Лабораторна роботи № 4.
Дослідження особливостей реалізації існуючих програмних систем, які використовують криптографічні механізми захисту інформації

Виконали: *Бараніченко Андрій, Гаврилова Анастасія, Дрозд Софія, Зібаров Дмитро, Колесник Андрій*

Мета роботи: Отримання практичних навичок побудови гібридних криптосистем.

Завдання: Для першого типу лабораторних робіт – дослідити основні задачі, що виникають при програмній реалізації криптосистем. Запропонувати методи вирішення задачі контролю доступу до ключової інформації, що зберігається в оперативній пам'яті ЕОМ для різних (обраних) операційних систем. Запропонувати методи вирішення задачі контролю правильності функціонування програми криптографічної обробки інформації. Порівняти з точки зору вирішення цих задач інтерфейси Crypto API, PKCS 11.
Підгрупа 1A. Реалізація для інтелектуальної картки, токена.
Підгрупа 1B. Реалізація для User Endpoint terminal.
Підгрупа 1C. Реалізація для провайдера хмарних послуг

Для роботи було обрано варіант Підгрупа 1A. Реалізація для інтелектуальної картки, токена.

Інтелектуальні картки (смарт-картки) - пристрої, носії ключової інформації, що призначені для використання в системах інформаційного доступу, для персональної ідентифікації, автентифікації, зберігання даних і обробки заявок. Крім того, вони забезпечують надійну автентифікацію безпеки для єдиного входу (SSO - single sign-on). Тобто, в основному цей пристрій можна побачити у форм-факторі звичайної банківської картки, але із певними відмінностями. Є різні типи смарт-карток, які поділяються за двома критеріями: функціоналість (на основі мікропроцесора/пам'яті) та зв'язок з зчитувачем (контактні/безконтактні/комбіновані/гібридні картки).

Смарт-токен – це носій ключової інформації з USB інтерфейсом виконаний у металевому корпусі. Пристрій призначений для використання в системах інформаційного доступу, кваліфікованого електронного підпису, документообігу, інших системах для авторизації користувачів, захищеного зберігання та використання ключової інформації, а також може використовуватися в якості модуля безпеки в центрах сертифікації ключів та інших системах. Іншими словами, це є звичайною «флешкою», що

використовує у собі, знову ж таки, спеціальний чіп, що вміє зберігати інформацію, редагувати її та виконувати криптографічні функції.

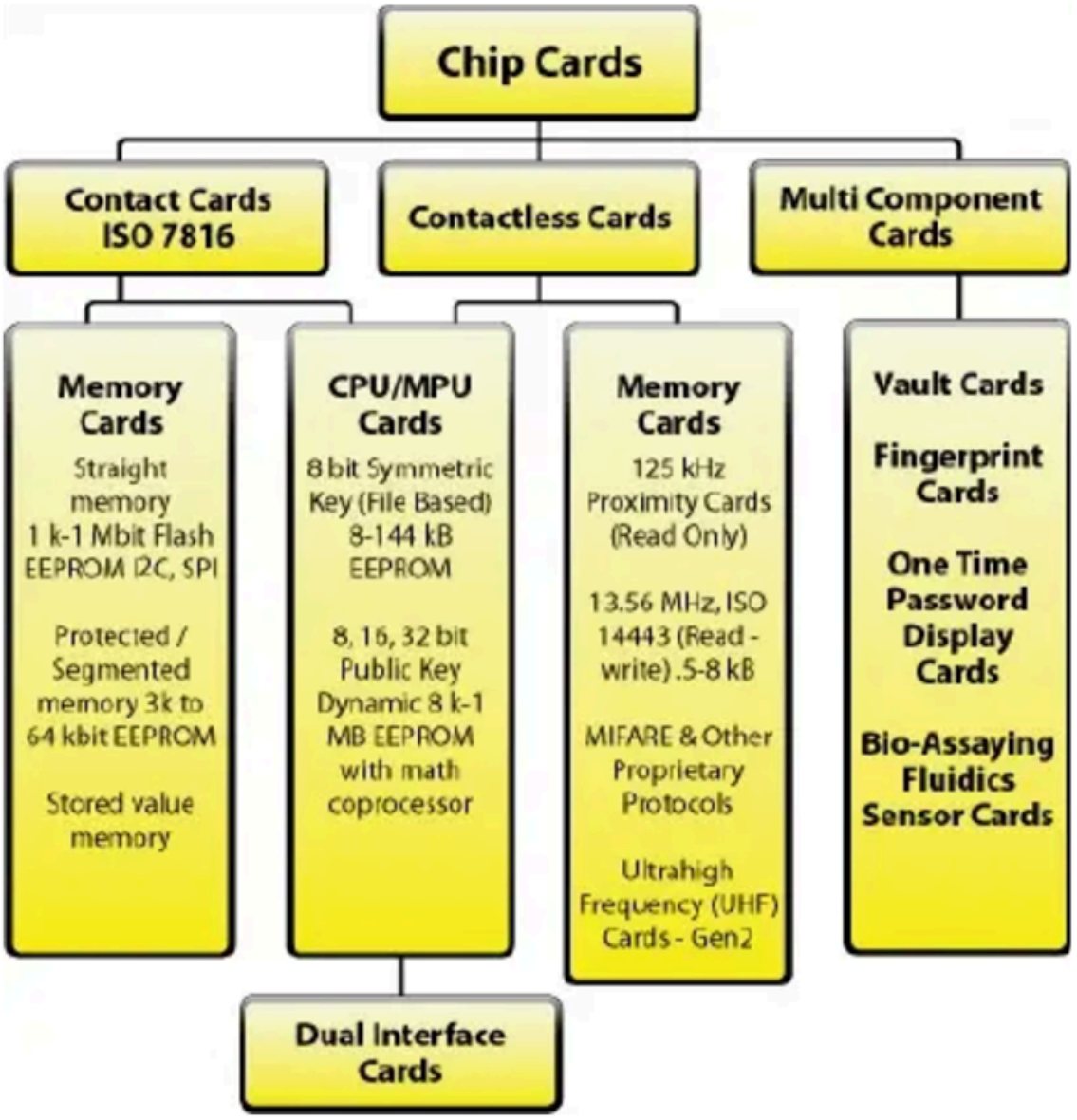

Рис.1 Типи смарт-карток

а) б)

Рис.2 Приклади токенів: а) - смарт картка (студентський квиток), б) - серія YubiKey 5 від Yubico

В криптографії є різні криптографічні інтерфейси і для Crypto API можна виділити наступні: для операційної системи Windows та для Linux.

Microsoft CryptoAPI (застаріла назва, нова – CNG(CryptoAPI Next Generation)) – інтерфейс прикладних програм (API), що входить до складу ОС Microsoft Windows, і дозволяє розробникам захищати свої розробки засобами криптографії. Даний інтерфейс підтримує як симетричну, так і асиметричну криптографію, можливість автентифікації за допомогою цифрових сертифікатів, генерація та перевірка цифрових підписів, алгоритми хеш-функції для перевірки цілісності даних, захист від сторонніх атак, а також криптографічно стійкий генератор псевдовипадкових чисел CryptGenRandom, що для генерації чисел використовує такі дані як: поточний ID процесу, кількість тактів з часу завантаження, поточний час, MD4-геш блоку середовища користувача, що містить ім'я користувача, ім'я комп'ютера, шлях тощо. Більш того, Microsoft CryptoAPI підтримує криптографію на еліптичних кривих, що є особливо актуальним, коли ми говоримо про застосування для інтелектуальної картки або токена, адже в цьому випадку ми дуже обмежені в обчислювальних і просторових ресурсах, а криптографія на еліптичних кривих вимагає менші розміри ключа при збереженні того ж рівня безпеки, що, наприклад, і RSA.

CryptoAPI містить в собі функції, що дозволять додаткам виконувати різні криптографічні операції, які виконуються незалежними модулями, які називають провайдерами криптографічних послуг (CSP). Кожен CSP має базу даних ключів, у якій він зберігає свої постійні ключі. Кожна база даних містить один або кілька контейнерів ключів, кожен з яких містить усі пари ключів, що належать певному клієнту. Кожному контейнеру привласнюється унікальне ім'я.

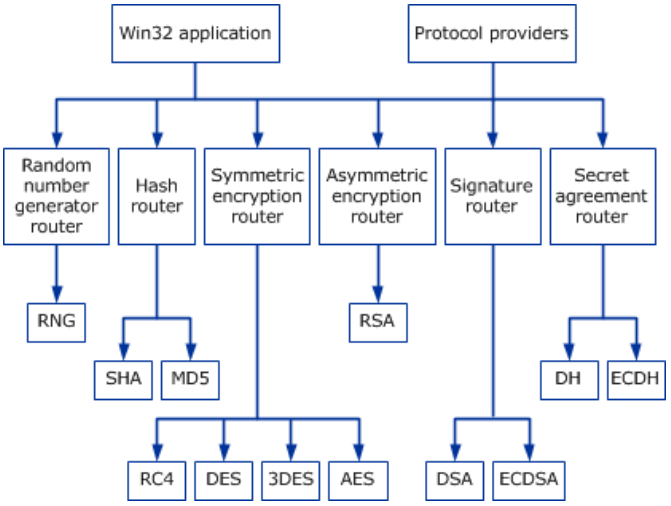

Рис.3 Криптопримітиви і криптоалгоритми, які підтримує CryptoAPI

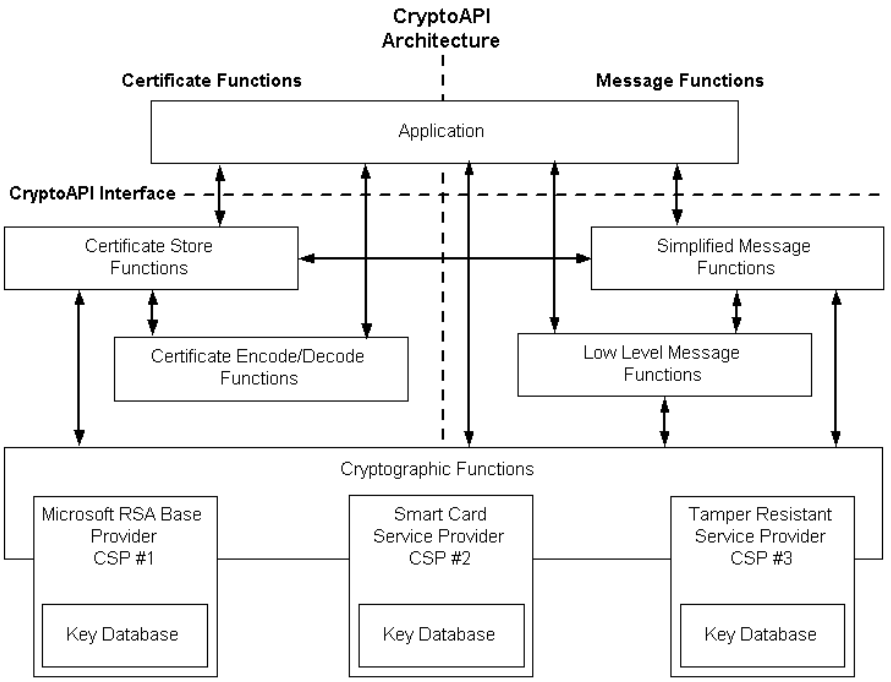

Рис.4 Архітектура CryptoAPI

Модель зберігання секретних ключів в CryptoAPI дозволяє адаптуватися до зміни вимог створення додатків, що використовують його криптографічні функції. Маршрутизатор зберігання ключів є центральною процедурою: через нього додаток отримує доступ до провайдерів зберігання ключів (KSP). Своєю чергою, маршрутизатор ізолює ключі як від додатку, так і від самого провайдеру зберігання. Ключі ізолюються таким чином, що вони ніколи не присутні в процесі подання заявок. Відкритий ключ зберігається окремо від закритого.

CryptoAPI підтримує наступні типи ключів:

- Публічний і секретний ключі Діффі-Гелмана.
- Публічний і секретний ключі для цифрового підпису DSA
- Публічний і секретний ключі RSA
- Публічний і секретний ключі для криптографії на еліптичних кривих

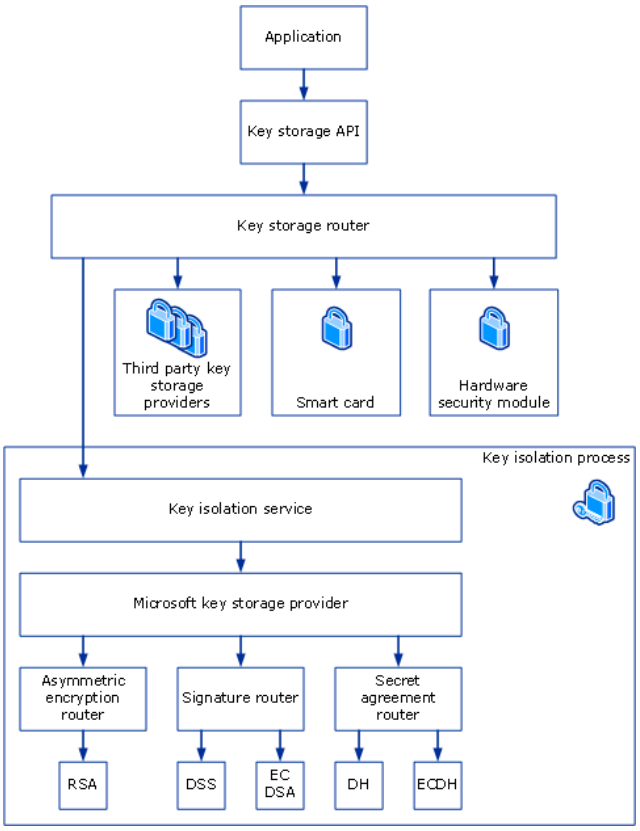

Рис. 5 Архітектура ізоляції ключів в CryptoAPI

Crypto API (Linux) - це криптографічна структура в ядрі Linux для різних частин ядра, які займаються криптографією, таких як IPsec і dm-crypt. Він був представлений у 2002 році і основною ціллю було забезпечення внутрішніх потреб, однак крім самого ядра користь від нього може отримати і користувач. На даний час він має величезний функціонал із генерування симетричних/асиметричних ключів, взаємодії із різноманітними криптопримітивами.

– IPsec – це набір безпечних мережевих протоколів, який автентифікує та шифрує пакети даних, щоб забезпечити безпечний зашифрований зв'язок між двома комп'ютерами через мережу. Він використовується у віртуальних приватних мережах (VPN).

– dm-crypt – прозора підсистема шифрування блокових пристроїв у ядрі Linux версії 2.6 і пізніших. Воно є частиною інфраструктури пристрою відображення пристроїв і використовує криптографічні процедури з Crypto API ядра.

PKCS11 - один із стандартів криптографії (сімейства Public-Key Cryptography Standards) з відкритим ключем, що відноситься до програмного інтерфейсу для створення та маніпулювання криптографічними токенами, де зберігається секретний ключ. Іноді іменується як Cryptoki (cryptographic token interface). Інтерфейс PKCS11

доступу до критоптографічних пристроїв (засобів криптографічоного захисту, смарткарток, серверам ключів тощо) є платформонезалежним. PKCS11 включає в себе набір криптографічних алгоритмів для симетричного і асиметричного шифрування і дешифрування, ідентифікації і аутентифікації криптографічних модулів, створення і перевірки цифрового підпису, схему генерації спільного ключа Діффі-Гелмана та постійного зберігання ключів (своєю чергою CryptoAPI не зберігає постійні ключі для симетричного щифрування). Також підтримується криптографія на еліптичних кривих. Для зберігання ключів PKCS11 використовує сховище CMOS.

Параметр	CryptoAPI	PKCS11
Платформна сумісність	Призначений для Windows	Кросплатформений
Простота використання	Вбудований у систему	Вимагає встановлення додаткової бібліотеки
Можливості	Підтримка стандартних алгоритмів для Windows (AES, DES, SHA-1 тощо)	Підтримка багатьох криптографічних алгоритмів
Гнучкість	Обмежена	Висока, завдяки стандартизації
Популярність	Широко використовується у Windows-додатках	Часто використовується для токенів та смарт-карток

Таким чином, для задач із смарт-картками та токенами PKCS11 є більш універсальним і практичним вибором, тоді як CryptoAPI більше підходить для розробки програм під Windows.

На останок можна коротко розказати про основні проблеми, що можуть виникнути при реалізації криптосистем на токенах та сматри картках. Відповідно до проведеного дослідження проблеми можна класифікувати наступним чином.
– Генератор псевдо випадкових чисел. У токенах дуже важливо мати надійне джерело ентропії для генерування ключових пар.

– Обмежена кількість пам’яті та обчислювальні можливості. Так як токен чи смарт-картка мають досить малий форм-фактор, то до них не вдастся приєднати більшу кількість пам’яті. Тоді коли на комп’ютері можна сказати набагато більші можливості для обчислень, то у токенах розробникам приходиться максимально оптимізовувати їх реалізації.

Можливість програмної реалізації криптографічного перетворення обумовлена тим, що кожен шифр є строго формальною алгоритмічною процедурою. При апаратній реалізації всі процедури виконуються спеціальними електронними схемами, що робить такий процес шифрування набагато швидшим, ніж програмний. Отже, перша задача, що стоїть перед програмістом при програмній реалізації шифру: максимально можлива його оптимізація. Також програмна реалізація шифру вимагає додаткових заходів щодо захисту від атаки на реалізацію. Але мабуть найбільшою проблемою реалізації будь-якого криптоалгоритму є питання безпечного зберігання і використання(розподіл) ключів.

Код

```
from PyKCS11 import PyKCS11Lib, PyKCS11Error, CKF_RW_SESSION, CKU_USER

# Завантаження бібліотеки PKCS11
# Шлях до бібліотеки залежить від конкретного токена (може бути .dll або .so файл)
lib_path = "/path/to/your/pkcs11_library.so"
lib = PyKCS11Lib()

try:
    lib.load(lib_path)
    print("Бібліотеку PKCS11 завантажено успішно.")
except Exception as e:
    print(f"Помилка завантаження бібліотеки: {e}")
    exit(1)

# Отримання списку слотів (фізичних або віртуальних пристроїв)
slots = lib.getSlotList(tokenPresent=True)
if not slots:
    print("Не знайдено доступних токенів.")
    exit(1)

# Підключення до першого токена
slot_id = slots[0]
session = lib.openSession(slotID=slot_id, flags=CKF_RW_SESSION)

try:
    # Аутентифікація користувача
    pin = "1234"  # PIN токена (у реальній програмі PIN потрібно вводити безпосередньо користувачем)
    session.login(pin, CKU_USER)
    print("Успішна аутентифікація.")

    # Пошук ключів (наприклад, закритих RSA ключів)
    template = [('CKA_CLASS', 'CKO_PRIVATE_KEY')]
    private_keys = session.findObjects(template)
```

```
    print(f"Знайдено {len(private_keys)} приватних ключів.")

    # Виконання криптографічної операції: підпис
    if private_keys:
        key = private_keys[0]  # Беремо перший ключ
        data_to_sign = b"Message to sign"  # Повідомлення для підпису
        mechanism = {'mechanism': 'CKM_SHA256_RSA_PKCS'}
        signature = session.sign(key, data_to_sign, mechanism)
        print(f"Підпис створено: {signature.hex()}")

    else:
        print("Ключів для підпису не знайдено.")

    # Завершення роботи
    session.logout()
    print("Сесія завершена успішно.")

except PyKCS11Error as e:
    print(f"Помилка під час роботи з токеном: {e}")

finally:
    session.closeSession()
```

В цьому підрозділі описується використання бібліотеки PKCS11 через модуль PyKCS11 на мові програмування Python. Код здійснює завантаження бібліотеки, аутентифікацію користувача, пошук приватних ключів та підписування даних на токені. Варто зазначити, що даний код є скоріше симуляцією роботи з токеном, оскільки реальний смарт-токен у членів команди відсутній.

Висновки

У цій лабораторній роботі було досліджено основні задачі, що виникають при програмній реалізації криптосистем, зокрема інтерфейси Microsoft CryptoAPI та PKCS11. Було проаналізовано їхні особливості, можливості та сфери застосування. Розглянуто, як ці інтерфейси дозволяють виконувати криптографічні операції, забезпечувати автентифікацію та захищати ключову інформацію.

У роботі також було реалізовано приклад використання бібліотеки PKCS11 через модуль PyKCS11 на Python. Код демонструє основні етапи роботи з токеном, включаючи завантаження бібліотеки, аутентифікацію користувача, пошук приватних ключів і підписування даних. Водночас цей приклад є симуляцією роботи, оскільки реальний смарт-токен був відсутній.

Результати роботи дозволяють зробити висновок, що PKCS11 є універсальним і кросплатформеним стандартом, який забезпечує широкі можливості роботи з криптографічними пристроями, тоді як Microsoft CryptoAPI є зручним інструментом для розробки додатків у середовищі

Windows. Представлений аналіз і реалізація показують, що ефективне використання цих інтерфейсів може значно спростити побудову безпечних інформаційних систем.