Ворнаков Лабораторная 1 Лабораторная работа реализована на python с использованием стандартных библиотек, а также https://github.com/mosquito/pygost (https://github.com/mosquito/pygost) для использования ГОСТовых функций

в функции demo производится демонстрация функционала реализованной PKI

код:

```
import json
from datetime import datetime, timedelta
from typing import Dict, Tuple
import secrets
from pygost.gost3410 import CURVES, prv_unmarshal, public_key, sign, verify
from pygost import gost34112012256
from time import sleep


class CryptoUtils:

    CURVE = CURVES["id-tc26-gost-3410-12-512-paramSetA"]

    @staticmethod
    def generate_gost_keypair() -> Tuple[dict, Tuple[int, int]]:
        """Генерация ключевой пары по ГОСТ 34.10-2012 (512 бит)"""
        prv_raw = secrets.token_bytes(64)
        prv = prv_unmarshal(prv_raw)
        pub = public_key(CryptoUtils.CURVE, prv)
        return {"curve": CryptoUtils.CURVE, "private": prv}, pub

    @staticmethod
    def sign_gost(data: bytes, private_key: dict) -> Tuple[int, int]:
        """Подпись данных по ГОСТ 34.10-2012"""
        digest = gost34112012256.new(data).digest()
        return sign(private_key["curve"], private_key["private"], digest)

    @staticmethod
    def verify_gost(data: bytes, signature: Tuple[int, int], public_key: Tuple[int, int]) -> bool:
        """Проверка подписи по ГОСТ 34.10-2012"""
        try:
            digest = gost34112012256.new(data).digest()
            return verify(CryptoUtils.CURVE, public_key, digest, signature)
        except (ValueError, KeyError):
            return False


class Certificate:

    def __init__(self, serial: int, ca_sign_algo: str, ca_name: str,
                 valid_from: datetime, valid_to: datetime,
                 key_algo: str, participant_id: str, public_key: Tuple[int, int],
                 signature: Tuple[int, int] = None):
        self.serial = serial
        self.ca_sign_algo = ca_sign_algo
        self.ca_name = ca_name
        self.valid_from = valid_from
        self.valid_to = valid_to
        self.key_algo = key_algo
        self.participant_id = participant_id
        self.public_key = public_key
        self.signature = signature

    def to_bytes(self) -> bytes:
        """Сериализация сертификата в bytes для подписи"""
        data = {
            "serial": self.serial,
            "ca_sign_algo": self.ca_sign_algo,
            "ca_name": self.ca_name,
            "valid_from": self.valid_from.isoformat(),
            "valid_to": self.valid_to.isoformat(),
            "key_algo": self.key_algo,
            "participant_id": self.participant_id,
            "public_key": self.public_key
        }
```

```
        return json.dumps(data, sort_keys=True).encode('utf-8')

    def verify(self, ca_public_key: Tuple[int, int]) -> bool:
        """Проверка подписи сертификата"""
        if not self.signature:
            return False
        data = self.to_bytes()
        return CryptoUtils.verify_gost(data, self.signature, ca_public_key)

    def is_valid(self, current_time: datetime = None) -> bool:
        """Проверка срока действия сертификата"""
        if current_time is None:
            current_time = datetime.now()
        return self.valid_from <= current_time <= self.valid_to


class CRL:

    def __init__(self, ca_name: str,
                 this_update: datetime, next_update: datetime,
                 revoked_certs: Dict[int, datetime] = None,
                 signature: Tuple[int, int] = None):
        self.ca_name = ca_name
        self.ca_sign_algo = "GOST3410-2012-512"
        self.this_update = this_update
        self.next_update = next_update
        self.revoked_certs = revoked_certs or {}
        self.signature = signature

    def to_bytes(self) -> bytes:
        """Сериализация CRL в bytes для подписи"""
        data = {
            "ca_name": self.ca_name,
            "ca_sign_algo": self.ca_sign_algo,
            "this_update": self.this_update.isoformat(),
            "next_update": self.next_update.isoformat(),
            "revoked_certs": {str(k): v.isoformat() for k, v in self.revoked_certs.items()}
        }
        return json.dumps(data, sort_keys=True).encode('utf-8')

    def is_revoked(self, cert_serial: int) -> bool:
        """Проверка, отозван ли сертификат"""
        return cert_serial in self.revoked_certs

    def add_revoked(self, cert_serial: int, revocation_time: datetime = None):
        """Добавление отозванного сертификата"""
        if revocation_time is None:
            revocation_time = datetime.now()
        self.revoked_certs[cert_serial] = revocation_time


class CertificateAuthority:

    def __init__(self, name: str):
        self.name = name
        self.private_key, self.public_key = CryptoUtils.generate_gost_keypair()
        self.certificates: Dict[int, Certificate] = {}
        self.crl = CRL(
            ca_name=self.name,
            this_update=datetime.now(),
            next_update=datetime.now() + timedelta(days=1))
        self._next_serial = 1

    def issue_certificate(self, participant_id: str, public_key: Tuple[int, int],
                          proof_of_private_key: Tuple[int, int] = None) -> Certificate:
        """Выпуск сертификата с проверкой знания закрытого ключа"""
```

```
        # Проверка уникальности идентификатора
        for cert in self.certificates.values():
            if cert.participant_id == participant_id:
                print(f"Участник {participant_id} уже зарегистрирован")
                return None

        # Проверка знания закрытого ключа (proof of possession)
        if proof_of_private_key:
            test_data = b"proof_of_private_key"
            if not CryptoUtils.verify_gost(test_data, proof_of_private_key, public_key):
                print("Не удалось подтвердить знание закрытого ключа")
                return None

        # Создание сертификата
        now = datetime.now()
        cert = Certificate(
            serial=self._next_serial,
            ca_sign_algo="GOST3410-2012-512",
            ca_name=self.name,
            valid_from=now,
            valid_to=now + timedelta(seconds=5),
            key_algo="GOST3410-2012-512",
            participant_id=participant_id,
            public_key=public_key
        )

        # Подпись сертификата
        data = cert.to_bytes()
        cert.signature = CryptoUtils.sign_gost(data, self.private_key)

        # Сохранение сертификата
        self.certificates[cert.serial] = cert
        self._next_serial += 1

        return cert

    def revoke_certificate(self, cert_serial: int, requester_id: str) -> bool:
        """Отзыв сертификата"""
        if cert_serial not in self.certificates:
            print(f"Сертификат с серийным номером №{cert_serial} не найден")
            return False

        # Добавление в CRL
        self.crl.add_revoked(cert_serial)

        # Подпись обновленного CRL
        data = self.crl.to_bytes()
        self.crl.signature = CryptoUtils.sign_gost(data, self.private_key)

        print(f"Сертификат №{cert_serial} отозван")
        return True

    def get_certificate(self, cert_serial: int) -> Certificate:
        """Получение сертификата по серийному номеру"""
        return self.certificates.get(cert_serial)

    def get_crl(self) -> CRL:
        """Получение текущего CRL"""
        return self.crl

    def verify_certificate(self, cert: Certificate) -> bool:
        """Полная проверка сертификата"""
        if not cert.verify(self.public_key):
            print("Недействительная подпись сертификата")
            return False
```

```
        if not cert.is_valid():
            print("Срок действия сертификата истек")
            return False

        if self.crl.is_revoked(cert.serial):
            print("Сертификат отозван")
            return False

        return True


class Participant:

    def __init__(self, name: str, ca: CertificateAuthority):
        self.name = name
        self.ca = ca
        self.key_algorithm = "GOST3410-2012-512"
        self.private_key, self.public_key = CryptoUtils.generate_gost_keypair()
        self.certificate: Certificate = None
        self.other_certs: Dict[int, Certificate] = {}

    def request_certificate(self) -> bool:
        """Запрос сертификата с подтверждением знания закрытого ключа"""
        # Создаем proof of possession
        test_data = b"proof_of_private_key"
        signature = CryptoUtils.sign_gost(test_data, self.private_key)

        cert = self.ca.issue_certificate(self.name, self.public_key, signature)
        if cert:
            self.certificate = cert
            return True
        return False

    def revoke_my_certificate(self) -> bool:
        """Запрос отзыва собственного сертификата"""
        if not self.certificate:
            print("Нет активного сертификата")
            return False
        return self.ca.revoke_certificate(self.certificate.serial, self.name)

    def get_participant_certificate(self, cert_serial: int) -> Certificate:
        """Получение сертификата другого участника"""
        if cert_serial not in self.other_certs:
            cert = self.ca.get_certificate(cert_serial)
            if cert:
                self.other_certs[cert_serial] = cert

        return self.other_certs.get(cert_serial)

    def verify_participant_certificate(self, cert: Certificate) -> bool:
        """Проверка сертификата другого участника"""
        if not cert:
            return False

        # Проверка подписи УЦ
        if not cert.verify(self.ca.public_key):
            print("Недействительная подпись УЦ")
            return False

        # Проверка отзыва
        crl = self.ca.get_crl()
        if crl.is_revoked(cert.serial):
            print("Сертификат отозван")
            return False

        # Проверка срока действия
```

```
        if not cert.is_valid():
            print("Сертификат просрочен")
            return False

        return True


def demo():
    print("=== Демонстрация работы Удостоверяющего центра ===")

    print("\n1. Создание Удостоверяющего центра...")
    ca = CertificateAuthority(name="GOST-CA")
    print(f"УЦ '{ca.name}' успешно создан")

    print("\n2. Создание участников системы...")
    alice = Participant("Alice", ca)
    bob = Participant("Bob", ca)
    carl = Participant("Carl", ca)
    print("Участники Alice, Bob и Carl созданы")

    print("\n3. Запрос сертификатов участниками...")
    print("Alice запрашивает сертификат...", "Успешно" if alice.request_certificate() else "Ошибка")
    print(f"Сертификат Alice: {alice.certificate.to_bytes()}")
    print("\nBob запрашивает сертификат...", "Успешно" if bob.request_certificate() else "Ошибка")
    print(f"Сертификат Bob: {bob.certificate.to_bytes()}")
    print("\n4. Обмен и проверка сертификатов...")
    bob_cert = alice.get_participant_certificate(bob.certificate.serial)
    alice_cert = bob.get_participant_certificate(alice.certificate.serial)

    print(f"Alice проверяет сертификат Bob (№{bob_cert.serial})...",
          "Валиден" if alice.verify_participant_certificate(bob_cert) else "Невалиден")
    print(f"Bob проверяет сертификат Alice (№{alice_cert.serial})...",
          "Валиден" if bob.verify_participant_certificate(alice_cert) else "Невалиден")

    print("\nПроверка истечения времени действия сертификатов...")
    sleep(6)
    print(f"Alice проверяет сертификат Bob (№{bob_cert.serial})...",
          "Валиден" if alice.verify_participant_certificate(bob_cert) else "Невалиден")
    print(f"Bob проверяет сертификат Alice (№{alice_cert.serial})...",
          "Валиден" if bob.verify_participant_certificate(alice_cert) else "Невалиден")

    print("\n5. Отзыв сертификата...")
    print("Carl отзывает невыданный сертификат...", "Успешно" if carl.revoke_my_certificate() else "Ошибка")
    print("Carl запрашивает сертификат...", "Успешно" if carl.request_certificate() else "Ошибка")
    carl_cert = alice.get_participant_certificate(carl.certificate.serial)
    print(f"Alice проверяет сертификат Carl (№{carl_cert.serial})...",
          "Валиден" if alice.verify_participant_certificate(carl_cert) else "Невалиден")
    print("Carl отзывает свой сертификат...",
          "Успешно" if carl.revoke_my_certificate() else "Ошибка")

    # 6. Проверка после отзыва
    print("\n6. Проверка после отзыва...")
    print("Alice проверяет сертификат Carl...",
          "Валиден" if alice.verify_participant_certificate(carl_cert) else "Невалиден")


if __name__ == "__main__":
    demo()
```

вывод программы такой:

```
=== Демонстрация работы Удостоверяющего центра ===

1. Создание Удостоверяющего центра...
УЦ 'GOST-CA' успешно создан

2. Создание участников системы...
Участники Alice, Bob и Carl созданы

3. Запрос сертификатов участниками...
Alice запрашивает сертификат... Успешно
Сертификат Alice: b'{"ca_name": "GOST-CA", "ca_sign_algo": "GOST3410-2012-512", "key_algo": "GOST3410-2012-512", "participant_id": "

Bob запрашивает сертификат... Успешно
Сертификат Bob: b'{"ca_name": "GOST-CA", "ca_sign_algo": "GOST3410-2012-512", "key_algo": "GOST3410-2012-512", "participant_id": "Bo

4. Обмен и проверка сертификатов...
Alice проверяет сертификат Bob (№2)... Валиден
Bob проверяет сертификат Alice (№1)... Валиден

Проверка истечения времени действия сертификатов...
Сертификат просрочен
Alice проверяет сертификат Bob (№2)... Невалиден
Сертификат просрочен
Bob проверяет сертификат Alice (№1)... Невалиден

5. Отзыв сертификата...
Нет активного сертификата
Carl отзывает невыданный сертификат... Ошибка
Carl запрашивает сертификат... Успешно
Alice проверяет сертификат Carl (№3)... Валиден
Сертификат №3 отозван
Carl отзывает свой сертификат... Успешно

6. Проверка после отзыва...
Сертификат отозван
Alice проверяет сертификат Carl... Невалиден

Process finished with exit code 0
```