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 № 2 з дисципліни
«Протоколи й алгоритми електронного голосування»

“**Протокол Е-голосування зі сліпими підписами**”

Виконав(ла) *ІП-13 Бабіч Денис* _____
(шифр, прізвище, ім'я, по батькові)

Перевірив *Нестерук А. О.* _____
(посада, прізвище, ім'я, по батькові)

Київ 2024

ЛАБОРАТОРНА РОБОТА № 2

Тема роботи: Протокол Е-голосування зі сліпими підписами.

Мета роботи: Дослідити протокол Е-голосування зі сліпими підписами.

Основне завдання: Змоделювати протокол Е-голосування зі сліпими підписами будь-якою мовою програмування та провести його дослідження. Для кодування повідомлень використовувати шифрування RSA, для реалізації ЕЦП використовувати алгоритм RSA.

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
4. кого проголосували інші виборці?
5. Перевірити чи може інший виборець чи стороння людина проголосувати замість іншого зареєстрованого виборця.
6. Чи може хтось (інший виборець, ВК, стороння людина) таємно змінити голос в бюлетені?
7. Чи може виборець перевірити, що його голос врахований при підведенні кінцевих підсумків?

Виконання завдання:

Для реалізації наборів повідомлення були створені класи VotePool, який представляє собою набір повідомлень за кожного кандидата та VotePoolsCluster, який представляє колекцію повідомлень VotePool.

```
class VotePool:

    def __init__(self, votes: list[VotePayload | SignedVote]) → None:
        self._votes = votes
        self._size = len(votes)

    def get_size(self) → int:
        return self._size

    def get_vote_payload(self, index: int) → VotePayload | SignedVote:
        if index < 0 or index ⩾ self._size:
            raise IndexError("Invalid index argument")
        return self._votes[index]
```

Рисунок 1.1 – Код класу VotePool

```
class VotePoolsCluster:

    def __init__(self, vote_pool: VotePool, cluster_size: int) → None:
        self._size = cluster_size
        self._vote_pools = [copy.deepcopy(vote_pool) for _ in range(0, cluster_size)]

    def get_size(self) → int:
        return self._size

    def get_vote_pool(self, index: int) → VotePool:
        if index < 0 or index ⩾ self._size:
            raise IndexError("Invalid index argument")
        return self._vote_pools[index]
```

Рисунок 1.2 – Код класу VotePoolsCluster

Сам клас, який представляє собою голос виборця називається VotePayload, він має методи для перемикання маскування та шифрування, код класу наведений на рисунку 1.3. Також клас SignedVote представляє собою підписаний голос, код наведений на рисунку 1.4.

```python
class VotePayload(IEncryptorRSA, IMask):

    def __init__(self, candidate_id: int, uuid: int, mask_key: bytes) -> None:
        self._uuid = uuid
        self._is_masked = False
        self._is_encrypted = False
        self._cipher = Fernet(mask_key)
        self._candidate_id = candidate_id

    def get_uuid(self) -> int:
        return self._uuid

    def get_candidate_id(self) -> int:
        return self._candidate_id

    def get_is_masked(self) -> bool:
        return self._is_masked

    def get_is_encrypted(self) -> bool:
        return self._is_encrypted

    def mask(self) -> None:
        self._is_masked = True
        self._uuid = self._cipher.encrypt(self._uuid.to_bytes(16))

    def unmask(self) -> None:
        self._is_masked = False
        self._uuid = int.from_bytes((self._cipher.decrypt(self._uuid)))

    def encrypt(self, public_key: rsa.PublicKey) -> None:
        self._is_encrypted = True
        self._uuid = rsa.encrypt(self._uuid.to_bytes(16), public_key)
        self._candidate_id = rsa.encrypt(self._candidate_id.to_bytes(), public_key)

    def decrypt(self, private_key: rsa.PrivateKey) -> None:
        self._is_encrypted = False
        self._uuid = int.from_bytes(rsa.decrypt(self._uuid, private_key))
        self._candidate_id = int.from_bytes(rsa.decrypt(self._candidate_id, private_key))
```

Рисунок 1.3 – Код класу VotePayload

```python
class SignedVote(IEncryptorRSA):

    def __init__(self, vote_payload: VotePayload, signature: bytes) -> None:
        self._is_encrypted = False
        self._signature = signature
        self._vote_payload = vote_payload

    def get_signature(self) -> bytes:
        return self._signature

    def get_is_encrypted(self) -> bool:
        return self._is_encrypted

    def get_vote_payload(self) -> VotePayload:
        return self._vote_payload

    def encrypt(self, public_key: rsa.PublicKey) -> None:
        self._is_encrypted = True
        self._vote_payload.encrypt(public_key)

    def decrypt(self, private_key: rsa.PrivateKey) -> None:
        self._is_encrypted = False
        self._vote_payload.decrypt(private_key)
```

Рисунок 1.4 – Код класу SignedVote

Клас виборця представлений класом VoterController, який зображений на рисунку 1.5, де є важливі методи, де create_vote_pools_cluster відповідає за створення кластеру повідомлень, а process_commission_response за обробку зворотнього повідомлення від ВК.

```python
class VoterController:

    def __init__(self, is_able_to_vote: bool) -> None:
        self._id = uuid.uuid4().int
        self._mask_key = Fernet.generate_key()
        self._is_able_to_vote = is_able_to_vote

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

    def get_mask_key(self) -> bytes:
        return self._mask_key

    def get_is_able_to_vote(self) -> bool:
        return self._is_able_to_vote

    def create_vote_pools_cluster(self, candidates: list[Candidate], size: int) -> VotePoolsCluster:
        vote_payloads = [VotePayload(candidate.get_id(), self._id, self._mask_key) for candidate in candidates]
        [vote_payload.mask() for vote_payload in vote_payloads]
        cluster = VotePoolsCluster(VotePool(vote_payloads), size)
        return cluster

    def process_commission_response(self, signed_vote_pool: VotePool, candidate: Candidate, comission_public_key: rsa.PublicKey) -> None:
        if signed_vote_pool is None:
            return None

        signed_vote = signed_vote_pool.get_vote_payload(candidate.get_id() - 1)

        try:
            rsa.verify(f"{hash(signed_vote.get_vote_payload())}".encode(), signed_vote.get_signature(), comission_public_key)
        except rsa.VerificationError:
            print(f"{Fore.RED}FAILURE{Style.RESET_ALL} | Invalid signature")
            return

        signed_vote.get_vote_payload().unmask()
        signed_vote.encrypt(comission_public_key)
        return signed_vote
```

Рисунок 1.5 – Код класу VoterController

Клас виборчої комісії має кілька цікавих полів, зокрема список учасників, які зареєстровані на вибори, журнал виборців та статус їх голосування, список результатів голосування.

```python
def __init__(self, candidates: list[Candidate], voters: list[VoterController]) -> None:
    self._public_results = {}
    self._candidates = candidates
    (self._public_key, self._private_key) = rsa.newkeys(nbits = 512)
    self._results = {candidate.get_id(): 0 for candidate in candidates}
    self._voters_registry = {voter.get_id(): VoterStatus.IDLE for voter in voters}
```

Рисунок 1.6 – поля класу CommissionContoller

Алгоритм валідації голосувальника та кластеру голосів зображено на рисунку 1.7.

```python
def process(self, voter: VoterController, vote_pools_cluster: VotePoolsCluster) -> list[SignedVote]:
    if voter is None:
        raise ValueError("voter cannot be None")

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

    print(f"{Fore.YELLOW}PROCESSING{Style.RESET_ALL} #{voter.get_id()} | ", end = '')

    if not self._verify_voter(voter):
        return None

    is_valid_cluster, out_vote_pool = self._validate_vote_pools_cluster(vote_pools_cluster)

    if not is_valid_cluster:
        return None

    self._voters_registry[voter.get_id()] = VoterStatus.ECHOED
    signed_vote_pool = self._sign_vote_pool(out_vote_pool)
    return signed_vote_pool
```

Рисунок 1.7 – Алгоритм верифікації користувача та кластеру повідомлень

Алгоритм обміну повідомленнями між виборцем та комісією зображено на рисунку 1.8, де голосувальник створює пакет повідомлень, комісія виконує попередню обробку пакету та виборця, повертає підписаний сліпим підписом набір голосів, користувач обирає бажаний голос і відправляє його назад.

```python
vote_pools_cluster = voter_1.create_vote_pools_cluster(candidates, CommissionController.VOTE_POOLS_CLUSTER_SIZE)
sgined_vote_pool = commission.process(voter_1, vote_pools_cluster)
final_vote = voter_1.process_commission_response(sgined_vote_pool, candidate_1, commission.get_public_key())
commission.register_final_vote(final_vote)
```

Рисунок 1.8 – Алгоритм комунікації між голосувальником та ВК

```
voter #0 PROCESSING #78064391161845219846573288880596150781 | 1st VERIFICATION: FAILURE | Unknown voter
voter #1 PROCESSING #23754240556668748662695072402595703219 6 | 1st VERIFICATION: SUCCESS | CLUSTER VALIDATION: SUCCESS | 2nd VERIFICATION: SUCCESS | DONE
voter #2 PROCESSING #16791940394149036720763081846980198446 6 | 1st VERIFICATION: FAILURE | Voter is not able to vote
voter #3 PROCESSING #12267452889957951872867368378193802963 7 | 1st VERIFICATION: SUCCESS | CLUSTER VALIDATION: FAILURE | Invalid pool's size
voter #3 PROCESSING #12267452889957951872867368378193802963 7 | 1st VERIFICATION: SUCCESS | CLUSTER VALIDATION: FAILURE | Invalid pool's content
voter #3 PROCESSING #12267452889957951872867368378193802963 7 | 1st VERIFICATION: SUCCESS | CLUSTER VALIDATION: FAILURE | Invalid voter
voter #4 PROCESSING #22661598661625710806365820066098778594 3 | 1st VERIFICATION: SUCCESS | CLUSTER VALIDATION: SUCCESS | 2nd VERIFICATION: SUCCESS | DONE
voter #4 PROCESSING #22661598661625710806365820066098778594 3 | 1st VERIFICATION: FAILURE | Voter has already voted
voter #5 2nd VERIFICATION: FAILURE | Signature verification failed
voter #6 PROCESSING #45227085891768274759685447131226221167 | 1st VERIFICATION: SUCCESS | CLUSTER VALIDATION: FAILURE | Invalid cluster size
voter #9 PROCESSING #36278709946153495228559109933658271 77 | 1st VERIFICATION: SUCCESS | CLUSTER VALIDATION: SUCCESS | 2nd VERIFICATION: SUCCESS | DONE

RESULTS

23754240556668748662695072402595703219 6: 1
22661598661625710806365820066098778594 3: 2
36278709946153495228559109933658271 77: 1

Candidate #1: 2 vote(s)
Candidate #2: 1 vote(s)

WINNER is candidate #1 with 2 vote(s).
```

Рисунок 1.9 – Результат роботи симуляції

Дослідження протоколу:

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

   Ні, не можуть, оскільки відбувається попередня валідація виборця. Звісно, що не можна відкидати варіант з шахраюванням з боку виборчої комісії.

2. Перевірити чи може виборець голосувати кілька разів.

   Ні, не можуть, оскільки відбувається попередня перевірка на те, чи голосував виборець на першому етапі (відправка пакету голосів) та на другому (коли голосувальник відправляє остаточний бюлетень). Звісно, що не можна відкидати варіант з шахраюванням з боку виборчої комісії.

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

   Так, оскільки після виборів виводиться список айді бюлетенів та їх вміст, але для цього потрібно знати якому виборцю належить який айді бюлетеня. Стосовно самої комісії, то особистість виборця може бути прихована за айді бюлетеня, оскільки остаточний бюлетень присилається без прив’язки до особи виборця.

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

   Це можливо зробити, якщо вгадати айді бюлетеня іншого виборця й проголосувати до нього. Звісно, що не можна відкидати варіант з шахраюванням з боку виборчої комісії.

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

   Таємно цього зробити не вдасться, оскільки публікується остаточний список бюлетенів та їх вміст після завершення процесу голосування.

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

   Так, оскільки виводиться список бюлетенів та їх вміст після голосування.

Висновок:

У ході виконання лабораторної роботи було змодельовано та досліджено протокол електронного голосування зі сліпими підписами, з використанням алгоритму RSA для шифрування та підписання повідомлень.

Під час дослідження було встановлено, що протокол успішно запобігає голосуванню неавторизованих осіб через попередню верифікацію виборців. Виборці не можуть проголосувати більше одного разу завдяки контролю на етапі перевірки підписаного бюлетеня. Окрім того, протокол забезпечує певний рівень анонімності голосування, оскільки голоси підписуються сліпими підписами та не мають прив’язки до особи виборця. Проте є можливість ідентифікувати виборця через ID бюлетеня, якщо буде доступ до відповідних даних. Протокол захищений від спроб проголосувати замість іншого виборця, але така можливість існує за умови доступу до ID бюлетеня іншого виборця. Крім того, зміна голосу після його надсилання без відома виборця також є малоймовірною, оскільки після завершення голосування публікується список бюлетенів та їх вміст.

Таким чином, протокол забезпечує базові функції захисту, верифікації виборців та шифрування голосів, але не може повністю гарантувати абсолютну анонімність і захищеність від маніпуляцій з боку виборчої комісії або сторонніх осіб. Для реальних систем електронного голосування доцільно впроваджувати додаткові заходи безпеки для посилення захисту голосів та анонімності виборців.