

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ СІКОРСЬКОГО»

ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

Лабораторна робота 4

Дослідження особливостей реалізації існуючих програмних систем, які використовують криптографічні механізми захисту інформації.

Підгрупа 2А

Виконали:

Галіца О.О.

Паршин О.Ю.

Литвиненко Ю.С.

ФІ-22мн

Перевірила:

Байденко П.В.

1 CryptoAPI

1.1 Основні відомості

CryptoAPI – інтерфейс програмування додатків, який забезпечує розробників стандартним набором функцій для роботи з криптопровайдером. Входить до складу операційних систем Microsoft. Більшість функцій CryptoAPI підтримуються, починаючи з Windows 2000.

Модна розділити весь інтерфейс Crypto API на 5 функціональних груп:

1. Базові криптографічні функції:

- функції шифрування/розшифрування даних;
- функції хешування та отримання цифрового підпису даних;
- функції ініціалізації криптопровайдера та роботи з отриманим контекстом;
- функції генерації ключів;
- функції обміну ключами.

2. Функції кодування/декодування. Під кодуванням в даному випадку мається на увазі отримання інформації, закодованої у форматі ASN.1.

3. Функції роботи із сертифікатами.

4. Високорівневі функції обробки криптографічних повідомлень.

5. Низькорівневі функції обробки криптографічних повідомлень.

Криптопровайдером називають незалежний модуль, що забезпечує безпосередню роботу з криптографічними алгоритмами. Кожен криптопровайдер повинен забезпечувати:

1. реалізацію стандартного інтерфейсу криптопровайдера;
2. роботу з ключами шифрування, призначеними для забезпечення роботи алгоритмів, специфічних для цього криптопровайдера;
3. неможливість втручання третіх осіб у схему роботи алгоритмів.

Криптопровайдери реалізуються у вигляді бібліотек, що динамічно завантажуються (DLL). Таким чином, досить важко вплинути на хід алгоритму, реалізованого в криптопровайдері, оскільки всі компоненти криптосистеми Windows повинні мати цифровий підпис, а отже також підписується і DLL криптопровайдера. Не повинно біти можливості змінити алгоритм у криптопровайдері шляхом зміни параметрів цього алгоритму. Таким чином вирішується задача забезпечення цілісності алгоритмів криптопровайдера.

Завдання забезпечення цілісності ключів шифрування вирішується з використанням контейнера ключів. Контейнером ключів називають частину бази даних ключів, яка містить пару ключів для обміну ключами та формування цифрового підпису. Як сховища пар ключів використовують, наприклад, область тимчасової пам'яті, ділянку реєстру, файл на диску, смарт-картки.

1.2 Вразливості

За останній час було виявлено всього дві вразливості CryptoAPI. Спочатку розглянемо криптографічний баг, який є небезпечним для Windows 10, Windows Server 2019 і Windows Server 2016. Уразливість у криптографічній бібліотеці crypt32.dll у Windows, яку назвали **CVE-2020-0601**, полягає в тому, що Windows CryptoAPI (Crypt32.dll) не повністю перевіряє сертифікати ECC (elliptic curve cryptography). Вразливі версії Windows перевіряють три параметри ECC, пропускаючи при цьому четвертий, критичний.

Зловмисник може використовувати цю вразливість, щоб підписати шкідливу програму так, щоб система прийняла файл за легітимний. Найгірше те, що вразливість також може використовуватися для підробки цифрових сертифікатів, що використовуються для шифрованих комунікацій. Фактично успішна експлуатація проблеми дозволяє проводити MitM-атаки і розшифровувати конфіденційну інформацію про підключення користувача.

Ще одну слабкість було виявлено у жовтні 2022 року. Ця вразливість є дуже серйозною та відома як **CVE-2022-34689**. Вона виникла у результаті неправильно реалізованих схем автентифікації, які піддаються атакам спуфінгу. Технічні деталі невідомі, а експлойт недоступний.

Про цю вразливість доволі мало відомостей. Зловмисник може маніпулювати наявним загальнодоступним сертифікатом x.509, щоб підробити свою особу та виконати такі дії, як, наприклад, автентифікація. Також відомо, що відбувається повна втрата цілісності. Наприклад, зловмисник може змінити будь-який (або усі) файли, захищені зараженим компонентом. Можна змінити лише деякі файли, але зловмисна зміна призведе до прямих серйозних наслідків. Зловмисник не потребує доступу до налаштувань або файлів для здійснення атаки. Спеціальних умов доступу або пом'якшувальних обставин також не існує. Зловмисник може повторно провести атаку на вразливий компонент.

1.3 Недоліки CryptoAPI

На жаль, CryptoAPI вже морально застарів, і хоча вже досить давно вийшов CryptoAPI 2.0 для .NET архітектури – це лише для C#, для C/C++ альтернативи немає, і навряд чи буде. Застарілість заключається у відсутності підтримки нових можливостей мов, що перетворюється у функції, які приймають купу параметрів не зовсім зрозуміло для чого, наприклад, ось сигнатура для деякого уявного методу шифрування:

```

BOOL CryptEncrypt(
    HCRYPTKEY hKey,
    HCRYPTHAS hHash,
    BOOL Final,
    DWORD dwFlags,
    BYTE* pbData,
    DWORD* pdwDataLen,
    DWORD dwBufLen
);

```

```

BOOL CryptHashData(HCRYPTHASH hHash,
    BYTE* pbData,
    DWORD dwDataLen,
    DWORD dwFlags
);

```

```

BOOL CryptSignHash(HCRYPTHASH hHash,
    DWORD dwKeySpec,
    LPCTSTR sDescription,
    DWORD dwFlags,
    BYTE* pbSignature,
    DWORD* pdwSigLen
);

```

Багато параметрів для якихось довжин, для додаткових параметрів, які, зрозуміло, передбачалось для ручної роботи з пам'яттю за допомогою вказівників, але в C++ давно є багато вбудованих реалізованих структур даних та в цілому оновлень, які дозволяють працювати з усім цим більш безболісно і зрозуміло (наприклад, `std::unique_ptr`, `std::map`, `std::vector`, `std::pair` тощо).

Тому в даній лабораторній роботі інтерфейси були підігнані під веб-стандарт Web Crypto API.

1.4 Web Cryptography API

З назви зрозуміло, що це Web стандарт, причому для реалізації в JavaScript. В основі всього лижить деякий об'єкт `CryptoSubtle`, через який відбувається взаємодія з криптографічними функціями. Наведемо його інтерфейс:

```

interface SubtleCrypto {
    Promise<any> encrypt(
        AlgorithmIdentifier algorithm,
        CryptoKey key,
        BufferSource data
    );

    Promise<any> decrypt(
        AlgorithmIdentifier algorithm,
        CryptoKey key,
        BufferSource data
    );

    Promise<any> sign(
        AlgorithmIdentifier algorithm,

```

```

        CryptoKey key,
        BufferSource data
    );

    Promise<any> verify(
        AlgorithmIdentifier algorithm,
        CryptoKey key,
        BufferSource signature,
        BufferSource data
    );

    Promise<any> digest(
        AlgorithmIdentifier algorithm,
        BufferSource data
    );

    Promise<any> generateKey(
        AlgorithmIdentifier algorithm,
        boolean extractable,
        sequence<KeyUsage> keyUsages
    );
}

```

Одразу видно, наскільки сигнатури стали чистішими і зрозумілішими, у функції передаються лише ті параметри, які чисто інтуїтивно і очікуєш побачити. І хоча це стандарт для JavaScript – можливості сучасного C++ дозволяють реалізувати подібний інтерфейс без значних труднощів, що і було зроблено в рамках цієї лабораторної роботи.

2 Результати

В ході лабораторної роботи була реалізована криптосистема Ель-Гамала з параметризованою довжиною ключа. В якості допоміжної бібліотеки для роботи з довгою арифметикою була використана бібліотека OpenSSL.

Тестове повідомлення: D4D2110984907B5625309D956521BAB4157B8B1ECE04043249A3D379AC112E5B9AF44E721E148D88A942744CF56A06B92D28A0DB950FE4CED2B41A0BD38BCE7D0BE1055CF5DE38F2A588C2C9A79A75011058C320A7B661C6CE1C36C7D870758307E5D2CF07D9B6E8D529779B6B2910DD17B6766A7EFEE215A98CAC300F2827DB

```

Microsoft Visual Studio Debu  X  +  v

Input message that will be encrypted:
D4D2110984907B5625309D956521BAB4157B8B1ECE04043249A3D379AC112E5B9AF44E721E148D88A942744CF56A06B92D28A0DB950FE4CED2B41A0BD38
BCE7D0BE1055CF5DE38F2A588C2C9A79A75011058C320A7B661C6CE1C36C7D870758307E5D2CF07D9B6E8D529779B6B2910DD17B6766A7EFEE215A98CAC
300F2827DB

c1:
BA77C00A34E3DE16DED833B9658779991C7DF01E25F2B2E00A985701754804655A51918FD59359FD103DC8F700E8C8B2386F97800D0B84C5AEEC521B443
374FF10BD51B2B7F02E912E65F3C28E2421847BF5B3D6DCBE6997810AD7AD6CE8682BEBACE5E1B4644879C428FAEED322960258BFAAFC987DA6D06C8CC6
E6B42012C8DBB7F5FD90514F579EBE407AB76547B8356A6A05B350374287C8682F65BA3CBFBF8C2A57DD4B9750DCE55BA805F697EF0273FCCA6AAB6513B
65E05AAEDF6E2DC9FF6185327A942B8F74E1A3EEC1984B16AB9883D26F2277AFFBE2173C662DAD72F2A511031A737BC901160B631907184E580CBD73331
F874A01622DCB37F1D0C

c2:
030A4FEF8FA403AEDC06083F79990A6003B02910A76092A69990DFD9487BE7EB8BF8D0D6C1FD44B4EB77932D66B3FE2BCC31A8DC6A062FA36CAB8A7FB5D
CBE4A6E13497BDF4735F9E50150F0A7CDD15938A05401CCFFC54D6661DAFF727A9A72962F03AC8CA0C1AF690141014BD39711E1D2C3EBBA5F1ACA03C2E2
F4376CC5C0B6635DC6347DD65E6E3F9BCD72FD9B712F3EEDEF3A57C10B4BB7F6A2BF0EAB417D7CBDC3634FEDE2B5369B1569BB288B50552B6A0F13B2621
7AD5A3AC8ACD617C87C9AA2549B31A7885603CF493C547DEDE2773C1C94A835AAB0C76952BDEDC9B5997CDD1E32FBD57B6AFE186320361ACF5D20CFA6E1
8E3468341BE6C275891A

-----
Input message that will be decrypted:
c1: BA77C00A34E3DE16DED833B9658779991C7DF01E25F2B2E00A985701754804655A51918FD59359FD103DC8F700E8C8B2386F97800D0B84C5AEEC521
B443374FF10BD51B2B7F02E912E65F3C28E2421847BF5B3D6DCBE6997810AD7AD6CE8682BEBACE5E1B4644879C428FAEED322960258BFAAFC987DA6D06C
8CC6E6B42012C8DBB7F5FD90514F579EBE407AB76547B8356A6A05B350374287C8682F65BA3CBFBF8C2A57DD4B9750DCE55BA805F697EF0273FCCA6AAB6
513B65E05AAEDF6E2DC9FF6185327A942B8F74E1A3EEC1984B16AB9883D26F2277AFFBE2173C662DAD72F2A511031A737BC901160B631907184E580CBD7
3331F874A01622DCB37F1D0C

c2: 030A4FEF8FA403AEDC06083F79990A6003B02910A76092A69990DFD9487BE7EB8BF8D0D6C1FD44B4EB77932D66B3FE2BCC31A8DC6A062FA36CAB8A7
FB5DCBE4A6E13497BDF4735F9E50150F0A7CDD15938A05401CCFFC54D6661DAFF727A9A72962F03AC8CA0C1AF690141014BD39711E1D2C3EBBA5F1ACA03
C2E2F4376CC5C0B6635DC6347DD65E6E3F9BCD72FD9B712F3EEDEF3A57C10B4BB7F6A2BF0EAB417D7CBDC3634FEDE2B5369B1569BB288B50552B6A0F13B
26217AD5A3AC8ACD617C87C9AA2549B31A7885603CF493C547DEDE2773C1C94A835AAB0C76952BDEDC9B5997CDD1E32FBD57B6AFE186320361ACF5D20CFA
A6E18E3468341BE6C275891A

m:
D4D2110984907B5625309D956521BAB4157B8B1ECE04043249A3D379AC112E5B9AF44E721E148D88A942744CF56A06B92D28A0DB950FE4CED2B41A0BD38
BCE7D0BE1055CF5DE38F2A588C2C9A79A75011058C320A7B661C6CE1C36C7D870758307E5D2CF07D9B6E8D529779B6B2910DD17B6766A7EFEE215A98CAC
300F2827DB
-----

```

Рис. 1: Шифрування та розшифрування

```

Microsoft Visual Studio Debu  X  +  v
-----
Input message that will be signed:
D4D2110984907B5625309D956521BAB4157B8B1ECE04043249A3D379AC112E5B9AF44E721E148D88A942744CF56A06B92D28A0DB950FE4CED2B41A0BD38
BCE7D0BE1055CF5DE38F2A588C2C9A79A75011058C320A7B661C6CE1C36C7D870758307E5D2CF07D9B6E8D529779B6B2910DD17B6766A7EFEE215A98CAC
300F2827DB

r:
D12FBF4D90A93E642BC586851417318F9926BA2E595464CE28CA06B3DB73096433B1E8CE8C25E984E56B615DAB2248EE43D7114049D6AE6575D2FF8EA8D
8B8109A4BE7DFBEC3B233F18EFC26A446E3FCBE570D0807AC70593C82182402E9581C8214AAF0E9E99C4218A9277C48F1570CD5C97E4D54C89F0BB928A9
944713B80E70182B53E8FE7E5F11BD143AB582CC14B0DC8FCCC043237266CA879F931A3806D5F1232F219FE170DB4A97A1A319D1A1DD39D7D5F3039E22E
A39EF0B213EF894053F1EBD07937F8F8222AB3649A498448BE11F8FF7C03F901BE25CAC803F88F1377917A5AADEF243D9FDF2B48D698A9F3696607B545
542F1AE3450DD749236A

s:
158564B3861653734F1DD67E611BC2D55E9819387E92EF6674639CA02518648D3A3762E843C7C2CBE0EC9A46BDD4714367BF9261C2BFB703545D338C1C1
AA926BAE2BF34B81980225DDBD6A30E1FAF44BECF32778F1278ACF15C114B819DABB065BEE52FA5D0C7F0A95EA8E62CB3CBFC907499FC9B8C8052BF613
CB91CC303D7E8DD7765C7EF8BE9046597CF8ACB7A454515F16CC47A6B26C3816AD20799DDF938A72D3B206ADB6709ECFE31E78EFA9143B396F87FECE57F
633AE7EFAFC9F60EB0F561094AEF8077BFF07A29B1E70769B6B8E65BD9E138630BC1CF07026CC6422B2D60296E6E1D41ADB8E4CB22C7B9BC01ABD945B6B
77CD9AB469CF0EE2E597

-----
Input message and sign that will be verified:
m: D4D2110984907B5625309D956521BAB4157B8B1ECE04043249A3D379AC112E5B9AF44E721E148D88A942744CF56A06B92D28A0DB950FE4CED2B41A0B
D38BCE7D0BE1055CF5DE38F2A588C2C9A79A75011058C320A7B661C6CE1C36C7D870758307E5D2CF07D9B6E8D529779B6B2910DD17B6766A7EFEE215A98
CAC300F2827DB

r: D12FBF4D90A93E642BC586851417318F9926BA2E595464CE28CA06B3DB73096433B1E8CE8C25E984E56B615DAB2248EE43D7114049D6AE6575D2FF8E
A8D8B8109A4BE7DFBEC3B233F18EFC26A446E3FCBE570D0807AC70593C82182402E9581C8214AAF0E9E99C4218A9277C48F1570CD5C97E4D54C89F0BB92
8A9944713B80E70182B53E8FE7E5F11BD143AB582CC14B0DC8FCCC043237266CA879F931A3806D5F1232F219FE170DB4A97A1A319D1A1DD39D7D5F3039E
22EA39EF0B213EF894053F1EBD07937F8F8222AB3649A498448BE11F8FF7C03F901BE25CAC803F88F1377917A5AADEF243D9FDF2B48D698A9F3696607B
545542F1AE3450DD749236A

s: 158564B3861653734F1DD67E611BC2D55E9819387E92EF6674639CA02518648D3A3762E843C7C2CBE0EC9A46BDD4714367BF9261C2BFB703545D338C
1C1AA926BAE2BF34B81980225DDBD6A30E1FAF44BECF32778F1278ACF15C114B819DABB065BEE52FA5D0C7F0A95EA8E62CB3CBFC907499FC9B8C8052BF
613CB91CC303D7E8DD7765C7EF8BE9046597CF8ACB7A454515F16CC47A6B26C3816AD20799DDF938A72D3B206ADB6709ECFE31E78EFA9143B396F87FECE
57F633AE7EFAFC9F60EB0F561094AEF8077BFF07A29B1E70769B6B8E65BD9E138630BC1CF07026CC6422B2D60296E6E1D41ADB8E4CB22C7B9BC01ABD945
B6B77CD9AB469CF0EE2E597

Sign is correct: 1
-----

D:\Documents\Projects\cryptographic-mechanisms\lab-3\x64\Release\lab-3.exe (process 17500) exited with code 1.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console
when debugging stops.
Press any key to close this window . . .|

```

Рис. 2: Підписування та перевірка підпису