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 № 1 з дисципліни

«Протоколи й алгоритми електронного голосування»

“**Простий протокол Е-голосування**”

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 № 1

Тема роботи: Простий протокол Е-голосування.

Мета роботи: Дослідити протокол простого Е-голосування.

Основне завдання: Змоделювати простий протокол Е-голосування будь-якою мовою програмування та провести його дослідження. Для кодування повідомлень використовувати метод гамування, 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, ЦВК, 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, ЦВК, стороння людина) таємно змінити голос в бюлетені?
7. Чи може виборець перевірити, що його голос врахований при підведенні кінцевих результатів?

Виконання завдання:

Сам бюлетень голосування представлений у вигляді класів VoteRecordPayload та SignedVoteRecord, де клас VoteRecordPayload представляє протокол без rsa-підпису, у якому зберігається айді кандидата, за який був відданий голос та контрольна хеш-сума на основі цього значення. Також реалізований алгоритм гамування через інтерфейс IGammaEncryptor. Повний код можна побачити на рисунку 1.1.

```python
class VoteRecordPayload(IGammaEncryptor):

    def __init__(self, candidate_id: int) -> None:
        self._is_encrypted = False
        self._candidate_id = candidate_id
        self._control_hash_sum = hash(candidate_id)

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

    def set_candidate_id(self, value: int) -> None:     # demo purposes only :)
        self._candidate_id = value

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

    def get_control_hash_sum(self) -> int:
        return self._control_hash_sum

    def toggle_gamma_encryption(self, xor_key: bytes) -> None:
        self._is_encrypted = not self._is_encrypted
        xor_key_value = int.from_bytes(xor_key)
        self._candidate_id ^= xor_key_value
        self._control_hash_sum ^= xor_key_value

    def __eq__(self, other: object) -> bool:
        if isinstance(other, VoteRecordPayload):
            return (self._is_encrypted == other._is_encrypted and
                    self._candidate_id == other._candidate_id and
                    self._control_hash_sum == other._control_hash_sum)
        return False

    def __hash__(self):
        return hash(self._candidate_id)
```

Рисунок 1.1 – Код класу VoteRecordPayload

Клас SignedVoteRecord фактично є класом-обгорткою навколо VoteRecordPayload, у якому через композицію зберігається посилання на об’єкт VoteRecordPayload і додається власне поле електронного цифрового підпису (ЕЦП). Повний код можна побачити на рисунку 1.2.

```python
class SignedVoteRecord(IGammaEncryptor):

    def __init__(self, vote_record: VoteRecordPayload, signature: bytes) -> None:
        self._is_encrypted = False
        self._signature = signature
        self._vote_record = vote_record

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

    def get_vote_record(self) -> VoteRecordPayload:
        return self._vote_record

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

    def toggle_gamma_encryption(self, xor_key: bytes) -> None:
        self._is_encrypted = not self._is_encrypted

        self._vote_record.toggle_gamma_encryption(xor_key)

        key_length = len(xor_key)
        signature_length = len(self._signature)
        self._signature = bytes([self._signature[i] ^ xor_key[i % key_length] for i in range(signature_length)])
```

Рисунок 1.2 – Повний код класу SignedVoteRecord

Виборець реалізований за допомогою класу VoterController, у якому зберігаються айді, прапорець можливості голосування, приватне поле закритого rsa-ключа, публічне поле відкритого ключа. У цього контролеру є метод vote, який створює протокол без підпису з занесеним значення кандидата та метод sign, який створює підписаний бюлетень, створюючи ЕЦП на основі значення айді кандидата, за якого виборець віддав голос. Варто також згадати про використання перечислення VoterDebugMode, який використовуватиметься під час тестування різних сценаріїв проведення голосування. Повний код класу VoterController можна побачити на рисунку 1.3.

```python
class VoterDebugMode(Enum):
    NONE = 0
    MISSING_SIGNATURE = 1
    MISSING_GAMMA_ENCRYPTION = 2
    VOTE_RECORD_SUBSTITUTION = 3

class VoterController:

    def __init__(self, id: int, is_able_to_vote: bool, debug_mode: VoterDebugMode = VoterDebugMode.NONE) -> None:
        self._id = id
        self._debug_mode = debug_mode
        self._is_able_to_vote = is_able_to_vote
        (self._public_key, self._private_key) = rsa.newkeys(nbits = 512)

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

    def get_debug_mode(self) -> VoterDebugMode:
        return self._debug_mode

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

    def get_public_key(self) -> rsa.PublicKey:
        return self._public_key

    def vote(self, candidate: Candidate) -> VoteRecordPayload:
        if candidate is None:
            raise ValueError("candidate is None.")

        return VoteRecordPayload(candidate.get_id())

    def sign(self, vote_record: VoteRecordPayload) -> SignedVoteRecord:
        if vote_record is None:
            raise ValueError("vote_record is None.")

        signature = rsa.sign(vote_record.get_candidate_id().to_bytes(), self._private_key, "SHA-256")
        return SignedVoteRecord(vote_record, signature)
```

Рисунок 1.3 – Код класу VoterController

Останнім важливим класом є CommissionController, який представляє ЦВК та має поле ключа гамування, яким шифруються бюлетені перед відправкою, словник, у який відбувається підрахунок голосів, список верифікованих виборців та 2 важливі методи, а саме process_voter та register_vote. Метод process_voter відповідає за верифікацію виборця та створення протоколу (рис. 1.4). За допомогою _verfiy_voter відбувається валідація людини, а саме перевірка чи має виборець можливість голосувати й чи не голосував виборець до цього (рис. 1.5). Метод _create_protocol відповідає за роботу над створенням бюлетеню (ініціалізація, підпис, шифрування гамування), де використовуються значення з перечислення VoterDebugMode для реалізації різних сценаріїв поведінки, а саме відсутність підпису, відсутність шифру гамування та спроба підміни голосу. Повний код методу _create_protocol можна побачити на рисунку 1.6.

```python
def process_voter(self, voter: VoterController, candidate: Candidate) -> None:
    if voter is None:
        raise ValueError("voter is None")

    if candidate is None:
        raise ValueError("candidate is None")

    print(f"Processing voter #{voter.get_id()} | ", end = '')

    if self._verify_voter(voter):
        vote_record = self._create_protocol(voter, candidate)
        self.register_vote(vote_record, voter.get_public_key())
```

Рисунок 1.4 – Код методу process_voter

```python
def _verify_voter(self, voter: VoterController) -> bool:
    if not voter.get_is_able_to_vote():
        print(f"ABORTED. Voter is not able to vote.")
        return False

    if voter.get_public_key() in self._vote_records:
        print(f"ABORTED. Voter has already voted.")
        return False

    self._verified_voters_keys.append(voter.get_public_key())
    return True
```

Рисунок 1.5 – Код методу _verify_voter

```python
def _create_protocol(self, voter: VoterController, candidate: Candidate) -> VoteRecordPayload | SignedVoteRecord:
    vote_record = voter.vote(candidate)

    if voter.get_debug_mode() is VoterDebugMode.VOTE_RECORD_SUBSTITUTION:
        FAKE_CANDIDATE_ID = 0
        vote_record.set_candidate_id(FAKE_CANDIDATE_ID)

    if voter.get_debug_mode() is not VoterDebugMode.MISSING_SIGNATURE:
        vote_record = voter.sign(vote_record)

    if voter.get_debug_mode() is not VoterDebugMode.MISSING_GAMMA_ENCRYPTION:
        vote_record.toggle_gamma_encryption(self._gamma_key)

    return vote_record
```

Рисунок 1.6 – Код методу _create_protocol

Ще одним важливим методом є register_vote, у якому бюлетень і зараховується у фінальний результат та у випадку виявлення порушення голос відкидається. Відбуваються наступні перевірки: наявність шифру гамування, чи підписаний бюлетень, чи був валідований виборець з відповідним публічним

ключем, після дешифрування алгоритму гамування перевіряється хеш-сума бюлетеня та верифікація rsa-підпису й зарахування голосу (рис. 1.7)

```
def register_vote(self, signed_vote_record: SignedVoteRecord, public_key: rsa.PublicKey) -> None:
    if signed_vote_record is None:
        raise ValueError("signed_vote_record is None")

    if not signed_vote_record.get_is_encrypted():
        print("ABORTED. Vote record must be encrypted.")
        return

    if isinstance(signed_vote_record, VoteRecordPayload):
        print("ABORTED. Vote record must be signed.")
        return

    if public_key not in self._verified_voters_keys:
        print("ABORTED. Voter must be validated.")
        return

    signed_vote_record.toggle_gamma_encryption(self._gamma_key)

    if hash(signed_vote_record.get_vote_record()) != signed_vote_record.get_vote_record().get_control_hash_sum():
        print("ABORTED. Hash control sum check failed.")
        return

    try:
        rsa.verify(int.to_bytes(signed_vote_record.get_vote_record().get_candidate_id()), signed_vote_record.get_signature(), public_key)
    except rsa.VerificationError:
        print(f"ABORTED. Signature verification failed.")
        return

    self._verified_voters_keys.remove(public_key)
    self._results[signed_vote_record.get_vote_record().get_candidate_id()] += 1
    self._vote_records[public_key] = signed_vote_record.get_vote_record().get_candidate_id()

    print(f"DONE. Registered vote for candidate #{signed_vote_record.get_vote_record().get_candidate_id()}.")
```

Рисунок 1.7 – Код методу register_vote

Вміст модуля main можна побачити на рисунку 1.8.

```
if __name__ == "__main__":
    candidate_1 = Candidate(1)
    candidate_2 = Candidate(2)
    commission = CommissionController([candidate_1, candidate_2])

    voter_1 = VoterController(1, True)
    voter_2 = VoterController(2, True)
    voter_3 = VoterController(3, True, VoterDebugMode.MISSING_SIGNATURE)
    voter_4 = VoterController(4, True)
    voter_5 = VoterController(5, True, VoterDebugMode.VOTE_RECORD_SUBSTITUTION)
    voter_6 = VoterController(6, True)
    voter_7 = VoterController(7, True, VoterDebugMode.MISSING_GAMMA_ENCRYPTION)
    voter_8 = VoterController(8, True)
    voter_9 = VoterController(9, False)

    commission.process_voter(voter_1, candidate_1)
    commission.process_voter(voter_1, candidate_1)
    commission.process_voter(voter_2, candidate_1)
    commission.process_voter(voter_3, candidate_1)
    commission.process_voter(voter_4, candidate_1)
    commission.process_voter(voter_5, candidate_1)
    commission.process_voter(voter_6, candidate_2)
    commission.process_voter(voter_7, candidate_2)
    commission.process_voter(voter_8, candidate_2)
    commission.process_voter(voter_9, candidate_2)

    voter_0 = VoterController(0, True)
    print(f"voter #{voter_0.get_id()} ", end = '')

    vote_record_0 = voter_0.vote(candidate_1)
    signed_vote_record_0 = voter_0.sign(vote_record_0)
    signed_vote_record_0.toggle_gamma_encryption(commission.get_gamma_key())
    commission.register_vote(signed_vote_record_0, voter_0.get_public_key())

    commission.print_results()
```

Рисунок 1.8 – Код у модулі main

```
Processing voter #1 | DONE. Registered vote for candidate #1.
Processing voter #1 | ABORTED. Voter has already voted.
Processing voter #2 | DONE. Registered vote for candidate #1.
Processing voter #3 | ABORTED. Vote record must be signed.
Processing voter #4 | DONE. Registered vote for candidate #1.
Processing voter #5 | ABORTED. Hash control sum check failed.
Processing voter #6 | DONE. Registered vote for candidate #2.
Processing voter #7 | ABORTED. Vote record must be encrypted.
Processing voter #8 | DONE. Registered vote for candidate #2.
Processing voter #9 | ABORTED. Voter is not able to vote.
voter #0 ABORTED. Voter must be validated.
RESULTS
Candidate #1: 3 vote(s)
Candidate #2: 2 vote(s)
WINNER is candidate #1 with 3 vote(s).
```

Рисунок 1.9 – Отриманий вивід

Дослідження протоколу:

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

   Ні, не можуть, оскільки перед цим відбувається валідація у методі process_voter. Навіть у випадку, якщо цей бюлетень просто кинути без верифікації, як це робить voter_0, то голос не буде зарахований, оскільки користувач не знаходиться у списку верифікованих виборців.

2. Перевірити чи може виборець голосувати кілька разів.

   Ні, не може, оскільки зберігається список відкритих ключів виборців, які проголосували і на етапі валідації нового відбувається його валідація завдяки перевірці списку на наявність ключа поточного виборця.

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

   Оскільки це простий алгоритм голосування, то ЦВК знає за кого проголосував кожен виборець за замовчуванням. Та й загалом всім відомо ключ шифрування гамування ЦВК, тому у випадку перехоплення такого бюлетеня він може бути легко розсекречений.

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

   Ні, оскільки кожен бюлетень підписується електронним підписом, що гарантує приналежність бюлетеня до людини, яка його підписала.

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

   У ЦВК є повнота прав над бюлетенем у моменті, коли він потрапив до неї, тому голос може бути перенаправлений за іншого кандидата. Проте, бюлетень неможливо змінити у випадку перехоплення, оскільки не буде співпадати хеш-сума повідомлення на етапі валідації бюлетеня.

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

   Ця функція не передбачена простим протоколом голосування. Проте, її досить просто реалізувати шляхом перевірки наявності відкритого ключа користувача у списку ключів виборців, які проголосували.

Висновок:

У ході виконання лабораторної роботи було змодельовано та досліджено простий протокол електронного голосування. Основним завданням було реалізувати модель голосування з використанням алгоритмів RSA для електронного підпису та гамування для шифрування повідомлень, а також провести дослідження різних сценаріїв, що можуть виникати в процесі голосування.

Під час дослідження було встановлено, що протокол надійно перевіряє, чи має виборець право голосувати, а користувачі, які не пройшли верифікацію, не можуть подати свій голос. Також кожен виборець може проголосувати лише один раз, і це забезпечується перевіркою відкритого ключа. Протокол захищений від можливості проголосувати замість іншого виборця, оскільки кожен бюлетень підписується унікальним електронним підписом виборця і голос неможливо змінити без порушення хеш-суми, що гарантує захист від несанкціонованої модифікації при перехопленні бюлетеня. Проте, ЦВК має можливість змінити бюлетень після його дешифрування, що є загальною вразливістю цього протоколу.

Таким чином, даний протокол має базові функції захисту й верифікації, однак не може повністю гарантувати анонімність та захищеність голосів від ЦВК. Це дозволяє зробити висновок, що для реальних систем електронного голосування слід впроваджувати додаткові заходи безпеки, які захистять як таємницю голосування, так і результат голосування від втручання сторонніх осіб чи організацій.