A tutorial on using Benaloh public key cryptosystem to encrypt text

M A Budiman$^1$ and D Rachmawati$^1$

$^1$Departemen Ilmu Komputer, Fakultas Ilmu Komputer dan Teknologi Informasi, Universitas Sumatera Utara, Jl. Universitas No. 9-A, Kampus USU, Medan 20155, Indonesia

Email: mandrib@usu.ac.id and dian.rachmawati@usu.ac.id

Abstract. The Benaloh public key cryptosystem is an asymmetric cryptosystem with the property of homomorphic encryption scheme. This study explains how to use Benaloh public key cryptosystem to encrypt a simple text. To increase the clarity of the computation, some intermediate variables which are not available in the original Benaloh cryptosystem have been used. The Benaloh algorithm has been implemented in Python programming language version 2.7.15.

1. Introduction

Securing confidential messages are essential in all digital communications. To preserve the confidentiality in today’s digital communication, modern cryptography comes into play. Modern cryptography use mathematical techniques and art to protect confidential messages. Most important mathematical techniques used in modern cryptography have come from the applied field of number theory.

The principal idea in modern cryptography is to encrypt a readable message into an obfuscated form. A readable message is called “plaintext” and its obfuscated form is called “ciphertext”. To transform plaintext into ciphertext, an encryption function and a secret key is used. The sender can then transmit the ciphertext by digital means to the intended recipient. The secret key is also transmitted to the recipient, but the transmission of the secret key must go through a different and securer channel. Upon receiving the ciphertext and the key, the recipient can make use of a decryption function to recover the plaintext. Mathematically speaking, the decryption function is the inverse of the encryption function [1][2].

A cipher is a set of two cryptographic algorithms; one is to perform the encryption and the other is to perform the decryption. If the encryption and decryption processes use the same key, then the cipher is said to be symmetric. All classical ciphers and all modern ciphers prior to 1976 are classified as symmetric ciphers.

The problem of symmetric cipher is the key should be distributed in a secure channel. A secure channel is often difficult and expensive to establish [3]. This problem was solved in 1976 with the introduction of the Diffie-Hellman key exchange protocol [4]. The other name for this protocol is Diffie-Hellman-Merkle protocol [5].

In the Diffie-Hellman-Merkle protocol, the notion of asymmetric cipher was introduced. In asymmetric cipher, different keys were used in the encryption and decryption processes; hence, the word
“asymmetric” is associated in the name. The key used in the encryption process is usually made public; thus, one might think that asymmetric cipher is synonymous with public key cryptosystem. However, this is not always the case. There is a cipher called Pohlig-Hellman which has two different keys for its encryption and decryption processes, but the two keys are kept private [6] [7]. A cipher which uses two different keys for encryption and decryption but none of the keys are public is called asymmetric non-public key cipher [8]. In fact, most public key cryptosystems can act as asymmetric non-public key cipher when needed.

The first algorithm which used the Diffie-Hellman-Merkle protocol is the Rivest-Shamir-Adleman (RSA) public key cryptosystem which was published in 1976 [9]. In 1984, Benaloh suggested a public cryptosystem which uses the RSA modulus, but can act as a homomorphic encryption scheme [10]. Homomorphic encryption scheme lets anyone to do some operations on the ciphertext and, as a result, gets a new ciphertext, which when decrypted it is as if the operations were done in the plaintext. This study describes how to use Benaloh public key cryptosystem to encrypt a simple text in step-by-step manner. The Benaloh public key cryptosystem has been implemented in Python programming language version 2.7.15.

2. Methods
In this section we explain the three phases of Benaloh cryptosystem, which are key generation, encryption, and decryption. As a scenario, suppose there are two people (Alice, the sender, and Bob, the recipient) that are willing to communicate securely using Benaloh cryptosystem.

2.1 The key generation phase
In the phase of key generation, Bob performs these acts:
1. Randomly, Bob generates \( p \) and \( q \), which are two large distinct prime numbers. This can be done using some primality test, such as Solovay-Strassen, Agrawal-Kayal-Saxena, and Fermat primality tests.
2. Bob computes \( n = pq \).
3. Bob computes \( \Phi(n) = (p - 1)(q - 1) \).
4. Bob chooses a block size \( r \) such that:
   a. \( r \) divides \( p - 1 \).
   b. \( \gcd(r, (p - 1) / r) = 1 \).
   c. \( \gcd(r, q - 1) = 1 \).
5. Bob chooses \( y \) such that \( x = y^{\Phi(n)/r} \mod n \neq 1 \).
6. Bob keeps \( (p, q) \) as the private keys.
7. Bob publishes \( (y, r, n) \) as the public keys.

2.2 The encryption phase
In the phase of encryption, Alice performs these acts:
1. Alice selects \( m \), which is the message to be secured.
2. Alice obtains Bob’s public keys, which are the values of \( (y, r, n) \).
3. Alice randomly choose \( u \), such that \( 0 < u < n \).
4. Alice computes the ciphertext \( c = y^m u^r \mod n \).
5. Alice sends Bob the ciphertext \( c \).

2.3 The decryption phase
In the phase of decryption, Bob performs these acts:
1. From Alice, Bob obtains the ciphertext \( c \).
2. Bob computes \( a = c^{\Phi(n)/r} \mod n \).
3. Bob lets \( md = 0 \). If \( x^{md} \mod n \neq a \), Bob increments \( md \) by 1. Bob keeps incrementing \( md \) by 1, until \( x^{md} \mod n = a \). If \( x^{md} \mod n = a \), then \( md \) is the original plaintext.

3. **Discussions**

In this section we explain how to use the Benaloh public key algorithm to secure a simple text. Suppose Alice wants to sends Bob a text message securely using the Benaloh public key algorithm. The message Alice wants to send is the letter “N”.

### 3.1 Key generation (Bob)

Firstly, Bob generates two prime numbers, \( p \) and \( q \). For tutorial purpose, here we only use 3-digit prime numbers.

\[
\begin{align*}
p &= 397 \\
q &= 191
\end{align*}
\]

Secondly, Bob computes \( n = pq \).

\[
n = pq = 397 \times 191 = 75827
\]

Thirdly, Bob computes \( \Phi(n) = (p-1)(q-1) \).

\[
\Phi(n) = (p-1)(q-1) = (397 - 1)(191 - 1) = 75240
\]

Fourthly, Bob choose a block size \( r = 99 \). Bob checks whether \( r \) satisfies the three rules.

\[
\begin{align*}
\text{Rule 1: } & r \text{ divides } (p-1). \text{ It means that } (p-1) \mod r = 0. \\
& (p-1) \mod r = (397 - 1) \mod 99 = 0. \text{ (OK)}
\end{align*}
\]

\[
\begin{align*}
\text{Rule 2: } & \gcd(r, (p-1) / r) = 1. \\
& \gcd(99, (397 - 1) / 99) = \gcd(99, 4) = 1. \text{ (OK)}
\end{align*}
\]

\[
\begin{align*}
\text{Rule 3: } & \gcd(r, q-1) = 1. \\
& \gcd(99, 191 - 1) = \gcd(99, 190) = 1. \text{ (OK)}
\end{align*}
\]

Fifthly, Bob chooses \( y \) such that \( x = y^{\Phi(n) / r} \mod n \neq 1 \). Bob choose \( y = 13213 \).

\[
x = y^{\Phi(n) / r} \mod n = 13213^{75240 / 99} \mod 75827 = 24640 \neq 1 \text{ (OK)}
\]

Sixthly, Bob keeps \( (p, q) \) as the private keys.

Seventhly, Bob publishes \( (y, r, n) \) as the public keys.

### 3.2 Encryption (Alice)

Firstly, Alice converts the text “N” into a number using an encoding table, such as ASCII table. In ASCII table, the symbol “N” has the value of 78. Thus, Alice lets \( m = 78 \).

Secondly, Alice obtains Bob’s public keys, \( (y, r, n) \).

\[
\begin{align*}
y &= 13213 \\
r &= 99 \\
n &= 75827
\end{align*}
\]
Thirdly, Alice randomly choose $u$, such that $0 < u < n$.

$u = 66183$

Fourthly, Alice computes the ciphertext $c = y^m u^r \mod n$.

$c = y^m u^r \mod n = 13213 \cdot 66183^{78} \mod 75827 = 67158$

Fifthly, Alice sends the ciphertext $c$ to Bob.

3.3 Decryption (Bob)

Firstly, Bob gets the value of $c$ from Alice.

$c = 67158$

Secondly, Bob computes $a = c^{\Phi(n)/r} \mod n$.

$a = c^{\Phi(n)/r} \mod n = 67158^{75240}/99 \mod 75827 = 47178$

Thirdly, Bob lets $md = 0$. If $x^{md} \mod n \neq a$, Bob increments $md$ by 1. Bob keeps incrementing $md$ by 1, until $x^{md} \mod n = a$. If $x^{md} \mod n = a$, then $md$ is the original plaintext.

$md = 0$
$x^{md} \mod n = 24640^0 \mod 75827 = 1 \neq a$

$md = 1$
$x^{md} \mod n = 24640^1 \mod 75827 = 24640 \neq a$

$md = 2$
$x^{md} \mod n = 24640^2 \mod 75827 = 58638 \neq a$

$md = 3$
$x^{md} \mod n = 24640^3 \mod 75827 = 32662 \neq a$

$\cdots$

$md = 78$
$x^{md} \mod n = 24640^{78} \mod 75827 = 47178 = a$

Bob finally gets the recovered plaintext, which is $md = 78$. Bob looks up the ASCII table, and number 78 refers to the letter “N”, which is exactly the text Alice wants Bob to read.

4. Conclusion

We have described a tutorial on how to secure a simple text using the Benaloh public key cryptosystem. The Benaloh cryptosystem has been implemented in the Python 2.7.15. To increase the clarity of the computation, one may find that we have used some intermediate variables which are not available in Benaloh’s original algorithm. This tutorial uses 3-digit prime numbers as Benaloh’s private keys. In
practice, we recommend using prime numbers with longer digits, for example 250-digit prime numbers, in order to increase the security.

5. References

[1] Kumar L P and Gupta A K 2016 Implementation of speech encryption and decryption using Advanced Encryption Standard 2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)

[2] Rachmawati D and Budiman M A 2017 An implementation of the H-Rabin algorithm in the Shamir three-pass protocol 2017 2nd International Conference on Automation, Cognitive Science, Optics, Micro Electro–Mechanical System, and Information Technology (ICACOMIT)

[3] Wang Y, Wang J and Chen X 2016 Secure searchable encryption: a survey Journal of Communications and Information Networks 1 52–65

[4] Diffie W and Hellman M 1976 New directions in cryptography IEEE Transactions on Information Theory 22 6 pp 644-654

[5] Hellman M 2002 An overview of public key cryptography IEEE Communications Magazine 40 42–9

[6] Pohlig S and Hellman M 1978 An improved algorithm for computing logarithms over GF(p) and its cryptographic significance (Corresp.) IEEE Transactions on Information Theory 24 106–10

[7] Schneier B 2015 Applied cryptography: protocols, algorithms, and source code in C Edition 20th anniversary edition (Indianapolis, IN: Wiley)

[8] Rachmawati D and Budiman M A 2018 Using the RSA as an asymmetric non-public key encryption algorithm in the Shamir three-pass protocol Journal of Theoretical & Applied Information Technology 96 17

[9] Rivest R L, Shamir A and Adleman L 1978 A method for obtaining digital signatures and public-key cryptosystems,” Communications of the ACM 21 2 pp 120-126

[10] Benaloh J 1994 Dense probabilistic encryption Proceedings of the workshop on selected areas of cryptography pp 120-128

[11] Fousse L, Lafourcade P and Alnuaimi M 2011 Benaloh’s dense probabilistic encryption revisited International Conference on Cryptology in Africa pp 348-362

Acknowledgments

We gratefully acknowledge that this study is funded by Universitas Sumatera Utara. The support is under the research grant Talenta USU of Year 2019 Contract Number: 4167/UN5.1.R/PPM/2019 Date: 1st April 2019.