Group List

姓名	學號
賴邑城	112550009
周廷威	112550013
傅永威	112550107
許有暢	112550168
蔡尚融	112550201

Summary

1. **Introduction**

”Civitas: Toward a Secure Voting System” describes and evaluates a novel and secure electronic voting system designed to address the significant challenges of maintaining integrity, confidentiality, and resistance to coercion in electronic voting environments, particularly in remote settings. The core of Civitas ’ s innovation lies in its application of several cryptographic techniques to ensure that elections can be both universally verifiable and coercion-resistant, without relying on trusted supervision of polling places, which makes it suitable for remote voting. Civitas is also the first voting system “ proved”to satisfy coercion resistance and verifiability, with experimental results showing cost, tabulation time, and security can be practical for real-world elections.

2. **Security Model**

The paper provides an extensive discussion of the security model adopted by Civitas including key properties like:

(a) Remote Voting

Civitas is designed for remote voting scenarios. It can address the lack of trusted supervision by enabling a system where the security does not depend on the physical security of voting booths.

(b) Security Assumptions

The system’s security is predicated on a series of realistic assumptions about the adversary’s capabilities and the environment. For instance, it assumes that the adversary can coerce voters and tamper with public network communications but cannot break through all layers of the system’s cryptographic defenses.

(c) Cryptographic Protocols

Civitas employs a wide range of cryptographic protocols, including zero-knowledge proofs, re-encryption schemes, and mixed networks, to protect the confidentiality and integrity of votes while ensuring that the election process is transparent and verifiable.

3. **Design and Implementation**

   The design of Civitas refines and builds upon the JCJ (Juels, Catalano, and Jakobsson) cryptographic voting scheme while introducing several key enhancements to the traditional electronic voting model. Civitas is constructed of 3 phases:

   (a) Setup Phase

   The Civitas system prepares for an election in the setup phase, ensuring that all necessary components are configured to support secure and efficient voting. This phase involves several key steps:

   i. Election Configuration: The election supervisor initiates the process by setting up the election parameters, including defining the ballot structure and the specific roles and responsibilities of various agents involved in the election, such as registration tellers and tabulation tellers.

   ii. Public Key Generation: The supervisor posts RSA public keys representing the election authorities; the tabulation tellers generate a distributed El Gamal public key; the registrar posts each voter's registration public key (also using RSA for convenience) and designation public key (El Gamal).

   iii. Voter Registration: Voters are registered by the system, during which their identities are verified and their voting credentials are published. These credentials are crucial for ensuring that only qualified voters can cast votes and that they can do so anonymously.

   iv. Credential Distribution: Once voters are registered, registration tellers distribute credentials to voters. These credentials are cryptographically protected to prevent tampering and unauthorized access.

   (b) Voting Phase

   The voting phase is when voters interact with the Civitas system to cast their votes. This phase is designed to be user-friendly while under strict security standards while maintaining voter privacy and confidence in the electoral process. This phase involves several key steps:

   i. Voter Authentication: Each voter authenticates themselves using their secure credentials. This step ensures that each vote is cast by a legitimate and qualified voter.

   ii. Secure Vote Casting: Voters cast their votes through a secure interface and the votes are encrypted using the public keys set up in the initial phase, ensuring that they remain confidential and tamper-proof.

   iii. Vote Submission: After a vote is cast and encrypted, it is submitted to one of the designated ballot boxes. These submissions are also accompanied by proofs that verify the integrity and correctness of the vote without revealing the voter's identity or choices.

(c) Tabulation Phase

The tabulation phase involves collecting, verifying, and counting all the votes to determine the election outcome. This phase is highly sensitive and is under strict security standards to ensure accurate results. This phase involves several key steps:

i. Vote Collection: All votes stored in the ballot boxes are collected. This process is secured to prevent any external interference or internal tampering.

ii. Vote Verification: Before counting, each vote is verified for its integrity and authenticity. This involves checking cryptographic signatures and validating the vote against the voter's credentials without revealing the voter's identity.

iii. Counting and Results Compilation: Once verified, the votes are decrypted and counted. The tabulation tellers are responsible for this process, and they ensure that all votes are counted correctly and that the final tally is accurate.

iv. Result Publication: After the votes are counted and verified, the results are compiled and published for public viewing. This process is also protected by cryptographic measures to ensure that the results cannot be tampered with after tabulation.

4. **Experimental Evaluation**

A significant portion of the paper is dedicated to the experimental evaluation of Civitas, where the authors presented a thorough analysis of the system's performance in terms of security, cost, and time efficiency. The experiments demonstrate that Civitas can achieve a balance between cost and security, offering an extendable solution that could be feasibly deployed in real-world elections. The analysis shows that the costs of implementing Civitas are competitive with current government election costs while providing much higher security guarantees.

5. **Contributions and Innovations**

Civitas's development led to several contributions to the field of electronic voting:

(a) Secure Voter Registration Protocol: Introducing a novel protocol for voter registration that distributes trust among multiple registration authorities to minimize the risk of voter impersonation or unauthorized voting.

(b) Scalable Vote Storage Design: Proposing a system architecture that maintains vote integrity and confidentiality without requiring expensive fault-tolerance mechanisms typically used in distributed systems.

(c) Performance Study: Providing empirical data that supports the viability of Civitas in large-scale elections, showing that it can handle the complexities of real-world voting scenarios effectively and efficiently.

Strength(s) of the paper

1. **Utilizes Advanced Cryptographic Techniques**
   Civitas utilizes a cutting-edge cryptographic framework, ensuring that all voting activities are secured through zero-knowledge proofs and secure multi-party computations. These technologies not only preserve the confidentiality of each vote but also enable every voter to independently verify that their vote has been correctly tallied, providing privacy and transparency that is rare in electronic voting systems.
2. **Innovative Methods to Accomplish Coercion Resistance**
   One of the most innovative aspects of Civitas is its ability to allow voters to submit their votes even under coercion without revealing their true voting intentions. By enabling the use of fake credentials that appear valid, Civitas protects voters from being forced into voting against their will, a critical feature that enhances voter freedom and integrity within electoral processes.
3. **Decentralizes Trust Architecture**
   By distributing electoral responsibilities across multiple independent agents, Civitas minimizes the risks of systemic fraud or tampering. This structure reduces the chance of a single point of failure, which may cause systematic breakdown in traditional systems. This design also ensures that no single entity has complete control over the whole electoral process.
4. **Empirical Validation**
   The robust empirical analysis provided in the paper validates the operational capabilities of Civitas under various scenarios, offering concrete data on its scalability, cost-effectiveness, and security. Such comprehensive validation is crucial for gaining the confidence of election authorities and the public, demonstrating the system's readiness for practical deployment.
5. **Contribution to Democratic Processes**
   Civitas could revolutionize democratic engagement by making voting more secure and accessible. Its implementation across diverse electoral environments could help curb voter disenfranchisement and increase participation rates, especially in regions where voters may be deterred by the current lack of secure voting options.

Weakness(es) of the paper

1. **System Complexity**
   The advanced cryptographic elements that make Civitas secure also contribute to its complexity. This complexity might not only be a barrier for the average voter but also for the election officials who must manage and operate the system. The need for specialized knowledge in order to effectively utilize the system could hinder its adoption, especially in less technologically advanced regions or countries.
2. **Assumption of Secure Client Devices**
   The system's reliance on secure client devices for voting is a significant vulnerability. Many voters may not have access to devices that can guarantee the security of the electoral voting system. This reliance exposes the system to risks where an individual's device could be compromised, potentially allowing malicious actors and activities that can influence the voting process.
3. **Lack of Network Security Details**
   While Civitas addresses many security aspects of electronic voting, the paper does not extensively explore solutions for network security threats such as Man-In-The-Middle attacks (MITM) or data breaches. This omission is critical as the integrity of the voting process could be undermined if data transmitted over the network is intercepted or altered.
4. **Practical Implementation Challenges**
   The logistical and financial implications of deploying Civitas on a large scale are not comprehensively and thoroughly discussed. The adaptation of existing electoral systems to integrate such a complex electronic voting system would require significant financial investment, extensive testing, and possibly long-term changes to existing electoral laws and regulations.
5. **Dependency on Technological Infrastructure**
   The effectiveness of Civitas relies heavily on a robust technological infrastructure. In regions where internet access is unreliable or where the general populace lacks technological literacy, the deployment of Civitas could face significant challenges, potentially exacerbating existing inequalities in voter participation.

Your own reflection

The paper concludes with reflections on the broader implications of Civitas for the future of secure electronic voting feasibility. The authors also propose further research directions, including enhancing the usability of the system for ordinary voters, reducing its operational complexity, and exploring more robust defenses against sophisticated cyber threats. They also discuss the potential for Civitas to serve as a model for future voting systems that aim to preserve democratic integrity in the digital age.

We also came up with some reflections, ideas, and research topics that remained undiscussed which can maybe help improve the system, which is shown below:

1. What I Learned from the Paper

   One of the key learnings we had was understanding the balance between robust security mechanisms and the practical usability of voting systems in environments with uncontrolled factors. The use of sophisticated cryptographic techniques, such as zero-knowledge proofs and secure multi-party computations, illustrated a detailed approach to ensuring that votes remain confidential and elections can be independently verified.

   Additionally, Civitas's architecture, which distributes trust across several independent agents, provides a significant shift from traditional centralized models, addressing many of the common vulnerabilities associated with such systems. This decentralized approach not only enhances security by reducing single points of failure but also increases the system's resilience against internal threats and systemic tampering.

2. Improving or Extending the Work Additionally, to further enhance and extend the work on Civitas, several strategic initiatives can be taken into account:

   (a) **User-Centered Design**

   To make Civitas more accessible, prioritizing the development of a user-centered design that simplifies interaction with the system should be helpful. Some sort of good UX/UI design can be adapted to create an intuitive and straightforward interface, hiding the system's complexity behind a user-friendly facade.

   (b) **Integration with Existing Infrastructure**

   Developing integration protocols that allow Civitas to work seamlessly with existing electoral infrastructure, such as voter registration databases and national ID systems, can ensure a smooth transition for voters from traditional voting systems to electronic ones.

   (c) **Advanced Cryptographic Techniques**

   Implementing post-quantum cryptographic algorithms can future-proof the system against potential quantum computing threats. Additionally, exploring the use of blockchain technology could provide a transparent and immutable record of votes, further enhancing trust and integrity.

   (d) **Comprehensive Accessibility Features**

   To ensure that all citizens, including those with disabilities, can use the system independently, Civitas should incorporate advanced accessibility features, adapting to various needs such as visual, auditory, and physical disabilities.

3. Unsolved Questions to Investigate Further research in several key areas could address unresolved questions and enhance the understanding and functionality of systems like Civitas.

   (a) **Voter Education and Acceptance**

How do we educate voters about the security and functionality of electronic voting systems to increase acceptance and trust? What strategies can be implemented to overcome skepticism and resistance to new technologies?

   (b) **Impact of Environmental Factors**

   What are the impacts of different environmental factors, such as internet availability and technological literacy, on the effectiveness of remote electronic voting systems? How can these challenges be mitigated in less developed regions?

   (c) **Longitudinal Studies on System Adoption**

   Conducting longitudinal studies to analyze the adoption, impact, and outcomes of electronic voting systems over multiple election cycles provides insights into their long-term viability and the evolution of voter behavior in response to these technologies.

4. Broader Impacts of the Proposed Technology The introduction of Civitas could potentially revolutionize the way elections are conducted worldwide, providing numerous societal, political, and technological benefits:

   (a) **Strengthening Democracies**

   By making voting more accessible and secure, Civitas could strengthen democratic processes around the world, especially in regions where election integrity is frequently questioned and compromised.

   (b) **Reducing Electoral Costs**

   The automation and digitization of voting processes could significantly reduce the logistical costs associated with elections, making it economically feasible to hold more frequent elections or referendums.

   (c) **Enabling Diaspora and Remote Voting**

   Civitas could facilitate the participation of diaspora and remote populations in their home country's elections, ensuring that all citizens have an opportunity to participate in their governance, regardless of their physical location.

5. Personal Reflection and Future Outlook

   The innovation and predictable success of Civitas could lead to the development of global standards for electronic voting systems, promoting interoperability and security benchmarks that transcend national boundaries.

   Deploying electronic voting systems like Civitas raises several ethical and legal challenges as well, including the risk of disenfranchisement, privacy concerns, and the potential for digital divides. Addressing these issues would require careful planning and robust legal frameworks.

   Beyond improving security and accessibility, Civitas could spur innovation in electoral processes, potentially introducing new forms of engagement such as liquid democracy and real-time policy feedback mechanisms.

   In summary, Civitas represents a significant advancement in electronic voting technology, with the potential to impact various aspects of electoral systems globally. While it offers robust solutions to many existing challenges, its successful implementation will require thoughtful consideration of usability, legal and ethical implications, and ongoing technological advancements.

Realization as a program

We develop a secure voting system to ensure that votes are cast and counted under stringent security conditions. This system is designed to prevent unauthorized access and manipulation, maintaining the integrity and confidentiality of the voting process.

```python
# server.py
import socket
import threading
import time
import sys

PORT = 5050
SERVER = socket.gethostbyname(socket.gethostname())
ADDR = (SERVER, PORT)
HEADER = 1024
FORMAT = 'utf-8'
DISCONNECT_MESSAGE = "!DISCONNECT"
VOTE_MESSAGE = "!VOTE"

server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind(ADDR)

clients = []
clients_voting = {}


def handle_client(conn, addr):
    print(f"[NEW CONNECTION] {addr} connected.")
    connected = True
    while connected:
        try:
            message = conn.recv(HEADER).decode(FORMAT)
            if message:
                print(f"[{addr}] {message}")
            if message == VOTE_MESSAGE:
                print(f"[{addr}] {message}")
                id = conn.recv(HEADER).decode(FORMAT)
                vote = conn.recv(HEADER).decode(FORMAT)
                if not vote:
                    print(f"[DISCONNECT] {addr} disconnected.")
                    connected = False
                    clients.remove(conn)
                    break
                if not ZKP(id, vote):
                    print(f"[ZKP] {addr} failed ZKP.")
                    continue
                if vote in clients_voting:
                    clients_voting[vote] += 1
                else:
                    clients_voting[vote] = 1
                print(f"[{addr}] Voted for {vote}")
                print(f"[VOTES] {clients_voting}")
                broadcast(f"[VOTES] {clients_voting}")
            elif message == DISCONNECT_MESSAGE:
                print(f"[DISCONNECT] {addr} disconnected.")
                connected = False
                clients.remove(conn)
            elif not message:
                print(f"[DISCONNECT] {addr} disconnected.")
                connected = False
                clients.remove(conn)
        except ConnectionResetError:
            print(f"[DISCONNECT] {addr} disconnected.")
            connected = False
            clients.remove(conn)
    conn.close()

def ZKP(ID, vote):
    return 1 #Validating id whithout knowing the who is the voter(Not implemented yet)

def broadcast(message):
    for client in clients:
        client.send(message.encode(FORMAT))

def start():
    server.listen()
    print(f"[LISTENING] Server is listening on {SERVER}")
    while True:
        conn, addr = server.accept()
        clients.append(conn)
        thread = threading.Thread(target=handle_client, args=(conn, addr))
        thread.start()
        print(f"[ACTIVE CONNECTIONS] {threading.activeCount() - 1}")

print("[STARTING] Server is starting...")
start()
```

```python
# client.py
import socket
import threading
import time
import sys

# Create a socket object
PORT = 5050
SERVER = socket.gethostbyname(socket.gethostname())
ADDR = (SERVER, PORT)
HEADER = 1024
FORMAT = 'utf-8'
DISCONNECT_MESSAGE = "!DISCONNECT"
VOTE_MESSAGE = "!VOTE"

client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect(ADDR)

def receive():
    connected = True
    while connected:
        try:
            message = client.recv(HEADER).decode(FORMAT)
            print(message)
        except ConnectionResetError:
            connected = False
    client.close()

def send():
    connected = True
    while connected:
        message = input()
        client.send(message.encode(FORMAT))
        if message == DISCONNECT_MESSAGE:
            connected = False
        elif message == VOTE_MESSAGE:
            id = input()
            vote = input()
            client.send(id.encode(FORMAT))
            client.send(vote.encode(FORMAT))
    client.close()

receive_thread = threading.Thread(target=receive)
send_thread = threading.Thread(target=send)
receive_thread.start()
send_thread.start()
```

Output:

```
# ternimal1.txt
D: \code\voting>python .\server.py
[STARTING] Server is starting...
[LISTENING] Server is listening on 26.64.28.222
[NEW CONNECTION] ('26.64.28.222', 9467) connected.
D: \code\voting\server.py:77: DeprecationWarning: activeCount() is deprecated, use
    active_count() instead
print(f" [ACTIVE CONNECTIONS] {threading.activeCount() - 1}")
[ACTIVE CONNECTIONS] 1
[NEW CONNECTION] ('26.64.28.222', 9470) connected.
[ACTIVE CONNECTIONS] 2
[NEW CONNECTION] ('26.64.28.222', 9473) connected.
[ACTIVE CONNECTIONS] 3
[('26.64.28.222', 9467)] !VOTE
[('26.64.28.222' , 9467)] !VOTE
[('26.64.28.222', 9467)] Voted for 1
[VOTES] {'1': 1}
[('26.64.28.222', 9470)] !VOTE
[('26.64.28.222' 9470)] !VOTE
[('26.64.28.222' 9470)] Voted for 2
[VOTES] {'1': 1, '2': 1}
[('26.64.28.222', 9473)] !VOTE
[('26.64.28.222' , 9473)] !VOTE
[('26.64.28.222' , 9473)] Voted for 1
[VOTES] {'1': 2, '2': 1}
```

```
# ternimal2.txt
D: \code\voting>python ./client.py
!VOTE
H123456789
1
[VOTES] {'1': 1}
[VOTES] {'1': 1, '2': 1}
[VOTES] {'1': 2, '2': 1}
```

```
# ternimal3.txt
D: \code\voting>python ./client.py
[VOTES] {'1': 1}
!VOTE
H23467582
2
[VOTES] {'1': 1, '2': 1}
[VOTES] {'1': 2, '2': 1}
```

```
# ternimal4.txt
D: \code\voting>python ./client.py
[VOTES] {'1': 1}
[VOTES] {'1': 1, '2': 1}
!VOTE
H220039499
1
[VOTES] {'1': 2, '2': 1}
```