به نام خدا

عباس یزدان مهر - محمود چوپانی

سوال انتخابات شورای شهر - تمرین اول رمز ارز

مقدمه

در این سوال از ما خواسته شده است انتخابات شورای شهر را با استفاده از رمزگذاری هومومورفیک و نگه‌داری دیتا در بلاک‌چین شبیه‌سازی کنیم.

ساختار کلی پروژه به شکل زیر است:

- فایل blockchain.py که حاوی پیاده‌سازی دیتا استراکچر لینکد لیست دو طرفه، اکستند شده با قابلیت هش کردن دیتای نودهای قبلی برای تضمین immutability است. این پیاده‌سازی در کلاس حل تمرین انجام شده و عینا همان کد استفاده شده است.
- فایل‌های election.py و ministry.py که اینترفیسی برای ارائه قابلیت‌های رای گیری و دسترسی به بلاک‌چین است. وجود این دو پیاده‌سازی از این جهت حائز اهمیت است که دسترسی کاربران مختلف به بلاک‌چین محلی و کشوری مدیریت و همچنین از coupling بیزینس لاجیک با دیتا استراکچر جلوگیری شود.
- فایل society.py که حاوی مدل‌های بیزینسی از قبیل انواع مردم، محدوده‌های رای‌گیری و جناح‌های مختلف انتخابات است.
- فایل main.py که شبیه‌سازی انتخابات را با استفاده از cli ممکن می‌کند. در این فایل شبیه‌سازی در قالب فانکشن کال نیز موجود است.

نگاه جزئی‌تر

در پیاده‌سازی لاجیک رای‌گیری، از پترن blocktype#payload برای قرار دادن دیتا در بلاک چین استفاده شده است. زیرا بلاک‌چین علاوه بر نگه داشتن آراء، حاوی تایپ‌های دیگری از دستورات خواهد بود. در این پیاده‌سازی سعی شده از دیتا استراکچر دیگری به جز بلاک‌چین برای نگه‌داری دیتا استفاده نشود.

کلاس Election

همان‌طور که پیش‌تر بیان شد، این کلاس واسطی بین ریکوئست‌های کاربر و دیتابیس است. فانکشن های مختلفی در این کلاس پیاده‌سازی شده که در ادامه به آن پرداخته می‌شود.

کانستراکتور

کانستراکتور این کلاس بلاک‌چین کشوری، پابلیک‌کی وزارت کشور و محدوده انتخابات را از بیرون دریافت، یک بلاک‌چین محلی ساخته و پابلیک‌کی در آن قرار داده می‌شد.

فانکشن public_key

این پراپرتی فانکشن، وظیفه کوئری زدن به بلاک‌چین و یافتن پابلیک‌کی وزارت کشور را دارد.

فانکشن already_voted

این فانکشن با کوئری زدن روی بلاک‌چین، بررسی می‌کند کاربر با هش کد ملی داده شده قبلا رای داده یا خیر.

فانکشن election_is_over

این فانکشن نیز با کوئری زدن روی بلاک‌چین، بررسی می‌کند که آیا انتخابات توسط مقامات به اتمام رسیده یا خیر. ملاک این فانکشن وجود بلاکی با دیتای پایان انتخابات است.

فانکشن vote

این فانکشن لاجیک رای دادن را در خود دارد.

```python
def vote(self, person: society.Commons, party: society.Party):
    if not self.city.is_ip_in_area_range(person.ip):
        raise Exception("You can vote only in city area.")
    if self.election_is_over():
        raise Exception("Election is over")
    if self.already_voted(person):
        raise Exception("Duplicated vote")

    self.local_chain.add_block(
        f"vote#{person.hashed_national_code}#{self.public_key.encrypt(party.value).ciphertext()}"
    )
```

برای رای دادن باید ip فرد رای دهنده در رنج ip محدوده انتخاباتی مورد نظرش باشد، باید انتخابات تمام نشده باشد و همچنین اولین رایش باشد.
رای فرد (+۱ یا -۱) با استفاده از پابلیک‌کی paillier وزارت کشور رمز و در بلاک چین قرار داده می‌شود.

فانکشن finish_election

این فانکشن چک پوینت اتمام انتخابات را در بلاک‌چین می‌گذارد.

```
def finish_election(self, requested_person):
    if not self.city.is_ip_in_area_range(requested_person.ip):
        raise Exception("You can finish election only in city area.")
    if self.election_is_over():
        raise Exception("Election is already over.")

    self.local_chain.add_block(f"finish#{datetime.datetime.now().timestamp()}")
```

برای این امر باید ip رای دهنده در رنج آیپی محدوده رای گیری باشد و انتخابات پیش از این تمام نشده باشد.

فانکشن count_votes

این فانکشن آراء و رأی دهندگان را می‌شمارد و نتیجه را در بلاک‌چین محلی و کشوری می‌گذارد.

```python
def count_votes(self, requested_person):
    if not self.city.is_ip_in_area_range(requested_person.ip):
        raise Exception("You can count votes only in city area.")
    if not self.election_is_over():
        raise Exception("Election isn't over.")
    if not self.local_chain.assert_correct():
        raise Exception("The election was rigged.")

    current_block = self.local_chain.head
    sum_votes = self.public_key.encrypt(0)
    voter_count = 0
    while current_block:
        data_type, payload = current_block.data.split("#", 1)
        if data_type != "vote":
            current_block = current_block.next_block
            continue
        _, raw_vote = payload.split("#")
        vote = phe.paillier.EncryptedNumber(self.public_key, int(raw_vote))
        sum_votes += vote
        voter_count += 1

        current_block = current_block.next_block

    self.local_chain.add_block(
        f"result#{voter_count}#{sum_votes.ciphertext()}"
    )
    self.country_chain.add_block(
        f"result#{self.city.id}#{voter_count}#{sum_votes.ciphertext()}"
    )
```

شرطی که علاوه بر چک‌های فانکشن‌های پیشین خواهیم داشت، بررسی دست‌کاری نشدن بلاک‌چین است.

همانطور که در تصویر نیز به چشم می‌خورد، در شمارش آراء از جمع کردن مقادیر به صورت هوموورفیک رمز شده آراء استفاده شده است.

کلاس Ministry

این کلاس حاوی لاجیک مختص وزارت است. این لاجیک محدود فقط شامل جنریت کردن کلیدهای paillier، ساخت بلاک چین کشوری، و قرار دادن کلید عمومی در بلاک‌چین است.

کانستراکتور

```
def __init__(self, country_name: str, ip_range: str):
    self.country_chain = BlockChain("#")
    self.public_key, self.private_key = phe.paillier.generate_paillier_keypair()
    self.country_chain.add_block(
        f"public_key#{self.public_key.n}"
    )
    self.country = Area(country_name, ip_range)
```

در مقدمه این کلاس لاجیک اصلی بیان شد. همچنین یک Area با رنج آیپی ای که از بیرون آمده، ساخته می‌شود. این Area از جهت بودن در شبکه کشور است.

فانکشن announce_election_result

تنها فانکشن این کلاس برای اعلام عمومی آراء به ملت است. یک مسئول پس از اتمام رای گیری و شمارش آراء، از بلاک چین کشوری مقدار encrypted مجموع آراء که پیش از این توسط مسئولین محلی در بلاک‌چین محلی شمرده شده و در بلاک چین کشوری قرار داده شده، decrypt می‌شود و به اطلاع همگان می‌رسد.

```
def announce_election_result(self, city_id, ip):
    if not self.country.is_ip_in_area_range(ip):
        raise Exception("You should disconnect your vpn.")
    if not self.country_chain.assert_correct():
        raise Exception("Cheating in election result.")

    election_result = self.country_chain.find_block(
        lambda block_data: block_data.startswith(f"result#{city_id}#")
    )
    if not election_result:
        raise Exception("Election votes have not been counted.")

    _, _, voter_count, encrypted_vote_result = election_result.data.split("#")
    return self.private_key.decrypt(phe.paillier.EncryptedNumber(
        self.public_key, int(encrypted_vote_result))), int(voter_count)
```

همان‌طور که در تصویر مشخص است، باید بودن در رنج آیپی کشور و دستکاری نشدن بلاک‌چین کشوری، نیازمندی اصلی اعلام همگانی آراء است.

مدل‌های society.py

مدل Human و بچه‌های آن

کلاس Human وظیفه نگه‌داشتن هش کدملی افراد را دارد.
کلاس Commons مردم عادی ای هستند که رای می‌دهند. این کلاس علاوه بر داشتن فیلد هش کد ملی که به واسطه فرزند Human بودن به او به ارث رسیده، فیلد ip را نیز داراست.
کلاس های LocalOfficials و LocalOfficials بچه‌های Commons هستند و از این جهت جدا شده اند که دسترسی مسئولین مختلف به فانکشنالیتی ministry مدیریت شود.

اینام کلاس Party

جناح های انتخاباتی را در خود دارد

```
class Party(int, enum.Enum):
    A = +1
    B = -1
```

کلاس Area

این کلاس برای نگه‌داشتن اسم شهر و مدیریت رنج آیپی درخواست دهندگان گذاشته شده است.

```
class Area:
    @classmethod
    def ip_pattern_to_regex(cls, ip_pattern: str):
        return re.compile(ip_pattern.replace("X", "\d{1,3}").replace(".", "\."))

    def __init__(self, area_id: str, ip_pattern: str):
        self.ip_pattern = self.ip_pattern_to_regex(ip_pattern)
        self.id = area_id

    def is_ip_in_area_range(self, ip):
        return self.ip_pattern.match(ip) is not None
```

فانکشن اصلی این کلاس تبدیل پترن رنج آیپی (X.X.X.X) به ریجکس و آماده‌سازی برای فانکشن is_ip_in_range است.

فایل main.py

این فایل با استفاده از دیزاین پترن state، استیت ماشین شبیه‌سازی CLI را مدیریت می‌کند.
فیلدهای گلوبال

```
officials = {
    "0": CountryOfficials("0", "1.1.1.1", "85.1.1.1"),

    "1": CountryOfficials("1", "10.1.1.1", "85.1.1.2"),
}
people = {
    "1": Commons("1", "1.1.1.2"),
    "2": Commons("2", "1.1.1.3"),
    "3": Commons("3", "1.1.1.4"),

    "100": Commons("100", "100.1.1.2"),
    "200": Commons("200", "100.1.1.3"),
    "300": Commons("300", "100.1.1.4"),

    "10": Commons("10", "10.1.1.2"),
    "20": Commons("20", "10.1.1.3"),
    "30": Commons("30", "10.1.1.4"),
}
ministry = Ministry("iran", "85.X.X.X")
elections = {}
```

این فیلد ها شامل مقادیر اولیه اجرای برنامه است officials و people شبیه‌سازی ای از دیتابیس وزارت و ثبت احوال است.
دیکشنری elections نیز به ما این امکان را می‌دهد بتوانیم در یک شبیه‌سازی چند انتخابات داشته باشیم.

استیت ماشین

اینترفیس State توسط استیت‌های برنامه پیاده‌سازی می‌شود.
با توجه به طولانی شدن گزارش از بیان جزئیات پیاده‌سازی خودداری می‌شود. اما به صورت کلی این استیت ها ورودی را از کاربر می‌گیرند، با توجه به استیت جاری یکی از فانکشن های کلاس های election و ministry را کال می‌کنند و به استیت بعدی می‌روند.

شبیه‌سازی

علاوه بر CLI دو فانکشن که حالات مختلف رای‌گیری را با فانکشن‌کال شبیه‌سازی می‌کند نیز اضافه شده اند.

رای گیری بدون تقلب

```python
def simulate_without_cheating():
    official = officials.get("0")
    election = Election(ministry.country_chain, ministry.public_key, society.Area("tehran", "1.1.1.X"))

    test_cases = (
        ("1", society.Party.A),
        ("100", society.Party.A),  # out of area
        ("2", society.Party.B),
        ("2", society.Party.B),  # duplicated vote
        ("3", society.Party.B),
    )

    for person_id, vote in test_cases:
        try:
            election.vote(people.get(person_id), vote)
        except Exception as e:
            print(str(e))
            continue
        print("Voted.")
    try:
        election.count_votes(official)
    except Exception as e:
        print(str(e))  # count before finishing election

    election.finish_election(official)
    election.count_votes(official)
    result, voter_count = ministry.announce_election_result("tehran", official.public_ip)
    print(result, voter_count)
```

به صورت مختصر، خط ۱۸۲ یک فرد از بیرون محدوده اقدام به رای گیری می‌کند.

در خط ۱۸۴ یک فرد رای تکراری می‌دهد.

در خط ۱۹۶ پیش از اتمام انتخابات، اقدام به شمردن رای ها می‌شود.

پس انتظار داریم در خروجی این سه حالت با خطا مواجه شوند. تعداد رای ها نیز با توجه به شرکت کنندگان، پیروزی جناح B با یک رای بیشتر بین ۳ شرکت کننده است.

```
Voted.
You can vote only in city area.
Voted.
Duplicated vote
Voted.
Election isn't over.
-1 3
```

عدد منفی یک به معنای یک رای بیشتر B و عدد ۳ به معنای ۳ شرکت کننده است.

رای گیری با تقلب

```python
def simulate_with_cheating():
    official = officials.get("1")
    election = Election(ministry.country_chain, ministry.public_key, society.Area("tehran", "10.1.1.X"))

    test_cases = (
        ("10", society.Party.A),
        ("20", society.Party.A),
        ("30", society.Party.A),
    )

    for person_id, vote in test_cases:
        election.vote(people.get(person_id), vote)
        print("Voted.")

    election.finish_election(official)
    backup_data = election.local_chain.head.next_block.data
    election.local_chain.head.next_block.data = "poisoned data"  # cheating
    try:
        election.count_votes(official)
    except Exception as e:
        print(str(e))
    election.local_chain.head.next_block.data = backup_data
    election.count_votes(official)
    result, voter_count = ministry.announce_election_result("tehran", official.public_ip)
    print(result, voter_count)
```

در این حالت در خط ۲۲۲، یکی از بلاک‌ها دستکاری می‌شود.

پس در خط ۲۲۴ که شمارش آراء را داریم، باید خطا داشته باشیم.

دیتای درست در بلاک قرار داده می‌شود و انتظار خواهیم داشت شمارش با موفقیت و با ۳ رای بیشتر از ۳ شرکت کننده به سود جناح A به پایان برسد.

```
Voted.
Voted.
Voted.
The election was rigged.
result: 3, count: 3
```

چالش‌ها

شاید تایپ سیف نبودن پایتون کمی تعداد اجرا را برای تست و فیکس کردن مشکلات بالا برد اما به طور خاص در حل این سوال چالش خاصی متوجه ما نشد.