Міністерство освіти і науки України

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

З лабораторної роботи № 6 з дисципліни

«Протоколи й алгоритми електронного голосування»

“**Протокол Е-голосування без підтвердження**”

Виконав(ла) *ІП-13 Бабіч Денис* _____

(шифр, прізвище, ім'я, по батькові)

Перевірив(ла) *Нестерук А. О.* _____

(посада, прізвище, ім'я, по батькові)

Київ 2024

ЛАБОРАТОРНА РОБОТА № 6

Тема роботи: Протокол Е-голосування без підтвердження.

Мета роботи: Дослідити протокол Е-голосування без підтвердження.

Основне завдання:

Змоделювати протокол Е-голосування без підтвердження будь-якою мовою програмування та провести його дослідження. Для кодування 6 повідомлень використовувати метод Ель-Гамаля, для кодування Е-бюлетеня використовувати метод BBS.

Умови: В процесі голосування повинні приймати участь не менше 2 кандидатів та не менше 4 виборців. Повинні бути реалізовані сценарії поведінки на випадок порушення протоколу (виборець не проголосував, проголосував неправильно, виборець не має права голосувати, виборець хоче проголосувати повторно, виборець хоче проголосувати замість іншого виборця та інші).

На основі змодельованого протоколу провести його дослідження (Аналіз повинен бути розгорнутим та враховувати всі можливі сценарії подій під час роботи протоколу голосування):

1. Перевірити чи можуть голосувати ті, хто не має на це права.
2. Перевірити чи може виборець голосувати кілька разів.
3. Чи може хтось (інший виборець, ВК, стороння людина) дізнатися за кого проголосували інші виборці?
4. Перевірити чи може інший виборець чи стороння людина проголосувати замість іншого зареєстрованого виборця.
5. Чи може хтось (інший виборець, ВК, стороння людина) таємно змінити голос в бюлетені?
6. Чи може виборець перевірити, що його голос врахований при підведенні кінцевих підсумків?

Виконання завдання:

Для реалізації завдання було створено класи комісій, а саме реєстраційної комісії, яка відповідає за реєстрацію користувачів, видачу їм айді та передачу авторизаційного токену, а також виборчої комісії, яка займається прийомом голосів та створенням пар ключів для кожного виборця. Код цих класів можна побачити на рисунках 1.1 – 1.5.

```python
class RegistrationCommissionController:

    _PASSPORT_ID_PLACEHOLDER: UUID = None
    _CREDENTIALS_CONTENT: str = string.ascii_letters + string.digits

    def __init__(self, voters_count: int) -> 'RegistrationCommissionController':
        self._voter_index: int = -1
        self._usernames: set[str] = set()
        self._voters: list[Voter] = list()
        self._voters_data: dict[UUID, UUID] = dict()
        self._voters_tokens: dict[UUID, AuthToken] = dict()

        for _ in range(voters_count):
            self._voters_data[uuid4()] = RegistrationCommissionController._PASSPORT_ID_PLACEHOLDER

    def get_voters(self) -> list[Voter]:
        return self._voters

    def get_voters_ids(self) -> list[UUID]:
        return list(self._voters_data.keys())

    def setup_voters_tokens(self, auth_tokens: list[AuthToken]) -> None:
        if auth_tokens is None:
            raise ValueError("auth_tokens cannot be None")

        for auth_token in auth_tokens:
            self._voters_tokens[auth_token.get_voter_id()] = auth_token
```

Рисунок 1.1 – Код класу реєстраційної комісії

```python
    def register_user(self, user: User) -> Credentials:
        if user is None:
            raise ValueError("user cannot be None")

        print(f"|LOG| [registration] {{{user.get_id()}}} ", end = '')

        if not user.get_is_eligible_voter():
            print(f"{STATUS_ICON_FAILURE} (User is not able to vote)")
            return

        self._voter_index += 1

        if user.get_passport_id() in self._voters_data.values():
            print(f"{STATUS_ICON_FAILURE} (User has already been registered)")
            return

        username: str = self._generate_username()
        password: str = self._generate_password()

        print(STATUS_ICON_SUCCESS)

        id: UUID = list(self._voters_data.keys())[self._voter_index]
        self._voters_data[id] = user.get_passport_id()
        auth_token: AuthToken = self._voters_tokens[id]
        self._voters.append(Voter(id, username, hash(password)))
        return Credentials(username, password, auth_token)

    def _generate_username(self) -> str:
        USERNAME_CHARACTERS_COUNT: int = 10
        username = self._generate_random_string(USERNAME_CHARACTERS_COUNT)

        while username in self._usernames:
            username = self._generate_random_string(USERNAME_CHARACTERS_COUNT)

        self._usernames.add(username)

        return username

    def _generate_password(self) -> str:
        PASSWORD_CHARACTERS_COUNT: int = 10
        return self._generate_random_string(PASSWORD_CHARACTERS_COUNT)

    def _generate_random_string(self, length: int) -> str:
        return ''.join(random.choice(RegistrationCommissionController._CREDENTIALS_CONTENT) for _ in range(length))
```

Рисунок 1.2 – Продовження класу реєстраційної комісії

```python
class ElectionCommissionController:

    def __init__(self, voters_ids: list[UUID], candidates: list[Candidate]) -> 'ElectionCommissionController':
        self._voted_voters: set[bytes] = set()
        self._bbs_keys: bbs.KeysPair = bbs.new_keys()
        self._voters_tokens: dict[UUID, AuthToken] = dict()
        self._results = {candidate.get_id(): 0 for candidate in candidates}
        self._voters_elgamal_keys: dict[elgamal.PublicKey, elgamal.PrivateKey] = dict()

        for voter_id in voters_ids:
            elgamal_public_key, elgamal_private_key = elgamal.Elgamal.newkeys(32)    "newkeys": Unknown word.
            self._voters_elgamal_keys[elgamal_public_key] = elgamal_private_key
            self._voters_tokens[voter_id] = AuthToken(voter_id, elgamal_public_key, self._bbs_keys)

    def get_voters_tokens(self) -> list[AuthToken]:
        return list(self._voters_tokens.values())
```

Рисунок 1.3 – Код класу виборчої комісії

```python
def register_vote(self, vote: Vote, elgamal_public_key: elgamal.PublicKey) -> None:
    if vote is None:
        raise ValueError("vote cannot be None")

    print(f"|LOG| [vote] {{{vote.get_id()}}} ", end = '')

    if elgamal_public_key not in self._voters_elgamal_keys.keys():
        print(f"{STATUS_ICON_FAILURE} (Unknown voter)")
        return

    decrypted_message: bytearray = elgamal.Elgamal.decrypt(vote.get_payload(), self._voters_elgamal_keys[elgamal_public_key])
    message_slices: list[bytearray] = decrypted_message.split(Vote.DELIMITER_BYTE)

    if len(message_slices) != Vote.CHUNKS_COUNT:
        print(f"{STATUS_ICON_FAILURE} (Invalid message)")
        return

    seed: int = int.from_bytes(message_slices[Vote.CHUNK_SEED_INDEX])
    voter_id: bytes = bytes(message_slices[Vote.CHUNK_VOTER_ID_INDEX])

    if voter_id in self._voted_voters:
        print(f"{STATUS_ICON_FAILURE} (Voter has already voted)")
        return False

    candidate_id: int = bbs.decrypt(int.from_bytes(message_slices[Vote.CHUNK_CANDIDATE_ID_INDEX]), seed, self._bbs_keys.get_public_key())

    if candidate_id not in self._results.keys():
        print(f"{STATUS_ICON_FAILURE} (Invalid candidate)")
        return False

    print(STATUS_ICON_SUCCESS)
    self._results[candidate_id] += 1
    self._voted_voters.add(voter_id)

def print_results(self) -> None:
    print("\n|RESULTS|\n")

    for candidate_id, vote_count in sorted(self._results.items(), key = lambda item: item[1], reverse = True):
        print(f"|REPORT| #{candidate_id}: {vote_count} votes")
```

Рисунок 1.4 – Продовження класу виборчої комісії

Ще одним важливим класом є реалізація сервісу проведення електронного голосування, який має методи для авторизації користувачів та голосування. Код відповідного класу можна побачити на рисунках 1.5 – 1.6.

```python
class VotingHandlerServie:

    def __init__(self, voters: list[Voter], election_commission: ElectionCommissionController) -> 'VotingHandlerServie':
        self._active_session: Voter = None
        self._voters: list[Voter] = voters
        self._active_session_index: int = 0
        self._election_commission: ElectionCommissionController = election_commission

    def login(self, username: str, password: str) -> None:
        print(f"|LOG| [auth] {{{self._active_session_index}}} ", end = '')

        for voter in self._voters:
            if voter.get_username() == username:
                self._active_session = voter
                break

        if self._active_session is None:
            print(f"{STATUS_ICON_FAILURE} (Invalid username)")
            return

        if hash(password) != self._active_session.get_password_hash():
            print(f"{STATUS_ICON_FAILURE} (Invalid password)")
            self._active_session = None
            return

        print(STATUS_ICON_SUCCESS)
        self._active_session_index += 1
```

Рисунок 1.5 – Код класу сервісу проведення виборів

```python
def vote(self, candidate: Candidate, auth_token: AuthToken) -> None:
    if candidate is None:
        raise ValueError("candidate cannot be None")

    if auth_token is None:
        raise ValueError("auth_token cannot be None")

    if self._active_session is None:
        return

    if self._active_session is None:
        print(f"{STATUS_ICON_FAILURE} (Invalid user)")
        return

    if auth_token.get_voter_id() != self._active_session.get_voter_id():
        print(f"{STATUS_ICON_FAILURE} (Invalid token)")
        return

    self._active_session = None
    vote: Vote = self._pack_vote_payload(candidate.get_id(), auth_token)
    self._election_commission.register_vote(vote, auth_token.get_elgamal_public_key())

def _pack_vote_payload(self, candidate_id: int, auth_token: AuthToken) -> Vote:
    encrypted_candidate_id: int = bbs.encrypt(candidate_id, auth_token.get_bbs_keys().get_seed(), auth_token.get_bbs_keys().get_public_key())
    payload: bytes = encrypted_candidate_id.to_bytes() + Vote.DELIMITER_BYTE + auth_token.get_bbs_keys().get_seed().to_bytes() + Vote.DELIMITER_BYTE + auth_token.get_voter_id().bytes
    encrypted_payload: elgamal.CipherText = elgamal.Elgamal.encrypt(payload, auth_token.get_elgamal_public_key())
    return Vote(encrypted_payload)
```

Рисунок 1.6 – Продовження класу сервісу проведення виборів

Бюлетень представлений за допомогою класу Vote та зберігає не лише айді кандидата за якого був відданий голос, а й форму-представлення повідомлення у байтах, відповідні зміщення за якими можна знайти необхідні поля для виборчої комісії.

```python
class Vote:

    DELIMITER_BYTE: bytes = b'\xFF'

    CHUNKS_COUNT: int = 3
    CHUNK_CANDIDATE_ID_INDEX: int = 0
    CHUNK_SEED_INDEX: int = 1
    CHUNK_VOTER_ID_INDEX: int = 2

    _next_id: int = -1

    def __init__(self, payload: elgamal.CipherText) -> 'Vote':
        Vote._next_id += 1
        self._payload = payload
        self._id = Vote._next_id

    def get_id(self) -> int:
        return self._id

    def get_payload(self) -> elgamal.CipherText:
        return self._payload
```

Рисунок 1.7 – Код класу бюлетеня

```python
class KeysPair:
    def __init__(self, public_key: int, private_key: tuple[int, int], seed: int):
        self._seed: int = seed
        self._public_key: int = public_key
        self._private_key: tuple[int, int] = private_key

    def get_seed(self) -> int:
        return self._seed

    def get_public_key(self) -> int:
        return self._public_key

    def get_private_key(self) -> tuple[int, int]:
        return self._private_key

def new_keys() -> KeysPair:
    p: int = 11
    q: int = 23
    seed: int = 7
    return KeysPair((p * q), (p, q), seed)

def encrypt(message: int, seed: int, public_key: int) -> int:
    random_bits = _generate_bits(message.bit_length(), seed, public_key)
    return message ^ random_bits

def decrypt(ciphertext: int, seed: int, public_key: int) -> int:
    random_bits = _generate_bits(ciphertext.bit_length(), seed, public_key)
    return ciphertext ^ random_bits

def _generate_bits(length: int, seed: int, public_key: int) -> int:
    result: int = 0
    x: int = (seed ** 2) % public_key

    for _ in range(length):
        x = (x ** 2) % public_key
        result = (result << 1) | (x & 1)

    return result
```

Рисунок 1.8 – Код реалізації Blum Blum Shub

Реєстраційне комісія видає кожному користувачу відповідний токен автентифікації, який зберігає айді виборця, публічний ключ Ель-Гамаля, який генерується власний для кожного виборця і пару bbs ключів

```python
class AuthToken:

    def __init__(self, voter_id: UUID, elgamal_public_key: elgamal.PublicKey, bbs_keys: bbs.KeysPair) -> 'AuthToken':
        self._bbs_keys = bbs_keys
        self._voter_id = voter_id
        self._elgamal_public_key = elgamal_public_key

    def get_voter_id(self) -> UUID:
        return self._voter_id

    def get_bbs_keys(self) -> bbs.KeysPair:
        return self._bbs_keys

    def get_elgamal_public_key(self) -> elgamal.PublicKey:
        return self._elgamal_public_key
```

Рисунок 1.9 – Код класу токену аунтефікації

```python
if __name__ == '__main__':
    candidate_1 = Candidate()
    candidate_2 = Candidate()

    user_1 = User(True)
    user_2 = User(True)
    user_3 = User(True)
    user_4 = User(False)

    # setup
    registration_commission = RegistrationCommissionController(User.get_users_total_count())
    election_commission = ElectionCommissionController(registration_commission.get_voters_ids(), [candidate_1, candidate_2])
    registration_commission.setup_voters_tokens(election_commission.get_voters_tokens())
    voting_handler = VotingHandlerServie(registration_commission.get_voters(), election_commission)

    print("\n|REGISTRATION|\n")

    # OK
    credentials_1: Credentials = registration_commission.register_user(user_1)

    # 2nd reg attempt
    _ = registration_commission.register_user(user_1)

    # OK
    credentials_2: Credentials = registration_commission.register_user(user_2)

    # OK
    credentials_3: Credentials = registration_commission.register_user(user_3)

    # Not eligible voter
    _ = registration_commission.register_user(user_4)
```

Рисунок 1.10 – Код верифікації роботи протоколу

```python
print("\n|VOTING|\n")

# OK
voting_handler.login(credentials_1.get_username(), credentials_1.get_password())
voting_handler.vote(candidate_1, credentials_1.get_auth_token())

print('\n', end = '')

# 2nd voting attempt
voting_handler.login(credentials_1.get_username(), credentials_1.get_password())
voting_handler.vote(candidate_1, credentials_1.get_auth_token())

print('\n', end = '')

# Invalid username
voting_handler.login("TEST_USERNAME", credentials_2.get_password())
voting_handler.vote(candidate_1, credentials_2.get_auth_token())

print('\n', end = '')

# Invalid password
voting_handler.login(credentials_2.get_username(), "TEST_PASSWORD")
voting_handler.vote(candidate_1, credentials_2.get_auth_token())

print('\n', end = '')

# OK
voting_handler.login(credentials_2.get_username(), credentials_2.get_password())
voting_handler.vote(candidate_1, credentials_2.get_auth_token())

print('\n', end = '')

# Invalid candidate
voting_handler.login(credentials_3.get_username(), credentials_3.get_password())
voting_handler.vote(Candidate(), credentials_3.get_auth_token())

print('\n', end = '')
```

Рисунок 1.11 – Продовження коду верифікації роботи протоколу

```python
# Invalid vote payload (structure)

from models import Vote
from utilities import bbs
from elgamal import elgamal

encrypted_candidate_id: int = bbs.encrypt(candidate_1.get_id(), credentials_3.get_auth_token().get_bbs_keys().get_seed(), credentials_3.get_auth_token().get_bbs_keys().get_public_key())
payload: bytes = encrypted_candidate_id.to_bytes() + Vote.DELIMITER_BYTE + credentials_3.get_auth_token().get_bbs_keys().get_seed().to_bytes()
encrypted_payload: elgamal.CipherText = elgamal.Elgamal.encrypt(payload, credentials_3.get_auth_token().get_elgamal_public_key())

election_commission.register_vote(Vote(encrypted_payload), credentials_3.get_auth_token().get_elgamal_public_key())

print('\n', end = '')

# OK
voting_handler.login(credentials_3.get_username(), credentials_3.get_password())
voting_handler.vote(candidate_2, credentials_3.get_auth_token())

print('\n', end = '')

# Unknown voter

elgamal_public_key, elgamal_private_key = elgamal.Elgamal.newkeys(32)    "newkeys": Unknown word.
election_commission.register_vote(Vote(encrypted_payload), elgamal_public_key)

election_commission.print_results()
```

Рисунок 1.12 – Продовження коду верифікації протоколу

```
|REGISTRATION|

|LOG| [registration] {1} ✅
|LOG| [registration] {1} ❌ (User has already been registered)
|LOG| [registration] {2} ✅
|LOG| [registration] {3} ✅
|LOG| [registration] {4} ❌ (User is not able to vote)

|VOTING|

|LOG| [auth] {0} ✅
|LOG| [vote] {0} ✅

|LOG| [auth] {1} ✅
|LOG| [vote] {1} ❌ (Voter has already voted)

|LOG| [auth] {2} ❌ (Invalid username)

|LOG| [auth] {2} ❌ (Invalid password)

|LOG| [auth] {2} ✅
|LOG| [vote] {2} ✅

|LOG| [auth] {3} ✅
|LOG| [vote] {3} ❌ (Invalid candidate)

|LOG| [vote] {4} ❌ (Invalid message)

|LOG| [auth] {4} ✅
|LOG| [vote] {5} ✅

|LOG| [vote] {6} ❌ (Unknown voter)

|RESULTS|

|REPORT| #1: 2 votes
|REPORT| #2: 1 votes
```

Рисунок 1.13 – Вивід логів симуляції голосування

Дослідження протоколу:

1. Перевірити чи можуть голосувати ті, хто не має на це права.

   Ні, не можуть, оскільки реєстраційна комісія перед реєстрацією перевіряє чи має користувач право голосу. А виборча комісія не може приймати голос від незареєстрованих користувачів. Звісно, що можливі спроби шахраювання з боку комісій.

2. Перевірити чи може виборець голосувати кілька разів.

   Ні, не може, оскільки у повідомленні разом з бюлетенем передається айді виборця і перед зарахуванням голосу перевіряється чи не голосував цей користувач до цього. Також не можна відкидати варіант з шахраюванням з боку комісій.

3. Чи може хтось (інший виборець, ВК, стороння людина) дізнатися за кого проголосували інші виборці?

   Якщо перехопити такий бюлетень, то розшифрувати його можна лише з приватним ключем Ель-Гамаля, а також ключами BBS. Виборча комісія загалом може дізнатися хто за кого проголосував, але певний рівень анонімізації досягається завдяки ідентифікації виборця за айді, що приховує справжню особистість.

4. Перевірити чи може інший виборець чи стороння людина проголосувати замість іншого зареєстрованого виборця.

   Це можливо у випадку отримання доступу до ідентифікаційного токену іншого користувача та якщо проголосувати раніше за цього виборця.

5. Чи може хтось (інший виборець, ВК, стороння людина) таємно змінити голос в бюлетені?

   Інший виборець не може цього зробити у випадку перехоплення, оскільки для цього потрібно розшифрувати повідомлення та знати його структуру. Проте, виборча комісія може легко змінити голос виборця, або віддати його іншому кандидату, оскільки не публікуються бюлетені.

6. Чи може виборець перевірити, що його голос врахований при підведенні кінцевих підсумків?

   Ні, оскільки не публікуються фінальний список бюлетенів.

Висновок:

У результаті виконання лабораторної роботи було змодельовано протокол електронного голосування без підтвердження, який забезпечує базові вимоги до безпеки та конфіденційності голосування. В процесі моделювання реалізовано систему реєстрації виборців, захист даних за допомогою методів Ель-Гамаля для шифрування повідомлень і BBS-генератора для створення бюлетенів, а також протестовано різні сценарії порушення протоколу.

Аналіз протоколу виявив як його сильні, так і слабкі сторони. Зокрема, протокол успішно блокує доступ до голосування незареєстрованих виборців та сторонніх осіб, проте не гарантує абсолютної анонімності голосування, оскільки виборча комісія має доступ до зашифрованих бюлетенів і може ідентифікувати виборців через їхній ID. Важливо зазначити, що виборці не мають можливості підтвердити врахування своїх голосів, що може знижувати рівень довіри до системи. Також залишається ризик маніпуляцій з боку виборчої комісії, оскільки вона має доступ до бюлетенів і може змінювати результати.

Загалом, виконана робота демонструє базові реальні принципи реалізації електронного голосування та підкреслює ключові аспекти, які потребують подальшого вдосконалення. Для підвищення надійності системи варто впровадити додаткові механізми прозорості, такі як публікація зашифрованих бюлетенів для аудиту, або використання криптографічних методів, що дозволяють перевірити врахування голосів без порушення їхньої конфіденційності.