Міністерство освіти і науки України
Національний технічний університет України
"Київський політехнічний інститут імені Ігоря Сікорського"
Фізико-технічний інститут

Лабораторна робота №4

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

Виконали: студенти гр. ФІ-32мн,
Костюк К.М., Панасюк Є.С, Пелешенко Л.І.

Перевірив:
Кудін А.М

Київ – 2023

Мета лабораторної роботи: Дослідити основні задачі, що виникають при програмній реалізації криптосистем.

Постановка задачі:

Запропонувати методи вирішення задачі контролю доступу до ключової інформації, що зберігається в оперативній пам’яті ЕОМ для 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 1А. Реалізація для інтелектуальної картки, токена.

Програмна реалізація криптосистем

Криптосистеми – це складний механізм, який використовує криптографічні алгоритми для забезпечення конфіденційності, цілісності та автентифікації даних. Програмна реалізація цих систем грає вирішальну роль у забезпеченні їхньої ефективності та безпеки. У зв’язку зі ростанням кількості цифрових транзакцій та обміну конфіденційною інформацією в Інтернеті виникає потреба у надійних та ефективних криптосистемах для забезпечення безпеки даних. При цьому можуть виникати різноманітні задачі, пов'язані з надійністю, ефективністю та коректністю криптографічних операцій.

Для сучасних криптографічних систем захисту інформації сформульовані загальноприйняті вимоги: зашифроване повідомлення повинно піддаватися читанню тільки при наявності ключа; число операцій, необхідних для визначення використаного ключа шифрування за фрагментом шифрованого повідомлення і відповідного йому відкритого тексту, має бути не менше загального числа можливих ключів; число операцій, необхідних для розшифрування інформації шляхом перебору різноманітних ключів повинно мати сувору нижню оцінку і не виходити за межі можливостей сучасних комп'ютерів (з урахуванням можливості використання мережевих обчислень); знання алгоритму шифрування не повинно впливати на надійність захисту; незначна зміна ключа повинно приводити до істотної зміни виду зашифрованого повідомлення навіть при використанні одного і того ж ключа; структурні елементи алгоритму шифрування повинні бути незмінними; додаткові біти, що вводяться в повідомлення в процесі шифрування, повинні бути повністю та надійно сховані в зашифрованому тексті; довжина шифрованого тексту повинна бути рівною довжині вихідного тексту; не повинно бути простих і легко встановлюваних залежностей між ключами, що послідовно використовуються в процесі шифрування.

Почати програмну реалізацію криптосистем необхідно із вибору криптографічних алгоритмів, які будуть використовуватись. Ця задача є ключовою, адже неправильний вибір може призвести до вразливостей та атак. Тож необхідно дуже ретельно підходити до вибору, аби алгоритм відповідав сучасним стандартам безпеки та виконував поставлену вами задачу. Важливо враховувати можливість атак, що спрямовані на різноманітні компоненти системи, включаючи самі криптографічні алгоритми, ключі, процеси обміну ключами та інші елементи. Базові аспекти, якими має володіти криптосистема – нерозрізнюваність, семантична стійкість, стійкість до перетворень.

Наступним на що необхідно звернути увагу є керування ключами. Даний аспект є надважливим для забезпечення надійності інформації у криптосистемі. Незалежно від того, чи маємо справу із системами шифрування, цифровими підписами чи іншими криптографічними методами, ефективне керування ключами є важливим для забезпечення конфіденційності, цілісності та доступності даних. Важливо використовувати надійні та криптографічно стійкі алгоритми для генерації ключів, такі що генерували б ключі достатньо довгими та випадковими для утруднення спроб їхнього вгадування; зберігалися в безпечному середовищі, недоступному для несанкціонованого доступу. Необхідно використовувати безпечні протоколи та алгоритми для передачі ключів між сторонами, які допоможуть уникнути атак та перехоплення, та продумати стратегію оновлення, захисту від втрати, видалення застарілих або скомпрометованих ключів.

Наступним кроком є взаємодія програмної частини системи із апаратним забезпеченням, що може надавати захист від фізичних атак, таких як витоки електромагнітних сигналів, впливи на напругу живлення, чи спроби неправомірного доступу; покращення швидкодії та оптимізації криптографічних операцій. Одним із прикладів є апаратні генератори випадкових чисел, які можуть забезпечити високий рівень ентропії для генерації безпечних ключів та інших параметрів криптографічних алгоритмів. Таким чином застосування апаратних засобів може значно поліпшити захист від різноманітних загроз, забезпечуючи високий рівень безпеки інформації.

Криптосистему варто забезпечити моніторингом, тобто постійним процесом спостереження, збору та аналізу даних з метою визначення стану системи, виявлення аномалій, атак або неправомірної активності, а також для забезпечення ефективного функціонування та безпеки. Це є ключовою складовою для вчасного виявлення та реагування на потенційні загрози.

Контроль доступу до ключової інформації

Правильний контроль доступу до ключової інформації в оперативній пам'яті і забезпечення правильності функціонування програм криптографічної обробки є критичними для забезпечення інформаційної безпеки. Для забезпечення цих аспектів необхідно використовувати механізми операційної системи або апаратні засоби для шифрування областей пам'яті, де зберігаються ключі. А також забезпечити, щоб розшифрування відбувалося тільки в потрібний момент. При моделювання доступу необхідно визначити різні ролі та рівні доступу для користувачів та програм. Наприклад, можна використовувати принцип найменших привілеїв (Principle of Least Privilege), щоб обмежити доступ до ключової інформації. Також однією із задач забезпечення контролю доступу є моніторинг. Цей механізм, разом із веденням журналу подій, допоможе виявити неправильний або несанкціонований доступ до ключових даних та виявити потенційні загрози

Допомогти вирішити дані задачі контролю доступу допоможе використання інтелектуальної картки або токену. Використовуйте карти з апаратною підтримкою криптографії (наприклад, JavaCard) для зберігання та обробки ключової інформації. Захистіть з'єднання між системою та картою шифруванням та аутентифікацією. Або ж використовуйте апаратно захищені токени для зберігання ключової інформації. Забезпечте апаратну ізоляцію для запобігання фізичному доступу.

Розглянемо детальніше, що таке інтелектуальна карта та токен, як вони працюють, а також ознайомимось із інтерфейсами Crypto API, PKCS 11.

Інтелектуальна картка (смарт-карта)

Інтелектуальна картка (часто називається також "смарт-карта") - це пластикова картка з вбудованим мікропроцесором та/або спеціальним чіпом, які надають їй додаткові функціональні можливості порівняно зі звичайними магнітними смужками чи фізичними маркерами. Основні характеристики інтелектуальних карток включають мікропроцесор або чіп, пам'ять, інтерфейси зв'язку, шифрування та безпека та інші додаткові функції.

Інтелектуальні картки можуть використовуватися в контексті ЕОМ для різних цілей, особливо у сферах, де потрібно забезпечити безпеку та автентифікацію. Розглянемо деякі способи.

- Автентифікація та контроль доступу: інтелектуальні картки використовуються для автентифікації користувачів на ЕОМ. Карта може містити унікальні ідентифікатори, ключі або біометричні дані, які

використовуються для перевірки ідентичності та надання доступу до ресурсів.

- Безпека інформації: можуть зберігати конфіденційну інформацію, таку як шифровані ключі чи сертифікати, що використовуються для захисту даних на ЕОМ від несанкціонованого доступу.
- Безконтактна ідентифікація: інтелектуальні картки використовують технології безконтактної передачі даних, такі як RFID (радіочастотна ідентифікація), що може бути використано для зручної ідентифікації користувачів без прямого контакту з ЕОМ.
- Електронний гаманець: використовуються як електронні гаманці для проведення безготівкових операцій, таких як оплата за товари та послуги через ЕОМ.
- Мережева безпека: можуть використовуватися для забезпечення безпеки при підключенні до мережі, дозволяючи аутентифікацію користувачів або пристроїв, щоб забезпечити захист від несанкціонованого доступу.

Узагальнюючи, можна сказати, що інтелектуальні картки в контексті ЕОМ використовуються для забезпечення безпеки, автентифікації та ефективного взаємодії користувачів з обчислювальною технікою.

Токен

Токен має багато значень, проте сьогодні розглянемо саме криптографічний токен, або як його ще називають – USB-ключ або апаратний токен. Він виконує роль своєрідного «сейфу» для електронного підпису – компактний пристрій, призначений для забезпечення інформаційної безпеки користувача, який також використовують для ідентифікації його власника, безпечного віддаленого доступу до інформаційних ресурсів тощо. Зазвичай це фізичний пристрій, що використовують для спрощення аутентифікації. Крім того, цей термін може стосуватися й програмних токенів, які видають користувачеві після успішної авторизації та які є ключем для доступу до певних програм. Найчастіше реалізовано такий пристрій у вигляді USB-накопичувача, що використовується для спрощення аутентифікації. Основними характеристиками криптографічних токенів є:

- зберігання криптографічних ключів (таких як приватні ключі для асиметричного шифрування чи цифрового підпису);
- використання для аутентифікації користувача або системи, дозволяючи лише власнику токена виконувати криптографічні операції;
- генерація випадкових чисел для створення надійних ключів та інших криптографічних параметрів;

- виконання операцій шифрування та підпису за допомогою вбудованих або збережених ключів;
- захист від фізичних атак, таких як вилучення ключів або спроби змінити апаратне забезпечення токена.

Більшість токенів мають стандартизовані інтерфейси, що вбудовані в сучасні операційні системи. Проте, для їхнього коректного функціонування може знадобитися встановлення сторонніх бібліотек чи драйверів.

І смарт-карта, і криптографічний токен – типи фізичних пристроїв, які використовуються для забезпечення безпеки та виконання криптографічних операцій. Розглянемо основні відмінності між ними.

По-перше, функціональність. Смарт-карта – це карта з вбудованим мікропроцесором та пам'яттю, яка може виконувати криптографічні операції, здійснювати аутентифікацію, зберігати ключі та інші конфіденційні дані. В той час як криптографічний токен може бути апаратним пристроєм або програмним забезпеченням, яке виконує функції збереження ключів та виконання криптографічних операцій, але він може не мати екрану чи клавіатури.

По-друге, форм-фактор. Смарт-карта – зазвичай це пластикова карта, яка виглядає подібно до звичайної банківської карти та може бути вбудована в кардридер або зчитувач карт. Криптографічний токен – це може бути USB-токен, який підключається до порту USB на комп'ютері, або інший апаратний пристрій, що виконує аналогічні функції.

По-третє, призначення. Смарт-карта зазвичай використовується для видачі ідентифікаційних карт, банківських карт, SIM-карт для мобільних телефонів тощо. А криптографічний токен – для забезпечення безпеки в інформаційних технологіях, включаючи захист ключів, аутентифікацію та виконання криптографічних операцій у комп’ютерних системах.

Обираючи між смарт-картою та криптографічним токеном, важливо враховувати вимоги конкретного використання та потреби в безпеці. Також варто врахувати, що вартість смарт-карти зазвичай є менш дорогою для виробництва, але вартість зчитувачів карт може додавати витрати. Тоді як криптографічний токен може бути трошки дорожчим, але не потребує додаткового обладнання для зчитування.

Crypto API

Cryptographic Application Programming Interface (спеціальний інтерфейс програмування криптографічних програм Microsoft Windows) – це інтерфейс програмування додатків, що входить до складу операційних систем Microsoft Windows, який надає послуги, що дозволяють розробникам захищати програми на основі Windows за допомогою криптографії. Це набір динамічно пов'язаних бібліотек, що забезпечує рівень абстракції, який ізолює програмістів від коду, що використовується для шифрування даних. Вперше було представлено в Windows NT 4.0 і вдосконалено в наступних версіях.

CryptoAPI підтримує криптографію як з відкритим ключем, так і з симетричним ключем, хоча постійні симетричні ключі не підтримуються. Він містить функції для шифрування та дешифрування даних, а також для автентифікації за допомогою цифрових сертифікатів. Він також містить криптографічно захищену функцію генератора псевдовипадкових чисел CryptGenRandom.

CryptoAPI працює з низкою CSP (постачальників криптографічних послуг), встановлених на машині. CSP — це модулі, які виконують фактичну роботу з кодування та декодування даних, виконуючи криптографічні функції. Постачальники HSM можуть надавати CSP, який працює з їх апаратним забезпеченням.

Windows Vista містить оновлення CryptoAPI, відомого як Cryptography API: Next Generation (CNG). Він має кращий факторінг API, щоб дозволити тим самим функціям працювати з використанням широкого спектру криптографічних алгоритмів, і включає ряд новіших алгоритмів, які є частиною пакету B Агентства національної безпеки (NSA). Він також гнучкий і підтримує підключення користувацьких криптографічних API до середовища виконання CNG. Однак постачальники зберігання ключів CNG все ще не підтримують симетричні ключі. CNG працює як у режимі користувача, так і в режимі ядра, а також підтримує всі алгоритми з CryptoAPI. Постачальник Microsoft, який реалізує CNG, міститься в Bcrypt.dll.

CNG також підтримує криптографію за еліптичною кривою, яка завдяки використанню коротших ключів для того самого очікуваного рівня безпеки, є більш ефективною, ніж RSA. CNG API інтегрується з підсистемою смарт-картки, включаючи модуль базового постачальника криптографічних послуг смарт-картки (Base CSP), який інкапсулює API смарт-картки. Виробники смарт-карт просто повинні зробити свої пристрої сумісними з цим, а не створювати рішення з нуля.

CNG також додає підтримку Dual_EC_DRBG, генератора псевдовипадкових чисел, визначеного в NIST SP 800-90A, який може наражати користувача на прослуховування Агентством національної безпеки, оскільки він містить клептографічний бекдор, якщо розробник не забуде створити нові базові точки за допомогою інший криптографічно захищений генератор псевдовипадкових чисел або справжній генератор випадкових чисел, а потім опублікувати згенероване початкове значення, щоб видалити бекдор NSA. Він використовується лише тоді, коли це вимагається явно.

CNG також замінює PRNG за замовчуванням на CTR_DRBG, використовуючи AES як блоковий шифр, оскільки попередній RNG, який визначено в нині заміненому FIPS 186-2, базується на DES або SHA-1, обидва з яких були зламані. CTR_DRBG є одним із двох алгоритмів у NIST SP 800-90, схваленим Schneier, іншим є Hash_DRBG.

PKCS#11

Це – один із стандартів сімейства Public-Key Cryptography Standards (PKCS), опублікованих RSA Laboratories. Він визначає платформонезалежний програмний інтерфейс доступу до криптографічних пристроїв (засобів криптографічного захисту інформації або смарт-карток). Іноді іменується як Cryptoki. Простіше кажучи, PKCS#11 визначає платформонезалежний набір функцій, структури, константи тощо. Для роботи з криптографічними пристроями. Ці функції можуть бути реалізовані всередині різних бібліотек, наприклад opensc-pkcs11. Бібліотеки можуть відрізнятися як реалізацією, а й самим набором функцій, типів і констант. Це можливо, оскільки стандарт PKCS#11 описує різні способи розширення, що дозволяє додавати свої функції, наприклад, для роботи з підписом CMS, флеш-пам’яттю і т.п. Функції PKCS#11 всередині – це обгортки для роботи з токенами та смарт-картами через APDU команди (APDU – «мова асемблера» для пристроїв).

APDU-формат визначає спосіб спілкування з різними пристроями за допомогою байтових послідовностей. Насправді це відбувається так:

1. Токену або смарт-карті надсилається байтова послідовність;
2. Операційна система пристрою обробляє цю послідовність у команду і надсилає у відповідь код повернення і, за необхідності, додаткову інформацію.

Такі команди можуть містити в собі все що завгодно, починаючи з прохання надіслати інформацію про пристрій або записати якісь дані на нього і закінчуючи прохання зашифрувати вказане повідомлення. Ці команди

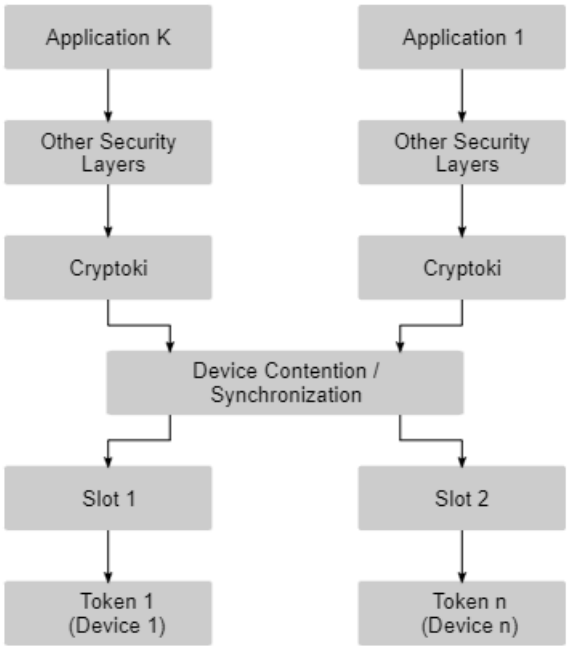
можуть бути розбиті на кілька, що дозволяє оптимізувати роботу з токеном та зробити її конвеєрною. Наприклад, при шифруванні повідомлення пристрою може передаватися невелика частина повідомлення, а він у відповідь надсилатиме результат шифрування тільки цієї частини. Ця особливість може бути використана під час роботи з даними великих розмірів.

APDU дає повну можливість для спілкування з токеном і смарт-картою, а PKCS#11-обгортка забезпечує бібліотеку для розробки, яка є текстовим еквівалентом шістнадцятковим байтовии послідовностям. Також PKCS#11 під час виконання однієї функції посилає кілька APDU команд, що значно спрощує роботу у порівнянні із програмуванням через APDU-команди.

У PKCS#11-бібліотеці є як стандартні функції, ті що явно описані у стандарті, так і певні розширення, додані розробниками бібліотеки і не описані у стандарті. Опис стандартних функцій можна знайти у документації PKCS#11. Про те, як працювати з розширеннями, треба дивитися у розробників конкретних бібліотек. Або ж функції можна розділити за призначенням, одні з яких – функції роботи зі слотами.

Слоти – це віртуальні пристрої бібліотеки для підключення токенів та смарт-карток. PKCS#11 надає функції для отримання списку слотів та очікування зміни стану слотів. Також за допомогою спеціальних функцій можна отримувати інформацію про стан слота, наприклад наявність токена в ньому, інформація про підключений токена і т.п. Одна з найважливіших операцій, яку найчастіше використовують, – отримання списку активних слотів.

Модель для PKCS#11 можна побачити на ілюстрації нижче, демонструючи, як програма передає свої запити маркеру через інтерфейс PKCS#11. Слот – інтерфейс фізичного пристрою. Наприклад, пристрій для зчитування смарт-карт представлятиме слот, а смарт-карта — маркер. Також можливо, що кілька слотів можуть використовувати один і той же маркер.



У PKCS#11 маркер розглядається як пристрій, який зберігає об’єкти та може виконувати криптографічні функції. Об’єкти зазвичай визначаються в одному з чотирьох класів:

- Об'єкти даних, які визначаються програмою
- Об'єкти сертифікатів, які є цифровими сертифікатами, такими як X.509
- Ключові об'єкти, які можуть бути відкритими, закритими або секретними криптографічними ключами
- Визначені постачальником об'єкти

Об’єкти в PKCS#11 далі визначаються як об’єкт маркера або об’єкт сеансу. Об’єкти маркерів видимі для будь-якої програми, яка має достатній дозвіл доступу та підключена до цього маркера. Важливим атрибутом об’єкта маркера є те, що він залишається на маркері, доки не буде виконано певну дію для його видалення.

З’єднання між маркером і програмою називається сеансом. Об’єкти сеансу є тимчасовими та залишаються існувати лише під час відкритого сеансу. Об’єкти сеансу видимі лише програмі, яка їх створила.

Доступ до об’єктів у PKCS#11 визначається типом об’єкта. Загальнодоступні об’єкти є видимими для будь-якого користувача чи програми, тоді як приватні об’єкти вимагають, щоб користувач мав увійти в цей маркер, щоб їх переглядати. PKCS#11 розпізнає два типи користувачів, а саме офіцера безпеки (security officer, SO) або звичайного користувача. Єдина

роль співробітника служби безпеки полягає в тому, щоб ініціалізувати маркер і встановити PIN-код доступу звичайного користувача.

Crypto API та PKCS#11 - це два різні інтерфейси для взаємодії з криптографічними функціями в інформаційних системах. Обираючи який із них обирати необхідно врахувати декілька відмінностей. А саме:

- Сфера застосування.
  Crypto API орієнтований на операційні системи та надає інтерфейс для роботи з криптографією на рівні ОС.
  PKCS#11 орієнтований на роботу з апаратними криптографічними токенами та дозволяє відокремлювати апаратну та програмну частину.
- Функціональність:
  Crypto API надає ширший спектр криптографічних функцій на рівні операційної системи.
  PKCS#11 спрямований на роботу з апаратними токенами, забезпечуючи стандартизований інтерфейс для взаємодії з ними.
- Інтеграція:
  Crypto API інтегрується з функціоналом операційної системи.
  PKCS#11 служить інтерфейсом для взаємодії з апаратними токенами, що дозволяє їм працювати з різними системами.

Висновки

Програмна реалізація криптосистем є дуже громіздким, важким та відповідальним процесом, у якому не варто нехтувати жодним процесом, адже кожен є критично важливим для стійкості та безпеки. Використання смарт-карт або токенів допоможуть забезпечити контроль доступу до ключової інформації, із задачами якого ми ознайомились у даній роботі. Успішна реалізація криптосистеми вимагає урахування різноманітних технічних деталей, які сприяють її правильному та ефективному функціонуванню, тож було розглянуто та досліджено інтерфейси Crypto API, PKCS#11, за допомогою яких можна реалізувати криптографічні системи із використанням смарт-карт та токенів.