НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

КОМП'ЮТЕРНИЙ ПРАКТИКУМ № 4

з дисципліни:

«МЕТОДИ РЕАЛІЗАЦІЇ КРИПТОГРАФІЧНИЇ МЕХАНІЗМІВ»

Дослідження особливостей реалізації існуючих програмних систем, які використовують криптографічні механізми захисту інформації варіант 1А

Виконала:
Студентка групи ФІ-22мн
Калитюк Дар'я

КИЇВ 2022

Тема: отримання практичних навичок побудови гібридних криптосистем.

Постановка задачі: 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.

Хід роботи

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.

Microsoft CryptoAPI (*застаріла назва, нова –* ***CNG(CryptoAPI Next Generation))*** – інтерфейс програмування додатків, що входить до складу ОС *Microsoft Windows,* і дозволяє розробникам захищати свої розробки засобами криптографії. Однойменний фрейморк існує і для ОС Linux, проте він включає в себе лише блокові шифри і геш-функції. *Microsoft CryptoAPI* підтримує як симетричну, так і асиметричну криптографію, можливість автентифікації за допомогою цифрових сертифікатів, цифрові підвиси, алгоритми узгодження, 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 ключів. 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.

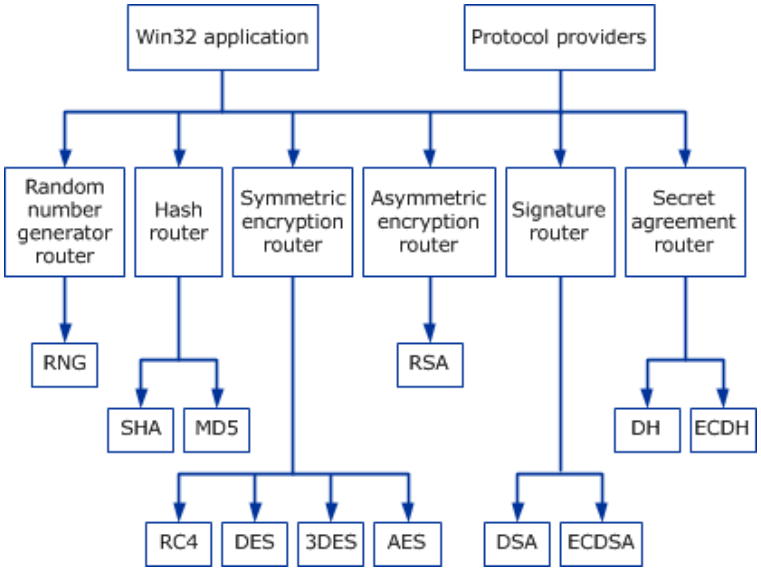


Рис. 1. Криптопримітиви і криптоалгоритми, які підтримує *CryptoAPI*

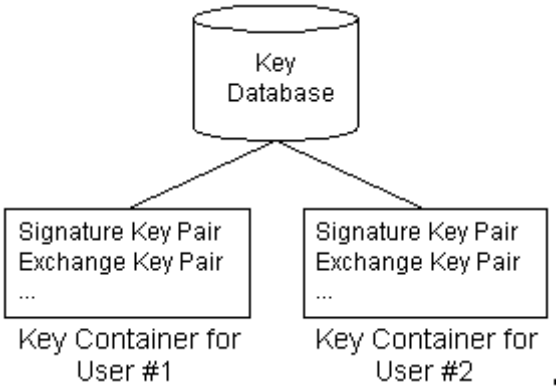


Рис. 2. Структура база даних ключів

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 збергіання. 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

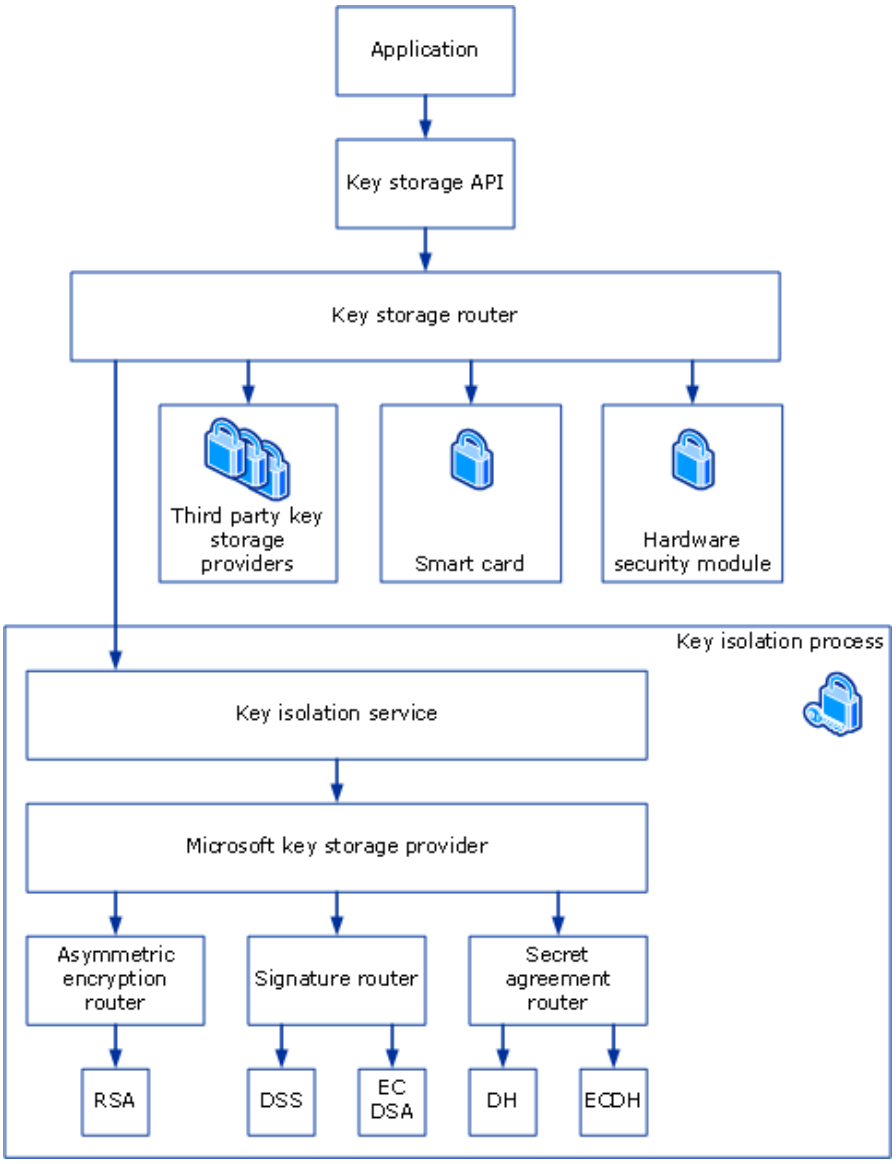


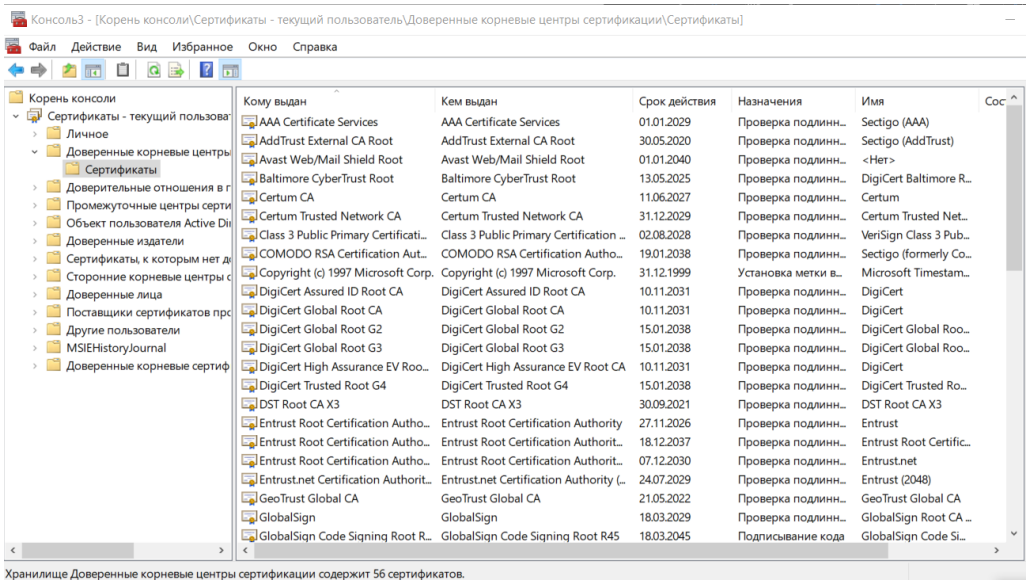
Рис. 3. Архітектура ізоляції ключів в *CryptoAPI*

PKCS11 (Public-Key Cryptography Standards) – стандарт асиметричної криптографії, розроблений RSA Labolatories. 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 дешифрвання, 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.

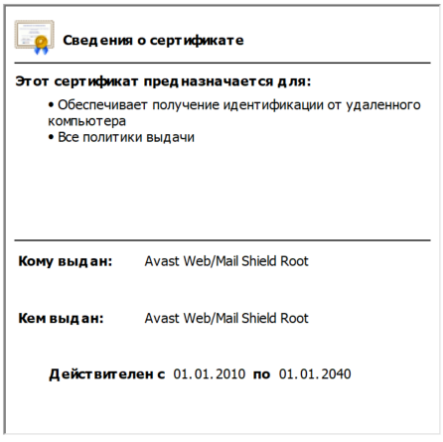
Деякі відомі додатки, що використовують у собі стандарти *PKCS11:*

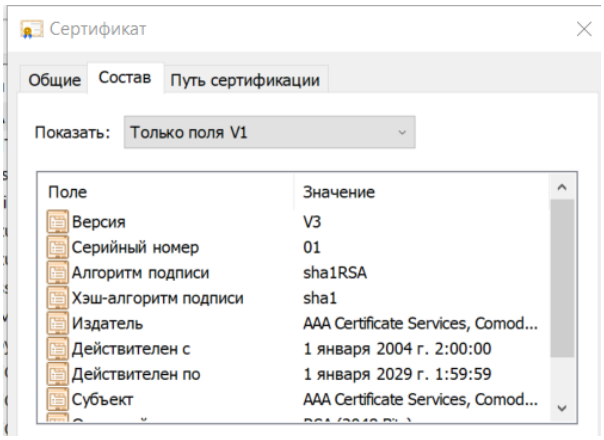
- Веб-браузер Mozilla Firefox
- Поштовий клієнт Mozilla Thunderbird
- Бібліотека OpenSSL
- OpenVPN
- OpenSSH

Microsoft Management Console — це компонент Microsoft Windows, який надає системним адміністраторам і досвідченим користувачам інтерфейс для налаштування та моніторингу системи. В ньому ми можемо, в тому числі, переглядати всі наші сертифікати. Ось, наприклад, список сертифікатів довірених центрів сертифікації: тут ми можемо побачити кому і ким був виданий кожен сертифікат (оскільки ми розглядаємо сертифікати довірених центрів сертифікації, то в обох полях стоять назви центрів), строк дії сертифікату, призначення і ім’я.



Можна переглянути вміст сертифікату, обмежитись лише полями версії V1, лише критичними розширеннями (Як от використання ключа і обмеження) тощо.



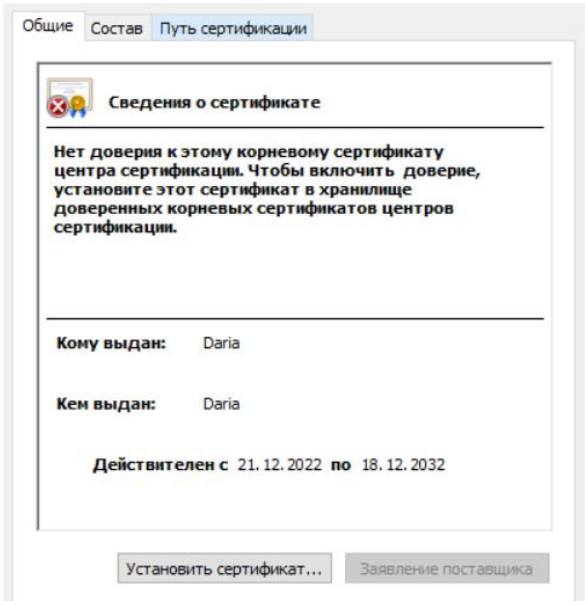


Засобами бібліотеки OpelSSL мовою Python можна написати функцію, яка дозволить нам видати собі сертифікат:

```
from OpenSSL import crypto

      def cert_gen():
          k = crypto.PKey()
          k.generate_key(crypto.TYPE_RSA, 4096)
          cert = crypto.X509()
          cert.get_subject().C = 'UA'
          cert.get_subject().ST = 'Kyiv_region'
          cert.get_subject().L = 'Kyiv'
          cert.get_subject().O = 'KPI'
          cert.get_subject().OU = 'FI22mn'
          cert.get_subject().CN = 'Daria'
          cert.get_subject().emailAddress = 'dashamelan2311@gmail.com'
          cert.set_serial_number(0)
          cert.gmtime_adj_notBefore(0)
          cert.gmtime_adj_notAfter(10*365*24*60*60)
          cert.set_issuer(cert.get_subject())
          cert.set_pubkey(k)
          cert.sign(k, 'sha512')
          with open('selfsigned.crt', 'wt') as f:
              f.write(crypto.dump_certificate(crypto.FILETYPE_PEM,
      cert).decode('utf-8'))

cert_gen()
```



Самопідписаний (кореневий) сертифікат дійсний 10 років. Щоб встановити довіру до нього, треба додати його в сховище довірених кореневих сертифікатів центрів сертифікації.

Висновки: було досліджено основні задачі, що виникають при програмній реалізації криптосистем. Розглянуто з точки зору вирішення цих задач інтерфейси Crypto API, PKCS 11, а також Microsoft Management Console, було

реалізовано функцію для створення самопідписаного сертифікату методами бібліотеки OpenSSL.