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 № 5 з дисципліни

«Протоколи й алгоритми електронного голосування»

“**Протокол Е-голосування з розділенням комісії на незалежні частини**”

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 № 5

Тема роботи: Протокол Е-голосування з розділенням комісії на незалежні частини.

Мета роботи: Дослідити протокол Е-голосування з розділенням комісії на незалежні частини.

Основне завдання:

Змоделювати протокол Е-голосування з розділенням комісії на незалежні частини будь-якою мовою програмування та провести його дослідження. Для кодування повідомлень використовувати метод RSA, для реалізації ЕЦП використовувати алгоритм DSA.

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 виборчих комісій, а саме центральної виборчої комісії, яка відповідає за реєстрацію користувачів, підбиття фінальних підсумків, а також звичайної виборчої комісії, яка займається прийомом голосів. Обидва класи зберігають проміжні та фінальні результати голосування у асоційованому масиві, де з кожним айді виборця асоціюється його голос (ще зашифрований, або вже ні). Також у центрального класу є метод для фінального об'єднання голосів після завершення процедури голосування. Код класу центральної виборчої комісії можна побачити на рисунках 1.1 – 1.4.

```
class CentralCommissionController:

    def __init__(self) -> 'CentralCommissionController':
        self._voters: dict[User, int] = dict()
        self._results: dict[int, int] = dict()
        self._candidates: dict[User, int] = dict()
        self._rsa_public_key, self._rsa_private_key = rsa.generate_keys(1024)
        self._election_commissions: list[ElectionCommissionController] = list()

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

    def get_candidates(self) -> list[int]:
        return list(self._candidates.values())

    def get_rsa_public_key(self) -> rsa.PublicKey:
        return self._rsa_public_key

    def get_election_commissions(self) -> list[ElectionCommissionController]:
        return self._election_commissions

    def init_election_commissions(self, election_commissions_count: int) -> list[ElectionCommissionController]:
        if election_commissions_count <= 1:
            raise ValueError("Invalid value of election_commission_count argument.")

        for i in range(election_commissions_count):
            self._election_commissions.append(ElectionCommissionController(i, self._voters.values()))

        return self._election_commissions
```

Рисунок 1.1 – Код класу центральної виборчої комісії

```
def merge_votes(self, commissions_results: list[PartialVote]) -> None:
    if commissions_results is None:
        raise ValueError("commission_results cannot be None.")

    print("\nMERGING\n")

    for vote in commissions_results:
        print(f"{vote.get_voter_id()}: ", end = '')

        if vote.get_voter_id() not in self._voters.values():
            print(f"{STATUS_ICON_REJECTED} (Invalid voter)")
            continue

        voter_votes_count: int = sum(1 for vote_payload in commissions_results if vote_payload.get_voter_id() == vote.get_voter_id())

        if voter_votes_count != len(self._election_commissions):
            print(f"{STATUS_ICON_REJECTED} (Voting procedure violation)")
            continue

        self._results[vote.get_voter_id()] = int(vote.get_partial_candidate_id()) * (int(self._results[vote.get_voter_id()]) if vote.get_voter_id() in self._results.keys() else 1)
        print(STATUS_ICON_APPROVED)

def decrypt_votes(self) -> None:
    print("\nDECRYPTION\n")

    results_copy: dict[int, int] = self._results.copy()

    for voter_id, vote in results_copy.items():
        print(f"{voter_id}: ", end = '')

        self._results[voter_id] = rsa.decrypt(vote, self._rsa_private_key)

        if self._results[voter_id] not in self._candidates.values():
            print(f"{STATUS_ICON_REJECTED} (Decryption failed)")
            self._results.pop(voter_id)
            continue

        print(STATUS_ICON_APPROVED)
```

Рисунок 1.2 – Продовження коду класу центральної виборчої комісії

```python
def print_results(self) -> None:
    print("\nFINAL VOTES\n")

    for voter_id, candidate_id in self._results.items():
        print(f"#{voter_id}: {candidate_id}")

    print("\nRESULTS\n")

    results = Counter(self._results.values())

    for candidate_id, vote_count in results.items():
        print(f"#{candidate_id}: {vote_count} votes")

def register_voter(self, user: User) -> VoterController:
    if user is None:
        raise ValueError("user cannot be None")

    print(f"VOTER REGISTRATION #{user.get_id()}: ", end = '')

    if not user.get_is_eligible_voter():
        print(f"{STATUS_ICON_FAILURE} (User is not able to vote)")
        return None

    if user in self._voters.keys():
        print(f"{STATUS_ICON_FAILURE} (User has been already registered as voter)")
        return None

    voter_controller = VoterController((len(self._voters) + 1))
    print(f"{STATUS_ICON_SUCCESS} ({voter_controller.get_id()})")
    self._voters[user] = voter_controller.get_id()
    return voter_controller
```

Рисунок 1.3 – Продовження коду класу центральної виборчої комісії

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

    print(f"CANDIDATE REGISTRATION #{user.get_id()}: ", end = '')

    if not user.get_is_eligible_candidate():
        print(f"{STATUS_ICON_FAILURE} (User is not able to be a candidate)")
        return None

    if user in self._candidates.keys():
        print(f"{STATUS_ICON_FAILURE} (User has been already registered)")
        return None

    candidate = Candidate(self._generate_candidate_id())
    self._candidates[user] = candidate.get_id()
    print(f"{STATUS_ICON_SUCCESS} ({candidate.get_id()})")
    return candidate

def _generate_candidate_id(self) -> int:
    CANDIDATE_ID_LOWER_BOUND: int = 2
    CANDIDATE_ID_UPPER_BOUND: int = 10

    number: int = 0
    factor1: int = 0
    factor2: int = 0

    while True:
        factor1 = random.randint(CANDIDATE_ID_LOWER_BOUND, CANDIDATE_ID_UPPER_BOUND)
        factor2 = random.randint(CANDIDATE_ID_LOWER_BOUND, CANDIDATE_ID_UPPER_BOUND)
        number = factor1 * factor2

        if number not in self._candidates.values():
            break

    return number
```

Рисунок 1.4 – Код генерації випадкового айді кандидата таким чином, щоб він завжди мав бодай 1 пару дільників

Код логіки розділення числа на пари множників наведено на рисунку 1.5.

```python
def get_divisors(number: int) -> list[int]:
    divisors = set()

    for i in range(1, int(number ** 0.5) + 1):
        if number % i == 0:
            divisors.add(i)
            divisors.add(number // i)

    return sorted(divisors)

def get_factors_pairs(number: int, is_prime_factor_allowed: bool) -> list[tuple[int, int]]:
    factor: int = 0
    divisors: list[int] = get_divisors(number)
    factors_pairs: list[tuple[int, int]] = list()

    for divisor in divisors:
        factor = number // divisor

        if divisor * factor != number:
            continue

        if not is_prime_factor_allowed and (divisor == number or factor == number):
            continue

        factors_pairs.append((divisor, factor))

    return factors_pairs
```

Рисунок 1.5 – Код знаходження дільників числа

Код класу локальної виборчої комісії наведено на рисунках 1.6 – 1.8.

```python
class ElectionCommissionController:

    def __init__(self, id: int, voters_ids: list[int]) -> 'ElectionCommissionController':
        if voters_ids is None:
            raise ValueError("voters_ids cannot be None")

        self._id: int = id
        self._voters_ids: list[int] = voters_ids
        self._results: list[PartialVote] = list()

    def get_votes(self) -> list[PartialVote]:
        return self._results

    def register_vote(self, vote: SignedPartialVote, dsa_public_key: dsa.DSAPublicKey) -> None:
        if vote is None:
            raise ValueError("vote cannot be None")

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

        print(f"\t[{self}] |PROCESSING| #{vote.get_partial_vote().get_voter_id()} ", end = '')

        if not self._verify_signature(vote, dsa_public_key):
            return

        if not self._verify_voter(vote.get_partial_vote().get_voter_id()):
            return

        self._results.append(vote.get_partial_vote())
        print(f"\r{STATUS_ICON_APPROVED}", end = '\n')

    def _verify_signature(self, vote: SignedPartialVote, dsa_public_key: dsa.DSAPublicKey) -> bool:
        print(f"|SIGNATURE VERIFICATION|: ", end = '')

        try:
            dsa_public_key.verify(vote.get_signature(), hash(vote.get_partial_vote().get_voter_id()).to_bytes(length = 32), hashes.SHA256())
        except Exception:
            print(f"{STATUS_ICON_FAILURE} (Invalid signature)", end = '')
            print(f"\r{STATUS_ICON_REJECTED}")
            return False

        print(f"{STATUS_ICON_SUCCESS} ", end = '')
        return True
```

Рисунок 1.6 – Код класу виборчої комісії

```python
def _verify_voter(self, voter_id: int) -> bool:
    print(f"|VOTER VERIFICATION|: ", end = '')

    if voter_id not in self._voters_ids:
        print(f"{STATUS_ICON_FAILURE} (Unknown voter)", end = '')
        print(f"\r{STATUS_ICON_REJECTED}")
        return False

    if any(vote.get_voter_id() == voter_id for vote in self._results):
        print(f"{STATUS_ICON_FAILURE} (Voter has already voted)", end = '')
        print(f"\r{STATUS_ICON_REJECTED}")
        return False

    print(f"{STATUS_ICON_SUCCESS}", end = '')
    return True

def print_results(self) -> None:
    MARKER_SIZE: int = 15

    print(f"\nINTERMEDIATE VOTES ({self._id})\n")

    for vote in self._results:
        print(f"{vote.get_voter_id()}: {str(vote.get_partial_candidate_id())[:MARKER_SIZE]}")

def __repr__(self) -> str:
    return f"ElectionCommissionController_{self._id}"
```

Рисунок 1.7 – Продовження коду класу локальної виборчої комісії

```python
# OK
votes_1 = voter_1.vote(candidate_1, central_commission.get_rsa_public_key())
election_commission_0.register_vote(votes_1[0], voter_1.get_dsa_public_key())
election_commission_1.register_vote(votes_1[1], voter_1.get_dsa_public_key())

# Second attempt
votes_1 = voter_1.vote(candidate_1, central_commission.get_rsa_public_key())
election_commission_0.register_vote(votes_1[0], voter_1.get_dsa_public_key())
election_commission_1.register_vote(votes_1[1], voter_1.get_dsa_public_key())

# Invalid encryption
rsa_public_key, _ = rsa.generate_keys(128)
votes_2 = voter_2.vote(candidate_2, rsa_public_key)
election_commission_0.register_vote(votes_2[0], voter_2.get_dsa_public_key())
election_commission_1.register_vote(votes_2[1], voter_2.get_dsa_public_key())

# Invalid signature
dsa_private_key: dsa.DSAPrivateKey = dsa.generate_private_key(key_size = 1024)
dsa_public_key: dsa.DSAPublicKey = dsa_private_key.public_key()
dsa_private_key.sign(hash(1).to_bytes(length = 32), hashes.SHA256())
votes_3 = voter_3.vote(candidate_2, central_commission.get_rsa_public_key())
election_commission_0.register_vote(votes_3[0], dsa_public_key)
election_commission_1.register_vote(votes_3[1], dsa_public_key)

# Unknown voter
unknown_voter = VoterController(0)
unknown_votes = unknown_voter.vote(candidate_1, central_commission.get_rsa_public_key())
election_commission_0.register_vote(unknown_votes[0], unknown_voter.get_dsa_public_key())
election_commission_1.register_vote(unknown_votes[1], unknown_voter.get_dsa_public_key())

# Invalid procedure # 1
votes_4 = voter_4.vote(candidate_2, central_commission.get_rsa_public_key())
election_commission_0.register_vote(votes_4[0], voter_4.get_dsa_public_key())

# OK
votes_5 = voter_5.vote(candidate_2, central_commission.get_rsa_public_key())
election_commission_0.register_vote(votes_5[0], voter_5.get_dsa_public_key())
election_commission_1.register_vote(votes_5[1], voter_5.get_dsa_public_key())

# OK
votes_6 = voter_6.vote(candidate_1, central_commission.get_rsa_public_key())
election_commission_0.register_vote(votes_6[0], voter_6.get_dsa_public_key())
election_commission_1.register_vote(votes_6[1], voter_6.get_dsa_public_key())

election_commission_0.print_results()
election_commission_1.print_results()

central_commission.merge_votes(election_commission_0.get_votes() + election_commission_1.get_votes())
central_commission.decrypt_votes()
central_commission.print_results()
```

Рисунок 1.8 – Код верифікації процедури голосування

```
REGISTRATION

CANDIDATE REGISTRATION #0: ✅ (40)
CANDIDATE REGISTRATION #1: ✅ (20)
VOTER REGISTRATION #2: ✅ (1)
VOTER REGISTRATION #3: ✅ (2)
VOTER REGISTRATION #4: ✅ (3)
VOTER REGISTRATION #5: ✅ (4)
VOTER REGISTRATION #6: ✅ (5)
VOTER REGISTRATION #7: ✅ (6)
VOTER REGISTRATION #7: ❌ (User has been already registered as voter)
VOTER REGISTRATION #8: ❌ (User is not able to vote)

VOTING

+      [ElectionCommissionController_0] |PROCESSING| #1 |SIGNATURE VERIFICATION|: ✅ |VOTER VERIFICATION|: ✅
+      [ElectionCommissionController_1] |PROCESSING| #1 |SIGNATURE VERIFICATION|: ✅ |VOTER VERIFICATION|: ✅
-      [ElectionCommissionController_0] |PROCESSING| #1 |SIGNATURE VERIFICATION|: ✅ |VOTER VERIFICATION|: ❌ (Voter has already voted)
-      [ElectionCommissionController_1] |PROCESSING| #1 |SIGNATURE VERIFICATION|: ✅ |VOTER VERIFICATION|: ❌ (Voter has already voted)
+      [ElectionCommissionController_0] |PROCESSING| #2 |SIGNATURE VERIFICATION|: ✅ |VOTER VERIFICATION|: ✅
+      [ElectionCommissionController_1] |PROCESSING| #2 |SIGNATURE VERIFICATION|: ✅ |VOTER VERIFICATION|: ✅
-      [ElectionCommissionController_0] |PROCESSING| #3 |SIGNATURE VERIFICATION|: ❌ (Invalid signature)
-      [ElectionCommissionController_1] |PROCESSING| #3 |SIGNATURE VERIFICATION|: ❌ (Invalid signature)
-      [ElectionCommissionController_0] |PROCESSING| #0 |SIGNATURE VERIFICATION|: ✅ |VOTER VERIFICATION|: ❌ (Unknown voter)
-      [ElectionCommissionController_1] |PROCESSING| #0 |SIGNATURE VERIFICATION|: ✅ |VOTER VERIFICATION|: ❌ (Unknown voter)
+      [ElectionCommissionController_0] |PROCESSING| #4 |SIGNATURE VERIFICATION|: ✅ |VOTER VERIFICATION|: ✅
+      [ElectionCommissionController_0] |PROCESSING| #5 |SIGNATURE VERIFICATION|: ✅ |VOTER VERIFICATION|: ✅
+      [ElectionCommissionController_1] |PROCESSING| #5 |SIGNATURE VERIFICATION|: ✅ |VOTER VERIFICATION|: ✅
+      [ElectionCommissionController_0] |PROCESSING| #6 |SIGNATURE VERIFICATION|: ✅ |VOTER VERIFICATION|: ✅
+      [ElectionCommissionController_1] |PROCESSING| #6 |SIGNATURE VERIFICATION|: ✅ |VOTER VERIFICATION|: ✅

INTERMEDIATE VOTES (0)

1: 100001596159413
2: 116889990090772
4: 100001596159413
5: 107555644844567
6: 784351946596399

INTERMEDIATE VOTES (1)

1: 678531056864967
2: 135120471711444
5: 784351946596399
6: 639154723859995
```

Рисунок 1.9 – Верифікація процедури голосування

```
MERGING

1: +
2: +
4: - (Voting procedure violation)
5: +
6: +
1: +
2: +
5: +
6: +

DECRYPTION

1: +
2: - (Decryption failed)
5: +
6: +

FINAL VOTES

#1: 40
#5: 20
#6: 40

RESULTS

#40: 2 votes
#20: 1 votes
```

Рисунок 1.10 – Продовження верифікації процедури голосування

Дослідження протоколу:

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

   Ні, не можуть, оскільки центральна виборча комісія на етапі реєстрації перевіряє чи має користувач право голосу. Звісно, що не можна відкидати варіанти шахраювання, бо центральна виборча комісія може дати право голосу тому, кому не потрібно, або інші виборчі комісії можуть прийняти голос від не валідного виборця (але центральна комісія додатково перевіряє цей момент на етапі злиття голосів).

2. Перевірити чи може виборець голосувати кілька разів.

   Ні, не може, оскільки виборча комісія перевіряє чи не надсилав цей виборець голос до цього, а центральна з'єднує голоси на фінальному етапі й також переконується у коректній кількості бюлетенів від кожного виборця. Також можливо шахраювання, якщо буде змова комісій.

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

   Так, бо публікуються результати у кінці проведення голосування й для цього потрібно знати айді іншого виборця.

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

   Це можливо лише у випадку, коли відомий реєстраційний айді іншого виборця, його dsa-ключі і якщо надіслати голос раніше за нього. Також можливо шахраювання, оскільки виборчі комісії можуть вписати голос замість когось, але такі дії будуть помічені ще на етапі голосування самим виборцем (не зможе надіслати голос) та після підбиття підсумків, оскільки виводиться список бюлетенів.

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

   Ні, бо айді кандидата за якого відданий голос розділяється на множники та шифрується приватним rsa-ключем, тому виборчі комісії не можуть

змінити голос (хоча й можуть помітити однакові патерни). Центральна комісія може змінити голос на етапі злиття бюлетенів, але оскільки результати публікуються, то змінити його таємно не вдасться.

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

   Так, бо значення бюлетенів публікуються зашифрованими на проміжному етапі (коли виборчі комісії публікують свої результати) та у кінці процедури голосування публікуються повноцінні розшифровані бюлетені.

Висновок:

У даній лабораторній роботі було реалізовано та досліджено протокол електронного голосування з розділенням комісії на незалежні частини. Для забезпечення захищеності повідомлень використовувався метод шифрування RSA, а цифровий підпис за алгоритмом DSA. Було створено симуляції, що включає центральну виборчу комісію, яка відповідає за реєстрацію виборців і підбиття підсумків, та локальні комісії, що приймають голоси. Така структура забезпечує поділ обов'язків і дозволяє підвищити надійність та захист процесу.

Аналіз моделі підтвердив відповідність протоколу основним вимогам безпеки. Голосувати можуть лише зареєстровані виборці, можливість багаторазового голосування виключена завдяки перевіркам на рівні комісій. Також забезпечується анонімність голосування, оскільки результати публікуються зашифрованими, що унеможливлює ідентифікацію виборця без знання його ID. Кожен голос шифрується, що захищає його від несанкціонованих змін, а виборці можуть перевірити врахування свого голосу за підсумками голосування.

Отже, змодельований протокол е-голосування демонструє високий рівень захищеності та стійкість до різних видів атак і маніпуляцій.