# Linnæus University 

Sweden

## Degree project

## Rabin's Cryptosystem



Author: Evan Huynh
Supervisor: Per Anders Svensson
Examiner: Marcus Nilsson
Date: 2021-05-28
Course Code: 2MA41E, 15 credits
Subject: Mathematics
Level: Bachelor

# Rabin's Cryptosystem <br> Evan Huynh 

May 2021


#### Abstract

In this paper we will explore Rabin's cryptosystem, one of the cryptographic algorithm that is similar to RSA developed by Michael O. Rabin based on the quadratic residue problem. We will introduce the background theory, the scheme and the security of Rabin and a basic padding scheme to use for Rabin's system. Also, there is another exploration of picking different type of primes and an algorithm to solve the quadratic residue problem when the prime $p \not \equiv 3(\bmod 4)$ and the experiment to measure the performance of that algorithm.

\section*{Acknowledgements}

I would like to express our gratitude to my supervisor Dr. Per Anders Svensson for his guidance throughout the making of this thesis. Thanks to his extraordinary knowledge passed to me along the process, I have learned a lot more interesting topics in the field of Cryptography. It is simply not possible for me without his advice. I also want to thank my mother, sister and high school teacher for providing me with emotional and psychological support. It has been a very long and stressful process for me and thank you everyone for supporting me along the way.


## Contents

1 Introduction ..... 4
2 History ..... 4
3 Basic Number Theory ..... 5
4 Rabin's Scheme ..... 8
4.1 Key Generation ..... 8
4.2 Encryption ..... 8
4.2.1 Background Theory ..... 8
4.2.2 Algorithm ..... 9
4.3 Decryption ..... 9
4.3.1 Background Theory ..... 9
4.3.2 Algorithm ..... 12
4.4 Security ..... 14
5 Problems ..... 15
5.1 Uniqueness of Ciphertext ..... 15
5.1.1 Message and Key size ..... 15
5.2 Four Different Outputs ..... 16
5.2.1 Problem ..... 16
5.2.2 Padding ..... 17
5.2.3 The Experiment ..... 18
5.2.4 Result ..... 18
5.3 Malleable attack ..... 20
6 Other choice of primes ..... 21
6.1 Tonelli-Shanks Algorithm ..... 22
6.2 Proof of Tonelli-Shanks algorithm ..... 24
6.3 The Experiment ..... 25
6.4 Result ..... 28
7 Discussion ..... 28
8 Appendix ..... 30
8.1 Mathematica code ..... 30
8.1.1 Preamble ..... 30
8.1.2 Textbook Simulation ..... 30
8.1.3 Padding test ..... 30
8.1.4 Tonelli-Shanks algorithm ..... 32
8.1.5 Tonelli-Shanks Theoretical Test ..... 34
8.1.6 Tonelli-Shanks Practical Test ..... 34

## 1 Introduction

Rabin's cryptosystem is an asymmetric cryptographic algorithm which, similar to RSA, relies on the difficulty of integer factorization for its security. There is currently belief that integer factorization is a hard problem since it is unsolved in the domain of computer science. Unlike RSA, the equivalence of breaking Rabin scheme and integer factorization has been proven, while there is no such proof for the other system [11, p. 292]. As the dominance of RSA grows, one wonders why a mathematically provably secure algorithm is not always the market choice, whereas the "good enough" solution can have massive influence over the modern society, not to mention that it is more of a belief rather than solid proof. With that being said, in this paper, we shall explore the history, theory, and description of the Rabin's cryptosystem.

## 2 History

Throughout the history of hiding the message, most of the cryptosystems before 1970 are based on shared secrets, that is the sender and receiver need to establish identical keys. If one of the party somehow lost the key and someone found it, their communication is no longer private, i.e. the adversary can easily eavesdropping to their conversation without the hardship of decoding the message. However, the art of keeping the secrets did not stop just by sending nonsensical numbers and letters. People also trying to hide it by talking in a different languages, using code words which are shared between small groups and even steganography, which is concealing the messages via special way of writing.

Up until 1970, most cryptosystems are based on symmetric algorithm. That is, in order to decrypt the ciphertext, both sender and receiver have to share the identical key, which is used for both encryption and decryption, hence the name. However, the research for a modern way of encryption is really needed. In 1976, Whitfield Diffie and Martin Hellman introduced the concept of asymmetric key cryptosystem. This later became known as Diffie-Hellman key exchange, although it is not exactly an encryption scheme. In 1977, just one year after, three computer scientists at MIT named Ron Rivest, Adi Shamir and Leonard Adleman invented a scheme and later published their work in 1978. Their algorithm is later known as RSA from their initials. This scheme uses the product of the modulo exponentiation of two large primes to encrypt and decrypt. The encryption and decryption processes use different keys and its difficulty is at best as hard as factoring large integers.

Rabin's cryptosystem was first introduced in 1979 by Michael O. Rabin, just one year after the publication of RSA. This paper was published in the MIT Laboratory for Computer Science. Interestingly enough, the publication of RSA also happened at MIT. It is no wonder why there are so much similarity between these two cryptosystems in term of inner working, description and sometimes the weaknesses as they might share. However, the popularity of one does not imply of the other. Nowadays there are more cryptographic schemes which are different class than both RSA and Rabin. People nowadays consider newer schemes based on elliptic curves and some other which does
not relies on the integer factorization, which is the basis for both RSA and Rabin.

## 3 Basic Number Theory

Before the introduction of Rabin's cryptosystem, one needs to grasp some concepts of number theory. This is the study of numbers, specifically, the study of integers, which based itself on the set $\mathbb{Z}$.

At first glance it is tempting to think that there are only four basic operations that one learned from elementary school which are addition, subtraction, multiplication and division. However, division is a dangerous operation since not all result from the divisions, even between two integers, are integer. For example, $\frac{1}{2}$ is not an integer. So is there a way to perform division on the set of integers? The answer comes from the Ancient Greek, from Euclid's time. It is called the division algorithm (or Euclidean division).

Theorem 1 (Division Algorithm (Euclidean Division) [18, p. 37]). Given that $a, b \in$ $\mathbb{Z}, b>0$, the division algorithm states that there are unique $q, r \in \mathbb{Z}$ such that $a=b q+r$, where $0 \leq r<b$.

Surprisingly, division algorithm is a theorem rather than just a definition, meaning there needs to be a proof before one can use it. Prior to proving this theorem, there is one important axiom that one needs to know.

The Well-ordering Principle [18, p. 6] Every non-empty set $S$ of the positive integers has the least element.

According to Kenneth H. Rosen written in his book Elementary Number Theory, this well-ordering principle is obvious yet important [18, p. 6]. It is easy enough to see this principle in practice by taking a finite subset from $\mathbb{Z}$ and sort its elements in order. Consider $A=\{6,8,4,2,0\}$, we can sort the set elements as $A=\{0,2,4,6,8\}$ and the least element of this set is 0 . This well-ordering property of the non-empty positive integer set allows us to prove the division algorithm.

Proof. This proof is inspired by the book [18, p. 38].
Part 1 (The existence of $q$ and $r$ ). Consider the set $S$ of all integers in the form of $a-b k$ where $k$ is an integer. We have

$$
S=\{a-b k \mid k \in \mathbb{Z}\} .
$$

Let $T$ be the set of all non-negative integers in $S$. We know that $T \neq \emptyset$ since $a-b k>0$ when $k<a / b$.

By the well-ordering property, $T$ has the least non-negative element, let it be $r=a-b q$.
Part $2(0 \leq r<b)$. If $r \geq b$ then

$$
r>r-b=a-b q-b=a-b(q+1) \geq 0
$$

which contradicts our choice of $r=a-b q$ be the least non-negative integer in the form of $a-b k$.

Part 3 ( $q$ and $r$ are unique). Assume that we have two equations $a=b q_{1}+r_{1}$ and $a=b q_{2}+r_{2}$. That means $b q_{1}+r_{1}=b q_{2}+r_{2}$. By some algebraic techniques, we will have $b\left(q_{1}-q_{2}\right)=r_{2}-r_{1}$. That means $b$ divides $r_{2}-r_{1}$.

Since $0 \leq r_{1}<b$ and $0 \leq r_{2}<b$, that gives $-b<r_{2}-r_{1}<b$. Hence, $b$ divides $r_{2}-r_{1}$ if and only if $r_{2}-r_{1}=0$, or $r_{2}=r_{1}$.

Since $b \neq 0$, that gives the equation $b\left(q_{1}-q_{2}\right)=0$ having the solution if and only if $q_{1}-q_{2}=0$, or $q_{1}=q_{2}$.

Definition 1 (Divisibility). Let $a$ and $b$ be integers, $b \neq 0$. We say that $a$ is divisible by $b$, or $b$ divides $a$, denote $b \mid a$ if there exists $q \in \mathbb{Z}$ such that $a=b q$.

Example 1. We have $2 \mid 4$ since $4=2 \cdot 2+0$.
Definition 2 (Modular arithmetic [18, p. 145]). Let $b$ be a positive integer. If $a$ and $r$ are integers, we say that $a$ is congruent to $r$ modulo $b$ if $b \mid(a-r)$.

Modular arithmetic is sometimes called clock arithmetic due to their similarity between this operation and counting hour since the number wrap around a circle, in this context, the clock.

Theorem 2 (Properties of modular arithmetic [18, p. 147]). For all $a, b, c, k, n \in \mathbb{Z}$, we have the following:

1. Reflexivity: $a \equiv a(\bmod n)$.
2. Symmetry: $a \equiv b(\bmod n) \Longleftrightarrow b \equiv a(\bmod n)$.
3. Transitivity: If $a \equiv c(\bmod n)$ and $c \equiv b(\bmod n)$ then $a \equiv b(\bmod n)$.

From theorem 2, we see that both $a$ and $r$ are equivalent (the same element) of the set congruence class modulo $b$, usually denoted $\mathbb{Z}_{b}$ from the algebraic context.

The main idea of proving such is to revert this equivalent relation, i.e. if $a \equiv b$ $(\bmod n)$ then there exists $q \in \mathbb{Z}$ such that $a=n q+b$ or $n \mid a-b$. This idea carries throughout the study of number theory and is also used to prove many theorems later in this thesis.

Aside from these basic properties of an equivalent relation, modular arithmetic also has these properties:

1. Addition and Subtraction: $a \equiv b(\bmod n)$ and $c \equiv d(\bmod n)$, then $a+c \equiv b+d$ $(\bmod n)$ and $a-c \equiv b-d(\bmod n)$.
2. With constant: If $k \in \mathbb{Z}$ and $a \equiv b(\bmod n)$, then $k+a \equiv k+b(\bmod n)$ and $k a \equiv k b(\bmod n)$.
3. Multiplication: If $a \equiv b(\bmod n)$ and $c \equiv d(\bmod n)$ then $a c \equiv b d(\bmod n)$.

Example 2. We have that $4=2 \cdot 2+0$ therefore $4 \equiv 0(\bmod 2)$.

4 and 0 are so-called congruent to each other since they belong to the same equivalence class in $\mathbb{Z}$, namely $\overline{0}$, or all integers that gives remainder 0 when divided by 2 .

Definition 3 (Coprime integers). Suppose two integers $a$ and $b$ are coprime (relatively prime $)$ then $\operatorname{gcd}(a, b)=1$.

The concept of coprime integers stems from the idea of prime numbers. They are the building blocks of integers, only having two factors, 1 and itself. Coprime is an extension of that concept, if two numbers have no common factors other than 1 , then from the perspective of integers, they are relatively prime to each other.

With that being said, we can use this view to define the modular inverses.
Definition 4 (Modular multiplicative inverse). Let $a$ and $n$ be integers where $n \neq 0$. We say that $b$ is a multiplcative inverse of $a \operatorname{modulo} n$ if $a b \equiv 1(\bmod n)$

If a modular multiplicative inverse of an integer exists then it will be unique, see Theorem 4. This applies to all integers. Finding such inverse can be done using extended Euclidean algorithm. ${ }^{1}$

Theorem 3 (Existence of modular multiplicative inverse). An integer $a$ has a multiplicative inverse modulo $n$ if and only if $\operatorname{gcd}(a, n)=1$.

Proof.
Part 1. Assume that $a$ has a multiplicative inverse. Then there exists $b$ such that $a b \equiv 1$ $(\bmod n)$. That means $a b-1=n k$ for some $k \in \mathbb{Z}$. Rearrange the equation, we have $a b-n k=1$. This is the linear Diophantine equation and only has the integer solutions if and only if there exists $c \in \mathbb{Z}$ such that $c \mid \operatorname{gcd}(a, n)$. In other words, $\operatorname{gcd}(a, n)=1$.
Part 2. Assume that $\operatorname{gcd}(a, n)=1$. Then the equation $a b-k n=1$ will have integer solutions for some $b, k \in \mathbb{Z}$. That means $a b-1=k n$. By the property of modular arithmetic, $a b \equiv 1(\bmod n)$ which implies that $b$ is the multiplicative inverse of $a$ with respect to modulus $n$.

Theorem 4 (Uniqueness of multiplicative modular inverse). Let $b$ be multiplicative inverse modulo $n$ of $a$. Then $b$ is unique.

Proof. From the previous theorem, we know that $\operatorname{gcd}(a, n)=1$. Let $b$ and $c$ be the inverses of $a$. We have $a b \equiv a c \equiv 1(\bmod n)$. By the property of modular arithmetic, $n \mid a(b-c)$, which means $n \mid(b-c)$ if and only if $b \equiv c(\bmod n)$.

Theorem 5 (Fermat's Little Theorem). Let $p$ be a prime and $a$ be an integer. Then

$$
a^{p} \equiv a \quad(\bmod p)
$$

There are many proofs of Fermat's little theorem, including one using Wilson's theorem [18, p. 216], which will be discussed later. However, this proof will use binomial theorem instead.

[^0]Proof. This proof is inspired by the wiki [4].
We will use the technique of induction.
Base case: $1^{2} \equiv 1(\bmod 2)$.
Assume that $a^{p} \equiv a(\bmod p)$ for $a=n$. Then by the Binomial Theorem, we have

$$
(n+1)^{p}=n^{p}+\binom{p}{1} n^{p-1}+\binom{p}{2} n^{p-2}+\cdots+\binom{p}{p-1} n+1
$$

We know that $\binom{p}{k}=\frac{p!}{k!(p-k)!}$ for $1 \leq k \leq p-1$. Furthermore, since $p$ is a prime, it is obvious that $p \nmid k!$ and $p \nmid(p-k)$ !, and $\binom{p}{k}$ is always positive integers in our case. That means $\binom{p}{k}$ contains the factor of $p$. In other words $p \left\lvert\,\binom{ p}{k}\right.$. Taking the modulo for $(n+1)^{p}$, we see that most of the terms disappear since $\binom{p}{k} n^{p-k} \equiv 0(\bmod p)$. That leaves us

$$
(n+1)^{p} \equiv n^{p}+1 \quad(\bmod p)
$$

The induction hypothesis says that $n^{p} \equiv n(\bmod p)$. Substituting into the equation, we get $(n+1)^{p} \equiv n+1(\bmod p)$.

## 4 Rabin's Scheme

With those background theory, we begin with the introduction to Rabin's cryptosystem. First, let us start with the key generation, since every encryption scheme, modern or classic, starts with a secret password.

### 4.1 Key Generation

For simplicity we choose a pair of primes $p$ and $q$ such that $p, q \equiv 3(\bmod 4)$. In Rabin's original paper, this way of choosing primes makes the decryption process more straightforward [17, p. 7].

Step 1: Pick a pair of prime $p$ and $q$ such that $p, q \equiv 3(\bmod 4)$.
Step 2: Let $n=p q$, that shall be our public key.
In order for the public key to be secure enough, we need to pick two prime $p$ and $q$ such that they are large and chosen in such a way that it prevents the use of some factorization methods (such as Fermat's factorization method [18, p. 130-134] or Pollard $p-1$ method [19, p. 317], as these methods were first introduced for students in cryptography and number theory) and far enough from each other, otherwise the attacker would be able to obtain the private key by factorizing it, thwarting our secure communication scheme.

### 4.2 Encryption

### 4.2.1 Background Theory

Definition 5 (Quadratic residue). Let $a$ and $n$ be an integer. If there exists $x$ such that

$$
x^{2} \equiv a \quad(\bmod n)
$$

then $a$ is called the quadratic residue modulo $n$. Otherwise $a$ is quadratic nonresidue modulo $n$.

### 4.2.2 Algorithm

Suppose we would like to encrypt $m$ to generate the cipher text $c$. Pick $0 \leq m<n$ and compute

$$
m^{2} \equiv c \quad(\bmod n) .
$$

Example 3. Alice wants to send Bob her favorite number. She knows that if she sends it in plaintext, the attacker would easily obtain it. Let us assume that her favorite number is 69 which she will send to Bob.

Bob picks two random primes, let us say 59 and 79 which he multiplies and obtains 4661 and sends it to Alice as his public key. Alice then takes her favorite number, squares it and takes the modulo with regard to Bob's public key. We have $69^{2}=4761 \equiv 100$ $(\bmod 4661)$.

She then sends 100 as the ciphertext to Bob. Of course, we assume that Eve also obtains 100 and 4661 as those two exchange the conversation over a public channel and Eve cannot factorize 4661, which makes this conversation "private."

### 4.3 Decryption

In order to decrypt Rabin's cryptosystem, we need to compute the square root residue of $c$, namely $x^{2}=c(\bmod n)$. Given the choice of $n$ is large enough, it is extremely hard to solve such equation if one does not know the factorization of $n$.

### 4.3.1 Background Theory

As always, there are some background theories that are attached to every part before the main description of Rabin's scheme.

Theorem 6 (Chinese Remainder Theorem (CRT) [18, p. 162]). Let $n_{1}, n_{2}, \ldots, n_{k}$ be positive pairwise relative primes. Let $a_{1}, a_{2}, \ldots, a_{k}$ be arbitrary integers. Then the system

$$
\begin{cases}x \equiv a_{1} & \left(\bmod n_{1}\right) \\ x \equiv a_{2} & \left(\bmod n_{2}\right) \\ \vdots & \\ x \equiv a_{k} & \left(\bmod n_{k}\right)\end{cases}
$$

has a unique solution modulo $N=n_{1} n_{2} \cdots n_{k}$.
Proof. This proof is inspired by the book [18, p. 162-163] and the video [12].
Part 1 (Existence). Let $N=n_{1} n_{2} \ldots n_{k}$ and $N_{i}=N / n_{i}$ for each $i=1,2, \ldots, k$. Basically, $N$ be the product of all modulo and $N_{i}$ be the product of all modulo except $n_{i}$ for $1 \leq i \leq k$.

First, we show that $\operatorname{gcd}\left(n_{i}, N_{i}\right)=1$. Suppose there exists $d \in \mathbb{Z}$ such that $d \mid n_{i}$ and $d \mid N_{i}$. Now, let $1 \leq j \leq k$ but $j \neq i$. Since all of the modulo are relatively prime, that
means $d\left|n_{j} \Rightarrow d\right| \operatorname{gcd}\left(n_{i}, n_{j}\right)$. But since $n_{i}$ and $n_{j}$ are relatively prime as well, that means $\operatorname{gcd}\left(n_{i}, n_{j}\right)=1$.

Next we will establish the solution. Since $\operatorname{gcd}\left(n_{i}, N_{i}\right)=1$, there exists a modular multiplicative inverse of $N_{i}$ modulo $n_{i}$. Let $x_{i}$ be an integer such that $N_{i} x_{i} \equiv 1\left(\bmod n_{i}\right)$.

Also we have $N_{i} x_{i} \equiv 0\left(\bmod n_{j}\right)$ for $j \neq i$. Again, this is possible since $N_{i}$ consists all the factors of $N$ but $n_{i}$ so $n_{j}$ is one of the factors that $N_{i}$ contains.

We claim that

$$
\begin{equation*}
x=x_{1} N_{1} a_{1}+x_{2} N_{2} a_{2}+\cdots+x_{k} N_{k} a_{k} \tag{1}
\end{equation*}
$$

is our solution. Applying the modulo $n_{i}$ for $1 \leq i \leq k$ to (1), we see that all other terms except $x_{i} N_{i} a_{i}$ are congruent to 0 .

Therefore $x \equiv 0+0+\cdots+x_{i} N_{i} a_{i}+\cdots+0\left(\bmod n_{i}\right)$. Since $x_{i} N_{i} \equiv 1\left(\bmod n_{i}\right)$, that leaves us $x \equiv a_{i}\left(\bmod n_{i}\right)$ for $1 \leq i \leq k$. Without loss of generality, this extends to all equations in our system.

Now we have found the solution, we have to prove its uniqueness. The strategy is based on the property of modular arithmetic. If $x \equiv y(\bmod N)$ and $x, y$ are the solutions, then they belong to the same equivalent class, or in our context, the same solution.

Part 2 (Uniqueness). Let $x, y$ be solutions. That means $x \equiv a_{i}\left(\bmod n_{i}\right)$ and $y \equiv a_{i}$ $\left(\bmod n_{i}\right)$. By the modular arithmetic property $x-y \equiv a_{i}-a_{i} \equiv 0(\bmod n)$. By definition, that means $n_{i} \mid x-y$. Since $N=n_{1} n_{2} \ldots n_{i} \ldots n_{k}$, which means that $N$ is a composite number consists of all factors, which also includes $n_{i}$, then $N \mid x-y$.

In conclusion $x \equiv y(\bmod N)$.
The theorem is thereby proven.
Lemma 1 (Quadratic residue equation [18, p. 416]). Let $p$ be an odd prime and $a$ an integer that does not divide $p$. Then the equation

$$
x^{2} \equiv a \quad(\bmod p)
$$

has either two incongruent solutions or no solutions modulo $p$.
Proof. This proof is inspired by the book [18, p. 416].
Assume that $x^{2} \equiv a(\bmod p)$ has a solution $x=x_{0} . \quad$ Then $x_{0}^{2} \equiv a(\bmod p)$ and $\left(-x_{0}\right)^{2}=x_{0}^{2} \equiv a(\bmod p)$, which is two of our solutions. Note that $x_{0} \not \equiv-x_{0}(\bmod p)$ since if $x_{0} \equiv-x_{0}(\bmod n)$ then by the property of modular arithmetic, we have $x_{0}-$ $\left(-x_{0}\right)=2 x_{0}$ will divide $p$. As $p$ is an odd prime, this is simply impossible. Note that $p \nmid x_{0}$ as well.

Let another solution be $x=x_{1}$ such that $x_{1} \not \equiv \pm x_{0}(\bmod p)$, that is $x_{1}$ is another solution to the equation than $x_{0}$ and $-x_{0}$. Then we have $x_{0}^{2} \equiv a(\bmod n)$ and $x_{1}^{2} \equiv a$ $(\bmod n)$. By the property of modular arithmetic, $x_{0}^{2}-x_{1}^{2}=\left(x_{0}-x_{1}\right)\left(x_{0}+x_{1}\right) \equiv a-a=0$ $(\bmod p)$. This means $p \mid x_{0}-x_{1}$ or $p \mid x_{0}+x_{1}$, which translates to $x_{0} \equiv x_{1}(\bmod p)$ or $x_{0} \equiv-x_{1}(\bmod n)$.

In conclusion, there are exactly two incongruent solutions, or no solution at all.

Lemma $2\left(\left[21\right.\right.$, p. 91]). Let $p$ be a prime and $a^{2} \equiv 1(\bmod p)$. Then $a \equiv \pm 1(\bmod p)$.
Proof. This is trivial since $p \mid a^{2}-1=(a-1)(a+1)$, either $p \mid a-1$ which means $a \equiv 1$ $(\bmod p)$, or $p \mid a+1$ which means $a \equiv-1(\bmod p)$.

Theorem 7 (Wilson's Theorem [18, p. 217]). If $p$ is prime then $(p-1)!\equiv-1(\bmod p)$.
Proof. This proof is inspired by the book [18, p. 218].
Consider $(p-1)!=1 \cdot(2 \cdot 3 \cdots \cdots \cdot(p-2)) \cdot(p-1)$. Since $p$ is prime, by Theorem 3 and 4, there exists a unique inverse $a^{-1}$ of $a$ for each integer $a$ with $1 \leq a \leq p-1$. By Lemma 2 , we have $1^{-1} \equiv 1(\bmod p)$ and $(p-1)^{-1} \equiv p-1(\bmod p)$ and these are the only elements the equal their own inverses. That means we can group the integers from 2 to $p-2$ into $(p-3) / 2$ pairs such that the product of the element in each pair is congruent to $1(\bmod p)$, or in another way

$$
2 \cdot 3 \cdots \cdots(p-3) \cdot(p-2) \equiv\left(2 \cdot 2^{-1}\right) \cdot\left(3 \cdot 3^{-1}\right) \cdots \equiv 1 \cdot 1 \cdots \cdots 1 \cdot 1 \quad(\bmod p)
$$

Then $(p-1)!\equiv 1(p-1) \equiv p-1 \equiv-1(\bmod p)$.
Definition 6 (Legendre Symbol [18, p. 417]). Let $p$ be an odd prime and $a$ be an integer that is not divisible by $p$. The Legendre symbol is defined as

$$
\left(\frac{a}{p}\right)= \begin{cases}1 & \text { if } a \text { is quadratic residue modulo } p \\ -1 & \text { if } a \text { is quadratic non-residue modulo } p\end{cases}
$$

Theorem 8 (Euler's criterion [18, p. 418]). Let $p$ be an odd prime and $a$ be an integer that is not divisible by $p$. Then

$$
\left(\frac{a}{p}\right) \equiv a^{(p-1) / 2} \quad(\bmod p)
$$

Proof. This proof is inspired by the book [18, p. 428], the wiki [3] and the video [13].
Consider that $a^{p-1}(\bmod p)$, by Fermat's Little Theorem, we have

$$
\begin{align*}
a^{p} & \equiv a \quad(\bmod p) \\
& \Longleftrightarrow a^{p-1} \equiv 1 \quad(\bmod p) \\
& \Longleftrightarrow a^{p-1}-1 \equiv 0 \quad(\bmod p) \\
& \Longleftrightarrow\left(a^{(p-1) / 2}-1\right)\left(a^{(p-1) / 2}+1\right) \equiv 0 \quad(\bmod p) \\
& \Longleftrightarrow a^{(p-1) / 2} \equiv 1 \quad(\bmod p) \text { or } a^{(p-1) / 2} \equiv-1 \quad(\bmod p) \tag{2}
\end{align*}
$$

If $a$ is quadratic residue modulo $p$ then there exists $b$ such that $a \equiv b^{2}(\bmod p)$.
Substitute $b$ into $a$ we have

$$
\begin{array}{rlrr}
a^{(p-1) / 2} & \equiv\left(b^{2}\right)^{(p-1) / 2} & & (\bmod p) \\
& \equiv b^{p-1} & (\bmod p) & \quad(\text { simplify the exponent }) \\
& \equiv 1 & (\bmod p) & (\text { Fermat's Little Theorem). }
\end{array}
$$

If $a$ is quadratic non-residue modulo $p$, that means $a \not \equiv b^{2}(\bmod p)$. Since $p$ is prime, that means the congruence $b x \equiv a(\bmod p)$ has a unique solution, namely $x \equiv b^{-1} a$ $(\bmod p)$. We also know that $x \not \equiv b(\bmod p)$ because that would give us $b^{2} \equiv a(\bmod p)$, which is a contradiction.

For each $1 \leq i \leq p-1$, we can group into $(p-1) / 2$ pairs such that each pair gives $i j \equiv a(\bmod p)$ where $i \neq j$. Similar to the proof of Wilson's Theorem, we have

$$
a \cdot a \cdots \cdots a \cdot a \equiv a^{(p-1) / 2} \equiv 1 \cdot 2 \cdots(p-1) \equiv(p-1)!\quad(\bmod p)
$$

and by Wilson's Theorem, $(p-1)!\equiv-1(\bmod p)$ so $a^{(p-1) / 2} \equiv-1(\bmod p)$ in this case.
Rewrite the proof in term of Legendre symbol, we conclude that

$$
a^{(p-1) / 2} \equiv\left(\frac{a}{p}\right) \quad(\bmod p)= \begin{cases}1 & \text { if } a \text { is quadratic residue modulo } p \\ -1 & \text { if } a \text { is quadratic non-residue modulo } p\end{cases}
$$

Remark 1. Fermat's Little Theorem has been mentioned and used a lot during the proof of Euler's Criterion showing the importance of Fermat's Little Theorem in the field of number theory. It also appears in abstract algebra and other fields of mathematics.

### 4.3.2 Algorithm

The inversion (decryption) algorithm is given in Rabin's original paper [17, p. 6-9] and the two books by Galbraith [7, p. 492] and Menezes et al [11, p. 292].

In order to decrypt the ciphertext, we have to find four values of the quadratic residue modulo $n$. Since $n$ has two factors $p$ and $q$, the equations $m^{2} \equiv c(\bmod n)$ will have four solutions. In order to find each of them, we need to find the solutions for the system

$$
\begin{cases}m^{2} \equiv c & (\bmod p) \\ m^{2} \equiv c & (\bmod q)\end{cases}
$$

Given by section 4.2.2 there exists a solution $m_{0}$ for the equation $m^{2} \equiv c(\bmod n)$. That implies

$$
\begin{cases}m_{0}^{2} \equiv c & (\bmod p) \\ m_{0}^{2} \equiv c & (\bmod q)\end{cases}
$$

By lemma 1 we know that for modulo $p$ there exist two incongruent solutions for the first equation. Similarly, there are two incongruent solutions for the second equation modulo $q$. By CRT we can recombine these into four solutions modulo $n$.

Using CRT, we need to find particular solution of each factor, namely, $m \equiv m_{p}$ $(\bmod p)$ and $m \equiv m_{q}(\bmod q)$.

Since $p, q \equiv 3(\bmod 4)$, compute the square root of $c$ modulo $p$ and $q$ using the formula

$$
\begin{cases}m_{p} \equiv c^{(p+1) / 4} & (\bmod p) \\ m_{q} \equiv c^{(q+1) / 4} & (\bmod q)\end{cases}
$$

The proof of this formula is in Rosen's book [18, p. 423-424]. We only prove it for $m_{p}$, similar arguments can be used for $m_{q}$.

Recall that $c^{(p-1) / 2} \equiv 1(\bmod p)$ by the Euler's criterion. We will prove that $c^{(p+1) / 4}$ is indeed particular solution to $m^{2} \equiv c(\bmod n)$. This follows from

$$
\left(c^{(p+1) / 4}\right)^{2}=c^{(p+1) / 2}=c^{p / 2+1 / 2}=c^{(p / 2-1 / 2)+1}=c^{(p-1) / 2} \cdot c \equiv 1 \cdot c=c \quad(\bmod p) .
$$

Similar computation can be done for $\left(c^{(q+1) / 4}\right)^{2} \equiv c(\bmod q)$.
If $p$ or $q \not \equiv 3(\bmod 4)$, we can some modular square root algorithms to search for these square roots, which will be discussed below.

We have successfully extracted two square roots modulo $p$ and $q$. Now, in order to solve the system, or solve this equation $m^{2} \equiv c(\bmod n)$, somehow combine these solutions in order to have four possible candidates. Note that if $x_{0}$ is the solution to $m^{2} \equiv c(\bmod n)$ then one other solution will be $-x_{0}(\bmod n)$, or simply $n-x_{0}(\bmod n)$.

We continue with the decryption algorithm. First, use the extended Euclidean algorithm to find $y_{p}$ and $y_{q}$ such that $y_{p} \cdot p+y_{q} \cdot q=1$.

Then, use the formula given by CRT to find four square roots of $c$ modulo $n$

$$
\left\{\begin{aligned}
m_{1} & \equiv y_{p} \cdot p \cdot m_{q}+y_{q} \cdot q \cdot m_{p} \quad(\bmod n) \\
m_{2} & \equiv n-m_{1} \quad(\bmod n) \\
m_{3} & \equiv y_{p} \cdot p \cdot m_{q}-y_{q} \cdot q \cdot m_{p} \quad(\bmod n) \\
m_{4} & \equiv n-m_{3} \quad(\bmod n) .
\end{aligned}\right.
$$

After that, one of the four candidates will be the message. It is usually decided by extra data such as padding, headers and other attached information.

Example 4. Bob has received Alice's message $c \equiv 100(\bmod 4661)$. He knows the factorization of his secret key, which is $4661=59 \cdot 79$. Let $p=59$ and $q=79$ in this case. Now, he starts to compute the following

$$
\begin{aligned}
& m_{p}=100^{(59+1) / 4}=100^{60 / 4}=100^{15} \equiv 49 \quad(\bmod 59) \\
& m_{q}=100^{(79+1) / 4}=100^{80 / 4}=100^{20} \equiv 10 \quad(\bmod 79) .
\end{aligned}
$$

Then finding $y_{p}$ and $y_{q}$ using extended Euclidean algorithm:

$$
\begin{aligned}
79 & =59-20 \\
59 & =20 \cdot 2+19 \\
20 & =19+1 \\
1 & =20-19 \\
& =20-59+20 \cdot 2 \\
& =59-20 \cdot 3 \\
& =59-79 \cdot 3+59 \cdot 3 \\
& =59 \cdot 4+79 \cdot(-3)
\end{aligned}
$$

so $y_{p}=4$ and $y_{q}=-3$.
Then, using CRT

$$
\begin{array}{llrl}
m_{1}=-4 \cdot 59 \cdot 10+3 \cdot 79 \cdot 10 & \equiv & 4592 & (\bmod 4661) \\
m_{2}=4661-4692 & \equiv & 69 & (\bmod 4661) \\
m_{3}=-4 \cdot 59 \cdot 10-3 \cdot 79 \cdot 10 & \equiv & 10 & (\bmod 4661) \\
m_{4}=4661-10 & \equiv & 4651 & (\bmod 4661) .
\end{array}
$$

Now Bob have to use extra information that is sent by Alice in order to select which one is her favorite number. There is no way that Bob would know it is either 10, 69, 4592, 4651 that is Alice's original message since all of them squared give exactly 100 modulo 4661. In that case, further instruction or information is needed.

### 4.4 Security

Rabin's scheme has been proven to be secure, unlike RSA where no such proof exists yet. According to Rabin, "breaking the RSA function is at most as hard as factorization, but is not known to be equivalent to factorization [...]" [17, p. 1-2]. The original proof of security is in his paper. The idea of his proof is that the inversion or brute-forcing the message is harder, i.e. taking more steps or longer time to perform, than factoring a number. This means breaking Rabin's cryptosystem is equivalent to integer factorization.

Lemma 3 ([9, Lemma A.69, p. 428-429]). Let $n=p q$ with $p, q$ are distinct primes. Let $u, v$ be the quadratic residue modulo $n$ with $u \not \equiv \pm v(\bmod n)$. Then the prime factors of $n$ can be computed from $u$ and $v$ using Euclidean Algorithm.

Proof. The proof is inspired by the book [9, p. 429].
Since $u$ and $v$ are quadratic residue modulo $n$ i.e. there exists $a \in \mathbb{Z}$ such that $a^{2} \equiv u$ $(\bmod n)$ and $a^{2} \equiv v(\bmod n)$. By the property of modular arithmetic, $u^{2}-v^{2} \equiv 0$ $(\bmod n)$.

We have $n=p q$ and

$$
\left\{\begin{array}{llll}
u \not \equiv v & (\bmod n) & \text { (given) } & \Longleftrightarrow p q \nmid u-v \\
u \not \equiv-v & (\bmod n) & (\text { given }) & \Longleftrightarrow p q \nmid u+v \\
u^{2} \equiv v^{2} & (\bmod n) & \text { (quadratic residue of } n) & \Longleftrightarrow p q \mid(u-v)(u+v)
\end{array}\right.
$$

Notice that $n=p q$ and $p, q$ are distinct primes, which means that the factors of $n$ are $1, p, q, p q$. Furthermore, $\operatorname{gcd}(u+v, n) \neq p q$ since $p q \nmid u+v$ and $\operatorname{gcd}(u+v, n) \neq 1$ since that means $\operatorname{gcd}(u-v), n)=p q$, which is not possible. Hence, the computation of $\operatorname{gcd}(u+v, n)$ yields one factor of $n$.

Example 5. Now, let $u=m_{1}$ and $v=m_{3}$ in the decryption algorithm and example 4. Now, $m_{1}-m_{3} \equiv 2 y_{q} \cdot q \cdot m_{p}(\bmod n=p q)$. Which means that

$$
\operatorname{gcd}(u-v, n)=\operatorname{gcd}\left(2 y_{q} \cdot q \cdot m_{p}, p q\right)=q
$$

is one factor of $n$. Similarly, computing $m_{1}+m_{3}$ yields the other factor of $n$.
Without loss of generality, this procedure can be extended for all four quadratic residues modulo $n$.

Example 6. Let $n=4661$ as in the decryption procedure and assume that we do not know the factorization of $n$ in this case. Let assume that we found two quadratic residue of $n: u=69, v=10$ such that $u^{2} \equiv v^{2} \equiv 69^{2} \equiv 10^{2} \equiv 100(\bmod n)$. Computing

$$
\operatorname{gcd}(69-10,4661)=\operatorname{gcd}(59,4661)=59
$$

which will be one factor of $n$, and

$$
\operatorname{gcd}(69+10,4661)=\operatorname{gcd}(79,4661)=79
$$

will be the other factor.
Lemma 3 implies that given the ability to find the square root modulo $n$, one can factorize $n$. Conversely, the ability to factorize $n$ yields the quadratic residue of $n$, as described in the decryption procedure of Rabin's cryptographic technique. [9, p. 428] This means cracking Rabin's scheme is equivalent to integer factorization, which means the cryptosystem is provably secure. However, the same thing cannot be said for RSA since there is no such proof exists for both factorization equivalency and no proof for other possible ways to attack RSA.

## 5 Problems

### 5.1 Uniqueness of Ciphertext

The size of public key and the sending message also matters. Unlike RSA where different keys generate different ciphertexts from the same message, using Rabin we only compute $m^{2} \equiv c(\bmod n)$ so sometimes one might compute the same ciphertext even with different keys of $n$. This is bad news. These similar ciphertexts is a weak point to attack the communication netowrk with bad Rabin implementations, especially with data harvesting techniques to gain valuable information without decrypting the messages or requiring to know the content of the conversations.

One might suggest that increasing the key size could potentially mitigate or even protecting these mediocre implementations. This is not the whole picture. To understand this, we shall introduce the relationship between message and key size in the context of Rabin's cryptosystem.

### 5.1.1 Message and Key size

Definition 7 (Small-sized message (SSM)). A small-sized message is $m$ such that $m^{2}<n$.

Example 7. Let $m=3, n=21$. Then $c \equiv m^{2}(\bmod 21) \equiv 9(\bmod 21)$.

SSM allows one to significantly decrease the possibility of collision between text since one could potentially filter the output by picking the smallest possible candidate. However, SSM is intended for demonstration only. In reality, it sabotages Rabin's scheme by making it easier to compute the quadratic residue by simply computing the square root without brute forcing. As shown in the example, Eve does not need to know the factorization of $n$ nor computing the quadratic residue of $c$, which is a hard problem. If the encryption key is larger than the ciphertext, then Eve simply read the original message by computing the squareroot of the ciphertext without even using CRT to recover the other 3 outputs, which she can easily recognize since $c$ is a perfect square. We have $m^{2} \equiv c(\bmod n)$. According to the rule of modular arithmetic, $m^{2}=n q+c$. Since $m^{2}<n, q=0$. Then $m^{2}=c$.

Definition 8 (Large-sized message (LSM)). A large-sized message is $m$ such that $m^{2} \geq n$.
Example 8. Let $m=9, n=21$. Then $c \equiv m^{2}(\bmod 21) \equiv 18(\bmod 21)$.
From the definition, the message is generally considered large-sized if it is large enough relative to the key. In order to generate a LSM, one can pad the message until it fits the key size, or one can reduce the key size if possible. However, both of these approaches have their own advantages and disadvantages.

However, a message cannot be represented by an integer that exceeds $n$ since everything is computed modulo $n$. Picking this makes the decryption impossible. Let us demonstrate this with a simple example.

Example 9. Let $m=30, n=21$. Then $c \equiv m^{2}(\bmod 21) \equiv 18(\bmod 21)$. We have $30 \equiv 9(\bmod 21)$. Computing the quadratic residue of 18 gives us four different results: $12,9,-9,-12(\bmod 21)$, and none of these gives the original message, which is 30 . The only clue we know is that the original message is in the residue class of 9 modulo 21.

We can generalize this result with a simple proof. Consider the message $m^{2}=c>n$. Prior to the encryption routine, compute $m \equiv m^{\prime}(\bmod p)$. Now, if $m>n$, we have that $m=q n+r, q>1$, therefore $m \neq r$. That means in the decryption routine, we get $c^{-1} \equiv r(\bmod n)$, and since $r \neq m$, we lose the original message.

This is why we only have SSM and LSM but not MSM (medium-sized message) because that would make the our definition redundant.

### 5.2 Four Different Outputs

### 5.2.1 Problem

From what we have seen, solving the quadratic residue of the cipher text always gives us four different unique outputs, and they are all valid when we perform the check for quadratic residue, namely $x^{2} \equiv c(\bmod n)$. This is not ideal for a cryptosystem and therefore requires a solution to differentiate between desired plaintext and other candidates. The usual method could be the sender informs how the condition of her message, by adding extra information, which, is the base idea for padding.

### 5.2.2 Padding

A commonly suggested solution is padding which is the practice of adding data to the original message prior to the encryption. The padded data can appear before, inside or after the message. Sometimes the padding itself is adding nonsensical data to obscure the message. Sometimes it works like a signature, or a standard form to recognize where the message should start and end, which is surrounded by some keywords. Reading this description may raise some confusion between padding and steganography as their principles are somewhat similar. However, steganography's objective is concealing the content, while padding is adding extra data.

There have been many padding schemes introduced over time. Consider a classic example of writing a letter.

Thursday, January 1, 1970
Dear Alice,
Happy New Year. Unfortunately I will not be able to send you an e-mail this year due to heavy network traffic. I hope everything goes well this year.
Sincerely yours,
Bob
From this example, we can see that the letter from Bob contains the metadata, namely, the date. We consider in this case the message starts after "Dear Alice" ends before "Sincerely yours" which suggest that they are the way of "padding the letter." In principle, not only they make the letter looks more standardized but also better format and follows some comprehension, which makes it easy to read in a typical fashion. It helps Alice to know what she is reading, what she would expects from Bob and where the main message should be.

The letter example also demonstrate how padding in modern cryptography works. It adds some data which signal standard of padding scheme, a bit of random data and also some data to tell where the message begins and ends. For example, consider Alice wants to send Bob her phone number, $0-123-456-7890$, the 0 suggest that where the actual phone number starts. If Bob wants to call her from another country, he needs to know which country Alice is currently living and replace the 0 pad with her country code, e.g $0046-1234567890$. For this Rabin's cryptosystem, since there are 4 different outputs of the same message, a padding is necessary to distinguish between the actual message and garbage data itself. We suggest a padding scheme as the following and call it Playground Padding.

Definition 9 (Playground Padding procedure). Let $m$ be the message written in decimal base, we think of $m$ as a string of decimal digits, or $m=\overline{m_{1} m_{2} \ldots m_{k}}$. The playground padding scheme for the message is defined as the following:

1. Concatenate $i$ digits 1 as prefix of $m$.
2. Concatenate $i$ digits 1 as suffix of $m$.
3. The message is now in the form of $\underbrace{1 \ldots 1}_{i \text { digits of } 1}\|m\| \underbrace{}_{i \text { digits of } 1_{1 \ldots 1}^{1 \ldots 1}}$, we could denote as $\overline{1 \cdots 1 m_{1} \ldots m_{k} 1 \ldots 1}$, in this case $\|$ denotes the string concatenation.
4. Encode the message into whatever base desired to send over the network.

Denote $(\mathrm{PP})_{i}, i=1,2, \ldots$ for $i$ digits padded before and after the original message.

### 5.2.3 The Experiment

We ran the simulations with encrypting the Alice's phone number with padding, ranged from one to five padded 1 into the phone number and test how many collisions happen. Of course, the padding scheme and the message is written in decimal and not binary. The key itself using the in the experiment is randomly generated to ensures every outputs are random and does not collide with itself and each padding scheme repeats maximum $i$ times.

Padding a number of digit 1 into the message, preferably two digits 1 before and after the message. For instance, Alice's phone number is 01234567890 , after padding with $(\mathrm{PP})_{1}$ becomes $\overline{1012345678901}$, after (PP) $)_{2}$ it becomes $\overline{110123456789011}$ and so on. Adding more digits reduces the probability of ambiguous message, however it comes at the cost of storing, processing and sending the message. Furthermore, the key has to be large enough, otherwise it could be lost in transmission if the modulo is smaller than the message itself.

The key for this experiment will be randomly generated and every message fits LSM's criteria to ensure fair testing. The simulation code was written in Mathematica (see Appendix 8.1.2).

### 5.2.4 Result



Figure 1: Playground Padding and Collisions

We plot the number of collisions against the number of $(\mathrm{PP})_{i}$ in figure 1. After some trials and errors in the test run, we decided that one millions trials for each $(\mathrm{PP})_{i}$, for
$1 \leq i \leq 5$. As seen on the graph, the number of collision candidates decrease significantly as the $i$ increases which suggests that collision rate for $(\mathrm{PP})_{1}$ is about $0.06,(\mathrm{PP})_{2}$ is 0.0006 and $(\mathrm{PP})_{3}$ is $6 \cdot 10^{-6}$ and inconclusive for $(\mathrm{PP})_{4}$ and $(\mathrm{PP})_{5}$.

Based on the trend observed, we project that for every consecutive $n$, the collision probability decreases by $10^{-2}$. This means increasing the $n$ decreases the number of collision significantly. For this, we conclude that even for a simple playground scheme, this idea of padding to increase the message recognition chance increases significantly.

This collision percentage fits our prediction. Consider the case for $\mathrm{PP}_{1}$, Alice will send Bob the padded $m_{p}=\overline{1 m 1}$ where $m=\overline{x_{1} x_{2} \ldots x_{k-1} x_{k}}$ is her original message represented with $k$ digits in decimal. Since we use the LSM standard for communication, we know that $m_{p}^{2} \leq\left(2 \cdot 10^{k+2}\right)^{2} \leq\left(5 \cdot 10^{2 k+3}\right)$. Therefore our sample space will be around $5 \cdot 10^{2 k+3}$ numbers. Based on the algorithm, we always obtain one answer in the form of $\overline{1 m 1}$, the three other candidate needs to have different form in order for us to build a system to recognize the message. However, for $x<5 \cdot 10^{2 k+3}$, there are numbers such as $11,101, \ldots, 191,1001, \ldots 1991, \ldots$, meaning that there are $1+10+100+\cdots=$ $\sum_{i=1}^{2 k+2} 10^{i}=10 \cdot \frac{10^{2 k+2}-1}{10-1}$ candidates.

The probability of getting a collision is

$$
\frac{10 \cdot \frac{10^{2 k+2}-1}{10-1}}{5 \cdot 10^{2 k+3}}
$$

and probability of not getting a collision is

$$
1-\frac{10 \cdot \frac{10^{2 k+2}-1}{10-1}}{5 \cdot 10^{2 k+3}}
$$

so the probability of not getting a collision thrice is

$$
\left(1-\frac{10 \cdot \frac{10^{2 k+2}-1}{10-1}}{5 \cdot 10^{2 k+3}}\right)^{3}
$$

and the probability of getting at least one collision is

$$
1-\left(1-\frac{10 \cdot \frac{10^{2 k+2}-1}{10-1}}{5 \cdot 10^{2 k+3}}\right)^{3}
$$

under the assumption of a uniform distribution of digits. In order to calculate the
probability, we compute the following limit

$$
\begin{aligned}
& \lim _{k \rightarrow \infty}\left(1-\left(1-\frac{10 \cdot \frac{10^{2 k+2}-1}{10-1}}{5 \cdot 10^{2 k+3}}\right)^{3}\right) \\
= & \lim _{k \rightarrow \infty}\left(1-\left(1-\frac{10 \cdot\left(10^{2 k+2}-1\right)}{45 \cdot 10^{2 k+3}}\right)^{3}\right) \\
= & \lim _{k \rightarrow \infty}\left(1-\left(1-\frac{10}{45} \cdot \frac{10^{2 k+2}-1}{10^{2 k+3}}\right)^{3}\right) \\
= & \lim _{k \rightarrow \infty}\left(1-\left(1-\frac{10}{45} \cdot \frac{1}{10}-\frac{10}{45} \cdot \frac{1}{10^{2 k+3}}\right)^{3}\right) \\
= & \lim _{k \rightarrow \infty}\left(1-\left(1-\frac{1}{45}-\frac{1}{45 \cdot 10^{2 k+2}}\right)^{3}\right) \\
= & 1-\left(1-\frac{1}{45}\right)^{3}=1-\left(\frac{44}{45}\right)^{3}
\end{aligned}
$$

which is approximately 0.06 or $6 \%$ in our case, as expected. Without loss of generality, we can expand this argument to see that the rate of collision of $\mathrm{PP}_{2}$ is $6 \cdot 10^{-4}$ or $0.06 \%$, of $\mathrm{PP}_{3}$ is $6 \cdot 10^{-6}$.

On the contrary, there are some drawbacks with padding. The first obvious is that message length increases significantly, in this case by around 100 times for every two padded digits. This means we also need to select bigger key size to adapt padded messages and possibly longer processing time and storage required as well. This performance penalty suggests that designing a padding scheme is not an easy job. That is why people have to compromise to have a balance between security and efficiency.

Of course, the name playground padding suggests that the algorithm is for testing and should not be used to develop any standard. This padding scheme is not provable secure but probably the inverse. As the messages get longer, the storage and bandwidth requires for storing them increases. However, processing data in decimal form is not ideal for computer since using binary is better. On the extra note, $\mathrm{PP}_{i}$ scheme can be done in any base with little to no changes.

### 5.3 Malleable attack

Since Rabin's cryptosystem is similar to RSA, one of its textbook weaknesses is the vulnerability to malleable attack.
Definition 10 (Malleability [16, p. 192]). In cryptography, the cryptosystem is malleable if the attacker is able to transform a ciphertext to another ciphertext which decrypt to a related plaintext.

That means the attacker does not neccesary have the knowledge of the plaintext itself but transformation of that plaintext into a related ciphertext. The idea of padding
stems from the concept of large size messages, namely, to make it large enough so that it is significantly harder to alter the ciphertext into something else. To understand how malleability works, we illustrate using the example, inspired the lecture note [1, p. 2].

Suppose Alice wants to send Bob message $m$ with public exponent $e=2$ and modulus $n$, she computes $c=m^{2}(\bmod n)$ and send $c$ as the ciphertext to Bob. Now, Eve wants to compute the transformation of $m$, say $f(m)=k m(\bmod n)$. She asks Bob to decrypt $c^{\prime}=k^{2} \cdot c(\bmod n) \equiv(k \cdot m)^{e}(\bmod n)$ and send her the results. We assume that Eve does not know the factorization of 4661, otherwise Eve does not need to use this trick.

Example 10 (No padding). We come back to the example described in the Decryption section. In this scenario, Alice does not use PP procedure. Now, somehow Eve intercepts Alice's ciphertext to Bob and asks him to compute the decryption of $f(m)=3^{2} c=9 c=900$ (mod 4661) instead. After decryption, Bob gives Eves four candidates 4631, 30, 207, 4454 for which Eve's simply multipliying each element by $3^{-1} \equiv 1554(\bmod 4661)$ and also obtain four possible integer candidates which are 10, 69, 4592, 4651.

Example 11 (With padding). It is arguably better for Eve to factorize integers on average instead of developing a clever way to defeat padding, even with simple ( PP$)_{n}$. First, she needs to makes sure that $k m$ for $k=2,3, \ldots$ fits the LSM standard, which is impossible to know since she does not know the message. Second, if $k m>n$, she could potentially pick $k=1 / l$ for $l=2,3, \ldots$. However, she is required to know the factorization of $c$ to correctly pick $l$, since $k^{2} \mid c$.

If Alice use $(\mathrm{PP})_{1}$ and send Bob $m^{2}=1691^{2} \equiv 2288(\bmod 4661)$. Let $k=3$, she asks Bob to decrypt $c^{\prime}=9 c \equiv 1467(\bmod 4661)$. Bob then gives Eve four candidates: 1757, 2904, 3258, 1403 and Eve only needs to multiply these by the $3^{-1} \equiv 1554$ (mod 4661), which has been demonstrated in example 10. Although Playground Padding does not mitigate much of the attack from Eve, it requires Eve to ask Bob decrypting the ciphertext and finding the modular inverse of her chosen number (in this case 3 ), which costs Eve some extra computing power.

It is no surprise that in practice, the textbook version of Rabin or RSA is never used. To help RSA more secure, there are many padding schemes that developed to make RSA secured against this attack. The most widely known and used is PKCS\#1, the first Public-Key Cryptography Standards. But not all padding schemes are secure as it might sound. For example, the old version v1.5, published in 1993, has a several weaknesses discovered throughout it existence [6]. The newest version of PKCS\#1 is v2.2, published in 2012 [14]. Potentially, we can use this for RSA padding for Rabin, given that they are just similar enough, but not equivalent. Of course, there are some other padding schemes that has been developed specifically for Rabin's cryptosystem like $\operatorname{HIME}(\mathrm{R})$ [2, p. 4].

## 6 Other choice of primes

One could propose that choosing primes such that $p \not \equiv 3(\bmod 4)$ and $q \not \equiv 3(\bmod 4)$, for example, $p \equiv 1(\bmod 4)$ or $p \equiv 1(\bmod 8)$. In that case, the formula for decryption in
section 4.3.2 does not apply. However, there are some algorithms capable of computing the quadratic residues such as Cipolla's, Tonelli-Shanks and Berlekamp-Rabin. TonelliShanks algorithm will be used to demonstrate the finding of quadratic residues when the factorization of the key is known.

### 6.1 Tonelli-Shanks Algorithm

Tonelli-Shanks algorithm is an algorithm to determine the square root of a given integer $n$ modulo prime $p$. It is best to use it for $p \equiv 1(\bmod 4)$ since there is already a formula to directly compute the square root of $n$ given that $p \equiv 3(\bmod 4)$ given in section 4.3.2. It was developed by Alberto Tonelli in 1891 and later improved by Daniel Shanks in 1973 (Shanks called this RESSOL).

In 1999, Ezra Brown published his paper "Square Roots from 1; 24, 51, 10 to Dan Shanks" in The College Mathematics Journal which explained several concepts for the Tonelli-Shanks algorithm [5]. It featured the core idea, stems from Lemma 2 and the proof of this algorithm.

Note that Lemma 2 is not true for composite $n$ since the ability of computing quadratic residue modulo $n$ implies the ability to factorize $n$. This has been shown in section 4.4 since we needs to use CRT to find all the quadratic residues modulo $n$ instead of taking $\pm \sqrt{c} \equiv m(\bmod n)$ as the inversion algorithm, since there exists two more quadratic residues of composite $n$. An example is that $a= \pm 1$ and $a= \pm 473$ both satisfy $x^{2} \equiv c$ $(\bmod 4661)[5]$.

If $a$ has square roots by Euler's criterion, and $p$ is an odd prime then we can write $p-1=q \cdot 2^{s}$ with $s>0$ and $q$ is odd. Let $x=a^{(q+1) / 2}$, we know that $x$ is almost square root since

$$
x^{2} \equiv a^{q+1} \equiv a^{s} a \quad(\bmod p)
$$

and if $a^{s} \equiv 1(\bmod p)$ then $x$ is a square root, off by a fudge factor, and using the Tonelli-Shank algorithm to keep updating it until we identify the correct answer [5, p. 91].

The description of Tonelli-Shanks algorithm is based on the book [15, p. 112-114], the video [20] and the paper [5, p. 91-92].
Input

- $p$ : a prime
- $n$ : an integer


## Process

Step 1: Check
(a) $n$ is a square by computing Euler's criterion: check if $n^{(p-1) / 2} \equiv 1(\bmod p)$. Return nothing (or error) if true.
(b) $n$ divides $p$. Return $r=0$ if true.

Step 2: Find $q$ and $s$ such that $p-1=q \cdot 2^{s}$ and $q$ is odd by continuously factorizing 2 out of $p-1$.
(a) Initialize

- $q=p-1$
- $s=0$
(b) Loop until $2 \nmid q$
i. Assign $s+1$ to $s$
ii. Assign $q / 2$ to $q$

Step 3: Select $z$ to be a quadratic non-residue of $p$.
(a) Initialize $z=2$
(b) Loop until Euler's criterion of $z$ is not $1:\left(\frac{z}{p}\right) \not \equiv 1(\bmod n)$
i. Assign $z+1$ to $z$

Step 4: Initialize

- $m=s$
- $c \equiv z^{q}(\bmod p)$
- $t \equiv n^{q}(\bmod p)$
- $r \equiv n^{(q+1) / 2}(\bmod p)$

Step 5: Loop until $t=1$
(a) Find the least $i$ such that $0<i<m$ and $t^{2^{i}}=1$
i. Initialize

- $i=0$
- $t_{e}=t$
ii. Loop until $t_{e} \neq 1$
- Assign $i+1$ to $i$
- Assign $t_{e} \equiv t_{e}^{2}(\bmod p)$
(b) Assign
- $b \equiv c^{2^{m-i-1}}(\bmod p)$
- $m=i$
- $c \equiv b^{2}(\bmod p)$
- $t \equiv t \cdot b^{2}(\bmod p)$
- $r \equiv r \cdot b(\bmod p)$

Step 6: Return $r$.

We can optimize the algorithm by adding another check. If $p \equiv 3(\bmod 4)$ then simply computing $r \equiv n^{(p+1) / 4}(\bmod p)$ gives us the direct answer we found. However, we decide against adding it as this would add too many optimizations for the algorithm, which will be explained in the experiment later.
Output

- $r$ from the algorithm
- Error if terminate early.

On average, Tonelli-Shanks algorithm requires

$$
\begin{equation*}
2 m+2 k+\frac{s(s+1)}{4}+\frac{1}{2^{s-1}}-9 \tag{3}
\end{equation*}
$$

modular multiplications, with $m$ is the number of digits of $p$ and $k$ is the number of 1 in the binary representation of $p, s$ is the number of exponent $s$ in the algorithm such that $p-1=q 2^{s}[21, \mathrm{p} .431]$. On the extra note, this is an probabilistic algorithm since it requires us to find quadratic non-residue modulo $p[10, \mathrm{p} .2]$. There is no known method to find the quadratic residue nor non-residue deterministically (else we would be able to defeat Rabin's scheme). This is why the performance is given in the average case, not the usual best, worst and average cases like most of the deterministic algorithms.

### 6.2 Proof of Tonelli-Shanks algorithm

The proof is based on the wiki [8] and Brown's paper [5, p. 91-92]. The strategy is to show that the algorithm will halt at some point and the correctness of output.

First, we want to show that these three loops invariants hold.

- $c^{2^{m-1}} \equiv 1(\bmod p)$
- $t^{2^{m-1}} \equiv 1(\bmod p)$
- $r^{2}=t n$

We have $z$ is the quadratic non-residue and $n$ is the quadratic residue, the initialization goes as follow:

- $c^{2^{m-1}}=z^{q^{s-1}}=z^{\frac{p-1}{2}} \equiv-1(\bmod p)($ since $z$ is a quadratic non-residue)
- $t^{2^{m-1}}=n^{q^{s-1}}=n^{(p-1) / 2} \equiv 1(\bmod p)$ (since $n$ is a quadratic residue)
- $r^{2}=n^{q+1}=t n$

After every loop, by substituting $c, m, t, r$ with $c^{\prime}, m^{\prime}, t^{\prime}, r^{\prime}$ after each iteration, we obtain the following

- We assign $b \equiv c^{2^{m-i-1}}(\bmod p)$.
- We find $i$ such that $0<i<m$ and $t^{2^{i}} \equiv 1(\bmod p)$. This guarantees step $5(\mathrm{a}) \mathrm{ii}$ with the check $t_{e}=1$ will terminate and return the least $i$ since we search every $i$ from 1 and up.
- We assign $m=i$ to be $m^{\prime}$.
- We assign $c \equiv b^{2}(\bmod p)$ to be $c^{\prime}$ then

$$
c^{\prime 2^{m^{\prime}-1}}=\left(b^{2}\right)^{2^{i-1}}=c^{2^{m-i} 2^{i-1}}=c^{2^{m-i+i-1}}=c^{2^{m-1}} \equiv-1 \quad(\bmod p)
$$

- We assign $t \equiv t b^{2}(\bmod p)$ to be $t^{\prime}$ then

$$
t^{2^{m^{\prime}-1}} \equiv\left(t b^{2}\right)^{2^{i-1}} \equiv t^{2^{i-1}} b^{2^{i}} \equiv-1 \cdot-1 \equiv 1 \quad(\bmod p)
$$

- Since $i$ is the smallest integer such that $t^{2^{i}} \equiv 1(\bmod p)$, that means $t^{2^{i-1}} \not \equiv 1$ $(\bmod p)$, therefore $t^{2^{i-1}} \equiv-1(\bmod p)$, per Euler's criterion.
- $b^{2^{i}}=c^{2^{m-i-1} 2^{i}}=c^{2^{m-1}} \equiv-1(\bmod p)$

Also, since $t^{2^{m-1}} \equiv 1(\bmod p)$, we always find

- We assign $r \equiv r \cdot b(\bmod p)$ to be $r^{\prime}$ then

$$
r^{\prime 2} \equiv r^{2} b^{2}=t n b^{2} \equiv t^{\prime} \cdot n \cdot 1 \equiv t^{\prime} n \quad(\bmod p)
$$

Since we know that $t^{2^{m-1}} \equiv 1(\bmod p)$, the test against $t=1$ at step 5 ensure that we have the condition to halt. Also, since we always find $i$ such that $0<i<m$ and assign new value of $i$ to $m, m$ decreases every iteration and the algorithm stops. At the end, we output the invariant $r$ such that $r^{2} \equiv n(\bmod p)$.

### 6.3 The Experiment

Similar the experiment in section 5.2.3, we decided to test the performance of the algorithm in a small experiment. There are two experiments, theoretical and practical performance. First, we decide on how many primes to measure since it is impractical to test every prime number in existence within certain ranges. After some trial and error, we limit the range of testing as follow

- For prime $p$ such that $0<p<10^{5}$, we test all these primes in this range.
- For $10^{5} \leq p \leq 10^{50}$, we randomly choose 10000 primes for each digit increment in decimal representation.
- We decided not to go further as 166 binary digits are large enough for our case.

We subdivide each prime by its number of digits in decimal base into each own group. We perform the theoretical and practical experiment for each group.

For the theoretical part, using equation (3) for each prime and then calculate the average for each group.

For the practical part, using randomly generated a random square for each prime such that they fit the LSM standard and run the algorithm to compute quadratic residue and measure how many loop counts. According to the algorithm design, we will count four loops

- the $p$-loop (find $p-1=q 2^{s}$ )
- the $z$-loop (find the least quadratic non-residue)

- the $t$ loop (run until $t=1$ )
and sum all up for each prime. Then, taking the average loop count for each group of primes.

We believe this is the best way to measure runtime because using the timer (measure how long each computation run) could give us different result every runtime. There are also many essential background processes to keep the operating system running (and Mathematica) that could affect the result. Measuring total loop runs yields an estimation on the performance of our implementation of Tonelli-Shanks algorithm regardless of the machine that we are using.


Figure 2: Theoretical test (average)


Figure 3: Practical test (average, standard deviation)


Figure 4: Practical test (maximum, minimum, average)

### 6.4 Result

The number of modular multiplications in the theoretical test increases linearly, while the practical test averages around 7 loop counters for almost all cases, except the case of two digit primes (which the average is 6 ) so we can safely ignore that as it is not significant. However, even with the all loops counted, once we look back at the maximum and minimum loop counts in the practical test, we notice that the average is significantly closer to the minimum loop count than maximum one. This indicates even something is not right with the design of this experiment, suggesting that something too much optimization has been done and other factors which we did not include. The prime suspect for this is the function PowerMod as we do not know how it works nor can we count how many "modular multiplications" as described in the theoretical test. Therefore, surprising that the experiment yield this behavior as the theoretical test expects a vastly different result from the practical one. This behavior is expected.

Zooming out the graph of the practical test with the maximum loop counts tells us a completely different story. The maximum loop counts vary significantly between each group but never exceed 200 in our case. This further confirms our suspicion of the optimization for the PowerMod function.

Testing Tonelli-Shanks algorithm in a modern environment presents new challenges. First of all, the data set is large so the waiting time for simulation was long. Obviously, it is simply impractical to test all primes in a given range so the limitation was introduced. Second of all, Mathematica is a closed source software, which means that our functions use in this case is only what we have. There are too many optimizations. For example, the PowerMod which only returns the result but does not tell us how many multiplications it takes in the CPU. Of course, with great numbers comes great computing time. But if the question is how great that is then it is not possible for us and outside of the scope of this paper to answer that question.

On the extra note, the theoretical Tonelli-Shanks algorithm yields a linear graph when it comes to measuring performance is interesting. We do not know whether this behavior is expected in our data set the test itself is too good to be true. There is not much information exists in this case to explain this linear behavior. On the practical side of the test, we should expect similar behavior if we are able to include the loop counts of the PowerMod function.

## 7 Discussion

Throughout the thesis we have discussed the history, theory, algorithm and some surrounding problems that Rabin's scheme trying to solve and arise in the process. Of course, the amount of accompanied theories make it impossible to put everything into section 3 and instead write the necessary one prior to the start of the description. This strategy however has one major flaw, that is, making the process of locating theory much harder since it distributes different definitions, lemmas, theorems, etc. throughout the paper. Considering the target audience for this paper, we think that this strategy of
distributing is better than unifying every background theories into one large section.
It is quite sad to see such a robust cryptosystem, provably secure yet not popular than RSA. It is much more simple to understand and implement than RSA. The proof that it is as secure as the integer factorization makes it more mathematically safe than the believably secure of RSA. However, we never said that RSA is a weak algorithm. The existence of belief and provability is interesting when it comes to applying mathematical concepts into the society. It is not the best algorithm that gets chosen. Sometimes, the good enough could be the best suited for the market. Understandably, the limiting in choosing primes, which, could be solved using RESSOL and four candidates as output meaning that implementing a good system to based on Rabin's scheme could be more complex than RSA, potentially sabotaging the communication if too much information is exchanged over the public network on finding the correct key.

The limitation of both time and computing resource meaning that we cannot perform large enough scale testing for $\mathrm{PP}_{i}$ and RESSOL. However, looking at the result, it is promising for us to have an efficiently designed system to communicate based on Rabin's scheme. We conclude that this is simply enough for the playground type of experiment to demonstrate the security and efficiency in the textbook version of Rabin, with some minor padding. In fact, there have been multiple attempts of developing a padding algorithms. The performance of RESSOL, one could expect more optimizations be introduced, not only the PowerMod itself but to the overall implementation.

## 8 Appendix

### 8.1 Mathematica code

### 8.1.1 Preamble

This should be included in the "Initialization cell" in order for the code to work.

```
SetDirectory[NotebookDirectory []];
file = FindFile["Functions/Tonelli-Shanks.nb"];
NotebookOpen[file, Visible -> True];
NotebookEvaluate[file, InsertResults -> True];
NotebookClose[file];
```


### 8.1.2 Textbook Simulation

```
p = 59;
q = 79;
n = p*q
(*Encryption *)
m}={69}
c}=\operatorname{PowerMod[m, 2, n]
a = 1; (*Change a = 3 for example for the malleability*)
Mod[m, n]
c = a^ 2*c;
(*Decryption*)
mp = PowerMod[c, (p + 1)/4, p];
mq}=\operatorname{PowerMod[c, (q + 1)/4, q];
{g, {yp, yq}} = ExtendedGCD[p, q];
m1 = Mod[yp*p*mq + yq*q*mp, n];
m2 = n - m1;
m}3=\operatorname{Mod}[yp*p*mq-yq*q*mp,n]
m4 = n - m3;
out1 = {m1,m2,m3,m4};
out1 = out1[[All, 1]];
ModularInverse[a, n];
Mod[out1*ModularInverse[a, n], n] (*Output*)
```


### 8.1.3 Padding test

```
(*Prime generator*)
m = {"01234567890" };
out5 = {};
Do[
m = StringJoin[" 1", m, "1"];
m}={\boldsymbol{ToExpression [m]};
out4 = 0;
Do[
(*Key generator, remember to pick large enough prime*)
p = RandomPrimeWithMax [m[[1]], 3, 4];
q = RandomPrimeWithMax [m[[1]], 3, 4];
n = p*q;
(*Encryption *)
c = PowerMod[m, 2, n];
(*Decryption, for m=3 (mod 4)*)
mp = PowerMod[c, (p + 1)/4, p]; mq = PowerMod[c, (q + 1)/4, q];
{g, {yp, yq}} = ExtendedGCD[p, q];
m1 = Mod[yp*p*mq + yq*q*mp, n];
n1 = IntegerDigits[m1][[1, 1]] = Mod[m1, 10][[1]] = 1;
m2 = n - m1;
n2 = IntegerDigits[m2][[1, 1]] = Mod[m2, 10][[1]] == 1;
m3 = Mod[yp*p*mq - yq*q*mp, n];
n3 = IntegerDigits[m3][[1, 1]] = Mod[m3, 10][[1]] == 1;
m4 = n - m3;
n4 = IntegerDigits[m4][[1, 1]] = Mod[m4, 10][[1]] == 1;
(*Output*)
mout = ToString /@ {m1[[1]], m2[[1]], m3[[1]], m4[[1]]}; (**
If using mod, remember to take the first element**)
comp1 = StringTake[ToString[m[[1]]], j];
comp2 = StringTake[ToString [m[[1]]], -j];
a1 = StringTake[mout[[1]], j]; b1 = StringTake[mout[[1]], -j ];
n1 = StringMatchQ[a1, comp1] && StringMatchQ[b1, comp2];
a2 = StringTake[mout[[2]], j]; b2 = StringTake[mout[[2]], -j];
n2 = StringMatchQ[a2, comp1] && StringMatchQ[b2, comp2];
a3 = StringTake[mout[[3]], j]; b3 = StringTake[mout[[3]], -j];
n3 = StringMatchQ[a3, comp1] && StringMatchQ[b3, comp2];
a4 = StringTake[mout[[4]], j]; b4 = StringTake[mout[[4]], -j ];
n4 = StringMatchQ[a4, comp1] && StringMatchQ[b4, comp2];
```

```
out1 = {m1, m2, m3, m4};
out2 = {n1, n2, n3, n4};
out3 = Count[out2, True];
If [out3 > 1, out4 += 1,
If[out3== 0,
Print["error", "\n", {i, m, mp, mq, yp, yq, out1}, "\n",
"detected"]]];
, {i, 1, 1000000}]; (*change 1000000 to how many test desired*)
m = ToString[m[[1]]];
```

AppendTo[out5, out4];
Print [\{m, out4 $\}]$;
, $\{\mathrm{j}, 1,5\}]$;
Print[out5];

### 8.1.4 Tonelli-Shanks algorithm

This is the file "Tonneli-Shank.nb" of the project.

```
(*Adapted from: https://www.youtube.com/watch?v=d7ZFCf95MAQ*)
(*Return: True if there is an integer a = x^2*)
EulerCriterionQ[n_, p_] :=
Module[{},
If [MOd[n, p] = 0, Return[True],
Return[PowerMod[n, (p - 1)/2, p]=1]
];
];
(*Return: {result, #loop, s}*)
TonelliShanks[n-, p_] :=
Module[{temp, q, s, m, b, c, i, t, r, z, count, mod4val},
mod4val = Mod[p, 4];
count = 1;
If [Mod[n, p] = 0, Return[{0, count, None }]];
If [Not[EulerCriterionQ [n, p]], Print["Not\_quadratic\_residue."];
Return[{None, None, None}];
];
(*Factorize q*)
q}=\textrm{p}-1
```

```
s = 0;
While [Mod[q, 2] = 0,
count += 1;
s += 1;
q}=\textrm{q}/2;]
(*Find quadratic nonresidue*)
z = 1;
While[EulerCriterionQ [z, p],
count += 1;
z += 1;
];
(*Let-part*)
m = s;
c = PowerMod[z, q, p];
t = PowerMod[n, q, p];
r = PowerMod[n, (q + 1)/2, p];
(*The loop*)
While[t != 1,
count += 1;
i = 0;
temp = t;
(*Repeated squaring*)
While[temp != 1,
count += 1;
i += 1;
temp = PowerMod[temp, 2, p];
];
b = PowerMod[c, Power[2,m-i - 1], p];
m = i;
c = PowerMod[b, 2, p];
t = Mod[t*b*b, p];
r = Mod[r*b, p];
(* Print [{m,r,b,t,c}];*)
];
Return[{r, count, s }];
];
```


### 8.1.5 Tonelli-Shanks Theoretical Test

```
FindS [p-] := Module[{q, s },
q}=\textrm{p}-1
s = 0;
While [Mod[q, 2] = 0,
s += 1;
q = q/2;];
Return [s];
];
aaa = OpenAppend["Output/outputTonelli5.txt"];
Do[
p = {};
If [i< < ,
q = Power[10, i - 1] - 1;
While[q< Power[10, i] - 1,
q = NextPrime[q];
AppendTo[p, q];
];
If [5<= i < 60,
p = RandomPrime[{10^(i - 1) - 1, 10^(i) - 1}, 10000];
, p = RandomPrime[{10^(i - 1) - 1, 10^(i) - 1}, 10000];
];
temp = DigitCount [p, 2];
m}=\mathbf{Total[temp, {2}];
k = temp[[All, 2]];
s = Map[FindS , p];
sum = 2m + 2 k + s (s - 1)/4 + 1/(Power[2, s - 1]) - 9;
mean = Mean[sum];
out2 = Round[mean];
Print[{i, out2 }];
WriteLine[aaa, {i, out2}];
, {i, 2, 50}];
Close[aaa];
```


### 8.1.6 Tonelli-Shanks Practical Test

```
aaa = OpenAppend["Output/outputTonelliPractical5.txt"];
Do[
p = {};
```

```
If[i< 5,
q = Power[10, i - 1] - 1;
While [q < Power[10, i] - 1,
q = NextPrime[q];
AppendTo[p, q];
];
If [5<= i,
Do[
q = RandomPrime[{10^(i - 1) - 1, 10^(i) - 1}];
AppendTo[p, q];
, {j, 1, 10000}
];
];
list1 = {};
list2 = Table[
TonelliShanks[
Power[RandomInteger[{Power[10, i - 1] - 1, Power[10, i] - 1}],
2], p[[i]]][[2]], {i, 1, Length[p]}];
out = {i, list2};
Print[out];
WriteLine[aaa, out];
, {i, 2, 50}
];
Close[aaa];
```


## References

[1] Alon Rosen and Salil Vadhan. Public-Key Encryption in Practice. Harvard University, Nov. 16, 2006.
[2] Muhammad Asyraf Asbullah and Muhammad Rezal Kamel Ariffin. Rabin-\$p\$ Cryptosystem: Practical and Efficient Method for Rabin Based Encryption Scheme. Nov. 17, 2014. arXiv: 1411.4398 [cs]. URL: http://arxiv.org/abs/1411.4398 (visited on 05/09/2021).
[3] Brilliant.org. Euler's Criterion. In: URL: https://brilliant.org/wiki/eulerscriterion/ (visited on 04/23/2021).
[4] Brilliant.org. Fermat's Little Theorem. In: URL: https://brilliant.org/wiki/ fermats-little-theorem/ (visited on 04/27/2021).
[5] Ezra Brown. "Square Roots from 1; 24, 51, 10 to Dan Shanks". In: The College Mathematics Journal 30.2 (Mar. 1999), pp. 82-95.
[6] Jean-Sébastien Coron et al. "New Attacks on PKCS\#1 v1.5 Encryption". In: Advances in Cryptology - EUROCRYPT 2000. Ed. by Bart Preneel. Red. by Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen. Vol. 1807. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 369381. URL: http://link.springer.com/10.1007/3-540-45539-6_25 (visited on 05/18/2021).
[7] Steven D. Galbraith. Mathematics of Public Key Cryptography. Cambridge ; New York: Cambridge University Press, 2012. 615 pp.
[8] HandWiki. Tonelli-Shanks Algorithm. In: URL: https://handwiki.org/wiki/ Tonelli\%E2\%80\%93Shanks_algorithm (visited on 05/14/2021).
[9] Hans Delfs and Helmut Knebl. Introduction to Cryptography: Principles and Applications. 3rd ed. New York, NY: Springer Berlin Heidelberg, 2015.
[10] Rajeev Kumar. An Algorithm for Finding Square Root modulo p. Oct. 31, 2020. arXiv: 2008.11814 [math]. URL: http://arxiv.org/abs/2008.11814 (visited on 06/02/2021).
[11] A. J. Menezes, Paul C. Van Oorschot, and Scott A. Vanstone. Handbook of Applied Cryptography. CRC Press Series on Discrete Mathematics and Its Applications. Boca Raton: CRC Press, 1997.
[12] Michael Penn. "Number Theory - Chinese Remainder Theorem Proof". url: youtube.com/watch?v=EolotL9HN8k (visited on 05/02/2021).
[13] Michael Penn. "Number Theory - The Legendre Symbol and Euler's Criterion". URL: https://www.youtube.com/watch?v=eKjjAr4EvmU (visited on 05/02/2021).
[14] K. Moriarty et al. PKCS \#1: RSA Cryptography Specifications Version 2.2. RFC8017. RFC Editor, Nov. 2016, RFC8017. URL: https://www.rfc-editor.org/info/ rfc8017 (visited on 05/18/2021).
[15] Ivan Morton Niven, Herbert S. Zuckerman, and Hugh L. Montgomery. An Introduction to the Theory of Numbers. 5th ed. New York: Wiley, 1991.
[16] Christof Paar and Jan Pelzl. Understanding Cryptography: A Textbook for Students and Practitioners. Heidelberg ; New York: Springer, 2010.
[17] Michael O. Rabin. "Digitalized Signatures and Public-Key Functions as Intractable as Factorization". In: Laboratory for Computer Science. Massachusetts Institute of Technology, Laboratory for Computer Science, 1979. URL: http://publications. csail.mit.edu/lcs/pubs/pdf/MIT-LCS-TR-212.pdf (visited on 05/17/2021).
[18] Kenneth H Rosen. Elementary Number Theory. 6th ed. Pearson, 2014.
[19] Alexander Stanoyevitch. Introduction to Cryptography with Mathematical Foundations and Computer Implementations. Discrete Mathematics and Its Applications. Boca Raton: Chapman \& Hall/CRC, 2011.
[20] Terry Jackson. "Finding Mod-p Square Roots with the Tonelli-Shanks Algorithm". Oct. 28, 2020. URL: https://www.youtube.com/watch?v=d7ZFCf95MAQ (visited on $05 / 17 / 2021$ ).
[21] Gonzalo Tornaría. "Square Roots Modulo p". In: LATIN 2002: Theoretical Informatics. Ed. by Sergio Rajsbaum. Vol. 2286. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 430-434. url: http: //link.springer.com/10.1007/3-540-45995-2_38 (visited on 05/14/2021).

## Linnæus University

Sweden
Faculty of Technology
SE-391 82 Kalmar | SE-351 95 Växjö
Phone +46 (0)772-28 8000
teknik@lnu.se
Lnu.se/fakulteten-for-teknik


[^0]:    ${ }^{1}$ See [18, p. 108-109] the proof of extended Euclidean algorithm.

