МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. В. ДАЛЯ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК ТА ІНЖЕНЕРІЇ

До захисту допускається
Т.в.о. завідувача кафедри
_____ Сафонова С.О.
«____»_____ 20__ р.

МАГІСТЕРСЬКА РОБОТА

НА ТЕМУ:

Метод кодування інформації в веб-додатках

Освітній рівень"Магістр"
Спеціальність 123 "Комп'ютерна інженерія"

Науковий керівник роботи:	_____ (підпис)	В.М.Барбарук (ініціали, прізвище)
Консультант з охорони праці:	_____ (підпис)	Я.О.Критська (ініціали, прізвище)
Студент:	_____ (підпис)	П.В.Петров (ініціали, прізвище)
Група:		КІ-18зм

Сєвєродонецьк 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет Інформаційних технологій та електроніки
Кафедра Комп'ютерних наук та інженерії
Освітній рівень магістр
Напрям підготовки _____
(шифр і назва)
Спеціальність 123 "Комп'ютерна інженерія"
(шифр і назва)

ЗАТВЕРДЖУЮ:

Т.в.о. завідувача кафедри _____
_____ С.О. Сафонова
«_____» _____20____р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Петрову Павлу Валентиновичу
(прізвище, ім'я, по батькові)

1. Тема роботи Метод кодування інформації в веб-додатках

керівник проекту (роботи) Барбарук Віктор Миколайович, к.т.н., доц.
(прізвище, м.'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від «11» 10 2019 р. № 136/15.15
2. Строк подання студентом роботи 10.01.2020
3. Вихідні дані до роботи Матеріали науково-дослідної практики, методи шифрування, файли для шифрування, середа розробки веб-застосунку для шифрування данних

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Аналіз та історія предметної області та постановка задачі, сучасні платформи для побудови web-застосунків, математична реалізація поставленої задачі, програмна реалізція веб-застосунку, охорона праці та безпека в надзвичайних ситуаціях, висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Електронні плакати

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці та безпека в надзвичайних ситуаціях	Критська Я.О. ст. викл. кафедри КНІ		

7. Дата видачі завдання 14.10.2019

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Отримання завдання до магістерської роботи	02.09.2019-15.09.2019	
2	Аналіз технічних засобів	16.09.2019-22.09.2019	
3	Розробка методу шифрування	23.09.2019-25.09.2019	
4	Програмна реалізація методу шифрування	26.09.2019-06.10.2019	
5	Розробка частини проекту "Охорона праці та безпеки в надзвичайних ситуаціях"	07.10.2019-25.11.2019	
6	Оформлення пояснювальної записки, автореферату та презентації	26.11.2019-9.01.2020	
7			

Студент _____ П.В.Петров
(підпис) (прізвище та ініціали)

Науковий керівник _____ В.М.Барбарук
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Петров П.В. Метод кодування інформації в веб-додатках.

Метою роботи є аналіз методів шифрування та розробка застосунку, який надає змогу шифрувати дані.

Об’єктом дослідження є методи шифрування даних.

Веб-застосунок був розроблений таким чином, щоб збільшити ефективність шифрування даних для користувачів за допомогою простого інтерфейсу та декілька можливих варіантів шифрування

У атестаційній роботі проводиться аналіз технологій для побудови сучасних веб-застосунків. Ґрунтуючись на отриманих даних створюється комплекс програмних рішень націлених на забезпечення ефективності застосування веб-застосунку.У результаті роботи здійснена програмна реалізація системи для шифрування даних.

Ключові слова: веб-застосунок, база даних, система керування базою даних, мова-програмування c#, javascript, jquery, spring , mvc, servlet , jsp, веб-сервер, asp .net core.

ABSTRACT

Petrov P.V. The method of coding information in the web add-ons.

The goal is to develop methods of data encryption methods and web applications development that allows you to encrypt data.

The object of the study is the data encryption methods.

Web application was developed to increase the efficiency for informing users with a simple interface and and several possible encryption options. In the masters work analyzes the technology to build modern web applications. Based on the received data created complex software solutions aimed at ensuring the effectiveness of exercise-class web application.

As a result of implemented software implementation of the system to encrypt data.

Keywords: web application, database management, database, programming language c#, javascript, jquery, spring/mvc, sql- injection, the web server, asp .net core.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

СУБД – Система Управління Базами Даних

MVC – Model View Controller

JSP – Java Server Page

JS – JavaScript

SQL – Structured Query Language

URI – Uniform Resource Identifier

API – Application Programming Interface

ASP – Active Server Page

CGI – Common Gateway Interface

FTP – File Transfer Protocol

HTML – Hyper Text Markup Language

HTTP – Hyper Text Transfer Protocol

JVM – Java Virtual Machine

DI – Dependency Injection

IOC – Inversion of Control

ВСТУП

У сучасний час широта охоплення глобальної мережі Інтернет має фактично планетарний масштаб. Практично для будь-якої компанії необхідність мати свій власний веб-ресурс є обов'язковим. Під час розвитку Інтернет удосконалювалися й розроблялися нові веб-технології , також у цей час ведення бізнесу, здійснення комерційних взаємодій, надання різних інформаційних послуг у глобальній мережі не є новим поняттям або явищем. В життя сучасної цивілізації з'явилась велика кількість нових можливостей, але обовов'язковою умовою є безпека особистих даних користувачів, чим багато хто нехтує.

Криптографія – невід'ємна деталь будь-якої діяльності, пов'язаної з розмежуванням прав доступу до інформації. Державні органи та зовнішньополітичні відомства, військово-промисловий комплекс, приватний бізнес (в тому числі інтернет-сервіси) – ось невелика частина списку споживачів, зацікавлених у збереженні своєї інформації в таємниці від третіх осіб. Розроблено безліч методів захисту інформації, серед яких: кодування, криптографія, стеганографія. Один з найпростіших і ефективних, криптографічний, спирається в основному на обчислювальні можливості сучасних комп'ютерів. Існуючий рівень забезпечення безпеки можна проаналізувати, якщо звернути увагу на загальнодоступну інформацію. Мова йде про експлуатації уразливості Heartbleed таких популярних інтернет-сервісів як Wikipedia, GitHub, IFTTT, а також ОС Android, деяких дистрибутивів Linux, роутерів Cisco systems та інших. У ході експлуатації даної уразливості зловмисники отримали особисті дані користувачів цих продуктів: імена, номери паспортів і банківських карт, тощо. Очевидно, що для приватної інформації подібні витоки неприпустимі, що змушує військові відомства та інші подібні організації шукати свої шляхи вирішення проблеми інформаційної безпеки. Їх практика показує, що використання сертифікованого побутового апаратного та програмного забезпечення (наприклад, материнських плат) зводить систему безпеки нанівець: обладнання та програми мають закриту архітектуру, а копіювання цінної інформації на незахищений носій не представляється складним завданням для користувача сертифікованого, але побутового обладнання і програмного забезпечення. Відомі уразливості можуть таємно експлуатуватися впродовж тривалого часу, якщо продукт має закриту архітектуру. Таким же чином з ужитку вийшли моноалфавітні шифри, такі як шифр Цезаря: закритість конкретної деталі алгоритму не означає можливості зворотного інжинірингу зловмисником. Тож можна зробити висновок про потреби ринку в ряді

вузькоспеціалізованих продуктів для забезпечення інформаційної безпеки з відкритою архітектурою.

Метою даної атестаційної роботи є аналіз побудови сучасних веб-застосунків, а так само створення веб-застосунку для шифрування даних за допомогою мови C#, ASP.NET MVC та інших сучасних технологій для розробки веб-застосунків.

1 АНАЛІЗ ТА ІСТОРІЯ ПРЕДМЕТНОЇ ОБЛАСТІ, ПОСТАНОВКА ЗАДАЧІ

1.1 Загальний аналіз предметної області

Криптографія - тайнопис, спеціальна система зміни звичайного листа, яка використовується з метою зробити текст зрозумілим лише для обмеженого числа осіб, які знають цю систему.

Спочатку криптографія вивчала методи шифрування інформації - оборотного перетворення відкритого (вихідного) тексту на основі секретного алгоритму або ключа в шифрований текст (шифротекст).

Це одна з найстаріших наук, її історія налічує кілька тисяч років.

Проблема захисту інформації шляхом її перетворення, що виключає її прочитання сторонньою особою, хвилювала людський розум з давніх часів. Історія криптографії - ровесниця історії людської мови. Більше того, спочатку писемність сама по собі була криптографічного системою, тому що в древніх суспільствах нею оволоділи тільки обрані. Священні книги Стародавнього Єгипту, Стародавньої Індії тому приклади.

В якості основного критерію періодизації криптографії можливо використовувати технологічні характеристики використовуваних методів шифрування.

Перший період (приблизно з 3-го тисячоліття до н. е..) характеризується пануванням моноалфавитной шифрів (основний принцип - заміна алфавіту вихідного тексту іншим алфавітом через заміну букв іншими буквами або символами). Другий період (хронологічні рамки - з IX століття на Близькому Сході (Ал-Кінді) і з XV століття в Європі (Леон Баттіста Альберті) - до початку XX століття) ознаменувався введенням в ужиток поліалфавітних шифрів. Третій період (з початку і до середини XX століття) характеризується впровадженням електромеханічних пристроїв в роботу шифрувальників. При цьому продовжувалося використання поліалфавітних шифрів.

Четвертий період - з середини до 70-х років XX століття - період переходу до математичної криптографії. У роботі Шеннона з'являються суворі математичні визначення кількості інформації, передачі даних, ентропії, функцій шифрування. Обов'язковим етапом створення шифру вважається вивчення його вразливості до різних відомим атакам - лінійному і диференціальному криптоанализу. Однак до 1975 року криптографія залишалася «класичною», або ж, більш коректно, криптографією з секретним ключем.

Сучасний період розвитку криптографії (з кінця 1970-х років по теперішній час) відрізняється зародженням та розвитком нового напрямку - криптографія з відкритим ключем. Її поява знаменується не тільки новими технічними можливостями, а й порівняно широким поширенням криптографії для використання приватними особами.

Сучасна криптографія утворює окремий науковий напрям на стику математики та інформатики - роботи в цій області публікуються в наукових журналах, організовуються регулярні конференції. Практичне застосування криптографії стало невід'ємною частиною життя сучасного суспільства - її використовують в таких галузях як електронна комерція, електронний документообіг (включаючи цифрові підписи), телекомунікації та інших.

1.2 Методи шифрування інформації в Стародавньому світі

Більшість сучасних дослідників пов'язують появу криптографії з появою писемності, вказуючи, що ці процеси відбулися майже одночасно.

Методи секретної переписки були винайдені незалежно в різних державах стародавнього Сходу, таких як Єгипет, Китай і Шумер, хоча сьогодні дуже важко судити про рівень розвитку криптології в цих суспільствах. Клинопис, рисуночное і ієрогліфічне письмо саме по собі було вкрай складно і вимагало тривалого навчання, так що питання про шифрування повідомлень часто просто не піднімалося, так як коло грамотних осіб був досить обмежений.

Всю складність даного питання ілюструє один приклад: знайдено безліч глиняних табличок з клинописними знаками, записаними в кілька шарів (первісна запис замазувати глиною і поверх неї наносилася нова).

Однак з розвитком фонетичного листа і значним спрощенням писемності, криптология отримує значний стимул до розвитку. Розвитку цієї галузі знань сприяли і розвиток торгівлі, військової справи і дипломатичної діяльності, які створювали необхідний попит на «продукцію» криптографов.

Найбільший розвиток в цей час криптографія отримує в полісах Стародавньої Греції, а пізніше в Римі. Основні криптографічні системи, багато з яких використовуються аж до наших днів були розроблені в Стародавній Греції і отримали широке практичне застосування в Римі. У Стародавній Греції використовувалися як шифри заміни, так і шифри перестановки. Так найбільш поширеним і отримав широку популярність в античному світі шифром заміни є т.зв. шифр Цезаря, описаний Светонием. Для того щоб

зашифрувати повідомлення, кожну його букву замінювали на іншу букву латинського алфавіту, але зі зрушенням вліво або вправо. Цезар у своїх посланнях до сенату заміняв всі букви на три віддалені зліва, Август застосовував той же шифр, але зі зрушенням в чотири знаки.

Найбільших успіхів у криптографії в античний період домоглася Спарта, де активно використовувалися всі відомі види шифрів і були створені перші дійшли до нас шифрувальні пристрій. Першим таким приладом, які реалізують шифр перестановки була т.зв. «Сціталла» (ОКОК VI-V ст. До н.е.). На циліндр певного діаметру по спіралі намотувався ремінь, на який наносили літери уздовж осі циліндра. У результаті в розгорнутому вигляді всі букви змішувалися, а якщо намотати ремінь на циліндр того ж діаметру, то повідомлення знову ставало зрозумілим.

Водночас стійкість даного шифру була невелика, а пізніше Архімед запропонував пристрій (т.зв. антісціталла), за допомогою якого розшифровка подібного повідомлення без потрібного циліндра була вельми простою і швидкою. Ремінь намотували на конічне «спис» і зрушували вгору і вниз до тих пір, поки не знаходили потрібний діаметр і текст повідомлення ставав зрозумілим.

1.3 Методи шифрування інформації в пізніше Середньовіччя і епоху Відродження

В епоху пізнього середньовіччя, з початком відродження античної спадщини і освіти, криптографія в Європі знаходить «друге народження», насамперед у середовищі інтелектуальної еліти того часу. Багато вчених середньовічного періоду прагнули приховати зроблені ними винаходи і відкриття. Так сучасні дослідники встановили, що склад чорного пороху було відкрито відомим англійським ученим середини XIII століття - Роджером Беконом (найвідоміше його винахід - окуляри), майже за сто років до «офіційної» дати створення пороху Бертольдом Шварцем. В одному з його праць були присутні незашифроване опис властивостей цієї речовини, але сам склад був зашифрований таким складним шифром перестановки, що розкрити його вдалося лише в наші дні з застосуванням ЕОМ.

Однак проблема використання середньовічними вченими криптографії у своїх творах, особливо шифрів перестановки, вельми неоднозначна.

І все ж розвиток криптології в пізніше Середньовіччя і ранній Новий час було прямо пов'язане з розквітом дипломатії. Так в 1401 році в герцогстві Мантуя був створений перший, що дійшов до нас шифр багатозначною заміни, причому по кілька позначень мали лише голосні літери, що може свідчити про знайомство укладача шифру з методами криптоаналізу, заснованими на частоті зустрічаються в тексті голосних букв.

Невідомо, чи була тісний зв'язок між розвитком європейської та східної криптографії. Крім того, якщо на сході криптографія була радше частиною лінгвістики, то в Європі вона була ближче до математики і природничих наук, що також визначило її специфіку.

У багатьох європейських державах, починаючи з XVI століття, з'являється посаду «секретаря за шифрами», єдиним заняттям якого, було створення шифрів для «своїх» дипломатичних служб і розшифровка «чужих» повідомлень. Уже в XV столітті закладаються теоретичні основи європейської криптології.

Знаменитий італійський архітектор Леон Батіста Альберті може бути названий «батьком» європейської криптології. Саме він у своїй праці «Трактат про шифри» вперше запропонував шифр многоалфавитной заміни, який робив повідомлення практично невскривающійся. Цей тип шифру часто називають таблицею Вінджера - англійського дипломата XVI століття, активно застосовував його на практиці.

Леон Альберті може вважатися видатним криптографом й тому, що створив перший в європейській історії наукова праця з криптології - «Трактат про шифри» 1466, в якому не тільки наводилися приклади можливих варіантів шифрування, але й обгрунтовувалася доцільність застосування криптографії на практиці, як найбільш дешевого і надійного інструменту захисту інформації.

Спеціальних навчальних закладів, де навчали б криптографічного діяльності, в той час не існувало. Криптологів рекрутували з найбільш освічених людей того часу, які знають математику та іноземні мови.

Взагалі ж криптографію в пізніше Середньовіччя і ранній Новий час використовували не тільки державні діячі, а й багато освічених людей. Так Леонардо да Вінчі шифрував свої роботи за допомогою дзеркала, записуючи слова задом наперед. Він використовував і інші, значно складніші шифри, деякі з яких до цих пір не розкриті.

З розвитком природничих наук до криптографічного діяльності все частіше залучаються талановиті та здібні математики. Одним з перших таких криптографов був Франсуа Вієт - основоположник практично всієї сучасної алгебри і видатний учений свого часу.

Мабуть самим невдахою з таких криптографов був Лейбніц - видатний німецький учений, математик, засновник Берлінської академії наук. Англійський король Георг I хотів

запросити Лейбніца, щоб той очолив британську криптографічну службу, але натрапив на різку протидію в особі Валліса, яка побоюється конкуренції з боку свого німецького колеги і який пригрозив перейти на бік Іспанії, видавши їй всі англійські секрети, яких Валліс, за характером своєї діяльності знав чимало. Активно виступав проти подібного призначення і Ньютон - голова Королівського наукового товариства, котрий оскаржував авторство Лейбніца в диференціальному численні. Не пощастило Лейбніцу і вдруге, коли його запросив до Росії Петро I, для не тільки для організації Російської академії наук, а й для створення російської криптографічного служби за європейським зразком, проте смерть Лейбніца не дозволила здійснитися планам Петра, вимушеного скористатися послугами менш іменитих криптологів.

1.4 Методи шифрування інформації Другої світової війни та сучасні методи шифрування

Перед початком Другої світової війни провідні світові держави мали електромеханічні шифрувальні пристрої, результат роботи яких вважався невскривающійся. Ці пристрої ділилися на два типи - роторні машини і машини на цевочной дисках. До першого типу відносять «Енігму», що використовувалася сухопутними військами Німеччини та її союзників, другого типу - американська М - 209.

В СРСР проводилися обидва типи машин.

Німеччина: «Енігма», «Fish»

Історія найвідомішої електричної роторної шифрувальної машини - «Енігма» - починається в 1917 році - з патенту, отриманого голландцем Хьюго Кохом. Наступного року патент був перекуплений Артуром Шербіусом (англ.), що почали комерційну діяльність з продажу примірників машини як приватним особам, так і німецьким армії і флоту.

Німецькі військові продовжують удосконалювати «Енігму». Без урахування настройки положення кілець (нім. Ringstellung), кількість різних ключів становило 1016. Наприкінці 1920-х - початку 1930 років, незважаючи на передані німецьким аристократом Хансом Тіло-Шмідтом дані по машині, що були екземпляри комерційних варіантів, британська і французька розвідка не стали братися за завдання криптоаналізу. Ймовірно, до того часу вони вже визнали, що шифр є невзламиваемим. Проте група з трьох польських математиків так не вважала, і, аж до 1939 року, вела роботи по «боротьбі» з

«Енігмою», і навіть вміла читати багато повідомлення, зашифрованими «Енігмою» (у варіанті до внесення змін до протоколу шифрування від грудня 1938). У одного з них, Маріана Реевского зародилася ідея боротися з криптографічного машиною за допомогою іншої машини. Ідея осяяла Реевского в кафе, і він дав машині ім'я «Бомба» за назвою круглого тістечка. Серед результатів, переданих британським розвідникам перед захопленням Польщі Німеччиною, були і «живі» екземпляри «Енігми», і електромеханічна машина «Bomba», що складалася з шести спарених «Енігма» і допомагала в розшифровці (прототип для пізнішої «Bombe» Алана Тьюринга) , а також унікальні методики криптоаналізу.

Подальша робота по злому була організована в Блетчлі-парку, сьогодні є одним з предметів національної гордості.

З сучасної точки зору шифр «Енігми» був не дуже надійним, але тільки поєднання цього фактора з наявністю безлічі перехоплених повідомлень, кодових книг, донесень розвідки, результатів зусиль військових і навіть терористичних атак дозволило «розкрити» шифр .

Однак з 1940 року вища німецьке командування почало використовувати новий метод шифрування, названий британцями «Fish». Для шифрування використовувалося новий пристрій «Lorenz SZ 40», розроблене на замовлення військових. Шифрування грунтувалося на принципі одноразового блокнота (шифр Вернама, одна з модифікацій шифру Віженер, описана в 1917 році) і при правильному використанні гарантувало абсолютну крипостійкість (що було доведено пізніше в роботах Шеннона). Проте для роботи шифру був потрібний «надійний» генератор випадкової послідовності, який би синхронізуватися на передавальної і приймаючій стороні. Якщо криптоаналитик зуміє передбачити наступне число, що видається генератором, він зможе розшифрувати текст.

На жаль для Німеччини, генератор, використовуваний в машинах «Lorenz SZ 40» виявився «слабким». Однак його злом все одно не можна було здійснити вручну - криптоаналітиків з Блетчли-парку потрібно було створити пристрій, який би перебирав всі можливі варіанти і рятувало б криптоаналітиків від ручного перебору. Таким пристроєм стала одна з перших програмованих обчислювальних машин «Colossus», створена Максом Ньюменом (англ. Max Newman) і Томмі Флауерс (англ. Tommy Flowers) за участю Алана Тьюринга в 1943 році (хоча деякі джерела вказують, що вона була зроблена для злому «Енігми»). Машина включала 1600 електронних ламп і дозволила скоротити час, необхідний на злом повідомлень, з шести тижнів до декількох годин.

З кінця 1990 років починається процес відкритого формування державних стандартів на криптографічні протоколи. Мабуть, найвідомішим є розпочатий в 1997 році конкурс AES, в результаті якого у 2000 році державним стандартом США для

криптографії з секретним ключем був прийнятий шифр Rijndael, зараз вже більш відомий як AES. Аналогічні ініціативи носять назви NESSIE (англ. New European Schemes for Signatures, Integrity, and Encryptions) в Європі та CRYPTREC (англ. Cryptography Research and Evaluation Committees) в Японії.

У самих алгоритмах в якості операцій, покликаних утруднити лінійний і диференціальний криптоаналіз крім випадкових функцій (наприклад, S-блоків, що використовуються в шифри DES і ГОСТ) стали використовувати більш складні математичні конструкції, такі як обчислення в поле Галуа в шифрі AES. Принципи вибору алгоритмів (криптографічних примітивів) поступово ускладнюються. Пред'являються нові вимоги, часто не має прямого відношення до математики, такі як стійкість до атак по сторонніх каналах. Для вирішення завдання захисту інформації пропонуються все нові механізми, в тому числі організаційні та законодавчі.

Також розвиваються принципово нові напрямки. На стику квантової фізики і математики розвиваються квантові обчислення і квантова криптографія. Хоча квантові комп'ютери лише справа майбутнього, вже зараз запропоновані алгоритми для злому існуючих «надійних» систем (наприклад, алгоритм Шора). З іншого боку, використовуючи квантові ефекти, можливо побудувати і принципово нові способи надійної передачі інформації. Активні дослідження в цій області йдуть з кінця 1980-х років.

У сучасному світі криптографія знаходить безліч різних застосувань. Крім очевидних - власне, для передачі інформації, вона використовується в стільникового зв'язку, платному цифровому телебаченні при підключенні до Wi-Fi і на транспорті для захисту квитків від підробок, і в банківських операціях, і навіть для захисту електронної пошти від спаму.

1.5 Аналіз існуючих систем шифрування

В сучасній криптографії практичне значення мають лише методи захисту з використанням ключа. Їх поділяють на два види: симетричні (інша назва - алгоритми з секретним ключем) та асиметричні (алгоритми з відкритим ключем). Симетричні системи шифрування базуються на одному ключі, що використовується і для шифрування, і для дешифрування (або ключ дешифрування можливо обчислити за ключем шифрування).

Їх перевагами є:

− велика пропускна здатність;

− відносно короткі ключі;

− їх можна використати як основу для створення різних криптографічних механізмів (псевдовипадкові генератори чисел, хешфункції, обчислювально-ефективні схеми підпису та ін.);

− можливість їх комбінування для підвищення криптостійкості.

Недоліки симетричних систем:

− складність збереження конфіденційності ключа;

− велика кількість ключів, що використовуються, у великій мережі;

− необхідність частої зміни ключів.

Асиметричні шифри використовують два ключі. Перший – відкритий – застосовується для шифрування інформації і може знаходитися у відкритому доступі. В той час як закритий ключ, що використовується для дешифрування, зберігається в таємниці. Причому ключ для дешифрування неможливо обчислити за ключем шифрування.

Переваги асиметричних алгоритмів:

− відсутня необхідність передачі єдиного секретного ключа усім; користувачам системи;

− в асиметричній криптосистемі тільки один секретний ключ;

− можливість не змінювати ключі значний час;

− у великих мережах менша кількість необхідних ключів.

Їх недоліки:

− складність корегування алгоритмів;

− більш довгі ключі для забезпечення тієї ж криптостійкості;

− вимагають значно більшої обчислювальної потужності.

При практичному застосуванні ці два підходи часто поєднуються. Це надає змогу збалансувати переваги і недоліки обох методів. З появою Інтернету й значною інформатизацією нашого суспільства використання криптографії перейшло на новий рівень і перестало бути прерогативою великих корпорацій і державних служб. Криптографічні методи стали широко використовуватися приватними особами в електронних комерційних операціях, телекомунікаціях та багатьох інших середовищах.

1.6 Постановка задачі дослідження

Необхідно створити веб-застосунок для шифрування данних, який буде надавати змогу користувачу шифрувати бажані дані. Задача включає в себе розробку реалізації функціональності інтерфейсу, розробку дизайну сервісу, проектування та реалізацію бази даних для майбутнього функціонала статистики, яка буде зберігати інформацію про запити користувачів системи.

Веб-застосунок має бути розробленний за допомогою сучасних актуальних технологій та мати належну архітектуру. Маэ бути дослідженні сучасні методології та технології для побудови веб-застосунків, проведене порівняння технологіх побудови веб-застосунків.

2 АНАЛІЗ СУЧАСНИХ ПЛАТФОРМ ДЛЯ ПОБУДОВИ WEB-ЗАСТОСУНКІВ

На даний момент існують і успішно застосовуються різні види технологій побудови Web застосунків серверної сторони. Усі такі застосунки мають загальну мету - реалізацію бізнес - логіки на стороні сервера й генерацію коду для клієнта. Також у всіх цих застосунків однакова архітектура взаємодії сервера й клієнта й загальний протокол взаємодії - HTTP. Загальна логіка роботи застосунка серверної сторони представлена на рисунку 2.1.

Рисунок 2.1 – Архітектура роботи серверної сторони

Як видно з рисунка 2.1, робота серверних застосунків відбувається в три основні етапи:

1) Запит. Клієнт, використовуючи web - браузер, ініціює запит до сервера.

2) Обробка запиту, підготовка відповіді. Після одержання запиту web-сервер проводить обробку запитуваного ресурсу. У випадку якщо запитується статичний ресурс, такий як HTML сторінка, Рисунок, документ, ця інформація форматується для протоколу HTTP і передається клієнтові в якості відповіді. Якщо ж запитується динамічний ресурс, запит передається на обробку відповідному до контейнера web - застосунків, де й відбувається подальша робота.

3) Після формування, дані передаються клієнтові за допомогою протоколу HTTP у якості відповіді. Відповідь містить дані (звичайно HTML код, або двійкові дані), а також додаткові параметри, передані в заголовках HTTP відповіді.

Робота застосунків серверної сторони завжди відбувається за описаним вище сценарієм. Такий підхід створює складності при розробці web - застосунків, основний з яких є відсутність стану в web - застосунка. Це означає, що застосунок працює винятково в режимі запит-відповідь, не маючи даних про попередні кроки користувача або якої-небудь інший постійної інформації. Як рішення - застосовується поняття користувацької сесії, яка дозволяє зберігати дані на сервері протягом сеансу роботи користувача.

Однак наявністю сесій складності при створенні web - застосунків повністю не усуваються. Чим більше можливостей надає платформа реалізації для застосунків серверної сторони в подоланні цих складностей, тем швидше й ефективніше може вестися розробка. Далі будуть розглянуті різні підходи до створення застосунків серверної сторони, їх гідності й недоліки, а також розглянуті конкретні платформи.

2.1 Підходи для створення web-застосунку

При розгляді платформ для створення серверних застосунків необхідно виділити декілька підходів:

- безпосередня обробка запитів і формування відповідей;
- вбудовування програмного коду в шаблони HTML сторінок.

Такий підхід надає найбільші можливості по керуванню обробкою й підвищенню продуктивності. Він передбачає передачу всіх даних про запит коду, що безпосередньо виконується, який може як сформувати відповідь зі сторінкою для користувача, так і відкрити на передачу потік двійкових даних, наприклад для передачі зображення. Однак при такому підході всі дані для передачі формуються програмним шляхом і це сповільнює розробку простих сторінок. Прикладом цього підходу служать технологія ASP. NET Core MVC.

Інший спосіб використовує шаблони сторінок користувача, оформлені особливим образом, що дозволяє вставляти в них ділянки програмного коду. Цей підхід особливо ефективний при створенні простих застосунків, основна інформація в яких статична, а динамічна інформація може бути з генерована простими програмними конструкціями. При розробці складних програмних систем цей варіант ускладнює взаємодія між компонентами й утрудняє реалізацію складної архітектури. Також він менш ефективний по продуктивності й обмежує можливості по реалізації складних сторінок. Прикладами цього підходу служать дуже популярні на даний момент технології ASP.NET Core, JSP, JSF.

Крім різного підходу до генерації сторінок платформи розробки в різному ступені задовольняють сучасним вимогам, висунутим при створенні складних Web-систем. Найбільш важливі із цих вимог, наявність яких робить систему привабливою для використання, наведені нижче:

- незалежність платформ;

- продуктивність, масштабованість;
- можливості розширення й інтеграції інших систем;
- простота використання, наявність засобів розробки;
- наявність програмних бібліотек.

Отже, було визначено ряд вимог, необхідних для сучасної платформи розробки. Нижче розглядаються найбільш популярні на даний момент платформи та оцінювання за наданими критеріями.

2.2 Огляд сучасних технологій побудови Web-застосунків

На даний момент існує багато розроблених технологій серверної сторони, як комерційних, так і вільно розповсюджуваних. Далі розглядаються найпоширеніші або перспективні технології, оскільки основна конкуренція йде між ними й при виборі основної технології реалізації в більшості випадків перевага віддається однієї з них.

Платформи розглядаються з погляду побудови складних Web-систем, тому деякі з популярних технологій не приводяться в докладному огляді через неможливість або недоцільність їх використання як базової платформи. Нижче приводяться тільки основні технології, потенційно придатні для створення складних гетерогенних Web систем.

1. Технологія .NET - це новітня розробка компанії Microsoft і заявлена як новий етап у розвитку засобів взаємодії між додатками. У даний момент вона доступна в якості доповнення .NET Framework до сімейства операційних систем Microsoft Windows. Також ведуться роботи зі створення .NET Framework на інших операційних системах. Платформа .NET спрощує розробку додатків і підвищує надійність коду. Зокрема, вона забезпечує автоматичне керування часом життя об'єктів, нейтральні до мов бібліотеки класів границі, що й перетинають, мов спадкування, обробку виключень і налагодження.

Основа .NET - Common Language Runtime (загальне середовище виконання мов) опирається на системні служби операційної системи й управляє виконанням коду, написаного на будь-якій сучасній мові програмування. Набір базових класів дає доступ до сервісів платформи, які розроблювачі можуть використовувати з будь-якої мови програмування. Common Language Runtime і базові класи разом становлять основу .NET платформи. NET пропонує також сервіси високого рівня ADO .NET - нове покоління ADO, яке використовує XML і SOAP для обміну даними;

ASP .NET - нова версія ASP, що дозволяє використання кожної мови з .NET стеку для програмування Web сторінок;

Web Forms - набір класів для побудови інтерфейсу користувача Web-орієнтованих застосунків.

Розгортання систем на платформі .NET здійснюється особливим образом. Вихідні коди компілюються не в команди процесора x86 або інші машинні коди. Замість цього компілятор створює код Проміжною Мовою Microsoft (Microsoft intermediate language - MSIL). Файл, що містить MSIL, може виконуватися на платформі будь-якого процесора, якщо операційна система, надає .NET CLR.

Важливою складовою частиною платформи .NET є нове середовище ASP.NET. Можливості ASP.NET настільки великі, що її складно назвати наступною версією ASP. У її основі лежить інші платформи, й основні мови програмування для неї обрані C# и Visual Basic. У той же час, нова технологія дозволяє писати ASP сторінки на будь-якій підходящій мові.

В ASP.NET закладено все, для того, щоб зробити весь цикл розробки web - застосунка більш швидким, а підтримку простіше. Нижче наведені основні можливості й принципи роботи ASP.NET.

- компілювання коду при першому обігу;
- широкий вибір бібліотек компонентів, що поставляються с.NET;
- підтримка потужного засобу розробки - Visual Studio;
- мовна незалежність у межах платформ для яких реалізовано загальне язикове середовище виконання CLR;
- можливості розширення за допомогою мультипроцесорних і кластерних рішень;
- потужні можливості по обробці помилок;
- об’єктно- орієнтовані мови розробок;
- розширені можливості повторного використання компонентів.

2. Технологія Common Gateway Interface (CGI), відрізняється від інших розглянутих технологій тим, що є найбільше низькорівневою і є стандартом інтерфейсу, який служить для зв'язку зовнішньої програми з Web-сервером.

Сам протокол розроблений таким чином, щоб можна було використовувати будь-яку мову програмування, яка може працювати зі стандартними обладнаннями введення/висновку. Оскільки така можливість є на рівні операційної системи, те, якщо не потрібно складний скрипт, його можна оформити у вигляді командного файлу.

Розглянемо основні переваги й недоліки технології CGI за виділеними критеріями:

- вбудованих засобів масштабованості технологія не передбачає, про це розроблювачам доводиться опікуватися окремо;

− програма CGI - представляє із себе готовий до виконання файл, що перешкоджає легкому розширенню системи;

− не накладає особливих умов на платформу й web - сервер, тому працює на всіх популярних платформах і web - серверах. Також технологія не прив'язана до конкретної мови програмування й може бути використана на будь-якій мові, що працює зі стандартними потоками введення/висновку;

− продуктивність CGI - програм не висока. Основною причиною цього є те, що при черговому звертанні до сервера для роботи CGI – програми створюється окремий процес, що вимагає великої кількості системних ресурсів.

Ці причини привели до того, що зараз розробці CGI - додатків віддають перевагу більш розвиненим платформам, що надають більше зручності розроблювачам, що володіють підвищеною продуктивністю. Однак більша маса вже розроблених додатків змушує зважати на технологію CGI, а її знання необхідне для розуміння роботи платформ високого рівня.

3. Технологія Java Server Pages (JSP) від компанії Sun Microsystems з'явилася надбудовою над технологією Java Servlets, що забезпечує більш швидку й просту розробку web - застосунків за допомогою застосування шаблонового підходу.

Для розуміння архітектури й переваг JSP необхідно знати технологію Java Servlets, оскільки вони тісно зв'язані. Сторінки Java Server Pages представляють із себе шаблони сторінок HTML, схожі із шаблонами PHP і ASP. Основною відмінністю від інших подібних технологій є те, що код, що перебуває усередині спеціальних тэгів не інтерпретується при звертанні до сторінки, а попередньо компілюється в Java Servlet. Статичні ділянки шаблону перетворяться у виклики до функцій для їхнього приміщення в потік висновку. Код компілюється так, ніби він перебував усередині сервлета. Компіляція JSP сторінок у сервлети є трудомісткої, але проводиться один раз - або при першому звертанні до сторінки, або при запуску сервлет - контейнера.

Технологія JSP поєднує шаблоновий підхід до побудови сайтів і всі переваги Java платформи. Завдяки цьому технологія одержала широке поширення як серед професійних комерційних розроблювачів, так і при створенні відкритих безкоштовних проектів. Важливим кроком до розширення шаблонового підходу стали так звані бібліотеки тэгів. Це гнучка можливість інтегрувати стандартні, сторонні, або власні програмні компоненти в сторінки. Простота створення й використання привели до великої популярності бібліотек тегів.

Завдяки роботі на основі Java технологія JSP не прив'язана до конкретної апаратної або програмної платформи. У такий спосіб JSP є відмінним розв'язком для використання в гетерогенних середовищах.

Продуктивність технології обмежена об'єктивними особливостями архітектури. По-перше, сторінки повинні бути відкомпільовані в сервлети, що забирає значний час. По-друге сервлети виконуються в середовищі виконання Java, тобто в режимі інтерпретації. Однак ці обмеження компенсуються додатковими можливостями. Сучасні контейнери підтримують кластеризацію серверів, що перекладає навантаження на апаратне забезпечення. Це є економічно виправданим і простим розв'язком. Завдання ж компіляції в сервлети є разовою й проводиться або при першому обігу, або при запуску сервлет - контейнера. У такий спосіб це не позначається на загальній продуктивності системи при розгляді за достатній період часу.

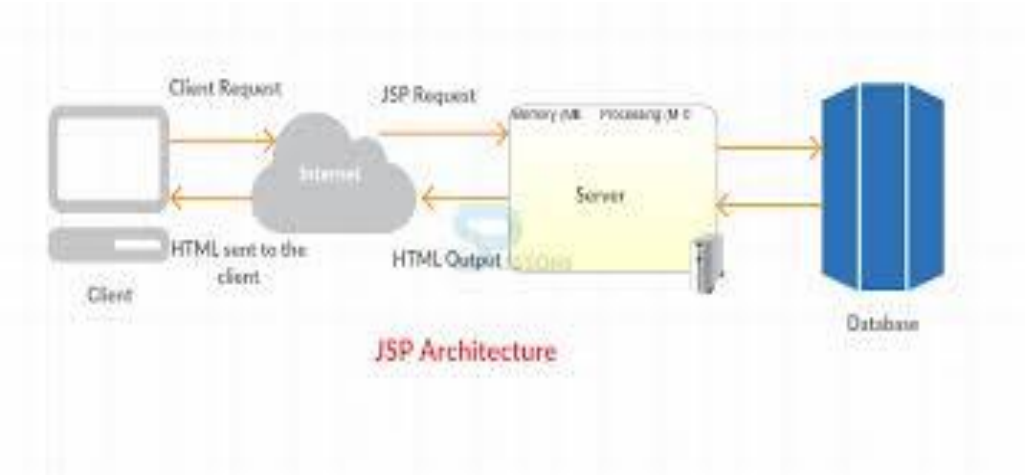


Рисунок 2.2 - Архітектура JSP

Основними перевагами JSP є простота розробки, характерна для шаблонового підходу, наявність великої кількості сторонніх бібліотек, легкість їх використання, потужні й різноманітні середовища розробки. При створенні складних Web - систем обмеження, що накладаються шаблоновим підходом серйознішають перешкодою до розвитку.

4. Spring Framework (або коротко Spring) - універсальний фреймворк з відкритим вихідним кодом для Java-платформи. Незважаючи на те, що Spring Framework не забезпечує будь-яку конкретну модель програмування, він став широко поширеним як альтернатива і заміна моделі Enterprise JavaBeans.

Перша версія була написана Родом Джонсоном, який вперше опублікував її разом з виданням своєї книги «Expert One-on- One Java EE Design and Development». Перший стабільний реліз 1.0 був випущений в 2004 році.

Незважаючи на те, що Spring Framework не забезпечував якусь конкретну модель програмування, він став широко поширеним в Javaспівтоваристві головним чином як альтернатива і заміна моделі Enterprise JavaBeans. Spring Framework надає велику свободу

Java- розробникам в проектуванні, крім того, він надає добре документовані і легкі у використанні засоби вирішення проблем, що виникають при створенні додатків корпоративного масштабу. Зазвичай Spring описують як полегшену платформу для побудови Java-додатків, але з цим твердженням пов'язані два цікавих моменти. По-перше, Spring можна використовувати для побудови будь-якої програми на мові Java (тобто автономних, веб додатків, додатків JEE і т.д.), що відрізняє Spring від багатьох інших платформ, таких як Apache Struts, яка обмежена тільки веб -Додатків. По-друге, характеристика "полегшена" насправді не має ніякого відношення до кількості класів або розміром дистрибутива, навпаки, вона визначає принцип всієї філософії Spring - мінімальний вплив. Платформа Spring є полегшеною в тому сенсі, що для використання ядра Spring ви повинні вносити мінімальні (якщо взагалі будь-які) зміни в код свого застосування.

Особливості ядра Spring Framework застосовні в будь-якому Java-додатку, і існує безліч розширень і удосконалень для побудови вебдодатків на Java Enterprise платформі. З цих причин Spring набув великої популярності і визнається розробниками як стратегічно важливий фреймворк.

5. Comet - це набір прийомів і засобів, що дозволяють швидко оновлювати дані в браузері (сторінки і їх елементи) без участі користувача. Від клієнта вимагається лише зайти на ресурс - з цього моменту сервер сам буде відправляти браузеру нові дані, якщо для нього є якась нова інформація. Відразу хочу звернути увагу на ключове відміну від AJAX, який буде працювати тільки в тому випадку, якщо треба відправити дані, і завершує своє життя відразу після отримання відповіді. Нажаль, створення запиту через AJAX - це завжди дорого і довго, та й браузер обмежує кількість одночасних з'єднань.

Ключовий момент Comet'a полягає в тому, що не клієнт (браузер), а сервер вирішує, коли треба відправити користувачеві дані - такий підхід називається server-push. Найпершим варіантом застосування такого підходу стали чати: нові повідомлення з'являються миттєво, сторінка не оновлюється - повне враження звичайного застосування. Звичайно, аналогічну функціональність давно можна реалізувати через Flash, але він повільний і закритий. Comet дозволяє обійтися звичайними простими засобами - JavaScript + HTML, отримуючи миттєвість оновлень і простоту.

6. Ruby on Rails (RoR) - це багаторівневий MVC-фреймворк для побудови веб-додатків, що використовують реляційні і NoSQL бази даних (наприклад, MySQL, MariaDB, PostgeSQL, MongoDB). Фреймворк написаний на мові програмування Ruby. Rails підходить як для розробки звичайних сайтів, які повинні бути реально швидкими,

відмовостійкими і працюють під високим навантаженням, так і для веб-додатків зі складною бізнес-логікою і динамічними web-інтерфейсами. Ruby on Rails є відкритим програмним забезпеченням і розповсюджується під ліцензією MIT.

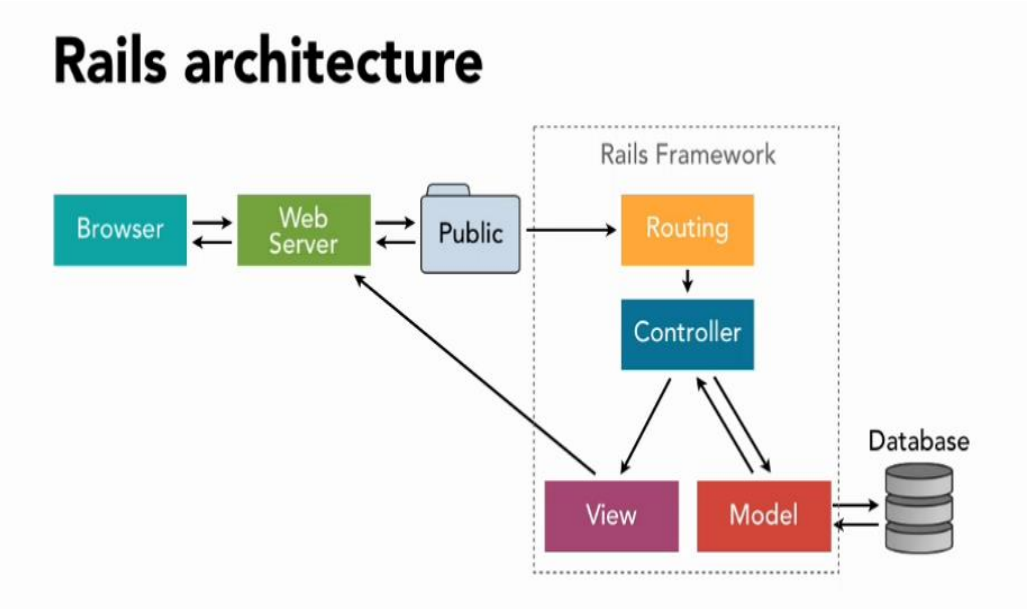


Рисунок 2.3 - Архітектура Ruby on Rails

Основними принципами розробки web-застосунків на Ruby on Rails є:

Принцип DRY (Do not repeat yourself) - фреймворк надає механізми повторного використання програмного коду. Це дозволяє не тільки мінімізувати дублювання коду, але і підвищити швидкість розробки.

Принцип Convention over configuration - за замовчуванням у фреймворку використовуються численні угоди по конфігурації, типові для більшості додатків. Це дуже спрощує створення додатків, так як явна специфікація конфігурації потрібно тільки в нестандартних випадках.

Автоматизоване тестування – в складі RoR поставляються засоби для проведення повністю автоматичного модульного, інтеграцій-ного і функціонального тестування, а ідеологія Ruby on Rails пе-редбачає використання методів розробки через тестування (TDD - Test Driven Development). Все це робить розроблені додатки надійними.

7. JavaServer Faces (JSF) – це фреймворк для веб-додатків, написаний на Java. Він служить для того, щоб полегшувати розробку користувальницьких інтерфейсів для Java web-додатків. Розширення Web-сервера не є самим зручним засобом розробки, жорстко прив'язують систему до певного Web-серверу, але демонструють максимальну продуктивність і дають найбільшу гнучкість у розробці.

Стан компонентів для користувача інтерфейсу зберігається, коли користувач запитує нову сторінку і потім відновлюється, якщо запит повторюється.

Технологія JavaServer Faces включає:

Набір API для подання компонент для користувача інтерфейсу і управління їх станом, обробкою подій, а також підтримку доступності.

Спеціальна бібліотека JSF тегів для вираження інтерфейсу JSF на JSF сторінці.

Створена бути гнучкою, технологія JavaServer Faces підсилює існуючі стандартні концепції призначеного для користувача інтерфейсу і концепції Веб-рівня без прив'язки розробника до конкретної мови розмітки, протоколу або клієнтського пристрою. Класи компонентів для користувача інтерфейсу, що поставляються разом з технологією JSF, містять функціональність компонент, а не специфічне для клієнта відображення, відкриваючи тим самим можливість рендеринга JSF- компонент на різних клієнтських пристроях. Поєднуючи функціональність компонентів інтерфейсу користувача зі спеціальними рендерерами, розробники можуть конструювати спеціальні теги для заданого клієнтського пристрою.

Як зручності технологія JSF надає специфічний рендерер і спеціальну бібліотеку JSF-тегів для рендеринга на HTML- клієнті, дозволяючи розробникам додатків на Java EE платформі використовувати технологію JSF в своїх додатках.

Очевидно, що платформа .NET і ASP.NET надали нові можливості по розробці Web - систем. Вони відповідають усім сучасним вимогам і дозволяють значно прискорити й спростити розробку складних додатків. Однак, на даний момент, .NET у повному обсязі існує тільки для платформи Windows. Розробки по переносу на інші системи ведуться, але ще не завершені і їх майбутні результати важко оцінити. Що стосується розробки сайтів, то ASP.NET сильно прив'язана до сервера IIS, але архітектура .NET дозволяє перенести застосунок ASP.NET на іншу платформу. Однак необхідно відзначити, що така система повинна мати можливості інтеграції із платформою .NET (особливо Web-сервіси), оскільки її майбутнє широке використання не викликає сумнівів.

2.3 Порівняльна характеристика базових технологій побудови web-застосунків

У попередньому підрозділі були розглянуті найбільш популярні базові технології побудови додатків серверної сторони. З розглянутого можна виділити наступні основні підходи до архітектури серверних додатків:

− окреме виконання запитів. При кожному запиті динамічного вмісту, запускається окрема програма для обробки запитів. Програма генерує вміст, передане клієнтові. Цей підхід використовується в класичних CGI-скриптах;

− нагромадження процесів, що виконуються. Модель окремого виконання запитів суттєво обмежує продуктивність. Варіант нагромадження процесів є розвитком цієї технології, підвищує продуктивність, при цьому зберігаючи максимальну гнучкість розробки. Підхід аналогічний попередньому, але при цьому якщо запит виконується повторно, нового запуску програми не відбувається, а обробка передається існуючому процесу. Даний підхід застосовується в технологіях Java Servlets, Fast CGI;

− шаблони сторінок. При запиті шаблони заповнюються динамічним змістом, звичайно, але необов'язково, створюваним мовою сценаріїв. Розширення Web - сервера. Web - сервер звертається до особливих розширень для обробки динамічного змісту. Розширення специфічні для Web-сервера. Цей підхід використовується в IS API, NSAPI.

Кожний із зазначених підходів має свої можливості й обмеження, і, відповідно, свою область застосування. Модель окремого виконання запитів суттєво обмежує продуктивність. Варіант нагромадження процесів є розвитком цієї технології, Шаблоновий підхід надзвичайно зручний при розробці невеликих систем, однак при збільшенні складності він починає гальмувати процес розробки й не є підходящим для великих систем. Він також відрізняється невисокою продуктивністю, хоча дослідження показують, що в певних умовах можуть демонструвати досить високі показники й конкурувати з підходом, підвищує продуктивність, при цьому зберігаючи максимальну гнучкість розробки. Шаблоновий підхід надзвичайно зручний при розробці невеликих систем, однак при збільшенні складності він починає гальмувати процес розробки й не є підходящим для великих систем. Він також відрізняється невисокою продуктивністю, хоча дослідження показують, що в певних умовах можуть демонструвати досить високі показники й конкурувати з підходом №2. Розширення Web-сервера не є самим зручним засобом розробки, жорстко прив'язують систему до певного Web-серверу, але демонструють максимальну продуктивність і дають найбільшу гнучкість у розробці.

Розглянемо платформи по вимогах, певних раніше. CGI не входить в огляд, оскільки є незручною у використанні, що й має низьку ефективність, а розширення серверів занадто сильно прив'язані до конкретних програмних продуктів. За схемою обробки запитів платформи розподіляються в такий спосіб:

− шаблони PHP - виконуються на Web - сервері Apache. Інтерпретатор може бути розширенням сервера (в експериментальному режимі IIS);

− шаблони Java Servlets – забагато процесів для кожного сервлета;

− шаблони JSP - шаблони. Виконується предкомпиляція в Java Servlets, дозволяючи використовувати схему нагромадження процесів;

− шаблони ASP.NET MVC. Використовується схема попередньої компіляції, а не інтерпретації коду. У результаті використовується розширення Web – сервера IIS. Можуть використовуватися й оброблювачі низького рівня;

− шаблони JSF. Шаблони інкапсулюють загальну функціональність різних уявлень додатків, так що цю функціональність доводиться описувати лише одного разу. Один шаблон використовується в декількох композиціях для створення уявлень додатки JSF;

− шаблони Spring Framework - кожен запит перехоплюється глобальним Front-контролером, який за специфічними параметрами визначає, якому з контролерів передати отриманий запит.

Основні оцінні характеристики платформ зрівняємо у зведеній таблиці, де "-"-відсутність підтримки, "+" - підтримка. Для порівняльних характеристик, таких як мова реалізації або продуктивність, оцінки відповідатимуть ступеню переваги технології. Результати можна побачити на таблиці 2.1

Таблиця 2.1 – Порівняльна характеристика платформ для розробки

	Spring	Comet	JSP	JSF	Ruby on Rails	ASP.NET
Кросплатформеність	+	+	+	+	+	+
Масштабованість	+	+	+	+	+	+
Розширюваність	+	+	+	+	-	+
Простота	-	-	-	-	-	+
Наявність бібліотек	+	-	+	+	-	+
Поділ дизайну й логіки	+	+	+	+	-	+

2.4 Концепція побудови сучасних web-застосунків

Концепція MVC широко застосовується для побудови сучасних web- застосунків. Метою даної концепції є поділ усього веб-застосунка на три компоненти: модель даних, представлення й контролер. Це шаблон проектування, який вперше був опублікований в 1970-х роках. Він являє собою зразок архітектури програмного забезпечення, заснований на принципі поділу представлення даних і функціоналу, де ці дані формуються і обробляються. У теорії, якщо слідувати принципу шаблону, то frond-ent і, back-end команди можуть працювати над різними частинами одних і тих же компонентів, не заважаючи один одному.

Модель (Model). Модель надає знання: дані й методи роботи із цими даними, реагує на запити, змінюючи свій стан. Не містить інформації, як ці знання можна візуалізувати.

Представлення, вид (View). Відповідає за візуалізацію. Часто в якості вистави виступає форма із графічними елементами.

Контролер (Controller). Забезпечує зв'язок між користувачем і системою: контролює введення даних користувачем і використовує модель і представлення для реалізації необхідної відповіді. Контролер виконує усі дії для зв'язку між моделлю та представленням.

Схематично концепція MVC представлена на рисунку 1.2.

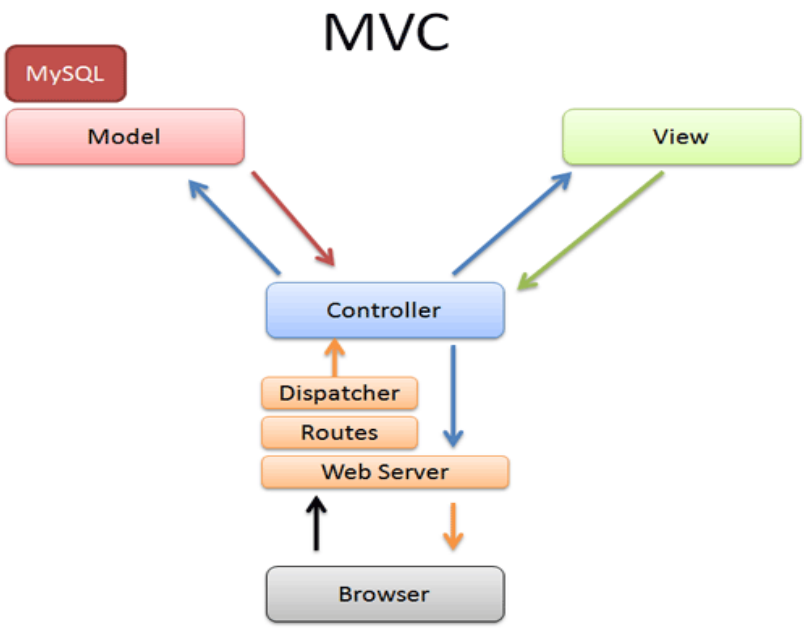


Рисунок 2.4 – Схематична вистава концепції MVC

Важливо відзначити, що як представлення, так і контролер залежать від моделі. Однак модель не залежить ні від вистави, ні від контролера. Тим самим досягається призначення такого поділу: воно дозволяє будувати модель незалежно від візуальної вистави, а також створювати кілька різних вистав для однієї моделі. У такий спосіб застосунок легше масштабувати й розширювати системи, що проектуються.

У даному розділі були розглянуті найбільш популярні технології для побудови сучасних веб-застосунків. Із проведеної порівняльної характеристики було визначено, що найбільш зручною в реалізації та подальшої підтримки є .NET технологія, а так само супутній стек технологій ASP.NET MVC, що є оптимальним вибором для створення захищеного та ефективного веб-застосунка. Вибір даного стека технологій обумовлюється безпекою, відкритістю й прозорістю коду, а так само можливістю вільного поширення. Також технологія була вибрана за відносну легкість у вивченні і застосуванні фреймворка в розробці і підтримці програми, впровадженням залежностей (DI) і інверсії управління (IoC) дозволяють писати незалежні один від одного компоненти, що дає переваги в командній розробці, переносимості модулів, а отже спрощує підтримку програми.

3 МАТЕМАТИЧНА МОДЕЛЬ ПОСТАВЛЕНОЇ ЗАДАЧІ

Дано повідомлення яке потрібно передати і ключ(пара ключів) для шифрування. Необхідно побудувати функцію для перетворення початкового повідомлення в зашифрований вигляд з метою унеможливлювання прочитання початкового тексту до початкового стану. Математичні дані, які використовують в шифруванні, можна представивти у вигляді множини, що представляють дані M – та C і кожна з двох функцій є відображенням однієї з цих множин в іншу:

- шифруюча функція E: M -> C;
- дешифруюча функція : D: C -> M.

Елементи цих множин – m та c – є аргументами відповідних функцій. Також у ці функції вже включено поняття ключа. Тобто, необхідний ключ для шифрування або дешифрування є частиною функції. Це дозволяє розглядати процеси шифрування абстрактно, незалежно від струкрури використовуваних ключів. Хоча, в загальному випадку для кожної з цих функцій аргументами є дані і вводиться ключ:

$$E_{K1}(m) = c, \tag{3.1}$$

$$D_{K2}(c) = m. \tag{3.2}$$

Якщо для шифрування і розшифрування використовується один і той же ключ K = K_1 = K_2, то такий алгоритм відносять до симетричних. Якщо ж з ключа шифрування алгоритмічного складно отримати ключ розшифрування, то алгоритм відносять до асиметричних, тобто до алгоритмів з відкритим ключем.

Вимоги до функцій:

- для застосування в цілях шифрування ці функції, в першу чергу, повинні бути взаємно оберненими (D = E-1);
- має бути виконана вимога DK2(EK1(m)) = m;
- має бути виконана вимога EK1(DK2(c)) = c;
- важливою характеристикою шифрувальної функції E є її криптостійкість. Непрямою оцінкою криптостійкості є оцінка взаємної інформації між відкритим текстом і шифротекстом, яка повинна прямувати до нуля.

3.1 Модель систем симетричного шифрування

Алгоритм і ключ вибирається заздалегіть і відомий обом сторонам. Збереженням ключа в секретності є важливим завданням для встановлення і підтримки захищеного каналу зв’язку.

Схема реалізації.

Є два співрозмовника, вони хочуть обмінюватися конфіденційною інформацією. Задача:

− генерація ключа. Перший учасник виюирає ключ шифрування d і алгоритм E, D(функції шифрування і дешифруання), потім посилає цю інформацію другому;

− шифрування і передача повідомлення. Другий учасник шифрує інформацію з використанням отриманого ключа d: E(m, d) = c та передає першому отриманий шифротекст c;

− розшифрування повідомлення. Перший за допомогою того ж ключа d, розшифровує шифртекст c : D(c, d) = m.

3.2 Модель систем асиметричного шифрування

Відкритий ключ передається по відкритому каналу і використовується для шифрування повідомлення або для перевірки цифрового пілпису. Для розшифрування повідомлення і для генерації цифрового підпису використовується секретний ключ.

Схема реалізації:

Є два співрозмовника, вони хочуть обмінюватися конфіденційною інформацією. Задача:

− генерація ключової пари. Другий вибирає алгоритм (E, D) і пару ключів – відкритий та закритий (e, d) та відправляє відкритий ключ e першому по відкритому каналу;

− шифрування і передача повідомлення. Перший шифрує інформацію з використанням відкритого ключа e: E(m, e) = c і передає отриманий шифотекс c;

− розшифрування повідомлення. Другий за допомогою закритого ключа d, розшифровує шифротекст c : D(c, d) = m;

− якщо необхідно налагодити канал зв’язку в обидві сторини, то перші дві операції необхідно виконати на обох сторонах, таким чином, кожен буде знати свої закритий, відкритий ключі та відкритий ключ співрозмовника. Закритий ключ кожної сторони не передається по незахищеному каналу, тим самим залишаючись в секретності.

В даному розділі сформульовано математичну модель поставлені задачі для обраних алгоритмів. Була проаналізована задача, коли два співрозмовника хочуть обмінятися конфіденційною інформацією. Для цього було досліджено генерацію ключової пари для асиметричного шифрування та генерація ключа для симетричного, обрано алгоритм для кожного із них, проаналізовано шифрування та передачу повідомлення за допомогою відкритого або звичайного ключа, залежачи від типу шифрування, також було досліджено спосіб розшифрування повідомлення за допомогою того ж ключа, що й для шифрування для симетричного алгоритму, та розшифруваннння за допомогою відкритого і закритого ключа для асиметричного алгоритму. Були розібрані вимоги до передачі закритого ключа.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ

4.1 Опис функцій

Основними користувачами цього програмного продукту можуть бути розробники сторонніх додатків, які хочуть використовувати шифрування та дешифрування даних у своїх потребах. Тому дане програмне забезпечення має виконувати три основні функції:

- генерування ключа, стійкого до криптоатак;
- шифрування даних обраним алгоритмом;
- дешифрування даних обраним алгоритмом.

4.2 Проектування веб-застосунку

Веб-застосунок буде реалізований як сервіс, а саме як Web API, для шифрування та дешифрування даних, який може бути використаний як частина будь-якої системи.

Веб-застосунок буде доступний як набір функцій для шифрування або дешифрування бажаних даних декількома різними варіантами шифрування.

Для реалізації веб-застосунку була обрана технологія ASP .NET Core.

ASP.NET Core — вільне та відкрите програмне забезпечення каркасу веб-застовсунків з продуктивністю вищою ніж у ASP.NET розроблена корпорацією Microsoft і співтовариством. Це модульна структура, яка працює як на повній платформі .Net Framework, так і на платформі .Net Core.

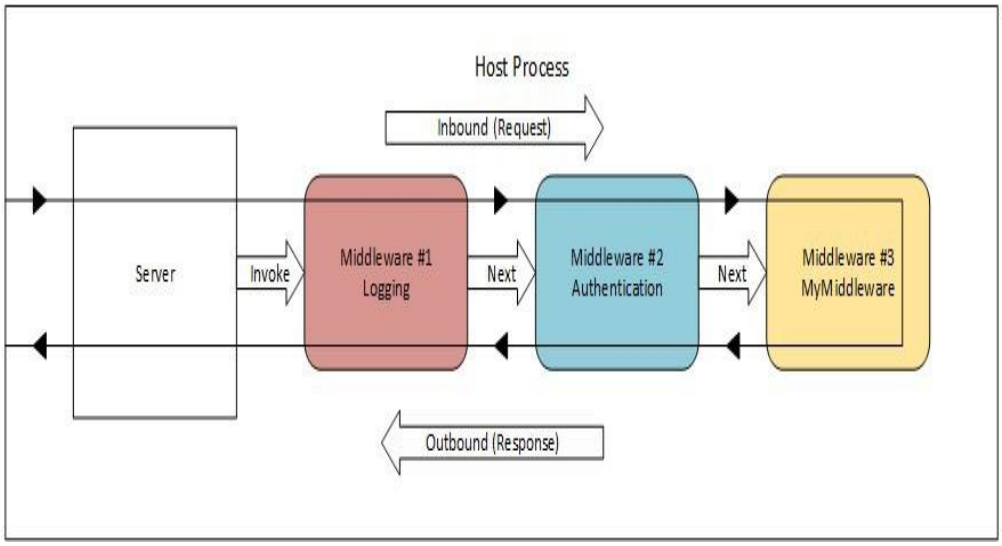


Рисунок 4.1 - Приклад використання middleware у процесі обробки запиту

ASP.NET Core повністю є opensource-фреймворком. Всі вихідні файли фреймворку доступні на GitHub. ASP.NET Core може працювати поверх крос-платформної середовища .NET Core, яка може бути розгорнута на основних популярних операційних системах: Windows, Mac OS X, Linux. І таким чином, за допомогою ASP.NET Core ми можемо створювати крос-платформні додатки. І хоча Windows як середовище для розробки і розгортання програми досі превалює, але тепер вже ми не обмежені тільки цією операційною системою. Тобто ми можемо запускати веб-додатки не тільки на ОС Windows, але і на Linux і Mac OS. А для розгортання веб-додатки можна використовувати традиційний IIS, або крос-платформний веб-сервер Kestrel. Хоча ASP.NET Core переважно націлене на використання .NET Core, але фреймворк також може працювати і з повною версією фреймворка .NET. Завдяки модульності фреймворка всі необхідні компоненти веб-додатки можуть завантажуватися як окремі модулі через пакетний менеджер Nuget. Крім того, на відміну від попередніх версій платформи немає необхідності використовувати бібліотеку System.Web.dll. ASP.NET Core включає в себе фреймворк MVC, який об'єднує функціональність MVC, Web API і Web Pages. У попередніх версії платформи дані технології реалізувалися окремо і тому містили багато дублюючої функціональності. Зараз же вони об'єднані в одну програмну модель ASP.NET Core MVC. А Web Forms повністю пішли в минуле. Крім об'єднання вищезазначених технологій в одну модель в MVC був доданий ряд додаткових функцій. Однією з таких функцій є тег-хелпери (tag helper), які дозволяють більш органічно поєднувати синтаксис html з кодом C #. ASP.NET Core характеризується розширюваністю. Фреймворк побудований з набору щодо незалежних компонентів. І ми можемо або використовувати вбудовану реалізацію цих компонентів, або розширити їх за допомогою механізму спадкування, або зовсім створити і застосовувати свої компоненти зі своїм функціоналом.

Фреймворк являє собою повний перепис, який об'єднує раніше окремі ASP.NET MVC та ASP.NET Web API у програмувальну модель.

Не зважаючи на те, що це є новим фреймворком, побудованим на новому веб-стеку, ASP.NET Core має високу ступінь сумісності концепцій з ASP.NET MVC, який об'єднує функціональність MVC, Web API та Web Pages. В попередніх версіях платформи дані технології реалізовані окремо і тому містять багато дублювальної функціональності. Тепер це об'єднано в одну програмну модель ASP.NET Core MVC. Веб-форми повністю вийшли в минуле. Програми ASP.NET Core підтримують програмні версії, в якій різні програми, що працюють на одному комп'ютері, можуть орієнтуватися на різні версії ASP.NET Core. Це не можливо з попередніми версіями ASP.NET Core.

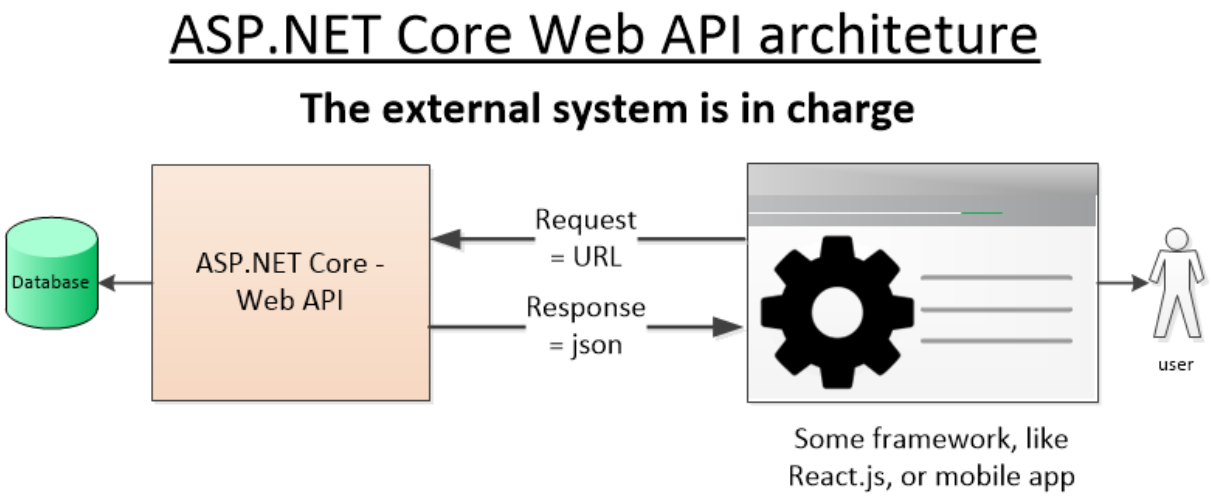


Рисунок 4.2 - Архітектура ASP .Net Core Web API

Переваги .NET Core:

- розробка та впровадження крос-платформних платформ;
- продуктивність є основним акцентом;
- спрощена модель хостингу;
- відкритий код (функції, які запитували спільноти);
- модульна структура розподілена як пакети NuGet ;
- Cloud-optimized runtime (оптимізована для Інтернету);
- Інтегрується з популярними рамками розробки клієнтської сторони.

4.3 Безпека даних в технології OpenVPN

OpenVPN є програмним додатком з відкритим вихідним кодом, який використовує можливості віртуальної приватної мережі (VPN) для створення захищеного з'єднання точка-точка або сайт-сайт на всьому маршруті проходження інформації, а також при використанні містків конфігурацій дистанційно. Усередині OpenVPN використовуються протоколи безпеки SSL (Secure Socket Layer - рівень захищених сокетів) або TLS (Transport Layer Security - безпека транспортного рівня), за допомогою яких користувачі можуть обмінюватися ключами.

Програмний додаток було написано Джеймсом Yonan і публікується під ліцензією GNU General Public License (GPL).

OpenVPN дозволяє партнерам перевірити справжність один одного, використовуючи заздалегідь узгоджений, яким користуються секретний ключ, сертифікати або ім'я користувача / пароль.

При використанні в конфігурації multiclient-сервера, це дозволяє серверу випустити сертифікат перевірки автентичності для кожного клієнта, використовуючи підпис і сертифікат повноважень. Він використовує бібліотеку OpenSSL шифрування, а також протокол SSLv3 / TLSv1, і містить безліч функцій безпеки та управління.

Користування глобальною мережею відкриває нам широкі можливості, одночасно приховуючи в собі масу небезпек, з якими стикається як звичайний користувач, так і цілі організації. Безпека і анонімність часто стають наріжним каменем, нехтування якими тягне за собою неприємні наслідки. Загублені фінансові кошти під час проведення валютних транзакцій, акаунти, які зазнали злому, конфіденційна інформація, яка стала загальнодоступною - список можна продовжувати нескінченно довго. І, на жаль, програмне забезпечення, таке, як антивіруси або фаєрволи, принципово не може вберегти нас від усіх ризиків, які неминучі при роботі в Інтернеті.

Так як інформація проходить найчастіше по незашифрованому каналу, вона стає вразливою. Антивірусне ПЗ забезпечує лише захист самого комп'ютера, але ніяк не переданих мережевих даних, а цією особливістю можуть скористатися зловмисники. Перехоплення даних листування в соц-мережах і відправляються анкетних даних, історії відвідувань та іншої інформації - небажані наслідки, які можуть бути наслідком низького рівня захисту.

Але рішення є - шифрування даних за допомогою OpenVPN. Технологія OpenVPN - одна з реалізацій VPN (віртуальної приватної мережі), покликана вирішити ці проблеми, для тих випадків, коли традиційне ПО для захисту безсило. Використовуються стандартні протоколи передачі транспортного рівня - TCP або UDP.

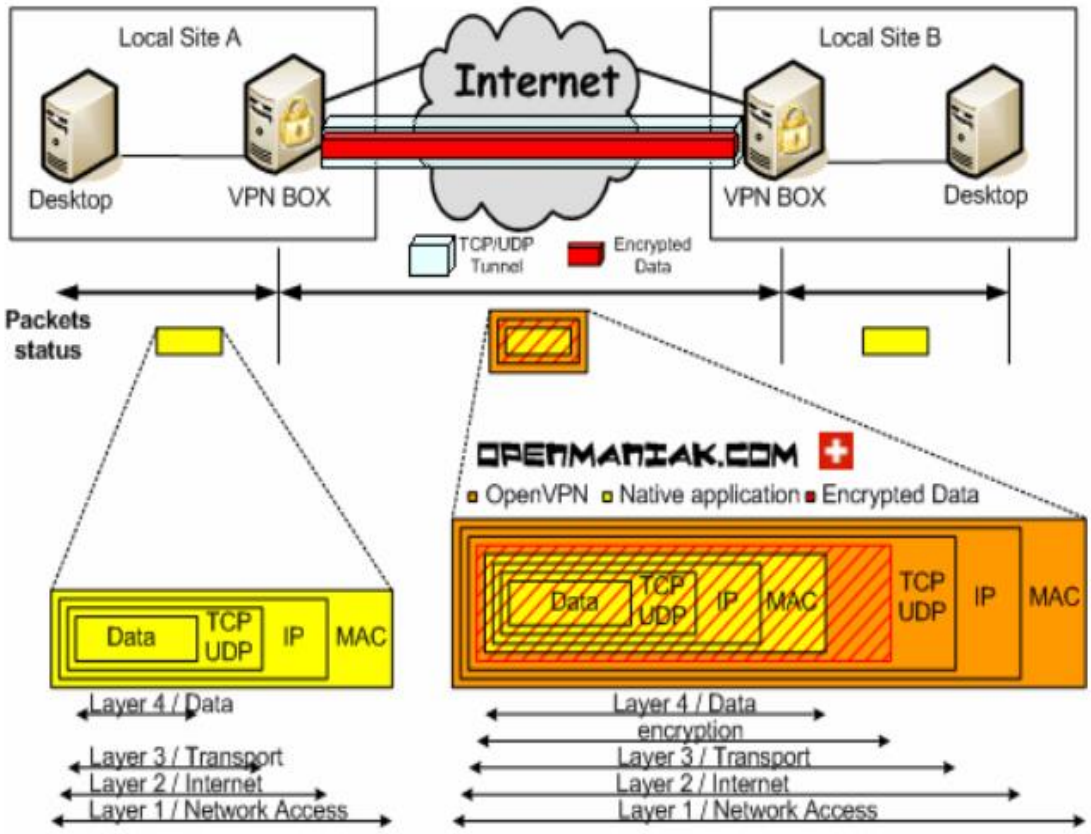


Рисунок 4.3 - Передача пакетів у локальній мережі та VPN

Спочатку встановлюється безпечний інформаційний канал між комп'ютером і VPN-сервером, який використовує принципи виділеного. Потім, створюється Peer-to-Peer з'єднання - тунель від точки до точки. Дані, які передаються і приймаються VPN-сервером, йдуть по шифрованому каналу, захист якого забезпечує безпеку і анонімність. Шифрування і дешифрування даних відбувається як на сервері, так і на клієнтській машині. З'єднання реалізується за допомогою створення віртуального TUN / TAP адаптера, емулює локальну мережу.

Користувачеві необхідно завантажити та встановити клієнтську частину програми OpenVPN. А провайдер послуги надасть всі необхідні дані для установки з'єднання з сервером. Незважаючи на простоту настройки OpenVPN і ясну структуру принципу роботи - це потужний інструмент захисту даних.

Технологія OpenVPN дає багато переваг:

– захист Інтернет-трафіку. Як уже зазначалося, дані йдуть по зашифрованому каналу, а значить, не доступні стороннім особам. Це може бути листування, передача файлів, потокове аудіо або відео, грошові перекази, дані валютних операцій та інша важлива інформація, яка повинна бути огороджена від сторонніх очей. Навіть в разі перехоплення трафіку OpenVPN, скористатися отриманими даними буде проблематично.

Для їх розшифровки необхідно задіяти серйозні комп'ютерні потужності - навіть сучасним кластерним системам знадобляться місяці, а то й роки, для злому ключа;

− анонімність в Інтернеті. VPN-сервер може розташовуватися, в залежності від уподобань, в будь-якій точці земної кулі. Це означає, що реальний IP адреса буде відповідати місцю розташування сервера. А це додатковий плюс до анонімності, адже навіть багато сайтів, де публікують коментарі до новин, зберігають дані IP відправника, доступні публічно;

− можливість обійти заборони. Найчастіше користування Інтернетом пов'язане з масою обмежень. Налаштування користування глобальною мережею на підприємстві можуть містити списки сайтів, на які не можна заходити, найчастіше - це соціальні мережі. Можлива також блокування операцій за певними протоколами, що перешкоджає повноцінному користування Інтернетом. Оскільки трафік віртуальної мережі не пов'язаний з настройками обмежень відвідуваних доменів або використання сервісів, виділений тунель OpenVPN з легкістю обійде ці перепони;

− робота без прив'язки до реального розташуванню. Багато сайтів Інтернету доступні не скрізь. Це може бути реалізовано з метою обмеження непотрібного трафіку, який додатково уповільнює роботу ресурсів. Деякі веб-сайти потрапляють під заборону у зв'язку з місцевим законодавством або цензурою. Гнучкість вибору IP забезпечить VPN-сервер, що дозволить отримати доступ до необхідних ресурсів;

− оптимізація трафіку. Незважаючи на високі швидкості, які надають Інтернет-провайдери, ліміти передачі даних все ж є. Часто організація встановлює межу Інтернет-трафіку для кожного зі своїх співробітників. Технологія OpenVPN стискає передані дані в рази, що дозволяє розширювати встановлені обмеження;

− виділений IP. Віддалене підключення володіє ще однією перевагою. Це може бути передача даних, або використання ресурсів комп'ютера або мережі, які територіально віддалені. Що і є додатковими опціями, які забезпечує виділений IP, наданий сервером OpenVPN;

OpenVPN - це простота, що володіє міццю, технологія, яка розширює свободу і захист, зводячи її на принципово новий рівень.

4.4 Проектування веб-застосунків за допомогою Angular

Angular (версія 2 і вище) - це відкрита і вільна платформа для розробки веб-додатків, написана на мові TypeScript, що розробляється командою з компанії Google, а також спільнотою розробників з різних компаній. Angular - це повністю переписаний фреймворк від тієї ж команди, яка написала AngularJS.

Angular - це платформа, яка спрощує розробку веб-додатків. Angular поєднує в собі декларативні шаблони, ін'єкцію залежностей, комплексний end-to-end інструментарій та вже імплементовані best practice для вирішення будь-якої складності завдань. Angular дозволяє створювати не тільки для веб-додатки але так само мобільні і десктопні програми.

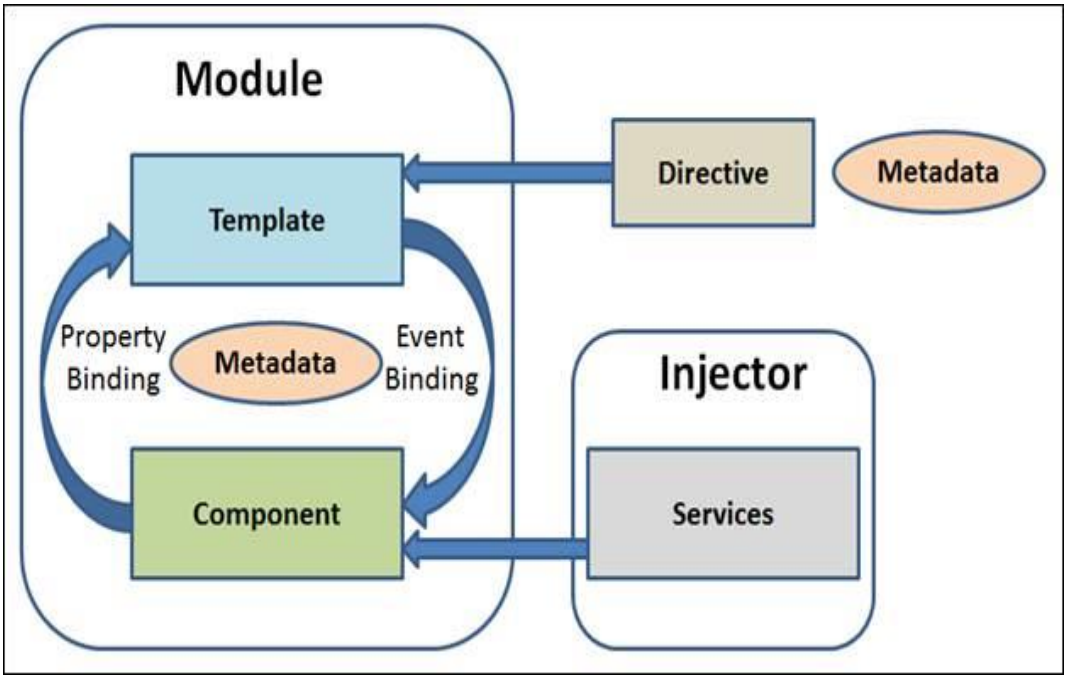


Рисунок 4.4 - Архітектура Angular веб-застосунку

AngularJS розроблений в 2009 році Мишко кенанеянина і Адамом Абронсом в Brat Tech LLC як програмне забезпечення позаду сервісу зберігання JSON-даних, що вимірюються мегабайтами, для полегшення розробки корпоративних додатків. Сервіс розташовувався на домені «GetAngular.com» і мав кількох зареєстрованих користувачів, перш ніж вони вирішили відмовитися від ідеї бізнесу і випустити Angular як бібліотеку з відкритим вихідним кодом.

Абронс покинув проект, але Хевері, що працює в Google, продовжує розвивати і підтримувати бібліотеку з іншими співробітниками Google Ігорем Мінар і Войта Джином.

AngularJS спроектований з переконанням, що декларативне програмування найкраще підходить для побудови призначених для користувача інтерфейсів і опису програмних компонентів [6], в той час як імперативне програмування відмінно підходить для опису бізнес-логіки [7]. Фреймворк адаптує і розширює традиційний HTML, щоб забезпечити двосторонню прив'язку даних для динамічного контенту, що дозволяє автоматично синхронізувати модель і уявлення. В результаті AngularJS зменшує роль DOM-маніпуляцій і покращує тестованого.

Цілі розробки:

– відділення DOM-маніпуляції від логіки додатки, що покращує тестованих коду;

– ставлення до тестування як до важливої частини розробки. Складність тестування безпосередньо залежить від структурованості коду.;

– поділ клієнтської і серверної сторони, що дозволяє вести розробку паралельно;

– проведення розробника через весь шлях створення програми: від проектування призначеного для користувача інтерфейсу, через написання бізнес-логіки, до тестування.

Angular дотримується MVC-шаблону проектування і заохочує слабкий зв'язок між поданням, даними і логікою компонентів. Використовуючи впровадження залежності, Angular переносить на клієнтську сторону такі класичні серверні служби, як відозавісимие контролери. Отже, зменшується навантаження на сервер і веб-додаток стає легше.

Angular дозволяє вам з "коробки" створювати великі і складні за частиною бізнес-логіки додатка. Angular було повним переосмисленням AngularJS, напевно, це було найболючіше, але воно того варте, сам фреймворк став куди чистіше і гнучкіше, більш enterprise-подібним і з цієї точки зору має високу масштабованість.

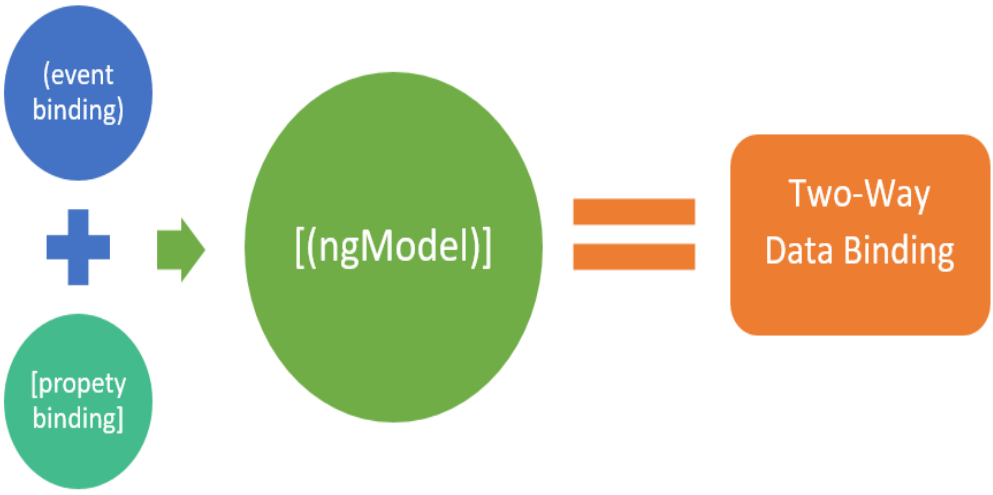


Рисунок 4.5 - Модель двобічної прив'язки моделі в Angular

Які плюси можна виділити:

- підтримка Google, Microsoft;
- інструменти розробника (CLI);
- єдина структура проекту;
- typeScript з "коробки" (ви можете писати строго типізований код);
- реактивний програмування з RxJS;
- єдиний фреймворк з Dependency Injection з "коробки";
- шаблони, засновані на розширенні HTML;
- кросбраузерності Shadow DOM з коробки (або його емуляція);
- кросбраузерності підтримка HTTP, WebSockets, Service Workers;
- не потрібно нічого додатково налаштовувати. Більше ніяких обгорток;
- більш сучасний фреймворк, ніж AngularJS (на рівні React, Vue);
- велике ком'юніті.

Мінуси:

- вище поріг входження через Observable (RxJS) і Dependency Injeciton;
- щоб все працювало добре і швидко потрібно витрачати час на додаткові оптимізації (він не супер швидкий, за замовчуванням, але швидше AngularJS).

Насправді, якщо ви плануєте розробляти велику enterprise-додаток, то в цьому випадку у вас немає архітектури для управління станом з "коробки" - потрібно додавати Mobx, Redux, CQRS / CQS або інший state-менеджер.

4.5 Проектування статистики

Для працездатності та обробки системою статистичних даних необхідно зберігати наступні дані:

- запит користувача (ID запиту, дата запиту);
- дані про шифрування (ID, тип шифрування або дешифрування, дата операції, час обробки запиту);
- статистика (ID запису, ID запиту користувача, ID даних про шифрування).

Тож, у майбутньому адміністратори, які будуть використовувати цей сервіс, будуть мати змогу використовувати ці статистичні дані на власні потреби, а саме:

- дізнатися, аякий шифр найчастіше використовувався;
- дізнатися час обробки запиту сервісом для майбутньої модернізації системи;

– дізнатися час шифрування та дешифрування файла.

4.6 Приклад використання

Веб-застосунок може бути використаний у будь-який спосіб будь-якою програмою, наприклад може бути у програмі, яка шифррує і дешифрує дані. Шифрвати можно такі дані як текст, картинки та будь-які файли.

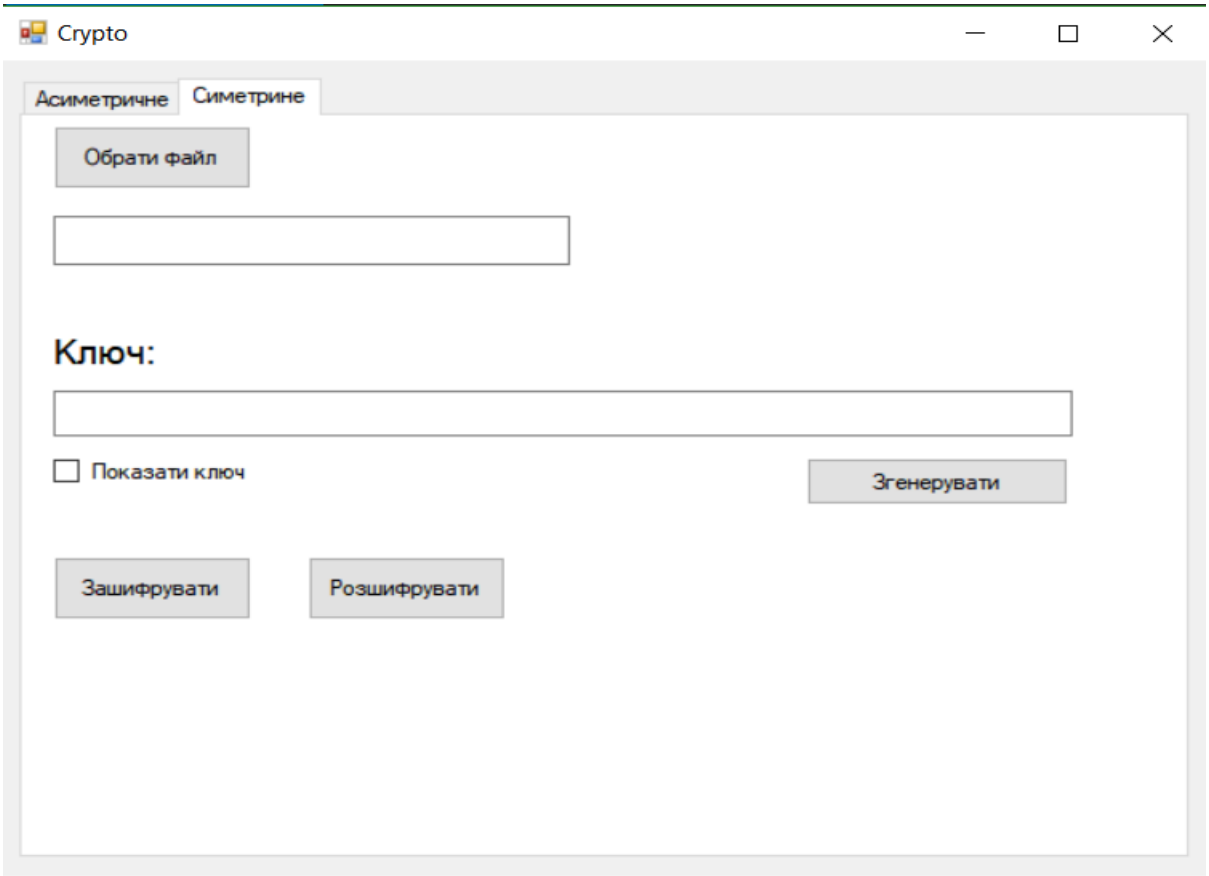


Рисунок 4.6 - Форма-приклад використання симетричного шифрування

На цій формі користувач може обрати файл, який хоче зашифрувати або розшифрувати. Кнопка «Згенерувати» служить для генерування ключа. Згенерований ключ записується у відповідне поле. При бажанні користувач моєе задати будь-який інший ключ. Кнопки «Зашифрувати» і «Розшифрувати» служать для відповідно шифрування та дешифрування файла. Після завершення шифрування виводитьяс повідомлення про успіх або помилку (Рис. 4.3).

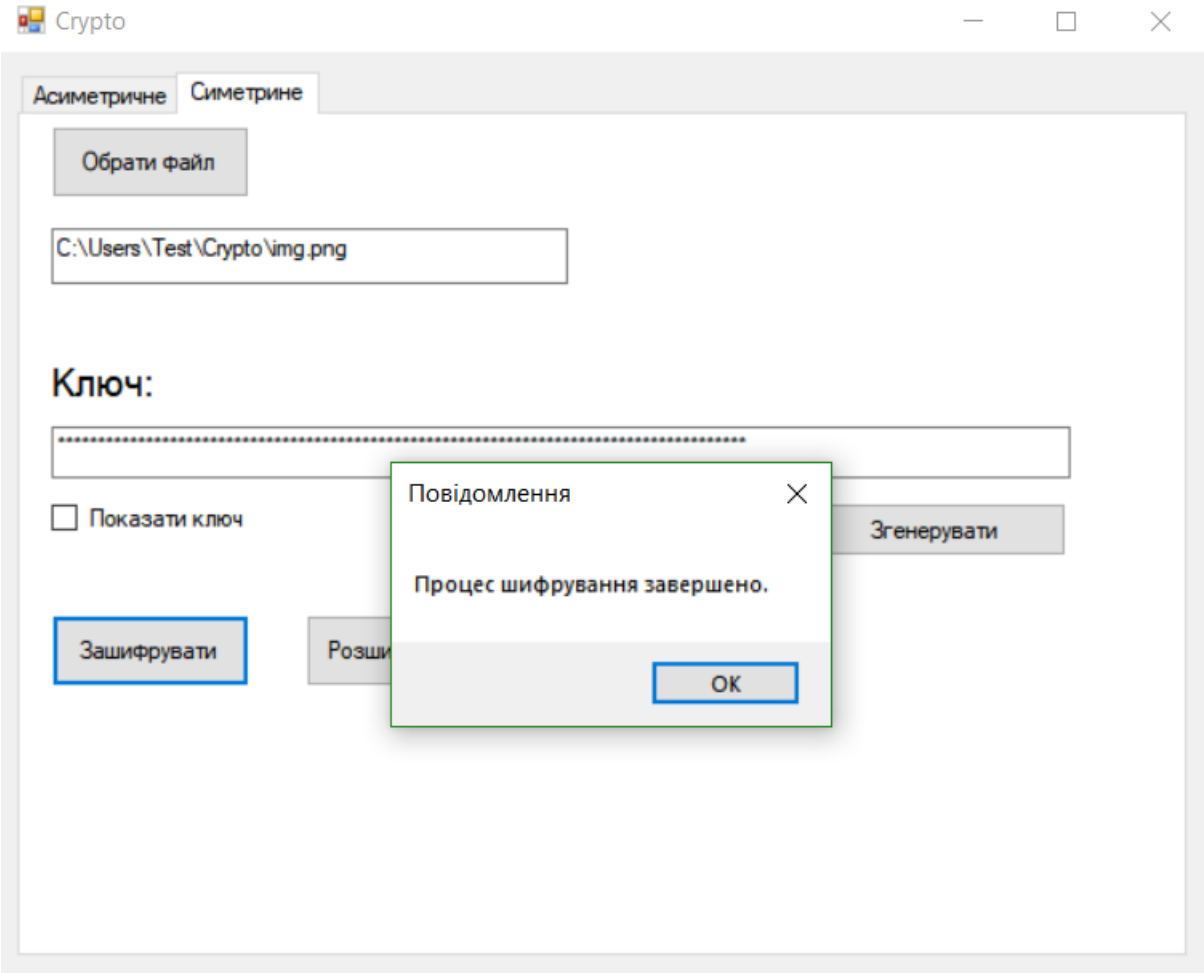


Рисунок - 4.7 – Результат шифрування або дешифрування

Друга вкладка показує приклад роботи з алгоритом RSA (Рис. 4.4). Кнопка «Згенерувати» дозволяє користувачу створити нові закритий та відкритй ключ з вказаною довжиною. «Обрати файл» дозволює завантажити файл за текстом у поле тексту. Користувач може також ввести текст вручну. Для того щоб зашифрувати, потрібно вибрати підпункт «Зашифрувати» цього програма вибере відповідний ключ для шифруваннння.

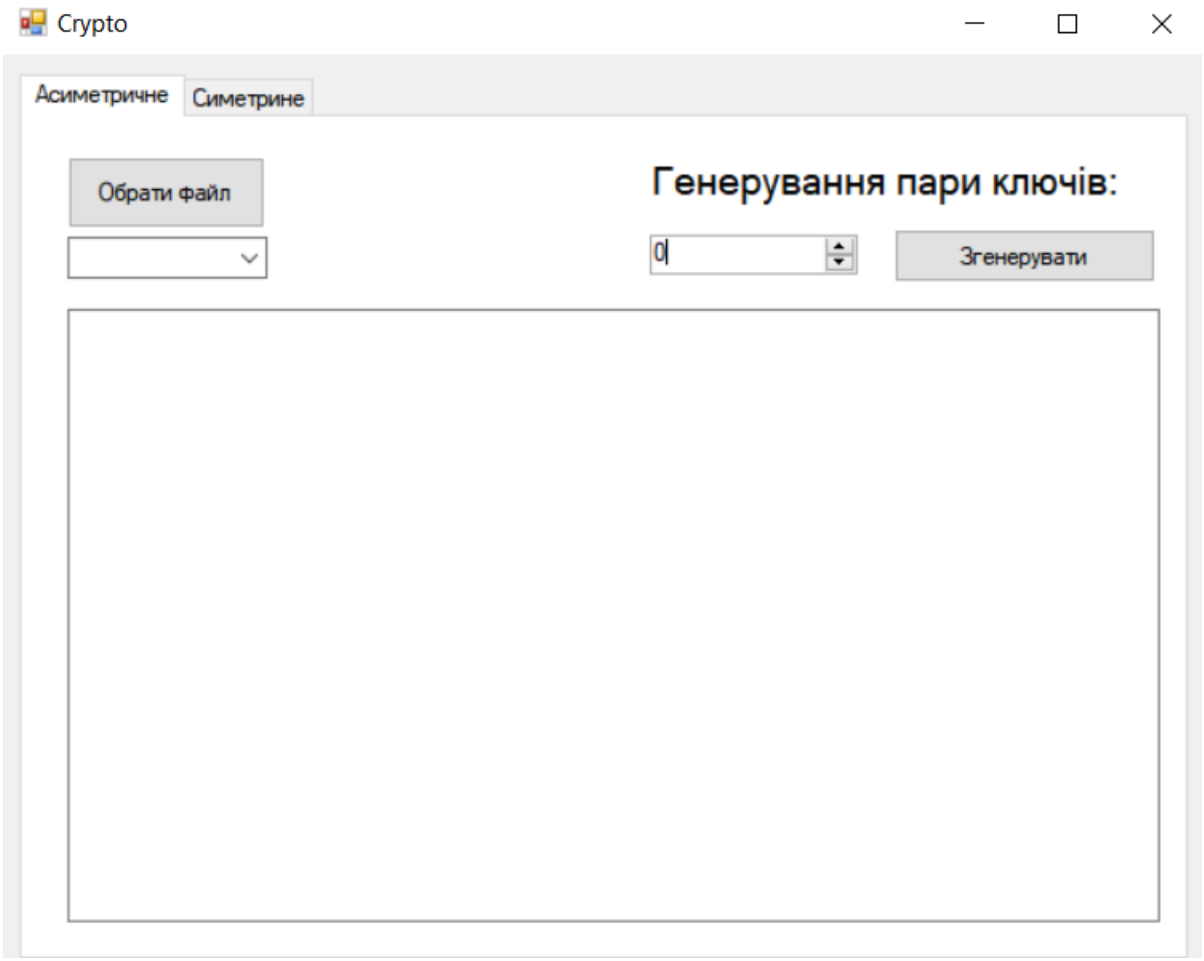


Рисунок 4.8 – Приклад шифрування алгоритмом RSA

Після вибору підпункту «Зашифрувати» відбувається шифрування тексту (Рис. 4.5). В результаті користувач отримає зашифрований текст, який побачить у полі вводу. Для розшифровки потрібно буде вибрати «Розшифрувати» та після цього програма вибере відповідний ключ та розшифрує дані.

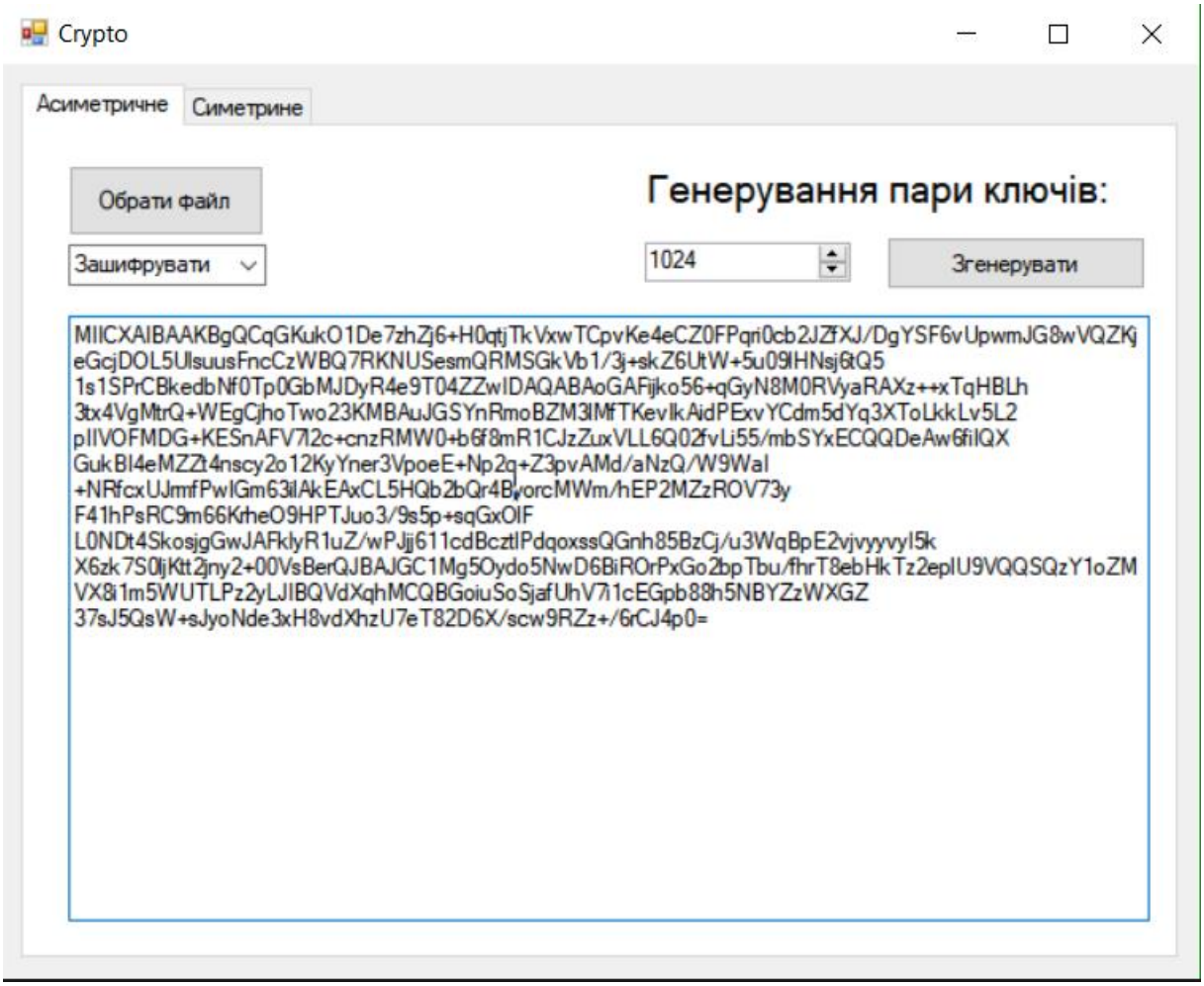


Рисунок 4.9 – Результат роботи алгоритма RSA

5. ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

5.1 Аналіз потенційних небезпечних і шкідливих виробничих чинників проектованого об’єкту, що мають вплив на персонал

У даному дипломному проекті розробляється програмне забезпечення.

Розроблене програмне забезпечення орієнтоване на роботу з персональним комп'ютером. Експлуатовані для вирішення внутрішньовиробничих завдань ПЕОМ типу IBM PC мають наступні характеристики:

споживана потужність	220 Вт;
робоча напруга	220 В;
напруга джерел живлення	+12 В; - 12 В; +5 В;
робоча частота	50 Гц.

Виходячи з приведених характеристик, вочевидь, що для людини існує небезпека поразки електричним струмом, унаслідок недбалого поводження з комп'ютером і порушення правил експлуатації, залишення частин ПЕОМ, що знаходяться під напругою, відкритими або знятих для ремонту вузлів.

Відповідно до [16] до легкої фізичної роботи відносяться всі види діяльності, виконувані сидячи і ті, що не потребують фізичної напруги. Робота користувача ПК відноситься до категорії 1а.

При роботі на ПЕОМ користувач піддається ряду потенційних небезпек. Унаслідок недотримання правив техніки безпеки при роботі з машиною(невиконання огляду відкритих частин ПЕОМ, що знаходяться під напругою або знятих для ремонту вузлів) для користувача існує небезпека поразки електричним струмом.

Джерелами підвищеної небезпеки можуть служити наступні елементи:

- розподільний щит;
- джерела живлення;
- блоки ПЕОМ і друку, що знаходяться в ремонті.

Ще одна проблема полягає у тому, що спектр випромінювання комп'ютерного монітора включає рентгенівську, ультрафіолетову і інфрачервону області, а також широкий діапазон хвиль інших частот. Небезпека рентгенівського проміння мала, оскільки цей вид випромінювання поглинається речовиною екрану. Проте велику увагу слід приділяти біологічним ефектам низькочастотних електромагнітних полів(аж до порушення ДНК).

Відповідно до [19], при обслуговуванні ПЕОМ мають місце фізичні і психофізичні небезпечні, а також шкідливі виробничі чинники:

- підвищене значення напруги в електричному ланцюзі, замикання якої може відбутися через тіло людини;
- підвищений рівень статичної електрики;
- підвищений рівень електромагнітних випромінювань;
- підвищена або знижена температура повітря робочої зони;
- підвищений або знижений рух повітря;
- підвищена або знижена вологість повітря;
- відсутність або недостатність природного світла;
- підвищена пульсація світлового потоку;
- недостатня освітленість робочого місця;
- підвищений рівень шуму на робочому місці;
- розумове перенапруження;
- емоційні навантаження;
- монотонність праці.

5.2 Заходи щодо техніки безпеки

Основним небезпечним чинником при роботі з ЕОМ є небезпека поразки людини електричним струмом, яка посилюється тим, що органи чуття людини не можуть на відстані знайти наявності електричної напруги на устаткуванні.

Проходячи через тіло людини, електричний струм чинить на нього складну дію, що є сукупністю термічної(нагрів тканин і біологічних середовищ), електролітичної(розкладання крові і плазми) і біологічної(роздратування і збудження нервових волокон і інших органів тканин організму) дій.

Тяжкість поразки людини електричним струмом залежить від цілого ряду чинників:

- значення сили струму;
- електричного опору тіла людини і тривалості протікання через нього струму;
- роду і частоти струму;
- індивідуальних властивостей людини і навколишнього середовища.

Розроблений дипломний проект передбачає наступні технічні способи і засоби, що застерігають людину від ураження електричним струмом [18]:

- заземлення електроустановок;
- занулення;
- захисне відключення;
- електричне розділення ятерів;
- використовування малої напруги;
- ізоляція частин, що проводять струм;
- огорожа електроустановок.

Занулення зменшує напругу дотику і обмежує година, протягом якого людина, ткнувшись до корпусу, може потрапити під дію напруги.

Струм однофазного короткого замикання визначається по наближеній формулі:

$$I_K = \frac{\text{Uф}}{\text{Zп} + \frac{\text{Zт}}{3}}\ , \qquad (5.1)$$

де Uф - номінальна фазна напруга мережі, В;

Zп - повний опір петлі, створене фазними і нульовими дротами, Ом;

Zт - повний опір струму короткого замикання на корпус, Ом.

Згідно таблиці 4 [18]: Zт /3 = 0,1 Ом.

Для провідників і жил кабелю для розрахунку повного опору петлі використовуємо формулу (5.2.):

$$Z\text{п} = \sqrt{\text{Rп}^2 + \text{Xп}^2}\ , \qquad (5.2)$$

де Rп = Rф + Ro - сумарний активний опір фазного Rф і нульового Ro дротів, Ом;

Xп - індуктивний опір паяння дротів, Ом.

Перетин 1 км мідного дроту S = 2.5 мм, тоді згідно таблицям 5 і 6 [19], має такий опір:

Xп = 0,11 Ом;

Rф = 7,55 Ом;

Ro = 7,55 Ом.

Отже, Rп = 7,55 + 7,55 = 15,1 Ом.

Тоді по формулі (4.2) знаходимо повний опір петлі :

$$Zп = \sqrt{15,1^2 + 0,11^2} \approx 15,1 \text{ (Ом)}.$$

Струм однофазного короткого замикання рівний:

$$Iк = \frac{220}{15,1 + 0,1} = 14,47 \text{ (А)}.$$

Дія плавкої вставки на ПЕОМ забезпечується, якщо виконується співвідношення:

$$Iк \geq k * Iн, \quad (5.3)$$

де Iн - номінальний струм спрацьовування плавкої вставки, А;
k - коефіцієнт кратності нелінійного струму Iн, А.

Коефіцієнт кратності нелінійного струму Iн розраховується по формулі (4.4.) :

$$Iн = P / U, \quad (5.4)$$

де P = 220 Вт - споживана потужність;
U = 220 В - робоча напруга;
k = 3 А - для плавких вставок.

Отже, Iн = 220 / 220 = 1 А.
Підставивши значення у вираз (5.3), одержимо:

$$14,47 > 3*1.$$

Таким чином, доведено, що апарат забезпечить спрацьовування(і захист) при підвищенні номінального струму.

5.3 Заходи, що забезпечують виробничу санітарію і гігієну праці

Вимоги до виробничих приміщень встановлюються [20], СНіП, відповідними ГОСТами і ОСТамі з урахуванням небезпечних і шкідливих чинників, що утворюються в процесі експлуатації електроустаткування.

Підвищення працездатності людини і збереження її здоров'я забезпечується стабільними метеорологічними умовами. Мікроклімат виробничих приміщень [21] визначається діючими на організм людини поєднаннями температури, вологості і швидкості руху повітря, а також температури навколишніх поверхонь. Значне коливання параметрів мікроклімату приводити до порушення систем кровообігу, нервової і потовидільної, що може викликати підвищення або пониження температури тіла, слабкість, запаморочення і навіть непритомність.

Відповідно до [22] встановлюють оптимальну і допустиму температуру, відносну вологість і швидкість руху повітря в робочій зоні . За відсутності надмірного тепла, вологи, шкідливих речовин в приміщенні досить природної вентиляції.

У приміщенні для виконання робіт операторського типу(категорія 1а), пов'язаних з нервово-емоційною напругою, проектом передбачається дотримання наступних нормованих величин параметрів мікроклімату (табл. 5.1).

Таблиця 5.1 - Санітарні норми мікроклімату робочої зони приміщень для робіт категорії 1а.

Пора року	Температура, С	Відносна вологість, %	Швидкість руху повітря, м/с
Холодна	22…24	40…60	0,1
Тепло	23…25	40…60	0,1

У приміщенні, де знаходиться ПЕОМ, повітрообмін реалізується за допомогою природної організованої вентиляції(з пристроєм вентиляційних каналів в перекриттях будівлі і вертикальних шахт) й установленого промислового кондиціонера фірми Mitsubishi, який дозволяє вирішити переважну більшість завдань по створінню та підтримці необхідних параметрів повітряного середовища. Цей метод забезпечує приток потрібної кількості свіжого повітря, визначеного в СНіП (30 $м^3$ в годину на одного працівника).

Шум на виробництві має шкідливу дію на організм людини. Стомлення операторів через шум збільшує число помилок при роботі, призводить до виникнення травм. Для

оператора ПЕОМ джерелом шуму є робота принтера. Щоб усунути це джерело шуму, використовують наступні методи. При покупці принтера слід вибирати найбільш шумозахисні матричні принтери або з великою швидкістю роботи(струменеві, лазерні). Рекомендується принтер поміщати в найбільш віддалене місце від персоналу, або застосувати звукоізоляцію та звукопоглинання(під принтер підкладають демпфуючі підкладки з пористих звукопоглинальних матеріалів з листів тонкої повсті, поролону, пеноплену).

При роботі на ПЕОМ, проектом передбачені наступні методи захисту від електромагнітного випромінювання : обмеження часом, відстанню, властивостями екрану.

Обмеження години роботи на ПЕОМ складає 3,5-4,5 години. Захист відстанню передбачає розміщення монітора на відстані 0,4-0,5 м від оператора. Передбачений монітор 20" TFT, Samsung 2043BW відповідає вимогам стандарту [23].

Стандарт [23] пред’являє жорсткі вимоги в таких областях: ергономіка(фізична, візуальна і зручність користування), енергія, випромінювання(електричних і магнітних полів), навколишнє середовище і екологія, а також пожежна та електрична безпека, які відповідають всім вимогам [24].

Для зниження стомлюваності та підвищення продуктивності праці обслуговуючого персоналу в колірній композиції інтер'єру приміщень для ПЕОМ дипломним проектом пропонується використовувати спокійні колірні поєднання і покриття, що не дають відблисків.

У проекті передбачається використання сумісного освітлення. У світлий час доби приміщення освітлюватиметься через віконні отвори, в решту часу використовуватиметься штучне освітлення.

Як штучне освітлення необхідно використовувати штучне робоче загальне освітлення. Для загального освітлення необхідно використовувати люмінесцентні лампи. Вони володіють наступними перевагами: високою світловою віддачею, тривалим терміном служби, хоча мають і недоліки: високу пульсацію світлового потоку.

При експлуатації ПЕОМ виробляється зорова робота. Відповідно до [25] ця робота відноситься до розряду 5а. При цьому нормоване освітлення на робочому місці(Ен) при загальному освітленні рівна 200 лк.

Приміщення завдовжки 12 м, шириною 10 м, заввишки 4 м обладнується світильниками типу ЛП02П, оснащеними лампами типу ЛБ зі світловим потоком 3120 лм кожна.

Виконаємо розрахунок кількості світильників в робочому приміщенні завдовжки a=12 м, шириною b=10 м, заввишки з=4 м, використовуючи формулу (5.5) розрахунку штучного освітлення при горизонтальній робочій поверхні методом світлового потоку:

$$n = (E \cdot S \cdot Z \cdot k)/(F \cdot U \cdot M), \quad (5.5)$$

де F - світловий потік = 3120 лм;

E - максимально допустима освітленість робочих поверхонь = 200 лк;

S - площа підлоги = 120 м2;

Z - поправочний коефіцієнт світильника = 1,2;

k - коефіцієнт запасу, що враховує зниження освітленості в процесі експлуатації світильників = 1,5;

n - кількість світильників;

U - коефіцієнт використовування освітлювальної установки = 0,6;

M - кількість ламп у світильнику =2.

Отже, $n = (200 \cdot 120 \cdot 1{,}2 \cdot 1{,}5)/(3120 \cdot 0{,}6 \cdot 2) = 12$.

Виходячи з цього, рекомендується використовувати 12 світильників. Світильники слід розміщувати рядами, бажано паралельно стіні з вікнами. Схема розташування світильників зображена на рис. 2.1.

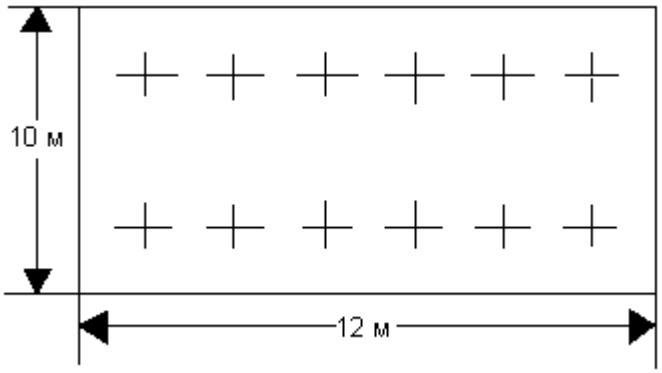


Рисунок 2.1 - Схема розташування світильників

5.4 Рекомендації по пожежній безпеці

Пожежі в приміщеннях, де встановлена обчислювальна техніка, представляють небезпеку для життя людини. Пожежі також пов’язані як з матеріальними втратами, так і з відмовою засобів обчислювальної техніки, що у свою чергу спричиняє за собою порушення ходу технологічного процесу.

Пожежа може виникнути при наявності горючої речовини та внесення джерела запалювання в горюче середовище. Пальними матеріалами в приміщеннях, де розташовані ПЕОМ, є:

− поліамід - матеріал корпусу мікросхеми, горюча речовина, температура самозаймання аерогелю 420 З ;

− полівінілхлорид - ізоляційний матеріал, горюча речовина, температура запалювання 335 З, температура самозаймання 530 З, кількість енергії, що виділяється при згоранні - 18000 - 20700 кДж/кг;

− стеклотекстоліт ДЦ - матеріал друкарських плат, важкозаймистий матеріал, показник горючості 1.74, не схильний до температурного самозаймання;

− пластика кабельний №489 - матеріал ізоляції кабелю, горючий матеріал, показник горючості більш 2.1;

− деревина - будівельний і обробний матеріал, матеріал з якого виготовлені меблі, горючий матеріал, показник горючості більше 2.1, теплота згорання 18731 - 20853 кДж/кг, температура запалювання 399 З, схильна до самозаймання [46].

Згідно [29] приміщення відносяться до категорії В(пожежовибухонебезпечним) і згідно правилам побудови електроустановок простір усередині приміщення відноситься до вогненебезпечної зони класу П - IIа (зони, розташовані в приміщеннях, в яких зберігаються тверді горючі речовини).

Потенційними джерелами запалення при роботі ПЕОМ є:

− іскри при замиканні і розмиканні ланцюгів;

− іскри і дуги коротких замикань;

− перегріви від тривалого перевантаження і наявності перехідного опору.

Продуктами згорання, що виділяються при пожежі, є : оксид вуглецю, сірчистий газ, оксид азоту, синильна кислота, акропеін, фосген, хлор та ін. При горінні пластмас, окрім звичайних продуктів згорання, виділяються різні продукти термічного розкладання: хлорангідрідні кислоти, формальдегіди, хлористий водень, фосген, синильна кислота, аміак, фенол, ацетон, стирол та ін., що шкідливо впливають на організм людини.

Для захисту персоналу від дії небезпечних і шкідливих чинників пожежі проектом передбачається застосування промислового протигаза з коробкою марки В(жовта).

Пожежна безпека об’єктів народного господарства регламентується [48] і забезпечується системами запобігання пожежам і протипожежному захисту. Для успішного гасіння пожеж вирішальне значення має швидке виявлення пожежі і своєчасний виклик пожежних підрозділів до місця пожежі.

Зменшити горюче навантаження не представляється можливим, тому проектом передбачається застосувати наступні способи і їх комбінації для запобігання утворенню(внесення) джерел запалення :

− застосування устаткування, що задовольняє вимогам електростатичної безпеки;

− застосування в конструкції швидкодіючих засобів захисного відключення можливих джерел запалення;

− виключення можливості появи іскрового заряду статичної електрики в горючому середовищі з енергією, рівної і вище мінімальної енергії запалення;

− підтримка температури нагріву поверхні машин, механізмів, устаткування, пристроїв, речовин і матеріалів, які можуть увійти до контакту з пальним середовищем, нижче гранично допустимої, становить 80% якнайменшої температури самозаймання пального.

− заміна небезпечних технологічних операцій більш безпечними;

− ізольоване розташування небезпечних технологічних установок і устаткування;

− зменшення кількості пальних і вибухонебезпечних речовин, що знаходяться у виробничих приміщеннях;

− запобігання можливості утворення пальних сумішей на лінії, вентиляційних системах і ін.;

− механізація, автоматизація та справність(потокова) виробництва;

− суворе дотримання стандартів і точне виконання встановленого технологічного режиму;

− запобігання можливості появи в небезпечних місцях джерел запалення;

− запобігання розповсюдженню пожеж і вибухів;

− використовування устаткування і пристроїв, при роботі яких не виникає джерел запалення;

− виконання вимог сумісного зберігання речовин і матеріалів;

− наявність громовідводу;

− організація автоматичного контролю параметрів, що визначають джерела запалення;

− ліквідація можливості самозаймання речовин і матеріалів .

Для запобігання пожежі в обчислювальних центрах проектом пропонується виконання наступних вимог :

− електроживлення ЕОМ повинно мати автоматичне блокування відключення електроенергії на випадок зупинки системи охолоджування і кондиціонування;

− система вентиляції обчислювальних центрів повинна бути обладнана блокуючими пристроями, що забезпечують її відключення на випадок пожежі;

− робочі місця повинні бути оснащені пожежними щитами, сигналізацією, засобами для сповіщення про пожежну небезпеку (телефонами), медичними аптечками для надання першої медичної допомоги, розробленим планом евакуації.

Для зниження пожежної небезпеки в приміщеннях використовуються первинні засоби гасіння пожеж, а також система автоматичної пожежної сигналізації, яка дозволяє знайти початкову стадію загоряння, швидко і точно оповістити службу пожежної охорони про час і місце виникнення пожежі.

Відповідно до [29] приміщення категорії В підлягають устаткуванню системами автоматичної пожежної сигналізації. Проектом передбачається застосування датчика типу ІДФ - 1(димовий фотоелектричний датчик), оскільки специфікою пожеж обчислювальної техніки і радіоапаратури є, в першу чергу, виділення диму, а потім - підвищення температури.

При виникненні пожежі в робочому приміщенні обслуговуючий персонал зобов'язаний негайно вжити заходи по ліквідації пожежі. Для ліквідації пожежі використовують вогнегасники (хімічно-пінні, пінні для повітря ОП-5, ОП-6, ОП-9, вуглекислотні ОУ-5), пісок, пожежний інвентар(сокири, ломи, багри, шерстяну або азбестову ковдри) [28]. Як засіб індивідуального захисту проектом передбачається використовування промислового протигаза з маскою, фільтруючої коробки В.

В якості організаційно-технічних заходів рекомендується проводити навчання робочого персоналу правилам пожежної безпеки.

5.5 Охорона навколишнього природного середовища

Діяльність за темою магістерської роботи, а саме розробці автоматизованої системи моделювання рівноважного складу впливає на навколишнє природнє середовище і регламентується нормами діючого законодавства [30 - 32].

Основним екологічним аспектом в процесі діяльності за даними спеціальностями є процеси впливу на атмосферне повітря та процеси поводження з відходами, які утворюються, збираються, розміщуються, передаються на відалення (знешкодження), утилізацію, тощо в ІТ галузі.

Вплив на атмосферне повітря при нормальних умовах праці не оказує, бо не має в приміщенні сканерів, принтерів та інших джерел викиду забруднюючих речовин в повітря робочої зони.

В процесі створення/розробки програми на робочому місці виникають процеси поводження з відходами ІТ галузі. Нижче надано перелік відходів, що утворюються в процесі роботи:

- відпрацьовані люмінесцентні лампи - I клас небезпеки
- змінні носії інформації - IV клас небезпеки
- відпрацьовані вогнегасники - IV клас небезпеки
- макулатура - IV клас небезпеки
- відпрацьовані фільтрувальні засоби індивід. захисту (респіратори, протигази) - IV клас небезпеки
- побутові відходи - IV клас небезпеки

ВИСНОВКИ

У ході виконання атестаційної роботи були визначені основні функції сучасних веб-застосунків, розглянуті найбільш популярні технології для побудови сучасних веб-застосунків. Так само, були виявлені і проаналізовані ефективніші способи створення веб-застосунків.

Так само було проведено огляд основних принципів побудови веб-застосунків. Була проведена порівняльна характеристика технологій для побудови сучасних веб-застосунків. На основі проведеного аналізу, було зроблено висновок про використання мови C#, а так само супутньої технології ASP.NET Core MVC в якості мови для створення веб-застосунків.

Були розглянуті різні види веб-застосунків для шифрування даних, їх позитивні та негативні сторони, їх потрібність у суспільстві та загальна характеристика веб-застосунків шифрування погодних умов.

Кажучи про шифрування даних, слід говорити про створення єдиного інформаційного простору, куди слід включити всілякі електронні джерела інформації (включаючи мережеві): віртуальні бібліотеки, бази даних, консультаційні служби, де дані користувача повинні бути захищені, бо в сучасному світі багото прикладів, коли великі компанії піддавалися кібер-атакам та дані мільйонів користувачів були втрачені або до них мали доступ злодії. В ході цієї роботи була реалізована система для шифрування даних, яка дозволятиме користувачам шифрувати дані різними видами шифрування.

Було проаналізовано декілька технологій для створення веб-застосунків, що задовольняли потреби сучасного веб-застосунку.

Проаналізувавши принцип роботи веб-застосунків та методи для ефективного шифрування даних була створена система, здатна забезпечити належний рівень безпеки, з зручним та простим для користування інтерфейсом.

Комплексне створення та використання веб-застосунків для шифрування даних рекомендується застосовувати у багатьох галузях.

У розділі «Охорона праці та безпека в надзвичайних ситуаціях» виконано аналіз потенційних небезпек при роботі із засобами обчислювальної техніки і механізмами, розроблені заходи щодо техніки безпеки, заходи, які забезпечують виробничу санітарію і гігієну праці, охорону навколишнього природного середовища та розраховане штучне освітлення, виконані рекомендації по пожежній безпеці.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1) Бьюли. А. Изучаем SQL. Учебное пособие [Текст] / А. Бьюли – Символ-Плюс, 2007. – 312 с.

2) Флэнаган. Д. JavaScript. Подробное руководство. Учебное пособие [Текст] / Д. Флэнаган. – Символ-Плюс, 2009. – 992 с

3) Шилдт Герберт. C# 4.0 полное руководство [Текст] / Шилдт Герберт – Вильямс, 2015 – 1056 с.

4) Д. Вебер. Технология Java в подлиннике [Текст] / Д. Вебер – БХВ – Петербург, 2001. – 1104 с.

5) Уилл Айверсон. Популярные Web-сервисы: практика использования [Текст] / Уилл Айверсон. Издательство «Кудиц-Образ», 2005 – 547с.

6) Билл Кеннеди, Чак Муссиано. HTML и XHTML. Подробное руководство (HTML & HXTML. The Definitive Guide) [Текст] / Билл Кеннеди, Чак Муссиано – Символ-Плюс, 201. – 752 с.

7) Эрик Мейер. CSS-каскадные таблицы стилей. Подробное руководство (Cascading Style Sheets: The Definitive Guide) [Текст] / Эрик Мейер – Символ-Плюс, 2008 – 576 с.

8) Б. Хеник. HTML и CSS. Путь к совершенству (HTML и CSS: The Good Parts) [Текст] / Б. Хеник – O'Reilly Media, 2011. – 336 с.

9) Дэвид Флэнаган. JavaScript. Подробное руководство (JavaScript. The Definitive Guide) [Текст] / Дэвид Флэнаган – O'Reilly Media, 2011. – 1096 с.

10) Джеффри Рихтер. CLR via C#. Программирование на платформе Microsoft.NET Framework 4.5 на языке C# [Текст] /Джеффри Рихтер – Питер, 2000. – 896 с.

11) Адам Фримен. ASP.NET MVC 4 [Текст] / Адам Фримен – Вильямс, 2013. – 688с.

12) Дино Эспозито. Разработка веб-приложений с использованием ASP.NET и AJAX [Текст] / Дино Эспозито – Питер, 2012. – 400с.

13) Пабло Дилеман. Изучаем Angular [Текст] / Пабло Дилеман – ДМК Пресс, 2017 – 354 с.

14) Гэри Маклин Холл. Проектирование классов и интерфейсов, шаблоны и принципы SOLID [Текст] / Гэри Маклин Холл – Вильямс, 2015. –432с.

15) Ньюкомер, Э. Веб-сервисы. XML, WSDL, SOAP и UDDI. Для профессионалов [Текст] / Э. Ньюкомер – Издательский дома «Питер», 2003. – 256 с.

16) ГОСТ 12.0.003-74 Опасные и вредные производственные факторы. Классификация. Режим доступу: www. URL: http://online.budstandart.com/ua/catalog/doc-page.html?id_doc=48127

17) НПАОП 40.1-1.21-98 «Правила безпечної експлуатації електроустановок споживачів». Затверджено Наказом Держнаглядохоронприці України № 4 від 09.01.98 – Режим доступу: www. URL: https://zakon.rada.gov.ua/laws/show/z0093-98

18) НПАОП 40.1-1.32-01 Правила устройства электроустановок. Электрооборудование специальных установок. Наказ №272 від 21.06.2001. Режим доступу: www. URL: https://dnaop.com/html/1692/doc-%D0%9D%D0%9F%D0%90%D0%9E%D0%9F_40.1-1.32-01

19) НПАОП 0.00-7.15-18 «Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями». Зареєстровано в Міністерстві юстиції України 25 квітня 2018 р. за № 508/31960. Режим доступу: www. URL: https://zakon.rada.gov.ua/laws/show/z0508-18

20) ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень». Вводиться в дію Постановою ВР № 42 від 01.12.1999. – Режим доступу: www. URL: https://zakon.rada.gov.ua/rada/show/va042282-99

21) ГОСТ 12.1.005-88. Система стандартов безопасности труда. Общие санитарно-гигиенические требования к воздуху рабочей зоны. – Режим доступу: www. URL: http://online.budstandart.com/ua/catalog/doc-page.html?id_doc=6264

22) ДСН 3.3.6.037-99 Санітарні норми виробничого шуму, ультразвуку та інфразвуку. Постанова N 37 від 01.12.99. Режим доступу: www. URL: https://zakon.rada.gov.ua/rada/show/va037282-99

23) ДСанПіН 3.3.2.007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин». Вводиться в дію Постановою ВР № 7 від 10.12.1998. – Режим доступу: www. URL: https://zakon.rada.gov.ua/rada/show/v0007282-98

24) ДБН В.2.5-28:2018 «Природне і штучне освітлення». – Режим доступу: www. URL: http://www.minregion.gov.ua/wp-content/uploads/2018/12/V2528-1.pdf ДБН В.2.5-28:2018 «Природне і штучне освітлення». – Режим доступу: www. URL: http://www.minregion.gov.ua/wp-content/uploads/2018/12/V2528-1.pdf

25) ГОСТ 12.1.044-89 Система стандартов безопасности труда. Пожаровзрывоопасность веществ и материалов. Номенклатура показателей и методы их определения. – Режим доступу: www. URL: http://online.budstandart.com/ua/catalog/doc-page.html?id_doc=51048

26) ДСТУ Б В.1.1-36:2016 «Визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою». Наказ від 15.06.2016 №158. Режим доступу: www. URL: https://zakon.rada.gov.ua/rada/show/v0158858-16

27) ГОСТ 12.1.004-91. "Система стандартов безопасности труда. Пожарная безопасность. Общие требования". - Режим доступу: www. URL: http://online.budstandart.com/ua/catalog/doc-page.html?id_doc=48679

28) НАПБ А.01.001-2014 “Правила пожежної безпеки в Україні” Наказ Міністерство внутрішніх справ України (МВС України) від 30.12.2014 № 1417.- Режим доступу: www. URL: https://zakon.rada.gov.ua/laws/show/z0252-15

29) ДСТУ Б В.1.1-36:2016 Визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою. Наказ від 15.06.2016 № 158. Режим доступу: www. URL: http://online.budstandart.com/ua/catalog/doc-page.html?id_doc=65419

30) Закони України «Про охорону навколишнього природного середовища». Вводиться в дію Постановою ВР № 4005-XII від 24.02.94, ВВР, 1994, № 27, ст.219. Режим доступу: www. URL: https://zakon.rada.gov.ua/laws/show/4004-12

31) Закон України «Про забезпечення санітарного та епідемічного благополуччя населення». Відомості Верховної Ради України (ВВР), 1994, № 27, ст.218. Режим доступу: www. URL: https://zakon.rada.gov.ua/laws/show/4004-12

32) Закон України «Про відходи». Відомості Верховної Ради України (ВВР), 1998, № 36-37, ст.242. Режим доступу: www. URL: https://zakon.rada.gov.ua/laws/show/187/98-вр

ДОДАТОК А.

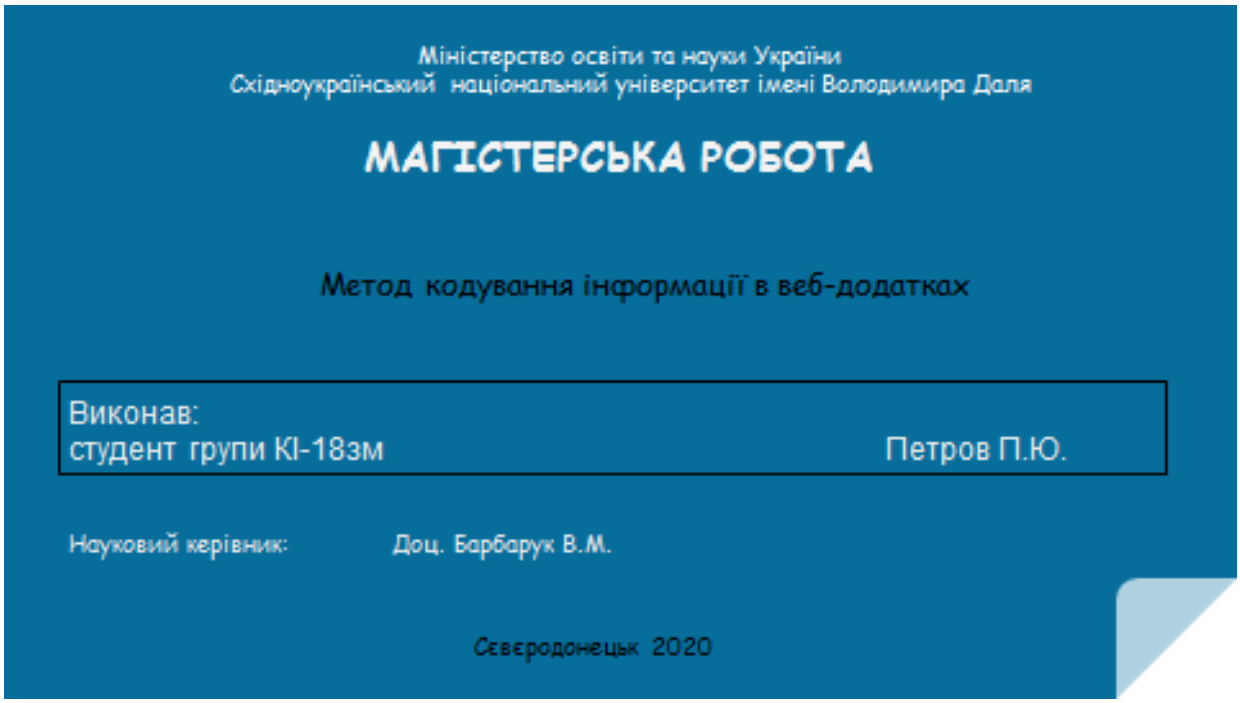
Фрагмент лістингу коду

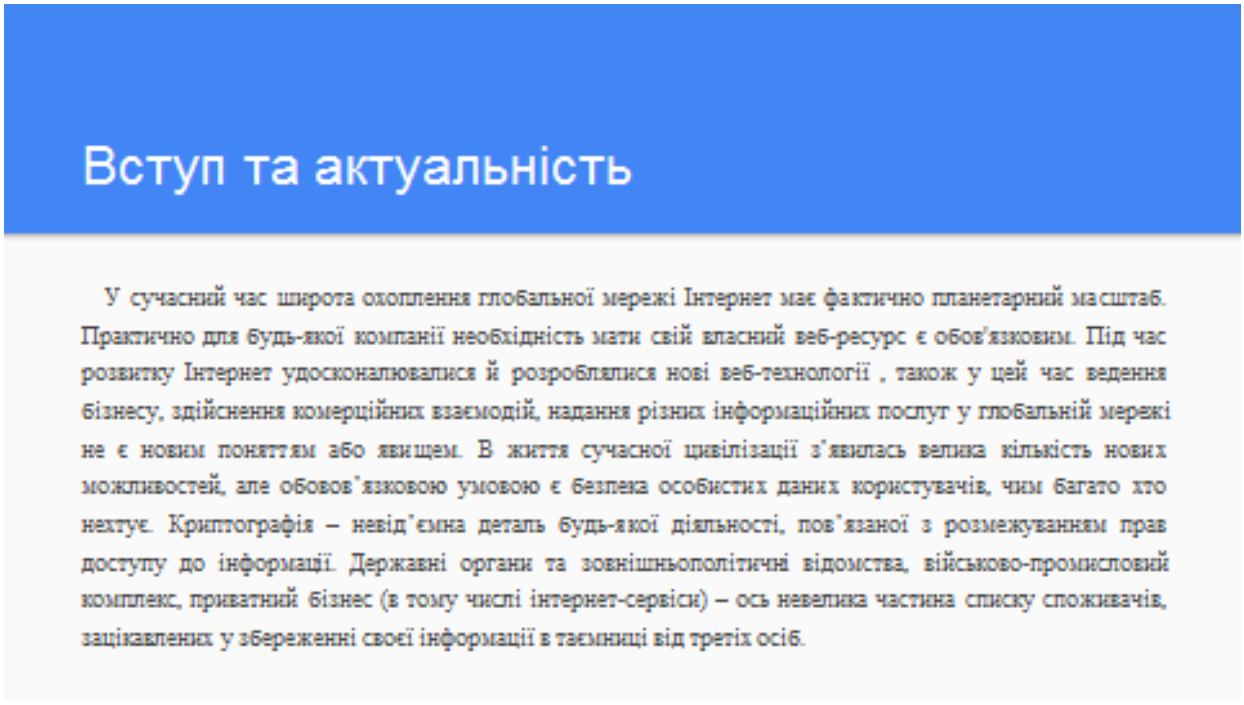
```
namespace ConsoleApplication2
{
    class Program
     {
         static void Main(string[] args)
         {
             Console.Title = "Symmetric Algorithms
Cryptography";
             Console.WriteLine("Enter File Path:");
             String path = Console.ReadLine();
             List<String> algorithms = new List<String>();
             algorithms.Add("DES");
             algorithms.Add("TripleDES");
             algorithms.Add("RC2");
             algorithms.Add("Rijndael");
             foreach (String algorithm in algorithms)
             {
                 SymmetricAlgorithm symmetricAlgorithm =
SymmetricAlgorithm.Create(algorithm);
                 symmetricAlgorithm.GenerateKey();
                 symmetricAlgorithm.GenerateIV();


                 String outPath = Path.GetDirectoryName(path) +
@"" +
                     Path.GetFileNameWithoutExtension(path) +
"_" + algorithm;
                 String encryptedPath = outPath + ".encrypted";
                 String decryptedPath = outPath + ".decrypted";
                 Encrypt(symmetricAlgorithm, path,
encryptedPath);
                 Decrypt(symmetricAlgorithm, encryptedPath,
decryptedPath);
                 Console.WriteLine(algorithm + " Done!");
             }
             Console.ReadLine();
         }
         private static void Encrypt(SymmetricAlgorithm
symmetricAlgorithm,
             String inPath, String outPath)
         {
             FileStream outStream = File.OpenWrite(outPath);
             ICryptoTransform transform =
symmetricAlgorithm.CreateEncryptor();
             CryptoStream cryptoStream = new
CryptoStream(outStream, transform, CryptoStreamMode.Write);
             Byte[] inFile = File.ReadAllBytes(inPath);
```

```
            cryptoStream.Write(inFile, 0, inFile.Length);
            cryptoStream.Close();
        }
        private static void Decrypt(SymmetricAlgorithm
symmetricAlgorithm,
            String inPath, String outPath)
        {
            ICryptoTransform transform =
symmetricAlgorithm.CreateDecryptor();
            Stream inStream = File.OpenRead(inPath);
            CryptoStream cryptoStream = new
CryptoStream(inStream, transform, CryptoStreamMode.Read);
            Byte[] buffer = new Byte[50];
            int length = cryptoStream.Read(buffer, 0,
buffer.Length);
            Stream outStream = File.OpenWrite(outPath);
            while (length > 0)
            {
                outStream.Write(buffer, 0, length);
                length = cryptoStream.Read(buffer, 0,
buffer.Length);

            }
            inStream.Close();
            outStream.Close();
        }
    }
 }
```

ДОДАТОК Б.

Електронні плакати





Чому варто використовувати шифрування даних?

Сьогодні головний ресурс за який змагаються багато великих компаній – це дані користувачів. Є багато прикладів коли компанії купували лише заради їх клієнтської бази даних. І тому забезпечення сохранності цих даних - першочергова задача у сучасних веб-застосунках, які зберігають дані користувачів. Такими даними є імейл, номер телефону, дані кредитних карток, прив'язаних до акаунтів користувачів, паролі і т.д..

Наразі багато злочинників, які намагаються заволодіти цими даними в своїх цілях, тому дуже важливо захищати ці дані.

Web-застосунки для шифрування можуть надати можливість :

- зашифрувати будь-які дані або файли, до яких має бути доступ тільки у користувача;
- отримати певну статистикику, виходячи з дій користувача;

Мета атестаційної роботи

- Дослідити сучасний стан шифрування даних, область застосування та способи методи шифрування файлів або даних.
- Аналіз сучасних платформ для побудови web-застосунків, способи створення, дослідження концепній проектування застосунків.
- Створити веб-застосунок для шифрування даних або файлів

Розвиток і можливості шифрування даних

Історія криптографії бере свій початок з давніх-давен. Це одна з найстаріших наук, її історія налічує кілька тисяч років. Перший період (приблизно з 3-го тисячоліття до н.е..) характеризується пануванням моноалфавітної шифрів (основний принцип - заміна алфавіту вихідного тексту іншим алфавітом через заміну букв іншими буквами або символами). Другий період (хронологічні рамки - з IX століття на Близькому Сході (Ал-Кінді) і з XV століття в Європі (Леон Баттіста Альберті) - до початку XX століття) ознаменувався введенням в ужиток поліалфавітних шифрів. Четвертий період - з середини до 70-х років XX століття - період переходу до математичної криптографії. Сучасний період розвитку криптографії (з кінця 1970-х років по теперішній час) відрізняється зародженням та розвитком нового напрямку - криптографія з відкритим ключем. Її поява знаменується не тільки новими технічними можливостями, а й порівняно широким поширенням криптографії для використання приватними особами.

Сучасна криптографія утворює окремий науковий напрям на стику математики та інформатики - роботи в цій області публікуються в наукових журналах, організовуються регулярні конференції.

Практичне застосування криптографії стало невід'ємною частиною життя сучасного суспільства - її використовують в таких галузях як електронна комерція(шифрування паролів, даних кредитної картки і т.д.), електронний документообіг (включаючи цифрові підписи), телекомунікації, розробка програмного забезпечення (обфускація програмного коду) та інших.

Випадки великих витоків даних користувачів

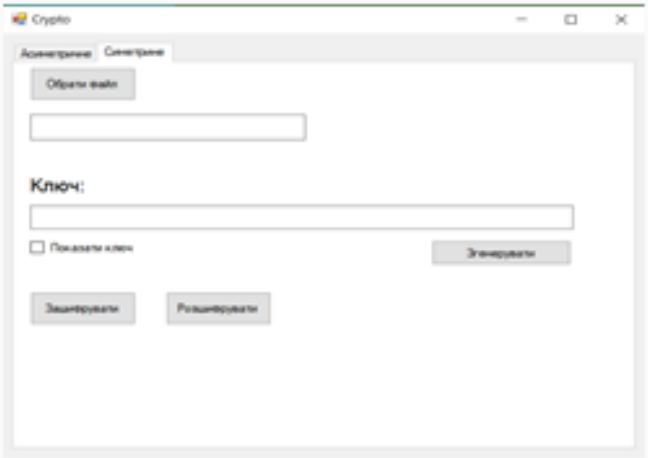
- У листопаді 2017 р. з'ясувалося, що **Uber** більше року приховував інформацію про крадіжку даних своїх клієнтів. Хакерська атака сталася в жовтні 2016 р. Хакери отримали імена, адреси електронної пошти та номери телефонів близько 57 млн користувачів і водіїв сервісу по всьому світу. Це величезна цифра, враховуючи кількість клієнтів по всьому світу. Також було викрадено приблизно 600 000 номерів водійських посвідчень.
- Компанія **CareFirst** BlueCross BlueShield, яка займається послугами медичного страхування в штаті Меріленд, Вірджинія і окрузі Колумбія, заявила, що скомпрометовані дані 1,1 мільйона колишніх і нинішніх клієнтів компанії. CareFirst, яка має в цілому 3,4 мільйона клієнтів, повідомила, що порушення мало місце в червні 2014 року, але виявлено було тільки в середині 2015 року.
- У Сінгапурі хакери атакували комп'ютери найбільшої в країні групи медичних установ SingHealth. Зловмисникам вдалося викрасти персональні дані 1,5 млн осіб, в тому числі кількох високопоставлених чиновників, випливає з прес-релізу Міністерства охорони здоров'я Сінгапуру в липні 2018 року.

Вибір інструмента для розробки застосунка

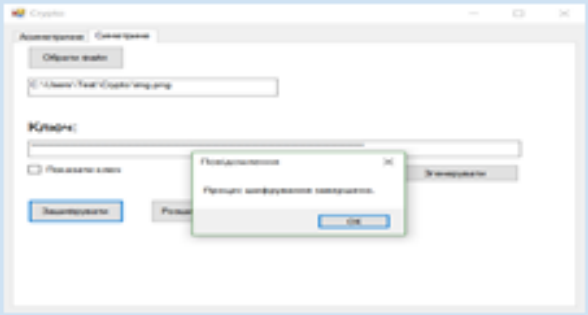
Як середовище розробки веб-застосунку було вибрано Microsoft Visual Studio 2017. Це продукт фірми Microsoft, що включає інтегроване середовище розробки програмного забезпечення та ряд інших інструментальних засобів. Ці продукти дозволяють розробляти як консольні програми, так і програми з графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms, а також веб-сайти, веб-додатки, веб-служби як в рідному, так і в керованому кодах для всіх платформ, що підтримуються Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, .NET Compact Framework та Microsoft Silverlight.

Visual Studio включає в себе редактор вихідного коду з підтримкою технології IntelliSense і можливістю найпростішого рефакторінга коду. Вбудований відладчик може працювати як відладчик рівня вихідного коду, так і відладчик машинного рівня. Решта вбудовуються інструменти включають в себе редактор форм для спрощення створення графічного інтерфейсу застосунку, веб-редактор, дизайнер класів і дизайнер схеми бази даних. Visual Studio дозволяє створювати і підключати сторонні додатки (плагіни) для розширення функціональності практично на кожному рівні, включаючи додавання підтримки систем контролю версій вихідного коду.

Приклад використання веб-застосунку у програмі шифрування даних

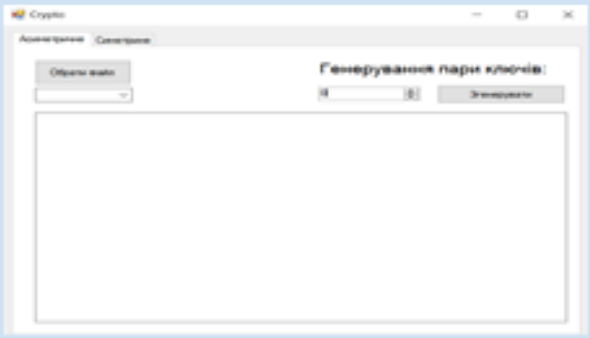


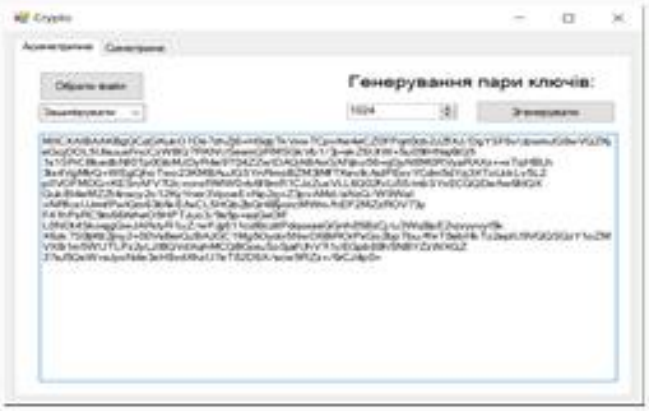
Приклад використання симетричного шифрування



Результат роботи симетричного шифрування

Приклад використання симетричного шифрування





Результат роботи асиметричного алгоритму RSA на прикладі шифрування тексту

Висновки

- У ході виконання дипломної роботи були визначені основні функції сучасних веб-застосунків, розглянуті найбільш популярні технології для побудови сучасних веб-застосунків.
- Були виявлені і проаналізовані ефективніші способи створення веб-застосунків. Так само було проведено огляд основних принципів побудови веб-застосунків. Була проведена порівняльна характеристика технологій для побудови сучасних веб-застосунків. На основі проведеного аналізу, було зроблено висновок про використання мови C#, а також супутньої технології ASP.NET Core MVC в якості мови для створення веб-застосунків.
- Була реалізована система для шифрування даних, яка дозволятиме користувачам використовувати її для інтеграції у будь-яку систему, де потребується шифрування даних.

Дякую за увагу