Course Code: CS594 **By: vchenn6, rathin3**
SUBJECT : Applied Cryptography

An Anonymous Verifiable Voting System

Introduction

In today's world, ensuring that voting is both secure and private is more important than ever. Our project focuses on creating a system where people can vote without anyone else knowing their choices. This kind of system uses special methods to keep votes secret and make sure that all votes are counted correctly.The key to our system is using what's called public-key encryption and mixnets. Public-key encryption helps keep each vote hidden and secure. Mixnets, on the other hand, shuffle all the votes together. This shuffling is done by a series of servers one after the other, which makes it hard to trace a vote back to the voter. After all the votes are mixed up, they end up in a random order, cutting any ties between a vote and the person who cast it.What makes our system even more reliable is something called zero-knowledge proofs. These are clever ways to check that all votes are shuffled correctly without actually showing the votes themselves. This ensures that no votes are changed or lost in the shuffle.By building this system, we aim to provide a voting method that not only keeps everyone's vote private but also lets everyone check that the voting process was fair and accurate, without compromising vote secrecy.

System Components

- **Voter**: The voter component is crucial in ensuring that the votes are cast securely and anonymously. In our system, each voter encrypts their vote using a public key issued by the auditor. This process ensures that the vote remains confidential and tamper-proof until decryption. The encryption method used ensures that the encrypted vote remains unreadable to anyone except the auditor who possesses the corresponding private key. This method aligns with traditional practices in cryptographic systems where data confidentiality is paramount.
- **Auditor**: The auditor serves a dual role in our system. Firstly, the auditor is responsible for generating and distributing the public key used for vote encryption. This key management process is critical as it ensures that all voters have a uniform and secure method of encrypting their votes. Secondly, after the voting process concludes, the auditor decrypts the collected votes using the private key. This step is sensitive and crucial as it translates encrypted data back into readable votes which are then tallied. The auditor also plays a key role in verifying the integrity of the election process, ensuring that the voting process has been fair and free from tampering.
- **Mixing Servers**: Mixing servers add an additional layer of anonymity to the voting process. Once votes are cast and collected, they are passed through a series of mixing servers. Each server shuffles and re-encrypts the votes, further anonymizing them by breaking the direct link between the voter and their vote. This process is crucial in protecting voter privacy and preventing any potential vote tracking or manipulation. The mixing servers also provide zero-knowledge proofs to demonstrate that they have correctly shuffled the votes without altering or losing any data, thus maintaining the integrity and anonymity of the voting process.

System Workflow

- **Key Distribution**: Prior to the election, a trusted entity, which we call the auditor, is responsible for creating a secure environment for the voting process. The auditor generates two very important pieces of information: a public key (often denoted as pki) and a private key (ski). The public key is akin to a locked mailbox: anyone can drop something inside (in this case, a vote), but only someone with a key (the private key) can open it and see what's inside. This public key is shared with all eligible voters, enabling them to encrypt their votes in preparation for casting.
- **Vote Casting**: In the voting phase, participants use the auditor's public key to encrypt their vote, which is essentially their chosen candidate. This encryption transforms the vote into a secure format that can't be read by others, ensuring that the voter's choice remains a secret. Once encrypted, the vote is submitted to the system, joining the pool of other encrypted votes which are already available.
- **Vote Mixing and Re-encryption**: After the voting period ends, the collected votes undergo a process called mixing, performed by a set of servers known as the mixnet. The mixnet's job is to shuffle all the encrypted votes, similar to mixing a deck of cards, so that no one can tell which vote came from which voter. But shuffling isn't enough on its own; to further enhance secrecy, the mixnet also re-encrypts the votes. This doesn't change who each vote is for; it just gives the encrypted vote a new disguise. While doing all this, the mixnet also creates a special kind of assurance, known as a zero-knowledge proof, which is a way to prove that the shuffling and re-encryption were done correctly without actually showing the original or final order of the votes.
- **Decryption by Auditor**: Following the mixnet's thorough shuffling and re-encryption, the auditor comes back into the picture. Using the private key that pairs with the earlier distributed public key, the auditor can unlock and decrypt the votes. Thanks to the mixnet's careful work, even though the auditor can see the votes, they can't tell who voted for who. The encryption methods ensure that the mixing process does not affect the auditor's ability to count the votes accurately.
- **Tallying and Result Announcement**: With the votes decrypted, the auditor counts them up. This is the moment of truth where the votes for each candidate are tallied, and the winner is determined. The integrity of the entire process, upheld by the zero-knowledge proofs, means that everyone can trust in the announced results, even though the inner workings of the votes' journey through the mixnet remain private.

Technical Implementation

- **Imports**: getPrime, GCD, inverse, and getRandomRange are imported from Crypto.Util.number module. These are used for generating prime numbers, calculating greatest common divisor, finding inverses modulo, and generating random numbers in a specified range. randrange and shuffle are imported from the random module for generating random numbers and shuffling lists, respectively. hashlib is imported for cryptographic hashing functions.
- **key generation**: Generates ElGamal public-private key pair. p is a large prime number, g is a generator, x is a randomly chosen integer in the range [2, p - 1], h is computed as g power x mod p and returns the public key tuple (p, g, h) and the private key x.
- **Encryption**: Encrypts a vote choice using the ElGamal encryption scheme.Generates a random number k in the range [2, p - 1].Computes the ciphertext pair (c1, c2) and returns the pair.
- **Vote shuffling**: Shuffles the encrypted votes to provide anonymity.For each encrypted vote (c1, c2), Generates a random number k' in the range [2, p - 1]. Computes new ciphertext pair (c1', c2') where c1' = (c1 * g power k') mod p and c2' = (c2 * h power k') mod p. Returns the shuffled and re-encrypted votes.
- **Zero-Knowledge Proof in Voting System**: In our voting system, zero-knowledge proofs play a crucial role in ensuring that the shuffling of votes is done correctly without revealing individual votes. We use a combination of commitment schemes and zero-knowledge proof techniques to achieve this. For each shuffled vote, commitments are generated for both the original and shuffled votes. These commitments are then used in a zero-knowledge proof to show that the commitments represent the same set of votes, thus proving the shuffle was honest without revealing the actual vote content. This method ensures that even if the votes are shuffled and re-encrypted, they remain anonymous and untraceable back to the original voter, maintaining voter privacy and the integrity of the voting process.
- **Verifying the Integrity of the Shuffle**: To verify the integrity of the voting shuffle, our system uses complex algorithms that check the consistency of vote encryption before and after the shuffle. By employing mathematical proofs that leverage the properties of the zero-knowledge proofs, the system can confirm that the order and content of votes have been altered only in terms of their sequence and not in their actual content. These proofs ensure that the shuffle does not introduce any bias or modify the votes, thereby upholding the fairness and transparency of the voting process.
- **Decryption and Tallying**: Decrypts a vote using the ElGamal decryption algorithm. For a given ciphertext pair (c1, c2) and private key x, computes: s = (c1 power x) mod p and m = (c2 * s power -1) mod p. Returns the decrypted vote m.

Zero-Knowledge Proofs for Mixnet Verification

A Zero-Knowledge Proof (ZKP) is a method by which one party (the prover) can prove to another party (the verifier) that a given statement is true, without conveying any additional information apart from the fact that the statement is indeed true. In the context of voting, this typically involves proving operations like shuffles and re-encryptions were performed correctly without revealing the original votes or the specifics of the shuffle process. Commitments in this code refer to cryptographic commitments made to specific values or states before and after the shuffle and re-encryption processes. These are used to ensure that the values haven't been tampered with and remain consistent through the process, without revealing the underlying values themselves. For each re-encryption or shuffle, a new commitment can be generated to the result. The commitments are then used in a zero-knowledge context to prove that the transformations (shuffles and re-encryptions) are legitimate.Each commitment in our system likely corresponds to a set of encrypted or transformed votes. The verification success messages you see (e.g., "Verification Success for commitment 0") indicate that a ZKP has been successfully completed for that particular set of data.For each set of data (or for each commitment), a ZKP would involve generating proofs that the data matches expected cryptographic patterns or behaviors (such as maintaining consistency in encryption despite re-ordering). This proves that the operations were performed correctly without revealing what those operations were exactly.When votes are shuffled and re-encrypted, your system needs to ensure that despite the operations, the integrity and anonymity of the votes are maintained. This involves proving relationships like logg (original vote) = logg (shuffled and re-encrypted vote) modulo some parameters, but without revealing the original vote, shuffled and re-encrypted vote, or the logs themselves.SILMPP stands for Shuffle Integer Logarithm Mix Proof Protocol, a specific type of zero-knowledge proof protocol designed to complex operations like shuffles in mixnets.The "SILMPP verification successful" message indicates that the zero-knowledge proofs required to verify the correctness of the shuffle and mix operations have been successfully validated. This means the integrity of the shuffle process (and thus the anonymity and correctness of the voting process) is maintained, and it has been verified without revealing any underlying data about the votes themselves.In our project, ZKPs are crucial for maintaining voter privacy and ensuring that no unauthorized information about voter choices or the order of votes is leaked while still allowing external verification of the election's integrity. These proofs support the claims of correct, unbiased shuffling and re-encryption by providing mathematical evidence that all actions were performed correctly, adhering to the designed protocols without exposing any sensitive details.

Literature Survey

- **A Verifiable Secret Shuffle and its Application to E-Voting - 05 November 2001**: Neff's paper presents an innovative approach to encrypting votes in such a manner that allows them to be shuffled and thus anonymized without the possibility of tracking back to the voter. His method not only ensures the privacy of individual votes but also allows for a verification process that doesn't compromise the secrecy of the ballot. This concept struck me as a robust solution to one of e-voting's most pressing challenges: maintaining voter anonymity while ensuring that every vote is both counted and verifiable. Neff's technique of employing mathematical algorithms to shuffle encrypted data addresses the dual need for transparency in the tallying process and privacy for the voter, an approach that has been integral to developing our own project's voting system.
- **Verifiable Shuffle of Large Size Ciphertexts.**: The authors introduce innovative techniques for shuffling ciphertexts—a process crucial for maintaining anonymity in electronic voting systems. Their work provides two efficient honest verifier zero-knowledge (HVZK) arguments that validate the correctness of a shuffle without compromising voter privacy. What resonates is their emphasis on minimizing the complexity of communication and computation, especially pertinent when handling a vast number of votes. By streamlining the verification process and reducing the round-complexity, Groth and Lu make significant headway in rendering large-scale voting systems both practical and trustworthy. Their research offers a new lens through which we can view anonymity and verification in voting protocols, setting a precedent for the development of more secure and scalable electronic voting systems.
- **A Review of Cryptographic Electronic Voting - 21 April 2022**: In the literature survey on "A Review of Cryptographic Electronic Voting," the paper comprehensively discusses various electronic voting schemes such as mix-net-based, homomorphic, blind signature-based, blockchain-based, post-quantum, and hybrid e-voting systems. It evaluates their structures, advantages, and disadvantages, providing a detailed analysis of their security properties and functional requirements. The review also addresses practical considerations in the design of e-voting systems and outlines potential research directions, highlighting the evolution of e-voting technologies and their implications for future electoral processes.
- **Verifiable Mix-Nets and Distributed Decryption for Voting from Lattice-Based Assumptions - 22 April 2022**: the authors provide a detailed examination of cryptographic techniques aimed at enhancing the security and transparency of electronic voting systems. The paper delves into the design and implementation of verifiable mix-nets, a crucial component in ensuring voter anonymity by reliably shuffling encrypted votes in a manner that precludes any subsequent tracing back to individual voters. The research further explores distributed decryption mechanisms, which allow for the collective decryption of votes without requiring a single party to hold the decryption key. This method significantly reduces the risk of vote tampering or privacy breaches by dispersing the decryption process among multiple trusted parties.One of the standout features of this study is its focus on the integration of zero-knowledge proofs with mix-nets and distributed decryption systems. Zero-knowledge proofs are used to affirm that each step of the vote shuffling and decryption was executed correctly without revealing any underlying vote data or decryption keys. This approach not only bolsters the integrity of the electoral process but also upholds the confidentiality essential to maintaining voter trust.

Code Implementation

Our implementation begins by interacting with users to gather the number of participants in the election and their respective votes. For simplicity and to maintain focus on the cryptographic aspects, voters input their choices as integer values corresponding to candidates (e.g., '1' for Candidate 1, '2' for Candidate 2). These inputs are then encrypted using the public key derived from our ElGamal key generation process, ensuring each vote is securely anonymized prior to further processing.

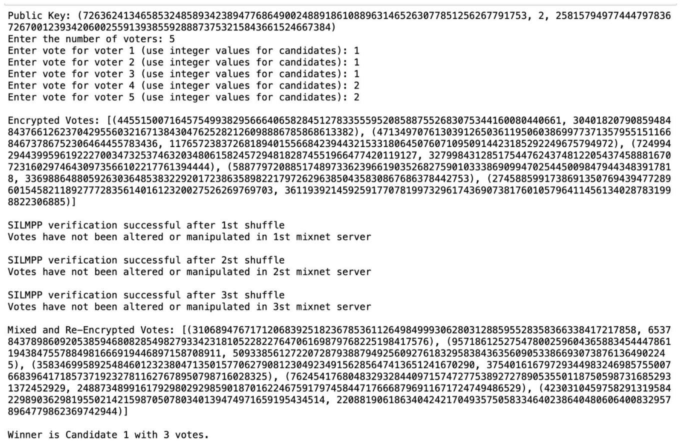
To implement the mixnet functionality, our system utilizes the mix_and_reencrypt_votes function. This function accepts the list of encrypted votes and performs two main actions: shuffling the votes to break any order-based patterns, and re-encrypting them to renew their cryptographic robustness. This simulates the role of a mixnet server in a real-world voting scenario. For each shuffle and re-encryption, corresponding to a single pass through a mixnet server, our system engages a zero-knowledge proof (ZKP) mechanism to verify the integrity of the shuffled votes without revealing their contents. This verification adheres to Neff's General k-Shuffle protocol, ensuring that the operations performed by the mixnet servers can be trusted to be fair and unbiased.

The system is configured to simulate the passage of votes through three sequential mixnet servers. This is intended to demonstrate robustness against potential manipulation by any single server and to showcase the layering of anonymity. After each round, a ZKP is executed, labeled as SILMPP verification successful if the zero-knowledge proof confirms the correct shuffle without vote alteration. For demonstration purposes, we include a scenario where vote manipulation is simulated to test the system's ability to detect and report such irregularities. Specifically, we compare the shuffled votes against the original encrypted votes using a straightforward comparison. If a discrepancy is detected, a specialized version of the ZKP, ZKP_check, is triggered. This function includes an additional parameter for the manipulated votes and evaluates whether the cryptographic commitments and responses still align with the expected results under Neff's protocol.

If the ZKP_check determines that the votes have been tampered with during the mixing process, it outputs a failure message, indicating precisely which mixnet server could be responsible for the tampering, enhancing the traceability and accountability within the system. For further verification and to illustrate the concept of vote manipulation, we include a line of code that artificially alters the encrypted votes, simulating a potential security breach by an untrustworthy mixnet server. This is intended to validate that our system can effectively detect and respond to unauthorized changes in the vote data.

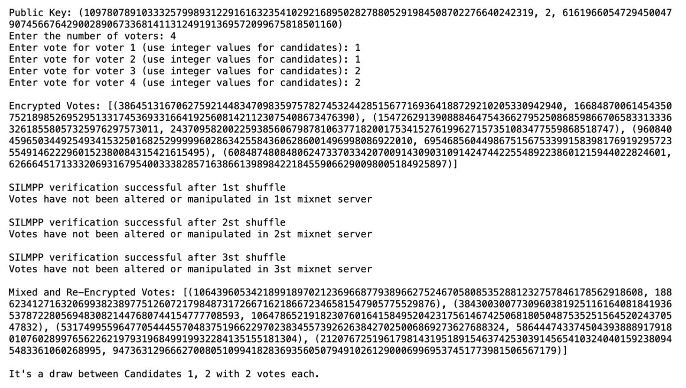
Test cases and Outputs

- **Test case - 1**: Lets assume we have 5 voters and voters have voted to respected individual whom they want to vote. The auditor will announce the winner respectively. Please find the attached screenshot for the output.



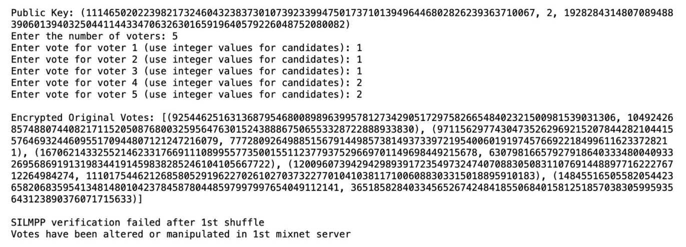
```
Public Key: (72636241346585324858934238947768649002488918610889631465263077851256267791753, 2, 25815794977444797836
72670012393420600255913938559288873753215843661524667384)
Enter the number of voters: 5
Enter vote for voter 1 (use integer values for candidates): 1
Enter vote for voter 2 (use integer values for candidates): 1
Enter vote for voter 3 (use integer values for candidates): 1
Enter vote for voter 4 (use integer values for candidates): 2
Enter vote for voter 5 (use integer values for candidates): 2

Encrypted Votes: [(44551500716457549938295666406582845127833555952085887552683075344160080440661, 30401820790859484
84376612623704295560321671384304762528212609888678586861 3382), (471349707613039126503611950603869977371357955151166
84673786752306464455783436, 11765723837268189401556684239443215331806450760710950914423185292249675794972), (724994
29443995961922270034732537463203480615824572948182874551966477420119127, 32799843128517544762437481220543745888167 0
72316029746430973566102217761394444), (588779720885174897336239661903526827590103338690994702544500984794434839178 1
8, 33698864880592630364853832292017238635898221797262963850435830867686378442753), (27458859917386913507694394772 89
60154582118927772835614016123200275262697 69703, 36119392145925917707819973296174369073817601057964114561340287831 99
8822306885)]

SILMPP verification successful after 1st shuffle
Votes have not been altered or manipulated in 1st mixnet server

SILMPP verification successful after 2st shuffle
Votes have not been altered or manipulated in 2st mixnet server

SILMPP verification successful after 3st shuffle
Votes have not been altered or manipulated in 3st mixnet server

Mixed and Re-Encrypted Votes: [(31068947671712068392518236785361126498499930628031288595528358366338417217858, 6537
84378986092053859468082854982793342318105228227647061698797682251984 17576), (957186125275478002596043658834544478 61
19438475578849816669194468971587 08911, 50933856127220728793887949256092761832958384363560905338669307387613649022 4
5), (35834699589254846012323804713501577062790812304923491562856474136512416 70290, 37540161679729344983246985755007
66839641718573719232781162767895079871 6028325), (7624541768048329328440971574727753892727890535501187505987316852 93
1372452929, 24887348991617929802929859018701622467591797458447176668796911671724749486529), (42303104597582913195 84
22989036298195502142159870507803401394749716591954 34514, 22088190618634042421704935750583346402386404806064008329 57
8964779862369742944)]

Winner is Candidate 1 with 3 votes.
```

- **Test case - 2**: Lets assume we have 4 voters and voters have voted to respected individual whom they want to vote. The auditor will announce the winner respectively. In this case lets assume both the parties have got equal number of votes. Please find the attached screenshot for the output.



```
Public Key: (10978078910333257998931229161632354102921689502827880529198450870227664 0242319, 2, 6161966054729450047
90745667642900289067336814113124919136957209967581 8501160)
Enter the number of voters: 4
Enter vote for voter 1 (use integer values for candidates): 1
Enter vote for voter 2 (use integer values for candidates): 1
Enter vote for voter 3 (use integer values for candidates): 2
Enter vote for voter 4 (use integer values for candidates): 2

Encrypted Votes: [(38645131670627592144834709835975782745324428515677169364188729210205330942940, 16684870061454350
75218985269529513317453693316641925608142112307540867347 6390), (154726291390888464754366279525086859866706583313336
32618558057325976297573011, 24370958200225938560679878106377182001753415276199627157351083477559868518747), (960840
45965034492549341532501682529999960286342558436062860014969980869 22010, 69546856044986751567533991583981769192957 23
55491462229601523800843154216154 95), (60848748084806247337033420700914309031091424744225548922386012159440228246 01,
62666451713332069316795400333828571638661398984221845590662900980051849 25897)]

SILMPP verification successful after 1st shuffle
Votes have not been altered or manipulated in 1st mixnet server

SILMPP verification successful after 2st shuffle
Votes have not been altered or manipulated in 2st mixnet server

SILMPP verification successful after 3st shuffle
Votes have not been altered or manipulated in 3st mixnet server

Mixed and Re-Encrypted Votes: [(10643960534218991897021236966877938966275246705808535288123275784617856291 8608, 188
62341271632069938238977512607217984873172667162186672346581547905775529876), (38430030077309603819251161640818419 36
53787228056948308214476807441547777 08593, 10647865219182307601641584952042317561467425068180504875352515645202437 05
47832), (53174995596477054445570483751966229702383455739262638427025006869273627688324, 58644474337450439388891791 8
01076028997656226219793196849919932284135155181304), (21207672519617981431951891546374253039145654103240401592380 94
5483361060268995, 94736312966627008051099418283693560507949102612900069969537451773981506567179)]

It's a draw between Candidates 1, 2 with 2 votes each.
```

- **Test case - 3**: Lets assume we have 5 voters and voters have voted to respected individual whom they want to vote. In this case votes have been altered. Please find the attached screenshot for the output.



```
Public Key: (11146502022398217324604323837301073923399475017371013949644680282623936371 0067, 2, 192828431480708948 8
39060139403250441144334706326301659196405792260487520 80082)
Enter the number of voters: 5
Enter vote for voter 1 (use integer values for candidates): 1
Enter vote for voter 2 (use integer values for candidates): 1
Enter vote for voter 3 (use integer values for candidates): 1
Enter vote for voter 4 (use integer values for candidates): 2
Enter vote for voter 5 (use integer values for candidates): 2

Encrypted Original Votes: [(92544625163136879546800898963995781273429051729758266548402321500981539031306, 10492426
85748807440821711520508768003259564763015243888675065533287228889 33830), (971156297743047352629692152078442821044 15
57646932446095517094480712124721 6079, 77728092649885156791449857381493733972195400601919745766922184996116233728 21
1), (16706214332552146233176691110899557735001551123779375296697011496984492 15678, 63079816657927918640333480040933
26956869191319834419145983828524610410566 7722), (120096073942942989391723549732474070883050831107691448897716222767
12264984274, 11101754462126858052919622702610270373227701041038117100608830331501889591 0183), (1484551650558205442 3
65820683595413481480104237845878044859799799765404911 2141, 36518582840334565267424841855068401581251857038305995935
64312389037607171 5633)]

SILMPP verification failed after 1st shuffle
Votes have been altered or manipulated in 1st mixnet server
```

Conclusion

In conclusion, our project successfully creates an Anonymous Verifiable Voting System that upholds the privacy of voters while ensuring the integrity and verifiability of the voting process. By integrating cryptographic key generation, secure vote encryption, mixnets for vote anonymization, and zero-knowledge proofs, the system sets a high standard for secure digital elections. It addresses the fundamental need for trust in electronic voting by allowing voters to confidently cast their ballots, knowing their choices are both private and counted accurately. The implementation of this system represents a significant step forward in the pursuit of a democratic process that is both transparent and resilient against tampering, making it a vital advancement for the future of voting in the digital age.

REFERENCES

- A verifiable secret shuffle and its application to e-voting. (ref:https://dl.acm.org/doi/pdf/10.1145/501983.502000)
- Verifiable Shuffle of Large Size Ciphertexts. (ref:https://iacr.org/archive/pkc2007/44500377/44500377.pdf)
- A Review of Cryptographic Electronic Voting. (ref:https://doi.org/10.3390/sym14050858)
- Verifiable Mix-Nets and Distributed Decryption for Voting from Lattice-Based Assumptions. (ref:https://eprint.iacr.org/2022/422)