МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

ФАКУЛЬТЕТ МЕХАТРОНІКИ ТА КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Алгоритмічне та програмне забезпечення інформаційної системи з елементами захисту від зовнішніх втручань»

Рівень вищої освіти другий (магістерський)
Спеціальність 122 Комп'ютерні науки
Освітня програма Комп'ютерні науки

Виконав: студент групи МгІТ-22

Черкасов Валентин Валентинович

Науковий керівник к.т.н., доц. Колиско М.І

Рецензент

Київ 2023

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

Факультет мехатроніки та комп'ютерних технологій
Кафедра комп'ютерних наук
Рівень вищої освіти другий (магістерський)
Спеціальність 122 Комп'ютерні науки
Освітня програма Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
_____Володимир ЩЕРБАНЬ.
"____" _____2023 року

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Черкасову Валентину Валентиновичу

1. Тема кваліфікаційної роботи Алгоритмічне та програмне забезпечення інформаційної системи з елементами захисту від зовнішніх втручань, науковий керівник роботи Колиско Мар'яна Ігорівна, доц.,к.т.н. затверджені наказом КНУТД від "_12_"вересня_2023 року №_210-уч_

2. Вихідні дані до роботи: Розробки кафедри комп'ютерних наук; рекомендована література, додатки.

3. Зміст дипломної роботи: Вступ; РОЗДІЛ 1 Постановка задачі; РОЗДІЛ 2 Проектування; РОЗДІЛ 3 Програмна реалізація; Висновки; Список літератури; ДОДАТОК А Окремі фрагменти програмного коду; ДОДАТОК Б Презентація.

4. Дата видачі завдання 1 вересня 2023

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапу кваліфікаційної роботи	Орієнтовний термін виконання	Примітка про виконання
1	Вступ	*15.09.2023*	
2	Розділ 1. Постановка задачі	*20.09.2023*	
3	Розділ 2 Проектування	*30.09.2023*	
4	Розділ 3. Програмна реалізація	*10.10.2023*	
5	Висновки	*25.10.2023*	
6	Оформлення (чистовий варіант)	*1.11.2023*	
7	Подача кваліфікаційної роботи науковому керівнику для відгуку (за 14 днів до захисту)	*4.11.2023*	
8	Подача кваліфікаційної роботи для рецензування (за 12 днів до захисту)	*6.11.2023*	
9	Перевірка кваліфікаційної роботи на наявність ознак плагіату (за 10 днів до захисту)	*8.11.2023*	
10	Подання кваліфікаційної роботи на затвердження завідувачу кафедри (з 7 днів до захисту)	*10.11.2023*	

З завданням ознайомлений:

Студент _____ Валентин ЧЕРКАСОВ

Науковий керівник _____ Мар’яна КОЛИСКО

АНОТАЦІЯ

Черкасов В.В. Алгоритмічне та програмне забезпечення інформаційної системи з елементами захисту від зовнішнього втручання

Магістерська робота за спеціальністю 122 Комп'ютерні науки. - Київський національний університет технологій та дизайну, Київ, 2023

Сьогодні інформаційна безпека відіграє важливу роль у будь-якій компанії. При зберіганні, збереженні та наданні доступу до будь-якої інформації її власник або уповноважена особа встановлює найбільш очевидний набір правил роботи з нею. Тому однією з найактуальніших тем у сфері інформаційної безпеки є захист інформаційних систем від несанкціонованого доступу до інформації, що цікавить зловмисників.

Метою даного дослідження є розробка системи криптографічного захисту інформації від зловмисників. Об'єктом автоматизації була обрана бухгалтерія підприємства. Для досягнення цього нам необхідно вирішити такі завдання: описати сферу діяльності підприємства; описати математичну модель захисту інформації; розробити криптоалгоритм.

Криптографічний захист інформації є надійним і недорогим засобом. Будь-який обсяг інформації від кількох байт до гігабайта, зашифрований за допомогою більш-менш захищеної криптосистеми, неможливо прочитати без знання ключа. І зовсім неважливо, чи зберігається він на жорсткому диску, на дискеті чи на компакт-диску, незалежно від того, яка операційна система працює.

ABSTRACT

Cherkasov V.V. Algorithmic and software support of the information system with elements of protection against external interference

Master's degree work on the specialty 122 Computer science. - Kyiv National University of Technology and Design, Kyiv, 2023

Today, information security plays a major role in any company. When storing, maintaining and providing access to any information, its owner, or an authorized person, imposes the most obvious set of rules for working with it. Therefore, one of the most pressing topics in the field of information security is the protection of information systems from unauthorized access to information of interest to intruders.

The purpose of this study is to develop a system for cryptographic protection of information from intruders. The accounting department of the company was chosen as the object of automation. To achieve this, we need to solve the following tasks: describe the scope of the enterprise; describe the mathematical model of information data protection; develop a crypto algorithm.

Cryptographic protection of information is a reliable and inexpensive means. Any amount of information from a few bytes to a gigabyte, being encrypted using a more or less secure cryptosystem, is unreadable without knowing the key. And it doesn’t matter at all whether it is stored on a hard disk, on a floppy disk or on a CD, no matter what operating system is running.

ЗМІСТ

ВСТУП

Інформація, як і будь-який продукт, має виробників, споживачів і власників, а тому наділена певними споживчими якостями. З погляду споживача, якість наявної інформації дає змогу отримувати економічний або інші ефекти. З погляду власника, збереження в таємниці комерційно важливої інформації дає змогу успішно конкурувати на ринку виробництва і збуту товарів і послуг. Це відповідно вимагає певних дій, спрямованих на захист конфіденційної інформації. Задовольнити сучасні вимоги щодо безпеки підприємства і захисту його конфіденційної інформації можна формуванням системи інформаційної безпеки.(1)

В епоху інформаційних технологій стає можливим майже миттєво отримувати інформацію, яка з'являється кожну секунду в різних куточках світу, зокрема щодо укладених торговельних угод, коливання валют і цінних паперів та багато іншого. У зв'язку із цим, все більшої актуальності набуває питання захисту інформаційних ресурсів бізнес-структур засобами сучасних інформаційних технологій і систем. При цьому основною метою створення системи захисту інформації стає забезпечення надійного зберігання та ефективного використання інформації в діяльності бізнес-структур.

Обмеження доступу до інформації – це стан захищеності інформації та інфраструктури, що її підтримує, від випадкових або навмисних дій, які можуть завдати збитку суб'єктам інформаційних відносин, зокрема, власникам і користувачам інформації та інфраструктури [2]. В Українській законодавчій базі термін «інформаційна безпека» наведено у Концепції національної програми інформатизації, затвердженій Законом України від 4 лютого 1998 року № 75/98, де «інформаційна безпека» – невід'ємна частина політичної, економічної, оборонної та інших складових національної безпеки. Об'єктами інформаційної безпеки є інформаційні ресурси, телекомунікації, канали інформаційного обміну, функціонування телекомунікаційних мереж і систем та інші елементи інформаційної інфраструктури країни» [2]. З інформаційної точки зору суб'єкт бізнесу являє собою комплекс компонентів, пов'язаних між

собою єдиною метою, структурними відносинами, технологіями інформаційного обміну. Зазначені компоненти в процесі функціонування суб’єкта можуть змінюватися, на них можуть впливати різного роду внутрішні і зовнішні чинники, які складно прогнозувати і оцінювати. Всі компоненти можна сформувати в чотири групи: – персонал; – технічні засоби інформатизації; – програмне забезпечення; – документи і вважати як об’єкти захисту інформації.

Використання всесвітньої мережі та нових технологій супроводжується такими явищами, як низький рівень культури безпеки, збільшення онлайн-користувачів і залежності від цифрової інфраструктури, поширення небажаного контенту, розвиток кібер-шахрайства, витоки інформації, втрата даних, несанкціонований доступ до інформації.. Небезпека, в першу чергу, загрожує інформації, що зберігається в інформаційних системах підприємства. В цю систему входять програмне забезпечення автоматизованої системи, програми для виконання конкретних завдань компанії, програмні оболонки, текстові редактори, пакети програм, бази даних. Інформація може надходити по локальній мережі з пристрою введення, а саме з клавіатури, з зовнішнього середовища, а саме з мережі Інтернет за системою SWIFT, від інших компаній. Будь-яка компанія хоче бути надійно захищена від загроз.

Метою дипломної роботи є аналіз і дослідження систем обмеження доступу до інформації, включаючи технічні та програмні засоби захисту інформації та розробка пропозицій щодо її покращення задля підвищення рівня безпеки підприємства. Об’єктом дослідження є процес захисту інформації бухгалтерського відділу підприємства з продажу електронної техніки. Предмет дослідження: методи, методики, моделі і інструменти захисту інформаційних систем від зовнішніх втручань. В якості об'єкту автоматизації в роботі обрано бухгалтерський відділ компанії. Для досягнення поставленої мети, нам необхідно вирішити наступні завдання:

- описати сферу діяльності підприємства;

- описати математичну модель інформаційного захисту даних;

- програмно реалізувати власний криптоалгоритм;

- інтегрувати алгоритм в програмний комплекс;

Практичне значення одержаних результатів полягає в здійсненні удосконалення системи на основі впровадження власного доповнення до вже існуючої системи обмеження доступу до інформації на підприємстві, що покращить захищеність інформації на даному об’єкті.

Статтю з оглядом існуючих проблем та їх причин було опубліковано у 2023р.

РОЗДІЛ 1 ПОСТАНОВКА ЗАДАЧІ

1.1 Характеристика підприємства

Завданням підприємства на якому передбачається використати систему - забезпечення задоволеності споживачів якісною продукцією в умовах ринкової конкуренції. Підприємство займається продажами і доставкою побутової техніки, оригінальних запасних частин і аксесуарів, а також забезпеченням гарантії і техобслуговування Уважне вивчення місцевої ринкової кон'юнктури і економічних умов дозволяє фірмі вести таку маркетингову активність, яка може забезпечити відповідність її продукту і послуг запитам споживачів

Служба експлуатації виконує наступні функції:

а) організовує завезення побутової техніки від компаній виробників;

б) забезпечує розтаможення і оформлення необхідних документів;

в) забезпечує продаж цієї продукції;

г) здійснює контроль якості обслуговування;

д) здійснює заміну і продаж різних деталей.

Плановий відділ виконує наступні функції:

а) займається складанням звідної виробничої програми

б) займається складанням штатного розпису;

в) нормування матеріальних і трудових витрат;

г) укладає угоди на завезення товарів із зарубіжних країн.

Бухгалтерський відділ підприємства займається складанням звідної бухгалтерської звітності. Документообіг підприємства представлений на Рисунку 1.1.

Тут представлені функції виробничої системи, подана модель в структурованому виді, показані потоки інформації які зв'язують різні підсистеми. Так само метою цієї функціональної схеми є уточнення функцій підсистем, показ взаємозв'язків і напрямів руху інформації підсистем, в який входить і список вхідних і вихідних файлів. Нас цікавить бухгалтерський відділ підприємства, який займається складанням звідної бухгалтерської звітності. При проведенні розрахунків потрібні і використовуються прикладні комерційні

програмні продукти, робота в мережі, отже активне застосування інформаційних технологій. Усе це з одного боку дозволяє підприємству добре орієнтуватися в ході і перспективах техніко- економічного розвитку виробництва, але і несе за собою риски.

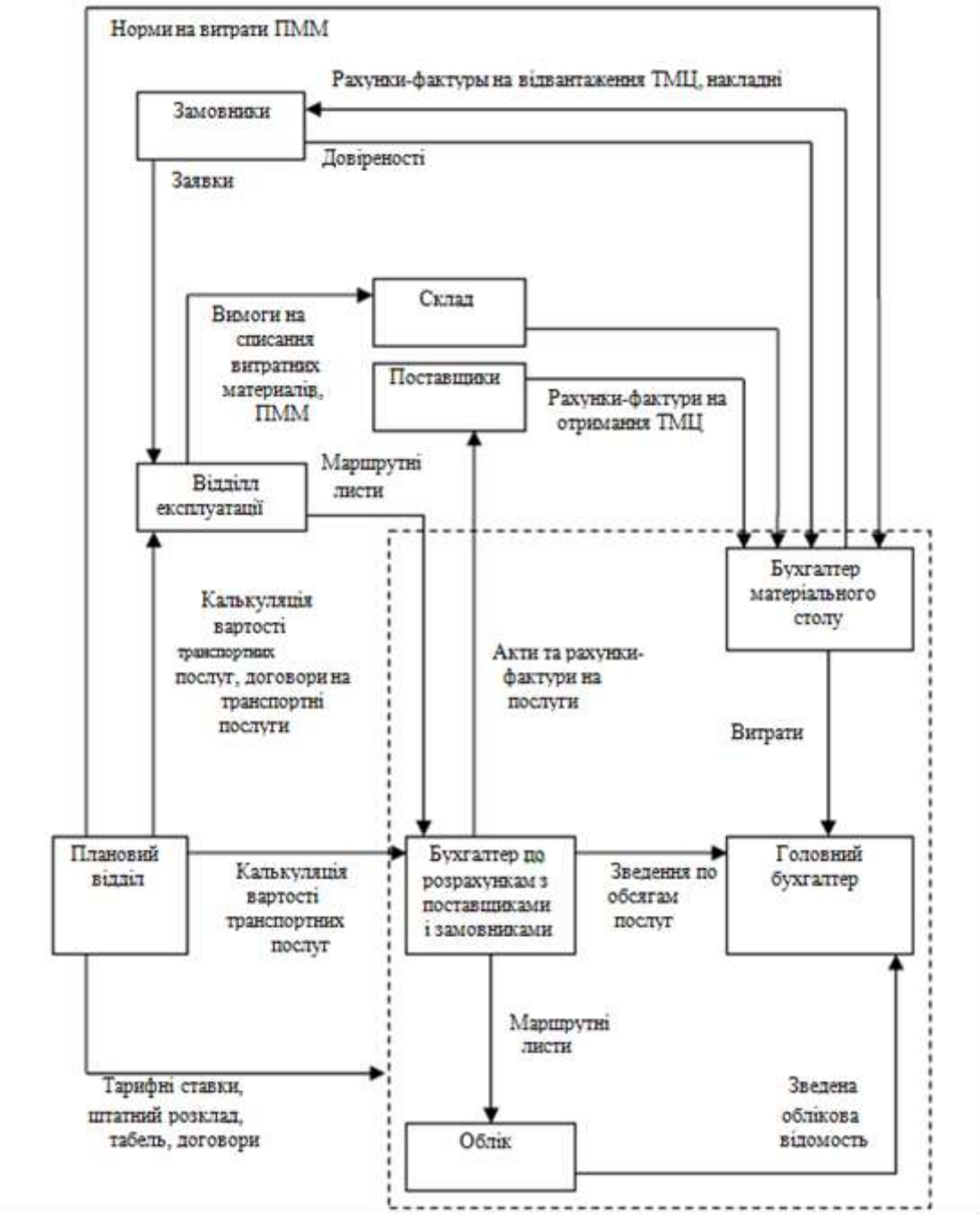

Рисунок 1.1.Документообіг підприємства

1.2 Обгрунтування необхідності захисту інформації

На сьогодні захист комерційної інформації грає не останню роль у будь-якій компанії. Оскільки інформаційні атаки можуть привести до великих втрат. Під інформаційною атакою розуміється умисне порушення правил співпраці на

ринку. У засобах масової інформації все частіше з'являється інформації про комп'ютерні зломи, про великі атаки на компанії. При зберіганні, підтримці і наданні доступу до будь-якої інформації її власник, або уповноважена особа, накладає найочевидніший набір правил щодо роботи з нею. А умисний доступ чи нанесення шкоди класифікується, як інформаційна атака. В усіх сферах людської діяльності з впровадженням комп'ютерів в тисячі разів зросли обсяги зберігання інформації в оцифрованому вигляді. На тепер скопіювати за півхвилини і понести дискету з файлом, що містить, наприклад, контакти постачальників, набагато простіше, ніж копіювати або переписувати гору паперів. А з появою комп'ютерних мереж навіть відсутність фізичного доступу до комп'ютера перестала бути гарантією збереження інформаціїІ тож на сьогодні однією з найактуальніших тем у сфері інформаційної безпеки є захист інформаційних систем від несанкціонованого доступу до інформації, що представляє інтерес для зловмисників або конкурентів.

Можливі наслідки атак на інформацію:

а) економічні втрати - розкриття комерційної інформації може привести до серйозних прямих збитків в компанії;

б) звістка про витік великого об'єму інформації зазвичай серйозно впливає на репутацію фірми, призводячи побічно до втрат в об'ємах торгових операцій;

в) фірми-конкуренти можуть скористатися крадіжкою інформації, якщо інформація залишилася непоміченою, для того, щоб повністю розорити фірму, нав'язуючи їй фіктивні або свідомо збиткові угоди;

г) підміна інформації, як на етапі передачі, так і на етапі зберігання в компанії може привести до значних збитків.

Інформація з точки зору інформаційної безпеки має наступні категорії:

а) конфіденційність - гарантія того, що конкретна інформація доступна тільки тому колу осіб, для кого вона призначена; порушення цієї категорії називається розкраданням або розкриттям інформації;

б) цілісність - гарантія того, що актуальна на тепер інформація існує в своєму початковому виді, тобто при зберіганні або передачі не було зроблено

несанкціонованих змін; порушення цієї категорії називається фальсифікацією повідомлення;

в) автентичність - гарантія того, що джерелом інформації є саме та особа, яка заявлена як її автор; порушення цієї категорії також називається фальсифікацією, але вже автора повідомлення.

Стадії створення системи захисту інформації :

а) передпроектна стадія, що включає передпроектне обстеження об'єкту інформатизації, розробку аналітичного обгрунтування необхідності створення СЗІ та і технічного (частки технічного) завдання на її створення;

б) стадія проектування (розробки проектів), що включає розробку СЗІ у складі об'єкту інформатизації;

в) стадія введення в дію СЗІ, що включає дослідну експлуатацію і приймально-здавальні випробування засобів захисту інформації, а також атестацію об'єкту інформатизації на відповідність вимогам безпеки інформації.

1.3 Моделі и методи систем захисту інформації

На перший план при вирішенні задач забезпечення інформаційної і кібербезпеки, а також приватності виходить створення системи управління інформаційною безпекою (СУІБ), яка охоплює всю інфраструктуру компанії. СУІБ дозволяє:

- централізовано і оперативно впливати на всю інформаційну інфраструктуру;
- проводити регулярний аудит і всеохоплюючий моніторинг, що дає об'єктивну інформацію про стан кібер- та інформаційної безпеки для прийняття оперативних рішень;
- накопичувати статичні дані про роботу інформаційної інфраструктури для прогнозування її розвитку;
- оцінювати ризики інформаційної безпеки.

Активна діяльність міжнародних організацій зі стандартизації підтверджує важливість питань забезпечення кібер- і інформаційної безпеки в системах інформаційних технологій та постійного удосконалення моделей, методів та механізмів безпеки інформаційних технологій. Основний підхід до

забезпечення інформаційної безпеки в ІС – стратегія захисту на основі ризику (Risk-Based Protection Strategy). Успішні програми управління ризиками передбачають побудову системи забезпечення інформаційної безпеки, яка інтегрована в організаційну і технічну інфраструктуру.

Виходячи з цих позицій NIST США розробив та впроваджує як методологічну основу забезпечення інформаційної та кібербезпеки концепцію Risk Management Framework (RMF). Концепція RMF впроваджує структурований гнучкий підхід до управління ризиками, що пов'язаний із впровадженням інформаційних систем у бізнес-процеси організації. Концепція RMF об'єднує серію документів NIST SP 800-XX. У 2018 р. у зв'язку з актуалізацією питання захисту персональних даних, прийняттям основних положень забезпечення кібербезпеки, впровадженням моделі довірчих систем NIST розпочав перегляд багатьох документів серії NIST SP 800-XX та видав нову версію NIST SP 800-37 – Risk Management Framework in Information Systems and Organizations. Основні цілі впровадження RMF: - забезпечення повторюваного процесу забезпечення інформаційної безпеки у відповідності до чинних ризиків; - впровадження цілісного загальносистемного підходу до управління ризиками безпеки та приватності; - впровадження єдиної методики класифікації (категоризації) інформаційних систем та загальних заходів безпеки; - забезпечення управління ризиками в режимі реального часу через впровадження надійних безперервних процесів моніторингу безпеки; - впровадження засобів автоматизації управління інформацією для прийняття ефективних рішень на основі ризиків для ІС; - забезпечення інтеграції вимог безпеки та заходів захисту в архітектуру підприємства. При втіленні системи керування інформаційною безпекою організація повинна визначити сукупність методів захисту інформації і організацію проведення аудиту системи безпеки, визначити ризики та можливості щоб: а) гарантувати, що система управління інформаційною безпекою може досягти запланованого результату(-ів);
б) запобігти або зменшити небажані ефекти; в) досягти постійного вдосконалення. Організація має визначити та застосовувати процес оцінювання

ризиків інформаційної безпеки, який: а) встановлює та підтримує критерії ризиків що містять критерії прийняття ризиків і критерії оцінки ризиків інформаційної безпеки; б) гарантує, що оцінки ризиків призводять до послідовних та порівняльних результатів; в) ідентифікує ризики інформаційної безпеки та ідентифікує власників ризиків; г) оцінює потенційні наслідки від ризиків; д) визначає пріоритети проаналізованих ризиків для їх обробки та усунення.

Методологія оцінки ризиків може бути якісною або кількісною, або їх деякою комбінацією. Якісна оцінка часто використовується для отримання загального рівня ризику і виокремлення головних ризиків. При якісному підході не використовуються кількісні або грошові вираження для об'єкта оцінки. При кількісному підході всім елементам оцінки ризиків привласнюють конкретні і реальні кількісні значення. Об'єктом оцінки може бути цінність активу в грошовому вираженні, ймовірність реалізації загрози, збитки від реалізації загрози, вартість захисних заходів та ін.

Кількісний підхід до оцінки ризиків може включати такі етапи:

1. Визначити цінність інформаційних активів в грошовому вираженні.

2. Оцінити в кількісному вираженні потенційний збиток від реалізації кожної загрози щодо кожного інформаційного активу.

3. Визначити ймовірність реалізації кожної із загроз ІБ.

4. Визначити загальний потенційний збиток від кожної загрози щодо кожного активу за контрольний період.

5. Провести аналіз отриманих даних по збитку для кожної загрози.

Ідентифікація критеріїв ризику визначає прийняття рішень щодо характеру можливих наслідків та способу їх вимірювання. Наявні на сьогодні методи оцінки ризиків в переважній кількості засновані на статистичних підходах. У більшості країн подібна статистика не ведеться.

Загальне оцінювання ризику дає змогу впроваджувати необхідні міри на рівні підрозділів, проєктів, конкретних ризиків або на рівні організації в цілому.

Ризики інформаційної безпеки характеризуються двома параметрами: потенційним збитком для організації та ймовірністю їх реалізації. Використання для аналізу ризиків сукупності цих двох характеристик дозволяє порівнювати ризики з різними рівнями шкоди і ймовірності, приводячи їх до вигляду зрозумілому для осіб, котрі приймають рішення щодо мінімізації ризиків в організації.

При створенні системи управління ІР постає питання вибору заходів захисту, що забезпеча зниження виявлених в процесі аналізу ризиків без надмірних витрат на впровадження і підтримку. Аналіз ризиків дозволяє визначити необхідну і достатню сукупність заходів, спрямованих на зниження ризиків безпеки, та розробити архітектуру СУІБ організації, максимально ефективну саме для специфіки її діяльності.

Розглянемо загальну характеристику та проведемо порівняльний аналіз широко застосовуваних методик і методів управління та оцінки ризиками інформаційної і кібербезпеки.

1. Фреймворк "NIST Risk Management Framework" – на базі американських урядових стандартів NIST (National Institute of Standards and Technology), включає в себе набір взаємозв'язаних стандартів: – Стандарт NIST SP 800-30 "Guide for Conducting Risk Assessments" ("Керівництво з проведення оцінки ризиків") сфокусований на інформаційній безпеці (ІБ) і операційних ризиках; Стандарт NIST SP 800-39 "Managing Information Security Risk" пропонує трирівневий підхід до управління ризиками: організація, бізнес-процеси, інформаційні системи. Даний стандарт описує методологію процесу управління ризиками: визначення, оцінка ризиків інформаційної безпеки, реагування та моніторинг ризиків; Стандарт NIST SP 800-37 "Risk Management Framework for Information Systems and Organizations" пропонує для забезпечення безпеки і конфіденційності використовувати підхід управління життєвим циклом систем; Стандарт NIST SP 800-137 "Information Security Continuous Monitoring" описує підхід до процесу моніторингу інформаційних

систем і IT-середовищ з метою контролю застосованих заходів обробки ризиків ІБ і необхідність їх перегляду.

2. Стандарти управління ризиками інформаційної безпеки Міжнародної організації зі стандартизації ISO (International Organization for Standardization): стандарт ISO/IEC 27005 2018 - "Information technology Security techniques Information security risk management" («Інформаційна технологія. Методи і засоби забезпечення безпеки. Менеджмент ризиків інформаційної безпеки») що входить в серію стандартів ISO 27000 та є логічно взаємопов'язаним з іншими стандартами по ІБ з цієї серії.

3. Методологія FRAP (Facilitated Risk Analysis Process) є відносно спрощеним способом оцінки основних ризиків інформаційної безпеки, з фокусом тільки на найкритичніших активах. Якісний аналіз проводиться за допомогою експертної оцінки.

4. Методологія OCTAVE (Operationally Critical Threat, Asset, and Vulnerability Evaluation) сфокусована на самостійній роботі членів бізнес-підрозділів. Вона використовується для масштабної оцінки всіх інформаційних систем і всіх бізнес-процесах компанії.

5. Стандарт AS / NZS 4360 є австралійським і новозеландським стандартом з фокусом не тільки на IT-системах, але і на бізнес-здоров'я компанії, тобто пропонує більш глобальний підхід до управління ризиками інформаційної безпеки (наприклад, в банку). Відзначимо, що даний стандарт зараз замінений на стандарт AS / NZS ISO 31000-2009.

6. Методологія FMEA (Failure Modes and Effect Analysis) пропонує проведення оцінки системи з точки зору її слабких місць для пошуку ненадійних елементів.

7. Методологія CRAMM (Central Computing and Telecommunications Agency Risk Analysis and Management Method) пропонує використання автоматизованих засобів для управління ризиками інформаційної безпеки.

8. Методологія FAIR (Factor Analysis of Information Risk) – Фреймворк для проведення кількісного аналізу ризиків, що пропонує модель побудови

системи управління ризиками на основі економічно ефективного підходу, прийняття поінформованих рішень, порівняння заходів управління ризиками, фінансових показників і точних ризик-моделей.

9. Концепція COSO ERM (Enterprise Risk Management) описує шляхи інтеграції ризикменеджменту зі стратегією і фінансової ефективністю діяльності компанії і акцентує увагу на важливість їх взаємозв'язки.

Проведемо порівняльний аналіз окремих перерахованих методів.

Методика NIST 800-30 є однією з найпопулярніших та широковживаних методик управління ризиками є методика оцінки ризиків. Ця методика передбачає попереднє оцінювання двох параметрів: потенційного збитку та ймовірності реалізації загрози. Використання такої методики передбачає такі етапи: – опис характеристик системи; – ідентифікація загроз; – ідентифікація вразливостей; – аналіз наявних засобів/заходів захисту; – визначення значення ймовірності; – аналіз впливу; – визначення значення ризику; – вибір засобів/заходів захисту; – документування отриманих результатів.

Методика CRAMM Методика CRAMM (CCTA Risk Analysis and Management Method), розроблена Службою безпеки Великобританії, базується на стандартах управління інформаційної безпеки серії BS7799 (в даний час перероблені в ISO 27000) і описує підхід до якісної оцінки ризиків. Методика є універсальною і придатна як для великих, так і для малих організацій, як державного, так і комерційного сектору. Версії програмного забезпечення CRAMM, орієнтовані на різні типи організацій, відрізняються своїми базами знань (profiles). Економічно обґрунтована стратегія управління ризиками ІБ дає змогу, в кінцевому підсумку, заощаджувати кошти, уникаючи невиправданих витрат.

Метод OCTAVE Метод OCTAVE (Operationally Critical Threat, Asset and Vulnerability Evaluation) розроблений в Університеті Карнегі-Мелон (США) і передбачає оцінювання критичності загроз, активів і вразливостей. Зміст методики полягає в тому, що для оцінки ризиків використовується послідовність відповідно організованих внутрішніх семінарів (workshops).

Величина ризику визначається як середнє значення річних втрат компанії в результаті реалізації загроз інформаційної безпеки (ІБ).

Методологія COBIT Методологія COBIT for Risk розроблена асоціацією ISACA (Information Systems Audit and Control Association) в 2013 р. і базується на кращих практиках управління ризиками (COSO ERM, ISO 31000, ISO\IEC 27xxx і ін.). Методологія містить більше 100 ризикових сценаріїв, що охоплюють такі категорії впливу: – створення та обслуговування портфелів IT-проєктів; – управління життєвим циклом програми/проєкту; – інвестиції в IT; – експертиза і навички персоналу IT; – операції з персоналом; – інформація; – архітектура; – IT-інфраструктура; – програмне забезпечення; – неефективне використання IT; і т.п.

Методи оцінки ризиків безпеки інформації MAGERIT та MEHARI. Інструментарій MEHARI складається з чотирьох модулів комплексне використання яких дозволяє адаптувати цей метод до використання у будь-якій організації. Розробники пропонують такий порядок проведення ОР: оцінка загрози; визначення ресурсів, які від неї постраждають; визначення заходів захисту (ЗЗ), що дозволяють забезпечити попередження; захист або відновлення бізнес-процесів організації після реалізації загрози. Для кожного етапу надані прикладні засоби. MEHARI надають: практичні рекомендації, таблиці, розрахункові формули та шкали оцінок. Супутня документація містить: інструкції та поради щодо ефективного використання бази знань (подана у форматі Excel), а також теоретичні відомості щодо управління ризиками ІБ.

NIST США розробив як методологічну основу забезпечення інформаційної та кібербезпеки концепцію Risk Management Framework (RMF). Концепція RMF впроваджує структурований гнучкий підхід до управління ризиками, що пов’язаний із впровадженням інформаційних систем у бізнес-процеси організації. Система управління інформаційною безпекою (СУІБ) забезпечує збереження конфіденційності, цілісності й доступності інформації за допомогою запровадження процесу управління ризиками та надає впевненості

зацікавленим сторонам. Аналіз ризиків інформаційної безпеки дозволяє визначити необхідну і достатню сукупність засобів захисту інформації і розробити архітектуру СУІБ організації, максимально ефективну для її специфіки діяльності. На основі наведеного аналізу можна стверджувати, що оптимальним варіантом для вибору методу управління ризиками ІБ в контексті забезпечення неперервності функціонування СУІБ є, зокрема, адаптація та удосконалення відомих методів шляхом їх логічного поєднання з урахуванням переваг та мінімізації недоліків цих методів.

1.4. Модель захисту інформації підприємства

Захист інформації - це комплекс заходів, спрямованих на забезпечення інформаційної безпеки. Таким чином, правильний з методологічної точки зору підхід до проблем інформаційної безпеки розпочинається з виявлення суб'єктів інформаційних стосунків і інтересів цих суб'єктів, пов'язаних з використанням інформаційних систем. Згідно з визначенням, інформаційна безпека залежить не лише від комп'ютерів, але і від підтримувальної інфраструктури, до якої можна віднести системи електро-, водо- і теплопостачання, кондиціонери, засоби комунікацій і, звичайно, обслуговуючий персонал. Ця інфраструктура має самостійну цінність, але нас цікавитиме лише те, як вона впливає на виконання інформаційною системою наказаних їй функцій. Нас зокрема цікавить захист локальної мережі підприємства.

Захищати мережу необхідно від таких загроз, як:

- копіювання носіїв інформації з подоланням заходів захисту;
- маскування під зареєстрованого користувача;
- містифікація (маскування під запити системи);
- використання програмних пасток;
- використання недоліків мов програмування і ОС;
- включення у бібліотеки програм блоків типу «троянського коня»;
- зловмисне виведення із ладу механізмів захисту;
- впровадження і використання комп'ютерних вірусів.

У системі захисту мережі кожен її вузол повинен мати індивідуальний

захист залежно від виконуваних функцій і можливостей мережі.

Структурно-функціональний склад мережі значною мірою впливає на засоби захисту програмного і інформаційного забезпечення. Програмні засоби захисту при цьому можуть бути як комплексними, призначеними для усієї мережі в цілому (наприклад, такі, як штатні засоби захисту мережевих ОС), так і одиничними, орієнтованими на забезпечення безпеки окремих компонентів. Важливою вимогою, що пред'являється до засобів захисту, може служити їх функціональна повнота.

Можна виділити наступні основні функції захисту програмного і інформаційного забезпечення, характерні для локальної мережі як складного об'єднання засобів електронно-обчислювальної техніки :

- адміністрування мережі;
- забезпечення ідентифікації і аутентифікації користувачів;
- розмежування доступу до ресурсів мережі;
- очищення «сміття»;
- забезпечення захисту даних, що передаються між елементами мережі;
- реєстрація дій, що вимагають доступу до ресурсів, що захищаються, виявлення фактів порушень;
- контроль цілісності (коректності процесу зміни станів) найбільш важливих компонентів програмного і інформаційного забезпечення обчислювальної мережі.

Метою аутентифікації є ідентифікація користувача перед наданням йому доступу до інформації в мережі. Для роботи з файлами, директоріями і утилітами, що знаходяться в мережі, користувач повинен отримати, дозвіл від ОС. Якщо необхідно, паролі можуть бути зашифровані перед передачею по каналу і перед записом на диск. Можуть бути встановлені мінімальна довжина пароля, вимоги періодичної зміни пароля і унікальності пароля. Періодична зміна паролів користувачами зменшує ризик їх розкриття, а велика довжина паролів утрудняє розгадування.

У загальному випадку необхідною умовою забезпечення необхідного рівня захищеності від НСД процедури аутентифікації видалених користувачів є застосування засобів шифрування на основі індивідуальних і відкритих ключів.

Коректне застосування засобів шифрування в комплексі з іншими засобами дозволяє забезпечити необхідний рівень захисту даних. Шляхом розмежування доступу встановлюється факт, дозволено або ні користувачеві мати доступ до деякого ресурсу мережі. Розмежування доступу до ресурсів обчислювальної мережі реалізується, як правило, на декількох рівнях.

Для забезпечення захисту даних, що передаються між елементами мережі, необхідно здійснювати заходи:

- запобігання розкриттю вмісту передаваних повідомлень;
- запобігання аналізу потоків повідомлень;
- запобігання і виявлення спроб модифікацій потоку;
- запобігання і виявлення переривань передачі повідомлень;
- виявлення ініціації неправдивого з'єднання.

Для вирішення завдання по забезпеченню захисту даних, що передаються між елементами мережі, зберігання критичних даних на довготривалих пристроях, що запам'ятовують, захисту цілісності програмного забезпеченні повідомлень використовуються криптографічні методи. До недавнього часу, незважаючи на наявність програмних засобів захисту, що претендували при відповідному налаштуванні на універсальне рішення завдань безпеки мережі, мережеві операційні системи реально не забезпечували скільки-небудь прийнятного рівня захищеності систем.

За шкалою федеральних стандартів США, прийнятих Національним центром комп'ютерної безпеки, засоби захисту мережевих ОС у кращому разі розташовувалися на нижніх відмітках. Нині з'явилися мережеві операційні системи, що отримали сертифікат рівня С2.

1.5 Постановка завдання

На сьогодні захист інформації грають головну роль у будь-якій компанії. Оскільки інформаційні атаки можуть привести до великих втрат. Під

інформаційною атакою розуміється умисне порушення правил. У засобах масової інформації все частіше з'являється інформації про комп'ютерні зломи, про великі атаки на компанії. При зберіганні, підтримці і наданні доступу до будь-якої інформації його володар, або уповноважена особа, накладає найочевидніший набір правив по роботі з нею. Це умисне нанесення шкоди класифікується, як інформаційна атака. У усіх сферах людської діяльності з впровадженням комп'ютерів в тисячі разів виріс зберігання інформації в електронному вигляді. І на сьогодні однією з найактуальніших тем у сфері інформаційної безпеки є захист інформаційних систем від несанкціонованого доступу до інформації, представляючий інтерес для зловмисників.

Криптографічний захист інформації є доволі невитратним коштом. Будь-який об'єм інформації від декількох байт до гігабайта, будучи зашифрований за допомогою більш менш стійкої криптосистеми, недоступний для прочитання без знання ключа. І вже абсолютно не важливо, зберігається він на жорсткому диску, на дискеті або на компакт-диску, неважливо під управлінням якої операційної системи. Проти самих новітніх технологій і мільйонних витрат тут стоїть математика. І цей бар'єр досі неможливо здолати. У наш час, це можна реалізувати за допомогою такої актуальної програми, як True Crypto.

Висновки по першому розділу

У цьому розділі була розглянуті питання організації захисту даних від зовнішніх втручань, обгрунтування необхідності захисту інформації на підприємствах, моделі й методик систем управління та оцінки ризиками інформаційної безпеки та проведено їх порівняльний аналіз.

Виходячи з проведеного аналізу літературних джерел і ресурсів інтернет, список завдань виглядатиме наступним чином. Для вирішення задачі, як зашифрувати пристрій або дані на пристрої, необхідно виконати наступні дії:

1. вивчити характеристику підприємства;
2. розглянути моделі і методи роботи програм для криптографічного захисту з відкритим кодом;
3. розробити програмну реалізацію алгоритму з його впровадженням в True Crypto.

Вимоги до програми, що розробляється :

1. - зручний графічний призначений для користувача інтерфейс (GUI);
2. - простота в користуванні;
3. - забезпечення шифрування усіх типів файлів;
4. - надійність функціонування;
5. - невисока вартість;
6. - здатність до зміни і доповнення.

РОЗДІЛ 2 ПРОЕКТУВАННЯ АРХІТЕКТУРИ

2.1. Структура моделі для захисту інформації

Сукупність захисних методів і засобів включає:

• програмні методи;

• апаратні засоби;

• захисні перетворення;

• організаційні заходи

Програмні методи захисту - це сукупність алгоритмів і програм, що забезпечують розмежування доступу і виключення несанкціонованого використання інформації. До програмних методів захисту інформації відносяться: - антивірусна програма; - програмний брандмауер і т. п.

Суть апаратного або схемного захисту полягає в тому, що в пристроях і технічних засобах обробки інформації передбачається наявність спеціальних технічних рішень, що забезпечують захист і контроль інформації. До апаратних методів захисту інформації відносяться: - електронний замок; - електромагнітний екран і т. п.

Суть методів захисних перетворень полягає в тому, що інформація, що зберігається в системі і передавана по каналах зв'язку, представляється в деякому коді, що унеможливлює її безпосереднє використання. До методів захисних перетворень відносяться: - методи заміни; - методи перестановки і т. п.

Організаційні заходи по захисту включають сукупність дій з підбору і перевірки персоналу, строга регламентація процесу розробки і функціонування інформаційної системи. До організаційних заходів по захисту інформації відносяться: - організація пропускного режиму у будівлю; - система відеоспостереження і т. п.

Тріада «конфіденційність, доступність, цілісність» тільки одна з існуючих моделей інформаційної безпеки. Служба конфіденційності забезпечує секретність інформації. Правильно конфігурована, ця служба відкриває доступ до інформації тільки аутентифікованим користувачам. Служба конфіденційності повинна

враховувати різні способи представлення інформації - у вигляді роздруків, файлів або пакетів, що передаються по мережах.

Механізми забезпечення конфіденційності :

1. Контроль фізичної безпеки
2. Контроль доступу до файлів на комп'ютері
3. Шифрування файлів

Вимоги до конфіденційності файлів

1. Ідентифікація і аутентифікація
2. Правильне налаштування комп'ютерної системи
3. Правильне управління ключами при використанні шифрування

Для гарантування безпеки інформації відділу бухгалтерії всередині локальної мережі підприємства пропонується застосувати вже готовий програмной продукт з відкритим кодом який забезпечить нам вже готовий зручний користувацький інтерфейс TrueCrypt куди додається власний алгоритм шифрування.

2.2. Структура програм для захисту інформації.

Для розробки пропонується застосувати вже готовий програмний продукт який забезпечить нас «рушієм», що дозволить здійснювати кодування-декодування «на льоту». Для цього обрано програму TrueCrypt – це безкоштовний продукт із відкритим вихідним кодом. Це одна з найкращих програм для шифрування файлів та розділів диска, навіть якщо порівнювати із комерційними продуктами. Вона працює на Widnows, Linux та OS X.

Основні її особливості:

• Створення віртуальних зашифрованих дисків (контейнерів) та робота з ними, як зі справжніми дисками.

• Шифрування розділів, жорстких або USB дисків повністю.

• Шифрування розділу або диска на якому інстальовано Windows (дозавантажувальна автентифікація).

• Шифрування автоматичне, відбувається в реальному часі (on-the-fly), інтуїтивно зрозуміле.

• Паралельність та конвейєрність дозволяє читати та записувати дані так швидко, якби диск не був зашифрований.

• Підтримує апаратне прискорення на сучасних процесорах.

• Переконливо приховує факт шифрування: використовує приховані томи та приховані операційні системи

TrueCrypt – це програма, яка дозволяє шифрувати файли на льоту. Це означає, що тут не потрібно шифрувати кожен файл окремо. Треба просто створити зашифроване сховище для файлів, підключити його та працювати як із звичайним диском операційної системи. Відкривши цей диск, ви бачите всі файли та папки, що знаходяться у зашифрованому сховищі, ніби вони знаходяться у вас на флешці, або переносному вінчестері.

Питання захисту інформації, без можливості шифрування на льоту призвело б до того, що людина, якій потрібно забезпечити захист, відмовилася б від цієї задумки сказавши, що дане рішення абсолютно не практично, тому що доводиться постійно зашифровувати-розшифровувати файли.

Давайте як приклад розглянемо просту ситуацію. Ви –бухгалтер. У вас є роботи, інформація про які повинна залишатися закритою за будь-яких обставин. У процесі роботи вам потрібно працювати з великою кількістю документів. У випадку, якщо у вас не використовується процес шифрування на льоту, для відкриття одного документа, вам потрібно виконати такі дії:

1. Розшифрувати зашифрований контейнер із документами у тимчасову, небезпечну папку
2. Відкрити документи, редагувати, зберегти зміни
3. Зашифрувати документи за новою
4. Видалити незашифровану копію документів

5. Для того, щоб файли були видалені напевно, необхідно запустити програму, яка затирає весь вільний простір, після чого звідти відновити видалені файли буде вже неможливо.

Дійсно складно. У цій процедурі потрібно зробити стільки дій, що після чергової розшифровки, ви просто відмовитеся від всієї цієї витівки з шифруванням і працюватимете по-старому, з документами звваленими у вас на робочому столі без шифрування.

А от коли всі файли всередині контейнера зашифровані і TrueCrypt у процесі вашої роботи із зашифрованими файлами виступає як посередник, розшифровує їх на льоту і розміщує в пам'яті, тож зашифровані фали для вас виглядають як звичайні незашифровані. Також, TrueCrypt стежить за тим, щоб на жорсткому диску вашого комп'ютера не залишалося слідів від документів, що редагуються - у разі необхідності він їх затирає. Крім того, що ви можете використовувати зашифрований контейнер, ви можете зашифрувати весь системний диск і тоді ви працюватимете з файлами в звичайному режимі - досить один раз при завантаженні системи ввести ключову фразу і все.

Давайте оцінимо, наскільки спрощується процедура, якщо використовувати TrueCrypt

1. Ви підключаєте зашифрований контейнер
2. Відкриваєте документи, редагуєте, зберігаєте зміни
3. Вимкніть зашифрований контейнер
4. Решта запитань за вас вирішує TrueCrypt.

Під час роботи файли можуть копіюватися з підключений томом True Crypto так само, як вони копіюються на будь-який нормальний диск. Файли автоматично дешифруються "на льоту" (у пам'ять) під час читання або копіювання із зашифрованого тому True Crypto. Вірно і зворотне - файли, що записуються або скопійовані тим же True Crypto, "на льоту" шифруються в пам'яті прямо перед записом на диск. Проте це не означає, що увесь файл, призначений для шифрування або дешифрування, повинен цілком потрапити в пам'ять перед

шифруванням або дешифруванням. Для True Crypto не потрібно додаткову пам'ять. У наступному абзаці коротко описано, як це усе працює.

Уявіть, що є файл з розширенням .avi, що зберігається на томі True Crypto, і тому він повністю зашифрований. Користувач, застосовуючи правильний пароль (і/або ключовий файл), підключає (відкриває) том True Crypto. Коли користувач двічі клікає на іконці відеофайлу, операційна система запускає програму, пов'язану з цим типом файлу, - як правило, це медіаплеєр. Медіаплеєр починає завантажувати невелику початкову частину відеофайлу із зашифрованого True Crypto - томи в пам'ять, щоб програти його. В процесі завантаження цієї невеликої частини, True Crypto автоматично дешифрує її в пам'яті. Розшифрована частина відео, що зберігається тепер в пам'яті, відтворюється медіаплеєром [22]. Після програвання цієї частини, медіаплеєр почне завантажувати наступну невелику частину відеофайлу із зашифрованого тому True Crypto в пам'ять, і процес повториться. Цей процес і називається шифруванням або ж дешифруванням "на льоту", він працює не лише з відео, але і з усіма типами файлів. Фрагмент лістингу програми представлений в додатку А.

2.3. Класифікація криптоалгоритмів

Для побудови механізмів безпеки із заданими цілями використовують структурні блоки, які грають роль набору певних примітивів. У криптографічних пристроях і протоколах такими примітивами виступають криптоалгоритми, симетричні і асиметричні шифри, хеш-функції і тому подібне. Характерною особливістю цих механізмів є те, що насправді вони визначають лише функції, які необхідно виконати, без урахування того як ці функції будуть реалізовані і впроваджені.

При розробці криптосистем і протоколів безпеки часто враховують умовні припущення відносно їх роботи. Кінцева реалізація представляється у формі ідеальної «чорної скринька», що виключає будь-які випадки внутрішнього спостереження або втручання з боку зловмисника. При таких припущеннях

рівень захищеності визначається тільки математичними властивостями криптоалгоритмів і розміром їх секретних параметрів.

Проте в реальності ці механізми не здатні самі по собі забезпечити практичний рівень безпеки. Більшість відомих атак на криптографічні системи використовують уразливості саме в реалізації криптоалгоритмів. Такі уразливості дозволяють зловмисникові повністю зламати або значно ослабити теоретичну стійкість кінцевих рішень безпеки. Стрімкий розвиток галузі криптографічних пристроїв, в першу чергу засобів ідентифікації (смарткарти, TPM модулі, токени, спеціалізовані мікроконтролери) ставить перед нами задачу всебічного аналізу наявних на сьогодні каналів просочування інформації, складнощі їх використання для атаки і методів протидії.

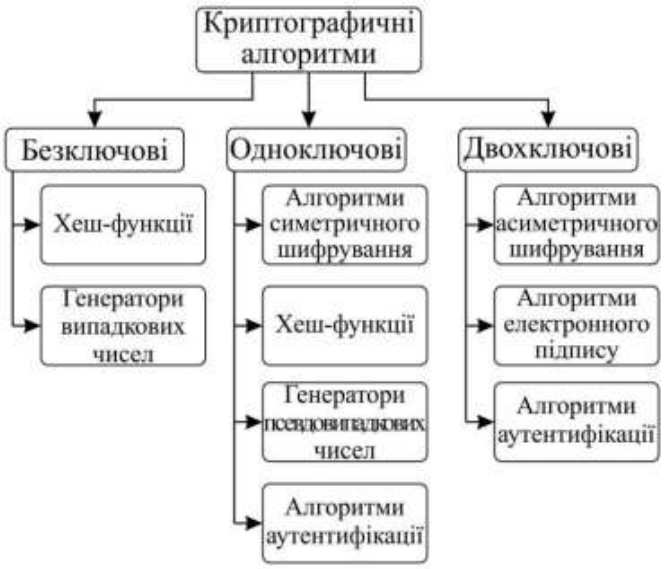

Рисунок 2.1 - Класифікація алгоритмів шифрування

Генератори випадкових чисел. Випадкові числа потрібні, в основному, для генерації секретних ключів шифрування, які, в ідеалі, мають бути абсолютно випадковими. Потрібні вони і для обчислення електронного цифрового підпису, і для роботи багатьох алгоритмів аутентифікації.

Першою принциповою ознакою, що дозволяє зробити розділення шифрів, є об'єм інформації, невідомій третій стороні. У тому випадку, коли зловмисникові повністю невідомий алгоритм виконаного над повідомленням перетворення, шифр називається тайнописом.

На відміну від тайнопису криптографією з ключем називають сьогодні алгоритми шифрування, в яких сам алгоритм перетворень широко відомий і доступний для досліджень кожному охочому, але шифрування робиться на основі невеликого об'єму інформації - ключа, відомого тільки посилачеві і одержувачеві ключа. В деяких випадках ключ формує людина, що відправляє повідомлення, в інших ключ створюється автоматично за допомогою програмного забезпечення або навіть запрошується з віддаленої бази цих ключів. У сучасній криптографії залежно від методик розмір ключа складає від 56 до 4096 біт. Запам'ятовувати ключ, який є насправді просто великою послідовністю чисел, звичайній людині досить складно що в двійковій, що в десятковій системі. Тому усі сучасні системи пропонують користувачеві вводити не ключ - набір цифр, а пароль - довільну текстову фразу. Пароль може складатися з одного або декількох слів, бути осмисленим або ні, головне - щоб він легко запам'ятовувався користувачем.

У криптоалгоритмах ключі, використовувані на передаваючій і приймальній сторонах, повністю ідентичні. Такий ключ несе в собі усю інформацію про засекречування повідомлення і тому не має бути відомий нікому, окрім двох сторін, що беруть участь в розмові. Тому відносно ключа симетричних систем часто називаються шифрами на секретному ключі. Загальна схема процесу передачі повідомлення приведена на Рисунку. 2.2.

Симетричне шифрування можна застосовувати як при відправці повідомлень між двома користувачами, розділеними великою відстанню, так і при відправці «послань» одним і тим же користувачем самому собі, але віддалено в часі.

Рисунок 2.2. Загальна схема передачі повідомлення.

Прикладом подібних відправлень є шифрування файлів на жорстких дисках і змінних носіях з тим, щоб інші користувачі тих же ресурсів не могли прочитати інформацію у відсутність власника.

2.4 Функціональна і математична модель криптографічного захисту

Після впровадження підсистеми захисту інформації у загальній функціональній моделі ведення бухгалтерського обліку стануться невеликі зміни. Ці зміни пов'язані тільки з тим, що документи, отримані в результаті обробки, зашифровуватимуться за допомогою криптографічної програми. Наприклад, інформаційні потоки результатів фінансової діяльності підприємства після впровадження підсистеми криптографічного захисту інформації виглядатимуть так, як показано на Рисунку 2.3.

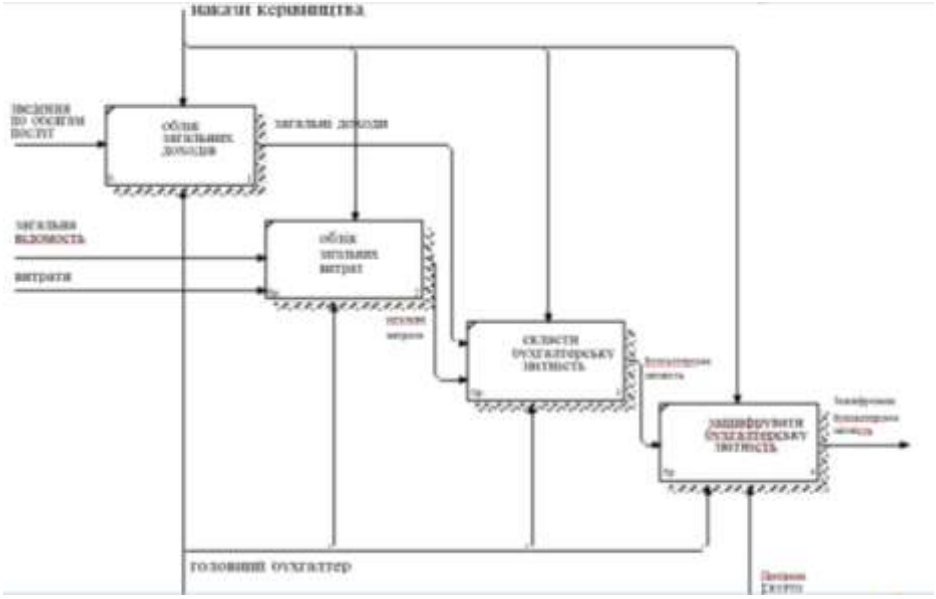

Рисунок 2.3 - Зведення фінансових результатів діяльності підприємства

Функціональна модель розробки програмного забезпечення є структурованим зображенням функцій, що виконуються в ході його проектування, а також інформації, що зв'язує між собою ці функції. Мета функціональної моделі: визначити функції підсистеми, показати перелік і взаємоув'язку завдань підсистеми, склад вхідних і вихідних документів. Функціональна модель відбиває функціональний зміст даного процесу і є структурованим зображенням функцій процесу, зв'язків між ними і з

середовищем, семантики, що відбиває ці функції. Методологічну основу проектування програмного забезпечення складає системний підхід, відповідно до якого, кожна система є сукупністю взаємозв'язаних об'єктів, що функціонують для досягнення спільної мети. Під проектом розуміють проектно-конструкторську і технологічну документацію, в якій представлений опис проектних рішень по створенню і експлуатації програмного забезпечення в конкретному програмно-технічному середовищі.

Технологія проектування програмного забезпечення є сукупністю методологій і засобів проектування, а також методів і засобів проектування (управління процесами створення і модернізації проекту). Для розробки алгоритму або рішення поставленої задачі, по-перше програміст має визначити по якій послідовності створюватиметься програма. Алгоритмом можна назвати порядок дій, які визначають процес перетворення початкових даних аж до результатів. Властивості алгоритмів можна визначити в наступному виді:

- результативністю;
- масовістю;
- однозначністю.

Однозначністю алгоритму являється тлумачення правил виконаних дій. При виконанні алгоритму має бути результат і програма має розв’язувати не одну конкретну, а цілий набор однотипних задач. Отримання певного результату визначає результативність алгоритму.

Основні процедури що складають алгоритм програми TrueCrypto :

а) шифрування;

б) розшифровка;

в) перевірка електронного цифрового підпису.

На рисунку 2.4 представлена основна загальна блок-схема алгоритму програми True Crypto

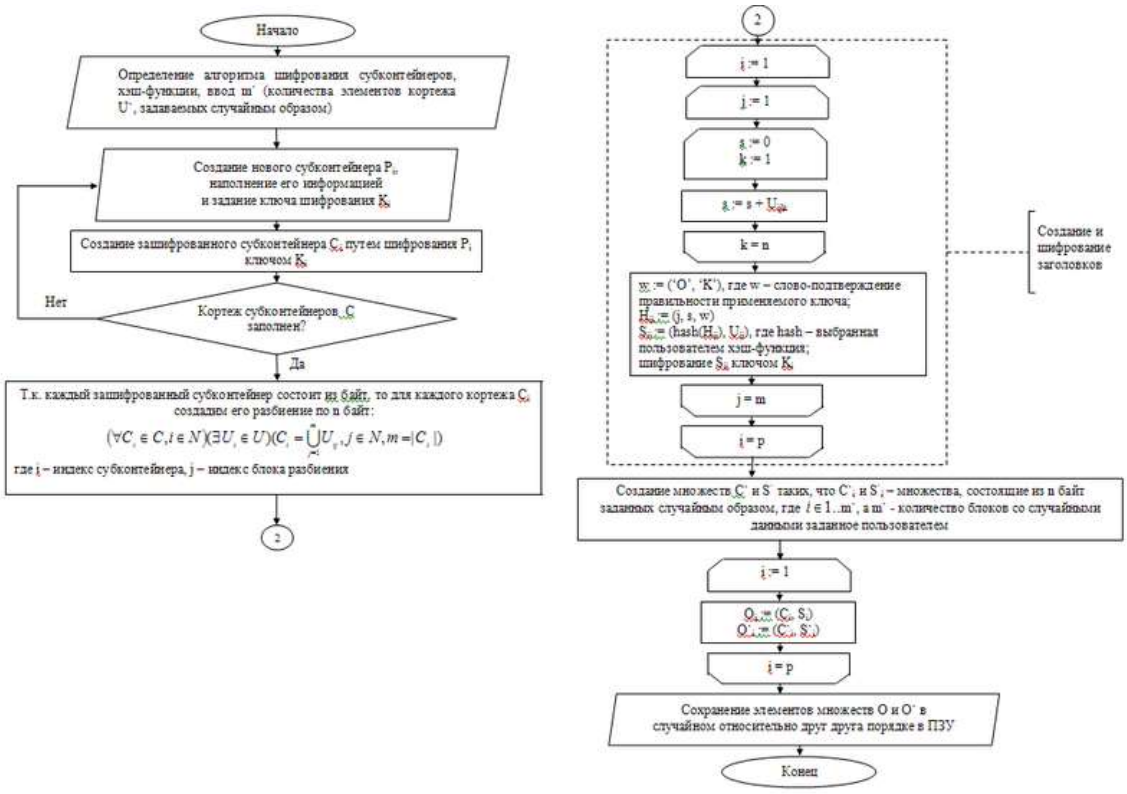

Рисунок 2.4 -Загальна блок-схема алгоритма для True Crypto

True Crypto для досягнення поставленої мети містить математичні і алгоритмічні перетворення, які відбуваються над даними. Тут використовується функція шифрування. Схему роботи алгоритму шифрування можна описати функціями

Z = EnCrypt (X, Key) – шифрування і

X = DeCrypt (Z, Key). - дешифрування

Для перетворень алгоритм шифрування використовує певний набір биективних (зворотних) математичних функцій. В якості биективної функції використовується функція додавання. Ключ є параметром блокового криптоалгоритму і є певним блоком двійкової інформації фіксованого розміру. Ключ утворюється на основі пароля, введеного користувачем. Криптоалгоритм є ідеально стійким, якщо прочитати зашифрований блок даних можна тільки перебравши усі можливі ключі, до тих пір, поки вихідне повідомлення не виявиться осмисленим. Оскільки по теорії вірогідності шуканий ключ буде знайдений з вірогідністю 1/2 після перебору половини усіх ключів, то на злом

ідеально стійкого криптоалгоритма з ключем довжини N знадобиться в середньому 2N-1 перевірок.

Таким чином, на функцію стійкого блокового шифрування Z = EnCrypt (X, Key) накладаються наступні умови:

а) функція EnCrypt має бути оборотною;

б) не може існувати інших методів прочитання повідомлення X по відомому блоку Z, окрім як повним перебором ключів;

в) не має існувати інших методів визначення, яким саме ключем Key було зроблено перетворення відомого повідомлення X в повідомлення Z, окрім як повним перебором ключів.

Пропоноване для такої перевірки програмне забезпечення підтверджує цифровий підпис, як криптографічно "перевірене", якщо:

- для підписання повідомлення в цифровій формі використовувався приватний ключ особи, яка підписала, і це вважається доведеним, якщо підпис пройшов перевірку публічним ключем особи, що підписала, бо публічний ключ цієї особи сходиться з цифровим підписом, створеним через приватний ключ тої особи;
- у повідомлення не були внесені зміни, і це вважається доведеним, якщо результат хешування, обчислений перевіряючим, ідентичний результату хешування, отриманому з цифрового підпису при перевірці.

Хеш-функцією називається одностороння функція, призначена для отримання дайджеста або "відбитків пальців" файлу, повідомлення чи деякого блоку даних. Спочатку функції хешування використовувалися як функції створення унікального образу інформаційних послідовностей довільної довжини, з метою ідентифікації та визначення їх достовірності. Сам образ має бути невеликим блоком фіксованої довжини, як правило, 30, 60, 64, 128, 256, або 512 біт. Тому операції пошуку, сортування та деякі інші з великими масивами або базами даних таким чином істотно спрощуються, займають набагато менше часу. Для забезпечення необхідної вірогідності помилки необхідно забезпечувати ряд вимог до функції хешування.

Хеш-функція має задовольняти цілому ряду умов :

- чутливість до таких змін як вставки, викиди, перестановки;
- необхідні значення мають бути нерозв'язними обчислювально;
- збіг документів має бути мінімальним.

Ці властивості дозволяють подавати на вхід хеш-функции паролі, тобто текстові рядки довільної довжини будь-якою національною мовою.

У програмі True Crypto хешування пароля здійснюється таким чином:

Спочатку пароль, введений користувачем, перетворюється в масив байт P [1 .. X]. Де P [1 .. X] – коди символів з введеного користувачем пароля (кодировка ANSI – Windows-1251). X - довжина пароля (в символах).

Потім, грунтуючись на паролі отримують ключ:

K [1 .. 255] – ключ.

K [i] = (P [1] + P [2] + … + P [n]) + P [i],//где i = 1 .. X

якщо X < 255, тоді проводять наступні перетворення

K [X + 1 .. 255] = 114, //114 - код для символа «r»

Z = K [1],

якщо K [i] = Z, то K [i] = K [i – 1] + Z,

Z = K [i],// де i = 2 .. 255,

При обробці ввідбувається стискання даних. Методи стискання даних можна розділити на два типи:

- неспотворюючі (loseless) методи стискання (що називаються також методами стискання без втрат) гарантують, що декодовані дані в точності співпадатимуть з початковими;
- спотворюючі (lossy) методи стискання (що називаються також методами стискання з втратами) можуть спотворювати початкові дані, наприклад за рахунок видалення несуттєвої частини даних, після чого повне відновлення неможливе.

Перший тип застосовують, коли дані важливо відновити в неспотвореному виді, це важливо для текстів, числових даних і тому подібне. Повністю оборотне

стискання, за визначенням, нічого не видаляє з початкових даних. Стискання досягається тільки за рахунок іншого, економічнішого, представлення даних.

Другий тип стискання застосовують, в основному, для відео зображень і звуку. За рахунок втрат може бути досягнута більш висока міра стискання. В цьому випадку втрати визначають несуттєве спотворення зображення (звуку) яке не перешкоджає нормальному сприйняттю, але при звіренні оригінала і відновленої копії можуть бути помічені.

Крім того, можна виділити:

- методи стискання загального призначення (general - purpose), які не залежать від фізичної природи вхідних даних і, як правило, орієнтовані на стискання текстів, виконуваних програм, об'єктних модулів і бібліотек і т. д., тобто даних, які в основному і зберігаються в ЕОМ;
- спеціальні (special) методи стискання, які орієнтовані на стискання даних відомої природи, наприклад, звуку, зображень і т. д. Тут за рахунок знання специфічних особливостей даних, що стискаються, досягають істотно кращої якості і/або швидкості стискання, чим при використанні методів загального призначення.

Програмою TrueCrypt підтримуються такі алгоритми шифрування: AES, Serpent та Twofish. Крім цього, є 5 різних комбінацій каскадних алгоритмів: AES-Twofish, AES-Twofish-Serpent, Serpent-AES, Serpent-Twofish-AES і Twofish-Serpent. Підтримуються такі хеш-функції: RIPEMD-160, SHA-256, SHA-512, Whirlpool. Бібліотека цих файлів є у відкритому доступі.

Проте треба вставити в систему власний алгоритм шифрування що не входить в стандартний набір.

2.5. Алгоритм шифрування

CAST це популярний 64 - бітовий шифр, що допускає розміри ключа аж до 128 біт, який був розроблений в Канаді Карлайслом Адамсом (Carlisle Adams) і Стаффордом Таваресом (Stafford Tavares). Автори стверджують, що назва обумовлена ходом розробки і повинна нагадувати про імовірнісний характер

процесу, а не про ініціали авторів. Описуваний алгоритм CAST використовує 64-бітовий блок і 64-бітовий ключ. CAST стійкий до диференціального і лінійного криптоаналізу. Сила алгоритму CAST знаходиться в його S -блоках. У CAST немає фіксованих S -блоків і для кожного застосування вони конструюються наново. Створений для конкретної реалізації CAST S-блок вже більше ніколи не міняється. Іншими словами S -блоки залежать від реалізації, а не від ключа. Northern Telecom використовує CAST у своєму пакеті програм Entrust для комп'ютерів Macintosh, PC і робочих станцій UNIX. Вибрані ними S -блоки не опубліковані. Загальна схема роботи показана на рисунку 2.5.

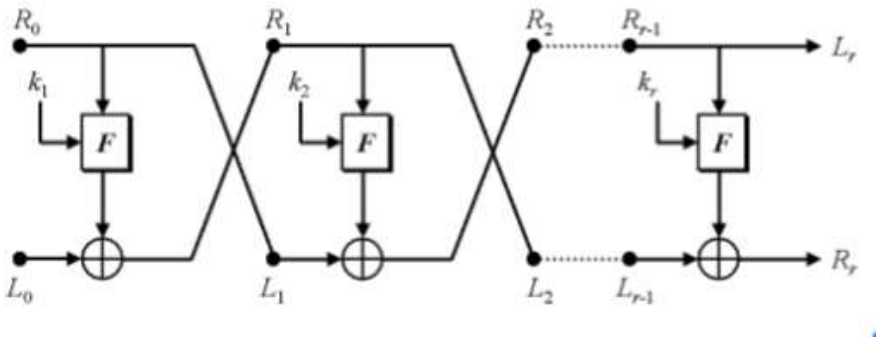

Рисунок 2.5. Загальна схема алгоритму шифрування

Шифрування відбувається наступним чином:

З початку тексту беремо 8 символів, що відповідає 64 бітам інформації. Ключ складається з 32 біт - 4 символи. Для кодування візьмемо UTF – 8 кодування зі змінною довжиною. Якщо символ може бути закодований одним байтом (бо номер пункту символу дуже маленький) то UTF - 8 закодує його одним байтом. Якщо треба 2 байти, то візьме 2 байти. Кодування повідомляє старшими бітами, скількома бітами кодується поточний символ. Такий спосіб економить місце, але так само і витрачає його у разі, якщо ці сигнальні біти часто використовуються. В даному випадку англійські символи кодують 7-ма бітами. Щоб було зручніше робити обчислення, було виконано наступне: для кожного символу, який займає менше 8 біт, - додати нулі в початок цього символу, щоб стало рівно 8 символів. Так і зроблено і для тексту і для ключа. Далі символи по

UTF - 8 переводилися в десяткові числа. Вісім символів що були взяті на початку діляться на 2 блоки L0 і R0, і кожній змінній належить відповідно по 4 символи.

В R0 поступає 4 символи і ключ маскування , що додається до кожного символа по модулю 2 (XOR). Далі відбувається зсув на 1 вліво.

Кожен символ всередині програми відділений пропуском, так програма розуміє, до якого з чисел звертається. Усі числа записані в два двовимірні масиви LL і RR. Кожен є одновимірним масивом, в осередках якого знаходиться ще один одновимірний масив. Перший масив складається з 16 оскільки у нас 16 раундів, другий з 4, оскільки по 4 символи використовуються для обчислення. Після того, як відбулися додавання і зсув, відбувається додавання по модулю 2 з лівою стороною тексту.

Далі текст з лівого боку переходить на праву сторону, а початкове значення правої сторони, на ліву сторону. Такі перестановки будуть 12 раундів. Потім щоб уникнути великих чисел після першого складання відбувається звичайне віднімання з числа раунду, раунду попереднього, а потім зсув на певне число. Після цього так само складається з лівою стороною і міняється місцями до 16 раунду.

Таким чином алгоритм виконує свою роботу для кожних восьми символів тексту, поки текст не закінчиться. Шифрування виводиться на екран символами відповідно до їх номерів в таблиці. Номери беруться з останнього раунду, після усіх перестановок.

Висновки до 2 розділу

В даному розділі було розглянуто структуру програмних засобів які застосовуються для захисту доступу доданих. Встановлено, що найбільш вигідним є запровадження ПЗ для шифрування даних: це найменш ресурсовитратний метод. Розглянуто класифікацію криптографічних алгоритмів. Встановлено, що найбільш ефективними є алгоритми, які гарантують безпеку даних за рахунок односторонніх обчислень, шифри з ключем, ніж ті, що в якості гаранту безпеки використовують той факт, що сам алгоритм є секретним.Як приклад розглядались програми що застосовують алгоритми шифрування.

Також докладніше розглянуто алгоритм шифрування CAST що пропонується інтегрувати в систему з відкритим кодом. Даний алгоритм додасть елемент унікальності нашій системі захисту інформації на підприємстві.

РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ

Завдання захисту інформації йде не лише «зсередини», від потреб бізнесу, але і «ззовні», від вимог міжнародних стандартів і законів. Вирішувати цю складну задачу треба комплексно, задіюючи і організаційні заходи, і технічні засоби. Одним з найефективніших способів захисту конфіденційної інформації є шифрування. Дійсно, якщо цифрові дані по суті є послідовністю нулів і одиниць, логічно притягнути математику. Тобто розробити операції, які дозволяють працювати з даними тільки «своїм», тим що мають ключ для розшифровки. Для усіх «чужих» зашифрована інформація незрозуміла а іноді і зовсім недоступна. При цьому хороша система шифрування може вирішувати відразу декілька завдань: запобігати витокам даних, перекривати спроби несанкціонованого доступу (аж до знищення інформації), забезпечувати захист резервних копій даних.

Виробники представляють на ринку безліч систем шифрування даних, що в основному реалізовують окремі аспекти ІБ. В даному випадку ми розглянемо комплексне програмно-апаратне рішення. Система True Crypto реалізує «прозоре» шифрування даних, що зберігаються на корпоративних серверах, забезпечуючи захист конфіденційної інформації. TrueCrypt поширюється під вільною ліцензією з відкритим початковим кодом.

У 2014 розробка програми була припинена, але незважаючи на це, програма як і раніше зберігає працездатність, Той факт, що TrueCrypt може мати уразливості у безпеці викликає серйозні побоювання, особливо враховуючи, що аудит програми не виявив подібних проблем. Саме тому і було вирішено додати свій алгоритм шифрування щоб і далі використати усе те хороше що є в оригінальній програмі.

Шифрувати можна не лише файли і теки, але і службову інформацію (наприклад, таблиці розміщення файлів). При записі даних на диск відбувається автоматичне шифрування в режимі реального часу (online), при читанні -

розшифровка. Для сторонніх осіб, що не мають прав доступу і ключів шифрування, інформація невидима, все одно що не існує.

Система True Crypto здійснює:

- шифрування носіїв даних (жорстких дисків і дискових масивів, CD/ DVD, магнітних стрічок, а також сховищ SAN);
- управління загальними ресурсами і доступом додатків до дисків;
- підтримку технології кластеризації Microsoft Cluster Service;
- використання різних алгоритмів шифрування (RC5, AES, XTS - AES і ГОСТ 28147-89);
- реалізацію заданих сценаріїв роботи.

Ключі шифрування даних на диску вводяться при завантаженні сервера та захищені кодом. Підібрати код неможливо: число спроб введення обмежене. Перезавантаження сервера теж нічого не дасть зловмисникові - в цьому випадку захищені диски недоступні: ключ шифрування знаходиться тільки в оперативній пам'яті і при перезавантаженні стирається з неї.

Надійність захисту даних, що зберігаються, гарантують сучасні криптостійкі алгоритми шифрування з довжиною ключа від 128 біт. Ми скористаємось можливістю відкритого коду і додамо туди свій алгоритм.

Адміністратор системи може вводити ключі шифрування і управляти рішенням (змінювати налаштування, підключати і відключати диски, розподіляти права доступу) видалено - як з будь-якого комп'ютера локальної мережі, так і через інтернет.

True Crypto складається з п'яти основних модулів (консоль управління, сервери шифрування, ключів і журналів, модуль подання сигналу тривоги) і додаткового набору сценаріїв роботи. Модулі взаємодіють за технологією «клієнт - сервер» з використанням протоколу TCP/IP. Для кожної сесії формується унікальний ключ шифрування, який забезпечує захист з'єднань і робить неможливим аналіз мережевого трафіку. В додатку А наведено частину модуля шифрування в який додано наш алгоритм.

3.1. Вибір мови програмування

В якості основного засобу розробки програмного забезпечення було обрано C++ оскільки програма в яку інтегрується наш алгоритм написана саме цією мовою. Але крім того є й інші аргументи.

З моменту свого створення C++ став однією з найширше використовуваних мов програмування у світі. Грамотно сконструйовані програми на мовах C++ швидкі і ефективні. Мова гнучкіша, ніж інші мови: вона може працювати і на найвищих рівнях абстракції і вниз на апаратному рівні. C++ надає стандартні бібліотеки з високим рівнем оптимізації. Вона забезпечує доступ до апаратних функцій низького рівня, щоб максимально збільшити швидкість і скоротити споживання пам'яті. C++ може створювати практично будь-який вид програми : ігри, драйвери пристроїв, HPC, хмара, настільний комп'ютер, впроваджені і мобільні застосування і багато що інше.

Спочатку, ця мова створювалася для системного програмування. Тому немає нічого дивовижного в тому, що вона активно використовується при розробці нових операційних систем і різного програмного забезпечення. При цьому, використання цієї мови дозволяє згодом проводити гнучке налаштування операційної системи.

Мова C++ чудово підходить для програмування вбудованих систем. В першу чергу те пов'язане з тим, що вона має високу продуктивність і при цьому простоту використання. Такий інструмент економічний з точки зору використання ресурсів. Це дозволяє виконувати будь-які програми з високою швидкістю. В результаті, вбудовані системи можуть працювати без уповільнення в режимі реального часу.

До переваг цієї мови також можна віднести:

Високу швидкість. Можна відкрити будь-який тестер швидкостей мов програмування, і побачити, що C++ є одною з найбільш високошвидкісних. При цьому, можна використати будь-яку мову для вирішення локальних завдань. Але

якщо необхідно написати усе застосування на одній мові, з цим завданням відмінно впорається C++.

Універсальність. Компілятори мови C++ є у будь-якій операційній системі. При цьому, написані на цій мові програми можуть без проблем виконуватися на будь-якій платформі.

Велике співтовариство. Мова постійно оновлюються і сюди впроваджуються різні корисні нововведення. Але і це не усе. C++ доповнюється бібліотеками і шаблонами, які можуть згодитися як досвідченим програмістам, так і розробникам-початківцям. Окрім цього, під C++ написана безліч корисних книг і самовчителів, які допоможуть швидше освоїти тонкощі мови.

Принципи C++ закладені у багато сучасних мов програмування. Тому ті. Хто його досконально вивчать, зможуть без зусиль освоїти Java, JavaScript або C#.

3.2. Реалізація Алгоритму шифрування даних

Починаємо розгляд алгоритму з заголовочного файла cast_128.h де опишемо клас CAST_128

```
#ifndef CAST_128_H
#define CAST_128_H
#include <Qt>
class CAST_128 {
public:
        enum {
        KEY_LEN = 128 / 32,
        MSG_LEN = 2
};
typedef qu_int32 Key[ KEY_LEN ];
typedef qu_int32 Message[ MSG_LEN ];
public:
        CAST_128();
        void encrypt( const Key key, Message msg );
        void decrypt( const Key key, Message msg );
private:
        void run( const Key key, Message msg, bool reverse = false );
private:
        typedef qu_int32 SType[ 256 ];
        static const SType S1;
        static const SType S2;
```

```
        static const SType S3;
        static const SType S4;
        static const SType S5;
        static const SType S6;
        static const SType S7;
        static const SType S8;
};

#endif // CAST128_H
```

Спочатку треба визначити типи Key і Message. Вони є масивами, що складаються з 4 і 2 (відповідно) 32-бітових беззнакових цілих чисел qu_int32.
Як для шифрування (encrypt()), так і для дешифрування (decrypt()) алгоритм CAST_128 приймає на вхід ключ key і кодоване повідомлення msg. У даній реалізації ми припускаємо, що ключ завжди має довжину 128 біт (для ключів меншого розміру зміни в алгоритмі виявляться мінімальними).

Принцип роботи функцій encrypt() і decrypt() дуже схожий, але є і відмінність: для розшифровки треба робити усе те ж саме, що і для шифрування, але тільки в зворотному порядку. Для цього ми створюємо допоміжну функцію run(), якій можна передати додатковий прапорець reverse (false для шифрування і true для дешифрування).

Передбачається, що шифрування і дешифрування буде здійснюватись «на місці». Тобто повідомлення msg кодуються без копіювання.

Для прикладу було заготовлено вісім масивів (S -блоків) S1, S2, ., S8, що складаються з 256 беззнакових цілих чисел qu_int32. Вони заздалегідь визначені і також наводяться в додатку, що описує роботу алгоритму. Як вони виглядають покажемо на прикладі S1 :

```
const CAST128::SType CAST128::S1 = {
    0x30fb40d4, 0x9fa0ff0b, 0x6beccd2f, 0x3f258c7a, 0x1e213f2f, 0x9c004dd3, 0x6003e540, 0xcf9fc949,
    0xbfd4af27, 0x88bbbdb5, 0xe2034090, 0x98d09675, 0x6e63a0e0, 0x15c361d2, 0xc2e7661d, 0x22d4ff8e,
    0x28683b6f, 0xc07fd059, 0xff2379c8, 0x775f50e2, 0x43c340d3, 0xdf2f8656, 0x887ca41a, 0xa2d2bd2d,
    0xa1c9e0d6, 0x346c4819, 0x61b76d87, 0x22540f2f, 0x2abe32e1, 0xaa54166b, 0x22568e3a, 0xa2d341d0,
    0x66db40c8, 0xa784392f, 0x004dff2f, 0x2db9d2de, 0x97943fac, 0x4a97c1d8, 0x527644b7, 0xb5f437a7,
    0xb82cbaef, 0xd751d159, 0x6ff7f0ed, 0x5a097a1f, 0x827b68d0, 0x90ecf52e, 0x22b0c054, 0xbc8e5935,
    0x4b6d2f7f, 0x50bb64a2, 0xd2664910, 0xbee5812d, 0xb7332290, 0xe93b159f, 0xb48ee411, 0x4bff345d,
```

0xfd45c240, 0xad31973f, 0xc4f6d02e, 0x55fc8165, 0xd5b1caad, 0xa1ac2dae, 0xa2d4b76d, 0xc19b0c50, 0x882240f2, 0x0c6e4f38, 0xa4e4bfd7, 0x4f5ba272, 0x564c1d2f, 0xc59c5319, 0xb949e354, 0xb04669fe, 0xb1b6ab8a, 0xc71358dd, 0x6385c545, 0x110f935d, 0x57538ad5, 0x6a390493, 0xe63d37e0, 0x2a54f6b3, 0x3a787d5f, 0x6276a0b5, 0x19a6fcdf, 0x7a42206a, 0x29f9d4d5, 0xf61b1891, 0xbb72275e, 0xaa508167, 0x38901091, 0xc6b505eb, 0x84c7cb8c, 0x2ad75a0f, 0x874a1427, 0xa2d1936b, 0x2ad286af, 0xaa56d291, 0xd7894360, 0x425c750d, 0x93b39e26, 0x187184c9, 0x6c00b32d, 0x73e2bb14, 0xa0bebc3c, 0x54623779, 0x64459eab, 0x3f328b82, 0x7718cf82, 0x59a2cea6, 0x04ee002e, 0x89fe78e6, 0x3fab0950, 0x325ff6c2, 0x81383f05, 0x6963c5c8, 0x76cb5ad6, 0xd49974c9, 0xca180dcf, 0x380782d5, 0xc7fa5cf6, 0x8ac31511, 0x35e79e13, 0x47da91d0, 0xf40f9086, 0xa7e2419e, 0x31366241, 0x051ef495, 0xaa573b04, 0x4a805d8d, 0x548300d0, 0x00322a3c, 0xbf64cddf, 0xba57a68e, 0x75c6372b, 0x50afd341, 0xa7c13275, 0x915a0bf5, 0x6b54bfab, 0x2b0b1426, 0xab4cc9d7, 0x449ccd82, 0xf7fbf265, 0xab85c5f3, 0x1b55db94, 0xaad4e324, 0xcfa4bd3f, 0x2deaa3e2, 0x9e204d02, 0xc8bd25ac, 0xeadf55b3, 0xd5bd9e98, 0xe31231b2, 0x2ad5ad6c, 0x954329de, 0xadbe4528, 0xd8710f69, 0xaa51c90f, 0xaa786bf6, 0x22513f1e, 0xaa51a79b, 0x2ad344cc, 0x7b5a41f0, 0xd37cfbad, 0x1b069505, 0x41ece491, 0xb4c332e6, 0x032268d4, 0xc9600acc, 0xce387e6d, 0xbf6bb16c, 0x6a70fb78, 0x0d03d9c9, 0xd4df39de, 0xe01063da, 0x4736f464, 0x5ad328d8, 0xb347cc96, 0x75bb0fc3, 0x98511bfb, 0x4ffbcc35, 0xb58bcf6a, 0xe11f0abc, 0xbfc5fe4a, 0xa70aec10, 0xac39570a, 0x3f04442f, 0x6188b153, 0xe0397a2e, 0x5727cb79, 0x9ceb418f, 0x1cacd68d, 0x2ad37c96, 0x0175cb9d, 0xc69dff09, 0xc75b65f0, 0xd9db40d8, 0xec0e7779, 0x4744ead4, 0xb11c3274, 0xdd24cb9e, 0x7e1c54bd, 0xf01144f9, 0xd2240eb1, 0x9675b3fd, 0xa3ac3755, 0xd47c27af, 0x51c85f4d, 0x56907596, 0xa5bb15e6, 0x580304f0, 0xca042cf1, 0x011a37ea, 0x8dbfaadb, 0x35ba3e4a, 0x3526ffa0, 0xc37b4d09, 0xbc306ed9, 0x98a52666, 0x5648f725, 0xff5e569d, 0x0ced63d0, 0x7c63b2cf, 0x700b45e1, 0xd5ea50f1, 0x85a92872, 0xaf1fbda7, 0xd4234870, 0xa7870bf3, 0x2d3b4d79, 0x42e04198, 0x0cd0ede7, 0x26470db8, 0xf881814c, 0x474d6ad7, 0x7c0c5e5c, 0xd1231959, 0x381b7298, 0xf5d2f4db, 0xab838653, 0x6e2f1e23, 0x83719c9e, 0xbd91e046, 0x9a56456e, 0xdc39200c, 0x20c8c571, 0x962bda1c, 0xe1e696ff, 0xb141ab08, 0x7cca89b9, 0x1a69e783, 0x02cc4843, 0xa2f7c579, 0x429ef47d, 0x427b169c, 0x5ac9f049, 0xdd8f0f00, 0x5c8165bf

Щоб продемонструвати роботу алгоритма треба розглянути кілька тестів.

Файл cast_128_main.cpp:

```
#include "cast_128.h"
#include <iostream>
#include <iomanip>
void showComponent( quint32 x ) {
std::cout << std::hex << std::setw( 8 ) << std::setfill( '0' ) << std::uppercase << x
<< " ";
}
void showMessage( const CAST128::Message msg ) {
        for( int i = 0; i < CAST128::MSG_LEN; ++i ) {
                showComponent( msg[ i ] ); std::cout << " ";
        }
        std::cout << std::endl;
```

```
}
void showKey( const CAST_128::Key key ) {
	for( int i = 0; i < CAST_128::KEY_LEN; ++i ) {
		showComponent( key[ i ] ); std::cout << " ";
	}
	std::cout << std::endl;
}
int main() {
	CAST_128 cast_128;
std::cout << "============== Test 1 ==============" << std::endl;
static const CAST_128::Key KEY = { 0x01234567, 0x12345678, 0x23456789,
0x3456789A };
CAST_128::Message msg = { 0x01234567, 0x89ABCDEF };

std::cout << Msg before: "; showMessage( msg );
cast_128.encrypt( KEY, msg );
std::cout << "Msg after encryption: "; showMessage( msg );
cast_128.decrypt( KEY, msg );
std::cout << "Msg after decryption: "; showMessage( msg );
std::cout << std::endl;
std::cout << "============== Test 2 ==============" << std::endl;
CAST_128::Key a = { 0x01234567, 0x12345678, 0x23456789, 0x3456789A };
CAST_128::Key b = { 0x01234567, 0x12345678, 0x23456789, 0x3456789A };
for( int i = 0; i < 1000000; ++i ) {
cast_128.encrypt( b, a );
cast_128.encrypt( b, a + 2 );
cast_128.encrypt( a, b );
cast_128.encrypt( a, b + 2 );
}

showKey( a );
showKey( b );

return 0;
```

В даному тесті був застосований ключ 0x01234567, 0x12345678, 0x23456789, 0x3456789A та повідомлення 0x01234567, 0x89ABCDEF. Спочатку повідомлення кодується через функцію encrypt(). Потім отримане повідомлення дешифрується за допомогою decrypt(). Після таких дій маємо отримати початкове повідомлення:

========== Test 1 ==========

Msg before: 01234567 89ABCDEF

Msg after encryption: 238B4FE5 847E44B2

Msg after decryption: 01234567 89ABCDEF

Якщо спробувати складніший тест де два ключі a та b шифруються по черзі один з одним 100 раз. Тож в результаті отримаєм:

Остаточно реалізація алгоритма

Файл cast_128.cpp:

```
#include "cast128.h"
#include <cstring>
#include <iostream>
#include <iomanip>
static const int ROUND_COUNT = 16;
static const quint64 MOD_2_32 = quint64( 2 ) << 31;
static const quint8 K_MAP[ sizeof( CAST_128::Key ) ] = {
    3,  2,  1,  0,
    7,  6,  5,  4,
   11, 10,  9,  8,
   15, 14, 13, 12
};
static quint8 g( const CAST_128::Key key, quint8 i ) {
   return ( ( quint8* ) key )[ K_MAP[ i ] ];
}
static void splitl( qu_int32 l, quint8* la, quint8* lb, quint8* lc, quint8* ld ) {
   *la = ( l >> 24 ) & 0xFF;
   *lb = ( l >> 16 ) & 0xFF;
   *lc = ( l >> 8  ) & 0xFF;
   *ld = ( l       ) & 0xFF;
}
static qu_int32 sumMod2_32( qu_int32 a, qu_int32 b ) {
   return ( qu_int64( a ) + qu_int64( b ) ) % MOD_2_32;
}
static qu_int32 subtractMod2_32( qu_int32 a, qu_int32 b ) {
   if( b <= a ) {
      return a - b;
   }
   return ( MOD_2_32 + a ) - b;
}
```

```
static quint32 cyclicShift( quint32 x, quint8 shift ) {
    quint8 s = shift % 32;
    return ( x << s ) | ( x >> ( 32 - s ) );
}
// ************************************************************************************************
CAST_128::CAST_128() {
}
 void CAST_128::encrypt( const CAST_128::Key key, CAST128::Message msg ) {
    run( key, msg );
}
void CAST_128::decrypt( const CAST128::Key key, CAST128::Message msg ) {
    run( key, msg, true );
}
void CAST_128::run( const CAST128::Key key, CAST128::Message msg, bool reverse ) {
    Key x = { 0 };
    memcpy( x, key, sizeof( Key ) );
    Key z = { 0 };
    qu_int32 K[ 32 ] = { 0 };
    for( int i = 0; i < 2; ++i ) {
        z[ 0 ] = x[ 0 ] ^ S5[ g( x, 0xD ) ] ^ S6[ g( x, 0xF ) ] ^ S7[ g( x, 0xC ) ] ^ S8[ g( x, 0xE ) ] ^ S7[ g( x, 0x8 ) ];
        z[ 1 ] = x[ 2 ] ^ S5[ g( z, 0x0 ) ] ^ S6[ g( z, 0x2 ) ] ^ S7[ g( z, 0x1 ) ] ^ S8[ g( z, 0x3 ) ] ^ S8[ g( x, 0xA ) ];
        z[ 2 ] = x[ 3 ] ^ S5[ g( z, 0x7 ) ] ^ S6[ g( z, 0x6 ) ] ^ S7[ g( z, 0x5 ) ] ^ S8[ g( z, 0x4 ) ] ^ S5[ g( x, 0x9 ) ];
        z[ 3 ] = x[ 1 ] ^ S5[ g( z, 0xA ) ] ^ S6[ g( z, 0x9 ) ] ^ S7[ g( z, 0xB ) ] ^ S8[ g( z, 0x8 ) ] ^ S6[ g( x, 0xB ) ];

        K[ 0 + i * 16 ] = S5[ g( z, 0x8 ) ] ^ S6[ g( z, 0x9 ) ] ^ S7[ g( z, 0x7 ) ] ^ S8[ g( z, 0x6 ) ] ^ S5[ g( z, 0x2 ) ];
        K[ 1 + i * 16 ] = S5[ g( z, 0xA ) ] ^ S6[ g( z, 0xB ) ] ^ S7[ g( z, 0x5 ) ] ^ S8[ g( z, 0x4 ) ] ^ S6[ g( z, 0x6 ) ];
        K[ 2 + i * 16 ] = S5[ g( z, 0xC ) ] ^ S6[ g( z, 0xD ) ] ^ S7[ g( z, 0x3 ) ] ^ S8[ g( z, 0x2 ) ] ^ S7[ g( z, 0x9 ) ];
        K[ 3 + i * 16 ] = S5[ g( z, 0xE ) ] ^ S6[ g( z, 0xF ) ] ^ S7[ g( z, 0x1 ) ] ^ S8[ g( z, 0x0 ) ] ^ S8[ g( z, 0xC ) ];

        x[ 0 ] = z[ 2 ] ^ S5[ g( z, 0x5 ) ] ^ S6[ g( z, 0x7 ) ] ^ S7[ g( z, 0x4 ) ] ^ S8[ g( z, 0x6 ) ] ^ S7[ g( z, 0x0 ) ];
        x[ 1 ] = z[ 0 ] ^ S5[ g( x, 0x0 ) ] ^ S6[ g( x, 0x2 ) ] ^ S7[ g( x, 0x1 ) ] ^ S8[ g( x, 0x3 ) ] ^ S8[ g( z, 0x2 ) ];
        x[ 2 ] = z[ 1 ] ^ S5[ g( x, 0x7 ) ] ^ S6[ g( x, 0x6 ) ] ^ S7[ g( x, 0x5 ) ] ^ S8[ g( x, 0x4 ) ] ^ S5[ g( z, 0x1 ) ];
        x[ 3 ] = z[ 3 ] ^ S5[ g( x, 0xA ) ] ^ S6[ g( x, 0x9 ) ] ^ S7[ g( x, 0xB ) ] ^ S8[ g( x, 0x8 ) ] ^ S6[ g( z, 0x3 ) ];

        K[ 4 + i * 16 ] = S5[ g( x, 0x3 ) ] ^ S6[ g( x, 0x2 ) ] ^ S7[ g( x, 0xC ) ] ^ S8[ g( x, 0xD ) ] ^ S5[ g( x, 0x8 ) ];
        K[ 5 + i * 16 ] = S5[ g( x, 0x1 ) ] ^ S6[ g( x, 0x0 ) ] ^ S7[ g( x, 0xE ) ] ^ S8[ g( x, 0xF ) ] ^ S6[ g( x, 0xD ) ];
        K[ 6 + i * 16 ] = S5[ g( x, 0x7 ) ] ^ S6[ g( x, 0x6 ) ] ^ S7[ g( x, 0x8 ) ] ^ S8[ g( x, 0x9 ) ] ^ S7[ g( x, 0x3 ) ];
        K[ 7 + i * 16 ] = S5[ g( x, 0x5 ) ] ^ S6[ g( x, 0x4 ) ] ^ S7[ g( x, 0xA ) ] ^ S8[ g( x, 0xB ) ] ^ S8[ g( x, 0x7 ) ];

        z[ 0 ] = x[ 0 ] ^ S5[ g( x, 0xD ) ] ^ S6[ g( x, 0xF ) ] ^ S7[ g( x, 0xC ) ] ^ S8[ g( x, 0xE ) ] ^ S7[ g( x, 0x8 ) ];
        z[ 1 ] = x[ 2 ] ^ S5[ g( z, 0x0 ) ] ^ S6[ g( z, 0x2 ) ] ^ S7[ g( z, 0x1 ) ] ^ S8[ g( z, 0x3 ) ] ^ S8[ g( x, 0xA ) ];
```

```
    z[ 2 ] = x[ 3 ] ^ S5[ g( z, 0x7 ) ] ^ S6[ g( z, 0x6 ) ] ^ S7[ g( z, 0x5 ) ] ^ S8[ g( z, 0x4 ) ] ^ S5[ g( x, 0x9 ) ];
    z[ 3 ] = x[ 1 ] ^ S5[ g( z, 0xA ) ] ^ S6[ g( z, 0x9 ) ] ^ S7[ g( z, 0xB ) ] ^ S8[ g( z, 0x8 ) ] ^ S6[ g( x, 0xB ) ];

    K[ 8  + i * 16 ] = S5[ g( z, 0x3 ) ] ^ S6[ g( z, 0x2 ) ] ^ S7[ g( z, 0xC ) ] ^ S8[ g( z, 0xD ) ] ^ S5[ g( z, 0x9 ) ];
    K[ 9  + i * 16 ] = S5[ g( z, 0x1 ) ] ^ S6[ g( z, 0x0 ) ] ^ S7[ g( z, 0xE ) ] ^ S8[ g( z, 0xF ) ] ^ S6[ g( z, 0xC ) ];
    K[ 10 + i * 16 ] = S5[ g( z, 0x7 ) ] ^ S6[ g( z, 0x6 ) ] ^ S7[ g( z, 0x8 ) ] ^ S8[ g( z, 0x9 ) ] ^ S7[ g( z, 0x2 ) ];
    K[ 11 + i * 16 ] = S5[ g( z, 0x5 ) ] ^ S6[ g( z, 0x4 ) ] ^ S7[ g( z, 0xA ) ] ^ S8[ g( z, 0xB ) ] ^ S8[ g( z, 0x6 ) ];

    x[ 0 ] = z[ 2 ] ^ S5[ g( z, 0x5 ) ] ^ S6[ g( z, 0x7 ) ] ^ S7[ g( z, 0x4 ) ] ^ S8[ g( z, 0x6 ) ] ^ S7[ g( z, 0x0 ) ];
    x[ 1 ] = z[ 0 ] ^ S5[ g( x, 0x0 ) ] ^ S6[ g( x, 0x2 ) ] ^ S7[ g( x, 0x1 ) ] ^ S8[ g( x, 0x3 ) ] ^ S8[ g( z, 0x2 ) ];
    x[ 2 ] = z[ 1 ] ^ S5[ g( x, 0x7 ) ] ^ S6[ g( x, 0x6 ) ] ^ S7[ g( x, 0x5 ) ] ^ S8[ g( x, 0x4 ) ] ^ S5[ g( z, 0x1 ) ];
    x[ 3 ] = z[ 3 ] ^ S5[ g( x, 0xA ) ] ^ S6[ g( x, 0x9 ) ] ^ S7[ g( x, 0xB ) ] ^ S8[ g( x, 0x8 ) ] ^ S6[ g( z, 0x3 ) ];

    K[ 12 + i * 16 ] = S5[ g( x, 0x8 ) ] ^ S6[ g( x, 0x9 ) ] ^ S7[ g( x, 0x7 ) ] ^ S8[ g( x, 0x6 ) ] ^ S5[ g( x, 0x3 ) ];
    K[ 13 + i * 16 ] = S5[ g( x, 0xA ) ] ^ S6[ g( x, 0xB ) ] ^ S7[ g( x, 0x5 ) ] ^ S8[ g( x, 0x4 ) ] ^ S6[ g( x, 0x7 ) ];
    K[ 14 + i * 16 ] = S5[ g( x, 0xC ) ] ^ S6[ g( x, 0xD ) ] ^ S7[ g( x, 0x3 ) ] ^ S8[ g( x, 0x2 ) ] ^ S7[ g( x, 0x8 ) ];
    K[ 15 + i * 16 ] = S5[ g( x, 0xE ) ] ^ S6[ g( x, 0xF ) ] ^ S7[ g( x, 0x1 ) ] ^ S8[ g( x, 0x0 ) ] ^ S8[ g( x, 0xD ) ];
}
 qu_int32 L[ ROUND_COUNT + 1 ] = { 0 };
L[ 0 ] = msg[ 0 ];
 qu_int32 R[ ROUND_COUNT + 1 ] = { 0 };
R[ 0 ] = msg[ 1 ];
 for( int i = 0; i < ROUND_COUNT; ++i ) {
    int rIndex = reverse ? ( ROUND_COUNT - 1 - i ) : i;
    quint32 Kmi = K[ rIndex ];
    quint8  Kri = K[ 16 + rIndex ] & 0x1F;
     quint32 I = 0;
    quint32 f = 0;
     quint8 Ia, Ib, Ic, Id;
     switch( rIndex % 3 ) {
    case 0:
        I = cyclicShift( sumMod2_32( Kmi, R[ i ] ), Kri );
        splitI( I, &Ia, &Ib, &Ic, &Id );
        f = sumMod2_32( subtractMod2_32( S1[ Ia ] ^ S2[ Ib ], S3[ Ic ] ), S4[ Id ] );
        break;
     case 1:
        I = cyclicShift( Kmi ^ R[ i ], Kri );
        splitI( I, &Ia, &Ib, &Ic, &Id );
        f = sumMod2_32( subtractMod2_32( S1[ Ia ], S2[ Ib ] ), S3[ Ic ] ) ^ S4[ Id ];
        break;
```

```
      case 2:
         I = cyclicShift( subtractMod2_32( Kmi, R[ i ] ), Kri );
         splitI( I, &Ia, &Ib, &Ic, &Id );
         f = subtractMod2_32( sumMod2_32( S1[ Ia ], S2[ Ib ] ) ^ S3[ Ic ], S4[ Id ] );
         break;
      }
      L[ i + 1 ] = R[ i ];
      R[ i + 1 ] = L[ i ] ^ f;
   }
   msg[ 0 ] = R[ ROUND_COUNT ];
   msg[ 1 ] = L[ ROUND_COUNT ];
}
// Визначення  для   S-блоків є в додатку
const CAST128::SType CAST128::S1 = { ... };
const CAST128::SType CAST128::S2 = { ... };
const CAST128::SType CAST128::S3 = { ... };
const CAST128::SType CAST128::S4 = { ... };
const CAST128::SType CAST128::S5 = { ... };
const CAST128::SType CAST128::S6 = { ... };
const CAST128::SType CAST128::S7 = { ... };
const CAST128::SType CAST128::S8 = { ... };
```

Робота алгоритму розпочинається з формування 32 ключів розгортки K. Для їх побудови використовується копія x ключа key і тимчасовий ключ z.

Заповнення ключів розгортки проходить в два цикли з використанням S - блоків S5, S6, S7 і S8. При цьому задіяна допоміжна функція g() для витягання i -го байта з ключа key :

```
static const qu_int8 K_MAP[ sizeof( CAST_128::Key ) ] = {
    3,  2,  1,  0,
    7,  6,  5,  4,
   11, 10,  9,  8,
   15, 14, 13, 12
};
static qu_int8 g( const CAST_128::Key key, qu_int8 i ) {
   return ( ( qu_int8* ) key )[ K_MAP[ i ] ];
}
```

Реалізація функції g() використовує карту перетворень K_MAP через порядок зберігання байтів в пам'яті (алгоритм CAST - 128 вимагає порядок від старшого до молодшого, а на архітектурі x86 байти зберігаються від молодшого до старшого). (слід зауважити що порядок може мінятися залежно від платформи

і реалізації компілятора.) Якщо key ={ 0x01234567, 0x12345678, 0x23456789, 0x3456789A }, то g(key, 0) == 0x01 (без K_MAP — 0x67), а g(key, 6) == 0x56 (без K_MAP — 0x34) і так далі

Кодування за допомогою 128-бітного ключа здійснюється в 16 раундів. Для кожного i -го раунду використовуються підключі маскування Km_i та підключі перестановки Kr_i. При шифруванні $Km_i = K[i]$ та $Kr_i = K[16 + i] \& 0x1F$ (5 найменш значущих бітів), а при розшифровці: $Km_i = K[15 - i]$ та $Kr_i = K[31 - i] \& 0x1F$.

Початкове повідомлення розбивається на ліву L і праву R частини. Первинна ініціалізація відбувається наступним таким чином: L[0] = msg[0] і R[0] = msg[1]. Далі над цими половинами проводяться маніпуляції. При цьому в різних раундах використовуються різні функції перетворень :

```
switch( rIndex % 3 ) {
case 0:
   I = cyclicShift( sumMod2_32( Kmi, R[ i ] ), Kri );
   splitI( I, &Ia, &Ib, &Ic, &Id );
   f = sumMod2_32( subtractMod2_32( S1[ Ia ] ^ S2[ Ib ], S3[ Ic ] ), S4[ Id ] );
   break;

case 1:
   I = cyclicShift( Kmi ^ R[ i ], Kri );
   splitI( I, &Ia, &Ib, &Ic, &Id );
   f = sumMod2_32( subtractMod2_32( S1[ Ia ], S2[ Ib ] ), S3[ Ic ] ) ^ S4[ Id ];
   break;

case 2:
   I = cyclicShift( subtractMod2_32( Kmi, R[ i ] ), Kri );
   splitI( I, &Ia, &Ib, &Ic, &Id );
   f = subtractMod2_32( sumMod2_32( S1[ Ia ], S2[ Ib ] ) ^ S3[ Ic ], S4[ Id ] );
   break;
}
```

Тут використовуються допоміжні функції cyclicShift() — циклічний побітовий зсув вліво; sumMod2_32() — сума по модулю 2_32; subtractMod2_32() — віднімання по модулю 2_32 та splitI() — функція для розбиття 32-бітового числа на чотири 8-бітових в порядку від старшого байта Ia до молодшого Id.

Для обчислення f-ції ми використовуємо S –блоки S1, S2, S3 і S4.

Результатом роботи i -го раунду стає: L[i + 1] = R[i] та

R[i + 1] = L[i] ^ f.

Остаточно закодоване повідомлення отримано у вигляді

msg[0] = R[16] та msg[1] = L[16].

3.3. Інтерфейс і робота з програмою.

3.3.1. Налаштування TrueCrypt

Демонструємо програму на прикладі Windows версії. Робота з Linux версією багато в чому схожа. За замовчанням у програму не вбудовані інші мови, крім англійської. Завантажити додаткові мови також можна.

Завантажений файл розпаковують та переміщають у каталог із встановленою програмою, наприклад 'C:Program FilesTrueCrypt'.

Запускаємо TrueCrypt.- Виберіть Settings -> Language, там оберете мову та натисніть OK.

Особливих ускладнень з установкою TrueCrypt на Widnows не було, тому одразу перейдемо до використання. Ось так (Рисунок 3.1.) виглядає початкове вікно програми

Рисунок 3.1.Початкове вікно роботи з програмою.

Для створення зашифрованого контейнера Вибираємо "Volume" -> "Create Volume". див рисунок 3.2.

Можна вибрати тип Тома з наступних варіантів:

Звичайний файл-контейнер. У цьому випадку буде створено том у вигляді окремого файлу, який можна буде копіювати, перейменувати або видалити.

• Несистемний диск (Розділ жорсткого диска, флешка тощо). Шифрується весь диск повністю.

• Системний диск. У цьому випадку буде зашифровано диск, на якому інстальовано Windows.

При тому що наявні широкі можливості щодо шифрування томів та системи в цілому, створимо простий файл-контейнер, для цього залишаємо перемикач радіо в положенні "Create an encrypted file container" і натискаємо "Next" «Створити зашифрований файловий контейнер

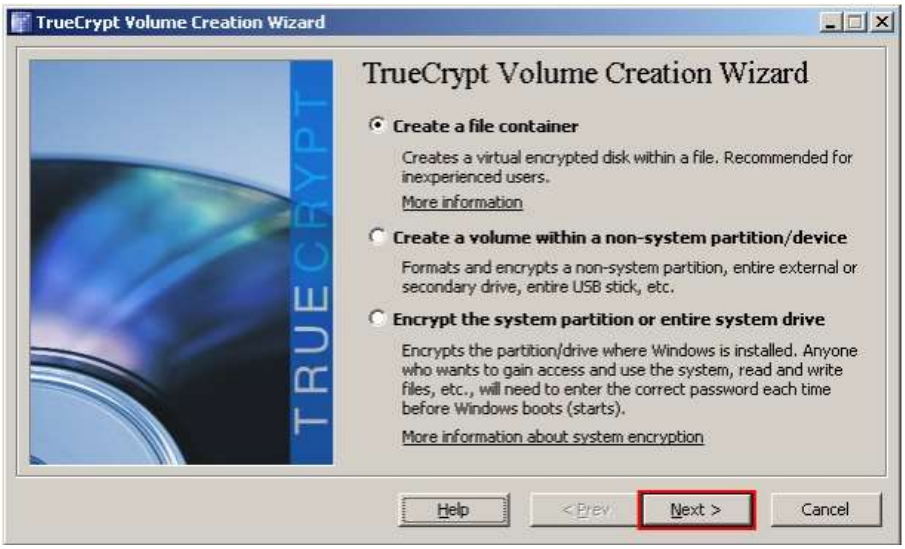

Рисунок 3.2. Вікно створення контейнера.

На наступному етапі нам пропонується створити звичайний чи прихований том. Звичайний том – коли під час введення пароля стають доступні зашифровані у ньому файли. Прихований том включає два рівні і два паролі – справжній і скажімо так, додатковий. При введенні додаткового пароля відображаються зашифровані файли, які не становлять особливої важливості. При введенні цього пароля з'являється можливість працювати з дійсно цінними даними. Зашифровані дані структурою не відрізняються від набору випадкових байтів. Зараз виберемо "Звичайний том TrueCrypt". Надалі вибирається розміщення тому – тут все більш

ніж зрозуміло. Можна вибрати будь-яке розташування, будь-яку назву та будь-яке розширення файлу.

Наступний етап - вибір алгоритму шифрування.Рисунок 3.3.

Рисунок 3.3. Вікно для вибору алгоритму шифрування.

На вибір представлені кілька алгоритмів шифрування та хешування. Тут можна вибрати як вбудовані алгоритми так і наш власний. Причому можна поєднувати одразу два і навіть три види шифрування. Проте використання одночасно двох або трьох рівнів шифрування може зашкодити швидкості запису на зашифрований диск.

Далі вводимо пароль.(Рисунок 3.4.) Тут треба розуміти, що асиметричні алгоритми шифрування (саме ті, з якими працює TrueCrypt), можна зламати тільки методом підбору – і більше ніяк! Тому якщо ви обрати пароль з 1-5

символів (адже так легко запам'ятати) або з осмисленого слова (теж легко запам'ятати), то такий пароль може бути підібраний шляхом брутфорсингу, тобто. перебирання різних комбінацій.

З іншого боку, якщо придумати довгий складний пароль з використанням різних символів та регістрів, при спробі доступу до ваших даних знову ж таки можуть скористатися брутфорсингом, але підбирати його вони будуть дуже довго. Але тут, знову ж таки, важливо не перестаратися. Якщо ви забудете свій пароль, дані на зашифрованому контейнері можна вважати безповоротно втраченими.

Рисунок 3.4. Діалогове вікно вводу пароля

Наступний крок. Вибір файлової системи. Рисунок 3.5.

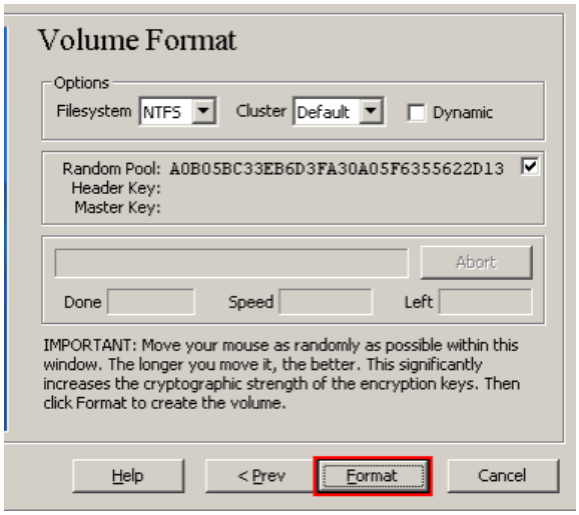

Рисунок 3.5. Діалогове вікно вибору файлової системи.

Якщо треба зберігати на зашифрований диск файли розміром більше 4 гігабайт, то обирають NTFS, в інших випадках - не важливо, яку файлову систему вибирати. Натискаємо "Розмітити".

Далі переходимо у головне вікно програми. Вибираємо букву диска, який буде присвоєно нашому віртуальному диску (абсолютно неважливо, яку букву ви оберете). Клацаємо кнопку з написом «Файл». Відкривається провідник Windows, у ньому вибираєте файл із контейнером. Натискаємо кнопку "Змонтувати", вводимо пароль.

3.3.2. Робота з програмою

Тепер будь-яким файловим менеджером можна перейти у новий диск. Відкриємо Мій Комп'ютер і побачимо, що наш підключений диск знаходиться поряд з іншими дисками системи. Робота з ним нічим не відрізняється від роботи, наприклад, зі звичайною флешкою – можна копіювати, видаляти, створювати папки, відкривати та ін.

Для відкриття фалатому є два шляхи:

• Якщо ви при установці TrueCrypt залишили параметри за замовчуванням, значить розширення ".tc" було зіставлено з програмою TrueCrypt і при подвійному натисканні миші на файл-сховище у нас відкриється головне вікно TrueCrypt з вибраним файлом.

• Другий варіант – це у головному вікні програми натиснути кнопку "Select file" і вибрати файл звідси.

Після того, як файл вибраний, вибираємо букву, на яку монтуватиметься в системі том і натискаємо "Mount".

Вводимо пароль або ключову фразу, що ви придумали при створенні тома.Якщо при введенні пароля ви не помилилися - ви побачите біля обраної літери інформацію, що відноситься до нашого тома.

Увага. Треба не забути затерти вільний простір, щоб припинити можливість відновлення прихованої інформації. Також не забувайте відмонтувати томи, які ви вже не використовуєте.

Щоб припинити доступ до диску, треба натиснути кнопку «Розмонтувати» у головному вікні програми.(Рисунок3.6.). Також програма містить багато налаштувань

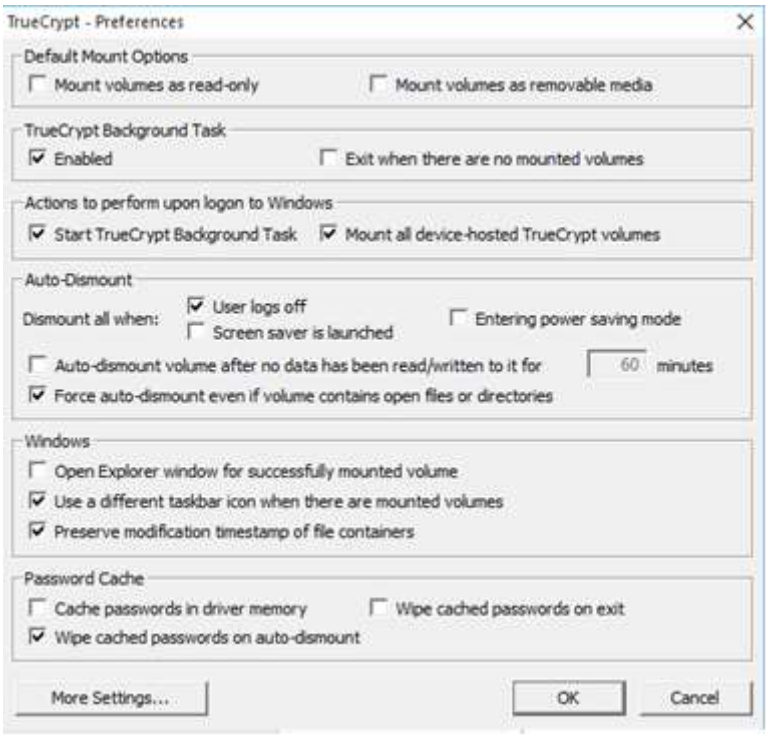

Рисунок 3.6. Вікно додаткових налаштувань.

Тут цікавими є дві з них. Перша називається "Автоматично розмонтувати томи при неактивності більше…". Тут все зрозуміло, якщо ви працювали із зашифрованим диском, а потім відійшли від комп'ютера на час більше встановленого, забувши цей диск розмонтувати, то програма це зробить сама після цього інтервалу часу – дуже корисно. І друга – налаштування гарячих клавіш.

Є ще один корисний спеціальний інструмент "Налаштування переносного диска TrueCrypt". Цей інструмент дозволяє не тільки зашифрувати переносний диск, а й копіює на нього копію TrueCrypt, тобто можна змонтувати флешку на будь-якому комп'ютері, відповідно це дозволяє завжди мати при собі важливі файли з можливістю доступу без встановлення додаткових програм.

Висновки до третього розділу

В третьому розділі було обгрунтовано вибір мови для програмної реалізації алгоритму шифрування.. Також наведено принципово важливі фрагменти програмного коду з поясненнями щодо їх функціональності. Надалі було продемонстровано яким чином даний алгоритм можна буде запустити на виконання, застосовуючи його інтеграцію в програмний комплекс з відкритим кодом. Докладно описано можливості які надає програма ля забезпечення обмеження доступу до файлів що становлять комерційну таємницю і мають бути захищені від зовнішніх втручань.

Висновок

Захист інформації – це комплекс заходів, спрямованих на забезпечення інформаційної безпеки. Таким чином, правильний з методологічного погляду підхід до проблем інформаційної безпеки починається з виявлення суб'єктів інформаційних відносин та інтересів цих суб'єктів, пов'язаних з використанням інформаційних систем (ІС). Загрози інформаційної безпеки – це зворотний бік використання інформаційних технологій.

Криптографічний захист інформації є надійним та недорогим засобом. Будь-який обсяг інформації від кількох байт до гігабайта, будучи зашифрований за допомогою більш менш стійкої криптосистеми, недоступний для прочитання без знання ключа. І вже зовсім не важливо, зберігається він на жорсткому диску, дискеті або на компакт-диску, неважливо під керуванням якоїсь операційної системи.

В дипломній роботі було розглянуто методи захисту інформації від зовнішніх втручань. Запропоновано застосування програмної системи з відкритим програмним кодом що виконувала шифрування даних. В бібліотечний модуль даної програми додано власний алгоритм шифрування інформації. В результаті комбінування вже наявних алгоритмів і власного дозволить забезпечити більш надійний захист інформації від потенційних зазіхань.

СПИСОК ЛІТЕРАТУРИ

1. Нормативно-правове забезпечення інформаційної безпеки: Збірник нормативно-правових документів / Уклад. О.Г. Корченко, Ю.О. Дрейс. — Житомир : ЖВІ НАУ, 2018. — 280 с.
2. Ліпкан В.А. Інформаційна безпека України в умовах євроінтеграції : навчальний посібник / Ліпкан В.А. Максименко Ю.Є., Желіховський В.М. – К. : КНТ, 2016. – 280 с.
3. Технології захисту інформації: навчальний посібник / С. Е. Остапов, С. П. Євсеєв, О. Г. Король. – Х.: Вид. ХНЕУ, 2016. – 476 с. (Укр. мов.)
4. Сенів М. М. Безпека програм та даних: навч. посібник / М.М. Сенів, В.С. Яковина. –Львів : Видавництво Львівської політехніки, 2015. –256 с.
5. Юдін О.К. Інформаційна безпека. Нормативно-правове забезпечення / О.К. Юдін // Підручник. — К. : НАУ, 2016. — 620 с.
6. Юдін О.К. Захист інформації в мережах передачі даних / О.К. Юдін, О.Г. Корченко, Г.Ф. Конахович // Підручник — К. : Вид-во DIRECTLINE, 2019. — 714 с.
7. Горбенко І.Д. Гриненко Т.О. Захист інформації в інформаційно-телекомунікаційних системах: Навч. посібник. Ч.1. Криптографічний захист інформації -Харків: ХНУРЕ, 2004 -368 с.
8. Лагун А. Е. Криптографічні системи та протоколи: нав. посібник / А. Е. Лагун. –Львів : Видавництво Львівської політехніки, 2013. –96 с
9. Задірака В.К., Олексюк О.С. Комп’ютерна криптологія: Підручник.-Київ:2019.-504с.
10. Горбенко І.Д., Горбенко Ю.И., «Прикладна криптологія. Теорія. Практика. Застосування». НТЛ., Львів, 2018, с.134
11. О.В.Вербіцький. Вступ до криптології. Видавництво НТЛ., Львів, 2018, с.248
12. А.О.Антонюк. Основи захисту інформації в автоматизованих системах. – К.: КМ Академія, 2020. – 244 с.

13. Risk Management Framework for Information Systems and Organizations A System Life Cycle Approach for Security and Privacy, 2018.
14. Методи захисту системи управління інформаційною безпекою. Вимоги (ISO/IEC 27001:2013; Cor 1:2014, IDT). ДСТУ ISO/IEC 27001:2015. 3. NIST Special Publication 800-30. Risk Management Guide for Information Technology Systems.
15. Потій О.В., Лєншин А.В. Дослідження методів оцінки ризиків у безпеці інформації та розробка пропозицій з їх вдосконалення на основі системного підходу // Зб. наук. праць Харків. ун-ту Повітряних Сил. 2010. Вип. 2(24). С. 85-91.
16. Аналіз методів оцінки ризиків інформаційної безпеки [Електроний ресурс]. Режим доступу: https://www.securitylab.ru/blog/personal/secinsight/19205.php.
17. 16. Табунщик, Г.В. Інженерія якості програмного забезпечення:навчальний посібник / Г.В. Табунщик, Р.К. Кудерметов, Т.І. Брагіна. – Запоріжжя: ЗНТУ, 2013. – 176с.
18. CRAMM user guide, Risk Analysis and Management Method, United Kingdom Central Computer and Telecommunication Agency (CCTA), UK, 2021.
19. Herbert Schildt. C++: The Complete Reference, 4th Edition
20. Microsoft WinAPI documentation — https://docs.microsoft.com/en-us/windows/win32/api/
21. Mishra J. Software Engineering. [Текст] / J. Mishra, A. Mohanty. – Dorling Kindersley (India), 2018. – 373 p.
22. ISO [Електронний ресурс] / iso.org. — 2021. — Режим доступу: https://www.iso.org/standard/54531.html

ДОДАТОК А Частини програмного коду

ДОДАТОК Б Презентація