Лабораторна робота №1
«Вибір та реалізація базових фреймворків та бібліотек»

Олефір П., Бещук А., Попов А.

Мета роботи: Вибір бібліотеки реалізації основних криптографічних примітивів з точки зору їх ефективності за часом та пам'яттю для різних програмних платформ. А саме порівняти бібліотеки OpenSSL, crypto++, CryptoLib, PyCrypto для розробки гібридної криптосистеми під Linux платформу

1 Хід роботи

1. Описати кожну з бібліотек та сферу їх використання;
2. Дослідити, які криптографічні примітиви підтримує кожна з бібліотек: алгоритми симетричного та асиметричного шифрування, хеш-функції, генерування випадкових послідовностей;
3. Вибрати бібліотеку для розробки гібридної криптосистеми під Linux платформу. Обгрунтування вибору.

2 Вступ

Оглядаючи набіл бібліотек треба зауважити, що вони мають різні реалізації, особливості. Також розробка бібліотек відбувалась для вирішення конкретних задач.

Оскільки ми не розглядаємо їх як окремий програмний продукт, то немає сенсу розглядати побудовані на їх основі додатки та програмні модулі. У ролі бібліотеки вони представляють собою лише певний набір примітивів

для виконання криптографічних операцій над інформацією у комп'ютерних системах, репрезентованою у вигляді бітових послідовностей. Для спрощення подальшого опису будемо сприймати бітові послідовності як файли у операційній системі так і будь-яку інформацію у комп'ютерних системах, оскільки в кінцевому випадку вся інформація у них репрезентується як послідовність бітів.

Оскільки на даний момент криптографічні ключі, які використовуються в криптографічних системах, мають апаратні реалізації, то необхідно звернути увагу на можливість їхнього використання.

Взаємодія апаратних ключів(токенів) представляє собою набір унікальних та пропрієтарних рішень специфічних для кожного виробника, з окремим набором програмних утиліт для налаштування самих токенів. Тобто ми можемо розглядати апаратні ключі або у вигляді абстракції файлу(або бітової послідовності) або як набір специфічних варіантів рішень. Для даної роботи розглянемо їх у вигляді файлів, оскільки огляд списку виробників та моделей апаратних ключів не є на меті.

3 OpenSSL

OpenSSL [1] — це придатний до комерційного використання і повнофункціональний набір інструментів для протоколів Transport Layer Security (TLS) і Secure Sockets Layer (SSL). Це також бібліотека криптографії загального призначення.

На даний час OpenSSL складається з чотирьох частин:

1. Libcrypto — це основна бібліотека для реалізації численних криптографічних примітивів та автентифікації. Крім того, вона надає набір допоміжних сервісів для реалізації таких протоколів, як CMS та OCSP;

2. Engine — функціонал libcrypto може бути розширений за допомогою API Engine. Зазвичай engines – це динамічно завантажувані модулі, які зареєстровані у libcrypto і використовують доступні інтерфейси для забезпечення реалізації криптографічного алгоритму. Зазвичай це альтернативні реалізації алгоритмів, які вже надаються libcrypto і використовуються (наприклад, для забезпечення апаратного прискорення алгоритму), але вони можуть включати алгоритми, не реалізовані в стандартному OpenSSL. Деякі engines надаються як частина дистрибутиву OpenSSL, а деякі — зовнішнім сторонам (наприклад ГОСТ);

3. Libssl — це бібліотека залежить від libcrypto та реалізує протоколи TLS та DTLS;

4. Applications (додатки) — це набір інструментів командного рядка, які використовують базові компоненти libssl та libcrypto для забезпечення набору криптографічних та інших функцій, таких як:

   (а) Генерація та перевірка ключів та параметрів;

   (б) Створення та перевірка сертифікатів;

   (в) Інструменти тестування SSL / TLS;

   (г) Інспекція ASN.1.

Сфера використання

OpenSSL широко використовується для забезпечення захищеного зв'язку у мережі інтернет через наявність додатків, котрі можуть виконувати необхідні функції та є легко конфігурованими. Як бібліотека криптографічних примітивів вона використовується рідше.

Наявні примітиви

Виходячи з документації проекту, повний перелік алгоритмів гешування та шифрування в OpenSSL може змінюватись в залежності від використання сторонніх engines, котрі можуть бути створені, незалежними від OpenSSL, розробниками.

Наявність чи відсутність реалізації певних криптографічних функцій може також бути обумовлена версією бібліотеки, оскільки проект активно підтримується спільнотою та постійно оновлюється.

Повний список реалізованих криптографічних примітивів у бібліотеці доволі великий, тож наведемо частковий [2]:

Табл. 1: Криптографічні примітиви бібліотеки OpenSSL

Алгоритми шифрування	AES, Camellia, 3DES, Blowfish, CAST5, IDEA, ARIA
Функції гешування	MD5, SHA-1, SHA-2, SHA-3, RIPEMD-160, Tiger, Whirlpool, BLAKE2
Генерація та обмін ключами	ECDH, DH, DSA, RSA
Підтримувані стандарти апаратних ключів	PKCS 11

Приклади роботи з бібліотекою

```
Генерація ключів RSA
pRSA = RSA_generate_key(GetKeyLength(), CRYPTO_RSA_KEY_EXP, NULL, NULL);
```

```
Шифрування алгоритмом RSA
int result = RSA_private_encrypt(
  plaintext_size,
  (unsigned char *)plaintext,
  (unsigned char *)ciphertext,
  pRSA,
  RSA_PKCS1_PADDING
);

Розшифровка шифротексту RSA
int result = RSA_private_decrypt(
  GetCipherTextSize(),
  (unsigned char *)ciphertext,
  (unsigned char *)plaintext,
  pRSA,
  RSA_PKCS1_PADDING
);
```

4 crypto++

crypto++ [3] — безкоштовна бібліотека класів криптографічних алгоритмів та схем класу C++. Бібліотека повністю підтримує 32-розрядні та 64-розрядні архітектури для багатьох основних операційних систем та платформ.

Сфера використання

Crypto++ широко використовується в академічних колах, студентських проектах, відкритих комерційних та некомерційних проектах. Має широкі функціональні можливості та високу швидкість роботи. На відміну від бібліотеки OpenSSL не має чіткої спеціалізації на якійсь окремій сфері використання.

Наявні примітиви

Бібліотека розробляється та оновлюється з 1995 року, тож повний список примітивів буде доволі великий. Наведемо частину з доступних в бібліотеці реалізацій, маючих відношення до нашої роботи.

Приклади роботи з бібліотекою

```
Генерація ключів RSA
InvertibleRSAFunction params;
params.GenerateRandomWithKeySize(rng, 3072);
RSA::PrivateKey privateKey(params);
RSA::PublicKey publicKey(params);
```

Табл. 2: Криптографічні примітиви бібліотеки crypto++

Алгоритми шифрування	AES (Rijndael), RC6, MARS, Twofish, Serpent, CAST-256 ARIA, Blowfish, Camellia, CHAM, HIGHT, IDEA, Kalyna (128/256/512), LEA, SEED, RC5, SHACAL-2, SIMECK, SIMON (64/128), Skipjack, SPECK (64/128), Simeck, SM4,Threefish (256/512/1024), Triple-DES (DES-EDE2 and DES-EDE3), TEA, XTEA
Функції гешування	BLAKE2b, BLAKE2s, Keccack (F1600), SHA-1, SHA-2, SHA-3, SHAKE (128/256), SipHash, Tiger, RIPEMD (128/160/256/320), SM3, WHIRLPOOL
Генерація та обмін ключами	Diffie-Hellman (DH), Unified Diffie-Hellman (DH2), Menezes-Qu-Vanstone (MQV), Hashed MQV (HMQV), Fully Hashed MQV (FHMQV), LUCDIF, XTR-DH
Підтримувані стандарти апаратних ключів	—

```
Шифрування алгоритмом RSA
RSAES_OAEP_SHA_Encryptor e(publicKey);

StringSource ss1(plain, true,
    new PK_EncryptorFilter(rng, e,
        new StringSink(cipher)
   )
);

Розшифровка шифротексту RSA
RSAES_OAEP_SHA_Decryptor d(privateKey);

StringSource ss2(cipher, true,
    new PK_DecryptorFilter(rng, d,
        new StringSink(recovered)
   ) // PK_DecryptorFilter
); // StringSource
```

5 CryptLib

CryptLib [4] складається з набору багаторівневих інструментів безпеки та пов'язаних з ними інтерфейсів програмування, що забезпечують інтегрований набір можливостей захисту інформації та зв'язку. Подібно до еталонної моделі мережі, cryptlib містить ряд шарів, які забезпечують кожен рівень абстракції, причому вищі шари спираються на можливості нижчих шарів.

Сфера використання

Оскільки бібліотека написана на мові низького рівня, то може бути використана для імплементації криптографічних систем для вбудованих систем та мікроконтролерів. Як і Crypto++ не має чіткої направленості на окрему сферу використання, проте відрізняється від неї типом ліцензування. Також цікавою особливістю цієї бібліотеки є наявність деталізованої документації [5] з прикладами використання.

Табл. 3: Криптографічні примітиви бібліотеки CryptLib

Алгоритми шифрування	AES, Blowfish, CAST, DES, Triple, IDEA, RC2, RC4, RC5, Skipjack
Функції гешування	MD2, MD4, MD5, RIPEMD-160, SHA-1, SHA-2
Генерація та обмін ключами	Diffie–Hellman, DSA, ECDSA, ECDH, Elgamal, RSA
Підтримувані стандарти апаратних ключів	PKCS 11

Приклади роботи з бібліотекою

```
Генерація ключів RSA
CRYPT_CONTEXT privKeyContext;
cryptCreateContext( &privKeyContext, cryptUser, CRYPT_ALGO_RSA );
cryptSetAttributeString( privKeyContext, CRYPT_CTXINFO_LABEL, label,
labelLength );
cryptGenerateKey( privKeyContext );


Шифрування алгоритмом RSA
cryptEncrypt( privKeyContext, buffer, length );

Розшифровка шифротексту RSA
cryptDecrypt( cryptContext, buffer, length );
```

6 PyCrypto

PyCrypto [6] — це пакет, що містить різні криптографічні модулі для мови програмування Python. Набір інструментів криптографії Python призначений забезпечити надійну та стабільну базу для написання програм Python, які потребують криптографічних функцій.

Основна ціль цієї бібліотеки — надати простий, послідовний інтерфейс для криптографічних алгоритмів. Деякі з цих інтерфейсів були модифіковані як PEP(Python Enhancement Proposal) 247, "API для криптографічних геш-функцій"та PEP 272, "API для алгоритмів блокового шифрування".

Деякі модулі бібліотеки реалізовані в C для підвищення продуктивності; інші написані на Python для зручності модифікації. Як правило, низькорівневі функції, такі як шифри та геш-функції, пишуться на мові C, тоді як менш критичні функції були написані на Python. В даний час криптографічні реалізації є прийнятно швидкими, але не надзвичайно хорошими.

Сфера використання

Бібліотека використовується для вирішення прикладних задач з упором на необхідність використання Python, котрий на поточний момент є одним з найпопулярніших мов з широким спектром використання та великою спільнотою.

Табл. 4: Криптографічні примітиви бібліотеки PyCrypto

Алгоритми шифрування	AES, ARC2, Blowfish, CAST, DES, DES3, IDEA, RC5
Функції гешування	MD2, MD4, MD5, RIPEMD, SHA1, SHA256
Генерація та обмін ключами	RSA, ElGamal, DSA, qNEW
Підтримувані стандарти апаратних ключів	—

Приклади роботи з бібліотекою

```
Генерація ключів RSA
privatekey = RSA.generate(modulus_length, Random.new().read)
publickey = privatekey.publickey()

Шифрування алгоритмом RSA
encrypted_msg = publickey.encrypt(a_message, 32)[0]
encoded_encrypted_msg = base64.b64encode(encrypted_msg)
```

```
Розшифровка шифротексту RSA
decoded_encrypted_msg = base64.b64decode(encoded_encrypted_msg)
decoded_decrypted_msg = privatekey.decrypt(decoded_encrypted_msg)
```

7 Висновки

Для подальшого виконання завдань з циклу лабораторних робіт було вирішено використати бібліотеку PyCrypto, оскільки вона є найбільш простою та не потребує значного часу на ознайомлення з документацією, але попри свою простоту має необхідні примітиви для побудови гібридної криптосистеми.

Інші бібліотеки реалізовані на C та C++, котрі являються більш складними в роботі мовами. Попри можливу наявність переваг у швидкості роботи, об'єму використовуваної пам'яті, мають значно складніші інтерфейси взаємодії та вищий поріг входу

Література

[1] OpenSSL: https://www.openssl.org/docs/OpenSSLStrategicArchitecture.html

[2] OpenSSL: The Open Source toolkit for SSL/TLS https://www.openssl.org/docs/man1.0.2/man1/ciphers.html

[3] crypto++: https://www.cryptopp.com/

[4] CryptLib: Security toolkit https://www.cs.auckland.ac.nz/ pgut001/cryptlib/

[5] CryptLib: https://www.cryptlib.com/downloads/manual.pdf

[6] PyCrypto: https://www.dlitz.net/software/pycrypto/doc/acknowledgements