UNIVERSITY OF LJUBLJANA
FACULTY OF COMPUTER AND INFORMATION SCIENCE

E-VOTING

AUTHOR
KATARINA MILAČIĆ

Cryptography and Computer Security
Prof. dr. Aleksandar Jurišić
Ljubljana, January 2018

Contents

1 Introduction

Along with the development of technology, electronic voting has been intensively studied. Up to now, many electronic voting schemes have been proposed, and both the security and the effectiveness have been improved. E-voting offers many advantages regarding technology, speed, privacy and security. In the following chapters I will present cryptography that lies behind e-voting, schemes that have been tested and finally conclude project with some security overview. Special attention is given to schemes that use homomorphic properties of encryption.

2 Properties of a voting system

Every voting system consists of following stages:

1) **Registration** - In the registration stage the authorities determine who is eligible to vote, maintain proper lists of the registered voters;
2) **Validation** - when the election begins, administrators validate the credentials of those attempting to vote;
3) **Collection** - At this stage the voted ballots are collected before the final stage of the tally;
4) **Tallying** - At this stage the accumulated votes are counted, agreed upon and published.

Every voting system needs to fulfil the following requirements:

Availability: A voting system must remain available during the whole election and must serve voters connecting from untrusted clients.

Eligibility: Only elective voters are allowed to cast one valid vote. Therefore no double votes are allowed.

Integrity: The integrity of the vote must be guaranteed.

Privacy: The connection between the vote of a user and the user herself must not be able without her help.

Fairness: Ensures that no (partial) results are published before the tallying has ended.

Receipt Freeness: To reduce coercion, the user does not gain any information about her vote. Therefore she cannot prove her vote to anybody.

Correctness: Election results must be counted properly and published correctly.

Robustness: The system should be able to tolerate faulty votes.

Universal Verifiability: After the tallying process, the results are published and can be verified by everybody.

Voter Verifiable: The voter herself must be able to verify that her vote was counted properly.

Coercion: Voting Systems must provide security aspects to prevent a coercer being able to force the voter to place a vote for a specific party, candidate etc. In theory a voting system must be built coercion-resistant to guarantee that a voter can place her vote as intended even in the presence of a coercer. This implies that even sold or leaked credentials cannot be used to place ballots.

3 Underlying cryptography

This chapter represents a brief introduction into cryptography behind e-voting systems, and the goal is to give reader a better understanding of schemes that are presented in chapter 4. Following topics will surely be familiar to a person who has some knowledge in cryptography.

3.1 Digital Signature

Digital signature (1) is used to authenticate that the message comes from a particular sender. This is done by attaching a code that acts as a signature. This signature guarantees the source and the integrity of the message.

E-Voting employs RSA encryption. Public and private key are utilized to perform encryption and decryption.

Key generation:
- choose two prime numbers p, q, $n = pq, \varphi(n) = (p-1)(q-1)$
- choose public exponent e which is less than and relatively prime to $\varphi(n)$
- $d = e^{-1} \bmod \varphi(n)$
- The public key is $\{n, e\}$ and the private key is $\{n, d\}$.

The equation for encryption and decryption are as follows:
Encryption:$C = M^e \bmod n$
Decryption: $M = C^d \bmod n$

In e-voting, digital signature is created by using RSA encryption. The process begins with the hashing of the message, M, to produce a message digest, H. The digest is then encrypted using the sender's private key $\{n, d\}$ to produce the signature

$$S: S = H^d \bmod n$$

To verify the message, the receiver will hash the message, M by using the same digest function. At the same time, the signature, S is decrypted using the receiver's public key:

$$H = S^e \bmod n$$

The results of the two processes are then compared. If they are equal then the message is authenticated and the integrity of the message is maintained.

3.2 Blind Signature

A blind signature (1) is a type of digital signature that allows a person to get another person to sign a message without revealing the content of a message. In E-Voting the signature is used to authenticate the voter without disclosing the content of a ballot. Hence the authority whose function is to verify the eligibility of a voter will not know whom a voter votes for.

A voter is required to get the signature of a validator when he votes. To ensure the secrecy of his ballot, a voter casts a ballot, *B*, blinds a ballot using a random number and sends it to the validator.

Let (n, e) be the validator's public key and (n, d) be his private key. A voter generates a random number r such that $gcd(r, n) = 1$ and sends the following to the validator:

$$B' = r^e B \bmod n$$

The random number r conceals the ballot from the validator. The validator then signs the blinded ballot after verifying the voter. The signed value is as follows:

$$S' = (B')^d = rB^d \bmod n$$

After receiving the validated ballot, the voter unblinds the ballot, to get the true signature, *S* of the validator for the ballot, by computing,

$$S = S'r^{-1} \bmod n = B^d$$

3.3 Mix-nets

A mix-net (2) is a multi-party protocol which is used in e-voting or other applications which require anonymity. It allows a group of senders to input a number of encrypted messages to the mix-net, which then outputs the messages in random order. It is common to construct mix-nets from shuffles.

A shuffle of ciphertexts $C_1, \dots, C_N$ is a set of ciphertexts $C_1', \dots, C_N'$ with the same plaintexts in permuted order. Shuffle protocols constructed from homomorphic encryption schemes allow us to for a given public key pk, messages M_1, M_2, and randomness ρ_1, ρ_2 the encryption function satisfies $E_{pk}(M_1 M_2,\ \rho_1 + \rho_2) = E_{pk}(M_1, \rho_1) E_{pk}(M_2, \rho_2)$.

A common construction of mix-nets is to let the mix-servers take turns in shuffling the ciphertexts. If the encryption scheme is semantically secure the shuffle $C_1', \dots, C_N'$ output by a mix-server does not reveal the permutation or the messages.

Potential problem here is that a malicious mix-server in the mix-net could substitute some of the ciphertexts without being detected. In a voting protocol, it could, for instance, replace all ciphertexts with encrypted votes for candidate X. Therefore, it is necessary to construct an interactive zero-knowledge argument that makes it possible to verify that the shuffle was done correctly (soundness), but reveals nothing about the permutation or the randomizers used (zero-knowledge).

3.4 Zero-knowledge proofs

In certain situations it is needed to prove some statement to someone without revealing any extra information, and zero-knowledge proofs allows us that. Zero-knowledge proofs (3) are stated as protocols for prover and verifier such, that when executed successfully, verifier becomes convinced that the statement holds. Still, verifier cannot extract any useful information besides the correctness of the statement.

3.4.1 Zero-knowledge proof of decryption

Example: A decryption of an ElGamal ciphertext can be proven using Chaum-Pedersen protocol (CP92) for proving plaintext equality (3). To prove that a given ciphertext $c = (x, y)$ encrypts a plaintext m, the prover shows that $\log_g y = \log_x {}^{y}/_{m}$:

Prover:
(1) selects $w, a \in Z_q$, $y = g^a$, $\left(\frac{y}{m}\right) = x^a$
(2) sends $(A, B) = (g^w, x^w)$ to the verifier.

Prover: answers with: $t = w + ac$
Verifier: checks if:
$g^t = Ay^c$
$x^t = B\left({}^{y}/_{m}\right)^c$

3.5 Shamir's Secret Sharing Scheme

Secret sharing protocol (4) is used in schemes presented in chapter 4, and for better understanding here is presented brief overview.

Secret sharing protocol allows the shares of the secret to be distributed to n participants and any t of them can collaborate to retrieve the secret. The secret sharing protocol consists of following phases:

1) Initialization: In this phase, the dealer who wants to distribute the secret K chooses different x_i, $1 \leq i \leq n$ that correspond to each participant. The x_i's are then published.

2) Secret Sharing: The secret K is distributed as shares to the participants securely in this phase. The dealer chooses a_i, $1 \leq i \leq t-1$ and constructs a polynomial $q(x)$ of degree $t-1$ such that the constant term $q(0)$ represents the secret.

$$q(x) = K + \sum_{i=1}^{t-1} a_i x^i$$

The shares that correspond to each participant are constructed by evaluating the polynomial at corresponding x_i values.

$$y_i = q(x_i), 1 \leq i \leq n$$

The shares y_i are then distributed to participants securely.

3) Secret Reconstruction: When t or more participants collaborate together, they can retrieve the secret K by combining the shares. Let $y_i = q(x_i), 1 \leq i \leq t$. Then by using Lagrange Interpolation, the polynomial of degree $t-1$ can be reconstructed from these t points using the formula

$$q(x) = \sum_{i=1}^{t} y_i \prod_{j=1, j\neq i}^{t} \frac{x_j - x}{x_j - x_i}$$

There is exactly one such polynomial of degree $\leq t-1$. The participants can obtain the secret K as

$$K = q(0) = \sum_{i=1}^{t} y_i \prod_{j=1, j\neq i}^{t} \frac{x_j}{x_j - x_i}$$

It is noted that less than t share holders get no information about the secret.

3.5.1 Secret Sharing Homomorphism

Secret sharing homomorphism (4) was introduced by Benaloh in 1987. It is noted that Shamir's scheme is additive homomorphic. He stated that any t of the n agents can determine the super secret and no conspiracy of fewer than t agents can gain any information at all about any of the sub secrets.

Shamir's secret sharing scheme has the $(+,+)$ homomorphism property. For example, assume there are two secrets: K_1, K_2 and are shared using polynomials $g(X)$ and $f(X)$. If we add the shares $h(i) = g(i) + f(i)$, $1 \leq i \leq n$, then each of these $h(i)$ can be treated as the share corresponding to the secret $K_1 + K_2$. The polynomial $h(X) = g(X) + f(X)$ gives us $h(0) = K_1 + K_2$.

Two operations are defined. One on the shares $\oplus$, and the other operation $\otimes$ on the encrypted shares such that for all participants

$$E_i(s_i) \otimes E_i(s_i') = E_i(s_i \oplus s_i')$$

If the underlying secret sharing scheme is homomorphic then by decrypting the combined encrypted shares, the recovered secret will be equal to $s_i \oplus s_i'$.

3.6 Homomorphic encryption

Schemes presented in chapter 4 use homomorphic encryption, so some attention is given to encryption schemes that have this property.

We say that an encryption scheme Enc() is homomorphic (3) if the following equality holds:

$$Enc_k(m_1) \cdot Enc_k(m_2) = Enc_k(m_1 + m_2)$$

If we interpret m_1 and m_2 as numbers, then homomorphic encryption allows for computing the ciphertext of $m_1 + m_2$ from the ciphertexts of m_1 and m_2 (but without knowledge of m_1 and m_2 themselves).

This property is commonly used in e-voting, since it allows us to work and calculate sum of encrypted votes without knowing what the plaintext of encrypted vote is.

3.6.1 ElGamal Encryption Scheme

The ElGamal encryption scheme (5) is a public-key encryption algorithm based on the Diffie–Hellman key exchange. The ElGamal encryption scheme can be defined over any cyclic group G. Its security depends upon the difficulty of a certain problem in G related to computing discrete logarithms. The ElGamal encryption scheme consists of three components: the key generation, the encryption algorithm, and the decryption algorithm.

Key Generation: The key generator works as follows:

Alice generates an efficient description of a cyclic group G, of order q, with generator g. Alice chooses a random $x \in \{1, \dots, q-1\}$

Alice computes

$$y = g^x$$

Alice publishes y along with the description of $G, q\ g$, as her public key. Alice retains x, as her private key which must be kept secret.

Encryption: The encryption algorithm works as follows:

To encrypt a message m, to Alice under her public key (G, q, g, y), Bob chooses a random $r \in \{1, \dots, q-1\}$, then computes

$$c_1 = g^r$$

Bob computes the shared secret

$$s = y^r$$

Bob converts his secret message m, into an element $m' \in G$. Bob computes

$$c_2 = m' \cdot s$$

Bob sends the ciphertext $(c_1, c_2) = (g^r, m' \cdot y^r)$ to Alice. Note that one can easily find y^r, if one knows m'. Therefore, a new r, is generated for every message to improve security. For this reason, r, is also called an ephemeral key.

Decryption: The decryption algorithm works as follows:

To decrypt a ciphertext (c_1, c_2), with her private key x, Alice computes the shared secret

$$t = c_1^x$$

and then computes

$$m' = c_2 \cdot t^{-1}$$

which she then converts back into the plaintext message m, where t^{-1} is the inverse of t in the group G (e.g., modular multiplicative inverse if G is a subgroup of a multiplicative group of integers modulo n).

The decryption algorithm produces the intended message, since

$$c_2 \cdot t^{-1} = (m' \cdot s) \cdot c_1^{-x} = m' \cdot y^r \cdot g^{-xr} = m' \cdot g^{xr} \cdot g^{-xr} = m'$$

Homomorphic Property: ElGamal encryption scheme has a homomorphic property. Given two encryptions

$$(c_{11}, c_{12}) = (g^{r_1}, m_1 y^{r_1}), (c_{21}, c_{22}) = (g^{r_2}, m_2 y^{r_2}),$$

where (r_1, r_2) are randomly chosen from $\{1,2, \dots, q-1\}$ and $m_1, m_2 \in G$, one can compute

$$(c_{11}, c_{12})(c_{21}, c_{22}) = (c_{11}c_{21}, c_{12}c_{22}) = (g^{r_1} g^{r_2}, (m_1 y^{r_1})(m_2 y^{r_2})) \\ = (g^{r_1 + r_2}, (m_1 m_2) y^{r_1 + r_2})$$

The resulted ciphertext is an encryption of $m_1 m_2$.

ElGamal Security: The security of the ElGamal scheme depends on the properties of the underlying group G. If the computational Diffie–Hellman assumption (CDH) holds in the underlying cyclic group G, then the ElGamal encryption function is one way. The CDH is the assumption that a certain computational problem within a cyclic group G is hard. Consider a cyclic group G of order q, the CDH assumption states that, given (g, g^a, g^b) for a randomly chosen generator g and random $a, b \in \{0 \dots, q-1\}$, it is computationally intractable to compute the value g^{ab}. If the decisional Diffie–Hellman assumption (DDH) holds in G, then ElGamal achieves semantic security.

Other schemes related to ElGamal which achieve security against chosen ciphertext attacks have also been proposed. The ElGamal encryption scheme is usually used in a hybrid cryptosystem, i.e., the message itself is encrypted using a symmetric cryptosystem and ElGamal is then used to encrypt the key used for the symmetric cryptosystem. This is because asymmetric cryptosystems like ElGamal are usually slower than symmetric ones for the same level of security, so it is faster to encrypt the symmetric key (which most of the time is quite small if compared to the size of the message) with ElGamal and the message (which can be arbitrarily large) with a symmetric cryptosystem.

3.6.2 Paillier Encryption Scheme

The Paillier encryption scheme (5), named after and invented by Pascal Paillier in 1999, is a probabilistic public-key algorithm.

The Paillier encryption scheme is composed of key generation, encryption, and decryption algorithms as follows:

Key Generation: Choose two large prime numbers p and q randomly and independently of each other, such that

$$\gcd\big(pq, (p-1)(q-1)\big) = 1$$

This property is assured if both primes are of equal length.

Compute

$$n = pq, \lambda = \text{lcm}(p-1, q-1)$$

where lcm stands for the least common multiple.

Select random integer g where $g \in Z^*_{n^2}$.

Ensure n divides the order of g by checking the existence of the following modular multiplicative inverse:

$$\mu = \Big(L\big(g^\lambda \bmod n^2\big)\Big)^{-1} \bmod n$$

where function L is defined as

$$L(u) = \frac{u-1}{n}$$

Finally, the public (encryption) key is (n, g) and the private (decryption) key is (λ, μ).

If using p, q of equivalent length, a simpler variant of the above key generation steps would be to set

$$g = n+1, \lambda = \varphi(n), \mu = \varphi(n)^{-1} \bmod n$$

where $\varphi(n) = (p-1)(q-1)$.

Encryption: Let m be a message to be encrypted where $m \in Z_n$. Select random r where $r \in Z^*_n$. Compute ciphertext as

$$c = g^m r^n \bmod n^2$$

Decryption: Let c be the ciphertext to decrypt, where $c \in Z^*_{n^2}$

Compute the plaintext message as:

$$m = L\left(c^{\lambda} \bmod n^2\right) \cdot \mu \bmod n$$

Homomorphic Properties: A notable feature of the Paillier scheme is its homomorphic properties. Given two ciphertexts $E(m_1, pk) = g^{m_1} r_1^n \bmod n^2$ and $E(m_2, pk) = g^{m_2} r_2^n \bmod n^2$, where $r_1$and r_2 are randomly chosen from Z, we have:

1) Homomorphic Addition of Plaintexts

The product of two ciphertexts will decrypt to the sum of their corresponding plaintexts, i.e.,

$$D(E(m_1, pk) \cdot E(m_2, pk) \bmod n^2) = m_1 + m_2 \bmod n$$

because

$E(m_1, pk) \cdot E(m_2, pk) = (g^{m_1} r_1^n)(g^{m_2} r_2^n) \bmod n^2 = g^{m_1+m_2}(r_1 r_2)^n \bmod n^2 = E(-m_1 + m_2, pk)$

The product of a ciphertext with a plaintext raising g will decrypt to the sum of the corresponding plaintexts, i.e.,

$$D(E(m_1, pk) \cdot g^{m_2} \bmod n^2) = m_1 + m_2 \bmod n$$

because

$$E(m_1, pk) \cdot g^{m_2} = (g^{m_1} r_1^n)^{m_2} \bmod n^2 = g^{m_1+m_2} r_1^{nm_2} \bmod n^2 = E(m_1 + m_2, pk)$$

2) Homomorphic Multiplication of Plaintexts

An encrypted plaintext raised to the power of another plaintext will decrypt to the product of the two plaintexts, i.e.,

$$D(E(m_1, pk)^{m_2} \bmod n^2) = m_1 m_2 \bmod n$$

Because

$$E\left(m_{1,}, pk\right)^{m_2} = (g^{m_1} r_1^n)^{m_2} \bmod n^2 = g^{m_1 m_2}\left(r_1^{m_2}\right)^n \bmod n^2 = E(m_1 m_2, pk)$$

More generally, an encrypted plaintext raised to a constant k will decrypt to the product of the plaintext and the constant, i.e.,

$$D(E(m_1, pk)^k \bmod n^2) = km_1 \bmod n$$

However, given the Paillier encryptions of two messages, there is no known way to compute an encryption of the product of these messages without knowing the private key.

Paillier Security: The Paillier encryption scheme provides semantic security against chosen-plaintext attacks. The semantic security of the Paillier encryption scheme was proved under the decisional composite residuosity (DCR) assumption—the DCR problem is intractable. The DCR problem states as follows: Given a composite N and an integer z, it is hard to decide whether z is a N-residue modulo N^2 or not, i.e., whether there exists y such that $z = y^n (mod\ n^2)$.

Paillier and Pointcheval however went on to propose an improved cryptosystem that incorporates the combined hashing of message m with random r. The hashing prevents an attacker, given only c, from being able to change m in a meaningful way.

3.6.2.1 Application to E-Voting

Consider a simple voting scheme in which an item up for debate can either be supported or opposed. Each voter casts his/her vote using the Paillier Cryptosystem, such that a vote in favour of the proposed item is the plaintext message 1, and a vote against the item is the message 0. Each voter chooses a random r to encrypt his/her vote with, but they all use the same officially designated public key (n, g). A tally, x, is kept of how many votes are cast, then all the encrypted votes are multiplied together, and the result is decrypted as some value, y. Since multiplying encrypted text results in the addition of plaintexts, this process will result in the addition of 1's and 0's as y, effectively tallying all of the votes for the proposition. Since the number of votes cast, x, is known, this leaves $x - y$ votes against the proposition, and the vote can be properly decided.

4 Schemes Based on Homomorphic Encryption

The main idea of this approach is that ballot is viewed as a number. The ballot is shared and encrypted (using either secret sharing or threshold cryptosystem) between M authorities. Cooperation of at least t out of M authorities is needed to reconstruct and decrypt the ballot. Now, using homomorphism properties, ballots are summed up and then the sum is decrypted and reconstructed. We will see how it is implemented in different schemes.

4.1 Cramer-Gennaro-Schoenmakers scheme

Here is described election scheme proposed by Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers (6).

The scheme consists of M authorities $A_1, \dots, A_M$ and a bulletin board. According to threshold cryptosystem, authorities share a common El Gamal public key, $B = (p, g, \gamma)$, where $\gamma = g^a$. Each authority owns his secret share s_i of private key a according to Shamir threshold secret sharing scheme. Threshold trust with threshold t on authorities is used. Also, each voter has his own public key pair.

In this description of scheme voter V can select between two different options, so that the ballot v can be one of numbers $v \in \{1, -1\}$. We get the result of elections by summing these numbers. If the sum is positive, then one option was selected more than the other, if it is negative — vice versa. If it is zero, both options have been selected the same number of times. As N is known, it is possible to compute exactly how many times each option was selected.

Firstly, voter computes g^v and encrypts it $(y_1, y_2) = (g^r, \gamma^r g^v)$. Then he constructs zero-knowledge proof of knowledge that shows that ballot was constructed correctly. It is enough to prove, that at least one of equations holds:

1) $\log_g y_1 = \log_\gamma {y_2}/{g^1}$ 2) $\log_g y_1 = \log_\gamma {y_2}/{g^{-1}}$

After that, encrypted ballot (y_1, y_2) and non-interactive proof of correctness are sent to the bulletin board. The message sent is signed using voter's private key. After the elections are over, submitted ballots are verified by authorities. Incorrect ballots are removed. Signatures of ballots are verified to ensure, that only eligible actors can vote. In order to ensure uniqueness, at most one ballot of each voter must be counted. Now homomorphism property of ElGamal encryption is used on encrypted ballots

$$(g^{r_1}, \gamma^{r_1} g^{v_1})$$
$$\dots$$
$$(g^{r_N}, \gamma^{r_N} g^{v_N})$$

in order to get

$$\left(g^{\sum r_i}, \gamma^{\sum r_i} g^S\right)$$

where $S = \sum v_i$ is the result of elections. Thereafter, g^S is decrypted according to the threshold cryptosystem by any t authorities.

Finally, it is needed to extract S from g^S. Generally, computing discrete logarithm is considered infeasible, but in our case $S \in [-N \dots N]$. So, it is enough to (pre)compute $g^{-N}, g^{-N+1}, \dots, g^1, g^0, g^1, \dots, g^{N-1}, g^N$ and compare these values with g^S.

This scheme satisfies eligibility, uniqueness, robustness and availability. Correctness and verifiability are also satisfied because both voters and authorities must prove correctness of their actions (authorities prove correctness of their actions implicitly when performing verifiabled encryption in the setting of threshold cryptosystem). Privacy is ensured as long as at most $t - 1$ authorities misbehave according to the threshold trust on them. Incoercibility is not satisfied, because voter can decrypt his ballot and present certificate for encryption (randomness used) to the coercer.

4.2 Schoenmakers scheme

In this scheme (7), the voter shares his vote among the authorities using secret sharing scheme. Computing the final tally exploits the homomorphic property of the secret sharing (the tally – sum of the secrets is reconstructed from the multiplied shares).

Initialization stage. Initialization of the publicly verifiable secret sharing scheme is performed (generators g, G of Z_p, public keys $h_j = g^{z_j}$ of the authorities are published).

Voting stage. Voter V_i chooses his vote $v_i \in \{0,1\}$ and a random $s_i \in Z_p$. He uses Sharim's secret sharing to share a secret g^{s_i} and publishes the value $U_i = g^{s_i + v_i}$. In addition, to show that indeed $v_i \in \{0,1\}$ he gives a proof that

$$\log_G C_0 = \log_g U_i \vee \log_G (GC_0) = \log_g U_i$$

($C_0 = G^{s_i}$ is published as a part of the secret sharing). Anyone can check the ballot in the bulletin board due to the public verifiability of the secret sharing scheme and the given proof of $v_i \in \{0,1\}$.

Counting stage. Suppose that the voter V_i, $i = 1, \dots, m$ succeeds in casting valid ballots. Firstly, all the respective encrypted shares $H_{ij} = h_j^{p_i(j)}, j = 1, \dots, t-1$ are accumulated

$$H_j^* = \prod_i H_{ij} = \prod_i h_j^{p_i(j)} = h_j^{\sum_i p_i(j)}$$

Now authorities can obtain $g^{\sum_i p_i(0)} = g^{\sum_i s_i}$, due to the homomorphic property. Combining with $\prod_i U_i = g^{\sum_i s_i + v_i}$ we gain $g^{\sum_i v_i} = g^T$. The final tally T can be now computed as in the CGS scheme (see section 4.1).

This scheme satisfies the following: Only eligible voters can write into the bulletin board. The voter cannot cast invalid or double vote, since he is required to give a proof of the validity of the vote. Privacy is protected by the security of the publicly verifiable secret sharing scheme. Anyone can verify the validity of the voter's vote. The publicly verifiable secret sharing scheme prevents the voter from distributing invalid shares to the authorities. Anyone can multiply the encrypted shares of the jth authority, and anyone can verify whether the jth authority had decrypted its sum of the shares correctly. Anyone can compute the final tally from the published sums of the shares. The scheme is not receipt-free, since the voter's receipt is the secret s_i.

4.3 Hamill and Snyder Scheme

A hybrid scheme was proposed by Hamill and Snyder (7). The possibility of coercing in the CGS scheme (section 4.1) is caused by the randomness k used in the encryption of the vote. The problem is that when the voter himself encrypts his vote, he knows what and how is encrypted, and he can be coerced to reveal it. In the HS scheme, possible votes are encrypted and permuted by the authorities, one after another. A permutation of the encrypted votes is sent through the untappable channel to the voter. Voter just points at the vote he chooses. The scheme is designed in such a way that only the voter gets to know the final permutation and he can lie about it to anyone else.

Votes will be encrypted as in the CGS scheme (section 4.1), using El Gamal encryption. Recall that ith choice encrypts to the $(x, y) = (g^k \bmod p,\ h^k G_i \bmod p)$, where (p, g, h) is the public El Gamal key and k is a random number.

Initialization stage. N authorities set up robust threshold El Gamal cryptosystem. The generators $G_1, \dots, G_L$ representing the possible choices are published. Authorities also create a public list of all standard-encrypted valid votes $e_1^{(0)}, \dots, e_L^{(0)}$, where $e_i^{(0)}$ is the encryption of the ith choice G_i using the randomness $k = 0$: $e_i^{(0)} = (1, G_i)$.

Voting stage. For each voter V, the authorities generate a list of the encryptions of all possible votes, from which the voter V selects the one representing his intention. In turn, for each authority A_j (where $j = 1, 2, \dots, N$): the A_j takes on input the list $e_1^{(j-1)}, \dots, e_L^{(j-1)}$, re-encrypts each element from the input list, and permutes the list in a random order. This way, A_j produces the output list $e_1^{(j)}, \dots, e_L^{(j)}$. Moreover, A_j is required to prove that the output list has been properly constructed. If the A_j fails in some way or the voter objects to A_j, then the A_j is ignored and it is put $e^{(j)} = e^{(j-1)}$. The first authority picks the standard list $e_1^{(0)}, \dots, e_L^{(0)}$ as the input.

Below is the more detailed description of the protocol:

1. A_j computes the output list $e_1^{(j)}, \dots, e_L^{(j)}$ as follows:

(a) A_j selects random permutation $\pi_j : \{1 \dots L\} \to \{1 \dots L\}$ and the random numbers $k_1, \dots, k_L \in_R Z_p$
(b) The $\pi_j(i)$th item in the final list is obtained by re-encrypting the ith item $e_i^{(j-1)}$ from the input list with the randomness k_i.

2. Without revealing the permutation π_j as well as the randomness $k_1, \dots, k_L$, the A_j shows that the output list $e_1^{(j)}, \dots, e_L^{(j)}$ is correctly constructed: For each $i = 1 \dots L$ the A_j proves

that there exists a re-encryption of the ith item $e_i^{(j-1)}$ of the input list in the output list $e_1^{(j)}, \dots, e_L^{(j)}$.

3. A_j secretly transfers the permutation π_j together with the private proof of its correctness through the untappable channel to the voter V. More precisely, A_j proves that for each i, $e_{\pi_j(i)}^{(j)}$ is the re-encryption of $e_i^{(j-1)}$.

4. If the voter does not accept the proof, he publicly complains about the authority. After that, the list is rolled back to the previous state $e_1^{(j-1)}, \dots, e_L^{(j-1)}$ and the jth authority is ignored.

The voter publicly announces the position i of his desired vote in the final list $e^{(N)}$.

Counting stage. Each voter has already chosen his vote from the final list produced for him by the authorities. His vote is encrypted, and then the result of the election can be computed as following – the encrypted votes are multiplied to obtain the encrypted sum of the votes, the authorities jointly decrypt the sum of the votes and publish the proof of correct decryption.

Achieved Properties

If the El-Gamal cryptosystem is secure, then this scheme provides eligibility, universal verifiability, computational privacy, robustness and receipt freeness.

Eligibility. Eligibility is achieved, as the voter cannot cast invalid vote, he votes at most once and he votes how he wishes (he can trace the permutation of the encrypted votes). Only one problem arises, when one of the authorities coerces. It can force the voter not to complain against its wrong proof of the permutation. Therefore, the voter will lose his track and he will vote randomly. For the coercer, random vote can be better than the almost sure vote for his enemy.

Privacy, robustness. The encrypted vote cannot be decrypted by an outsider or by a group consisting of at most t authorities. Given a list of encrypted votes and a shuffled list, it is infeasible to find out the correct permutation.

Universal Verifiability. Any observer can check whether the authority A_j shuffled the list correctly. This way, any observer can verify whether the last list $e^{(N)}$ is the list of the encryptions of all possible votes. Any observer can compute a product of all encrypted votes selected by the voters, and any observer can verify whether the final tally has been correctly decrypted by the authorities.

Receipt-freeness. Assume that the coercer or vote-buyer does not collude with the authorities. Voter can interact only at two points: he can revert shuffling of at most $N - t - 1$ authorities, and he points at the encrypted vote of his choice. From each authority he receives

permutation together with the proof of its correctness. Untappable channel guarantees that no one is able to intercept this message. Thus, the voter can substitute received data with his own permutation, and due to the no transferability of the designated-verifier proof, he can construct a proof of its correctness as well. This way he can deceive the possible coercer. If the coercer colludes with some of the authorities (at most with t of them), he will know the permutations they made. Therefore, the voter cannot lie about their permutations. If he knows the colluding authorities, he will lie about the permutation of reliable authority. Otherwise he selects one authority randomly and lies for its permutation. Apparently, receipt-freeness holds as long as the voter knows at least one authority not colluding with the coercer. Coercer can only force the voter to vote randomly.

4.4 Scheme based on Secret Sharing Homomorphism

The proposed system (4) is a modification of the existing electronic voting scheme's used in India. Here is proposed solution that uses Internet and secret sharing homomorphism.

In the current Electronic Voting System, when a vote is casted, the corresponding candidates data base entry is updated and it can be easily tracked. But in the proposed scheme, it is difficult to track the vote because the shares of the votes are added to all the servers. Let us assume that there are m candidates $C_1, \dots, C_m$ and n voters $V_1, \dots, V_n$ .Then the binary encoding of the vote corresponding to each candidate will consist of $(\lfloor \log_2 n \rfloor + 1) \times m$ number of bits. Here we consider the fact that all voters may vote to the same candidate. So the number of bits required for the representation of votes for each candidate is equal to the number of bits required to represent the total number of voters which is $(\lfloor \log_2 n \rfloor + 1) \times m$.

The encoding of the vote corresponding to each contesting candidate is explained below with an example. Let us consider that there are three candidates and seven voters. So the total number of bits of each encoded vote will be nine. The bit pattern corresponding to the vote of each candidate is obtained by setting the corresponding bit C_i to 1 in the code $00C_3 00C_2 00C_1$ and other bits C_i to 0. For example the code that corresponds to the vote of candidate C_3 is 001000000(64). So depending on the vote casted, it is encoded into a decimal code of 1,8 or 64 respectively.

The encoded vote is then shared using Shamir's threshold secret sharing scheme. The shares are then sent to different Collection Centres (CC). The Collection Centres are responsible for receiving and summing up the shares corresponding to each vote casted. If there are p collection centres $CC_1, \dots, CC_p$ and a threshold $t < p$ is set so that we can get back the result from any t collection centres. This provides trust and reliability. Based on the number of collection centres and threshold set up, Shamir's scheme can be used for a threshold (t, p) secret sharing. A $t-1$ degree polynomial $q(x)$ is constructed with constant term representing the encoded vote value in decimal. The other coefficients are chosen randomly from the field Z_p, where p is larger than the encoded vote values and the number of

participants. The shares are generated by evaluating the polynomial $q(x)$ at p different values $x_1, \dots, x_p$. These $x_1, \dots, x_p$ values represent different collection centres. These shares are then sent securely to the p collection centres. Any t of them can be used for result evaluation and verification. The shares look totally random and the collection centres have no idea regarding which secret (vote) share it is, from the share value. The collection centres are responsible for summing up the shares they receive for vote tallying. Collection centres behave as group of authorized parties.

The Result Computation module is responsible for computing and declaring the final result. From the sum of shares stored on collection centres, the result can be obtained using Lagrange Interpolation. If there are p collection centres and a (t, p) threshold secret sharing scheme is used, then any t of the share sum from the collection centres can be used for computing the final result. These t shares can be used to get back a $t-1$ degree polynomial $Q(x)$ and the encoded result will be $Q(0)$. The result is then decoded by converting $Q(0)$ into binary and then separating the bits corresponding to each candidate. The decimal equivalent of the separated bits represents the total vote obtained by each candidate. Based on this the result can be announced.

Algorithm 1: E-Voting

Input: Vote casted by the voters
Output: Sum of the shares of the votes

Let m denote the number of candidates and n denote number of voters.
Set V = $(\lfloor \log_2 n \rfloor + 1) \times m$ bits for encoding the votes.
Choose an appropriate field Zp.
for each vote $i = 1 : n$ do
$enc\_vote = bin\_decimal\ (set\_bit\ (V))$

V is set according to the vote casted, enc_vote is the encoded vote in decimal

Pick $t-1$ random numbers $a_1, \dots, a_{t-1}$ from Zp
Construct the polynomial
$q(x) = a_1 x + \cdots + a_{t-1} x^{t-1}$ + encvote
for $j = 1 : p$ **do**
Generate share $V_{ij} = q(j)$, where V_{ij} is the jth share of ith vote
Send the share V_{ij} to C_j^{th} collection centre through a secure communication channel
end

for each Collection Centre $j = 1 : p$ **do**
Sum of shares $SCC_j = SCC_j + V_{ij}$
end

end

Algorithm 2: Result Computation

Input: Share sum of the votes from collection centre
Output: Votes obtained by each candidate

for each randomly chosen t Collection Centre $j = 1 : t$ **do**
retrieve SCC_j
end

Interpolate using SCC_j and corresponding x_i values to obtain the polynomial $Q(x)$
Obtain the secret value $Q(0)$.
Decode $Q(0)$ and obtain the binary representation.
Each $(\lfloor \log_2 n \rfloor + 1)$ bits will represent each candidates vote.
Publish the final results.

5 Security

Despite the extensive research going on in the field of electronic and remote voting there are still a few open problems to consider.

Security threats may be the result of system failure, intimidation, vote buying and selling, and hacker and computer viral tampering. Threats must be identified, and reasonably dealt with to ensure the fairness, freedom, and openness of an election.

The secure platform problem describes the dilemma that the voter should be able to vote using a home computer or other device owned by the voter but realistically cannot, because it is easy to compromise said devices. Most voting protocols assume some sort of trusted device to perform the voting on, but a lot of voter-owned devices are either already infected with malicious software or could easily be infected, if someone wants to manipulate an election. Malicious software could change the voters choice, not send the vote, pretend to have voted correctly but misbehave, record the choice and lift the anonymity of the voter with the recorded ballot. The list of possibilities is limitless. Furthermore the platform the election logic is running on (e.g. tallying-servers, bulletin boards, etc.) could be compromised as well.

Typically voting schemes and systems are analyzed regarding theoretical threats that could be possible because of weak cryptography or flaws in the protocol design. But in real-world systems there is also the component of the implementation. A real-world election system is a piece of software running on servers connected to the Internet. As such they are potentially vulnerable to network-based attacks like DDoS, Man-in-the-Middle, packet sniffing, compromised key attacks, etc. Another point of concern are the software-components used to implement the voting protocols. They could contain back-doors or use vulnerable libraries thus potentially opening the whole system-network to attackers.

Another source of threats is fraud-related. Fraud-related threats may be attributed to intimidation, and vote buying and selling. These threats are not unique to electronic election systems. However, in a traditional voting system, intimidation, and vote buying and selling, although happen, may not be effective as it is almost impossible for a voter to prove that he/she has voted in favour of a candidate set by the deal. Electronic election systems allow voters to cast their votes from their own chosen places. It is possible for voters to show their choices of candidates. Worse yet, the buyers may purchase a large of voting rights from voters and cast their votes on their behalf. This can never happen in a traditional system if properly administrated.

To the general public, IT is a new concept. In general, the public normally will reject a new concept if they do not have any prior experience with it. They do not understand the technology and hence, naturally remain sceptical. To earn the public confidence, the process using an electronic election system must be as transparent as possible and must be subject to the public examination. Of course, such an open approach would theoretically give more clues and hints to hackers.

In voting it is assumed that the voters will exercise their rights ethically and legally. This assumption is not realistic, since some corrupted voters may be willing to sell their voting rights to a buyer and let the buyer vote on their behalf. To ensure the identity of a voter, other means, such as fingerprint and retinal scan, may be used. These methods are technically feasible, but are expensive to implement.

If actual security threats are not properly addressed in an electronic election system, security threats perceived by the public will be high and the election will not be viewed as being fair, free, and open.

6 Conclusion

In this project I presented some of the underlying cryptography behind e-voting. Special attention was given to homomorphic encryption, its property of decrypting sum of encrypted votes without knowing plaintext of each of them, and its use in various schemes.

E-voting was tested and applied in many countries. As it offers many advantages regarding development, technology, speed and security, it is time to expand its use and change people's perspective regarding this new, modern approach to voting.

Bibliography

1. *Secure E-Voting With Blind Signature.* **S. Ibrahim, M. Kamat, M. Salleh, S. R. A.l Aziz.** 2003.

2. *Efficient Zero-Knowledge Argument for Correctness of a Shuffle.* **S. Bayer, J. Groth.** 2012. https://link.springer.com/content/pdf/10.1007%2F978-3-642-29011-4_17.pdf.

3. *Introduction to Electronic Voting.* **Zagorski, F.** 2012. https://zagorski.im.pwr.wroc.pl/courses/voting_2012/voting.pdf.

4. *Secret Sharing Homomorphism and Secure E-voting.* **Binu V.P, Divya G Nair, Sreekumar A.** 2016. https://arxiv.org/pdf/1602.05372.pdf.

5. [book auth.] R. Paulet, E. Bertino Xun Yi. *Homomorphic Encryption and Applications.* 2014. https://www.google.si/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&ved=0ahUKEwiP097TkP7YAhVD_qQKHaDxCUoQFgguMAE&url=http%3A%2F%2Fwww.springer.com%2Fcda%2Fcontent%2Fdocument%2Fcda_downloaddocument%2F9783319122281-c1.pdf%3FSGWID%3D0-0-45-1487904-p177033600&usg=.

6. *Electronic Voting Schemes.* **Murk, O.** 2000. http://www.academia.edu/761518/Electronic_Voting_Schemes.

7. —. **Rjaskova, Z.** 2002. https://people.ksp.sk/~zuzka/elevote.pdf.

8. *Survey on Remote Electronic Voting.* **A. Schneider, C. Meter, P. Hagemeister.** 2017. https://arxiv.org/pdf/1702.02798.pdf.

9. *e-Voting - A survey and introduction.* **Rossler, T.** 2004. https://www.a-sit.at/pdfs/evoting_survey.pdf.