МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
Факультет кібербезпеки, комп'ютерної та програмної інженерії
Кафедра комп'ютерних інформаційних технологій

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри
_____Савченко А.С.

"____" _____2020 р.

ДИПЛОМНА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ "МАГІСТРА"

ЗА СПЕЦІАЛІЗАЦІЄЮ "ІНФОРМАЦІЙНІ УПРАВЛЯЮЧІ СИСТЕМИ ТА ТЕХНОЛОГІЇ (ЗА ГАЛУЗЯМИ)"

Тема: "Web-сервіс сховища файлів та безпечного обміну ними в мережі Інтернет"

Виконавець: Поліщук Володимир Леонідович

Керівник: к.т.н., доцент Райчев Ігор Едуардович

Нормоконтролер: _____ Райчев І.Е.

Київ 2020

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії

Кафедра Комп'ютерних інформаційних технологій

Галузь знань, спеціальність, спеціалізація: 12 "Інформаційні технології", 122 "Комп'ютерні науки", "Інформаційні управляючі системи та технології (за . галузями)"

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____Савченко А.С.
«____»_____2020 р.

ЗАВДАННЯ

на виконання дипломної роботи студента

Поліщука Володимира Леонідовича
(прізвище, ім'я, по батькові)

1. Тема роботи: "Web-сервіс сховища файлів та безпечного обміну ними в мережі Інтернет"

Затверджена наказом ректора від " " за № .

2. Термін виконання роботи: з 2019р. до лютого 2020р.

3. Вихідні дані до роботи: огляд існуючих рішень сховищ даних, розроблених на базі хмарних технологій. Новітні концепції криптографії та захисту інформації в мережі Інтернет.

4. Зміст пояснювальної записки: Титульний аркуш; Завдання на виконання дипломної роботи; Реферат; Зміст; Вступ; Розділ 1; Розділ 2; Розділ 3; Висновки; Список використаних джерел; Додаток А.

5. Перелік обов'язкового графічного матеріалу: інформативні пояснювальні рисунки, діаграми, презентація в *MS PowerPoint*.

6. Календарний план-графік

№ пор.	*Завдання*	*Термін виконання*	*Підпис керівника*
1.	Розроблення та затвердження календарного плану виконання дипломної роботи	14.10.2019	
2.	Підбір і вивчення літературних та інших джерел	15.10.2019-22.10.2019	
3.	Проведення консультацій з науковим керівником щодо виконання дипломної роботи	14.10.2019-09.02.2020	
4.	Підготовка та оформлення матеріалу за розділами 1, 2	23.10.2019-11.11.2019	
5.	Підготовка та оформлення матеріалу за розділом 3	12.11.2019-30.11.2019	
6.	Проведення досліджень та опрацювання їх результатів	01.12.2019 31.12.2019	
7.	Оформлення пояснювальної записки та ілюстративного матеріалу	02.01.2020-28.01.2020	
8.	Підготовка до захисту та попередній захист дипломної роботи на випусковій кафедрі	29.01.2020-31.01.2020	
9.	Підписання необхідних документів у встановленому порядку та підготовка до захисту дипломної роботи в ЕК	31.01.2020	

7. Дата видачі завдання: 14.10.2019 р.

Керівник дипломної роботи _____ Райчев Ігор Едуардович
(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання _____Поліщук Володимир Леонідович
(підпис випускника) (П.І.Б.)

РЕФЕРАТ

Магістерська дипломна робота «Web-сервіс сховища файлів та безпечного обміну ними в мережі Інтернет»: містить 95 сторінок, 25 рисунків, 11 літературних джерел, 1 додаток.

Об’єкт дослідження: програмне забезпечення захисту і передачі інформації.

Мета роботи: розробка програмного рішення безпечної передачі та захисту файлів.

Метод дослідження: дослідження існуючих рішень для обміну файлами та їх захисту.

Результати магістерської роботи: дослідження алгоритмів шифрування та хешування використані в реалізації застосунку. Створений додаток в певній можна рахувати хмарним рішенням, адже файли користувачів зберігатимуться віддалено, а не на локальному комп’ютері.

У процесі роботи над дипломним проектом, було проведено аналіз і досліджено існуючі методи захисту інформації. Розроблено сервіс для захисту, зберігання та передачі файлів.

ЗАХИСТ ІНФОРАМЦІЇ, БЕЗПЕЧНИЙ ОБМІН ДАНИМИ, ХМАРНІ ТЕХНОЛОГІЇ, ЗБЕРІГАННЯ ІНФОРМАЦІЇ, ХЕШУВАННЯ ДАНИХ, КРИПТОГРАФІЯ.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

КІТ	—	Комп’ютерні інформаційні технології
ІБ	—	Інформаційна безпека
ІС	—	Інформаційна система
ЗІ	—	Захист інформації
КбА	—	Кібернетична атака
КСЗІ	—	Комплексна система захисту інформації
ХТ	—	Хмарні технології
DES	—	Data Encryption Standard
AES	—	Advanced Encryption Standard

ВСТУП

З самого початку розвитку системи інформаційної безпеки (ІБ) розроблялися для військових відомств. Розголошення такої інформації могло привести до величезних жертв, у тому числі і людських. Тому конфіденційності (тобто нерозголошенню інформації) у системах безпеки приділялася особлива увага. Вважалося, що надійно захистити повідомлення і дані від перехоплення може тільки повне їхнє шифрування. Очевидно, через це початковий етап розвитку комп'ютерної безпеки міцно пов'язаний із крипто-шифрами. Однак сьогодні інформація має вже не настільки «убивчу» силу, і завдання збереження її в секреті втратило колишню актуальність. Зараз головні умови безпеки інформації — її доступність і цілісність.

Доступність інформації означає, що будь-який файл або ресурс системи має бути доступний у будь-який час (за дотримання прав доступу). Якщо якийсь ресурс недоступний, то він є марним.

Цілісність інформації означає, що завдяки захисту інформації забезпечується незмінність інформації під час її збереження або передачі.

Конфіденційність інформації, що забезпечується криптографією (шифруванням), не є головною вимогою при проектуванні захисних систем. Система безпеки повинна в першу чергу гарантувати доступність і цілісність інформації, а потім вже (якщо необхідно) її конфіденційність.

Принцип сучасного захисту інформації можна виразити таким чином — пошук оптимального співвідношення між доступністю і безпекою.

Метою дипломного проекту є оцінка та аналіз існуючих методів та підходів у сфері захисту та передачі файлів і як результат реалізація програмного рішення, що задовольняє вимоги щодо безпеки та доступності інформації сучасних користувачів мережі Інтернет.

РОЗДІЛ 1

Хмарні сховища даних

1.1. Характеристика та огляд хмарних технологій

Хмарні технології — це парадигма, що передбачає віддалену обробку та зберігання даних. Ця технологія надає користувачам мережі Інтернет, доступ до комп'ютерних ресурсів сервера і використання програмного забезпечення як онлайн-сервіса. Тобто якщо є підключення до Інтернету то можна виконувати складні обчислення, опрацьовувати дані використовуючи потужності віддаленого сервера.

Поява ХТ стала можливою у процесі розвитку технологій хмарних обчислень (англ. Cloud Computing), які реалізуються за умов динамічного масштабного доступу до розподілених зовнішніх мережевих ресурсів. Надання такого доступу, як відокремлена послуга, залишається різновидом ХТ.

ХТ зазвичай здійснюються в мережі Інтернет за допомогою сучасних інтернет-браузерів. Для реалізації ХТ використовують віртуальні машини, що функціонують у великих дата-центрах і замінюють собою фізичні персональні комп’ютери (ПК) та сервери. Головна відмінність від звичайного використання програмного забезпечення в ХТ полягає в тому, що користувач може поєднувати внутрішні ресурси свого комп’ютерного пристрою та програмні ресурси, які надаються йому як інтернет-сервіс. При цьому він має повний доступ до управління власними даними, але не може управляти операційною системою чи програмною базою, за допомогою яких ця робота відбувається.

ХТ мають цілу низку переваг: користувач може задіяти віртуальний комп’ютер практично будь-якої конфігурації для виконання ресурсоємних завдань; може працювати в будь-якому місці за умов використання

комп’ютерного пристрою, що має підключення до інтернету; користувач застрахований від збоїв у роботі пристрою і може за потреби ділитися результатами роботи з іншими користувачами. Перевагою для користувачів також є й те, що, на відміну від установлення платних програм на окремому ПК, ХТ. у більшості безкоштовні або розрахунки проводять у вигляді абонентської плати. Для організацій перевагою використання ХТ є зниження витрат на обслуговування, підтримку, модернізацію та адміністрування комп’ютерного обладнання і програмного забезпечення.

1.1.1. Переваги використання

- непотрібні потужні комп'ютери;
- менше витрат на закупівлю програмного забезпечення і його систематичне оновлення;
- необмежений обсяг збереження даних;
- доступність з різних пристроїв і відсутня прив’язка до робочого місця;
- забезпечення захисту даних від втрат та виконання багатьох видів навчальної діяльності, контролю і оцінювання, тестування он-лайн, відкритості освітнього середовища;
- економія коштів на утримання технічних фахівців.

1.1.2. Класифікація

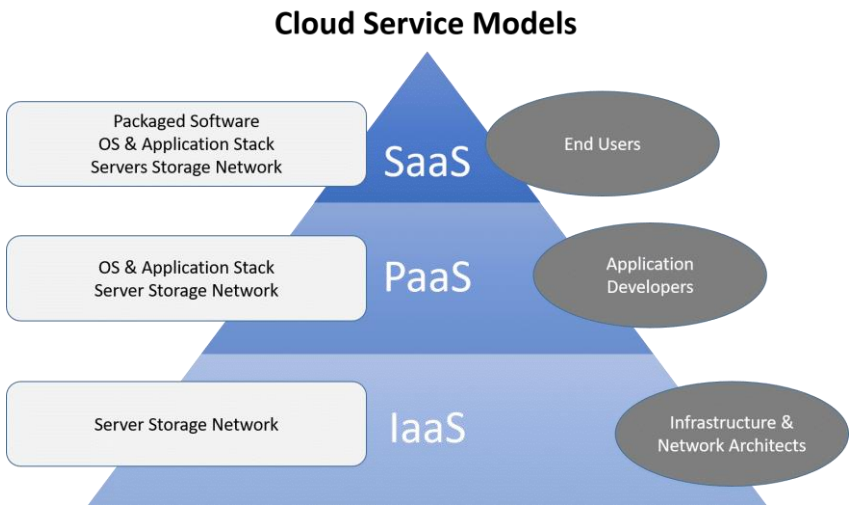


Рис. 1.1. Категорії, на які поділяються хмарні технології

ХТ за формою подання можуть бути розділені на такі категорії: додатки, платформи та інфраструктури, серед яких виділяють більш деталізовані типи:

1. як сервіс зберігання даних (Storage-as-a-Service), дисковий простір на вимогу. Ця послуга дає можливість зберігати дані в зовнішньому сховищі у «хмарі». Для користувача це додатковий логічний диск або папка. Сервіс є базовим для інших ХТ, оскільки входить до складу практично кожного з них;

2. сервіс баз даних (Database-as-a-Service), який надає можливості працювати з базами даних так, ніби система управління базами даних була встановлена на локальному ресурсі. У цьому разі набагато легше організовувати передачу інформації між різними виконавцями та додатками;

3. інформаційний сервіс (Information-as-a-Service), дає можливість віддалено використовувати будь-які види та архіви інформації (інсайдерська та галузева інформація для технічного і фундаментального аналізу, новинні стрічки телеграфних агентств, пропозиції з купівлі-продажу препаратів, кредитні історії, дорожній трафік тощо), яка може змінюватися в часі;

4. сервіс управління процесами (Process-as-a-Service) є віддаленим ресурсом, який може зв’язати воєдино кілька ресурсів, таких як послуги або дані, що містяться в межах однієї хмари або інших доступних хмарах, для створення єдиного бізнес-процесу. Бізнес-процес можна подати як додаток, що інтегрує базові послуги та інформацію, які скомбіновані в певну послідовність, що формує процес. Такі процеси завжди легше змінювати, ніж самі додатки;

5. додаток як сервіс (Application-as-a-Service) може мати назву «програмне забезпечення як сервіс» (Software as a Service), тобто будь-який додаток або програма, які користувач може запускати через інтернет;

6. сервіс-платформа (Platform-as-a-Service) — це повна платформа, що містить додатки, інтерфейси, бази даних, їх зберігання і тестування;

7. сервіс-інтеграція програм (Integration-as-a-Service) — можливість отримувати з хмари повний інтеграційний пакет, у тому числі програмні інтерфейси між додатками, семантичну медіацію, управління алгоритмом і дизайн

інтегрованого пакета. Сюди входять відомі послуги і функції пакетів централізації, оптимізації та інтеграції корпоративних додатків;

8. сервіс-безпека (Security-as-a-Service) — забезпечує безпечний доступ до корпоративної інформації, у тому числі ідентифікацію користувача, розпізнавання прав доступу тощо, які надаються з хмари;

9. сервіс адміністрування та управління (Management/Governace-as-a-Ser-vice) дає можливість керувати і задавати параметри роботи одного або багатьох ХТ: топологія, використання ресурсів, віртуалізація, тимчасові параметри роботи сервісів;

10. сервіс інфраструктур (Infrastructureas-as-a-Service) надає клієнту комп'ютерні інфраструктури: сервери, системи зберігання даних, мережеве устаткування, а також програми для управління цими ресурсами (замовник сплачує лише за те, що йому в певний час необхідно, з можливістю гнучкого збільшення чи зменшення обсягу використаних ресурсів);

11. сервіс-дані (Desktopas-as-a-Service) клієнти отримують повністю готове до роботи стандартизоване віртуальне робоче місце, яке кожен користувач може додатково налаштовувати під свої завдання. Користувач отримує доступ не до окремої програми, а до програмного комплексу, необхідного для повноцінної роботи;

12. сервіс робоче місце (Workspaceas-as-a-Service) — на відміну від попереднього сервісу дозволяє користувачеві отримувати доступ лише до програмного забезпечення, а всі обчислення відбуваються безпосередньо на ПК користувача.

1.1.3. Моделі розгортання

Приватна хмара – це хмарна інфраструктура, яка призначена для використання виключно однією організацією, що включає декілька користувачів (наприклад, підрозділів). Приватна хмара може перебувати у власності, керуванні та експлуатації як самої організації, так і третьої сторони (чи деякої їх комбінації). Така хмара може фізично знаходитись як в, так і поза юрисдикцією власника.

Публічна хмара – це хмарна інфраструктура, яка призначена для вільного використання широким загалом. Публічна хмара може перебувати у власності, керуванні та експлуатації комерційних, академічних (освітніх та наукових) або державних організацій (чи будь-якої їх комбінації). Публічна хмара перебуває в юрисдикції постачальника хмарних послуг.

Гібридна хмара – це хмарна інфраструктура, що складається з двох або більше різних хмарних інфраструктур (приватних, громадських або публічних), які залишаються унікальними сутностями, але з'єднанні між собою стандартизованими або приватними технологіями, що уможливлюють переносимість даних та прикладних програм (наприклад, використання ресурсів публічної хмари для балансування навантаження між хмарами).

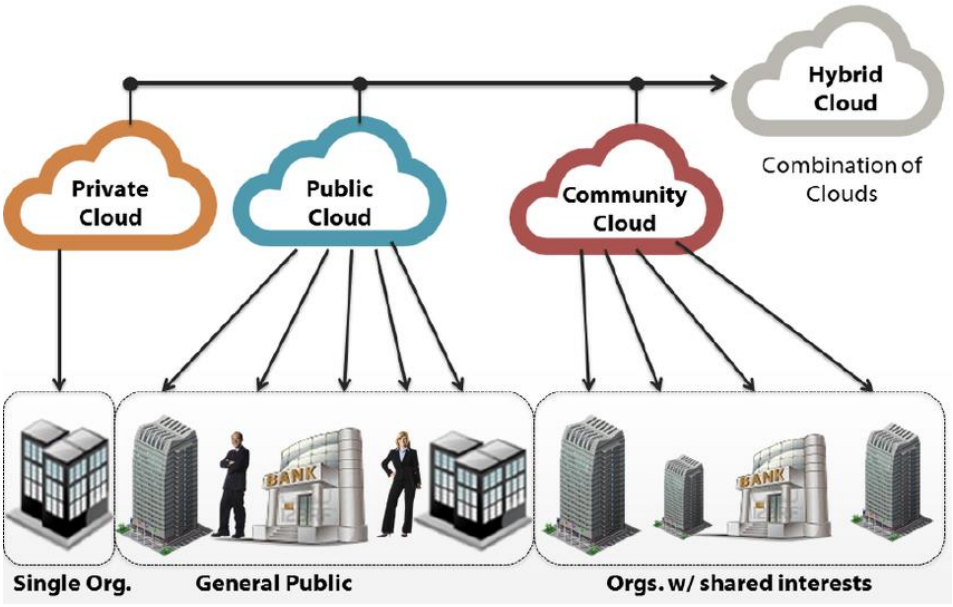


Рис. 1.2. Моделі розгортання хмарних сервісів

1.2. Хмарні сервіси зберігання даних

У сучасному світі у галузі інформаційно-комунікаційних технологій спостерігається бурхливий розвиток хмарних технологій. Відповідно до цього виникають численні хмарні сервіси, що все частіше застосовуються у різних сферах людської діяльності. Одним із найпоширеніших подібних сервісів є хмарні сховища даних. Ідею хмарних сервісів запропонували в 60-х роках Джон Маккарті і Джозеф

Ліклайдер – відомі вчені у галузі штучного інтелекту та обчислювальної техніки. Але до недавнього часу хмарні технології були суто професійними, тобто недоступними або незнайомими для пересічного користувача. Тим не менш, можливість перетворити ”хмарність” у бізнес - для компаній, і зручність використання хмарних сховищ звичайними людьми, принесли їх у масовий Інтернет. Хмарне сховище даних - модель онлайн-сховища, в якому дані зберігаються на численних розподілених в мережі серверах, що надаються у користування клієнтам, в основному, третьою стороною.

Таким чином, замість розміщення файлів на носіях зовнішньої пам’яті (або на вінчестерах комп’ютерів) інструменти і результати роботи поступово переносяться та розміщуються у хмарному сховищі даних або у ”хмарі“. Хмара представляє собою сукупність серверів (центр обробки даних, ЦОД), часто віддалених один від одного на великі відстані, об'єднаних високошвидкісною мережею і виконуючих специфічні завдання. Точне число серверів назвати важко (компанії тримають його в секреті), на сьогодні кількість серверів оцінюється в 2-2,5 млн і прогнозується їх збільшення до 10 млн. ЦОД підключені до Інтернету безліччю каналів, і коли користувач заходить почитати пошту або відредагувати фотографії, він потрапляє на найближчий і найменш завантажений вузол, який здійснює обробку інформації. Як взаємодіють між собою сервери всередині інфраструктури - таємниця розробника. А завдання користувача полягає в тому, щоб увійти в Інтернет, пройти авторизацію на обраному сервісі (див рис. 1.3).

За таких умов дані доступні з багатьох комп’ютерів. При цьому важливу роль відіграє те, що багато таких сервісів є безкоштовними або мають невисоку вартість. Серед переваг використання хмарних сховищ даних можна виділити такі: доступ до даних здійснюється з будь-якого місця та в будь-який час за наявності під’єднання до глобальної мережі Інтернет; користувач сплачує тільки за те місце у сховищі, яке фактично використовує або користується певним обсягом дискового простору хмарного сховища безкоштовно; економія дискового простору на жорсткому диску комп’ютера; усі процедури із збереження цілісності даних забезпечуються провайдером хмарного центру.

Рис. 1.3. Приклад хмарної інфраструктури

До недоліків належать: небезпека у процесі зберігання та пересилання даних, особливо конфіденційних, приватних; загальна продуктивність при роботі з даними в ”хмарі” може бути нижчою, ніж при роботі з локальними копіями даних; необхідна наявність стабільного та швидкісного підключення до Інтернету. Отже, основною різницею між хмарним сховищем даних та звичайними носіями даних є:синхронізація даних між різними комп’ютерами, резервне копіювання файлів з комп’ютера у “хмару”, спільна робота певної групи осіб з окремими файлами та папками.

1.3. Аналіз існуючих рішень

GoogleDrive

GoogleDrive – один з найпопулярніших хмарних сховищ даних, що дозволяє користувачам зберігати свої дані на серверах в хмарі і ділитися ними з іншими користувачами в інтернеті. Після активації, користувачу доступно Google Docs. По суті, Google Docs достатньо для роботи з документами, трансформували його в

хмарний сервіс і додали безкоштовного простору. У сервісі можна зберігати не тільки документи, але і фотографії, музику, відео та багато інших файли – всього близько 30 типів. Але взагалі все дуже зручно і звично для користувачів Google-сервісів. Кожному користувачеві Google Drive надається безкоштовно 15Gb простору. Якщо виділеного обсягу недостатньо, можна придбати додатково до 30 ТБ.

Крім доступу до сервісу через веб-інтерфейс, є можливість доступу через клієнти для Windows, Mac OS і Android, iOS.

OneDrive

OneDrive базується на хмарної організації інтернет-сервіс зберігання файлів з функціями файлообміну.

OneDrive створений в серпні 2007 року компанією Microsoft. Зараз OneDrive один з флагманів хмарних сховищ даних.

Перевагою сервісу OneDrive в тому, що він відразу інтегрований з Office 365, тому безпосередньо з програми можна створювати, редагувати та зберігати файли Excel, OneNote, PowerPoint і Word в службі Windows Live OneDrive.

Сервіс OneDrive дозволяє зберігати на даний момент безкоштовно 5 Гбайт (хоча раніше пропонувалося 15 Гбайт) інформації в упорядкованому за допомогою стандартних папок вигляді. Для зображень передбачений попередній у вигляді ескізів, а також можливість їх перегляду в вигляді слайдів.

Dropbox

Dropbox - хмарне сховище даних, що дозволяє користувачам зберігати свої дані на серверах в хмарі і розділяти їх з іншими користувачами в інтернеті. Його робота побудована на синхронізації даних.

На відміну від основних конкурентів, при роботі з Dropbox редаговані файли не копіюються повністю на сервер - здійснюється передача тільки зміненої частини, попередньо стиснутої. Вважається, що саме цей факт багато в чому пояснює відому оперативність роботи з Dropbox, в порівнянні з аналогами.

Dropbox дозволяє користувачеві розміщувати файли на віддалених серверах за допомогою клієнта або з використанням веб-інтерфейсу через браузер. Хоча

головний акцент технології робиться на синхронізації та обміні інформацією, Dropbox веде історію завантажень, щоб після видалення файлів з сервера була можливість відновити дані. Також ведеться історія зміни файлів, яка доступна на період останніх 30 днів, крім цього доступна функція безстрокової історії зміни файлів «Pack-Rat».

Mega

Mega - (MEGA Encrypted Global Access) - амбітний хмарний файлообмінник Кім Доткома (Kim Dotcom), засновника легендарного Megaupload.

Особливість Mega: шифрує весь контент прямо в браузері за допомогою алгоритму AES; користувачі можуть передавати один одному файли в зашифрованому вигляді, при цьому всі дані зберігаються в «хмарі»; ключі доступу до файлів не публікуються у відкритому доступі, а поширюються по схемі Friend-to-Friend, між довіряють один одному користувачами.

Яндекс.Диск

Яндекс.Диск - безкоштовний хмарний сервіс від Яндекса, що дозволяє користувачам зберігати свої дані на серверах в хмарі і передавати їх іншим користувачам в інтернеті. Робота побудована на синхронізації даних між різними пристроями. В даний час реєстрація користувачів доступна всім. Раніше, до запуску Яндекс.Диска, функції зберігання призначених для користувача файлів на Яндексі виконував сервіс Яндекс.Народ.

Крім того, Яндекс.Диск може виступати в якості служби хмарного сервісу, інтегруючись в офісний пакет MicrosoftOffice 2013, а недавно з'явилася можливість автоматичного завантаження фото та відеофайлів з цифрових камер і зовнішніх носіїв інформації на Яндекс.Диск. При цьому користувачу надаються додатково 32 ГБ простору на півроку.

Хмара@mail.ru

Хмара@mail.ru - нове і дуже перспективний хмарне сховище даних від компанії Mail.Ru Group, що дозволяє користувачам зберігати свої дані в хмарі і

синхронізувати дані на різних пристроях, а також ділитися ними з іншими користувачами. Сервіс відкрився відносно недавно - в кінці літа 2013.

Фішкою хмарного сховища даних Хмара@mail.ru - великий розмір дискового простору, що надається безкоштовно. Користувачі можуть відразу безкоштовно отримати 25 Гбайт хмарного сховища.

Користуватися хмарою можна не тільки через веб-інтерфейс, але через десктопні (для Windows і Mac OS) і мобільні додатки для Android і iOS. Для самих просунутих користувачів зроблений спеціальний клієнт під Linux.

Функція, з самого початку доступна в мобільних додатках - автозавантаження фотографій з телефону. Якщо підключена ця функція, всі фото, зроблені за допомогою пристрою, миттєво розташовуються у «Хмарі».

Bitcasa

Bitcasa - хмарне сховище даних, яке дозволяє зберігати необмежену кількість вашої інформації. Довгоочікуваний реліз вийшов зі стадії бета в 2013 році.

Bitcasa була заснована колишніми співробітниками Mastercard, VeriSign, Classmates.com і Mozy. А це значить, що творці сервісу не з чуток знайомі з рішеннями для резервного копіювання даних онлайн, а також із засобами, що забезпечують безпеку зберігання призначених для користувача файлів.

Ідея сервісу в тому, щоб надати всім бажаючим необмежений простір для хмарного зберігання даних. Що ж стосується безпеки, то Bitcasa виконує шифрування на стороні користувача, і всі дані передаються в «хмару» вже в зашифрованому вигляді. За заявою творців, ніхто із співробітників компанії не може отримати доступ до призначених для користувача даних.

Yunpan 360

Yunpan 360 - китайське хмарне сховище даних, яке спочатку безкоштовно і назавжди надає 36 Терабайт. Крім того, даний дисковий простір можна ще й збільшити, є користувачі, у яких більше 100 Терабайт на Yunpan 360.

Таким чином, всі інші хмарні сховища даних тьмяніють від таких обсягів дискового простору.

Однак, чому ж Yunpan 360 поки так погано відомий за межами Китаю? Відповідь проста - Yunpan 360 поки тільки на китайській мові, там немає навіть англійської версії, не кажучи вже про інші мови.

Box.net

Box.net - хмарне сховище даних, яке дозволяє зберігати ваші файли в мережі, а також спільно над ними працювати.

Переваги Box.net - це можливість перегляду офісних документів власними силами, а також можливість розшарити файли або папки для колег прямо з мобільного. Крім того, розробникам вдалося інтегрувати в додаток пошук Android за рахунок чого пошук файлів став швидше і точніше.

Box може служити сховищем контенту в Google Docs, включаючи листи і презентаціями. Люди можуть створювати, відкривати, редагувати і спільно працювати над Google Docs безпосередньо з коробки, і всі зміни будуть збережені назад в ящик в режимі реального часу. Співробітники можуть використовувати Google Docs "надійні" редагування і можливості спільної роботи в режимі реального часу, щоб виконати свою роботу швидше. Google Docs також може конвертувати інші типи файлів, такі як документи Microsoft Word або PDF-файлів в Google Doc для стрімких створення контенту і спільної роботи.

1.4. Концепція та пропозиція нового Web-сервісу

Всі запропоновані варіанти надають змогу колаборативного доступу до файлу, що дозволяє декільком користувачам скачувати спільний файл. Проте у більшості розглянутих сервісів (а саме у їх безкоштовних варіантах) доступ до файлу надається простим посиланням. Проблема у такому підході полягає в тому, що сторонні користувачі, отримавши доступ до посилання, легко можуть скачати файл і отримати доступ до небажаних персональних даних.

Ще одним із способів надання доступу до файлу іншим користувачам є можливість делегувати їм права доступу (тобто права на читання файлу, або на його видалення). Проте такий підхід має недолік – всі користувачі повинні бути

зареєстровані у даному веб-сервісі, а це, у свою чергу, зумовлює їх довіряти свої персональні дані до сервісу та витрачати час на реєстрацію і налаштування прав доступу для кожного з них. Лише після цього власник файлу зможе надати їм права доступу, а останні, у свою чергу, зможуть скачати файл. У випадку одноразового доступу до файлу усім користувачам потрібно повторити одну і ту ж дію реєстрації і це потрібно лише для того, щоб скачати файл лише один раз.

Концепцією нового файлообмінника є сервіс, що надає користувачеві місце під його файли і цілодобовий доступ до них через web, по протоколу HTTP. Такий сервіс дозволяє зручно «обмінюватися» файлами. На спеціальній сторінці користувач завантажує файл на сервер файлообмінника, який віддає користувачеві постійне посилання, яке він може розсилати по e-mail, публікувати в блогах, на форумах або пересилати через різні мессенджери. Перейшовши по такому посиланню будь-який інший користувач може завантажити початковий файл. Сервіс можна використовувати як звичайний файлообмінник. Проте його важливою ознакою є безпека. Питанню захищеності файлів було виділено велике значення. Тому окрім звичайного функціоналу файлообмінника передбачено захищений варіант для зберігання файлів та обміну ними.

При завантажені файлу на сервер, зареєстрований користувач зможе згенерувати унікальний ключ доступу до конкретного файлу. Завдяки цьому доступ до файлу мають можливість отримати лише ті, хто має цей ключ. Такий підхід доволі простий і водночас безпечний, бо дозволяє користувачеві самому вирішувати, хто зможе завантажувати файл.

Висновок

Сучасний світ важко собі уявити без хмарних сховищ зберігання даних, які міцно і надійно увійшли в наше комп'ютерне життя. Хмарне сховище - це онлайн-сховище, всі дані якого зберігаються на серверах. Доступ до цього сховища вже на платній, або безкоштовній основі надається в користування клієнтам.

В хмарних сховищах дані зберігаються і обробляються в так званій “хмарі”: для клієнта це виглядає як один великий віртуальний сервер, куди він може завантажити інформацію.

Оскільки використання мобільних пристроїв, таких як смартфони та планшети збільшується щодня, потреба в послугах хмарних сервісів зберігання даних також швидко зростає.

Розглянуті існуючі рішення хмарних сховищ розраховані на вирішення спільної задачі – надійне збереження файлів користувачів. При цьому кожне із рішень надає додаткові властиві тільки йому особливі функції та пропозиції. Проте вони є в певній мірі перегруженими цим додатковим функціоналом, який часто лише ускладнює користувацький інтерфейс.

РОЗДІЛ 2

Захист інформації

2.1. Актуальність та проблематика

Сьогодні інформація є досить важливим та дорогим ресурсом. Сама інформація є продуктом тому інтерес до її збереження є дуже високим у сучасному світі. Порушення безпеки називають комп’ютерним злочином. Такі злочини можна розділити на механічні, апаратні і програмні. Кожний з таких засобів має свої переваги і недоліки. Найбільш надійним і ефективним захистом є програмні засоби, а механічні і апаратні відходять на другий план захисту.

У сучасному світі все більше виробництв і послуг спираються на інформаційні технології. Виробництво і постачання енергії, очищення і постачання питної води, керування транспортом, освітлення міст, зв'язку, доступ людей до інформації, охорона здоров'я, оплата товарів і послуг, волевиявлення під час виборів і референдумів, і навіть електронне урядування – все це реалії нашого життя. Ми залежимо від безперервності та коректності функціювання комп'ютерних систем об’єктів критичної інфраструктури, і атаки з боку та засобами кібер простору на такі системи спричиняють реальні загрози для безпеки людей і суспільства.

Створення безпечних комп'ютерних систем і додатків є критичною задачею і метою діяльності мережевих інженерів і програмістів, а також предметом теоретичного дослідження як у галузі телекомунікацій та інформатики, так і економіки.

У зв'язку із складністю і трудомісткістю більшості процесів і методів захисту цифрового обладнання, інформації та комп'ютерних систем від ненавмисного чи несанкціонованого доступу вразливості комп'ютерних сис-

тем становлять значну проблему для їхніх користувачів.

Захист інформації в сучасних комп’ютерних ІС є пріоритетним завданням. Викрадення конфіденційної інформації, знищення даних, викривлення інформації, виведення з ладу комп’ютерних систем – далеко не повний перелік усіх ризиків, що виникають у процесі експлуатації та використання сучасних ІС.

Основою інформаційної безпеки є акт збереження конфіденційності, цілісності та доступності інформації (ЦРУ), який гарантує, що інформація не буде порушена будь-яким способом, коли виникнуть серйозні проблеми. Ці проблеми включають в себе, але не обмежуються цим, стихійні лиха, несправність комп’ютера чи сервера, фізичну крадіжку тощо.

Згідно із загальноприйнятим визначенням, безпечна комп'ютерна інформаційна система — це ідеальна система, яка коректно і у повному обсязі реалізує ті і лише ті цілі, що відповідають намірам її власника. На практиці побудувати складну систему, що задовольняє цьому принципові, неможливо, і не лише з огляду на ймовірність виникнення несправностей і помилок, але й через складність визначення і формулювання часто-густо суперечливих очікувань проектувальника системи, програміста, законного власника системи, власника даних, що обробляються, та кінцевого користувача. Навіть після їхнього визначення у багатьох випадках важко або й неможливо з'ясувати, чи функціонує програма у відповідності із сформульованими вимогами. У зв'язку з цим забезпечення безпеки зводиться найчастіше до управління ризиком: визначення потенційних загроз, оцінка ймовірності їхнього настання та оцінка потенційної шкоди, із наступним ужиттям запобіжних заходів в обсязі, що враховує технічні можливості й економічні обставини.

1.2. Джерела помилок у системах безпеки

Некоректна реалізація комп'ютерною інформаційною системою функцій, запланованих її автором чи власником, часто призводить до збитків і трагедій.

Прикладами є втрата зонду NASA "Марінер-1" у 1962 році через помилку у коді, написаному на мові Фортран, чи смертельні випадки серед пацієнтів через вади програмного забезпечення апаратів для радіотерапії Therac-25 у 80-х роках.

Із появою модемного зв'язку, глобальних мереж й Інтернету загрозу почала становити несанкціонована взаємодія із системою третіх осіб, хоча при цьому система може функціонувати у відповідності до намірів та очікувань її авторів та власників. Сценарії, що можуть призвести до несанкціонованого використання системи, можна класифікувати за їхнім походженням.

Помилки проектування

Цей термін означає, що програма будується на помилкових засадах, наприклад, на хибному розумінні засад функціонування комп'ютерних мереж і використовуваних комунікаційних протоколів. До помилок цього роду можна віднести використання нестійких шифрів, як це мало місце у випадку протоколу Нідгема — Шредера, застосованого у протоколі Kerberos, а також хибний вибір механізмів автентифікації чи повна довіра до інформації, надісланої клієнтом в архітектурі клієнт-сервер. Наслідком таких помилок можуть бути некоректні результати роботи додатку й одержання помилкових даних.

Помилки реалізації

До цієї групи належать технічні помилки, яких програмісти припускаються через свою недостатню обізнаність або неуважність. Прикладом є недостатня перевірка параметрів або результатів системних викликів, що може призвести до таких уразливостей, як переповнення буфера, невміле застосування функції ***printf()*** (англ. ***format string attack***) чи цілочисельне переповнення. Поширеним результатом помилок реалізації є можливість одержання повного контролю над процесом особою, що не має відповідних прав, чи можливість безпосередньої взаємодії з операційною системою.

Помилки конфігурації

Ця категорія об'єднує помилки адміністраторів, які налаштовують програмне забезпечення для користувачів. Такі помилки можуть виникати внаслідок нерозуміння документації чи особливостей функціонування програмного засобу, або ж через недбалість. Прикладом такого роду помилок є встановлення слабких паролів для привілейованих облікових записів чи надання надмірних прав без відповідного контролю доступу.

Помилки оператора

До цієї групи належать дії користувачів, які не мають повного розуміння роботи програмного забезпечення і принципів функціонування комп'ютерних систем. Приклади таких дій — запуск вкладень електронних листів від ненадійних відправників, ігнорування застережних повідомлень, випадкова зміна налаштувань програми, а також втрата носія із резервною копією даних.

Варто зазначити, що необережність зі сторони користувача є дуже поширеною і серйозною проблемою. Наприклад, за результатами проведених свого часу опитувань, понад 70% учасників опитування були готові повідомити свій пароль до комп'ютерної системи в обмін на плитку шоколаду.

Суперечливі питання класифікації

Дві останні групи помилок є предметом тривалих суперечок. Частина спеціалістів вважають, що автора системи не можна звинувачувати у некоректному конфігуруванні і застосуванні програмного забезпечення, і у зв'язку з цим такі помилки не повинні розглядатися як технічні вади системи безпеки. Інші твердять, що згідно із принципом найменшого здивування, якщо програма, не будучи інтуїтивно зрозумілою, сприяє цим самим збільшенню кількості помилок через дії користувача чи адміністратора, то це є недоліком самої програми.

Для кращого розуміння методів захисту комп'ютерної системи, що розглядатимуться в подальшому, слід також ознайомитися з поширеними типами

атак, які можуть бути здійснені. Такі небезпеки зазвичай можна віднести до однієї з наступних категорій.

Чорний хід (бекдор)

Чорний хід, або бекдор у комп'ютерній системі, криптосистемі чи алгоритмі — це метод обходу звичайного процесу аутентифікації, забезпечення віддаленого доступу до комп'ютера, одержання доступу до незашифрованої інформації тощо.

Бекдор в системі авторизації може прийняти форму добре закодованої комбінації логіну і пароля, яка дає доступ до системи.

Хоча число бекдорів в системах, що використовують власницьке програмне забезпечення (ПЗ, де сирцевий код не є загальнодоступним) не так багато, проте вони теж часто потрапляють під атаку. Програмістам навіть вдалося таємно впровадити велику кількість доброякісного коду, як великодні яйця в програми; такі випадки можуть включати в себе утримання від виконання будь-яких дій, звичайно, якщо це не дозволено.

Крім того, можна створити бекдор без зміни початкового коду програми, або навіть його зміни після компіляції. Це може бути зроблено шляхом переписування компілятора так, що він визнає код під час компіляції, який запускає включення бекдору в складеному виводі. Коли змінений компілятор знаходить такий код, він компілює його, як звичайно, але при цьому вставляє бекдор (можливо, процедуру розпізнавання пароля). Таким чином, коли користувач надає ці данні, він отримує доступ до деяких (швидше за все, до не задокументованих) аспектів роботи програми.

DoS-атака

На відміну від інших атак, DoS-атаки застосовуються не для одержання несанкціонованого доступу чи керування системою, а для того, щоб унеможливити роботу останньої. В результаті атаки акаунт окремої жертви може виявитися

заблокованим унаслідок умисного багаторазового введення невірного пароля, або ж унаслідок перевантаження мережі буде заблоковано усіх її користувачів. На практиці цьому виду атак дуже складно перешкодити, оскільки для цього необхідно проаналізувати поведінку цілих мереж, а не лише поведінку невеличкої частини коду.

Одним із найпоширеніших методів нападу є насичення атакованого комп'ютера або мережевого устаткування великою кількістю зовнішніх запитів (часто безглуздих або неправильно сформульованих) таким чином атаковане устаткування не може відповісти користувачам, або відповідає настільки повільно, що стає фактично недоступним. Взагалі відмова сервісу здійснюється:

- примусом атакованого устаткування до зупинки роботи програмного забезпечення/устаткування або до витрат наявних ресурсів, внаслідок чого устаткування не може продовжувати роботу;
- заняттям комунікаційних каналів між користувачами і атакованим устаткуванням, внаслідок чого якість сполучення перестає відповідати вимогам.

Атаки безпосереднього доступу

Користувач, який одержав несанкціонований доступ до комп'ютера (чи його частини), може встановлювати на ньому різні типи програмного (у тому числі модифікації операційних систем, віруси, програмні кілоґґери) та апаратного (апаратні кілоґґери, пристрої для прослуховування) забезпечення, внаслідок чого безпека системи опиниться під загрозою. Такий порушник може також легко скачати великі об'єми даних на зовнішні носії. Ще одним видом атак безпосереднього доступу є завантаження операційної системи з зовнішнього носія із наступним зчитуванням даних з жорсткого диску (дисків). Цей різновид атак є зазвичай єдиним методом атакування комп'ютерів, що не підключені до інтернету.

2.3. Вимоги та методи безпеки веб сервісів

При правильному підході, коли безепека розглядається як система, і їй приділяється достатньо уваги вже на стадії проектування, тоді веб–системи відрізняються надійностю і головне не вимагають в подальшому процесі роботи негайного *“латання дір”* в пошуках способів захисту додатку.

Тому необхідно піклуватись про захист веб сервісу на самому початку, під час проектування та створення проекту. Далі перейдемо до аспектів, які необхідно враховувати.

Бази даних

- якщо це можливо, потрібно використовувати шифрування для зберігання інформації, що ідентифікує користувача, а також для зберігання конфіденційних даних, на кшталт токенів доступу, адресу електронної пошти або платіжних реквізитів (такий підхід, зокрема, дозволить обмежити запити до бази до рівня пошуку за точним збігом).
- якщо СУБД підтримує економічне шифрування даних, що зберігаються, увімкніть його для захисту інформації, що знаходиться на дисках. Крім того, потрібно перевірити, щоб всі резервні копії баз теж були зашифровані.
- важливо використовувати для доступу до баз даних облікові записи користувачів з мінімальними привілегіями. Не варто застосовувати обліковий запис суперкористувача. Потрібно перевіряти систему на наявність невикористовуваних облікових записів і облікових записів із занадто слабкими паролями.
- варто виконувати збереження і передачу даних облікових записів, маркерів доступу до системи та іншої секретної інформації використовуючи сховище ключів, розраховане на подібні сценарії роботи.

- необхідно захистити систему від атак шляхом SQL-ін'єкцій, використовуючи тільки підготовлені SQL-запити.

Аутентифікація

- паролі повинні бути захешовані з використанням відповідного криптоалгоритма, наприклад, bcrypt. Не варто користуватися самописними функціями хешування.
- хорошою манерою є використання перевірених часом, добре зарекомендованих компонентів для організації входу в систему, відновлення забутого пароля і скидання пароля. Не варто «винаходити велосипед». Справа в тому, що при самостійній розробці дуже складно забезпечити коректну роботу подібних механізмів у всіх можливих сценаріях.
- потрібно впроваджувати прості вимоги до паролів, але вони не повинні загрожувати безпеці системи, а стимулювати користувачів до створення довгих паролів, що складаються з випадкових наборів символів.

Захист від DOS атак

- необхідно переконатися в тому, що DOS-атаки на публічний API не вплинуть на працездатність сайту. Обмеження частоти запитів в найповільніших місцях API і в тих частинах проекту, які пов'язані з аутентифікацією, може значно зменшити ризик. Наприклад, мова може йти про модулі входу в систему і про підсистеми створення токенів доступу. Для захисту серверних підсистем від DOS-атак як приклад можна розглянути використання CAPTCHA в тих частинах проекту, які доступні із зовнішнього світу.
- безпечним буде встановлення розумних обмежень на розміри структур даних, а також на виконувані користувачами запити, які вони можуть відправляти в систему.

- варто розглянути можливість використання системи для пом'якшення наслідків розподілених DOS-атак, наприклад, за допомогою глобального кешуючого проксі-сервісу на кшталт CloudFlare. Під час атаки це допоможе проекту вистояти, а в звичайний час знизить навантаження на сервери і прискорить завантаження сайту.

Веб трафік

- ще одним засобом безпеки веб сервісу є використання TLS протоколу для всього сайту, а не тільки для захисту системи авторизації. Не варто застосовувати TLS тільки для захисту форми авторизації. У перехідний період треба задіяти HTTP-заголовок strict-transport-security для примусового використання протоколу HTTPS.
- cookie-файли повинні мати атрибут httpOnly, вони повинні бути захищеними, серед їх атрибутів повинні бути присутніми параметри path і domain.
- треба використовувати політику захисту контента (CSP, Content Security Policy), не даючи дозволів unsafe-inline і unsafe-eval. Таку політику непросто налаштувати, але це коштує витрачених сил і часу. Механізм Subresource Integrity може бути використаний для CDN-контенту.
- HTTP-заголовки X-Frame-Option і X-XSS-Protection потрібно використовувати у відповідях клієнтів.
- варто застосовувати механізм HSTS для забезпечення доступу до системи тільки за допомогою TLS. Не потрібно довіряти клієнтським системам, забезпечте використання HTTPS на стороні сервера.
- потрібно використовуйте CSRF-токени у всіх формах, застосовуйте новий атрибут куки-файлів SameSite для захисту від CSRF-атак. Його підтримують сучасні версії браузерів.

API

- важливо перевірити, щоб в загальнодоступних API не було ресурсів, які можна виявити методом перебору.
- користувачі повинні бути повністю автентифіковано і відповідним чином авторизовані перед тим, як вони зможуть користуватися API.

Перевірка та перетворення даних

- дані, введені користувачами, необхідно перевіряти на стороні клієнта. Це дозволить швидше отримати відгук на свої дії. Однак, ніколи не потрібно повністю довіряти цій перевірці.
- повинна бути перевірка всього того, що надходить від користувача. Це може бути здійсненно за допомогою використання білих списків на сервері. Не варто вставляти незакодовані дані, введені користувачем, в HTTP-запити і відповіді. Ні в якому разі не треба використовувати потенційно небезпечні дані, введені користувачем, в SQL-запитах або в інший серверній логіці.

2.4. Криптографічні методи захисту даних

2.4.1. Огляд

Криптографія займається пошуком, дослідженням і розробкою математичних методів перетворення інформації, основою яких є шифрування. Розвинулась з практичної потреби передавати важливі відомості найнадійнішим чином. Для математичного аналізу криптографія використовує інструментарій абстрактної алгебри та теорії ймовірностей.

З криптографічних методів забезпечення конфіденційності інформації в інформаційних системах використовуються в основному методи шифрування. Під шифруванням розуміється процес перетворення відкритої інформації в зашифровану інформацію (шифртекст) або процес зворотного перетворення зашифрованої інформації у відкриту. Перетворення відкритої інформації в закриту отримало назву зашифрование, а перетворення зашифрованої інформації у відкриту - расшифрование.

Процес шифрування полягає в проведенні оборотних математичних, логічних, комбінаторних та інших перетворень вихідної інформації, в результаті яких зашифрована інформація являє собою хаотичний набір букв, цифр, інших символів і двійкових кодів.

Для шифрування інформації використовуються криптографічний алгоритм і ключ. Як правило, алгоритм для певного методу шифрування є незмінним. Вихідними даними для алгоритму зашифрування служать відкрита інформація і ключ зашифрування. Ключ містить керуючу інформацію (двійковий код), яка визначає вибір перетворення на певних кроках алгоритму і величини операндів, використовувані при реалізації алгоритму шифрування.

2.4.2. Класифікація алгоритмів шифрування

Існує цілий ряд алгоритмів шифрування даних, які можна розбити на дві великі групи, що показано на рисунку.

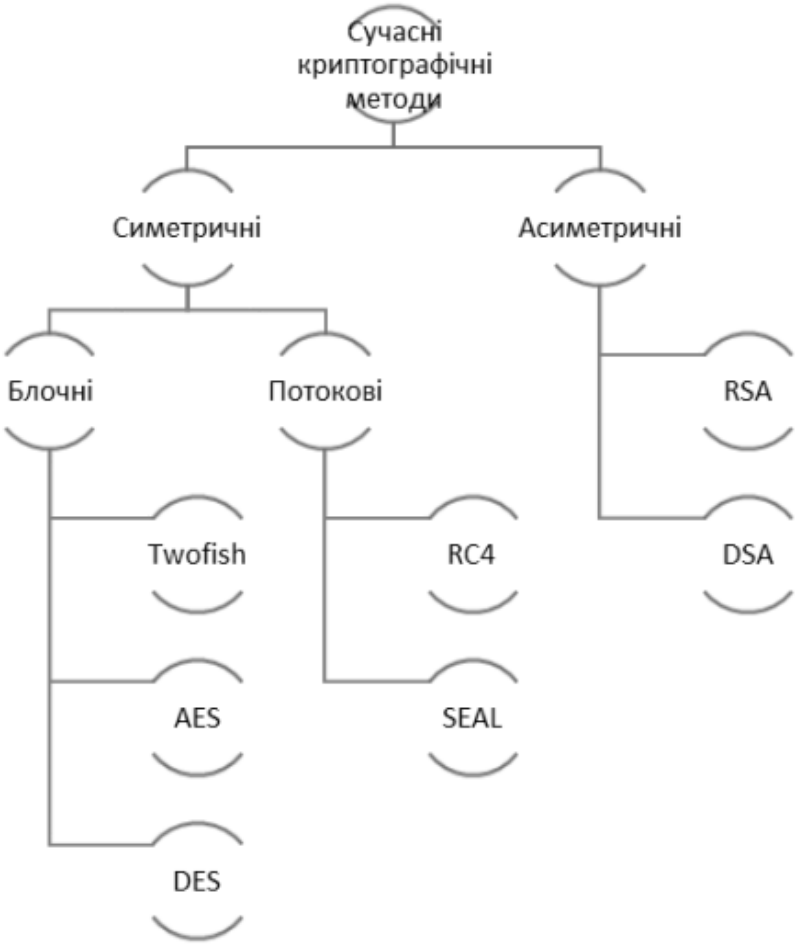


Рис. 2.1. Класифікація алгоритмів шифрування

Проведемо порівняльний аналіз симетричної та асиметричної криптографії.

2.4.2.1. Симетричне шифрування

До алгоритмів симетричного шифрування належать методи шифрування, в яких і відправник, і отримувач повідомлення мають однаковий ключ (або, що менш поширено, ключі різні, але споріднені та легко обчислюються).

Сучасні дослідження симетричних алгоритмів шифрування зосереджено, в основному, навколо блочних та потокових алгоритмів шифрування та їхнього застосування. Блочний шифр подібний до поліалфавітного шифру Альберті: блочні шифри отримують фрагмент відкритого тексту та ключ, і видають на виході шифротекст такого самого розміру. Оскільки повідомлення зазвичай довші за один блок, потрібен деякий метод склеювання послідовних блоків. Було розроблено декілька методів, що відрізняються в різних аспектах. Вони є режимами дії блочних шифрів та мають обережно обиратись під час застосування блочного шифру в криптосистемі.

Шифри Data Encryption Standard (DES) та Advanced Encryption Standard (AES) є стандартами блочних шифрів, затверджених урядом США.

Потокові шифри, на відміну від блочних, створюють ключ довільної довжини, що накладається на відкритий текст побітово або політерно, в дечому подібно до одноразової дошки. В потокових шифрах, потік шифротексту обчислюється на основі внутрішнього стану алгоритму, який змінюється протягом його дії. Зміна стану керується ключем, та, в деяких алгоритмах, ще і потоком відкритого тексту. RC4 є прикладом добре відомого, та широко розповсюдженого потокового шифру.

Криптографічні хешувальні функції (англ. *cryptographichashfunctions*, або англ. *messagedigestfunctions*) не обов'язково використовують ключі, але часто використовуються і є важливим класом криптографічних алгоритмів. Ці функції отримують дані (часто, ціле повідомлення), та обчислюють коротке, фіксованого

розміру число (*хеш*). Якісні хешувальні функції створені таким чином, що дуже важко знайти колізії (два відкритих тексти, що мають однакове значення хешу).

Коди аутентифікації повідомлень (англ. *Messageauthenticationcode*, MAC) подібні до криптографічних хешувальних функцій, за виключенням того, що вони використовують секретний ключ для аутентифікації значення хешу при отриманні повідомлення. Ці функції пропонують захист проти атак на прості хешувальні функції.

2.4.2.2. Асиметричне шифрування

На відміну від симетричних, асиметричні алгоритми шифрування використовують пару споріднених ключів — відкритий та секретний. При цьому, не зважаючи на пов'язаність відкритого та секретного ключа в парі, обчислення секретного ключа на основі відкритого вважається технічно неможливим.

В асиметричних криптосистемах, відкритий ключ може вільно розповсюджуватись, в той час як приватний ключ має зберігатись в таємниці. Зазвичай, *відкритий ключ* використовується для шифрування, в той час як *приватний* (*секретний*) ключ використовується для дешифрування.

Рис. 2.2. Схематичне представлення асиметричного шифрування

Головне досягнення асиметричного шифрування в тому, що воно дозволяє людям, що не мають наперед наявної домовленості про безпеку, обмінюватися секретними повідомленнями. Необхідність відправникові й одержувачеві погоджувати таємний ключ по спеціальному захищеному каналу цілком відпала. Для того, щоб відправити конфіденційне повідомлення відправникові необхідний

відкритий ключ. З його допомогою це повідомлення шифрується, а потім відсилається. Одержувач цього повідомлення розшифровує його за допомогою закритого ключа. В цьому випадку ніхто, крім власника закритого ключа, не може дешифрувати повідомлення. Прикладами криптосистем з відкритим ключем є Схема Ель-Гамаля (названа на честь автора, Тахера Ель-Гамаля), RSA(названа на честь винахідників: Рона Рівеста, Аді Шаміра і ЛеонардаАдлмана), ДіффіГеллмана і DSA, англ. Digital Signature Algorithm (винайдений Девідом Кравіцом).

2.5. Data Encryption Standard

Стандарт шифрування даних (DES) — блоковий шифр із симетричними ключами, розроблений Національним Інститутом Стандартів і Технології (NIST – National Institute of Standards and Technology).

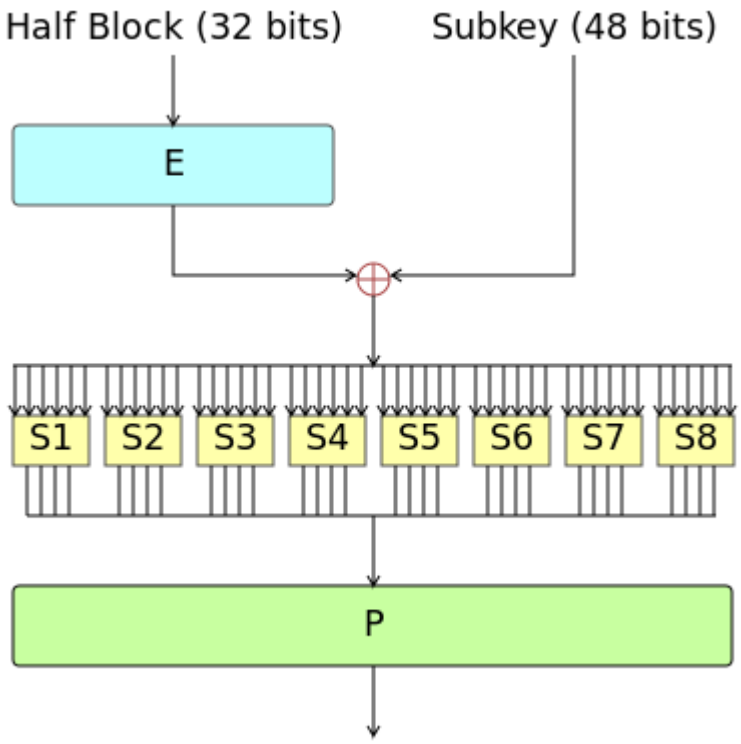


Рис. 2.3. Data Encryption Standard

Опис роботи алгоритму

DES є блочним шифром - дані шифруються блоками по 64 біти - 64 бітний блок явного тексту подається на вхід алгоритму, а 64-бітний блок шифрограми

отримується в результаті роботи алгоритму. Крім того, як під час шифрування, так і під час дешифрування використовується один і той самий алгоритм (за винятком дещо іншого шляху утворення робочих ключів).

Ключ має довжину 56 біт (як правило, в джерельному вигляді ключ має довжину 64 біти, де кожний 8-й біт є бітом паритету, крім того, ці контрольні біти можуть бути винесені в останній байт ключа). Ключем може бути довільна 64-бітна комбінація, яка може бути змінена у будь-який момент часу. Частина цих комбінацій вважається слабкими ключами, оскільки може бути легко визначена. Безпечність алгоритму базується на безпечності ключа.

На найнижчому рівні алгоритм є ніщо інше, ніж поєднання двох базових технік шифрування: перемішування і підстановки. Цикл алгоритму, з яких і складається DES є комбінацією цих технік, коли як об'єкти перемішування виступають біти тексту, ключа і блоків підстановок.

Початкова перестановка

На вході подаються 64-бітний блок даних, які переставляються згідно з таблицею:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
58	50	42	34	26	18	10	2	60	52	44	36	28	20	12	4	62	54	46	38	30	22	14	6	64	56	48	40	32	24	16	8

33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
57	49	41	33	25	17	9	1	59	51	43	35	27	19	11	3	61	53	45	37	29	21	13	5	63	55	47	39	31	23	15	7

Рис. 2.4. Початкова перестановка IP

Далі 16 разів повторюються наступні операції:

Функція розширення блоку

Функція E розширює 32-бітовий вектор R_{i-1} до 48-біового вектора $E(R_{i-1})$ шляхом повторення деяких біт з R_{i-1} згідно з таблицею:

_	1	2	3	4	_	_	5	6	7	8	_	_	9	10	11	12	_	_	13	14	15	16	_	_	17	18	19	20	_	_	21	22	23	24	_	_	25	26	27	28	_	_	29	30	31	32	_
32	1	2	3	4	5	4	5	6	7	8	9	8	9	10	11	12	13	12	13	14	15	16	17	16	17	18	19	20	21	20	21	22	23	24	25	24	25	26	27	28	29	28	29	30	31	32	1

Рис. 2.5. Розширення вектора

Перший рядок - номери вхідних біт, другий рядок - вихідні біти. Повторення номерів, означає повторення біт.

Додавання раундового ключа k_i

Блок 48 біт XOR'иться з раундовим ключем k_i

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7	
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8	S_1
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0	
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13	
0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10	
1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5	S_2
2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15	
3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9	
0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8	
1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1	S_3
2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7	
3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12	
0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15	
1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9	S_4
2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4	
3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14	
0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9	
1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6	S_5
2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14	
3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3	
0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11	
1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8	S_6
2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6	
3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13	
0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1	
1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6	S_7
2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2	
3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12	
0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7	
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2	S_8
2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8	
3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11	

Рис. 2.6. Таблиці підстановки (S-блоки мають такий вигляд)

"Розширені" біти використовуються для визначення номера 0-1-2-3 таблиці (ліва колонка).

Перестановка

Далі виконується перестановка

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
16	7	20	21	29	12	28	17	1	15	23	26	5	18	31	10	2	8	24	14	32	27	3	9	19	13	30	6	22	11	4	25

Рис. 2.7. Перестановка P

XOR лівої і правої частин

За формулою $R_i = L_i XOR f(R_i, k_i)$ отримуємо значення R_i.

Після 16-ти раундів застосовується кінцева перестановка біт.

Вхідний перший біт ставить на місце номер 40, другий біт на місце номер 8 і т.д.

В DES використовується 16 циклів, вихідними даними для кожного з них 64 є біти відкритого тексту (на вході першого раунду) чи 64 біти з результату роботи попереднього раунду (у всіх наступних), ключ і блоки підстановок.

Алгоритм використовує лише стандартні елементарні логічні операції (зсув, сума по модулю два(XOR)) на числах довжиною 64 біти, що значно полегшує програмування алгоритму і прискорює його роботу у інтегральних мікросхемах. Циклічний характер алгоритму в сумі з ідеальною здатністю до запрограмовування робить алгоритм швидким і легким до розуміння.

Приклад роботи алгоритму

Для ключа 00 00 00 00 00 00 00 00 (HEX)

та тексту 00 00 00 00 00 00 00 00 (HEX)

Результати шифрування відкритого тексту ключем

8C A6 4D E9 C1 B1 23 A7 (HEX)

(test vector 8ca64de9c1b123a7)

2.6. Advanced Encryption Standard

Advanced Encryption Standard (AES), також відомий під назвою Rijndael — симетричний алгоритм блочного шифрування (розмір блока 128 біт, ключ 128/192/256 біт), фіналіст конкурсу AES і прийнятий як американський стандарт шифрування урядом США. Вибір припав на AES з розрахуванням на широке використання і активний аналіз алгоритму, як це було із його попередником, DES.

Тип шифрування AES передбачає використання для перетворення інформації в захищений вид і зворотного декодування одного і того ж ключа, який відомий і відправляє, і приймаючій стороні, на відміну від симетричного шифрування, в якому передбачено застосування двох ключів – закритого і відкритого. Таким чином, неважко зробити висновок, що, якщо обидві сторони знають правильний ключ, процес шифрування і дешифрування проводиться досить просто.

Опис роботи алгоритму

В принципі, алгоритм, запропонований Рейменом і Дейцменом, і AES не одне і те ж. Алгоритм *Рейндол* підтримує широкий діапазон розміру блоку та ключа. AES має фіксовану довжину у 128 біт, а розмір ключа може приймати значення 128, 192 або 256 біт. В той час як Рейндол підтримує розмірність блоку та ключа із кроком 32 біт у діапазоні від 128 до 256. Через фіксований розмір блоку AES оперує із масивом 4×4 байт, що називається *станом* (версії алгоритму із більшим розміром блоку мають додаткові колонки).

Для ключа 128 біт алгоритм має 10 раундів у яких послідовно виконуються операції

1. *subBytes*()
2. *shiftRows*()
3. *mixcolumns*() (у 10-му раунді пропускається)
4. *xorRoundKey*()

SubBytes()

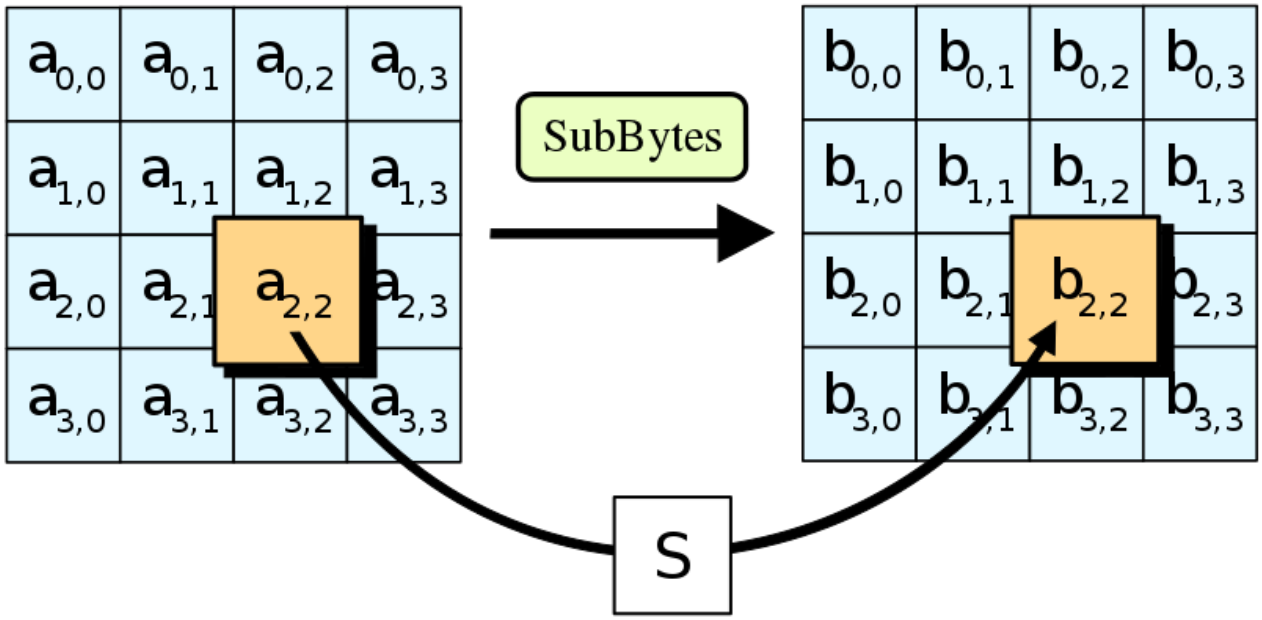


Рис. 2.8. У процедурі SubBytes , кожен байт в state замінюється відповідним елементом у фіксованій 8-бітній таблиці пошуку, S; $b_{ij} = S(a_{ij})$.

Процедура SubBytes() обробляє кожен байт стану незалежно, проводячи нелінійну заміну байтів використовуючи таблицю замін (S-box). Така операція забезпечує не лінійність алгоритму шифрування. Побудова S-box складається з двох кроків. По-перше, проводиться отримання зворотного числа в полі Галуа . По-друге, до кожного байту b з яких складається S-box застосовується така операція:

$$B'_i = b_i \oplus b_{(i+4) \bmod 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+6) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus c_i$$

де $0 \leq i < 8 0 \leq i < 8$,іде$b_i$ є i-ийбіт b, а c_i – i-ий біт константи $c = 63_{16} = 99_{10} = 01100011_2$. Таким чином, забезпечується захист від атак, заснованих на простих алгебраїчних властивостях.

ShiftRows()

ShiftRows працює з рядками таблиці State. При цій трансформації рядка стану циклічно зсуваються на r байтів по горизонталі, залежно від номера рядка. Для нульового рядка r = 0, для першого рядка r = 1 і т. д. Таким чином кожна колонка вихідного стану після застосування процедури ShiftRows складається з байтів з кожної колонки початкового стану. Для алгоритму Rijndael патерн зсуву рядків для 128 - і 192-бітних рядків однаковий. Однак для блоку розміром 256 біт відрізняється від попередніх тим, що 2, 3, і 4-і рядки зміщуються на 1, 3, і 4 байти, відповідно.

Фактично це проста перестановка байтів таблиці 4x4 State.

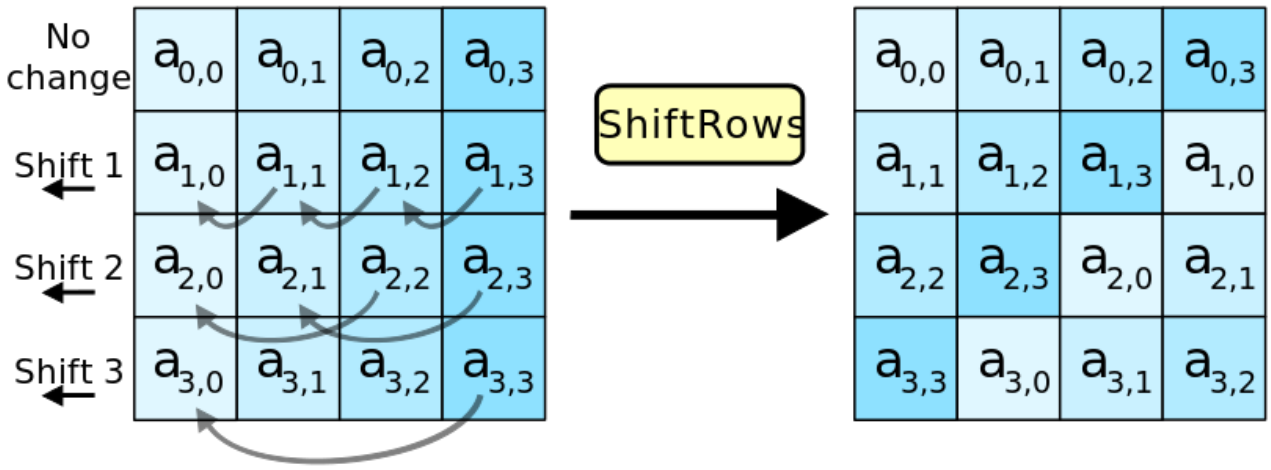


Рис. 2.8. У процедурі ShiftRows , байти в кожному рядку state циклічно зсуваються в ліво. Розмір зміщення байтів кожного рядка залежить від її номера

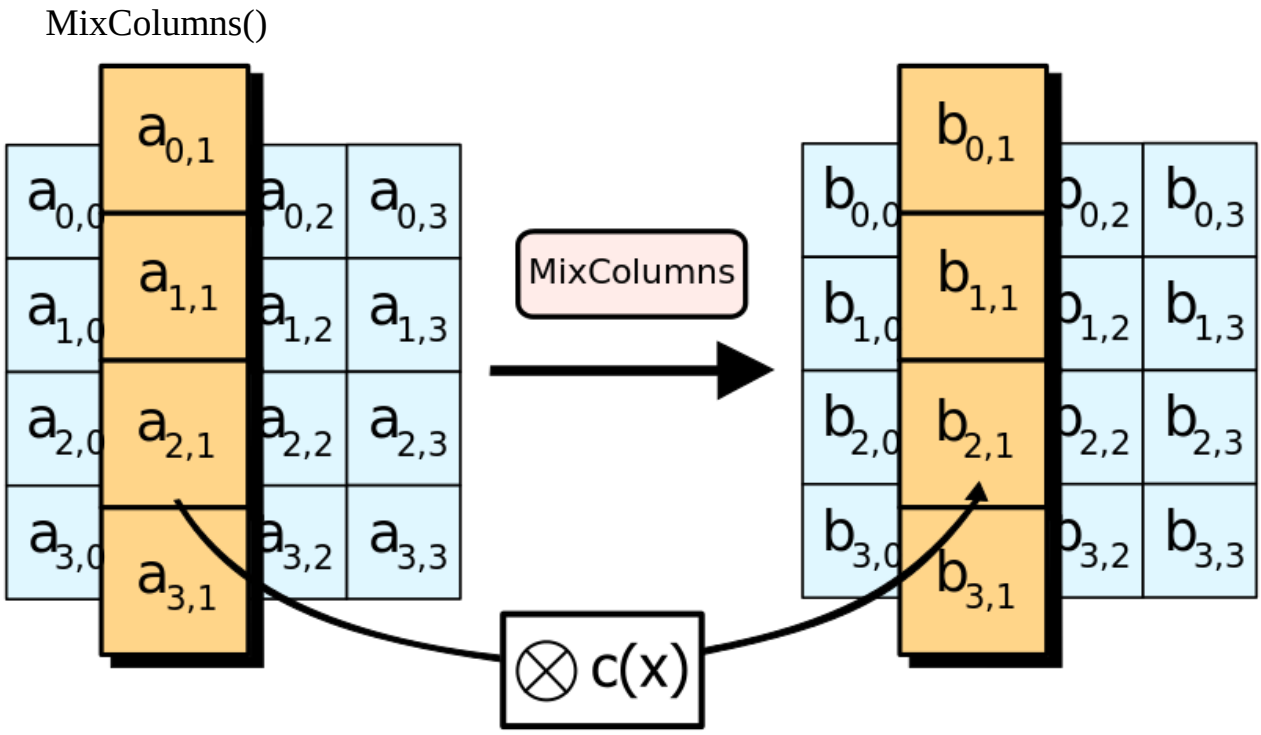


Рис. 2.9.У процедурі MixColumns , кожна колонка стану перемножується з фіксованим многочленом c (x).

У процедурі MixColumns , чотири байти кожної колонки State змішуються, використовуючи для цього зворотну лінійну трансформацію. MixColumns опрацьовує стан по колонках, трактуючи кожну з них як поліном четвертого степеня. Над цими поліномами виконується множення в $GF(2^8)$ по модулю $x^4 + 1$ на фіксований многочлен $c(x) = 3x^3 + x^2 + x + 2$. Разом з ShiftRows , MixColumns вносить дифузію в шифр.

Під час цієї операції, кожен стовпчик множиться на матрицю, яка для 128-бітного ключа має вигляд

$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix}.$$

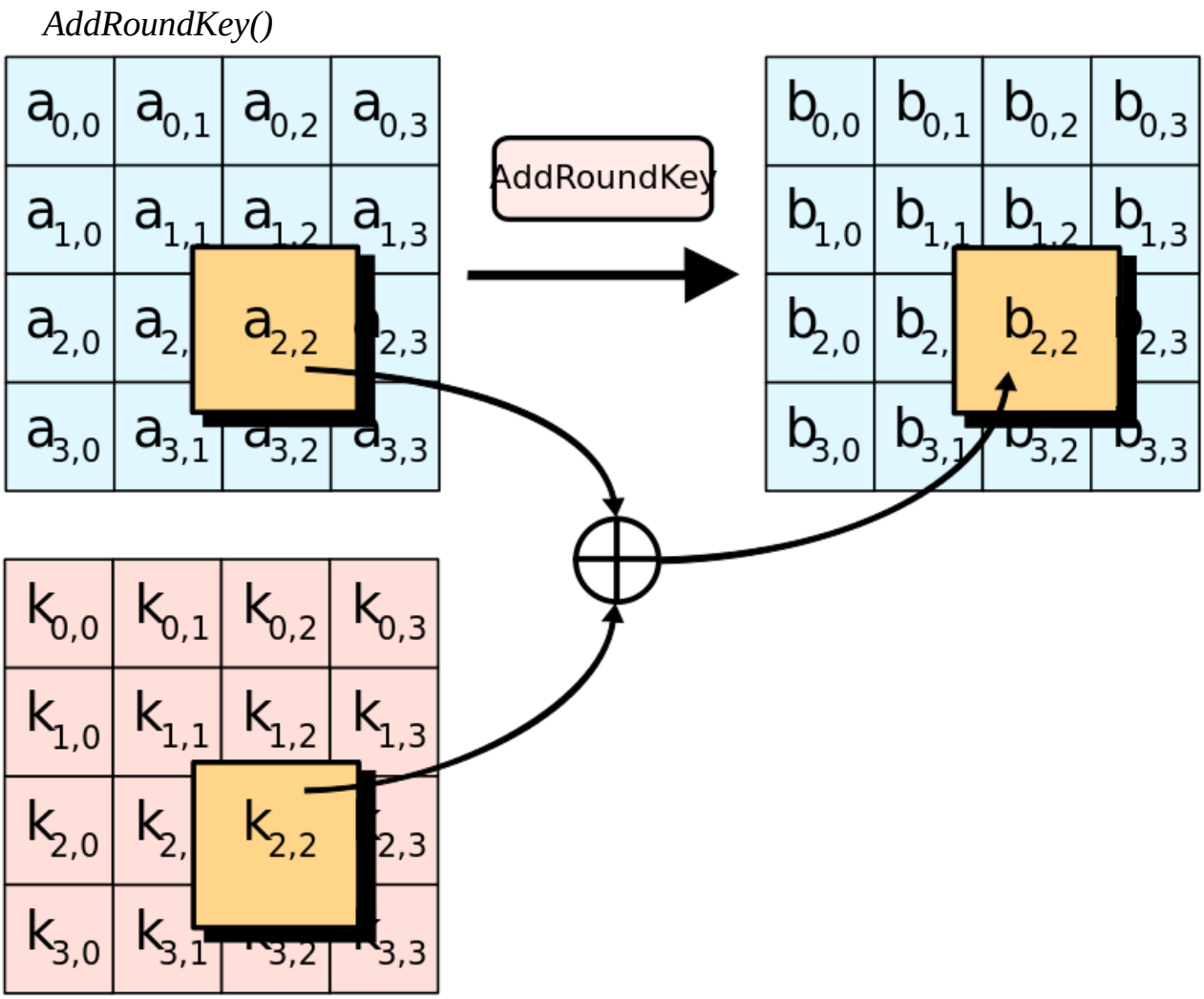


Рис. 3.10.Упроцедурі AddRoundKey , кожен байт стану об'єднуєтьсяз RoundKey використовуючи операцію XOR.

У процедурі AddRoundKey , RoundKey кожного раунду об'єднується зі State. Для кожного раунду Roundkey виходить з CipherKey використовуючи процедуру KeyExpansion ; кожен RoundKey такого ж розміру, що і State. Процедура виробляє побітовий XOR кожного байта State з кожним байтом RoundKey .

2.6. Криптографічна хеш-функція

Криптографічна геш-функція — це геш-функція, яка є алгоритмом, що приймає довільний блок даних і повертає рядок встановленого розміру, (криптографічне) геш-значення, таке що (випадкові або навмисні) зміни даних (з дуже високою ймовірністю) змінять геш-значення. Дані до кодування часто звуть «повідомлення», а геш-значення іноді називають дайджест повідомлення (англ. *messagedigest*) або просто дайджест, дослівно «стислий виклад».

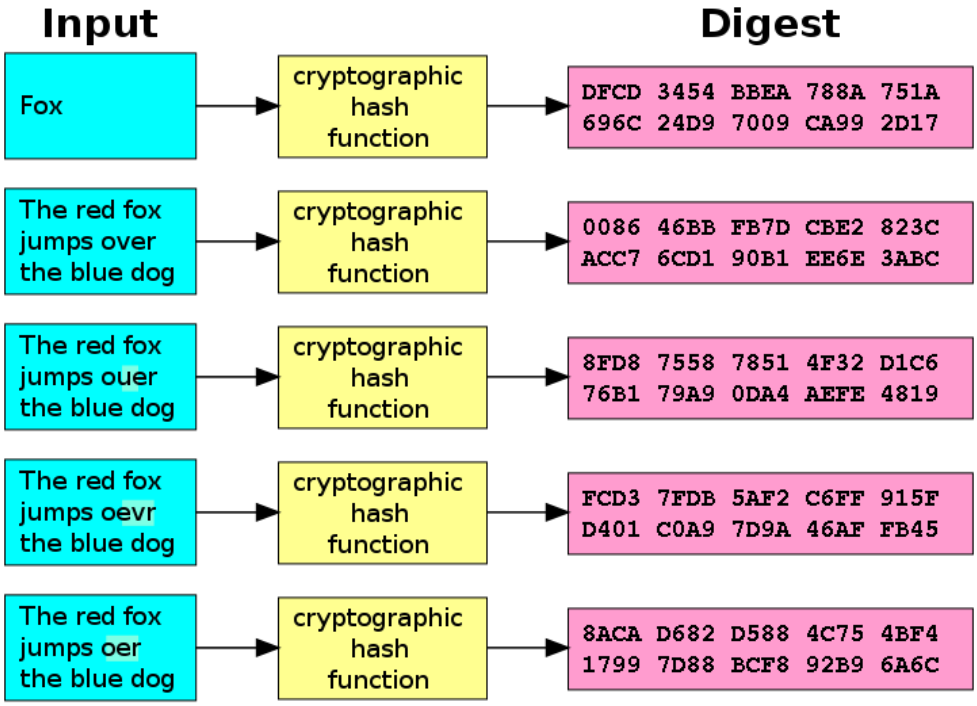


Рис. 2.11.Криптографічна геш-функція (а саме, SHA-1) в роботі. Зауважте, що навіть мала зміна даних на вході (тут в слові «over») значно змінює вихід, через так званий лавиновий ефект.

Ідеальна криптографічна геш-функція має чотири основні або значимі властивості:

- легкість обчислення геш-значення для будь-якого повідомлення
- нездійсненно утворити повідомлення для заданого геш-значення
- нездійсненно змінити повідомлення без зміни геша
- нездійсненно знайти два різних повідомлення з тим самим гешем

Властивості

Більшість криптографічних геш-функцій спроектовано так, щоб на вхід подавався рядок довільної довжини, а на виході ми отримували геш-значення встановленої довжини.

Криптографічна геш-функція повинна витримувати всі відомі типи криптографічних атак. Щонайменше, вона повинна мати наступні властивості:

- *Першовзорова стійкість.* Для певного гешу h має бути складно знайти повідомлення m, таке що $h = hash(m)$. Ця концепція стосується односторонньої функції. Функції, що не відповідають цій властивості вразливі до атак знаходження першовзору.
- *Друга першовзорова стійкість.* Для певного входу m_1 має бути складно знайти інший вхід m_2 — де $m_1 != m_2$, такий що $hash(m_1) != hash(m_2)$. Цю властивість іноді називають *слабка колізійна стйкість* (англ. *weakcollisionresistance*), і функцій, що не мають цієї властивості вразливі до атак знаходження першовзору другого роду.
- *Колізійна стійкість.* Повинно бути складно знайти два різних повідомлення m_1 і m_2, таких що $hash(m_1) = hash(m_2)$. Така двійка зветься криптографічна геш-колізія. Цю властивість іноді називають *сильна колізійна стійкість* (англ. *strongcollisionresistance*). Воно вимагає щонайменше вдвічі довшого геш-значення ніж потрібно для першовзорової стійкості, інакше колізії можна знайти за допомогою атаки «днів народження».

Ці властивості мають на увазі, що зловмисник не може змінити вхідні дані без зміни дайджеста. Отже,якщо два рядки мають однаковий дайджест можна бути впевненим, що й самі вони такі.

2.6.1. Хешування паролів

Хешування паролів – метод, що дозволяє користувачам запам`ятовувати, наприклад, не 128 байт, тобто 256 шестнадцятірічний цифр ключа, а деяке осмислений вираз, слово чи послідовність символів, що називається паролем. Дійсно, при розробці будь-якого крипто-алгоритмами слід враховувати, що в половині випадків кінцевим користувачем системи є людина, а не автоматична

система. Це ставить питання про те, зручно, і взагалі чи реально людині запам'ятати 128-бітний ключ. Насправді межа запам'ятовуваності лежить на кордоні 8-12 подібних символів, а, отже, якщо ми будемо примушувати користувача оперувати саме ключем, тим самим ми практично змусимо його до запису ключа на будь-якому листку паперу або електронному носії, наприклад, в текстовому файлі. Це, звичайно, різко знижує захищеність системи. Для вирішення цієї проблеми були розроблені методи, які перетворюють осмислений рядок довільної довжини – пароль, у вказаний ключ заздалегідь заданої довжини. У переважній більшості випадків для цієї операції використовуються так звані хеш-функції. Хеш-функцією в даному випадку називається таке математичне або алгоритмічне перетворення заданого блоку даних, яке володіє наступними властивостями

- хеш-функція має нескінченну область визначення,
- хеш-функція має кінцеву область значень,
- вона незворотня,
- зміна вхідного потоку інформації на один біт змінює близько половини всіх біт вихідного потоку, тобто результату хеш-функції.

Ці властивості дозволяють подавати на вхід хеш-функції паролі, тобто текстові рядки довільної довжини на будь-якою національною мовою і, обмеживши область значень функції діапазоном 0 .. 2N-1, де N - довжина ключа в бітах, отримувати на виході досить рівномірно розподілені по області значення блоки інформації – ключі.

2.6.2. Алгоритм хешування bcrypt

bcrypt — адаптивна криптографічна функція формування ключа, що використовується для безпечного зберігання паролів. Розробники: Нільс Провос і David Mazières. Функція заснована на шифрі Blowfish, вперше представлена на USENIX у 1999 році. Для захисту від атак за допомогою райдужних таблиць bcrypt використовує сіль (salt); крім того, функція є адаптивною, час її роботи легко настроюється і її можна сповільнити, щоб ускладнити атаки перебором.

Алгоритм bcrypt використовує алгоритм налаштування ключів з «Eksblowfish»:

```
EksBlowfishSetup(cost, salt, key)
state InitState()
state  ExpandKey(state, salt, key)
repeat (2^cost)
state  ExpandKey(state, 0, key)
state  ExpandKey(state, 0, salt)
returnstate
```

Функція InitState відповідає оригінальній функції з шифру Blowfish; для заповнення масиву P і S-box використовується дробна частина числа.

Функція ExpandKey:

```
ExpandKey(state, salt, key)
    for(n = 1..18)

        P_n    key[32(n-1)..32n-1]      P_n //treat the key as cyclic

ctext     Encrypt(salt[0..63])

    P_1    ctext[0..31]

    P_2    ctext[32..63]
    for(n = 2..9)

ctext     Encrypt(ctext    salt[64(n-1)..64n-1]) //encrypt using the
current key schedule and treat the salt as cyclic

        P_2n-1)    ctext[0..31]

        P_2n    ctext[32..63]
    for(i = 1..4)
        for(n = 0..127)

ctext     Encrypt(ctext    salt[64(n-1)..64n-1]) //as above

            S_i[2n]     ctext[0..31]

            S_i[2n+1]     ctext[32..63]
returnstate
```

Для обчислення хешу bcrypt обробляє вхідні дані еквівалентно шифруванню 'eksblowfish(посилений_ключ, input)':

```
bcrypt(cost, salt, key, input)
state  EksBlowfishSetup(cost, salt, key)
ctextinput
repeat(64)
ctext  EncryptECB(state, ctext) // Blowfish в режимі ECB
return Concatenate(cost, salt, ctext)
```

Висновок

Збереження інформації є досить не простою задачею. Тому що існує багато різних загроз втрати даних та порушення прав інформації. У сучасному світі великою загрозою збереженням даних є велика різноманітність Dos-атак, атак безпосереднього доступу в мережі. Тому виникає потреба в непростій системі захисту інформації, що дозволить зберегти не тільки саму цілісність, а й слідкувати за захистом інформації від зовнішніх факторів. За основу вирішення такої задачі можна застосувати відомі та надійні систем захисту інформації. Такі як: захист прикладних систем, захист зберігання даних, захист спеціалізованих систем, криптографічний захист інформації, комплексні системи захисту та захист інформації на веб-ресурсах тощо.

Криптографічний алгоритм, або шифр, — математична формула, що описує процеси шифрування і розшифрування. Щоб зашифрувати відкритий текст, крипто-алгоритм працює в сполученні з ключем — словом, числом або фразою. Те саме повідомлення одним алгоритмом, але різними ключами буде перетворюватися в різний шифротекст. Захищеність шифротексту цілком залежить від двох речей: стійкості криптоалгоритму і таємності ключа. Криптоалгоритм плюс усілякі ключі і протоколи, що приводять їх у дію, складають криптосистему.

Для сучасної криптографії властиве застосування відкритих алгоритмів шифрування, що припускають використання обчислювальних засобів. Відомо більше десятка перевірених алгоритмів шифрування, які, при використанні ключа достатньої довжини і коректної реалізації алгоритму, роблять шифрований текст недоступним для практичного криптоаналізу. У багатьох країнах прийняті національні стандарти шифрування. У 2001 році в США прийнятий стандарт симетричного шифрування AES на основі алгоритму Rijndael з довжиною ключа 128, 192 і 256 біт. Алгоритм AES прийшов на зміну колишньому алгоритмові DES.

РОЗДІЛ 3

Реалізація Web-сервісу сховища файлів та безпечного обміну ними в мережі Інтернет

3.1. Призначення та основні можливості сервісу

Файлообмінник — сервіс, що надає користувачеві місце під його файли і цілодобовий доступ до них через web, по протоколу HTTP. Такий сервіс дозволяє зручно «обмінюватися» файлами. На спеціальній сторінці користувач завантажує файл на сервер файлообмінника, який віддає користувачеві постійне посилання, яке він може розсилати по e-mail, публікувати в блогах, на форумах або пересилати через різні мессенджери. Перейшовши по такому посиланню будь-який інший користувач може завантажити початковий файл.

Принцип роботи

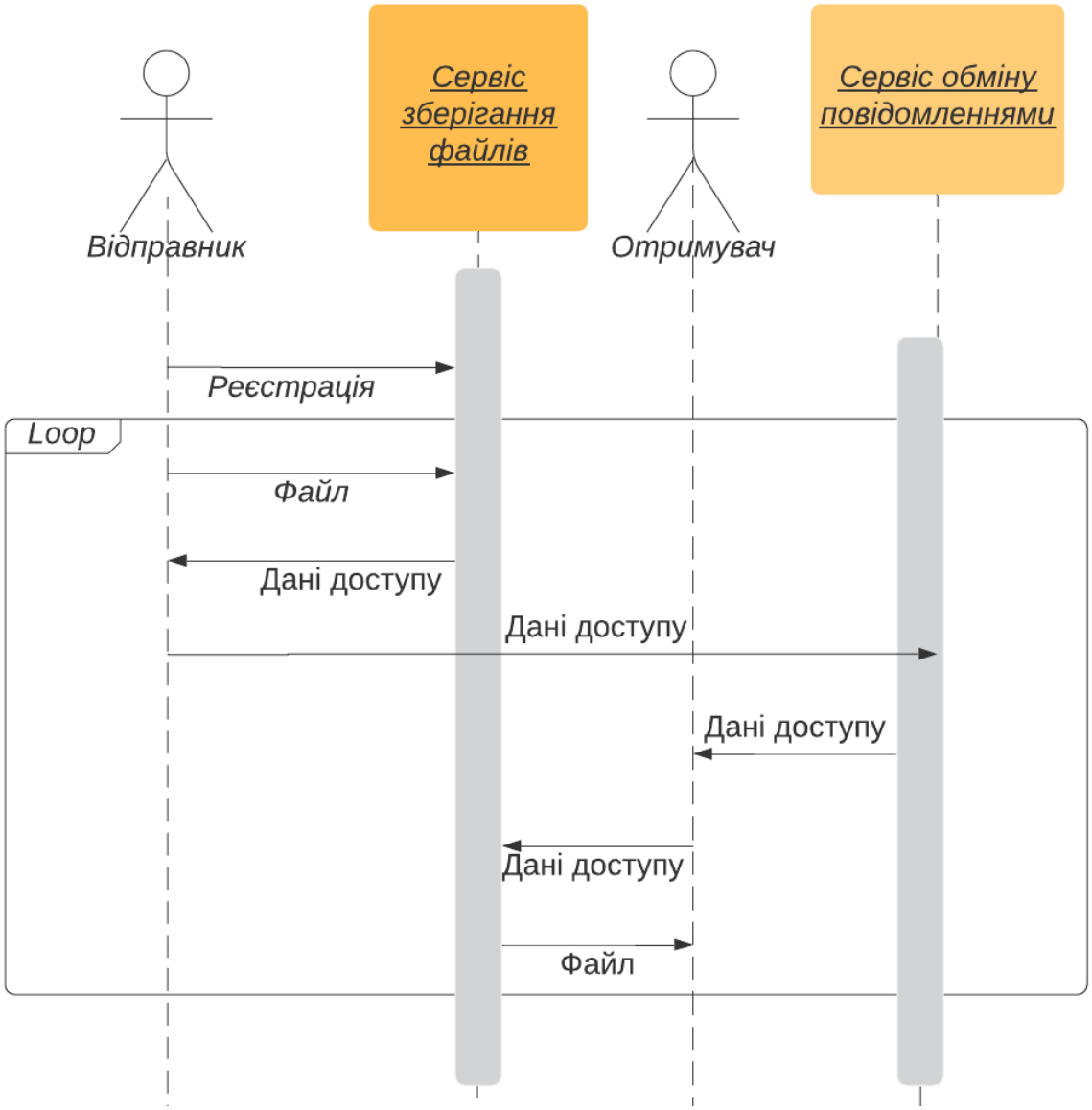
1. Користувач реєструється на сайті
2. Зареєстрований користувач завантажує файл на сервер
3. Файлообмінник генерує посилання (разом із унікальним ключем доступу, якщо необхідно)
4. Власник файлу розповсюджує посилання в інтернеті(разом із ключем доступу, якщо такий існує)
5. Інші користувачі заходять через посилання
6. У разі необхідності вводять ключ доступу
7. Скачують файл, який знаходиться за конкретним посиланням

Сервіс можна використовувати як звичайний файлообмінник. Проте його важливою ознакою є безпека. Питанню захищеності файлів було виділено велике значення. Тому окрім звичайного функціоналу файлообмінника передбачено захищений варіант для зберігання файлів та обміну ними.

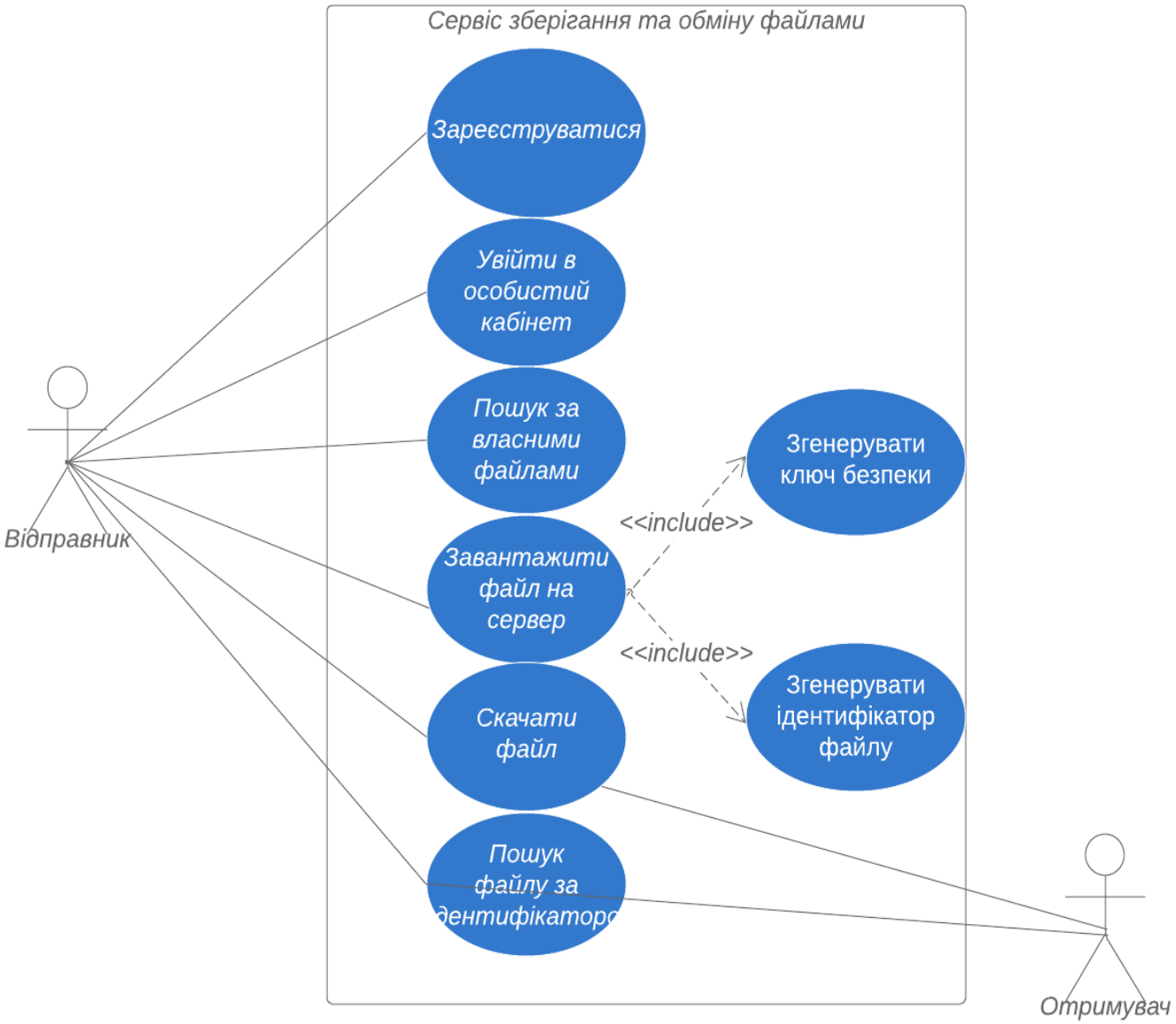
При завантажені файлу на сервер, зареєстрований користувач зможе згенерувати унікальний ключ доступу до конкретного файлу. Завдяки цьому доступ до файлу мають можливість отримати лише ті, хто має цей ключ. Тобто це дозволяє користувачеві самому вирішувати, кому дозволити завантажувати файл.

Окрім цього, варто зауважити ще декілька компонентів, де необхідно було приділити увагу безпеці даних користувача.

Діаграма послідовності



Діаграма сценаріїв використання



Хешування пароля

Під час реєстрації, користувач відправляє дані на сервер, після чого поле з паролем вилучається та хешується за допомогою алгоритму bcrypt. Перед хешуванням на основі унікальних даних користувача генерується додатковий унікальний ключ (сіль), який потім конкатенується з паролем і разом ці дані знову подаються на вхід алгоритму хешування. Після цього дані записуються в базу даних.

Шифрування файлів

Перед завантаженням файлу на сервер він повинен бути зашифрованим і захищеним від «man in the middle attack». Тому ще на клієнтській стороні

кожен файл шифрується за допомогою AES, що генерує шифр за допомогою унікального ключа, згенерованого випадково. Потім за допомогою цього ключа файл можна буде розшифрувати на стороні клієнта, під час його скачування.

Використання JWT

Для авторизації та аутентифікації додаток використовує технологію JSON Web Tokens (пояснення в наступному розділі), яка допомагає впевнитися, що дані, отримані від конкретного клієнта і що вони не були підроблені чи зіпсовані.

3.2. Розробка програми

3.2.1. Технології, що використовувались для розробки

Scala – це мультипарадигмова мова програмування, що поєднує властивості об'єктно-орієнтованого та функційного програмування. Назва Scala утворена зі слів «scalable» (маштабовна) та «language» (мова), для того щоб задекларувати, що мова може рости разом з вимогами користувачів.

Програми мовою Scala виконуються на віртуальній машині Java за умови приєднання до дистрибутиву файлу scala-library.jar. Scala сумісна із існуючими програмами мовою Java, тобто код Scala може викликатися із Java-програм і навпаки. Починаючи з версії 2.11 Scala потребує принаймні Java 6[1], а версія 2.12 потребуватиме Java 8 та матиме кращу інтеграцію із новими можливостями цієї версії Java.

Абстрактні типи в Scala дуже схожі на абстрактні типи сигнатур в SML і OCaml, узагальнені в контексті повноцінних компонентів. Програми багато в чому схожі на Java-програми, і можуть вільно взаємодіяти з Java-кодом. Scala включає одноманітну об'єктну модель в тому сенсі, що будь-яке значення є об'єктом, а будь-яка операція – викликом методу. Це також функціональна мова в тому сенсі, що функції – це повноправні значення.

В Scala включені потужні і одноманітні концепції абстракцій як для типів, так і для значень. Вона містить гнучкі симетричні конструкції домішок для композиції класів і trait-ів. Вона дозволяє виробляти декомпозицію об'єктів шляхом порівняння із зразком. Зразки і вирази були узагальнені для підтримки природної обробки XML-документів. В цілому, ці конструкції дозволяють

легко висловлювати самостійні компоненти, які використовують бібліотеки Scala, не користуючись спеціальними мовними конструкціями.

Мова надає легковаговий синтаксис для визначення анонімних і каррінгових функцій. Кожна конструкція повертає значення. В Scala використовується більшість керуючих структур Java, але традиційне для Java вираз for в їх число не входить. Замість цього існує for-comprehension, який дає можливість прямого перебору елементів масиву (або списку, або перерахування) без необхідності в індексації.

PlayFramework — каркас розробки з відкритим кодом, написаний на Scala і Java, використовує паттерн проектування Модель-Представлення-Контроллер (MVC). Націлений на підвищення продуктивності використовуючи домовленості перед конфігурацією, гаряче перевантаження коду і відображення помилок в браузері. Розробку Play надихнули такі каркаси як Ruby on Rails і Django.

У ядрі присутні наступні функції:

- чиста система RESTful
- модуль для простої аутентифікації користувачів
- схема перевірки на основі анотацій
- JSON та XML-аналізатори та маршаллери
- вбудована база даних для швидкого розгортання / тестування
- функціональність автоматичного завантаження файлів

MySQL — вільна система керування реляційними базами даних. Ця система керування базами даних (СКБД) з відкритим кодом була створена як альтернатива комерційним системам. MySQL з самого початку була дуже схожою на mSQL, проте з часом вона все розширювалася і зараз MySQL — одна з найпоширеніших систем керування базами даних. Вона використовується, в першу чергу, для створення динамічних веб-сторінок, оскільки має чудову підтримку з боку різноманітних мов програмування.

MySQL — компактний багатопотоковий сервер баз даних. Характеризується високою швидкістю, стійкістю і простотою використання.

MySQL вважається гарним рішенням для малих і середніх застосувань. Сирцеві коди сервера компілюються на багатьох платформах. Найповніше можливості сервера виявляються в UNIX-системах, де є підтримка багатопоточності, що підвищує продуктивність системи в цілому.

Можливості сервера MySQL:

- простота у встановленні та використанні;
- підтримується необмежена кількість користувачів, що одночасно працюють із БД;
- кількість рядків у таблицях може досягати 50 млн;
- висока швидкість виконання команд;
- наявність простої і ефективної системи безпеки.

JavaScript – динамічна, об'єктно-орієнтована прототипна мова програмування. Реалізація стандарту ECMAScript. Найчастіше використовується для створення сценаріїв веб-сторінок, що надає можливість на стороні клієнта (пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером.

JavaScript має низку властивостей об'єктно-орієнтованої мови, але завдяки концепції прототипів підтримка об'єктів в ній відрізняється від традиційних мов ООП. Крім того, JavaScript має ряд властивостей, притаманних функціональним мовам, — функції як об'єкти першого класу, об'єкти як списки, каррінг, анонімні функції, замикання(closures).

JavaScript має С-подібний синтаксис, але в порівнянні з мовою Сі має такі корінні відмінності:

- об'єкти, з можливістю інтроспекції і динамічної зміни типу через механізм прототипів
- функції як об'єкти першого класу
- обробка винятків

- автоматичне приведення типів
- автоматичне прибирання сміття
- анонімні функції

HTML (*Hyper Text Markup Language*) — стандартна мова розмітки веб-сторінок в Інтернеті. Більшість веб-сторінок створюються за допомогою мови HTML (або XHTML). Документ HTML оброблюється браузером та відтворюється на екрані у звичному для людини вигляді.
CSS (*Cascading Style Sheets*) — спеціальна мова, що використовується для опису зовнішнього вигляду сторінок, написаних мовами розмітки даних.

Json Web Tokens - це стандарт токена доступу на основі JSON, стандартизованого в RFC 7519. Використовується для верифікації тверджень.

JWT складається з трьох частин: заголовка, вмісту і підпису. Заголовок (*Header*) це JSON елемент, який описує до якого типу токену належить даний і які методи шифрування використовувались. Вміст (англ. *Payload*) складається з елемента JSON який описує твердження. Структура підпису визначається стандартом JSON Web Signature (JWS, RFC 7515).

Щоб згенерувати підпис заголовок та вміст кодуються в Base64, записуються в один рядок через крапку, а потім цей рядок хешується визначеним методом. Заголовок, вміст і підпис кожен кодуються в Base64 і розділяються в токені крапкою.

JWT передається в заголовках HTTP, зазвичай двома способами:

- в полі Authorization як Bearer-Token
- в полі Cookie

3.2.2. Архітектура сервісу

При проектуванні системи файлообмінника було вибрано архітектуру клієнт-сервер. Архітектура клієнт-сервер є одним із архітектурних шаблонів програмного забезпечення та є домінуючою концепцією у створенні

розподілених мережних застосунків і передбачає взаємодію та обмін даними між ними. Вона передбачає такі основні компоненти:

- набір серверів, які надають інформацію або інші послуги програмам, які звертаються до них;
- набір клієнтів, які використовують сервіси, що надаються серверами;
- мережа, яка забезпечує взаємодію між клієнтами та серверами.

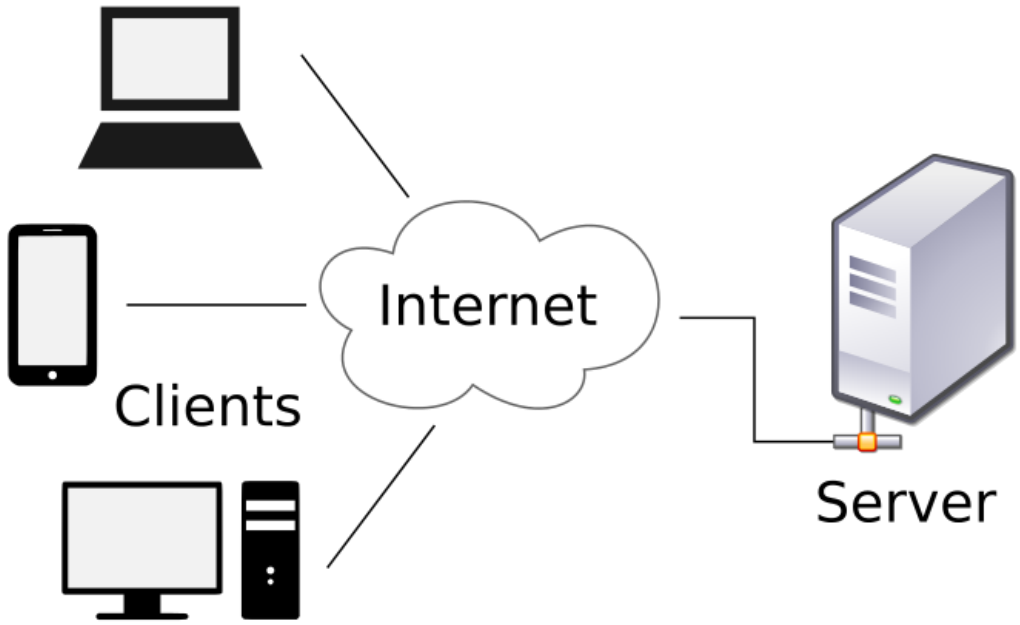


Рис. 3.1. Архітектура «клієнт-сервер»

Модель клієнт-серверної взаємодії визначається перш за все розподілом обов'язків між клієнтом та сервером. Логічно можна відокремити три рівні операцій:

- рівень представлення даних, який по суті являє собою інтерфейс користувача і відповідає за представлення даних користувачеві і введення від нього керуючих команд;
- прикладний рівень, який реалізує основну логіку застосунку і на якому здійснюється необхідна обробка інформації;
- рівень управління даними, який забезпечує зберігання даних та доступ до них.

В данному випадку клієнт та сервер спілкуються за допомогою HTTPпротоколу. HTTP — протокол передачі даних, що використовується в комп'ютерних мережах. Назва скорочена від Hyper Text Transfer Protocol, протокол передачі гіпер-текстових документів.

Основним призначенням протоколу HTTP є передача веб-сторінок (текстових файлів з розміткою HTML), хоча за допомогою нього успішно передаються і інші файли, які пов'язані з веб-сторінками (зображення і застосунки), так і не пов'язані з ними (у цьому HTTP конкурує з складнішим FTP).

HTTP припускає, що клієнтська програма — веб-браузер — здатна відображати гіпертекстові веб-сторінки та файли інших типів у зручній для користувача формі. Для правильного відображення HTTP дозволяє клієнтові дізнатися мову та кодування символіввеб-сторінки й/або запитати версію сторінки в потрібних мові/кодуванні, використовуючи позначення із стандарту MIME.

Також за основу стилю комунікації між клієнтом та сервером був вибраний підхід REST. REST – це стиль архітектури програмного забезпечення для побудови розподілених масштабованих веб-сервісів, заснований на маніпулюванні ресурсами і специфікації HTTP. REST ілюструє розвиток Web архітектури, характеризуючи і регулюючи макро взаємодію чотирьох Web компонентів, а саме серверів, мережевих шлюзів, проксі і клієнтів, без застосування обмежень до індивідуальних учасників. Таким чином, REST по суті визначає правильну поведінку учасників. В конкретному випадку клієнт та сервер комуні кують за допомогою RESTAPI, реалізованому на стороні сервера.

Загальновживаний термін API означає «програмний інтерфейс додатку». Наприклад, API використовується для опису особливостей програмної бібліотеки, способів взаємодії з деякою програмою тощо.

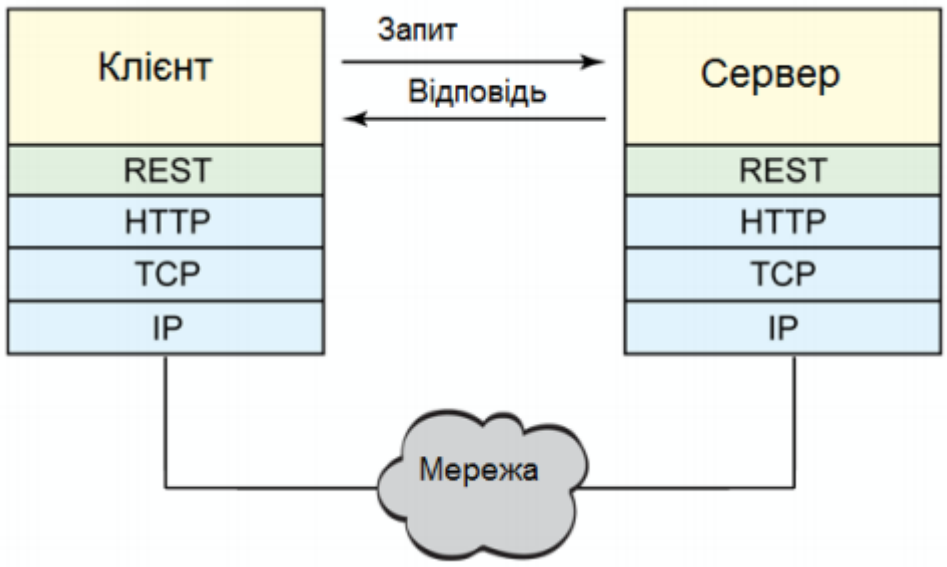


Рис. 3.2. Комунікація за допомогою REST API

Компоненти серверної частини

UserController. На рівні управління запитами цей компонент відповідає за запити, що стосуються користувача. Компонент працює у випадках реєстрації, авторизації та аутентифікації користувача.

ProfileController.Майже те ж саме, що й UserController, але працює на рівень вище, з профілями користувача.

FileController. Цей компонент відповідальний за обслуговування запитів, що стосуються файлів. Він є точкою входу, куди завантажуються файли. Також обслуговує запити щодо метаінформації файлів.

UserService. Цей компонент працює на рівні бізнес логіки. Він виконує основну роботу для обробки, перетворення даних користувача. Тут відбувається хешування паролів. Дану сутність можна назвати посередником між компонентом мережі та компонентом рівня доступу до даних користувача.

ProfileService. Сервіс працює з про файлами на рівні бізнес логіки.

*FileService.*Цей компонент працює з метаданими файлів на рівні бізнес логіки, а також направляє файли в потрібні директорії у файловій системі серверу.

Filema User. Компоненти, що працюють з інформацією про користувачів та їхніх файлів на рівні доступу до даних.

Компоненти клієнтської частини

userform. В клієнтській частині цей компонент реалізовує логіку реєстрації та входу на сторінку до персональних файлів.

profile. Основна логіка роботи з персональними файлами на клієнтській стороні відбувається тут. Тут реалізовано відображення файлів користувача а також маніпуляції над цими файлами. Компонент є відправною точкою, звідки користувач може завантажити файл, згенерувати ключ доступу тощо. Саме тут відбувається шифрування файлів перед відправкою.

repository. Ця сутність відповідає за реалізацію та логіку системи обміну файлами, а саме за скачування файлів при переході по посиланню. Тут відбувається комунікація з сервером, щоб дізнатись, чи правильний ключ доступу ввів користувач. Також цей компонент відповідає за розшифрування файлів та видачу їй користувачам в такому вигляді, у якому він був завантажений власником.

3.2.3. Модель даних

Одним із компонентів архітектури є база даних, яка управляється за допомогою СУБД MySQL. Для комунікації з базою даних в додатку був розроблений окремий рівень доступу до даних (DataAccessLayer). За допомогою цього програма звертається до серверу бази даних із запитом для запису, модифікації або видалення даних з бази.

Розглянемо відношення таблиць в базі даних

Як видно на рисунку, в базі даних знаходяться дві таблиці – UseriFile. Кожен зареєстрований користувач має наступні поля:

1. id(bigintautoincrement) – унікальний ідентифікатор користувача
2. name (varchar (255)) – ім’я користувача
3. email (varchar (255)) – електронна пошта
4. password (varchar (255)) – пароль
5. salt (varchar (255)) – сіль (для хешування паролю з алгоритмом bcrypt)

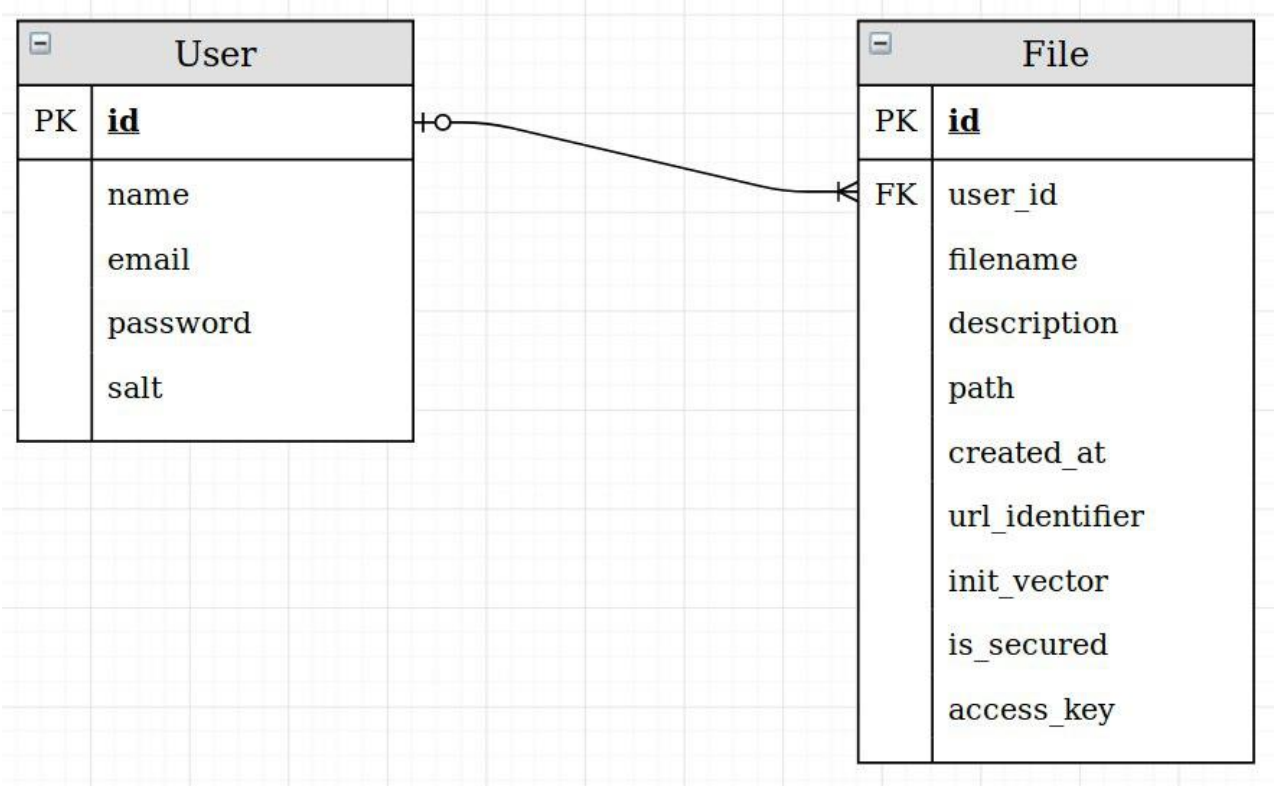


Рис. 3.3. Схема моделі даних

В базі даних також зберігається метаінформація про кожен файл. Ця інформація представлена наступними полями:

1. id (bigint) – унікальний ідентифікатор файлу в базі
2. user_id (bigint) – унікальний ідентифікатор власника файлу
3. filename(varchar (255)) – повна назва файлу з розширенням
4. description (varchar (255)) – опис файлу
5. path(varchar (255)) – шлях до до файлу у файловій системі сервера
6. created_at(varchar (255)) – дата та час завантаження файлу
7. url_identifier(varchar (255)) – унікальна urlадреса, за якою можна скачати файл
8. init_vector(varchar (255)) – ініціюючий вектор (або ключ) для розшифрування контенту файлу
9. is_secured(tinyint) – булеве значення, яке означає чи має даний файл згенерований ключ доступу
10. access_key(varchar (255)) – згенерований ключ доступу до файлу

3.3. Представлення роботи сервісу

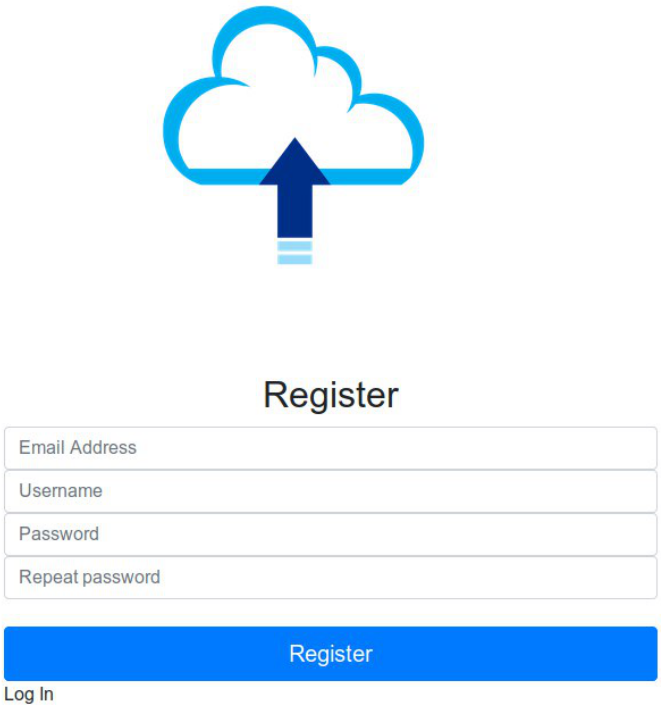


Рис. 3.4. Сторінка реєстрації

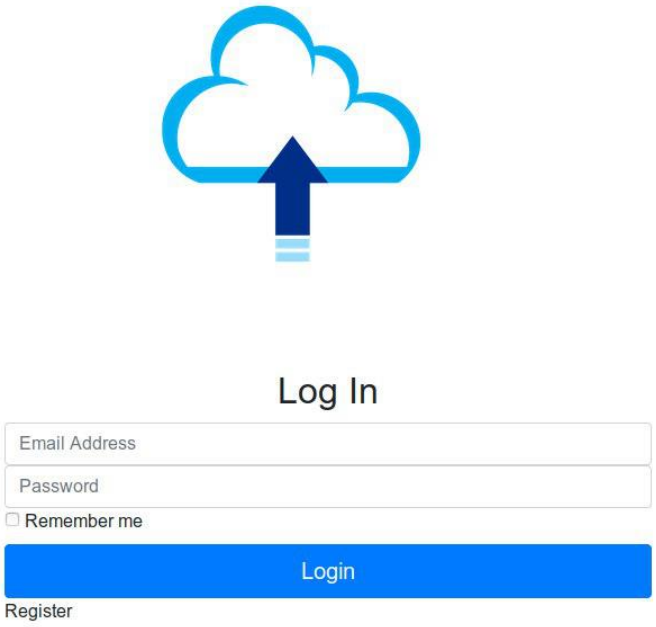


Рис. 3.5. Сторінка входу в приватне сховище

Як вже відомо, графічний інтерфейс був розроблений за допомогою наступних технологій: JavaScript, HTML і CSS. На клієнтській стороні додаток розроблений таким чином, що сторінку не потрібно перезавантажувати після кожної дії користувача, наприклад, у відповідь натискання на кнопку. Такий підхід до розробки веб сторінок називається single page application. Тобто сторінку потрібно завантажити лише раз. Далі контент буде відображатись асинхронно без необхідності запитувати сторінку заново.

Після входу в приватне сховище користувач може отримати доступ до всіх завантажених файлів. На сторінці з файлами можна також добавити новий файл.

На даній сторінці можна здійснювати динамічний пошук по файлах, тобто знову ж сторінку не потрібно перезавантажувати для цього. Окрім цього файл можна без зворотньо видалити з персонального сховища. Кнопка “Newfile” відкриває перед користувачем інтерфейс, де можна завантажити новий файл на сервер.

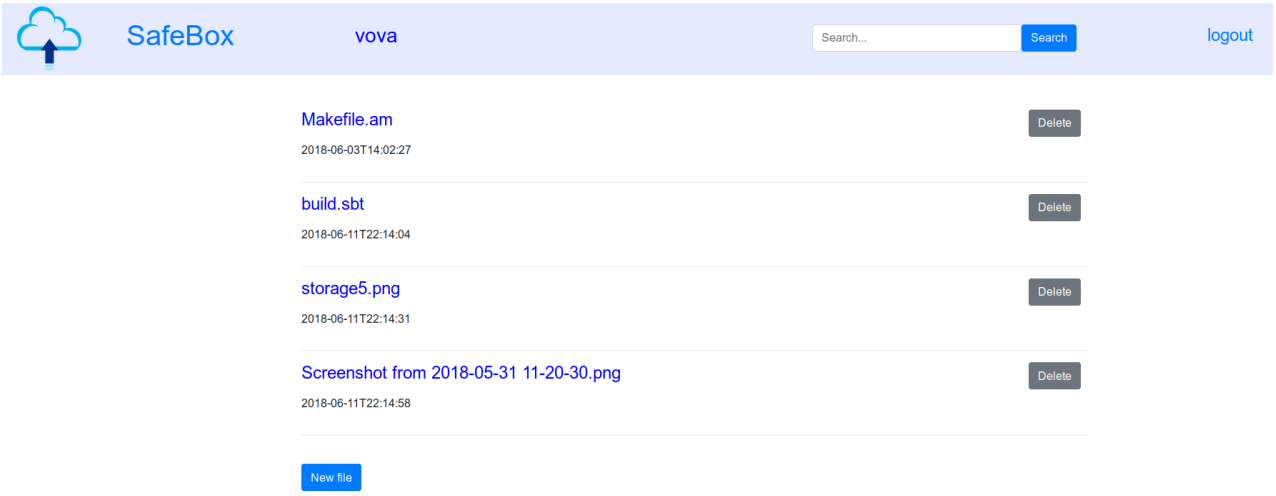


Рис. 3.6. Сторінка приватного сховища

Як видно з рисунка, інтерфейс дозволяє користувачеві вибрати локальний файл на його коп’ютері, написати короткий опис файлу. Якщо необхідно, то можна поставити галочку у полі “Generate security key” для того,

щоб згенерувати унікальний ключ доступу до файлу. Після натискання на кнопку “Upload” файл буде зашифрований та завантажений на сервер.

Рис. 3.7. Графічний інтерфейс завантаження файлу на сервер

Після завантаження файлу в хмару, користувачу відкриється нове вікно з інформацією про файл, яку ввів його власник, а також додатковими полями – посиланням на файл, за яким можна його скачати, а також унікальний згенерований ключ доступу.

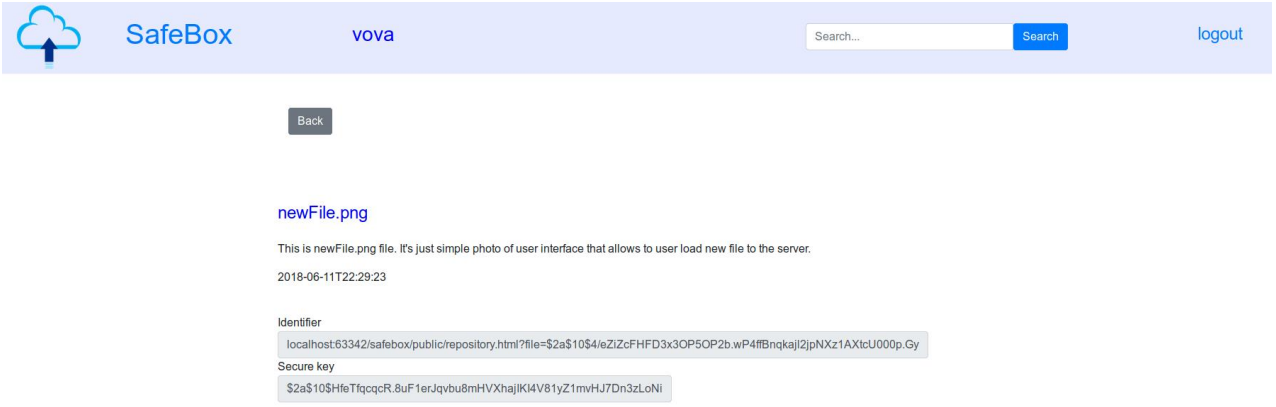


Рис. 3.8. Сторінка з інформацією про завантажений файл

Окрім назви файлу в цьому інтерфейсі також є опис файлу, а також дата завантаження файлу. Додатково на цю сторінку можна потрапити, натиснувши на файл у меню зі списком файлів.

Якщо скопіювати та вставити посилання якогось файлу у браузерну строку пошуку, то користувача перенаправить на сторінку з конкретним файлом, якщо такий існує. Якщо силка неправильна, то з’явиться сторінка з помилкою 404, що означає – ресурс не був знайдений. На цій сторінці також можна побачити назву файлу, його опис та дату. Якщо файл не має згенерованого ключа доступу, то натиснувши на кнопку “Download”, скачування файлу почнеться. У разі наявності ключа доступу, на сторінці також присутнє поле, куди його потрібно ввести. Якщо ключ вірний, то після натискання кнопки скачування, файл буде завантажений, якщо ні – нічого не відбудеться.

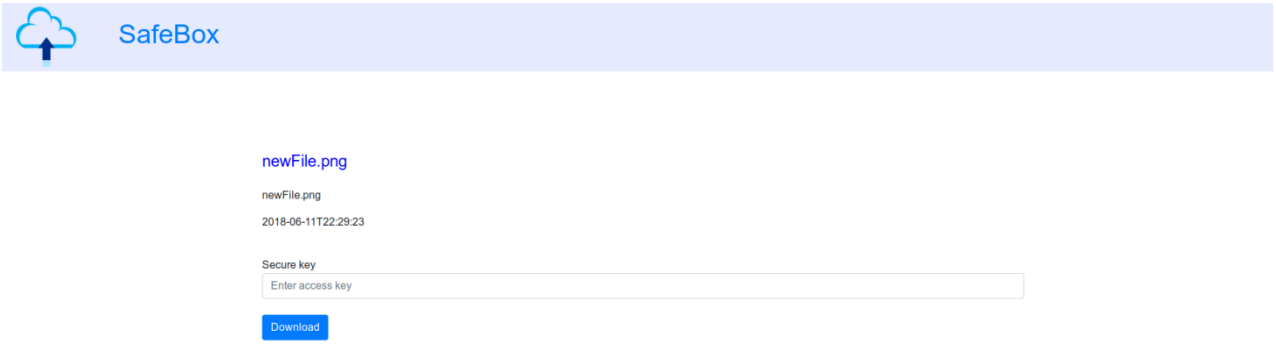


Рис. 3.9. Сторінка завантаження файлу

ВИСНОВКИ

Під час виконання дипломного проекту були досліджені різні методи та засоби захисту інформації. Було пояснено та обгрунтовано актуальність та сучасні проблеми у сфері безпеки приватних даних.

Проведено дослідження та вивченно різноманітні застосунки, ціль котрих – зберігати файли користувача та надавати цілодобовий доступ до них. Однією із найпоширеніших тенденцій у цій сфері є використання хмарних технологій. Основними перевагами хмарних рішень є:

- не потрібні великі обчислювальні потужності ПК - по суті будь-який смартфон, планшет і т.д., при відкритті вікна браузера отримує величезний потенціал.
- відмовостійкість;
- певний рівень безпеки;
- висока швидкість обробки даних;
- економія на покупці софта - всі необхідні програми вже є в сервісі, де будуть працювати додатки;
- Ваш власний вінчестер не наповнюється - всі дані зберігаються в мережі.

Проведено дослідження найбільш популярних алгоритмів шифрування, а також були розглянуті стандарти, що є загальноприйнятими (AES, DES). Після проведених досліджень можна зробити висновок, що криптографія та методи захисту даних з кожним роком вдосконалюються, а це в свою чергу робить можливість несанкціонованого доступу до персональної і закритої інформації складнішим.

Після аналізу всіх розглянутих тем, було вирішено зробити безпечний сервіс зберігання та обміну файлами, яке б охоплювало всі вищезгадані теми. Однією із основних задач при проектуванні було зробити систему якомога простішою для користувача, оскільки існуючі продукти є занадто перевантаженими додатковими функціями.

При виборі інструментів для реалізації були вибрані нові й водночас потужні технології (мова програмування Scala, PlayFramework), а також відносно давні, проте стабільні, що добре зарекомендували себе під час років розробки (СУБД MySQL, мови JavaScript, HTML, CSS).

Як кінцевий результат дипломного проекту було розроблено просту, але в той же час надійну систему, яка дозволяє безпечно зберігати файли та обмінюватись ними.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Andress, J. (2014). *The Basics of Information Security: Understanding the Fundamentals of InfoSec in Theory and Practice*. Syngress. p. 240. ISBN 9780128008126.

2."*What is Information Security? (with pictures)*". *wiseGEEK*.Retrieved 2017-10-06.

3. Schlienger, Thomas; Teufel, Stephanie (2003). "Information security culture-from analysis to change". *SouthAfricanComputerJournal*. 31: 46–52.

4.Зегжды,П.Д. *Теория и практика обеспечения информационной безопасности* / П. Д. Зегжды. – М.: Яхтсмен, 2002

5.Антонюк, А. О. *Основи захисту інформації в автоматизованих системах* / А. О. Антонюк.– К.: КМ Академія, 2003

6.*Вербіцький О. В. Вступ до криптології.* — Л. : ВНТЛ, 1998. — 248 с.

7.Alfred J. Menezes; Paul C. van Oorschot; Scott A. Vanstone (August 2001). *HandbookofAppliedCryptography* CRCPress. ISBN 0-8493-8523-7.

8. *Бауэр Ф. Расшифрованные секреты.* — М. : Мир, 2007. — 550 с.

9. *Мао В. Современная криптография.* — М. : Диалектика, 2005. — 768 с.

10. *Oleshchuk V. A. On Public-Key Cryptosystem Based on Church-Rosser String-Rewriting Systems // Computing and Combinatorics: First Annual International Conference,* COCOON '95. — Springer, 1995. — С. 264–269

11. Suetonius Tranquillus, Gaius (2008). *Lives of the Caesars (Oxford World's Classics)*. New York: Oxford University Press. p. 28. ISBN 978-0-19-953756-3.

Додаток А

FileController.scala

```
importcom.google.inject._
importdto.Error._
importdto.FileMetadata._
importdto.JsonResult._
importdto._
importplay.api.libs.json.{JsError,
JsSuccess, Json}
importservices.FileManagerService
importscala.concurrent.Future
importdto.User._
importplay.api.mvc.Results.Unauthorized
importscala.concurrent.ExecutionContext.Impl
icits.global
classFileController@Inject() (scc:
SecuredControllerComponents,
fileManager: FileManagerService)
extendsSecuredController(scc) {
defupload=AuthenticatedAction(parse.multipar
tFormData).async { request =>
Future (
      request.body.file("newFile").map {
multipart =>

fileManager.saveUploadedTempFile(multipart,
request.session.get("id").get.toString)
match {
caseSome(fileMetadata) =>
Ok(Json.toJson(JsonResult(Success.name,
Json.toJson(fileMetadata).toString())))
caseNone=>
InternalServerError(Json.toJson(JsonResult(U
nsuccess.name, Error("File could not be
uploaded.")))
        }
      }.getOrElse {
BadRequest(Json.toJson(JsonResult(Unsuccess.
name, Error("Missing file"))))
      }
    )
  }
defmetaUpload=AuthenticatedAction(parse.json
).async( request =>
Future (
      request.body.validate[FileMetadata]
match {
casee: JsError=>
BadRequest(Json.toJson(JsonResult(Unsuccess.
name, Error("Json validation error"))))
casefileMetadata: JsSuccess[FileMetadata] =>
val userId = request.session.get("id").get
val fileData = fileMetadata.get

fileManager.updateUserUploadedFileData(userI
d, fileData.filename, fileData.description,
fileData.initVector, fileData.isSecured)
match {
caseNone=>
BadRequest(Json.toJson(JsonResult(Unsuccess.
name, Error("Something went wrong try
again."))))
caseSome(f) =>
Ok(Json.toJson(JsonResult(Success.name,
Json.toJson(f).toString())))
          }
      }
    )
  )
deffiles=AuthenticatedAction.async { request
=>
Future (
      request.session.get("id") match {
caseSome(id) =>
val files =
fileManager.getUserFiles(id.toLong)
Ok(Json.toJson(JsonResult(Success.name,
Json.toJson(files).toString())))
case _ =>
Unauthorized("Authorize for this type
requests.")
      }
    )
  }
deffile=Action(parse.json).async { request
=>
Future (
      request.body.validate[FileMetadata]
match {
casee: JsError=>
BadRequest(Json.toJson(JsonResult(Unsuccess.
name, Error("Json validation error"))))
casefileMetadata: JsSuccess[FileMetadata] =>
```

```
val fileData = fileMetadata.get

fileManager.getUserFileByIdentifierSecured(f
ileData.urlIdentifier, fileData.accessKey)
match {
caseNone=>
BadRequest(Json.toJson(JsonResult(Unsuccess.
name, Error("Something went wrong try
again."))))
caseSome(f) =>
Ok.sendFile(f)
          }
      }
    )
  }
defmetaDownload(file: String) =Action.async
{ request =>
Future (

fileManager.getUserFileByIdentifier(file)
match {
caseNone=>
BadRequest(Json.toJson(JsonResult(Unsuccess.
name, Error("Something went wrong try
again."))))
caseSome(f) =>
Ok(Json.toJson(JsonResult(Success.name,
Json.toJson(f.copy(accessKey
="")).toString())))
      }
    )
  }
defdelete=AuthenticatedAction(parse.json).as
ync { request =>
Future(
      request.body.validate[FileMetadata]
match {
casee: JsError=>
BadRequest(Json.toJson(JsonResult(Unsuccess.
name, Error("Json validation error"))))
casefileMetadata: JsSuccess[FileMetadata] =>
val fileData = fileMetadata.get
          fileManager.deleteFile(fileData)
match {
caseNone=>
BadRequest(Json.toJson(JsonResult(Unsuccess.
name, Error("Something went wrong try
again."))))
caseSome(f) =>
Ok
          }
      }
    )
  }
}
```

HomeController.scala

```
importjavax.inject._
importplay.api.mvc._
/**
 * This controller creates an `Action` to
handle HTTP requests to the
 * application's home page.
 */
@Singleton
classHomeController@Inject()(cc:
ControllerComponents)
extendsAbstractController(cc) {
defindex=Action {
Ok(views.html.index("Your new application is
ready."))
  }
}
```

ProfileController.scala

```
importcom.google.inject._
importdto.{Error, JsonResult, Success,
Unsuccess}
importplay.api.libs.json.Json
importservices.ProfileService
importscala.concurrent.Future
importdto.Profile._
importscala.concurrent.ExecutionContext.Impl
icits.global
classProfileController@Inject() (scc:
SecuredControllerComponents, profileService:
ProfileService)
extendsSecuredController(scc:
SecuredControllerComponents) {
defprofile(name: String)
```

```
=AuthenticatedAction.async { implicit
request =>
Future (
      profileService.asembleProfile(name)
match {
caseNone=>
NotFound(Json.toJson(JsonResult(Unsuccess.na
me, Error("There is no such user"))))
caseSome(u) =>
Ok(Json.toJson(JsonResult(Success.name,
Json.toJson(u).toString())))
      }
    )
  }
}
```

Secured.scala

```
nportjavax.inject.Inject
nportplay.api.http.FileMimeTypes
nportplay.api.i18n.{Langs, MessagesApi}
nportplay.api.mvc.Results._
nportplay.api.mvc._
nportscala.concurrent.{ExecutionContext,
     Future}
lassAuthenticatedRequest[A](val
     userId:String, request: Request[A])
     extendsWrappedRequest[A](request)
lassAuthenticatedActionBuilder@Inject()(
     parser:
     BodyParsers.Default)(implicitec:
     ExecutionContext)
ctendsActionBuilderImpl(parser) {
verridedefinvokeBlock[A](request:
     Request[A], block: Request[A]
     =>Future[Result]):Future[Result] =
     {
   request.session.get("id") match {
aseSome(id) =>

     block(newAuthenticatedRequest[A](id
     , request))
ase _ =>
uture(Unauthorized("Authorize for this
     type requests."))
   }
 }

aseclassSecuredControllerComponents@Inje
     ct()( authenticatedActionBuilder:
     AuthenticatedActionBuilder,
ctionBuilder: DefaultActionBuilder,
arsers: PlayBodyParsers,
essagesApi: MessagesApi,
angs: Langs,
ileMimeTypes: FileMimeTypes,
ecutionContext:
     scala.concurrent.ExecutionContext

     ) extendsControllerComponents
lassSecuredController@Inject()(scc:
     SecuredControllerComponents)
     extendsAbstractController(scc) {
efAuthenticatedAction:AuthenticatedActio
     nBuilder=
     scc.authenticatedActionBuilder
```

UserController.scala

```
importcom.google.inject._
importdto._
importplay.api.mvc.{AbstractController,
ControllerComponents}
importdto.User._
importdto.Error._
importplay.api.libs.json.{JsError,
JsSuccess, Json}
importservices.{ProfileService,
UserService}
importdto.JsonResult._
importscala.concurrent.ExecutionContext.
Implicits.global
importscala.concurrent.Future
classUserController@Inject() (cc:
ControllerComponents, userService:
UserService, profileService:
ProfileService)
extendsAbstractController(cc:
ControllerComponents) {
defregister=Action(parse.json).async {
request =>
```

```
Future (
      request.body.validate[User] match
{
casee: JsError=>
BadRequest(Json.toJson(JsonResult(Unsucc
ess.name, Error("Json validation
error"))))
caseuser: JsSuccess[User] =>

userService.createUser(user.get) match {
caseNone=>
BadRequest(Json.toJson(JsonResult(Unsucc
ess.name, Error("There is such
email"))))
caseSome(u) =>
Ok(Json.toJson(JsonResult(data = u)))
        }
      }
    )
  }
deflogin=Action(parse.json).async {
implicit request =>
Future (
      request.body.validate[User] match
{
casee: JsError=>
BadRequest(Json.toJson(JsonResult(Unsucc
ess.name, Error("Json validation
error"))))
caseuser: JsSuccess[User] =>

userService.validateUser(user.get) match
{
caseNone=>
BadRequest(Json.toJson(JsonResult(Unsucc
ess.name, Error("Invalid data"))))
caseSome(u) =>
val profile =
profileService.formUserProfile(u)
Ok(Json.toJson(JsonResult(Success.name,
Json.toJson(profile).toString())))
                .withSession(("id",
u.id.toString), ("name", u.name),
("email", u.email))
          }
      }
    )
```

FileManagerService.scala

```
importcom.google.inject._
importcom.github.t3hnar.bcrypt._
importscala.reflect.io.Path
importjava.io.{File=>JFile}
importjava.nio.file._
importjava.nio.file.attribute.PosixFilePe
rmission
importjava.nio.file.{Files=>JFiles}
importjava.time.LocalDateTime
importjava.util.Date
importrepository.{Files=>MetaFiles}
importcom.typesafe.config.Config
importdto.FileMetadata
importorg.apache.commons.lang3.SystemUtil
s
importplay.api.libs.Files.TemporaryFile
importutils.Utils
importdto.FileMetadata._
importplay.api.mvc.MultipartFormData
importscala.collection.JavaConversions
@Singleton
classFileManagerService@Inject()(config:
Config) {
val uploadPath
=Path(config.getString("uploadDir"))
val webPath =
config.getString("uploadDir")
val hostnameUrl =
config.getString("hostnameUrl")
typeWebPath=Long=>String
val mislaidPath:WebPath="/storage/"+
_.toString +"/files"
defmislaidPhotoPath(mislaidId:
Long):String=>String=
mislaidPath(mislaidId) +"/"+ _
defmislaidPhotoUrl(mislaidId:
Long):String=>String= hostnameUrl +
webPath + mislaidPhotoPath(mislaidId)(_)
defgetUserFiles(userId:
Long):List[FileMetadata]
=MetaFiles.filesByUserId(userId).map(file
Repo2Dto)
defsaveUploadedTempFile(file:
```

```
MultipartFormData.FilePart[TemporaryFile]
, userId: String):Option[FileMetadata] =
{
val pathDir = (uploadPath /
userId).createDirectory()
val filename = file.filename
val fullPath =
s"${pathDir.path}/$filename"
val exists
=Files.exists(Paths.get(fullPath))
if (exists) {
MetaFiles.filesByUserId(userId.toLong).he
adOption.map(fileRepo2Dto)
    } else {

file.ref.moveTo(Paths.get(fullPath),
replace =true)
Some(MetaFiles.create(userId.toLong,
filename, "", fullPath,
LocalDateTime.now(),
hashedFilename(filename, userId), "",
false, ""))
    }
  }
defupdateUserUploadedFileData(userId:
String, filename: String, description:
String, initVector: String, isSecured:
Boolean):Option[FileMetadata] = {
val filePath =
s"$uploadPath/$userId/$filename"
val exists
=Files.exists(Paths.get(filePath))
if (exists) {
val accessKey =if(isSecured)
generateAccessKey(userId, filename)
else""
MetaFiles.updateUserFileMetadata(userId.t
oLong, filename, description, initVector,
isSecured, accessKey)
MetaFiles.findUserFile(userId.toLong,
filename).map(fileRepo2Dto)
    } else {
MetaFiles.deleteUserFileMetadata(userId.t
oLong, filename)
None
    }
  }
defgetUserFileByIdentifierSecured(identif
ier: String, key:
String):Option[java.io.File] = {
MetaFiles.getUserFileByIdentifierAndKey(i
dentifier, key).map(f
=>newjava.io.File(f.path))
  }
defgetUserFileByIdentifier(identifier:
String):Option[FileMetadata] = {
MetaFiles.getFileByIdentifier(identifier)
.map(fileRepo2Dto)
  }
privatedefhashedFilename(filename:
String, userId: String):String= {
    (filename + userId).bcrypt
  }
privatedefgenerateAccessKey(userId:
String, filename: String):String= {
    (userId + filename).bcrypt
  }
defdownloadFile(url: String, toName:
String=>String, directoryName: String):
(String, JFile) = {
val filename =
url.substring(url.lastIndexOf("/") +1)
val originalFileId =
toName(filename).toLowerCase
val pathDir = (uploadPath /
directoryName).createDirectory()
val filePath = pathDir / originalFileId
Utils.fileDownloader(url, filePath)
    (filename, filePath.jfile)
  }
defdeleteFile(fileMetadata:
FileMetadata):Option[FileMetadata] = {
MetaFiles.getFileByIdentifier(fileMetadat
a.urlIdentifier) match {
caseSome(fileRepo) =>
val file =newJFile(fileRepo.path)
        file.delete()
MetaFiles.deleteFileById(fileRepo.id)
Some(fileMetadata)
caseNone=>None
    }
  }
}
```

ProfileService.scala

```
importcom.google.inject._
importdto.{Profile, User}
@Singleton
classProfileService@Inject()
(userService: UserService,
fileManagerService: FileManagerService)
{
defasembleProfile(name:
String):Option[Profile] = {
val user =
userService.getUserWithName(name)
val profile = user match {
caseSome(user) =>
val files =
fileManagerService.getUserFiles(user.id)
Some(Profile(user, files))
caseNone=>
None
    }
    profile
  }
defformUserProfile(user: User):Profile=
{
val files =
fileManagerService.getUserFiles(user.id)
Profile(user, files)
```

UserService.scala

```
importdto.User
importcom.github.t3hnar.bcrypt._
importcom.google.inject._
importrepository.Users
importUser.repo2Dto
@Singleton
classUserService {
defcreateUser(user: User):Option[User] = {
Users.findByEmail(user.email).orElse(Users
.findByName(user.name)) match {
caseNone=>
Some(Users.create(user.name, user.email,
hashePassword(user), generateSalt(user)))
caseSome(_) =>
None
    }
  }
defvalidateUser(user: User):Option[User] =
{
Users.findByEmail(user.email).orElse(Users
.findByName(user.name)) match {
caseSome(u) if
user.password.isBcrypted(u.password) =>
Some(u)
case _ =>None
    }
  }
defgetUserWithName(name:
String):Option[User] = {
Users.findByName(name) match {
caseSome(u) =>
Some(u)
case _ =>None
    }
  }
defgetUserWithId(id: Long):Option[User] =
{
Users.findById(id) match {
caseSome(user) =>
Some(user)
caseNone=>
None
    }
  }
//use if checked that user data is unique
privatedefgenerateSalt(user: User):String=
{
    (user.name + user.email +
user.password).bcrypt
  }
privatedefhashePassword(user:
User):String= {

user.password.bcrypt(generateSalt(user))
  }
}
```

Files.scala

```
importjava.time.LocalDateTime
importjava.util.Date
importscalikejdbc._
importscalikejdbc.config.DBs
caseclassFiles(id: Long,
userId: Long,
filename: String,
description: String,
path: String,
createdAt: LocalDateTime,
```

```
urlIdentifier: String,
initVector: String,
isSecured: Boolean,
accessKey: String)
objectFilesextendsSQLSyntaxSupport[Fi
les] {
DBs.setupAll()
defapply(c:
SyntaxProvider[Files])(rs:
WrappedResultSet):Files=
apply(c.resultName)(rs)
defapply(c: ResultName[Files])(rs:
WrappedResultSet):Files=newFiles(
    id = rs.get(c.id),
    userId = rs.get(c.userId),
    filename = rs.get(c.filename),
    description =
rs.get(c.description),
    path = rs.get(c.path),
    createdAt = rs.get(c.createdAt),
    urlIdentifier =
rs.get(c.urlIdentifier),
    initVector =
rs.get(c.initVector),
    isSecured = rs.get(c.isSecured),
    accessKey = rs.get(c.accessKey)
  )
val f =Files.syntax("files")
defcreate(userId: Long,
filename: String,
description: String,
path: String,
createdAt: LocalDateTime,
urlIdentifier: String,
initVector: String,
isSecured: Boolean,
accessKey: String)(implicitsession:
DBSession= autoSession):Files= {
val id = withSQL {
      insert.into(Files).namedValues(
        column.userId -> userId,
        column.filename -> filename,
        column.description ->
description,
        column.path -> path,
        column.createdAt ->
createdAt,
        column.urlIdentifier ->
urlIdentifier,
        column.initVector ->
initVector,
        column.isSecured ->
isSecured,
        column.accessKey -> accessKey
      )

}.updateAndReturnGeneratedKey().apply
()
Files(id, userId, filename,
description, path, createdAt,
urlIdentifier, initVector, isSecured,
accessKey)
  }
deffilesByUserId(userId:
Long)(implicitsession: DBSession=
autoSession):List[Files] = withSQL {
    select.from(Files as
f).where.eq(f.userId, userId)
  }.map(Files(f)).list().apply()
deffindUserFile(userId: Long,
filename: String)(implicitsession:
DBSession= autoSession):Option[Files]
= withSQL {
    select.from(Files as
f).where.eq(f.userId,
userId).and.eq(f.filename, filename)
  }.map(Files(f)).single().apply()
defupdateUserFileMetadata(userId:
Long, fileName: String, description:
String, initVector: String,
isSecured: Boolean, accessKey:
String)

(implicitsession: DBSession=
autoSession):Int= withSQL {
    update(Files as f).set(
      f.description -> description,
      f.initVector -> initVector,
      f.isSecured -> isSecured,
      f.accessKey -> accessKey)
      .where.eq(f.userId,
userId).and.eq(f.filename, fileName)
  }.update().apply()
defdeleteUserFileMetadata(userId:
Long, filename:
String)(implicitsession: DBSession=
```

```
autoSession) = withSQL {
    delete.from(Files as
f).where.eq(f.userId,
userId).and.eq(f.filename, filename)
  }
defdeleteFileById(fileId:
Long)(implicitsession: DBSession=
autoSession) = withSQL {
    delete.from(Files as
f).where.eq(f.id, fileId)
  }.update().apply()
defgetUserFileByIdentifierAndKey(iden
tifier: String, key:
String)(implicitsession: DBSession=
autoSession):Option[Files] = withSQL
{
    select.from(Files as
f).where.eq(f.urlIdentifier,
identifier).and.eq(f.accessKey, key)
  }.map(Files(f)).single().apply()
defgetFileByIdentifier(identifier:
String)(implicitsession: DBSession=
autoSession):Option[Files] = withSQL
{
    select.from(Files as
f).where.eq(f.urlIdentifier,
identifier)
  }.map(Files(f)).single().apply()
}
```

Users.scala

```
importscalikejdbc._
importscalikejdbc.config._
caseclassUsers(id: Long,
name: String,
email: String,
password: String,
salt: String)
objectUsersextendsSQLSyntaxSupport[Use
rs]{
defapply(c: SyntaxProvider[Users])(rs:
WrappedResultSet):Users=apply(c.result
Name)(rs)
defapply(c: ResultName[Users])(rs:
WrappedResultSet):Users=newUsers(
    id = rs.get(c.id),
    name = rs.get(c.name),
    email = rs.get(c.email),
    password = rs.get(c.password),
    salt = rs.get(c.salt)
  )
DBs.setupAll()
val u =Users.syntax("users")
defcreate(name: String, email: String,
password: String, salt:
String)(implicitsession: DBSession=
autoSession):Users= {
val id = withSQL {
      insert.into(Users).namedValues(
        column.name -> name,
        column.email -> email,
        column.password -> password,
        column.salt -> salt
      )

}.updateAndReturnGeneratedKey().apply(
)
Users(id, name, email, password, salt)
  }
deffindByEmail(email:
String)(implicitsession: DBSession=
autoSession):Option[Users] = withSQL {
    select.from(Users as
u).where.eq(u.email, email)
  }.map(Users(u)).single.apply()
deffindByName(name:
String)(implicitsession: DBSession=
autoSession):Option[Users] = withSQL {
    select.from(Users as
u).where.eq(u.name, name)
  }.map(Users(u)).single.apply()
deffindById(id: Long)(implicitsession:
DBSession= autoSession):Option[Users]
= withSQL {
    select.from(Users as
u).where.eq(u.id, id)
  }.map(Users(u)).single.apply()
}
```

Profile.js

```
var profileJson =
localStorage.getItem("profile");
var profile =JSON.parse(profileJson);
var user = profile.user;
```

```
var files = [];
//server config
var host ="localhost";
var port ="9000";
var serverName ="http://"+ host +":"+
port;
document.getElementById("newFileButton").
addEventListener("click",
showNewFileForm);
requestUserFiles();
document.getElementById("logo").addEventL
istener("click", function () {
cleanFiles();
setProfile();
});
document.getElementById("backButton").add
EventListener("click", function () {
cleanFiles();
setProfile();
});
functioncleanFiles() {
    files.forEach(function (file) {
var node
=document.getElementById("fileNode"+
file.id);
        node.style.display="none";
        filesContainer.removeChild(node);
        node
=document.getElementById("hr-fileNode"+
file.id);
        filesContainer.removeChild(node);
    })
}
functionrequestUserFiles(user) {
var req =request("GET", serverName
+"/my/files");
    req.onload=function () {
if(req.status===200) {
var data =JSON.parse(req.response);
            files =JSON.parse(data.data);
setProfile();
        } else {
console.log("Error ocured while uploading
files.")
        }
    }
}
functionshowNewFileForm() {
document.getElementById("fileListContaine
r").style.display="none";
document.getElementById("fileContainer").
style.display="none";
document.getElementById("newFileFormConta
iner").style.display="block";
document.getElementById("backButton").sty
le.display="block";
}
functionsetProfile() {
document.getElementById("profileName").in
nerHTML = profile.user.name;
document.getElementById("newFileFormConta
iner").style.display="none";
document.getElementById("fileContainer").
style.display="none";
document.getElementById("fileListContaine
r").style.display="block";
document.getElementById("backButton").sty
le.display="none";
fileList();
}
functionfileList() {
if(files.length>0) {
        files.forEach(function (file) {
var html =
'<div class="col">'+
'   <h4><p
style="cursor:pointer;color:blue;"
id="file'+ file.id+'">'+
file.filename+'</p></h4>\n'+
'   <p>'+ file.createdAt +'</p>\n'+
'</div>'+
'<div class="col-1">'+
'   <button id="fileDelete'+ file.id+'"
type="button" class="btn btn-
secondary">Delete</button>'+
'</div>';
addFileElement("filesContainer", "div",
"fileNode"+ file.id, html);
document.getElementById("file"+
file.id).addEventListener("click",
function() {
document.getElementById("fileListContaine
r").style.display="none";
document.getElementById("newFileFormConta
iner").style.display="none";
document.getElementById("fileContainer").
```

```
style.display="block";
document.getElementById("filename").inner
HTML = file.filename;
document.getElementById("fileDescription"
).innerHTML = file.description;
document.getElementById("createdAt").inne
rHTML = file.createdAt;
document.getElementById("identifier").inn
erHTML =

location.hostname+":"+
location.port+"/safebox/public"+"/reposit
ory.html?file="+ file.urlIdentifier;
var keyField
=document.getElementById("secureKey");
if (file.isSecured ===false) {

keyField.style.display="none";
                } else {
document.getElementById("key").innerHTML
= file.accessKey;
                }
document.getElementById("backButton").sty
le.display="block";
            });
document.getElementById("fileDelete"+
file.id).addEventListener("click",
function () {
deleteFile(file);
            })
        })
    }
}
functiondeleteFile(file) {
var req =requestJson("DELETE", serverName
+"/file", file);
    req.onload=function (ev) {
if (req.status===200) {
var node
=document.getElementById("fileNode"+
file.id);
            node.remove();
            node
=document.getElementById("hr-fileNode"+
file.id);
            node.remove();
        } else {
setProfile();
console.log("Error occured while deleting
file.")
        }
    }
}
functionaddFileElement(parentId,
elementTag, elementId, html) {
// Adds an element to the document
var p =document.getElementById(parentId);
var newElement
=document.createElement(elementTag);
    newElement.setAttribute('id',
elementId);
    newElement.setAttribute('class',
'row');
    newElement.innerHTML = html;
    p.appendChild(newElement);
var hr =document.createElement("hr");
    hr.setAttribute("id", "hr-"+
elementId);
    hr.style.width="102%";
    p.appendChild(hr);
}
var fileSelect
=document.getElementById("fileSelect");
var uploadButton
=document.getElementById("uploadButton");
document.getElementById("uploadButton").a
ddEventListener("click", sendFile);
functionsendFile() {
    uploadButton.innerHTML
='Uploading...';
var files = fileSelect.files;
if(files.length===0) {
        uploadButton.innerHTML ='Upload';
return;
    }
    reader =newFileReader();
var file = files[0];
    reader.onload=function(e) {
var data = e.target.result;
        (async () => {
var mode ='AES-GCM',
                length =256,
                ivLength =12;
var encrypted =awaitencrypt(data,
'password', mode, length, ivLength);
console.log(encrypted); // { cipherText:
```

```
ArrayBuffer, iv: Uint8Array }
var enc =newTextDecoder("utf-8");
var encryptedFile
=newFile([encrypted.cipherText],
file.name);
var formData =newFormData();
            formData.append("newFile",
encryptedFile, file.name);
var req =sendFileData("POST", serverName
+"/upload", formData);
            req.onload=function () {
if (req.status===200) {
var description
=document.getElementById("description").v
alue;
var checkBox
=document.getElementById("isSecured");
var isSecured;
if (checkBox.checked===true) {
                        isSecured =true;
                    } else {
                        isSecured =false;
                    }
console.log(encrypted.iv);
var meta =newFileMetadata(0, user.id,
file.name, description, Date.now(), "",
JSON.stringify(encrypted.iv), isSecured,
"");
var xhr =requestJson("POST", serverName
+"/metaupload", meta);
                    xhr.onload=function
(ev) {
if(xhr.status===200) {
console.log("File uploaded
successfully");
var file
=JSON.parse(JSON.parse(xhr.response).data
);
fileInfo(file);
                        } else {
console.log("error ocured while uploading
file metadata")
                        }
                    };

uploadButton.innerHTML ='Upload';
                } else {
console.log(req.response);
console.log("error ocured");

uploadButton.innerHTML ='Upload';
                }
            }
var decrypted =awaitdecrypt(encrypted,
'password', mode, length);
var enc =newTextDecoder("utf-8");
console.log(enc.decode(decrypted)); //
Secret text
        })();
    };
    reader.readAsArrayBuffer(file);
}
functionfileInfo(file) {
document.getElementById("backButton").sty
le.display="block";
document.getElementById("fileListContaine
r").style.display="none";
document.getElementById("newFileFormConta
iner").style.display="none";
document.getElementById("fileContainer").
style.display="block";
document.getElementById("filename").inner
HTML = file.filename;
document.getElementById("fileDescription"
).innerHTML = file.description;
document.getElementById("createdAt").inne
rHTML = file.createdAt;
document.getElementById("identifier").inn
erHTML =
        location.hostname+":"+
location.port+"/safebox/public"+"/reposit
ory.html?file="+ file.urlIdentifier;
var keyField
=document.getElementById("secureKey");
if (file.isSecured ===false) {
        keyField.style.display="none";
    } else {
document.getElementById("key").innerHTML
= file.accessKey;
    }
}
//helpers
functionrequest(method, where, data) {
var xhr =newXMLHttpRequest();
    xhr.open(method, where, true);
    xhr.withCredentials =true;
```

```
    xhr.send();
return xhr;
}
functionsendFileData(method, where, data)
{
var xhr =newXMLHttpRequest();
    xhr.open(method, where, true);
    xhr.withCredentials =true;
    xhr.setRequestHeader("encoding",
"multipart/form-data");
    xhr.send(data);
return xhr;
}
functionrequestJson(method, where, data)
{
var xhr =newXMLHttpRequest();
var body =JSON.stringify(data);
console.log(body);
    xhr.open(method, where, true);
    xhr.withCredentials =true;
    xhr.setRequestHeader('Content-type',
'application/json; charset=utf-8');
    xhr.send(body);
return xhr;
}
//file meta dto
functionFileMetadata(id, userId,
filename, description, createdAt,
urlIdentifier, initVector, isSecured,
accessKey) {
this.id= id;
this.userId = userId;
this.filename= filename;
this.description= description;
this.createdAt = createdAt;
this.urlIdentifier = urlIdentifier;
this.initVector = initVector;
this.isSecured = isSecured;
this.accessKey= accessKey;
}
asyncfunctiongenEncryptionKey (password,
mode, length) {
var algo = {
        name:'PBKDF2',
        hash:'SHA-256',
        salt:newTextEncoder().encode('a-
unique-salt'),
        iterations:1000
    };
var derived = { name: mode, length:
length };
var encoded
=newTextEncoder().encode(password);
var key =await
crypto.subtle.importKey('raw', encoded, {
name:'PBKDF2' }, false, ['deriveKey']);
return crypto.subtle.deriveKey(algo, key,
derived, false, ['encrypt', 'decrypt']);
}
asyncfunctionencrypt (text, password,
mode, length, ivLength) {
var algo = {
        name: mode,
        length: length,
        iv:
crypto.getRandomValues(newUint8Array(ivLe
ngth))
    };
var key =awaitgenEncryptionKey(password,
mode, length);
var encoded = text;
return {
        cipherText:await
crypto.subtle.encrypt(algo, key,
encoded),
        iv: algo.iv
    };
}
asyncfunctiondecrypt (encrypted,
password, mode, length) {
var algo = {
        name: mode,
        length: length,
        iv: encrypted.iv
    };
var key =awaitgenEncryptionKey(password,
mode, length);
var decrypted =await
crypto.subtle.decrypt(algo, key,
encrypted.cipherText);
return decrypted;
}
```

Repository.js

```
 var queryStartsAt =query(url);
```

```
//server config
var host ="localhost";
var port ="9000";
var serverName ="http://"+ host +":"+
port;
if (queryStartsAt ===0) {
    location.href="notfound.html";
}
var queryStr = url.substr(queryStartsAt,
url.length- queryStartsAt);
var fileIdStartsAt =fileId(queryStr);
if (fileIdStartsAt ===0) {
    location.href="notfound.html";
}
var identifier =
queryStr.substr(fileIdStartsAt +1,
queryStr.length- fileIdStartsAt);
if (identifier.length===0) {
    location.href="notfound.html";
}
functionquery(href) {
returnfind(href, '?');
}
functionfileId(query) {
returnfind(query, '=');
}
//helper
functionfind(str, symbol) {
var startsAt =0;
var temp;
for(i =0; i < str.length; i++) {
if(str[i] === symbol) {
            startsAt = i;
return startsAt;
        }
    }
return startsAt;
}
functionskipSpace(fileId) {
var i =0;
while(fileId[i] ===' ') {
        i++;
    }
return fileId.substr(i, fileId.length);
}
functionrequest(method, where) {
var xhr =newXMLHttpRequest();
    xhr.open(method, where, true);
    xhr.withCredentials =true;
    xhr.send();
return xhr;
}
functionrequestJson(method, where, data)
{
var xhr =newXMLHttpRequest();
var body =JSON.stringify(data);
console.log(body);
    xhr.open(method, where, true);
    xhr.withCredentials =true;
    xhr.responseType ='blob';
    xhr.setRequestHeader('Content-type',
'application/json; charset=utf-8');
    xhr.send(body);
return xhr;
}
// ---------------------- Extensions to
MIME --------- //
// List of mime types
// combination of values from Windows 7
Registry and
// from
C:\Windows\System32\inetsrv\config\appli
cationHost.config
// some added, including .7z and .dat
var extToMIME = [
    [".323", "text/h323"],
    [".3g2", "video/3gpp2"],
    [".3gp", "video/3gpp"],
    [".3gp2", "video/3gpp2"],
    [".3gpp", "video/3gpp"],
    [".7z", "application/x-7z-
compressed"],
    [".aa", "audio/audible"],
    [".AAC", "audio/aac"],
    [".aaf", "application/octet-
stream"],
    [".aax", "audio/vnd.audible.aax"],
    [".ac3", "audio/ac3"],
    [".aca", "application/octet-
stream"],
    [".accda",
"application/msaccess.addin"],
    [".accdb", "application/msaccess"],
    [".accdc",
"application/msaccess.cab"],
    [".accde", "application/msaccess"],
```

```
    [".accdr",
"application/msaccess.runtime"],
    [".accdt", "application/msaccess"],
    [".accdw",
"application/msaccess.webapplication"],
    [".accft",
"application/msaccess.ftemplate"],
    [".acx", "application/internet-
property-stream"],
    [".AddIn", "text/xml"],
    [".ade", "application/msaccess"],
    [".adobebridge", "application/x-
bridge-url"],
    [".adp", "application/msaccess"],
    [".ADT", "audio/vnd.dlna.adts"],
    [".ADTS", "audio/aac"],
    [".afm", "application/octet-
stream"],
    [".ai", "application/postscript"],
    [".aif", "audio/x-aiff"],
    [".aifc", "audio/aiff"],
    [".aiff", "audio/aiff"],
    [".air", "application/vnd.adobe.air-
application-installer-package+zip"],
    [".amc", "application/x-mpeg"],
    [".application", "application/x-ms-
application"],
    [".art", "image/x-jg"],
    [".asa", "application/xml"],
    [".asax", "application/xml"],
    [".ascx", "application/xml"],
    [".asd", "application/octet-
stream"],
    [".asf", "video/x-ms-asf"],
    [".ashx", "application/xml"],
    [".asi", "application/octet-
stream"],
    [".asm", "text/plain"],
    [".asmx", "application/xml"],
    [".aspx", "application/xml"],
    [".asr", "video/x-ms-asf"],
    [".asx", "video/x-ms-asf"],
    [".atom", "application/atom+xml"],
    [".au", "audio/basic"],
    [".avi", "video/x-msvideo"],
    [".axs", "application/olescript"],
    [".bas", "text/plain"],
    [".bcpio", "application/x-bcpio"],
    [".bin", "application/octet-
stream"],
    [".bmp", "image/bmp"],
    [".c", "text/plain"],
    [".cab", "application/octet-
stream"],
    [".caf", "audio/x-caf"],
    [".calx", "application/vnd.ms-
office.calx"],
    [".cat", "application/vnd.ms-
pki.seccat"],
    [".cc", "text/plain"],
    [".cd", "text/plain"],
    [".cdda", "audio/aiff"],
    [".cdf", "application/x-cdf"],
    [".cer", "application/x-x509-ca-
cert"],
    [".chm", "application/octet-
stream"],
    [".class", "application/x-java-
applet"],
    [".clp", "application/x-msclip"],
    [".cmx", "image/x-cmx"],
    [".cnf", "text/plain"],
    [".cod", "image/cis-cod"],
    [".config", "application/xml"],
    [".contact", "text/x-ms-contact"],
    [".coverage", "application/xml"],
    [".cpio", "application/x-cpio"],
    [".cpp", "text/plain"],
    [".crd", "application/x-
mscardfile"],
    [".crl", "application/pkix-crl"],
    [".crt", "application/x-x509-ca-
cert"],
    [".cs", "text/plain"],
    [".csdproj", "text/plain"],
    [".csh", "application/x-csh"],
    [".csproj", "text/plain"],
    [".css", "text/css"],
    [".csv", "text/csv"],
    [".cur", "application/octet-
stream"],
    [".cxx", "text/plain"],
    [".dat", "application/octet-
stream"],
    [".datasource", "application/xml"],
    [".dbproj", "text/plain"],
```

```
    [".dcr", "application/x-director"],
    [".def", "text/plain"],
    [".deploy", "application/octet-
stream"],
    [".der", "application/x-x509-ca-
cert"],
    [".dgml", "application/xml"],
    [".dib", "image/bmp"],
    [".dif", "video/x-dv"],
    [".dir", "application/x-director"],
    [".disco", "text/xml"],
    [".dll", "application/x-
msdownload"],
    [".dll.config", "text/xml"],
    [".dlm", "text/dlm"],
    [".doc", "application/msword"],
    [".docm", "application/vnd.ms-
word.document.macroEnabled.12"],
    [".docx",
"application/vnd.openxmlformats-
officedocument.wordprocessingml.document
"],
    [".dot", "application/msword"],
    [".dotm", "application/vnd.ms-
word.template.macroEnabled.12"],
    [".dotx",
"application/vnd.openxmlformats-
officedocument.wordprocessingml.template
"],
    [".dsp", "application/octet-
stream"],
    [".dsw", "text/plain"],
    [".dtd", "text/xml"],
    [".dtsConfig", "text/xml"],
    [".dv", "video/x-dv"],
    [".dvi", "application/x-dvi"],
    [".dwf", "drawing/x-dwf"],
    [".dwp", "application/octet-
stream"],
    [".dxr", "application/x-director"],
    [".eml", "message/rfc822"],
    [".emz", "application/octet-
stream"],
    [".eot", "application/octet-
stream"],
    [".eps", "application/postscript"],
    [".etl", "application/etl"],
    [".etx", "text/x-setext"],
    [".evy", "application/envoy"],
    [".exe", "application/octet-
stream"],
    [".exe.config", "text/xml"],
    [".fdf", "application/vnd.fdf"],
    [".fif", "application/fractals"],
    [".filters", "Application/xml"],
    [".fla", "application/octet-
stream"],
    [".flr", "x-world/x-vrml"],
    [".flv", "video/x-flv"],
    [".fsscript", "application/fsharp-
script"],
    [".fsx", "application/fsharp-
script"],
    [".generictest", "application/xml"],
    [".gif", "image/gif"],
    [".group", "text/x-ms-group"],
    [".gsm", "audio/x-gsm"],
    [".gtar", "application/x-gtar"],
    [".gz", "application/x-gzip"],
    [".h", "text/plain"],
    [".hdf", "application/x-hdf"],
    [".hdml", "text/x-hdml"],
    [".hhc", "application/x-oleobject"],
    [".hhk", "application/octet-
stream"],
    [".hhp", "application/octet-
stream"],
    [".hlp", "application/winhlp"],
    [".hpp", "text/plain"],
    [".hqx", "application/mac-
binhex40"],
    [".hta", "application/hta"],
    [".htc", "text/x-component"],
    [".htm", "text/html"],
    [".html", "text/html"],
    [".htt", "text/webviewhtml"],
    [".hxa", "application/xml"],
    [".hxc", "application/xml"],
    [".hxd", "application/octet-
stream"],
    [".hxe", "application/xml"],
    [".hxf", "application/xml"],
    [".hxh", "application/octet-
stream"],
    [".hxi", "application/octet-
stream"],
```

```
    [".hxk", "application/xml"],
    [".hxq", "application/octet-
stream"],
    [".hxr", "application/octet-
stream"],
    [".hxs", "application/octet-
stream"],
    [".hxt", "text/html"],
    [".hxv", "application/xml"],
    [".hxw", "application/octet-
stream"],
    [".hxx", "text/plain"],
    [".i", "text/plain"],
    [".ico", "image/x-icon"],
    [".ics", "application/octet-
stream"],
    [".idl", "text/plain"],
    [".ief", "image/ief"],
    [".iii", "application/x-iphone"],
    [".inc", "text/plain"],
    [".inf", "application/octet-
stream"],
    [".inl", "text/plain"],
    [".ins", "application/x-internet-
signup"],
    [".ipa", "application/x-itunes-
ipa"],
    [".ipg", "application/x-itunes-
ipg"],
    [".ipproj", "text/plain"],
    [".ipsw", "application/x-itunes-
ipsw"],
    [".iqy", "text/x-ms-iqy"],
    [".isp", "application/x-internet-
signup"],
    [".ite", "application/x-itunes-
ite"],
    [".itlp", "application/x-itunes-
itlp"],
    [".itms", "application/x-itunes-
itms"],
    [".itpc", "application/x-itunes-
itpc"],
    [".IVF", "video/x-ivf"],
    [".jar", "application/java-
archive"],
    [".java", "application/octet-
stream"],
    [".jck",
"application/liquidmotion"],
    [".jcz",
"application/liquidmotion"],
    [".jfif", "image/pjpeg"],
    [".jnlp", "application/x-java-jnlp-
file"],
    [".jpb", "application/octet-
stream"],
    [".jpe", "image/jpeg"],
    [".jpeg", "image/jpeg"],
    [".jpg", "image/jpeg"],
    [".js", "application/x-javascript"],
    [".json", "application/json"],
    [".jsx", "text/jscript"],
    [".jsxbin", "text/plain"],
    [".latex", "application/x-latex"],
    [".library-ms",
"application/windows-library+xml"],
    [".lit", "application/x-ms-reader"],
    [".loadtest", "application/xml"],
    [".lpk", "application/octet-
stream"],
    [".lsf", "video/x-la-asf"],
    [".lst", "text/plain"],
    [".lsx", "video/x-la-asf"],
    [".lzh", "application/octet-
stream"],
    [".m13", "application/x-
msmediaview"],
    [".m14", "application/x-
msmediaview"],
    [".m1v", "video/mpeg"],
    [".m2t", "video/vnd.dlna.mpeg-tts"],
    [".m2ts", "video/vnd.dlna.mpeg-
tts"],
    [".m2v", "video/mpeg"],
    [".m3u", "audio/x-mpegurl"],
    [".m3u8", "audio/x-mpegurl"],
    [".m4a", "audio/m4a"],
    [".m4b", "audio/m4b"],
    [".m4p", "audio/m4p"],
    [".m4r", "audio/x-m4r"],
    [".m4v", "video/x-m4v"],
    [".mac", "image/x-macpaint"],
    [".mak", "text/plain"],
    [".man", "application/x-troff-man"],
    [".manifest", "application/x-ms-
```

```
manifest"],
    [".map", "text/plain"],
    [".master", "application/xml"],
    [".md", "text/plain"],
    [".mda", "application/msaccess"],
    [".mdb", "application/x-msaccess"],
    [".mde", "application/msaccess"],
    [".mdp", "application/octet-
stream"],
    [".me", "application/x-troff-me"],
    [".mfp", "application/x-shockwave-
flash"],
    [".mht", "message/rfc822"],
    [".mhtml", "message/rfc822"],
    [".mid", "audio/mid"],
    [".midi", "audio/mid"],
    [".mix", "application/octet-
stream"],
    [".mk", "text/plain"],
    [".mmf", "application/x-smaf"],
    [".mno", "text/xml"],
    [".mny", "application/x-msmoney"],
    [".mod", "video/mpeg"],
    [".mov", "video/quicktime"],
    [".movie", "video/x-sgi-movie"],
    [".mp2", "video/mpeg"],
    [".mp2v", "video/mpeg"],
    [".mp3", "audio/mpeg"],
    [".mp4", "video/mp4"],
    [".mp4v", "video/mp4"],
    [".mpa", "video/mpeg"],
    [".mpe", "video/mpeg"],
    [".mpeg", "video/mpeg"],
    [".mpf", "application/vnd.ms-
mediapackage"],
    [".mpg", "video/mpeg"],
    [".mpp", "application/vnd.ms-
project"],
    [".mpv2", "video/mpeg"],
    [".mqv", "video/quicktime"],
    [".ms", "application/x-troff-ms"],
    [".msi", "application/octet-
stream"],
    [".mso", "application/octet-
stream"],
    [".mts", "video/vnd.dlna.mpeg-tts"],
    [".mtx", "application/xml"],
    [".mvb", "application/x-
msmediaview"],
    [".mvc", "application/x-miva-
compiled"],
    [".mxp", "application/x-mmxp"],
    [".nc", "application/x-netcdf"],
    [".nsc", "video/x-ms-asf"],
    [".nws", "message/rfc822"],
    [".ocx", "application/octet-
stream"],
    [".oda", "application/oda"],
    [".odc", "text/x-ms-odc"],
    [".odh", "text/plain"],
    [".odl", "text/plain"],
    [".odp",
"application/vnd.oasis.opendocument.pres
entation"],
    [".ods", "application/oleobject"],
    [".odt",
"application/vnd.oasis.opendocument.text
"],
    [".one", "application/onenote"],
    [".onea", "application/onenote"],
    [".onepkg", "application/onenote"],
    [".onetmp", "application/onenote"],
    [".onetoc", "application/onenote"],
    [".onetoc2", "application/onenote"],
    [".orderedtest", "application/xml"],
    [".osdx",
"application/opensearchdescription+xml"]
,
    [".p10", "application/pkcs10"],
    [".p12", "application/x-pkcs12"],
    [".p7b", "application/x-pkcs7-
certificates"],
    [".p7c", "application/pkcs7-mime"],
    [".p7m", "application/pkcs7-mime"],
    [".p7r", "application/x-pkcs7-
certreqresp"],
    [".p7s", "application/pkcs7-
signature"],
    [".pbm", "image/x-portable-bitmap"],
    [".pcast", "application/x-podcast"],
    [".pct", "image/pict"],
    [".pcx", "application/octet-
stream"],
    [".pcz", "application/octet-
stream"],
    [".pdf", "application/pdf"],
```

```
    [".pfb", "application/octet-
stream"],
    [".pfm", "application/octet-
stream"],
    [".pfx", "application/x-pkcs12"],
    [".pgm", "image/x-portable-
graymap"],
    [".pic", "image/pict"],
    [".pict", "image/pict"],
    [".pkgdef", "text/plain"],
    [".pkgundef", "text/plain"],
    [".pko", "application/vnd.ms-
pki.pko"],
    [".pls", "audio/scpls"],
    [".pma", "application/x-perfmon"],
    [".pmc", "application/x-perfmon"],
    [".pml", "application/x-perfmon"],
    [".pmr", "application/x-perfmon"],
    [".pmw", "application/x-perfmon"],
    [".png", "image/png"],
    [".pnm", "image/x-portable-anymap"],
    [".pnt", "image/x-macpaint"],
    [".pntg", "image/x-macpaint"],
    [".pnz", "image/png"],
    [".pot", "application/vnd.ms-
powerpoint"],
    [".potm", "application/vnd.ms-
powerpoint.template.macroEnabled.12"],
    [".potx",
"application/vnd.openxmlformats-
officedocument.presentationml.template"]
,
    [".ppa", "application/vnd.ms-
powerpoint"],
    [".ppam", "application/vnd.ms-
powerpoint.addin.macroEnabled.12"],
    [".ppm", "image/x-portable-pixmap"],
    [".pps", "application/vnd.ms-
powerpoint"],
    [".ppsm", "application/vnd.ms-
powerpoint.slideshow.macroEnabled.12"],
    [".ppsx",
"application/vnd.openxmlformats-
officedocument.presentationml.slideshow"
],
    [".ppt", "application/vnd.ms-
powerpoint"],
    [".pptm", "application/vnd.ms-
powerpoint.presentation.macroEnabled.12"
],
    [".pptx",
"application/vnd.openxmlformats-
officedocument.presentationml.presentati
on"],
    [".prf", "application/pics-rules"],
    [".prm", "application/octet-
stream"],
    [".prx", "application/octet-
stream"],
    [".ps", "application/postscript"],
    [".psc1", "application/PowerShell"],
    [".psd", "application/octet-
stream"],
    [".psess", "application/xml"],
    [".psm", "application/octet-
stream"],
    [".psp", "application/octet-
stream"],
    [".pub", "application/x-
mspublisher"],
    [".pwz", "application/vnd.ms-
powerpoint"],
    [".qht", "text/x-html-insertion"],
    [".qhtm", "text/x-html-insertion"],
    [".qt", "video/quicktime"],
    [".qti", "image/x-quicktime"],
    [".qtif", "image/x-quicktime"],
    [".qtl", "application/x-
quicktimeplayer"],
    [".qxd", "application/octet-
stream"],
    [".ra", "audio/x-pn-realaudio"],
    [".ram", "audio/x-pn-realaudio"],
    [".rar", "application/octet-
stream"],
    [".ras", "image/x-cmu-raster"],
    [".rat", "application/rat-file"],
    [".rc", "text/plain"],
    [".rc2", "text/plain"],
    [".rct", "text/plain"],
    [".rdlc", "application/xml"],
    [".resx", "application/xml"],
    [".rf", "image/vnd.rn-realflash"],
    [".rgb", "image/x-rgb"],
    [".rgs", "text/plain"],
    [".rm", "application/vnd.rn-
```

```
realmedia"],
    [".rmi", "audio/mid"],
    [".rmp", "application/vnd.rn-
rn_music_package"],
    [".roff", "application/x-troff"],
    [".rpm", "audio/x-pn-realaudio-
plugin"],
    [".rqy", "text/x-ms-rqy"],
    [".rtf", "application/rtf"],
    [".rtx", "text/richtext"],
    [".ruleset", "application/xml"],
    [".s", "text/plain"],
    [".safariextz", "application/x-
safari-safariextz"],
    [".scd", "application/x-
msschedule"],
    [".sct", "text/scriptlet"],
    [".sd2", "audio/x-sd2"],
    [".sdp", "application/sdp"],
    [".sea", "application/octet-
stream"],
    [".searchConnector-ms",
"application/windows-search-
connector+xml"],
    [".setpay", "application/set-
payment-initiation"],
    [".setreg", "application/set-
registration-initiation"],
    [".settings", "application/xml"],
    [".sgimb", "application/x-sgimb"],
    [".sgml", "text/sgml"],
    [".sh", "application/x-sh"],
    [".shar", "application/x-shar"],
    [".shtml", "text/html"],
    [".sit", "application/x-stuffit"],
    [".sitemap", "application/xml"],
    [".skin", "application/xml"],
    [".sldm", "application/vnd.ms-
powerpoint.slide.macroEnabled.12"],
    [".sldx",
"application/vnd.openxmlformats-
officedocument.presentationml.slide"],
    [".slk", "application/vnd.ms-
excel"],
    [".sln", "text/plain"],
    [".slupkg-ms", "application/x-ms-
license"],
    [".smd", "audio/x-smd"],
    [".smi", "application/octet-
stream"],
    [".smx", "audio/x-smd"],
    [".smz", "audio/x-smd"],
    [".snd", "audio/basic"],
    [".snippet", "application/xml"],
    [".snp", "application/octet-
stream"],
    [".sol", "text/plain"],
    [".sor", "text/plain"],
    [".spc", "application/x-pkcs7-
certificates"],
    [".spl",
"application/futuresplash"],
    [".src", "application/x-wais-
source"],
    [".srf", "text/plain"],
    [".SSISDeploymentManifest",
"text/xml"],
    [".ssm",
"application/streamingmedia"],
    [".sst", "application/vnd.ms-
pki.certstore"],
    [".stl", "application/vnd.ms-
pki.stl"],
    [".sv4cpio", "application/x-
sv4cpio"],
    [".sv4crc", "application/x-sv4crc"],
    [".svc", "application/xml"],
    [".swf", "application/x-shockwave-
flash"],
    [".t", "application/x-troff"],
    [".tar", "application/x-tar"],
    [".tcl", "application/x-tcl"],
    [".testrunconfig",
"application/xml"],
    [".testsettings",
"application/xml"],
    [".tex", "application/x-tex"],
    [".texi", "application/x-texinfo"],
    [".texinfo", "application/x-
texinfo"],
    [".tgz", "application/x-
compressed"],
    [".thmx", "application/vnd.ms-
officetheme"],
    [".thn", "application/octet-
stream"],
```

```
    [".tif", "image/tiff"],
    [".tiff", "image/tiff"],
    [".tlh", "text/plain"],
    [".tli", "text/plain"],
    [".toc", "application/octet-
stream"],
    [".tr", "application/x-troff"],
    [".trm", "application/x-
msterminal"],
    [".trx", "application/xml"],
    [".ts", "video/vnd.dlna.mpeg-tts"],
    [".tsv", "text/tab-separated-
values"],
    [".ttf", "application/octet-
stream"],
    [".tts", "video/vnd.dlna.mpeg-tts"],
    [".txt", "text/plain"],
    [".u32", "application/octet-
stream"],
    [".uls", "text/iuls"],
    [".user", "text/plain"],
    [".ustar", "application/x-ustar"],
    [".vb", "text/plain"],
    [".vbdproj", "text/plain"],
    [".vbk", "video/mpeg"],
    [".vbproj", "text/plain"],
    [".vbs", "text/vbscript"],
    [".vcf", "text/x-vcard"],
    [".vcproj", "Application/xml"],
    [".vcs", "text/plain"],
    [".vcxproj", "Application/xml"],
    [".vddproj", "text/plain"],
    [".vdp", "text/plain"],
    [".vdproj", "text/plain"],
    [".vdx", "application/vnd.ms-
visio.viewer"],
    [".vml", "text/xml"],
    [".vscontent", "application/xml"],
    [".vsct", "text/xml"],
    [".vsd", "application/vnd.visio"],
    [".vsi", "application/ms-vsi"],
    [".vsix", "application/vsix"],
    [".vsixlangpack", "text/xml"],
    [".vsixmanifest", "text/xml"],
    [".vsmdi", "application/xml"],
    [".vspscc", "text/plain"],
    [".vss", "application/vnd.visio"],
    [".vsscc", "text/plain"],
    [".vssettings", "text/xml"],
    [".vssscc", "text/plain"],
    [".vst", "application/vnd.visio"],
    [".vstemplate", "text/xml"],
    [".vsto", "application/x-ms-vsto"],
    [".vsw", "application/vnd.visio"],
    [".vsx", "application/vnd.visio"],
    [".vtx", "application/vnd.visio"],
    [".wav", "audio/wav"],
    [".wave", "audio/wav"],
    [".wax", "audio/x-ms-wax"],
    [".wbk", "application/msword"],
    [".wbmp", "image/vnd.wap.wbmp"],
    [".wcm", "application/vnd.ms-
works"],
    [".wdb", "application/vnd.ms-
works"],
    [".wdp", "image/vnd.ms-photo"],
    [".webarchive", "application/x-
safari-webarchive"],
    [".webtest", "application/xml"],
    [".wiq", "application/xml"],
    [".wiz", "application/msword"],
    [".wks", "application/vnd.ms-
works"],
    [".WLMP",
"application/wlmoviemaker"],
    [".wlpginstall", "application/x-
wlpg-detect"],
    [".wlpginstall3", "application/x-
wlpg3-detect"],
    [".wm", "video/x-ms-wm"],
    [".wma", "audio/x-ms-wma"],
    [".wmd", "application/x-ms-wmd"],
    [".wmf", "application/x-
msmetafile"],
    [".wml", "text/vnd.wap.wml"],
    [".wmlc",
"application/vnd.wap.wmlc"],
    [".wmls", "text/vnd.wap.wmlscript"],
    [".wmlsc",
"application/vnd.wap.wmlscriptc"],
    [".wmp", "video/x-ms-wmp"],
    [".wmv", "video/x-ms-wmv"],
    [".wmx", "video/x-ms-wmx"],
    [".wmz", "application/x-ms-wmz"],
    [".wpl", "application/vnd.ms-wpl"],
    [".wps", "application/vnd.ms-
```

```
works"],
    [".wri", "application/x-mswrite"],
    [".wrl", "x-world/x-vrml"],
    [".wrz", "x-world/x-vrml"],
    [".wsc", "text/scriptlet"],
    [".wsdl", "text/xml"],
    [".wvx", "video/x-ms-wvx"],
    [".x", "application/directx"],
    [".xaf", "x-world/x-vrml"],
    [".xaml", "application/xaml+xml"],
    [".xap", "application/x-silverlight-
app"],
    [".xbap", "application/x-ms-xbap"],
    [".xbm", "image/x-xbitmap"],
    [".xdr", "text/plain"],
    [".xht", "application/xhtml+xml"],
    [".xhtml", "application/xhtml+xml"],
    [".xla", "application/vnd.ms-
excel"],
    [".xlam", "application/vnd.ms-
excel.addin.macroEnabled.12"],
    [".xlc", "application/vnd.ms-
excel"],
    [".xld", "application/vnd.ms-
excel"],
    [".xlk", "application/vnd.ms-
excel"],
    [".xll", "application/vnd.ms-
excel"],
    [".xlm", "application/vnd.ms-
excel"],
    [".xls", "application/vnd.ms-
excel"],
    [".xlsb", "application/vnd.ms-
excel.sheet.binary.macroEnabled.12"],
    [".xlsm", "application/vnd.ms-
excel.sheet.macroEnabled.12"],
    [".xlsx",
"application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet"],
    [".xlt", "application/vnd.ms-
excel"],
    [".xltm", "application/vnd.ms-
excel.template.macroEnabled.12"],
    [".xltx",
"application/vnd.openxmlformats-
officedocument.spreadsheetml.template"],
    [".xlw", "application/vnd.ms-
excel"],
    [".xml", "text/xml"],
    [".xmta", "application/xml"],
    [".xof", "x-world/x-vrml"],
    [".XOML", "text/plain"],
    [".xpm", "image/x-xpixmap"],
    [".xps", "application/vnd.ms-
xpsdocument"],
    [".xrm-ms", "text/xml"],
    [".xsc", "application/xml"],
    [".xsd", "text/xml"],
    [".xsf", "text/xml"],
    [".xsl", "text/xml"],
    [".xslt", "text/xml"],
    [".xsn", "application/octet-
stream"],
    [".xss", "application/xml"],
    [".xtp", "application/octet-
stream"],
    [".xwd", "image/x-xwindowdump"],
    [".z", "application/x-compress"],
    [".zip", "application/x-zip-
compressed"]
];
var req =request("GET", serverName
+"/metadownload?file="+ identifier);
req.onload=function () {
if (req.status===200) {
console.log(req.response);
var respData =JSON.parse(req.response);
var file =JSON.parse(respData.data);
document.getElementById("filename").inne
rHTML = file.filename;
document.getElementById("fileDescription
").innerHTML = file.filename;
document.getElementById("createdAt").inn
erHTML = file.createdAt;
document.getElementById("repositoryFile"
).style.display="block";
if(file.isSecured ===false) {
document.getElementById("secureKey").sty
le.display="none";
        }
document.getElementById("downloadButton"
).addEventListener("click", function() {
var key
=document.getElementById("key").value;
            key =skipSpace(key);
```

```
if (file.isSecurede && key.length===0) {
document.getElementById("keyIsEmptyAlert
").style.display="block";
return;
            } else {
                file.accessKey= key;
var downloadReq =requestJson("POST",
serverName +"/file", file);

downloadReq.onload=function() {
// Only handle status code 200
if(downloadReq.status===200) {
// Try to find out the filename from the
content disposition `filename` value
var disposition =
downloadReq.getResponseHeader('content-
disposition');
var matches =
/"([^"]*)"/.exec(disposition);
var filename = (matches !=null&&
matches[1] ? matches[1] :
file.filename);

filetype="text/*";

extension=filename.substring(filename.la
stIndexOf("."));
for (var i =0; i < extToMIME.length;
i++) {
if
(extToMIME[i][0].localeCompare(extension
)===0) {

filetype=extToMIME[i][1];
break;
                            }
                        }
var fileReader =newFileReader();

fileReader.onload=function() {
var encrypted =this.result;
var enc =newTextDecoder("utf-8");
// The actual download
                            (async () =>
{
var mode ='AES-GCM',

length =256;

var ivObj =JSON.parse(file.initVector);
var iv
=newUint8Array(Object.values(ivObj));
var decrypted =awaitdecrypt(encrypted,
'password', iv, mode, length);
var blob =newBlob([decrypted], {type:
filetype});
var link =document.createElement('a');

link.href=window.URL.createObjectURL(blo
b);

link.download = filename;
document.body.appendChild(link);

link.click();
document.body.removeChild(link);
                            })();
                        };

fileReader.readAsArrayBuffer(downloadReq
.response);
                    }
// some error handling should be done
here...
                };
            }
        });
    } else {
console.log("error fetching metadata");
        location.href="notfound.html";
    }
};
asyncfunctiondecrypt (encrypted,
password, iv, mode, length) {
var algo = {
        name: mode,
        length: length,
        iv: iv
    };
var key =awaitgenEncryptionKey(password,
mode, length);
var decrypted =await
crypto.subtle.decrypt(algo, key,
encrypted);
return decrypted;
}
asyncfunctiongenEncryptionKey (password,
```

```
mode, length) {
var algo = {
        name:'PBKDF2',
        hash:'SHA-256',
        salt:newTextEncoder().encode('a-
unique-salt'),
        iterations:1000
    };
var derived = { name: mode, length:
length };
var encoded
=newTextEncoder().encode(password);
var key =await
crypto.subtle.importKey('raw', encoded,
{ name:'PBKDF2' }, false,
['deriveKey']);
return crypto.subtle.deriveKey(algo,
key, derived, false, ['encrypt',
'decrypt']);
}
```

Userforms.js

```
//se
rver
conf
ig
        var host ="localhost";
        var port ="9000";
        var serverName ="http://"+ host +":"+
        port;
        //hiding login/register forms
        document.getElementById("register").styl
        e.display='none';
        document.getElementById("login").style.d
        isplay='block';
        location.href=
        location.pathname+"#login";
        document.getElementById("loginLink").add
        EventListener("click",
        hideUserRegisterForm);
        document.getElementById("registerLink").
        addEventListener("click",
        hideUserLoginForm);
        functionhideUserRegisterForm() {
            location.href=
        location.pathname+"#login";
        document.getElementById("register").styl
        e.display='none';
        document.getElementById("login").style.d
        isplay='block';
        }
        functionhideUserLoginForm() {
            location.href=
        location.pathname+"#register";
        document.getElementById("login").style.d
        isplay='none';
        document.getElementById("register").styl
        e.display='block';
        }
        //login actions
        document.getElementById("loginButton").a
        ddEventListener("click", login);
        functionlogin() {
        var user =newUser(0, "", "", "");
            user.email
        =document.getElementById("loginEmail").v
        alue;
            user.password
        =document.getElementById("loginPassword"
        ).value;
        var req =request("POST", serverName
        +"/login", user);
            req.onload=function () {
        if (req.status===200) {
        var body =JSON.parse(req.response);
        var profile = body.data;

        localStorage.setItem("profile",
        profile);

        location.href="profile.html";
                } elseif (req.status>=400) {
        var respBody =JSON.parse(req.response);
        var data =JSON.parse(respBody.data);
        errorAlert("loginError", data.reason)
                }
            }
            req.onerror=function () {
        errorAlert("registerError", "Something
        went wrong. Check your internet
        connection and retry later.")
            }
        }
        //register action
        document.getElementById("registerButton"
```

```
).addEventListener("click", register);
functionregister() {
var password
=document.getElementById("registerPasswo
rd").value;
var confirmedPassword
=document.getElementById("registerConfir
mPassword").value;
var email
=document.getElementById("registerEmail"
).value;
var name
=document.getElementById("registerName")
.value;
if (isRegisterDataValid(name, email,
password, confirmedPassword) ==false) {
errorAlert("registerError", "Invalid
input data. Please check that all fields
are filled and passwords are matched.");
    } else {
var user =newUser(0, "", "", "");
        user.email = email;
        user.name= name;
        user.password = password;
var req =request("POST", serverName
+"/register", user)
        req.onload=function () {
if (req.status===200) {
successAlert("registerSuccess", "You are
successfully registered. Now you can log
in.")
            } elseif (req.status>=400) {
var respBody =JSON.parse(req.response);
var data =JSON.parse(respBody.data);
errorAlert("registerError", data.reason)
            }
        }
        req.onerror=function () {
errorAlert("registerError", "Something
went wrong. Check your internet
connection and retry later.")
        }
    }
}
//forms/dto
functionUser(id, name, email, password)
{
this.id= id;
this.name= name;
this.email = email;
this.password = password;
}
//helpers
functionrequest(method, where, data) {
var xhr =newXMLHttpRequest();
var body =JSON.stringify(data);
    xhr.open(method, where, true);
    xhr.withCredentials =true;
    xhr.setRequestHeader('Content-type',
'application/json; charset=utf-8');
    xhr.send(body);
return xhr;
}
functionisRegisterDataValid(name, email,
password, confirm) {
if (name ===""|| email ===""|| password
===""|| confirm ===="") {
returnfalse;
    } elseif (password !== confirm) {
returnfalse;
    }
returntrue;
}
functionsuccessAlert(alertName,
alertText) {
document.getElementById("registerErrorAl
ert").style.display="none";
document.getElementById(alertName
+"Alert").style.display="block";
document.getElementById(alertName).inner
HTML = alertText;
}
functionerrorAlert(alertName, alertText)
{
document.getElementById(alertName
+"Alert").style.display="block";
document.getElementById(alertName).inner
HTML = alertText;
}
```

Index.html

```
<!D
OCT
YPE
htm
```

```
l>
        <htmllang="en">
        <head>
        <metacharset="UTF-8">
        <title>Safebox</title>
        <!-- Bootstrap CSS -->
        <linkrel="stylesheet"href="https://stackp
        ath.bootstrapcdn.com/bootstrap/4.1.1/css/
        bootstrap.min.css"integrity="sha384-
        WskhaSGFgHYWDcbwN70/dfYBj47jz9qbsMId/iRN3
        ewGhXQFZCSftd1LZCfmhktB"crossorigin="anon
        ymous">
        <linkrel="stylesheet"href="css/main.css">
        </head>
        <body>
        <divclass="login-image">
        <imgsrc="images/storage5.png">
        </div>
        <divclass="user-form"id="login">
        <h2class="form-signin-heading center-
        text">Log In</h2>
        <inputid="loginEmail"type="text"class="fo
        rm-
        control"name="username"placeholder="Email
        Address"required=""
        autofocus=""/>
        <inputid="loginPassword"type="password"cl
        ass="form-
        control"name="password"placeholder="Passw
        ord"
        required=""/>
        <labelclass="checkbox">
        <inputtype="checkbox"value="remember-
        me"id="rememberMe"name="rememberMe">
        Remember me
        </label>
        <buttonid="loginButton"class="btn btn-lg
        btn-primary btn-block">Login</button>
        <div>
        <pid="registerLink">Register</p>
        </div><br>
        <divid="loginErrorAlert"class="alert
        alert-danger"style="display:
        none"role="alert">
        <pid="loginError"></p>
        </div>
        </div>
        <divclass="user-form"id="register">
        <h2class="form-signin-heading center-
        text">Register</h2>
        <inputid="registerEmail"type="text"class=
        "form-
        control"name="email"placeholder="Email
        Address"required=""autofocus=""/>
        <inputid="registerName"type="text"class="
        form-
        control"name="username"placeholder="Usern
        ame"required=""autofocus=""/>
        <inputid="registerPassword"type="password
        "class="form-
        control"name="password"placeholder="Passw
        ord"required=""/>
        <inputid="registerConfirmPassword"type="p
        assword"class="form-
        control"name="passwordRepeat"placeholder=
        "Repeat password"
        required=""/><br>
        <buttonid="registerButton"class="btn btn-
        lg btn-primary btn-
        block">Register</button>
        <div>
        <pid="loginLink">Log In</p>
        </div><br>
        <divid="registerSuccessAlert"class="alert
        alert-success"style="display: none">
        <pid="registerSuccess"></p>
        </div>
        <divid="registerErrorAlert"class="alert
        alert-danger"style="display:
        none"role="alert">
        <pid="registerError"></p>
        </div>
        </div>
        <!-- Optional JavaScript -->
        <!-- jQuery first, then Popper.js, then
        Bootstrap JS -->
        <scriptsrc="https://code.jquery.com/jquer
        y-3.3.1.slim.min.js"integrity="sha384-
        q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smX
        Kp4YfRvH+8abtTE1Pi6jizo"crossorigin="anon
        ymous"></script>
        <scriptsrc="https://cdnjs.cloudflare.com/
        ajax/libs/popper.js/1.14.3/umd/popper.min
        .js"integrity="sha384-
        ZMP7rVo3mIykV+2+9J3UJ46jBk0WLaUAdn689aCwo
        qbBJiSnjAK/l8WvCWPIPm49"crossorigin="anon
```

```
ymous"></script>
<scriptsrc="https://stackpath.bootstrapcd
n.com/bootstrap/4.1.1/js/bootstrap.min.js
"integrity="sha384-
smHYKdLADwkXOn1EmN1qk/HfnUcbVRZyYmZ4qpPea
6sjB/pTJ0euyQp0Mk8ck+5T"crossorigin="anon
ymous"></script>
<!--user -->
<scriptsrc="js/userform.js"></script>
</body>
</html>
```

Notfound.html

```
<!DOCTYPE html>
                <htmllang="en">
                <head>
                <metacharset="UTF-8">
                <title>404</title>
                </head>
                <body>
                <divclass="error"style="text-
                align: center; margin-top:
                10%;">
                <h2>Oops!</h2>
                <h1> 404 </h1>
                <h2>Not Found</h2>
                <div>
                            Sorry, an error
                has occured. Requested page
                not found!
                </div>
                </div>
                </body>
                </html>
```

Profile.hmtl

```
<!D
OCT
YPE
htm
l>
      <htmllang="en">
      <head>
      <metacharset="UTF-8">
      <title>Title</title>
      <linkrel="stylesheet"href="https://stackp
      ath.bootstrapcdn.com/bootstrap/4.1.1/css/
      bootstrap.min.css"integrity="sha384-
      WskhaSGFgHYWDcbwN70/dfYBj47jz9qbsMId/iRN3
      ewGhXQFZCSftd1LZCfmhktB"crossorigin="anon
      ymous">
      </head>
      <body>
      <divclass="container-
      fluid"style="background-color: #E5EAFF">
      <divclass="row align-items-center h-100">
      <divid="logo"class="col-md-1">
      <imgsrc="images/storage5.png"width=80%hei
      ght=70%>
      </div>
      <divclass="col-md-2">
      <!-- Logo -->
      <divclass="logo">
      <h1class="text-left"><astyle="text-
      decoration:none;"href="profile.html">Safe
      Box</a></h1>
      </div>
      </div>
      <divclass="col-md-4">
      <h3style="cursor:pointer;color:blue;"id="
      profileName"></h3>
      </div>
      <divclass="col-md">
      <divclass="row">
      <divclass="col-lg-12">
      <divclass="input-group form">
      <inputtype="text"class="form-
      control"placeholder="Search...">
      <spanclass="input-group-btn">
      <buttonclass="btn btn-
      primary"type="button">Search</button>
      </span>
      </div>
      </div>
      </div>
      </div>
      <divclass="col-md text-center">
      <divclass="logo">
      <h4><astyle="text-
      decoration:none;"id="logout"href="profile
      .html">logout</a></h4>
      </div>
      </div>
      </div>
      </div>
```

```
<br><br>
<divid="fileListContainer"class="containe
r">
<divid="filesContainer"class="container">
</div>
<br>
<divclass="container">
<buttonid="newFileButton"class="btn btn-
primary"type="button">New file</button>
</div>
</div>
<divid="backButton"class="container"style
="display: none;">
<divclass="row">
<divclass="col-md">
<buttontype="button"class="btn btn-
secondary">Back</button>
</div>
</div>
</div>
<br>
<br>
<br>
<br>
<divid="newFileFormContainer"class="conta
iner"style="display: none">
<divclass="row">
<divclass="col-12">
<divclass="row">
<divclass="col">
<inputtype="file"id="fileSelect"name="new
File"multiple/>
</div>
</div>
<br>
<divclass="row">
<divclass="col">
<inputclass="form-
control"type="text"id="description"placeh
older="File description">
</div>
</div>
<br>
<divclass="row">
<divclass="col">
<divclass="form-check">
<inputtype="checkbox"class="form-check-
input"id="isSecured">
<labelclass="form-check-
label"for="isSecured">Generate security
key</label>
</div>
</div>
</div>
<divclass="row">
<divclass="col-md text-right">
<buttonclass="btn btn-
primary"id="uploadButton">Upload</button>
</div>
</div>
</div>
</div>
</div>
<divclass="container"id="fileContainer"st
yle="display: none">
<divclass="row">
<h4><pstyle="cursor:pointer;color:blue;"i
d="filename">FILENAME</p></h4>
</div>
<divclass="row">
<pid="fileDescription">FILE
DESCRIPTION</p>
</div>
<divclass="row">
<pid="createdAt">CREATED AT</p>
</div>
<br>
<divclass="row">
            Identifier
</div>
<divclass="row">
<spanclass="input-group-
text"id="identifier"></span>
</div>
<divid="secureKey">
<divclass="row">
                Secure key
</div>
<divclass="row">
<spanid="key"class="input-group-
text"></span>
</div>
</div>
</div>
<scriptsrc="js/profile.js"></script>
<scriptsrc="https://code.jquery.com/jquer
```

```
y-3.3.1.slim.min.js"integrity="sha384-
q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smX
Kp4YfRvH+8abtTE1Pi6jizo"crossorigin="anon
ymous"></script>
<scriptsrc="https://cdnjs.cloudflare.com/
ajax/libs/popper.js/1.14.3/umd/popper.min
.js"integrity="sha384-
ZMP7rVo3mIykV+2+9J3UJ46jBk0WLaUAdn689aCwo
qbBJiSnjAK/l8WvCWPIPm49"crossorigin="anon
ymous"></script>
<scriptsrc="https://stackpath.bootstrapcd
n.com/bootstrap/4.1.1/js/bootstrap.min.js
"integrity="sha384-
smHYKdLADwkXOn1EmN1qk/HfnUcbVRZyYmZ4qpPea
6sjB/pTJ0euyQp0Mk8ck+5T"crossorigin="anon
ymous"></script>
</body>
</html>
```

Repository.html

```
<!D
OCT
YPE
htm
l>
<htmllang="en">
<head>
<metacharset="UTF-8">
<title>Title</title>
<linkrel="stylesheet"href="https://stackp
ath.bootstrapcdn.com/bootstrap/4.1.1/css/
bootstrap.min.css"integrity="sha384-
WskhaSGFgHYWDcbwN70/dfYBj47jz9qbsMId/iRN3
ewGhXQFZCSftd1LZCfmhktB"crossorigin="anon
ymous">
</head>
<body>
<divclass="container-
fluid"style="background-color: #E5EAFF">
<divclass="row align-items-center h-100">
<divid="logo"class="col-md-1">
<imgsrc="images/storage5.png"width=80%hei
ght=70%>
</div>
<divclass="col-md-2">
<!-- Logo -->
<divclass="logo">
<h1class="text-left"><astyle="text-
decoration:none;"href="profile.html">Safe
Box</a></h1>
</div>
</div>
</div>
</div>
<br>
<br>
<br>
<br>
<br>
<divclass="container"id="repositoryFile"s
tyle="display: none;">
<divclass="row">
<h4><pstyle="cursor:pointer;color:blue;"i
d="filename">FILENAME</p></h4>
</div>
<divclass="row">
<pid="fileDescription">FILE
DESCRIPTION</p>
</div>
<divclass="row">
<pid="createdAt">CREATED AT</p>
</div>
<br>
<divid="secureKey">
<divclass="row">
                Secure key
</div>
<divclass="row">
<inputtype="text"class="form-
control"id="key"placeholder="Enter access
key">
</div>
<br>
</div>
<divclass="row">
<buttonclass="btn btn-
primary"id="downloadButton">Download</but
ton>
</div>
<divid="keyIsEmptyAlert"class="alert
alert-danger"style="display:
none"role="alert">
<p>Enter access key to download.</p>
</div>
</div>
```

```
<scriptsrc="js/repository.js"></script>
<scriptsrc="https://code.jquery.com/jquery-3.3.1.slim.min.js"integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"crossorigin="anonymous"></script>
<scriptsrc="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/popper.min.js"integrity="sha384-ZMP7rVo3mIykV+2+9J3UJ46jBk0WLaUAdn689aCwoqbBJiSnjAK/l8WvCWPIPm49"crossorigin="anonymous"></script>
<scriptsrc="https://stackpath.bootstrapcdn.com/bootstrap/4.1.1/js/bootstrap.min.js"integrity="sha384-smHYKdLADwkXOn1EmN1qk/HfnUcbVRZyYmZ4qpPea6sjB/pTJ0euyQp0Mk8ck+5T"crossorigin="anonymous"></script>
</body>
</html>
```