0, 0, 0,\ 1] \\
Q(0) \rightarrow F(0) &= [\ 0,\ 0, 0, 1, 0,\ 0] \\
Q(2) \rightarrow F(2) &= [\ 0,\ 1, 0, 0, 0,\ 0] \\
Q(3) \rightarrow F(3) &= [\ 0,\ 0, 0, 0, 1,\ 0]
\end{aligned}
$$

The vectors $F(x)$ form the rows of our matrix $F$:

$$
F = \begin{bmatrix} F(-8) \\ F(-4) \\ F(-2) \\ F(0) \\ F(2) \\ F(3) \end{bmatrix} = \begin{bmatrix} 1,1,0,1,1,0 \\ 1,1,0,0,1,1 \\ 1,1,0,0,0,1 \\ 0,1,0,1,0,0 \\ 0,1,0,1,0,0 \\ 0,0,0,0,1,0 \end{bmatrix}
$$

Now we look for a collection of rows such that

$$F(i_1) \oplus F(i_2) \oplus ... \oplus F(i_r) = 0,$$

where $\oplus$ stands for the bit-by-bit XOR operation. This step can be done using standard row-reducing techniques. Observe that $F(-4) \oplus \mathrm{F}(-2) \oplus \mathrm{F}(3) = 0$. Take the corresponding $Q(-4), Q(-2)$ and $Q(3)$ and write them as:

$$\begin{aligned} Q(-4) &= (69-4)^2 \mod 4841 \\ Q(-2) &= (69-2)^2 \mod 4841 \\ Q(3) &= (69+3)^2 \mod 4841 \end{aligned}$$

On the other hand, we can use their factorizations for a second set of relations:

$$\begin{aligned} Q(-4) &\equiv (-1) \cdot 2^3 \cdot 7 \cdot 11 \pmod{4841} \\ Q(-2) &\equiv (-1) \cdot 2^5 \cdot 11 \pmod{4841} \\ Q(3) &\equiv 7^3 \pmod{4841} \end{aligned}$$

The requested congruence $u^2 \equiv v^2 \pmod{N}$ can be constructed as follows:

$$Q(-4)Q(-2)Q(3) \equiv 2^8 \cdot 7^4 \cdot 11^2 \pmod{4841}$$

Note that the left hand side is $Q(-4)Q(-2)Q(3) = (69-4)^2(69-2)^2(69+3)^2$ and the right and side is $2^8 \cdot 7^4 \cdot 11^2$. Therefore, both sides are powers of two. The left integer

$$u = (69-4)(69-2)(69+3) = 3736 \pmod{4841}$$

and the right integer

$$v = \sqrt{(-1)2^3 \cdot 7 \cdot 11 \cdot (-1)2^5 \cdot 11 \cdot 7^3} = 2^4 \cdot 7^2 \cdot 11 \equiv 3783 \pmod{4841}.$$

As $u + v \neq i \cdot N$, we obtain the factors of $N$. Indeed, $\gcd(3736 - 3783, 4841) = 47$ and $\gcd(3736 + 3783, 4841) = 103$. So $N = 47 \cdot 103$.

## 6. FACTORIZATION OF RSA MODULI

Note that the security of RSA cryptosystem is based on the assumption that the factorization of the modulus $N$ is "difficult". Given an RSA encryption with the modulus $N$ and public key $e$. Assume that an adversary is able to factor $N$ and finds its factors $p$ and $q$ such that $N = pq$. The adversary is able to compute the secret key $d$ as

$$d = e^{\wedge}\{-1\} \bmod (p-1)(q-1)$$

and break the encryption.

In 1991 RSA Security (the company founded by the inventors of RSA) announced a list of challenge moduli of various lengths and prices for their factorization. The shortest modulus includes 100 decimal digits (or 330 bits) with the price tag of US$ 1,000. The longest modulus contains 617 decimal digits (or 2048 bits) with the award of US$ 200,000. Soon after the announcement, Manasse and Lenstra [1999] factored the first modulus

$$\begin{aligned}\text{RSA-100} = {} & 152260502792253336053561837813263742971806811496138068865790849458012296325895289765400035069200613937975227936943673922808872755445627854565536638199 \\ & \times 400946909509208810306837352927614683892148997240611522605027922533360535618378132637429718068114961380688657908494580122963258952897654000350692006139\end{aligned}$$

They used the muliple-polynomial quadratic sieve (MPQS) with the workload of 7 MIPS years (i.e. it takes 7 years by a CPU that performs $10^6$ instructions per second).

The current world record for the RSA challenge is factorization of RSA-768 (232 decimal digits) done in 2009 by an international team [Kleinjung et al. 2010]. The integer and its factors are:

$$\begin{aligned}\text{RSA-768} = {} & 123018668453011775513049495838496272077285356959533479219732245215172640050726365751874520219978646938995647494277406384592519255732630345373154826850791702612214291346167042921431160222124047927473779408066535141959745985690214341333478071698956898786044169848212690817704794983713768568912431388982883793878002287614711652531743087737814467999489 \\ & \times 367460436667995904282446337996279526322791581643430876426760322838157396665112792333734171433968102700927987363089 17\end{aligned}$$

The factorization was done using the number field sieve (NFS) and required around $3 \times 10^5$ MIPS years. The authors [Kleinjung et al. 2010] argue that RSA with 1024 bit moduli may be vulnerable to factorization within a decade by an academic effort. In 2007, RSA Security withdrew from the challenge. Consequently, the authors of factoring RSA-768 could not claim the US$ 50,000 award.

## 7. FACTORING WITH QUANTUM ALGORITHMS

A breakthrough is due to Shor [1997] who showed that integer factorization can be done on quantum computer in polynomial time. The idea is to use a periodic function

$$f(x) = a^x = a^{x+r} \mod N,$$

where $N$ is our integer that needs to be factored, a is $a$ random integer $a \in \mathbb{Z}_N$ that is co-prime to $N$ (or gcd $(a, N) = 1$) and $r$ is a period. The Shor quantum algorithm shows how to determine (with a non-negligible probability) the period $r$. Once $r$ is found and even, then we can determine factors as

$$a^r = 1 \mod N \Rightarrow \left(a^{\frac{r}{2}} - 1\right)\left(a^{\frac{r}{2}} + 1\right) = 0 \mod N$$

and gcd $\left(a^{\frac{r}{2}} - 1, N\right)$ is likely to produce a nontrivial factor of $N$.

We need to introduce basic notation that is necessary for our discussion. Given an orthonormal basis $\{|x_1\rangle, \ldots, |x_n\rangle\}$ of $n$-dimensional Hilbert space, where $|x_i\rangle$, are complex number for $i = 1,\ldots, n$. Note that the "ket" notation $|x\rangle$, has been introduced by Dirac. Behaviour of any quantum system is described by the following relation

$$\alpha_1 |x_1\rangle + \ldots + \alpha_n |x_n\rangle, \tag{2}$$

where $\alpha_i$, $i = 1, \ldots n$; are complex numbers such that $\sum_i^n \alpha_i^2 = 1$.

The system described by Eq (2) is probabilistic. When observed, however, it becomes deterministic and it can be in one of the $n$ states $x_i$ with probability $\alpha_i^2$. The reader who would like to get more detailed exposition of quantum computation is referred to any of many textbooks on quantum computing [Hirvensalo 2001]. Now we are ready to present the Shor quantum algorithm (see Algorithm 4).

**Algorithm 4:** Shor Quantum Algorithm for Finding Period of $f(x) = a^x \mod N$

**Input** : A periodic function $f(x) = a^x \mod N$

**Output**: An integer $r$ such that $a^r = 1 \mod N$

1 *Prepare an initial quantum state*

$$\frac{1}{\sqrt{m}} \sum_{x=0}^{m-1} |x\rangle\ |0\rangle,$$

*for $q$ qubits, where $m = 2^q$ ($m > N$);*

2 *Apply the function $f(x)$ on the initial state*

$$\frac{1}{\sqrt{m}} \sum_{x=0}^{m-1} |x\rangle\ |f(x)\rangle = \frac{1}{\sqrt{m}} \sum_{x=0}^{m-1} |x\rangle\ |a^x\rangle = \frac{1}{\sqrt{m}} \sum_{\beta=0}^{r-1} \sum_{\alpha=0}^{b} |\alpha r + \beta\rangle\ |a^\beta\rangle$$

*Note that as $f(x)$ is periodic so $x = \alpha r + \beta$, where $\beta < r$ and $b$ is an integer such that $m - br < r$;*

3 *Compute quantum Fourier transform (QFT) on the resulting state*

$$\frac{1}{\sqrt{m}} \sum_{\beta=0}^{r-1} \sum_{\alpha=0}^{b} \frac{1}{\sqrt{m}} \sum_{y=0}^{m-1} e^{\frac{2\pi i y(\alpha r+\beta)}{m}} |y\rangle\ |a^\beta\rangle = \frac{1}{m} \sum_{\beta=0}^{r-1} \sum_{y=0}^{m-1} e^{\frac{2\pi i y\beta}{m}} \sum_{\alpha=0}^{b} e^{\frac{2\pi i y\alpha r}{m}} |y\rangle\ |a^\beta\rangle$$

*and reorder the sum* ;

4 *Perform a measurement on the quantum state and get $y \in \mathbb{Z}_m$;*

5 *Calculate convergents $\frac{y_i}{z_i}$ of $\frac{y}{m}$ using the Euclid algorithm and return the smallest $z_i$ such that $a^{z_i} = 1 \mod N$;*

The following example is taken from [Hirvensalo 2001]. Let $N = 15$ and $a = 7$. We would like to find the period of $a$. Assume that $m = 16$ or we need a quantum state with 4 qubits. We follow the steps of the algorithm.

1. Prepare an initial state $\frac{1}{4}\sum_{x=0}^{15}|x\rangle\,|0\rangle$

2. Apply the function $f(x)7^x \mod 15$ on the state and get

$$\begin{aligned}
&\frac{1}{4}\big(|0\rangle|1\rangle + |1\rangle|7\rangle + |2\rangle|4\rangle + \ldots + |15\rangle|13\rangle\big) \\
&= \frac{1}{4}\Big(\big(|0\rangle + |4\rangle + |8\rangle + |12\rangle\big)|1\rangle \\
&\quad + \big(|1\rangle + |5\rangle + |9\rangle + |13\rangle\big)|7\rangle \\
&\quad + \big(|2\rangle + |6\rangle + |10\rangle + |14\rangle\big)|4\rangle \\
&\quad + \big(|3\rangle + |7\rangle + |11\rangle + |15\rangle\big)|13\rangle\Big)
\end{aligned}$$

3. After using QFT in $\mathbb{Z}_{16}$, we obtain

$$\frac{1}{4}\Big(\big(|0\rangle+|4\rangle+|8\rangle+|12\rangle\big)|1\rangle + \big(|0\rangle+i|4\rangle-|8\rangle-i|12\rangle\big)|7\rangle + \big(|0\rangle-|4\rangle+|8\rangle-|12\rangle\big)|4\rangle + \big(|0\rangle-i|4\rangle-|8\rangle+i|12\rangle\big)|13\rangle\Big)$$

4. Measurements give one of integers from the set $\{0, 4, 8, 12\}$ each occurs with probability $\frac{1}{4}$.
5. Observations 4 and 12 produce the correct period 4, while 0 and 8 fail.

As the reader may guess, the Shor algorithm can be used for factorization of integers. Details and an evaluation of the probability that the Shor factorization algorithm succeeds/fails can be found in [Hirvensalo 2001].

**Algorithm 5:** Shor Quantum Factorization Algorithm

**Input** : An odd integer $N$
**Output**: Nontrivial factors of $N$

*Choose at random* $a \xleftarrow{R} \mathbb{Z}_N$;
**if** gcd $(a, N) \neq 1$ **then**
  *Return the factor* $p = \gcd(a, N)$ *and exit*;
**end**
*Calculate the period* $r$ *of* $f(x) = a^x \mod N$ *using Algorithm 4 with* $m = 2^q$ *and* $N^2 \leq m < 2N^2$, *where* $q$ *is the number of qubits* ;
**if** $a^r \neq 1 \mod N$ *or* $r$ *is odd* **then**
  *Return "FAIL" and exit*;
**end**
*Compute factors* $p_{1,2} = \gcd(N, a^{\frac{r}{2}} \pm 1)$;

The computational complexity of Algorithm 5 is $O((\log N)^3)$ and the probability of success is at least $\Omega\left(\frac{1}{\log\log N}\right)$. The overall workload needed to factor $N$ is $O((\log N)^3 \log\log N)$. This means that the factorization runs in polynomial time and is "easy". As the result, the RSA cryptosystem is easy to break on quantum computer (or alternatively on a quantum factoring device).

Indeed, in 2001 a group from IBM factored 15 ($3 \times 5$) using a quantum computer with 7 qubits [Vandersypen et al. 2001]. In 2012, the integer 21 ($3 \times 7$) was factored [Martin-López et al. 2012] – this is so far the best result achieved using the Shor algorithm. It turns out that adiabatic quantum computation can simulate

the Shor algorithm and solves factorization in polynomial time. The best result achieved so far using the technique is factorization of 56153 (which is equal to $(233 \times 241)$ [Dattani and Bryans 2014].

## 8. CONCLUSIONS

Public-key cryptography is based on the assumption that there are so called "one-way functions" or in other way functions that are "easy" to compute but difficult to reverse. In this work, we took a look at one of such function. We know that having two integers we can easily find their product. The textbook multiplication of two integers each $n$-bit long takes $O(n^2)$ steps. However, knowing an integer $N$, it is "hard" to find its factors on classical computers. Security of the RSA encryption is directly related to difficulty of factorization. A concerted effort of mathematicians and cryptographers led to a significant improvements of conventional factorization algorithms. However, none of them could break the sub-exponential complexity barrier. Shor demonstrated that factorization is easy on quantum computer. Needless to say that this breaks the RSA cryptosystem at least theoretically. Interestingly, the attention has been switched from development of new more efficient algorithms to implementation of quantum computing. The jury is still out and we do not know if a full scale quantum computer can be constructed. Even if this may not be possible, it is likely that specialised quantum factoring devices can be implemented. The revolution in theory and practice of quantum factorization spells the end of RSA cryptosystems (together with other cryptosystems based on Discrete Logarithm). It is no surprise that in 2017 NIST announced a competition for quantum-resistant public-key cryptographic algorithms [NIST 2018].

**Table 1.** The comparison of the factoring algorithms

| Algorithm | Complexity | Comments |
|---|---|---|
| Sieve of Eratosthenes | $O(2_2^n)$ | exponential |
| Quadratic Sieve | $O\left(e^{(1+o(1))(\log n \log\log n)^{\frac{1}{2}}}\right)$ | sub-exponential |
| Number Field Sieve | $O\left(e^{(1.92+o(1))(\log n)^{\frac{1}{3}}(\log n \log\log n)^{\frac{2}{3}}}\right)$ | sub-exponential |
| Shor's Quantum Algorithm | $O(n^3 \log n)$ | polynomial |

The comparison of the factoring algorithms are given in Table 1 above. Note that $n = \log N$ is the length of integer $N$. The reader who would like to explore other classical factorization algorithms (not covered in this work) is referred to [Knuth 1997; Crandall and Pomerance 2001; Wagstaff Jr. 2013].

## ACKNOWLEDGEMENTS

*The paper was written while the author was visiting Gdynia Maritime University in June 2018. The author wishes to thank Professor Andrzej Borys for his support during his stay at the University. Special thanks go to anonymous referees for their critical comments that have improved the presentation and the paper readability. The author has been supported by the Australian Research Council grant DP180102199.*

## REFERENCES

Crandall, R., Pomerance, C., 2001, *Prime Numbers: A Computational Perspective,* Springer.

Dattani, N.S., Bryans, N., 2014, *Quantum Factorization of 56153 with only 4 Qubits,* Quantum Physics, arXiv:1411.6758,

Hirvensalo, M., 2001, *Quantum Computing,* Natural Computing Series, Springer.

Kleinjung, T., Aoki, K., Franke, J., Lenstra, A.K., Thomé, E., Bos, J.W., Gaudry, P., Kruppa, A., Montgomery, P.L., Osvik, D.A., te Riele, H., Timofeev, A., Zimmermann, P., 2010, *Factorization of a 768-bit RSA Modulus*, CRYPTO'10 Proceedings of the 30th Annual Conference on Advances in Cryptology, August 15–19, Santa Barbara, CA, USA, pp. 333–350.

Knuth, D., 1997, *The Art of Computer Programming*, vol. 2, Seminumerical Algorithms, 3rd ed., Addison-Wesley, Boston, MA, USA.

Lehmer, D.H., Powers, R.E., 1931, *On Factoring Large Numbers*, Bull. Amer. Math. Soc., vol. 37, no. 10, pp. 770–776.

Manasse, M., Lenstra, A.K., 1999, *RSA Honor Roll,* http://www.ontko.com/pub/rayo/primes/hr_ rsa.txt (20.08.2018).

Martin-López, E., Laing, A., Lawson, T., Alvarez, R., Zhou, Xiao-Qi, O'Brien, J.L., 2012, *Experimental Realization of Shor's Quantum Factoring Algorithm using Qubit Recycling,* Nature Photonics, vol. 6, no. 11.

NIST, 2018, *Post-Quantum Cryptography,* https://csrc.nist.gov/Projects/Post-Quantum-Cryptography, (20.08.2018).

Pieprzyk, J., Hardjono, T., Seberry, J., 2003, *Fundamentals of Computer Security*, Springer.

Pomerance, C., 1996, *A Tale of Two Sieves,* Notices Amer. Math. Soc, vol. 43, pp.1473–1485.

Rivest, R., Shamir, A., Adleman, L., 1978, A *Method for Obtaining Digital Signatures and Public Key Cryptosystems*, Communications of the ACM, vol. 21, no. 2, pp. 120–126.

Shor, P.W., 1997, *Polynomial-time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*, SIAM Journal on Computing 26.5, pp. 1484–1509.

Vandersypen, L.M.K., Steffen, M., Breyta, G., Yannoni, C.S., Sherwood, M.H., Chuang, I.L., *Experimental Realization of Shor's Quantum Factoring Algorithm using Nuclear Magnetic Resonance,* Nature, vol. 414 no. 6866, pp.883–887.

Wagstaff, S.S. Jr., 2013, *The Joy of Factoring*, American Mathematical Society, Providence, RI, USA.