# Study and Improvement of Zeus Election System

Muhamad Azaiman Bin Zamzuri
School of Computing
*Universiti Teknologi Malaysia*
Johor, Malaysia
muhamadazaiman@graduate.utm.my

***Abstract*— Technology is advancing rapidly; traditional voting methods are now slowly being replaced by E-voting, which provides people a more digital and convenient way to vote [1]. Zeus Election is an example of E-voting system. It is an open-source polling system for election that offers verifiable online election. It is web-based application and currently has served more than 4000 electronic polls and 600000 voters. In this paper, we will study the cryptographic approaches and algorithms that applied to Zeus Election system, thus propose for an improvement to strengthen its security by applying the alternative cryptographic algorithms.**

***Keywords—E-voting, Public-Key Cryptography, Asymmetric-Key Cryptography, Symmetric-Key Cryptography, AES, RSA, SHA-256 Hashing.***

## I. INTRODUCTION

As our living environments has become more digitalized and number of users grown tremendously, privacy is a matter of increasing concern. Nowadays, more elections are conducted online. These elections platform will require the users to create a new account with their identity before they can vote, and end-to-end connection from user's device to system's server should be encrypted and should not exposed to anywhere. Therefore, data protection is needed to against privacy violation from illegal hacking activities in any election platform. In this case study, a security scheme will be proposed for increasing an existing system's security. Every voting system's primary purpose is to ensure that the votes of the voters are counted; therefore, electronic democratic governance that ensures a transparent and trustworthy election is required. The conventional way of voting entails casting a vote on a real paper ballot. This is vulnerable to time-consuming procedures, ballot snatching, a lack of voter privacy, and raises concerns about the fairness of the democratic process. The application that are chosen by us are Zeus Elections. It is an open-source polling system for election. Several different ballot types and election systems are supported, such as simple questions with predefined answers, party lists, score voting, or single transferable vote (STV). Zeus Elections has served more than 4328 electronic polls and 643861 voters. Based on our analysis toward the system, the functionalities are quite complete but there is no guarantee for data security or any privacy protection by the website administrator and provider. Therefore, Zeus Election is a decent system for us to evaluate their security environment, thus make some improvements by proposing the crypto methods that suitable for the system. In this paper, we propose to use Advanced Encryption Standard (AES) as symmetric algorithms and RSA (Rivest Shamir Adleman) and Message Hashing (SHA-256) as information digest to improve the security services of the existing Zeus E-voting system.

## II. PROBLEM

While thousands of individuals have profited from these tools, we have discovered that it is vulnerable to cross-site scripting attacks. Cross-Site Scripting (XSS) is a client-side code injection attack that can be used to persuade a user to make a malicious request. If an attacker can persuade a voter to click on a specially planned link, the voter will be taken to a page that has been manipulated, allowing the attacker to violate ballot confidentiality or influence votes. Voting is the most important feature of any election system because when a user is voting, no one is allowed to view or modify the content of its ballot. In order to secure the voting channel, encryption such as AES, RSA is required for a better security.

## III. RELATED WORKS

There are several solutions that could be adopted to enhance the channel security between ballots and destinations. Examples include Advanced encryption standard [2], Digital Signature Algorithm (DSA) [3], Rivest Shamir Adleman (RSA) [4], Homomorphic encryption [5] and many others encryption structure that providing advance algorithm in securing the ballots. Most of these solutions required the authentication signature on the server side in order to decrypt a ballot's cipher text. Strong security is formed in this case; thus, we will refer to the above work for cryptographic experiments. In addition, message digest is also needed to make a ballot's data even more protected with integrity. This means that verification is required to ensure that the ballot was sent by the same user and not altered by other unauthorized persons. In this situation, we choose SHA-256 [6] because it's not an encryption but a hash algorithm where a user's message can be completely shuffled. It is a one-way cryptographic hash. It is highly secured because the encrypted message by this algorithm can never be decrypted [7].

Based on our proposed solution, we will implement the experiment by using AES, RSA and SHA-256.

## IV. METHODOLOGY

To examine the effectiveness of our proposed solution, we will make an experiment for the voting process starting from the preparation, voting process and information digest by shuffling.

## V. EXPERIMENTAL RESULTS

There are 3 stages to carry out this experiment, from stage 0 to stage 2 where the AES, RSA and SHA-256 are implemented. The Results are presented in the form of a table and any relevant figures in the references.

## VI. STAGE 0: VOTER PREPARATION STAGE

The system administrator adds users to the voting system at this step. The user's name and email address are used by the administrator. The encryption system receives the email as an input message. The system will then encrypt the data and generate a secret message, which the user will receive and use as the system's default login password. AES Encryption is the encryption algorithm used here.

### *A. AES Algorithm*

It is a symmetric encryption, meaning it only takes one key to encrypt and decrypt a message, which is known as a secret key. AES encryption is a symmetric key encryption with a high level of security. We'll use AES-128, which means the key length is 128 bits in this case. After that, we'll encrypt the data in 128-bit blocks for ten rounds.

## B. AES Key

After that, we'll begin with the essential expansion. This will accept a four-word (16-byte) key as input and output a 44-word matrix (176 bytes). This matrix of words, each of which is unique, is enough to give keys for AddRoundKey in all 10 rounds, with each AddRoundKey procedure using a four-word array [10].

## C. AES Encryption

For this example, we will choose the following plaintext and key, as shown in Table 1. To make demonstration easier, we will use the same value as our reference from the AES

*Table 1: Key and Message chosen for example.*

| | |
|---|---|
| ***Plaintext (message)*** | 32 43 f6 a8 88 5a 30 8d 31 31 98 a2 e0 37 07 34 |
| ***Subhead*** | 2b 7e 15 16 28 ae d2 a6 ab f7 15 88 09 cf 4f 3c |

Before we start with the encryption, the message will need to be divided into a block of 4x4 matrix.

| | | | |
|---|---|---|---|
| 32 | 88 | 31 | E0 |
| 43 | 5a | 31 | 37 |
| F6 | 30 | 98 | 07 |
| A8 | 8d | A2 | 34 |

This message will first go through the initial process transformation where it will combine with the first AddRoundKey.

The key will go through Key Expansion process to find the w0-w43.

***Key: 2b 7e 15 16 28 ae d2 a6 ab f7 15 88 09 cf 4f 3c***

From the key, we can get w0-w3:

***w0 = 2b7e1516*** ***w1 = 28aed2a6*** ***w2 = abf71588*** ***w3 = 09cf4f3c***

To get the value of w4-w34, we need to perform Key Expansion. Figure 1 illustrates the process of key expansion.

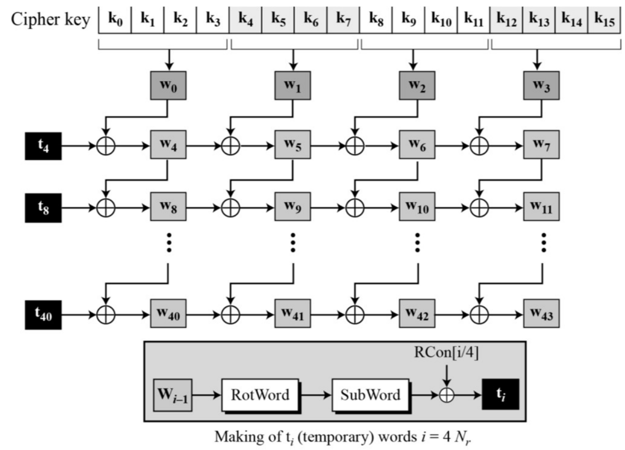


*Figure 1: Process of Key Expansion*

*Table 2: Key expansion for each round.*

| *i* | *temp* | *After Rot Word* | *After Sub Word* | *Rcon [i/4]* | *After XOR Rcon* | *w[i–4]* | *XOR w[i-4]* |
|---|---|---|---|---|---|---|---|
| 4 | 09cf4f3c | cf4f3c09 | 8a84eb01 | 01000000 | 8b84eb01 | 2b7e1516 | a0fafe17 |
| 5 | a0fafe17 | | | | | 28aed2a6 | 88542cb1 |
| 6 | 88542cb1 | | | | | abf71588 | 23a33939 |
| 7 | 23a33939 | | | | | 09cf4f3c | 2a6c7605 |
| 8 | 2a6c7605 | 6c76052a | 50386be5 | 02000000 | 52386be5 | a0fafe17 | f2c295f2 |
| 9 | f2c295f2 | | | | | 88542cb1 | 7a96b943 |
| 10 | 7a96b943 | | | | | 23a33939 | 5935807a |
| 11 | 5935807a | | | | | 2a6c7605 | 7359f67f |
| 12 | 7359f67f | 59f67f73 | cb42d28f | 04000000 | cf42d28f | f2c295f2 | 3d80477d |
| 13 | 3d80477d | | | | | 7a96b943 | 4716fe3e |
| 14 | 4716fe3e | | | | | 5935807a | 1e237e44 |
| 15 | 1e237e44 | | | | | 7359f67f | 6d7a883b |
| 16 | 6d7a883b | 7a883b6d | dac4e23c | 08000000 | d2c4e23c | 3d80477d | ef44a541 |
| 17 | ef44a541 | | | | | 4716fe3e | a8525b7f |
| 18 | a8525b7f | | | | | 1e237e44 | b671253b |
| 19 | b671253b | | | | | 6d7a883b | db0bad00 |
| 20 | db0bad00 | 0bad00db | 2b9563b9 | 10000000 | 3b9563b9 | ef44a541 | d4d1c6f8 |
| 21 | d4d1c6f8 | | | | | a8525b7f | 7c839d87 |
| 22 | 7c839d87 | | | | | b671253b | caf2b8bc |
| 23 | caf2b8bc | | | | | db0bad00 | 11f915bc |
| 24 | 11f915bc | f915bc11 | 99596582 | 20000000 | b9596582 | d4d1c6f8 | 6d88a37a |
| 25 | 6d88a37a | | | | | 7c839d87 | 110b3efd |
| 26 | 110b3efd | | | | | caf2b8bc | dbf98641 |
| 27 | dbf98641 | | | | | 11f915bc | ca0093fd |
| 28 | ca0093fd | 0093fdca | 63dc5474 | 40000000 | 23dc5474 | 6d88a37a | 4e54f70e |
| 29 | 4e54f70e | | | | | 110b3efd | 5f5fc9f3 |
| 30 | 5f5fc9f3 | | | | | dbf98641 | 84a64fb2 |
| 31 | 84a64fb2 | | | | | ca0093fd | 4ea6dc4f |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 32 | 4ea6dc4f | a6dc4f4e | 2486842f | 80000000 | a486842f | 4e54f70e | ead27321 |
| 33 | ead27321 | | | | | 5f5fc9f3 | b58dbad2 |
| 34 | b58dbad2 | | | | | 84a64fb2 | 312bf560 |
| 35 | 312bf560 | | | | | 4ea6dc4f | 7f8d292f |
| 36 | 7f8d292f | 8d292f7f | 5da515d2 | 1b000000 | 46a515d2 | ead27321 | ac7766f3 |
| 37 | ac7766f3 | | | | | b58dbad2 | 19fadc21 |
| 38 | 19fadc21 | | | | | 312bf560 | 28d12941 |
| 39 | 28d12941 | | | | | 7f8d292f | 575c006e |
| 40 | 575c006e | 5c006e57 | 4a639f5b | 36000000 | 7c639f5b | ac7766f3 | d014f9a8 |
| 41 | d014f9a8 | | | | | 19fadc21 | c9ee2589 |
| 42 | c9ee2589 | | | | | 28d12941 | e13f0cc8 |
| 43 | e13f0cc8 | | | | | 575c006e | b6630ca6 |

The key will then be passed to the AddRoundKey method as an input. As a result, an AddRoundKey stage begins and ends the encryption. Without this step, other operations can be reversed to obtain the plaintext, even if the key is unknown.

The encryption starts with the key expansion and then moves on to the AddRoundKey stage, which is followed by nine rounds that each include all four operations. Then there's the tenth round, which just takes three steps. ***Substitute*** bytes, ***ShiftRows***, ***MixColumns***, and ***AddRoundKey*** are the four procedures in each round except the last. The MixColumns process is skipped in the final round.

Figure below show the encryption process.

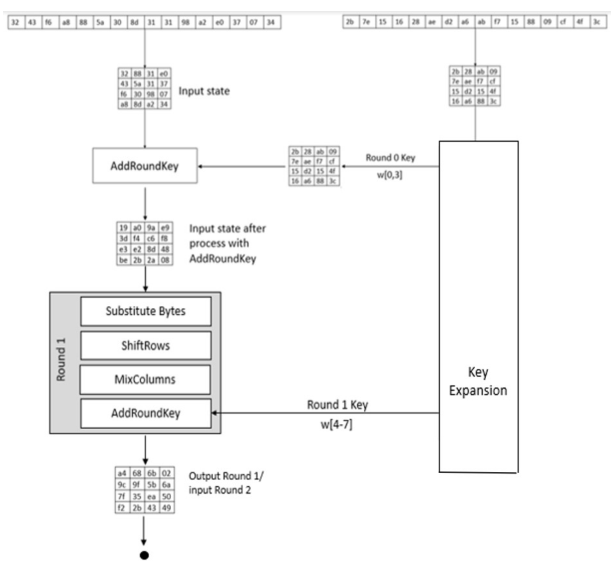


*Figure 2: Encryption process, Round 0-1*

The process (or Round) will continue until round 10, with the exception that Round 10 emits the MixColumns Process. This is done to ensure that the encryption can be reversed or decrypted.

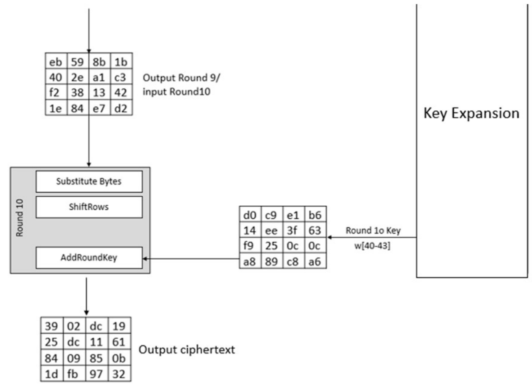


*Figure 3: Encryption process, Round 10*

Once the encryption process is completed, the output ciphertext that is in hexadecimal form will be converted into Base64.

| | |
|---|---|
| ***Output:*** | ***OQLcGSXcEWqECYULHfuXMg==*** |

This output then will be used by the users as the user as a default password to log in to the voting system.

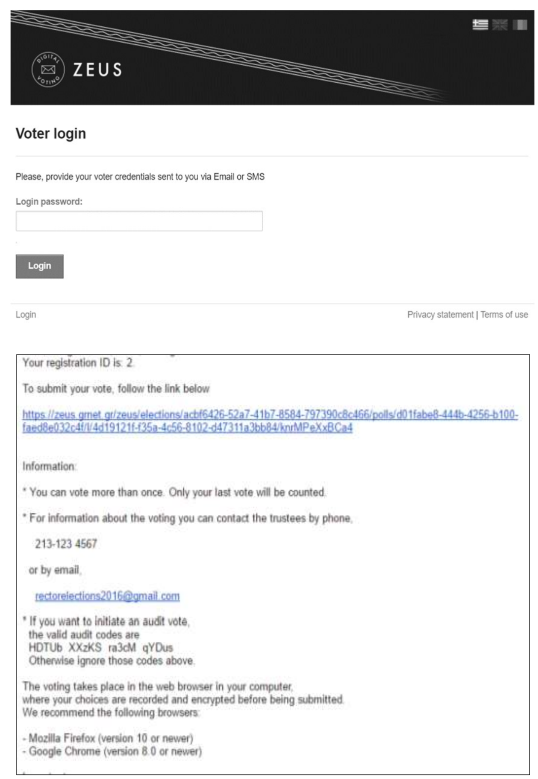


*Figure 4: User email with the generated password.*

The user can now use their email and newly generated email to enter the voting system.

## VII. STAGE 1: VOTING STAGE

This is where the voting part of the system takes place. To gain access to the system, the user must type the voting URL into their computer, which will then submit a secure HTTP request to the server. As a private key, the server will choose r1 at random. It will choose $g^{r1}$ as a public key at the same time. This public key gr1 will be returned to the computer to indicate that the connection has been established, allowing the user to access the voting system and view the website. The process of a user gaining access to the system is depicted in the diagram below.

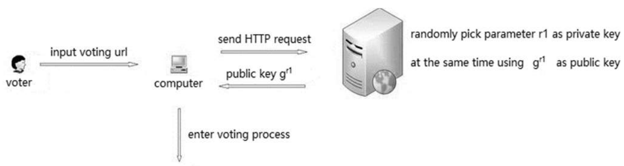


*Figure 5: User accessing the system.*

Once the user enters the website, they will input a vote, m to the computer. This online system will then generate a randomly picked parameter, s which then generates a ciphertext $(g^{r1})g^m$, where m is the vote. The system will also produce a certificate $\{h^s g^{r1}, g^{r1}\}$ for the user to validate or verify that they have made the vote and the vote has been processed by the system.

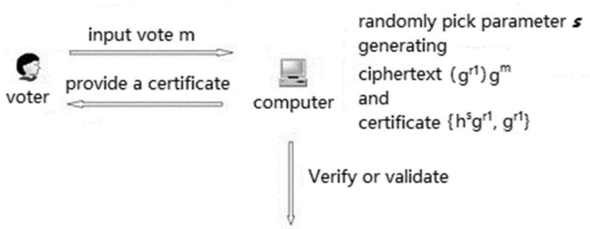


*Figure 6: User input vote*

The user's verification process is the following stage. The user will request to see the parameter s, which is the ciphertext generated in the previous phase in order to verify. If that parameter matches the one that was previously generated, the vote is valid.

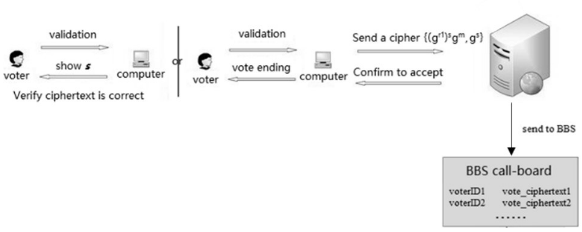


*Figure 7: Vote validation process.*

## A. RSA Algorithm

This section will show the implementation of RSA Encryption in the system. RSA is the most commonly used and recognized public-key encryption. The name RSA is taken from its inventors, Rivest, Shamir, and Adleman. The RSA cryptography is a block cipher where the plaintext and ciphertext are integers between 0 and n - 1 for some n. A typical size for n is 1024 bits, or 309 decimal digits [8].

## B. RSA Key Generation

The key generation process of the RSA algorithm is based on these rules:

1. Select two prime numbers, $p$ and $q$ such that $p$ and $q$ are not equal.
2. $n = p.q$
3. Select $e$, where $1 < e < \varphi(n)$.
   a. $gcd(e, \varphi(n)) = 1$
   b. $e$ and $\varphi(n)$ are coprime.
4. Calculate $d$, where $d = e - 1 \bmod \varphi(n)$
5. Public key = $(e,n)$
6. Private key = $(d)$

## C. RSA Encryption

The value n is the public RSA modulus, and $\varphi(n)=(p-1)(q-1)$ is the Euler totient. The public key e is freely chosen but must be a coprime to $\varphi(n)$. For this example, we will apply the following value. We will perform encryption for Confidentiality where we encrypt using public key and decrypt using private key.

We will start by choosing the value of p and q.

***p= 157 and q=239.***

Then, we perform some calculations to get the value of ***n***, ***e***, and ***d.***

*Table 3: Calculation of RSA global parameters*

| | |
|---|---|
| Calculate ***n*** | ***n = p*q***<br>***n*** = 157*239<br>***n*** = 37523 |
| Calculate ***φ(n)*** | ***φ(n) = (p – 1)(q – 1)***<br>***φ(n)*** = (157 – 1)(239 – 1)<br>***φ(n)*** = (156)(238)<br>***φ(n)*** = 37128 |
| Calculate ***e*** | ***gcd(***37128, ***e ) = 1***<br>***1 < e < φ(n)***<br>***e*** = 7253<br>thus,<br>***gcd(***37128, 7253***) = 1*** |
| Calculate ***d*** | $d = e^{-1} \bmod \varphi(n)$<br>$d = 7253^{-1} \bmod 37128$<br>$d = 7253^{(n)-1} \bmod 37128$<br>$d = 7253^{(37128)-1} \bmod 37128$<br>$d = 7253^{(9216)-1} \bmod 37128$<br>$d = 7253^{9215} \bmod 37128$<br>$d = 31277$ |

Then we get the value:

***p= 157, q=239, n=37523, e= 7253, d= 31277.***

The message we demo as below:

***Message (plaintext) = Vote for Person A***

The Input text will be separated into segments of Size 1 (the symbol '#' is used as separator).

| |
|---|
| V # o # t # e #  # f # o # r #  # P # e # r # s # o # n #  # A |

Numbers input in base 10 format.

| |
|---|
| 086 # 111 # 116 # 101 # 032 # 102 # 111 # 114 # 032 # 080 # 101 # 114 # 115 # 111 # 110 # 032 # 065 |

Encryption into ciphertext c[i] = $m[i]^e$ (mod n)

*Table 4: RSA Encryption*

| $m[i]^e$ (mod n) | ciphertext |
|---|---|
| $086^{7253}$(mod37523) | 06479 |
| $111^{7253}$(mod37523) | 05396 |
| $116^{7253}$(mod37523) | 16351 |
| $101^{7253}$(mod37523) | 13516 |
| $032^{7253}$(mod37523) | 18629 |
| $102^{7253}$(mod37523) | 05044 |
| $111^{7253}$(mod37523) | 05396 |
| $114^{7253}$(mod37523) | 18924 |
| $032^{7253}$(mod37523) | 18629 |
| $080^{7253}$(mod37523) | 27736 |
| $101^{7253}$(mod37523) | 13516 |
| $114^{7253}$(mod37523) | 18924 |
| $115^{7253}$(mod37523) | 06665 |
| $111^{7253}$(mod37523) | 05396 |
| $110^{7253}$(mod37523) | 20734 |
| $032^{7253}$(mod37523) | 18629 |
| $065^{7253}$(mod37523) | 01384 |

Produced ciphertext.

| |
|---|
| 14437 # 28888 # 24126 # 01339 # 06783 # 08880 # 28888 # 10418 # 06783 # 11710 # 01339 # 10418 # 01601 # 28888 # 28412 # 06783 # 26277 |

Decryption into plaintext m[i] = $c[i]^d$ (mod n)

*Table 5: RSA Decryption*

| $c[i]^d$ (mod n) | plaintext |
|---|---|
| $06479^{31277}$ (mod 37523) | 00086 |
| $05396^{31277}$ (mod 37523) | 00111 |
| $16351^{31277}$ (mod 37523) | 00116 |
| $13516^{31277}$ (mod 37523) | 00101 |
| $18629^{31277}$ (mod 37523) | 00032 |
| $05044^{31277}$ (mod 37523) | 00102 |
| $05396^{31277}$ (mod 37523) | 00111 |
| $18924^{31277}$ (mod 37523) | 00114 |
| $18629^{31277}$ (mod 37523) | 00032 |
| $27736^{31277}$ (mod 37523) | 00080 |
| $13516^{31277}$ (mod 37523) | 00101 |
| $18924^{31277}$ (mod 37523) | 00114 |
| $06665^{31277}$ (mod 37523) | 00115 |
| $05396^{31277}$ (mod 37523) | 00111 |
| $20734^{31277}$ (mod 37523) | 00110 |
| $18629^{31277}$ (mod 37523) | 00032 |
| $01384^{31277}$ (mod 37523) | 00065 |

Decrypted ciphertext

| |
|---|
| 00086 # 00111 # 00116 # 00101 # 00032 # 00102 # 00111 # 00114 # 00032 # 00080 # 00101 # 00114 # 00115 # 00111 # 00110 # 00032 # 00065 |

Decrypted ciphertext in text form:

| |
|---|
| **Output: Vote for Person A** |

*D. RSA Message Authentication*

After successfully perform encryption and decryption of ballot's information by using RSA, we can now do the message authentication to enhance the integrity of ballot's data. This step is very important because it shows that the message really comes from the sender and has not been altered. For instance, we can shuffle the message by implementing SHA-256 because the signature of hash must be equal between user and receiver.

For certificate verification, we will only need the public RSA parameters, the modulus ***n,*** and public key ***e.***

In the real environment, the RSA key size is much larger, unlike the example we show in the previous part. The key can be in 512 or 1024 or 2048, or even 4096 bits. This makes the algorithm stronger and hard for attackers to break.

We will show another example to demonstrate the RSA signature generation and validation. We will use 1024 bits key size.

*Table 6: RSA key in 1024 bits size*

| | |
|---|---|
| ***Public key (e)*** | MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCxe<br>0PnyCClvsyYWb+lXZ+XBwDEtny8uAACtnUaRXkQYkmnd<br>Wpgr71xfCayZWEZPnDH2hpMMtN75Q9+ejdHq8gqC9s205Zs<br>NWewM0sRig4O5Fznd+t+WMD6NU0zXrROes0q8ZaZCWqS<br>bounZpIyl/xNCmzMbgaIOUVhpiSldcHojQIDAQAB |
| ***Private key (d)*** | MIICXQIBAAKBgQCxe0PnyCClvsyYWb+lXZ+XBwDEtny8u<br>AACtnUaRXkQYkmndWpgr71xfCayZWEZPnDH2hpMMtN75<br>Q9+ejdHq8gqC9s205ZsNWewM0sRig4O5Fznd+t+<br>WMD6NU0zXrROes0q8ZaZCWqSbounZpIyl/xNCmzMbgaIO<br>UVhpiSldcHojQIDAQAB<br>AoGANOL/bC0VlW5Stz9fPV61tKJwly2t4+qMjkJiiM6U8c3oF<br>s+FQIR91jhhq51MHuKdZuBWH8ixfmTWhiDTIjdNMMNAtb<br>Gsn/OCS1nXiQIH/crMD+AX0G5zdn1/02rAkJd5P4w5g9zg4Jm<br>uygI8jdRfDmoODTpsW8nlq//yC2oJsYECQQDZ3xYw8GvqlQ<br>O2dLpd2mtSuQLGUNIly/VI0ckG4EgRs7lNTyY/e59WRwRblI<br>Wf981M3adrZREHFz85rhjjnvztAkEA0IqowEvRcKNrMP3kN<br>weOgYgRXcB4pvHBhO6nuT1B26iT4B6Zush7e8wlSm4A2jkU<br>sXrcl27gVm7QbwgkkLTGIQJBAJjOY5UNetL7krAMbI3Y3H8<br>Xbb/D/bAuvalGIdVlxoZL5EI5qhzKSrXLd1337ESHG4G20G5<br>9YxTzFBATcDdtUf0CQE159k/a2yjZzc0ZxlubdxowjyMhirGb<br>R6Y3dCCh3YHaE3ZEaCC4swe/RGtuiuqIImP9nU61Zqs16EIX<br>6F3Ki4ECQQCUeKSnK6AcPHj9wOlJ3L6ibo+Mju434B5f8DQ<br>vpJhV5FkQHeneaNQcgMhuFIh5U5H26q16h3FcwZxBhTmhW<br>+/A |

## VIII. STAGE 2: SHUFFLE

From here, we will use the same message as before, that is ***Vote for person A.***

There are several algorithms for RSA signature generation, but the one we will use today is SHA256 with RSA. SHA256 with RSA signature is an efficient asymmetric encryption method used in many secure APIs. The three basic steps in this algorithm:

1. Hashing.
2. Padding the hash for signature generation.
3. Modular exponentiation using the private exponent and the modulus.

Using the previous values stated in table 6 and the message stated, we can get the following value of signature.

elOUMLix3fR/IcyTOYqs81RdPq1Y0z+TtRLFjG4Pk+xJ
EhIwdZnSoAsnsPoajUsCmOS/J3umfzZnn2ieopp8MZxu
4E7WthSEJOGO5HcTdKADMpJ3qMym9+moLkrI13w
kbSJyYEjMTFP8DxYKp8OyEEr8LUyUxbJHAC67D/7
GjEY=

This signature is sent together with the ciphertext, and the receiver can verify the ciphertext's validity by comparing the digital signature's value to the value above. The decrypted ciphertext is not the same as the original plaintext if the ciphertext is signed by another person's key. The value of the signature is likewise different. In the shuffle process, the ciphertext sent by the voter is encrypted numerous times to increase the key required to decrypt the ciphertext, so if the ciphertext is obtained by the middleman, they will not be likely to be able to decrypt the cipher text.

## IX. SECURITY ANALYSIS

This paper implements a combination of two cryptography algorithms, that is AES and RSA. Table below shown the example of data inputs of multiple value bits and random private keys.

*Table 7: Recommended Security Bit Level*

| ***Security Bit Level*** | ***Key size*** | |
|---|---|---|
| | ***AES*** | ***RSA*** |
| 80 | - | 1024 |
| 112 | - | 2048 |
| 128 | 128 | 3072 |
| 192 | 192 | 7680 |
| 256 | 256 | 15360 |

Based on our findings and experiments, RSA can be concluded as a very effective encryption algorithm suitable for this study because of the straightforward implementation

## X. CONCLUSION AND FUTURE WORK

In this paper we use AES, RSA and SHA-256 hash algorithm to make sure it gives great security. We show the work by using the AES as the key generator and make it unique to do the secret message to the voter. Given the RSA is the process between voters to the system. In addition, using SHA-256 to validate the ballot's data.

As afore mentioned , the summary of the proposal is that we know that this application requires improvement in their security to make sure the voting system attain integrity. What can be implemented in the system is our own original idea after referring to some security implementation done previously on e-voting system and are referred to make sure the proposal can be implement properly. We hope by using this suggestion will strengthen their security. Zeus election system will need this proposal to ensure the system security strength in order to gain the trust from their client.

For future work, I believe that there is a lot more encryptions can be considered in securing the channel for ballot to be sent securely to the server. One encryption that should be useful in this situation is homomorphic encryption [5]. A homomorphic encryption [9] strategy permits client to work on ciphertext. At the point when client decodes the resultant figure, it is same as though tasks are done on plaintext. In this manner, utilizing homomorphic encryption guarantees clients that their information is secure in all state: stockpiling, transmission and preparing. Therefore, any improvements to the Zeus election system are bound to take the system's security to the next level.

## ACKNOWLEDGMENT

First all, by the grace of God we can complete the report for our mini project.

We are so grateful to our dear lecturer, DR. RASHIDAH BINTI KADIR for teaching us SECR3443 Introduction to Cryptography for this semester. We have learned a lot from the lectures and now we feel like we are ready to implement what we learn in the real working environment.

We would also like to again thank our lecturer for also giving us the chance to do this mini project, which requires us to work together in a team and implement what we've learn the whole semester into a working idea.

This completion of project would also not be possible without the hard work of our team members

## REFERENCES

[1] Gupta, L. K., et al. (2019). "AES Based Online Voting System." International Journal of Computer Sciences and Engineering 7(3): 915-918.

[2] Fauziah, N. A., et al. (2018). Design and Implementation of AES and SHA-256 Cryptography for Securing Multimedia File over Android Chat Application. 2018 International Seminar on Research of Information Technology and Intelligent Systems (ISRITI): 146-151

[3] Ashok, P., et al. (2016). "A Survey on Crypt- Algorithms in Voting System." Indian Journal of Science and Technology 9(38).

[4] Wahab, I. H. A., et al. (2021). "Cryptography Implementation for electronic voting security." E3S Web of Conferences 328.

[5] Patil, M. D., et al. (2020). "Online Voting System using Homomorphic Encryption." ITM Web of Conferences 32.

[6] Chowdhury, S., et al. (2020). "SHA-256 in Parallel Blockchain Technology: Storing Land Related Documents." International Journal of Computer Applications 175(35): 33-38.

[7] war.org.uk, 2020. [Online]. Available: http://www.iwar.org.uk/comsec/resources/cipher/sha256-384-512.pdf. [Accessed: 31- Jan- 2022].

[8] Stallings, W. (2011). THE RSA ALGORITHM. In Cryptography and network security: Principles and practice, fifth edition. Upper Saddle River: Prentice Hall

[9] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(leveled) fully homomorphic encryption without bootstrapping,". in Proceedings of the 3rd Innovations in Theoretical Computer Science Conference. ACM, 2012, pp. 309–325.

[10] Dworkin, M.J., Barker, E.B., Nechvatal, J.R., Foti, J., Bassham, L.E., Roback, E., Dray, Jr., J.F.: Advanced Encryption Standard (AES). FIPS PUB 197, November 2001. Retrieved February 04, 2021, from https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf