ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКААЦІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЗАХИСТУ ІНФОРМАЦІЇ

КАФЕДРА ІНФОРМАЦІЙНОЇ ТА КІБЕРНЕТИЧНОЇ БЕЗПЕКИ

Пояснювальна записка

до магістерської роботи

на тему:

«ДОСЛІДЖЕННЯ ЕЛЕКТРОННИХ ЦИФРОВИХ ПІДПИСІВ ЗА КРИТЕРІЄМ «СТІЙКІСТЬ – СКЛАДНІСТЬ»»

Виконав: студент 6 курсу, групи БСДМ-62
Спеціальності 125 Кібербезпека
Освітньо-професійної програми
«Інформаційна та кібернетична безпека»
(шифр і назва спеціальності)
_____ Супрунов В.С._____
(прізвище та ініціали)
Керівник__Кожухівський А.Д.___
(прізвище та ініціали)
Рецензент_____
(прізвище та ініціали)
Нормоконтролер _Чумак Н.С___

КИЇВ – 2022

РЕФЕРАТ

Текстова частина магістерської роботи: 82 сторінок, 15 рисунків, 40 джерел.

Об'єкт дослідження – застосування електронно цифрового підпису в електронному документообігу.

Предмет дослідження – методи та механізми шифрування електронно цифрового підпису.

Мета роботи – розробка рекомендацій щодо ефективності використання електронно цифрового підпису за критерієм «стійкість – складність».

Методи дослідження – опрацювання літератури за даною темою, аналіз експлуатаційної документації, актуальних рішень та їх порівняння, проведення експерименту.

В роботі вирішенні поступово основні завдання захисту електронного документообігу.

Проведенно аналіз та дослідження різних типів електронних цифрових підписів та їх механізмів захисту електронного документообігу.

Досліджено методи і механізми захисту електронними цифровими підписами електронного документообігу .

Досліджено можливості програмного комплексу XML Digital Signature API.

Розробленно рекомендації щодо вдосконалення методів застосування електронних цифрових підписів.

Галузь використання – забезпечення безпеки при використанні електронно цифрового підпису в електронному документообігу.

ЦИФРОВИЙ ПІДПИС КЛЮЧ РОЗШИФРУВАННЯ ДАНИХ ЗАШИФРУВАННЯ ДАНИХ КРИПТОГРАФІЧНЕ ПЕРЕТВОРЕННЯ АВТЕНТИФІКАЦІЯ ЦІЛІСНІСТЬ ІНФОРМАЦІЇ МОДИФІКАЦІЯ ЗАВІРЕННЯ

ABSTRACT

The text part of the master's thesis: 82 pages, 15 drawings, 40 sources.

The object of research is the application of electronic digital signature in electronic document circulation.

The subject of research is the methods and mechanisms of encryption of an electronic digital signature.

The purpose of the work is to develop recommendations on the effectiveness of using an electronic digital signature according to the "stability - complexity" criterion.

Research methods – study of the literature on this topic, analysis of operating documentation, current solutions and their comparison, conducting an experiment.

The work is gradually solving the main tasks of electronic document flow protection.

The analysis and research of various types of electronic digital signatures and their mechanisms for protecting electronic document flow has been carried out.

The methods and mechanisms of electronic digital signature protection of electronic document circulation have been studied.

The capabilities of the XML Digital Signature API software complex have been studied.

Recommendations for improving the methods of using electronic digital signatures have been developed.

Area of application – cybersecurity of information and telecommunication system.

DIGITAL SIGNATURE KEY DATA DECRYPTION DATA ENCRYPTION CRYPTOGRAPHIC TRANSFORMATION AUTHENTICATION INFORMATION INTEGRITY MODIFICATION VERIFICATION

Список скорочень

ЕД – електронний документ;

ЕЦП – електронний цифровий підпис;

К – ключ;

СЦ – сертифікаційний центр;

ЗД – зашифрування даних;

РД – розшифрування даних;

ЦІ – цілісність інформації;

КП – криптографічний процесор;

КЦ – криптографічний центр;

КІС – корпоративної інформаційної системи.

ЗМІСТ

ВСТУП

Актуальність дослідження. Основна характеристика нової електронної епохи — а відхід від повільного фізичного транспортування матеріалу товарів до світлошвидкісної передачі нематеріальних, але дуже цінна інформація. Нове спілкування системи, і особливо Інтернет, може значно змінити спосіб надання державних послуг громадянам, оскільки вони мають нові можливості підвищення доступу до інформаційної магістралі.

Однак очікується зміна угод і служби все ще затьмарені занепокоєнням щодо безпеки і правова невизначеність. Системи електронної пошти, наприклад, і документи, які вони мають, сприйнятливі до вразливості, такі як витік і модифікація вміст документа або видавання себе за законного користувачів. Щоб мати можливість покладатися на електронні записи, потрібна впевненість, що ці записи є автентичними (тобто що вони дійсно надійшли від особи, яка стверджує, що є відправником) і що вони не були змінено. У більшості випадків також потрібна конфіденційність. Більш того, в ділових операціях це дуже важливо, щоб транзакція була не тільки автентичною, з незмінним змістом, але водночас і відправник не повинен мати можливість заперечити свою дію. Це є загалом відомий під назвою невідмовності. В цих умов безпека стала однією з основні вимоги до комунікацій. через Інтернет.

Об'єкт дослідження – процес побудови системи захисту інформації механізмами електронних цифрових підписів.

Предмет дослідження – методи та механізми протидії модивікації інформації в електронному документообігу.

Мета роботи – розробка рекомендацій щодо вдосконалення електронних цифрових підписів в електронному документообігу.

Завдання магістерської роботи:

Актуальність механізмів захисту інформації електронних цифрових підписів.

Дослідження різних типів електронних цифрових підписів, їх загальних характеристик та механізмів впливу на КІС.

Аналіз методів і шляхів використання ефективних засобів захисту електронного документообігу криптографічними перетвореннями.

Розробка рекомендацій щодо вдосконалення методів застосування електронних цифрових підписів на об’єктах інформаційної діяльності.

Методи дослідження – опрацювання літератури за даною темою, аналіз експлуатаційної документації, актуальних рішень та їх порівняння, проведення експерименту.

Практичне значення одержаних результатів: рекомендації щодо вдосконалення електронних цифрових підписів за критерієм може бути використані у сфері забезпечення кібербезпеки у корпоративних інформаційних системах.

Галузь застосування — кібербезпека інформаційно-телекомунікаційної системи.

Розділ 1. Загальні положення про електронні цифрові підписи

1.1 Загальні поняття ЕЦП

В даний час для захисту електронних документів від підробки і несанкціонованих модифікацій широко використовується електронний цифровий підпис (ЕЦП). У діяльності підприємства, а також в особистих цілях електронний підпис виконує функції, аналогічні до підпису людини. Наявність електронного підпису гарантує, що документ був створений саме цією людиною, тому документ з ЕЦП рівнозначний своєму паперовому аналогу.

Електронний підпис є сукупністю електронних даних, за допомогою якої можна легко і надійно упевнитися в тому, що:

- Саме цей користувач згенерував дану електронний підпис. Жоден інший користувач не може згенерувати таку ж електронний підпис;
- Саме цей документ або набір документів підписав користувач. Документ не був ніяк змінений або замінений після накладення підпису.

Для роботи з електронними підписами використовується пара ключів - закритий та відкритий. Ці ключі генеруються програмно. Закритий ключ зберігається у користувача і нікому не передається, а відкритий реєструється в центрі сертифікації ключів.

Для генерації пари ключів і зберігання секретного ключа загальновизнаних електронних підписів можуть використовуватися спеціальні апаратні пристрої.

У web-додатках підтримка ЕЦП реалізована двома шляхами: з використанням JavaScript-бібліотеки і з використанням агента підпису. Користувач може вибирати реалізацію в залежності від типу ключа (файл або апаратний пристрій). Агент підпису підтримує як файлові ключі, так і апаратні пристрої. JavaScript-бібліотека через політику безпеки браузерів підтримує тільки файлові ключі.

Для накладання ЕЦП в мобільному додатку необхідно записати секретний ключ на мобільний пристрій і налаштувати на роботу з ним мобільний додаток.

Електронний цифровий підпис (ЕЦП) дозволяє підписувати документи онлайн. Таким документом може бути договір між двома суб’єктами

господарювання, акти виконаних робіт, заяви, тощо. Чому електронний документообіг таких популярний? Він значно підвищує ефективність бізнес-процесів за рахунок економії часу та мінімізації помилок та скорочує витрати підприємства на друк та доставку документів.

Не варто плутати електронний цифровий підпис (ЕЦП) із підписом, який ви залишаєте цифровою ручкою на планшеті. ЕЦП не схожий на ваш підпис у паспорті, оскільки це не графічний елемент, а набір цифрових даних. Зазвичай, це файли з розширенням: dat, zs2, sk, jks, pk8, pfx.

Електронний документ (ЕД) — документ, створений за допомогою засобів комп'ютерної обробки інформації, який може бути підписаний електронним цифровим підписом (ЕЦП). Такий документ має повну юридичну силу та регламентується ст. 8 Закону України «Про електронний цифровий підпис».

Електронний цифровий підпис (ЕЦП) – це ваш онлайн-підпис, який отримується криптографічним способом. ЕЦП широко поширений у світі метод захисту та збереження електронного документа від копіювання, модифікації та підробки.

Електронний підпис на електронному документі, створеному за допомогою пристрою електронного підпису, який ідентифікований підписант має унікальне право використовувати для підписання цього документа, якщо цей пристрій не було скомпрометовано, і якщо підписантом є фізична особа, яка уповноважена підписувати документ в силу свого правового статусу та/або його відносин з суб'єктом, на якому від імені підпису виконується.

Пристрій електронного підпису, код або інший механізм, який використовується для створення електронних підписів. Якщо пристрій використовується для створення електронного підпису особи, то код або механізм повинні бути унікальними для цієї особи на момент створення підпису, і вона повинна мати унікальне право на його використання. Пристрій скомпрометований, якщо код або механізм доступний для використання будь-якою іншою особою.

Також передбачено, що системи, які отримують е-документи з

електронними підписами, також повинні демонструвати певні вимоги до функціональності. Система, що схвалюється, повинна бути в змозі забезпечити доказ наступних вимог:

Підпис дійсний на момент підписання Система повинна мати можливість довести, що електронний підпис дійсний на момент підписання. Коли документ підписаний, електронний підпис повинен відповідати вимогам чинного електронного підпису.

Документ не може бути змінений без виявлення після підписання Система повинна мати можливість довести, що електронний документ не може бути змінений без виявлення після підписання. Електронні документи з електронними підписами не можуть бути змінені в будь-який час — під час або після передачі — після підписання без виявлення. Система повинна вміти довести, що зміст документа такий же, як і зміст на момент підписання. В даний час це, як правило, включає в себе якесь програмне забезпечення для шифрування.

Можливість перегляду контенту Система повинна мати можливість довести, що підписанти мали можливість переглядати вміст. Перед фактичним підписанням підписанти повинні мати можливість переглянути зміст, для якого запитується їх підпис.

Можливість перегляду заяви про сертифікацію Система повинна бути в змозі довести, що підписанти розглянули заяву про сертифікацію. Перед фактичним підписанням підписанти повинні мати можливість переглянути заяви про сертифікацію, включаючи попередження про те, що помилкова сертифікація тягне за собою кримінальну відповідальність, щоб встановити, що вони розуміли наслідки свого підпису і мали намір підписати. Це важливо, якщо хтось коли-небудь буде притягнутий до кримінальної відповідальності за кримінальне шахрайство.

Підтвердження квитанції Система повинна мати можливість довести підтвердження отримання. Система автоматично відправляє підтвердження про отримання документа на «позадіапазонну» адресу. Зазвичай це паперова пошта або адреса електронної пошти, яка не має тих самих елементів керування, що й

ті, що використовуються для доступу до облікового запису онлайн-подання. Це гарантує, що якщо випадково пристрій підпису було скомпрометовано, власник пристрою буде повідомлений за межами системи про те, що хтось зробив подання на своє ім'я. Це поширена практика, яку використовують сайти онлайн-покупок — після здійснення покупки на сайті ви отримуєте сповіщення про те, що покупка була здійснена з підтвердженням в окремій системі електронної пошти.

Договори електронного підпису Система повинна мати можливість довести, що підписанти підписали угоди про електронний підпис. Підписанти уклали угоди про електронний підпис, пов'язані з використанням їхніх підписних пристроїв. Договір електронного підпису може бути укладений в електронному вигляді, але також може бути виконаний на папері. Угода повинна включати наступне: Підписант погоджується захистити свій підписний пристрій, такий як пароль або апаратний маркер, від компрометації; Підписант погоджується повідомити про будь-які докази компромісу; і Підписант розуміє, що підпис, який вони подають в електронному вигляді разом із пристроєм, має таку ж юридичну силу та обов'язок, як і власноручний письмовий підпис. Зазвичай підписанти оформляють цю угоду, коли реєструються в системі для отримання свого пристрою електронного підпису.

Підтвердження особи з юридичною визначеністю Система повинна вміти доводити тотожність з юридичною визначеністю. Це вимога, яка служить для встановлення особи фізичної особи, якій видано (або реєструється) пристрій електронного підпису, з достатньою кількістю доказів, які вона буде тримати в суді. Це єдиний випадок у всіх цих вимогах, в якому CROMERR є багаторівневим з точки зору пріоритетних та непріоритетних звітів. Для непріоритетних звітів вимога не вказує, як має здійснюватися підтвердження особи. Для пріоритетних звітів підтвердження особи повинно бути зроблено до виконання підпису, і це має бути зроблено одним із двох визначених методів.

1.2 Аналіз ЕЦП в механізмах безпеки.

Автоматичний засіб електронного цифрового підпису - засіб електронного цифрового підпису, який реалізує в автоматизованих інформаційних системах криптографічні алгоритми накладення та/або перевірки електронного цифрового підпису без участі фізичної особи;

Акредитація - процедура документального засвідчення компетентності центру сертифікації ключів здійснювати діяльність, пов'язану з обслуговуванням посилених сертифікатів ключів;

Блокування сертифіката ключа - тимчасове зупинення чинності сертифіката ключа;

Відкритий ключ електронного цифрового підпису (далі - відкритий ключ) - параметр асиметричного алгоритму криптографічного перетворення, який використовується для перевірки електронного цифрового підпису, доступний користувачу;

Електронний підпис - дані в електронній формі, які додаються до інших електронних даних або логічно з ними пов'язані та призначені для ідентифікації підписувача цих даних;

Електронний цифровий підпис - вид електронного підпису, отриманого в результаті криптографічного перетворення електронних даних, які додаються підписувачем до інших електронних даних або логічно з ними пов'язуються, з метою ідентифікації підписувача, підтвердження цілісності цих даних та/або для автентифікації інформаційних систем та/або осіб, що користуються цими системами. Електронний цифровий підпис накладається за допомогою особистого ключа та перевіряється за допомогою відкритого ключа;

Засвідчення чинності відкритого ключа - процедура формування сертифіката відкритого ключа;

Засіб електронного цифрового підпису - апаратно-програмний або апаратний пристрій чи програмне забезпечення, які реалізують криптографічні алгоритми генерації пар ключів, накладення та/або перевірки електронного цифрового підпису;

Компрометація особистого ключа - будь-яка подія та/або дія, що призвела або може призвести до несанкціонованого використання особистого ключа;

Користувач - фізична або юридична особа, яка використовує електронний цифровий підпис для перевірки цілісності електронних даних та ідентифікації підписувача, ідентифікації та автентифікації інформаційної системи;

Надійний засіб електронного цифрового підпису - засіб електронного цифрового підпису, в якому реалізований безпечний механізм накладення електронного цифрового підпису та має сертифікат відповідності або позитивний експертний висновок за результатами державної експертизи у сфері захисту криптографічної інформації. Підтвердження відповідності та проведення державної експертизи цих засобів здійснюється у порядку, визначеному законодавством;

Особистий ключ електронного цифрового підпису (далі - особистий ключ) - параметр асиметричного алгоритму криптографічного перетворення, який використовується для накладення електронного цифрового підпису, доступний тільки підписувачу;

Підписувач – фізична особа або уповноважений представник юридичної особи, які на законних підставах володіють особистим ключем та від свого імені або за дорученням особи, яку вони представляють, або юридична особа, фізична особа - підприємець, які на законних підставах володіють особистим ключем, що використовується в автоматичному засобі електронного цифрового підпису, та від свого імені, накладають електронний цифровий підпис;

Посилений сертифікат відкритого ключа (далі - посилений сертифікат ключа) - сертифікат ключа, який відповідає вимогам цього Закону та виданий відповідно до правил посиленої сертифікації, що встановлюються центральним засвідчувальним органом та контролюючим органом;

Послуги електронного цифрового підпису - надання у користування засобів електронного цифрового підпису, допомога під час генерації пар ключів, обслуговування сертифікатів ключів (їх формування, розповсюдження, скасування, блокування, поновлення), надання інформації про чинні, блоковані

та скасовані сертифікати ключів, послуги фіксування часу, консультації та інші послуги, визначені цим Законом, що надаються на договірних засадах;

Сертифікат відкритого ключа (далі - сертифікат ключа) - електронний документ, виданий центром сертифікації ключів, засвідчувальним центром чи центральним засвідчувальним органом, який засвідчує чинність і належність відкритого ключа підписувачу. На сертифікат ключа накладається електронний цифровий підпис центру сертифікації ключів, засвідчувального центру чи центрального засвідчувального органу, що його видав;

Сертифікат власного відкритого ключа центрального засвідчувального органу - сформований центральним засвідчувальним органом посилений сертифікат відкритого ключа з використанням відповідного йому особистого ключа центрального засвідчувального органу;

Строк чинності відкритого ключа - проміжок часу, впродовж якого відкритий ключ може використовуватися для перевірки електронного цифрового підпису, який визначається строком чинності особистого ключа електронного цифрового підпису;

Технологічний електронний цифровий підпис - вид електронного цифрового підпису, який призначений для автентифікації інформаційних систем;

Шлях сертифікації відкритого ключа електронного цифрового підпису - послідовність (ланцюжок) сертифікатів, яка сформована таким чином, що:

Перший сертифікат виданий пунктом довіри (центральним засвідчувальним органом);

Останній сертифікат виданий для кінцевого суб'єкта (підписувала) і містить відкритий ключ;

Імена видавців (центрального засвідчувального органу, або засвідчуваального центру, або (акредитованого) центру сертифікації ключів) та суб'єктів (засвідчувального центру, або (акредитованого) центру сертифікації ключів, або підписувала) сертифікатів утворюють послідовність. У всіх сертифікатах цієї послідовності, за винятком першого і останнього, ім'я видавця поточного сертифіката збігається з ім'ям попереднього сертифіката суб'єкта, а

ім'я суб'єкта поточного сертифіката збігається з ім'ям видавця наступного сертифіката".

Автентифікація - процес установлення відповідності між суб'єктом і тим, за кого він себе видає. У випадку вдалого завершення цього процесу можна з упевненістю вважати, що суб'єкт, що перевіряється, дійсно є тим, ким він представляється. Автентификация в протоколі SSL/TLS здійснюється шляхом перевірки електронно-цифрового підпису. ЕЦП суб'єкта, що автентифікує, може бути перевірена за допомогою його відкритого ключа, але при цьому створити коректний підпис можна тільки володіючи закритим ключем суб'єкта, відомим тільки власникові (тобто суб'єктові). Таким чином, знаючи відкритий ключ клієнта і запропонувавши йому підписати блок даних, можна, перевіривши його підпис і точно його ідентифікувати.

Сертифікація - За допомогою описаного вище процесу автентификації можна упевнитися в дійсності джерел спілкування при взаємодії двох об'єктів. Однак для цього необхідно, щоб взаємодіючі об'єкти до початку фактичної передачі даних обмінялися деякою ключовою інформацією. До цієї інформації відносяться використовуваний для автентифікації алгоритм і ключ. Проблема виникає, коли необхідно переконатися в дійсності об'єкта, ніколи до цього не взаємодіяли. Єдиним способом досягти цього є делегування третій стороні права підтвердження такої дійсності. Ця третя сторона називається Засвідчуючого Центру (ЗЦ).

Цифровий сертифікат - це документ, що видає ЗЦ кожному з взаємодіючих об'єктів, дійсність яких він раніше засвідчив. Сертифікат містить інформацію про об'єкт (як, наприклад, назва організації, ім'я (сервера або людини)), його відкритий ключ і саме головне ЕЦП самого ЗЦ, передбачається, що всі учасники обміну безумовно довіряють ЗЦ. Сертифікати можуть служити як для забезпечення безпеки Web-повідомлень, так і для захисту поштових повідомлень і підпису програмного забезпечення. Існують три види цифрових сертифікатів, використовуваних у SSL/TLS: сертифікат сервера (Site Certificate), персональний сертифікат (Personal Certificate) і сертифікат ЗЦ (CA Certificate)

Сертифікат сервера дозволяє упевнитися в дійсності сервера - абстрактного ресурсу.

Персональний сертифікат служить для ідентифікації клієнтів на ресурсі. За допомогою персональних сертифікатів можна забезпечити набагато більш високий рівень безпеки при розмежуванні доступу на сервер, чим за допомогою інших механізмів.

СА сертифікат. Сертифікат ЗЦ служить для перевірки дійсності самого ЗЦ. З його допомогою підписуються сертифікат сервера і персональний сертифікат. Він повинний зберігатися на сервері і на клієнтській машині, для того щоб сервер і клієнтський додаток могли перевірити дійсність підписаних їм ЗЦ сертифікатів.

1.3 ЕЦП в електронному документообігу.

1) Створення підпису та перевірка: ця область зосереджена на стандартах, пов'язаних зі створенням, доповненням та перевірка цифрових підписів, що охоплює:

I. політика та вимоги безпеки для програм створення підпису, доповнення та перевірки;

II. вираження правил і процедур, яких слід дотримуватися під час створення, розширення, перевірки та надовго термін наявності підписів;

III. формати підписів і упаковка підписів і підписаних документів;

IV. профілі захисту відповідно до загальних критеріїв для створення/доповнення/перевірки підпису програми.

2) Створення підпису та інших пов'язаних пристроїв: ця область зосереджена на стандартах, пов'язаних із безпечним підписом пристрої створення, які використовуються постачальниками довірчих послуг (TSP), а також інші типи пристроїв, що підтримують підписи та пов'язані служби, такі як автентифікація.

3) Криптографічні набори: Ця область охоплює аспекти стандартизації, пов'язані з використанням криптографічного підпису. набори, тобто набір пов'язаних з підписом алгоритмів, включаючи алгоритми генерації ключів, алгоритми підпису з параметрами та методом заповнення, алгоритмами перевірки та хеш-функціями.

4) Постачальники довірчих послуг, що підтримують цифрові підписи та пов'язані послуги: це включає видачу TSP сертифікати відкритих ключів (як кваліфікованих ЄС, так і не кваліфікованих) для фізичних і юридичних осіб, у тому числі веб-серверні сертифікати, постачальники послуг із відмітками часу, TSP, що пропонують послуги перевірки підпису, і TSP надання послуг дистанційного створення підпису (також звані серверами підпису). Поточний список охоплює ці послуги підтримка цифрового підпису, що існує на сьогоднішній день; інші довірчі послуги можуть бути визначені в майбутньому.

5) Постачальники довірчих програмних послуг: це охоплює TSP, які

пропонують додаткові послуги із застосуванням цифрових технологій підписи та які покладаються на створення/перевірку підписів у нормальній роботі. Це включає а саме послуги рекомендованої пошти та інші послуги електронної доставки, а також послуги збереження (довгострокового архівування) даних. Це список може бути розширено, оскільки будуть визначені інші служби, що застосовують підписи.

6) Статус довірчої служби перераховує постачальників: Ця область охоплює стандартизацію, пов'язану з наданням довіри, списки статусу послуг і надання довірених списків.

Додаткова область, область 0, як зображено на малюнку 1, збирає цей документ, а також дослідження та інші вступні результати, пов'язані з фреймворком.

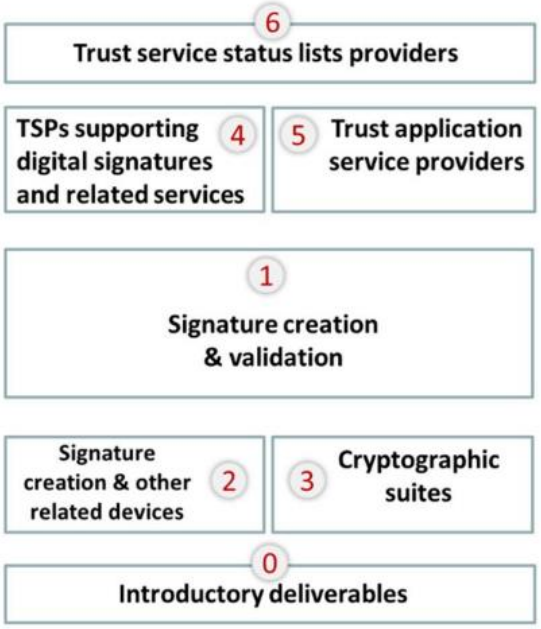


Рис 111 Огляд структури фреймворку для стандартизації підписів

Документи, необхідні для стандартизації кожної з вищевказаних функціональних областей, були організовані навколо такі п'ять типів документів:

- Керівництво: Цей тип документа не містить жодних нормативних вимог, але надає бізнес-орієнтований характер вказівки щодо звернення до сигнатурної (функціональної) області, щодо вибору застосовних стандартів та їх варіанти для конкретного

контексту реалізації бізнесу та пов’язаних з ним вимог до бізнесу впровадження стандарту (або серії стандартів), щодо оцінки впровадження бізнесу проти стандарт (або серія стандартів) тощо.

- Вимоги до політики та безпеки: цей тип документа визначає вимоги до політики та безпеки для служби та системи, включаючи профілі захисту. Це об’єднує використання інших технічних стандартів і вимоги до безпеки, фізичні, процедурні та кадрові вимоги до систем, що реалізують ці технічні стандарти.
- Технічні специфікації: Цей тип документа визначає технічні вимоги до систем. Це включає але не обмежується технічними архітектурами (опис стандартизованих елементів системи та їх взаємозв’язки), формати, протоколи, алгоритми, API, профілі певних стандартів тощо.
- Оцінка відповідності: цей тип документа стосується вимог щодо оцінки відповідності а система, яка заявляє про відповідність певному набору технічних специфікацій, політики або вимог безпеки (включаючи профілі захисту, якщо це можливо). В першу чергу це стосується правил оцінки відповідності (наприклад, загальні критерії оцінки продуктів або оцінки систем і послуг).
- Перевірка відповідності та сумісності: цей тип документа стосується вимог і специфікацій для встановлення тестів сумісності чи систем тестування або для налаштування тестів чи систем тестування, які забезпечать автоматизовані перевірки відповідності продуктів, послуг або систем певним наборам(ам) технічних характеристик.

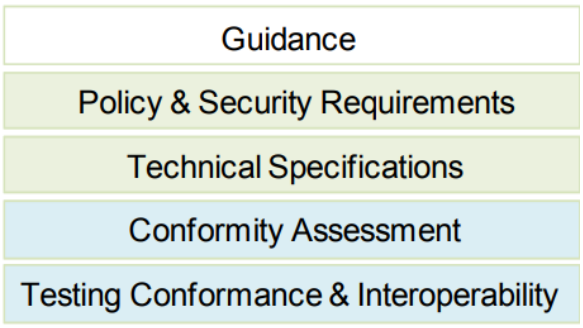


Рис 111 Ілюстрація типів документів у рамках стандартизації підписів

1.3.1 Структурна схема ЗЦ

ЗЦ є програмно - апаратним комплексом, що виконує функції формування сертифіката ключа підпису. Ключі можуть створюватися як зовні на відповідному програмному забезпеченні користувача, так і засобами ЗЦ. Носієм інформації про випущений сертифікат виступає відчужуваний носій (Floppy Disk, або інший носій, використання якого визначається специфікою автоматизованої системи).

ЗЦ спроектований і повинний експлуатуватися, як довірена, замкнута, автоматизована система, що модифікується не. Під замкнутістю розуміється неможливість впливу на склад і компоненти системи іншим способом, крім як визначеним регламентом образом. Під немодифікованістю розуміється відсутність у системі засобів розробки прямо або побічно впливають на функціональну однозначність виконуваних процедур.

ЗЦ побудований по модульному принципі і містить у своєму складі наступні компоненти (підсистеми):

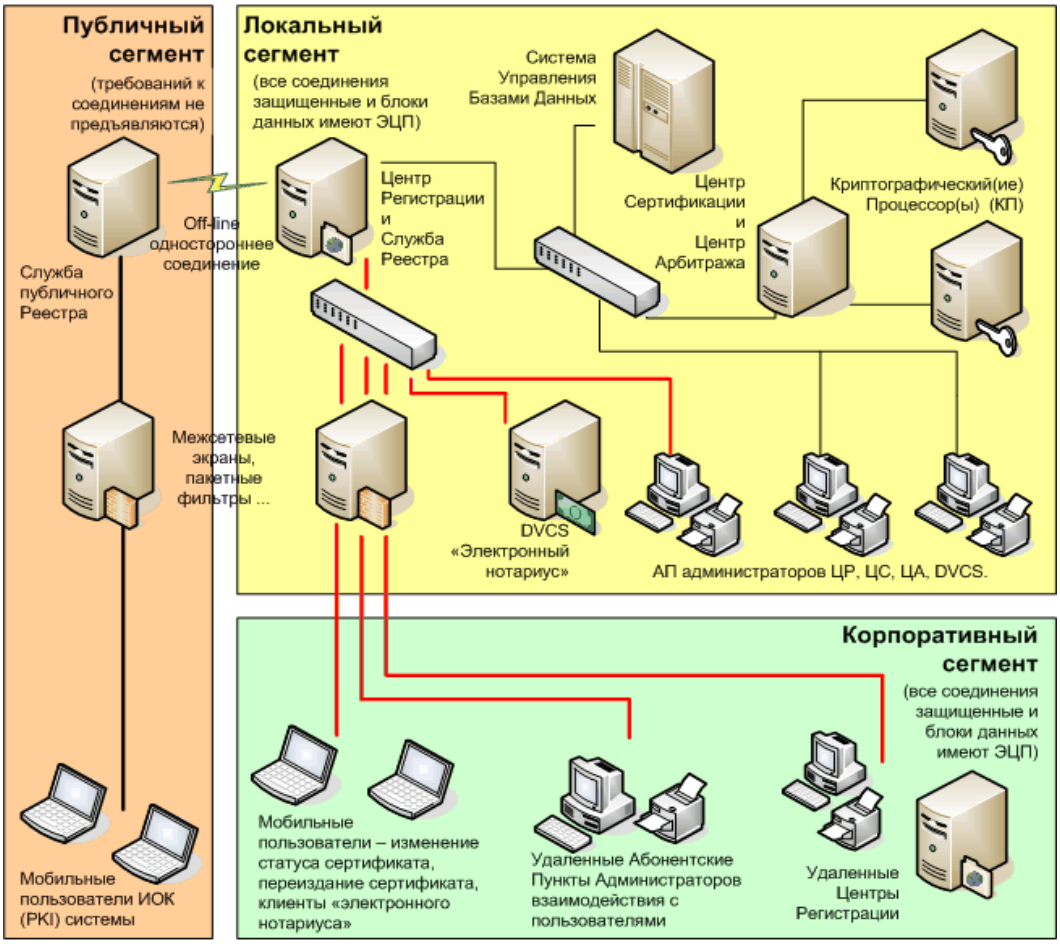


Рис. 1.1 Структурна схема ЗЦ

Підсистеми ЗЦ виконані з захистом від НСД. Контроль за незмінністю програмного середовища здійснює спеціалізований сервіс, що конфігурується при інсталяції ЗЦ і веде робочі журнали контролю цілісності ПО, доступ до яких має обслуговуючий персонал ЗЦ.

1.3.2 Приклади побудови систем заснованих на інфраструктурі відкритих ключів

Однією з технологій, що дозволяє вирішити більшість проблем зв'язаних з безпекою в комп'ютерних мережах, є інфраструктура відкритих ключів.

За допомогою інфраструктури відкритих ключів вирішуються наступні задачі:

- Забезпечення приховання переданої інформації.
- Забезпечення цілісності цієї інформації, тобто захист від

модифікації. Існують технічні рішення, що дозволяють розміщати на серверах ресурсу завірену інформацію, має електронний цифровий підпис, перевірка якого (у тому числі й у ON-LINE) дозволяє підтвердити цілісність і авторство розміщеної інформації.

- Забезпечення автентифікації ресурсу, тобто доказ того, що ресурс є саме тим, за кого він себе видає.
- Забезпечення автентифікації користувачів при доступі до ресурсу.

Узявши за основу ці корисні властивості, реалізовані в системі, можна побудувати систему захищеного електронного документообігу, у якій поняття документообігу багато ширше чим простий рух наказів або листів усередині організації.

Рис. 1.2 Система захищеного електронного документообігу

Запропонована схема дозволяє вирішити IT задачі організацій фактично будь-якого профілю, а використання типових стандартних рішень роблять розробки прозорими для розуміння й аналізу.

1.3.3 Перевірка дійсності ЕЦП на реальний момент часу

Перевірка дійсності ЕЦП на реальний момент часу реалізований на основі рекомендацій RFC 3029 по роботі зі спеціалізованим сервісом "Електронного нотаріуса". Основна форма пропонує висновок відсортованих запитів (або запитів у зв'язуванні з DVC квитанціями) по фільтру - ім'я сервера DVCS. З операцій над записами доступні:

Новий запит:

- Ініціює майстер DVC заявок на перевірку ЭЦП;
- Відкрити відзначену заявку або квитанцію;
- Видалити запит цілком (у зв'язуванні з квитанцією) або тільки квитанцію;

Експортувати DVC заявку або квитанцію.

1.3.4 COM – технологія

COM - це Component Object Model - компонентна об'єктна модель. Суттю використання даної технології є те, що програми будуються з компонентів, що складаються з об'єктів, і доступ до зареєстрованих об'єктів Криптографічного Центра (КЦ) на обробку завірених документів стає доступний як з HTML форм, так і з інших програм установлених на комп'ютері користувача.

1.3.5 ЕЦП HTML форми

Використання COM об'єктів зі складу ПО для виконання процедур вироблення і перевірки електронного цифрового підпису над інформацією з HTML форм дозволяє ефективним образом вирішити багато задач з областей, що базуються на Web-технологіях, таких як, вилучене керування електронними платежами (системи типу "Банк-Клієнт") і будь-яких інших, де необхідно авторизований запит із завіреними даними на обслуговування, наприклад системи електронної торгівлі, різні системи керування об'єктами, платного інформаційного забезпечення, різні платіжні і т.п.

2.1 Аналіз методів і шляхів захисту електронного документообігу криптографічними перетвореннями.

2.1.1Методи, які використовуються для криптоаналізу

Криптоаналіз — це дослідження та процес аналізу та дешифрування шифрів, кодів і зашифрованого тексту без використання справжнього ключа. Крім того, ми можемо сказати, що це техніка доступу до простого текстового вмісту повідомлення, коли ви не маєте доступу до ключа розшифровки.

Простіше кажучи, криптоаналіз — це практика, наука або мистецтво дешифрування зашифрованих повідомлень.

Фахівці з криптоаналізу вивчають шифри, криптосистеми та шифротекст, щоб зрозуміти їхні функції. Потім вони використовують ці знання, щоб знайти або вдосконалити методи, щоб послабити або перемогти їх. Однак, як ми побачимо, його можна використовувати як у добрих, так і в поганих цілях.

Отже, криптограф — це той, хто пише код шифрування, який використовується в кібербезпеці, тоді як криптоаналітик — це той, хто намагається зламати ці коди шифрування. Дві протилежні сторони медалі кібербезпеки, замкнені у конфлікті, намагаючись перебороти одну, постійно винаходячи нові заходи та контрзаходи. Це суперництво стимулює інновації в галузі кібербезпеки.

Не дивно, що хакери використовують криптоаналіз. Потенційні хакери використовують криптоаналіз для викорінення вразливостей криптосистеми, а не атаку грубою силою. Уряди використовують криптоаналіз для розшифровки зашифрованих повідомлень інших країн. Компанії, що спеціалізуються на продуктах і послугах кібербезпеки, використовують криптоаналіз для перевірки своїх функцій безпеки. Навіть науковий світ бере участь у дій, дослідники та академіки шукають слабкі місця в криптографічних алгоритмах і протоколах.

Говорячи про хакерів, слід зазначити, що як чорні, так і білі хакери використовують криптоаналіз. Хакери «чорного капелюха» використовують його для вчинення кіберзлочинів, а хакери «білого капелюха» використовують

його для проведення тестування на проникнення за вказівками організацій, які наймають їх для перевірки їх безпеки.

2.1.1.1 Метод грубої сили

Груба сила - це те, що раніше описувало пошук відповіді без скорочень і елегантності. Наприклад, у нас є формули, які допомагають підсумовувати послідовні цілі числа, які можна змінити для таких речей, як ряди кратних чисел. Однак вони не працюють для менш акуратно розробленого набору чисел, як-от підсумовування значень у безперервному наборі даних, щоб отримати середнє значення — вам потрібно просто робити суми по одному або використовувати комп'ютер для підсумовування їх. Оскільки цей метод є прямим і не потребує нічого, крім точності — жодної уяви чи розуміння, — його називають «грубою силою» як метафору.

Загалом, чим вище ви просуваєтеся в математиці, тим більше дізнаєтеся про швидкі шляхи та про те, як створювати власні скорочення на основі структур, властивих математиці. Один професор сказав нам: «Математика — це єдина галузь, у якій лінь називають красою й елегантністю». Доведення — це, в основному, формальний спосіб, яким математик показує, що його ярлик надійний.

2.1.1.2 Метод зашифрованого тексту

У криптографії атака лише зашифрованим текстом (COA) або атака зашифрованим текстом — це модель атаки для криптоаналізу , де передбачається, що зловмисник має доступ лише до набору зашифрованих текстів. Хоча зловмисник не має каналу доступу до відкритого тексту до шифрування, у всіх практичних атаках лише зашифрованим текстом зловмисник все ще має певні знання про відкритий текст. Наприклад, зловмисник може знати мову, якою написаний відкритий текст, або очікуваний статистичний розподіл символів у відкритому тексті. Дані та повідомлення стандартного протоколу зазвичай є частиною відкритого тексту в багатьох розгорнутих системах і

зазвичай можуть бути вгадані або ефективно відомі в рамках атаки лише зашифрованого тексту на ці системи.

2.1.1.3 Метод відомого відкритого тексту

Цю атаку легше реалізувати порівняно з атакою лише зашифрованого тексту. При відомій атаці відкритого тексту аналітик, швидше за все, має доступ до частини або всього відкритого тексту зашифрованого тексту. Мета криптоаналітика полягає в тому, щоб виявити ключ, який мета використовує для шифрування повідомлення, і використати ключ для розшифровки повідомлення. Після виявлення ключа зловмисник може розшифрувати кожне повідомлення, зашифроване цим конкретним ключем. Відомі атаки відкритого тексту покладаються на те, що зловмисник знаходить або вгадує все або частину зашифрованого повідомлення або, навпаки, навіть вихідний формат відкритого тексту.

2.1.1.4 Метод Chosen-Plaintext

Атака з вибраним відкритим текстом (CPA) — це модель атаки для криптоаналізу, яка передбачає, що зловмисник має можливість вибирати довільні відкриті тексти для шифрування та отримувати відповідні зашифровані тексти. Метою атаки є отримання додаткової інформації, яка знижує безпеку схеми шифрування. У гіршому випадку атака з використанням обраного відкритого тексту може відкрити секретний ключ схеми.

Це, на перший погляд, здається нереальною моделлю; звичайно малоймовірно, що зловмисник зможе переконати людину-криптографа зашифрувати велику кількість відкритих текстів за вибором зловмисника. Сучасна криптографія, з іншого боку, реалізована в програмному або апаратному забезпеченні та використовується для різноманітних програм; у багатьох випадках атака обраним відкритим текстом часто цілком здійсненна. Атаки за допомогою вибраного відкритого тексту стають надзвичайно важливими в контексті криптографії з відкритим ключем, де ключ шифрування є відкритим і зловмисники можуть зашифрувати будь-який відкритий текст, який вони

виберуть.

Будь-який шифр, який може запобігти атакам обраного відкритого тексту, також гарантовано захищений від атак відомого відкритого тексту та лише зашифрованого тексту; це консервативний підхід до безпеки.

Можна виділити дві форми атаки вибраного відкритого тексту:

• Пакетна атака з вибраним відкритим текстом, коли криптоаналітик вибирає всі відкриті тексти перед тим, як будь-який із них буде зашифровано. Це часто означає некваліфіковане використання «атаки обраного відкритого тексту».

• Адаптивна атака вибраного відкритого тексту, коли криптоаналітик виконує серію інтерактивних запитів, вибираючи наступні відкриті тексти на основі інформації з попередніх шифрувань.

Нерандомізовані (детерміновані) алгоритми шифрування з відкритим ключем уразливі до простих атак типу «словник», коли зловмисник створює таблицю ймовірних повідомлень і відповідних їм зашифрованих текстів. Щоб знайти розшифровку деякого спостережуваного зашифрованого тексту, зловмисник просто шукає зашифрований текст у таблиці. Як наслідок, визначення безпеки з відкритим ключем під час атаки обраного відкритого тексту вимагає ймовірнісного шифрування (тобто рандомізованого шифрування). Звичайні симетричні шифри, в яких той самий ключ використовується для шифрування та дешифрування тексту, також можуть бути вразливими до інших форм атаки за допомогою вибраного відкритого тексту, наприклад, диференціального криптоаналізу блокових шифрів.

Під час Другої світової війни зловмисники союзників, які розгадували повідомлення, зашифровані на машині Enigma, використовували техніку під назвою Gardening. Садівництво можна розглядати як атаку вибраного відкритого тексту.

2.1.1.5 Метод обраного зашифрованого тексту

Атака за допомогою вибраного зашифрованого тексту (CCA) — це модель атаки для криптоаналізу, у якій криптоаналітик збирає інформацію, принаймні

частково, шляхом вибору зашифрованого тексту та отримання його розшифровки під невідомим ключем. Під час атаки зловмисник має шанс ввести в систему один або кілька відомих зашифрованих текстів і отримати отримані відкриті тексти. З цієї інформації зловмисник може спробувати відновити прихований секретний ключ, який використовується для дешифрування.

Кілька інших безпечних схем можна знищити під час атаки за допомогою обраного зашифрованого тексту. Наприклад, криптосистема El Gamal є семантично захищеною під час атаки вибраного відкритого тексту, але ця семантична безпека може бути тривіально порушена під час атаки вибраного зашифрованого тексту. Ранні версії заповнення RSA, що використовувалися в протоколі SSL, були вразливі до складної адаптивної атаки вибраного зашифрованого тексту, яка розкривала ключі сеансу SSL. Атаки за допомогою обраного зашифрованого тексту також мають наслідки для деяких самосинхронізованих потокових шифрів. Розробники стійких до втручання криптографічних смарт-карт повинні бути особливо уважними до цих атак, оскільки ці пристрої можуть бути повністю під контролем зловмисника, який може видати велику кількість вибраних зашифрованих текстів у спробі відновити прихований секретний ключ.

Якщо криптосистема вразлива до атаки обраного зашифрованого тексту, розробники повинні бути обережними, щоб уникнути ситуацій, у яких зловмисник міг би розшифрувати вибраний зашифрований текст (тобто уникати надання оракула дешифрування). Це може бути складніше, ніж здається, оскільки навіть частково вибрані зашифровані тексти можуть допускати непомітні атаки. Крім того, деякі криптосистеми (такі як RSA) використовують той самий механізм для підписання повідомлень і для їх дешифрування. Це дозволяє атаки, коли хешування не використовується для повідомлення, яке підписується. Кращим підходом є використання криптосистеми, яка доведено безпечна під час атаки вибраного зашифрованого тексту, включаючи (серед іншого) RSA-OAEP, Cramer-Shoup і багато форм автентифікованого симетричного шифрування.

2.1.1.5.1 Різновиди атак Chosen-Cyphertext

Атаки за допомогою вибраного шифротексту, як і інші атаки, можуть бути адаптивними або неадаптивними. Під час неадаптивної атаки зловмисник заздалегідь вибирає зашифрований текст або зашифровані тексти для розшифровки, і не використовує отримані відкриті тексти для інформування про свій вибір щодо додаткових зашифрованих текстів. Під час адаптивної атаки з вибраним зашифрованим текстом зловмисник робить свій вибір зашифрованого тексту адаптивно, тобто залежно від результату попередніх розшифрувань.

2.1.1.5.1.1 Атаки під час обіду

Особливо відзначеним варіантом атаки вибраного зашифрованого тексту є атака «в обідній час», «опівночі» або «індиферентна» атака, під час якої зловмисник може робити адаптивні запити вибраного зашифрованого тексту, але лише до певного моменту, після чого зловмисник повинен продемонструвати деяку покращену здатність атакувати систему. Термін «атака в обідній час» означає ідею, що комп'ютер користувача з можливістю дешифрування доступний для зловмисника, поки користувач не обідає. Ця форма атаки була першою, яку зазвичай обговорювали: очевидно, якщо зловмисник має можливість робити адаптивні вибрані запити зашифрованого тексту, жодне зашифроване повідомлення не буде безпечним, принаймні до тих пір, поки ця можливість не буде вилучена. Цю атаку іноді називають «атакою неадаптивного обраного зашифрованого тексту»; тут «неадаптивний» означає той факт, що зловмисник не може адаптувати свої запити у відповідь на виклик, який надається після закінчення терміну дії можливості робити вибрані запити зашифрованого тексту.

2.1.1.5.1.2 Адаптивна атака зашифрованого тексту

(Повна) адаптивна атака з вибраним зашифрованим текстом — це атака, під час якої зашифровані тексти можуть вибиратися адаптивно до та після того, як запитуваний зашифрований текст надається зловмисникові, лише з умовою, що сам запитувальний зашифрований текст не може бути запитаний. Це сильніша атака, ніж атака в обідній час, і її зазвичай називають атакою CCA2

порівняно з атакою CCA1 (в обідній час). Небагато практичних атак мають таку форму. Скоріше, ця модель важлива для її використання в доказах захисту від атак обраного зашифрованого тексту. Доказ того, що атаки в цій моделі неможливі, означає, що будь-яка реалістична атака за допомогою обраного зашифрованого тексту не може бути виконана.

2.1.1.6 Метод зустрічі посередині

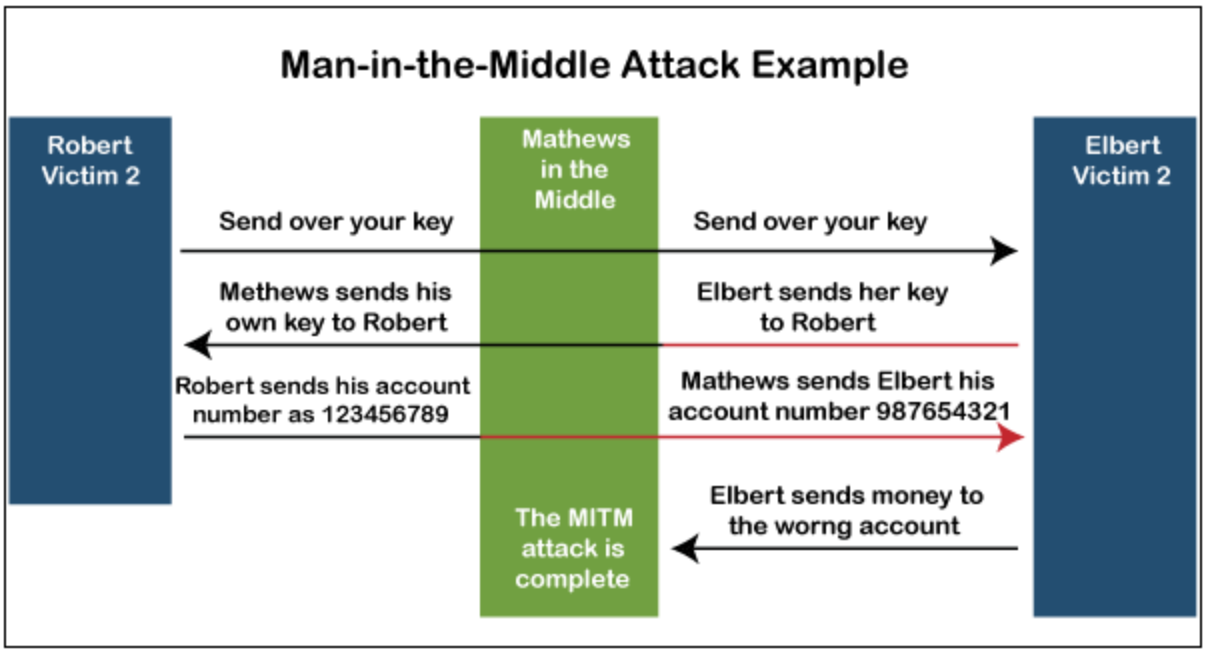


Рис 111 Meet-in-the-Middle method

Атака MITM — це форма кібератаки, коли зловмисник знайомить користувача з певною зустріччю між двома сторонами, маніпулює обома сторонами та отримує доступ до даних, які двоє людей намагалися доставити один одному. Атака типу "людина посередині" також допомагає зловмисникові зламати передачу даних, призначених для когось іншого, які взагалі не повинні надсилатися, без розпізнавання будь-якого учасника, поки не стане надто пізно. У певних аспектах, як-от MITM, MitM, MiM або MIM, можна посилатися на атаки MITM.

Якщо зловмисник опиняється між клієнтом і веб-сторінкою, відбувається атака "Людина посередині" (MITM). Ця форма нападу має багато різних способів.

Щоб перехопити фінансові облікові дані для входу, можна використовувати шахрайський банківський веб-сайт. Між користувачем і реальною веб-сторінкою банку фальшивий сайт знаходиться «посередині».

Існує кілька причин і стратегій використання хакерами атаки MITM. Зазвичай вони намагаються отримати доступ до будь-чого, наприклад до номерів кредитних карток або даних для входу в систему. Вони також шпигують за приватними зустрічами, які можуть містити корпоративні таємниці або іншу корисну інформацію.

Особливістю майже кожної атаки є те, що зловмисник видає себе за людину, якій ви довіряєте (або веб-сторінку).

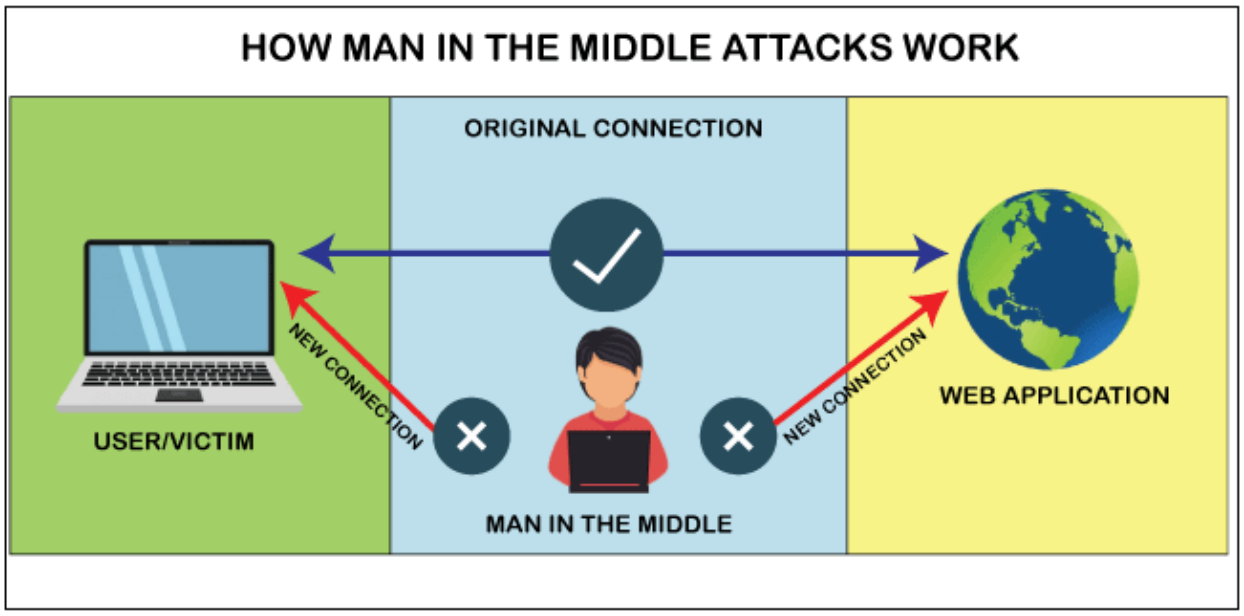


Рис 111 How man in the middle attacks work

Прослуховування Wi-Fi

Можливо, ви бачили сповіщення з пропозицією «Це з'єднання небезпечне», якщо ви користувалися пристроєм у кафе. Громадський Wi-Fi зазвичай пропонується «як є», без будь-яких обіцянок щодо якості обслуговування.

За незашифрованими мережами Wi-Fi легко спостерігати. Хоча це схоже на дискусію в громадському місці – будь-хто може приєднатися. Ви можете обмежити свій доступ, встановивши для свого комп'ютера режим «публічний»,

що вимикає мережеве виявлення. Це запобігає використанню системи іншими користувачами мережі.

Деякі інші атаки Wi-Fi snooping відбуваються, коли зловмисник встановлює власну точку доступу Wi-Fi "Evil Twin". Зловмисник робить посилання через мережеву адресу та паролі ідентичними справжнім. Користувачі будуть посилатися на «злісного близнюка» ненавмисно або автоматично, дозволяючи зловмиснику втручатися в їхні дії.

Підробка DNS

Сайт працює з числовими IP-адресами, наприклад 192.156.65.118 є однією з адрес Google.

Наприклад, сервер використовується кількома сайтами для інтерпретації адреси у впізнаваний заголовок: google.com. DNS-сервер або DNS – це сервер, який перетворює 192.156.65.118 на google.com.

Зловмисник може створити шахрайський веб-сервер. Шахрайський сервер переносить певну веб-адресу на унікальну IP-адресу, що називається «спуфінгом».

IP-спуфінг

Багато пристроїв, підключених до однієї мережі, містять IP-адресу, як ми всі знаємо. Кожен пристрій має свою IP-адресу в кількох внутрішніх мережах підприємства. У IP-спуфінгу зловмисники імітують затверджену IP-адресу консолі. Для мережі це відображається так само, як система авторизована.

Це може бути причиною несанкціонованого доступу до мережі. Вони повинні зберігати тишу та відстежувати дії, інакше також може бути здійснена атака типу «Відмова в обслуговуванні» (DoS). У атаці Middle-in-the-man IP-спуфінг також може використовуватися шляхом розміщення між двома пристроями.

Наприклад, пристрій A і пристрій B припускають, що вони спілкуються один з одним, але обидва перехоплюються та повідомляються зловмиснику.

Пристрій А= = = = Зловмисник= = = = Пристрій В

Згідно зі звітами IBM X-Force Threat Intelligence 2018, 35 відсотків операцій зі вторгненнями включають хакерів, які використовують експлойти MITM. Це представлено на круговій діаграмі нижче.

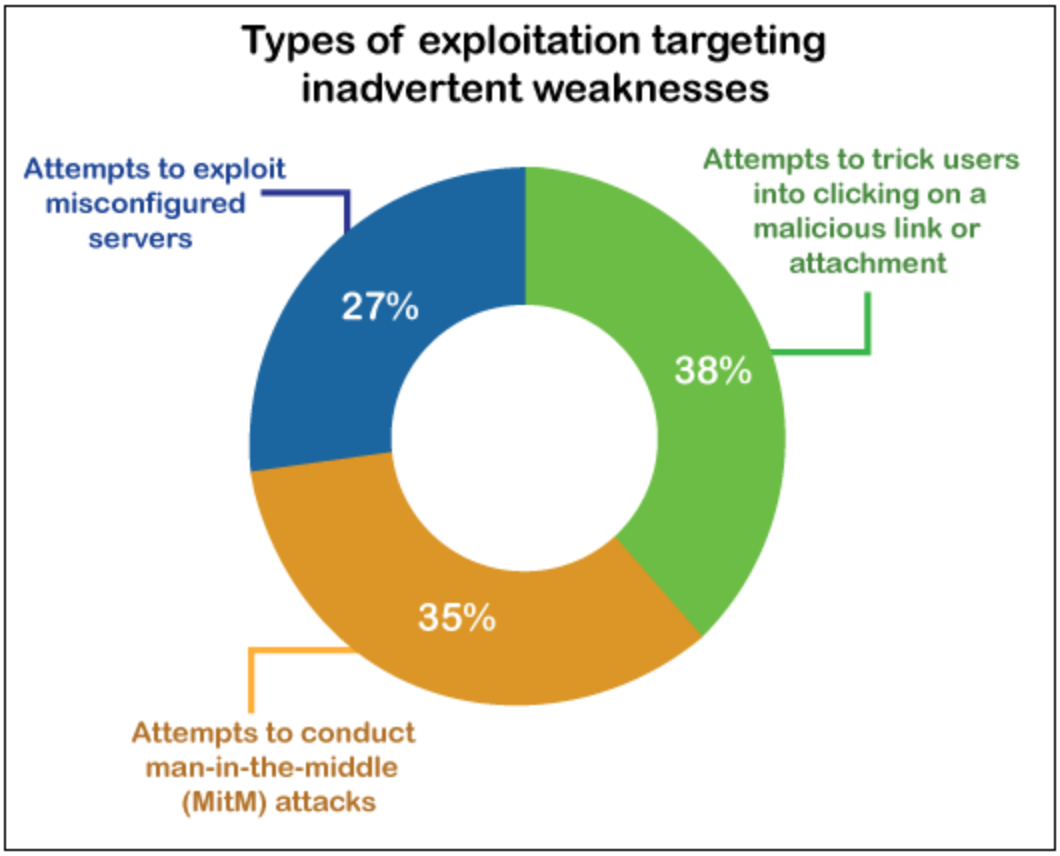


Рис 111 Types of exploitation targeting inadvertent weaknesses

Підробка HTTPS

Копіювання веб-сторінки HTTPS наразі неможливо. Проте теоретичний підхід до обходу HTTPS був проілюстрований експертами з кібербезпеки. Зловмисник створює авторитетну адресу.

Він використовує літери міжнародних алфавітів, а не стандартні сценарії. Це діє як фішингові електронні листи з незвичними символами, які ви могли використовувати. Rolex може бути написано Rólex, наприклад.

Підробка ARP

ARP відноситься до протоколу розпізнавання адрес.

ARP-запит надсилається клієнтом, а зловмисник створює шахрайську

відповідь. Зловмисник у цій ситуації схожий на комп'ютерний модем, який дає зловмиснику доступ до потоку трафіку. Зазвичай це обмежено локальними мережами (LAN), які використовують протокол ARP.

Злом електронної пошти

Зловмисник використовує систему електронної пошти користувача для такого роду вторгнення в кібербезпеку. Зловмисник також тихо спостерігає, збираючи дані та підслуховуючи обговорення електронною поштою. Зловмисники можуть мати шаблон сканування, який шукає цільові ключові слова, такі як «фінансова» або «прихована політика демократів».

Завдяки соціальній інженерії злом електронної пошти працює ідеально. Щоб імітувати онлайн-друга, зловмисники можуть використовувати відповідні дані з якоїсь зламаної електронної адреси. Фішинг також можна використовувати, щоб обманом змусити користувача завантажити шкідливі програми.

Злом сесії

Зазвичай ця форма атаки MITM часто використовується для злому платформ соціальних мереж. Веб-сторінка містить «сеансовий файл cookie браузера» на комп'ютері жертви для більшості платформ соціальних мереж. Якщо особа відходить, цей файл cookie відхиляється. Але під час сеансу файл cookie пропонує ідентифікаційні дані, дані про вплив і моніторинг.

Викрадення сеансу відбувається, коли конфігураційний файл cookie викрадено зловмисником. Це може виникнути, якщо обліковий запис жертви не зламано зловмисними програмами або програмами. Це може статися, якщо користувач використовує XSS крос-скриптове вторгнення, під час якого хакер впроваджує шкідливий сценарій на сайт, який часто відвідують.

Зачистка SSL

SSL відноситься до рівня захищених сокетів. SSL — це стандарт безпеки, який використовується, якщо ви бачите https:/ поруч із адресою веб-сайту, а не http:/. Зловмисник отримує доступ і направляє пакети даних від користувача за допомогою SSL Stripping:

Користувач = = = = Зашифрований веб-сайт Користувач = = = = Автентифікований веб-сайт

Користувач намагається перейти на захищений веб-сайт. В обліковому записі клієнта зловмисник шифрує та посилається на захищений веб-сайт. Зазвичай зловмисник розробляє підроблений дизайн, щоб представити його клієнту. Жертва думає, що вона ввійшла на звичайний веб-сайт, але насправді вона ввійшла на веб-сайт хакера. Зловмисник справді «вилучає» сертифікат SSL із з'єднання даних жертви.

Атака MITB

Це форма атаки, яка використовує недоліки безпеки веб-переглядача.

Зловмисними атаками будуть трояни, настільні хробаки, уразливості Java, атаки SQL-ін'єкції та додатки для веб-перегляду. Вони зазвичай використовуються для збору фінансової інформації.

Зловмисне програмне забезпечення викрадає їхні паролі, коли користувач входить у свій банківський рахунок. У деяких випадках сценарії зловмисного програмного забезпечення можуть переміщувати гроші, а потім змінювати отримання транзакції, щоб приховати транзакцію.

Виявлення атаки Man-in-the-middle

Без вжиття відповідних заходів визначити атаку MITM важче. Напад "людина посередині" теоретично триватиме безконтрольно, доки не стане надто пізно, коли вам не потрібно буде свідомо оцінювати, чи відстежували вашу взаємодію. Зазвичай основною технікою виявлення потенційних атак завжди є пошук відповідної авторизації сторінки та запровадження певного типу автентифікації; однак ці підходи можуть потребувати подальшого судово-медичного дослідження постфактум.

Замість того, щоб намагатися ідентифікувати атаки, коли вони діють, необхідно керувати запобіжними заходами, щоб уникнути атак MITM, коли вони трапляються. Щоб підтримувати безпечне середовище, пам'ятаючи про свої

звички серфінгу та визначаючи можливо небезпечне середовище, може бути важливо.

2.1.1.7 Частотний аналіз англійського алфавіту

Методологія частотного аналізу базується на тому факті, що в будь-якій мові кожна літера має своюіндивідуальність. Найбільш очевидною рисою, якою володіють літери, є частота, з якою вони з'являються в мові. Зрозуміло, що в англійській мові буква «Z» зустрічається набагато рідше, ніж, скажімо, «A». У минулі часи, якщо ви хотіли дізнатися частотність літер у мові, вам потрібно було знайти великий фрагмент тексту та підрахувати кожну частотність. Однак тепер у нас є комп'ютери, які можуть виконувати важку роботу за нас. Але насправді нам навіть не потрібно робити цей крок, оскільки для більшості мов існують бази даних частот літер, які були розраховані на основі перегляду мільйонів текстів і, отже, дуже точні.

З цих баз даних ми виявили, що «E» є найпоширенішою літерою в англійській мові, вона з'являється приблизно в 12% випадків (тобто трохи більше однієї з десяти букв є «E»). Наступною за поширеністю літерою є «T» (9%).

Ми можемо використовувати цю інформацію, щоб допомогти нам зламати код, наданий одноалфавітним шифром підстановки. Це працює, тому що якщо "e" було зашифровано в "X", то кожен "X" був "e". Отже, найпоширенішою літерою в зашифрованому тексті має бути «X».

Таким чином, якщо ми перехоплюємо повідомлення, і найпоширенішою літерою є "P", ми можемо здогадатися, що "P" було використано для шифрування "e", і таким чином замінити всі "P" на "e". Звичайно, не кожен текст має однакову частоту, і, як видно вище, "t" і "a" також мають високі частоти, тож можливо, що "P" був одним із них. Однак навряд чи це буде "z", оскільки це рідко зустрічається в англійській мові. Повторюючи цей процес, ми можемо досягти значного прогресу в розкритті повідомлення.

Якби ми просто розставили всі літери по порядку та замінили їх, як у частотах, це, швидше за все, спричинило б лепет. Зловмиснику доводиться

використовувати інші «особистісні риси» листів, щоб розшифрувати повідомлення. Це може включати розгляд загальних пар літер (або диграфів): слів із 2 літер небагато; є лише кілька літер, які виглядають як подвійні (SS, EE, TT, OO та FF є найпоширенішими). В англійській мові є лише два розумних слова, які складаються з однієї літери. Інші загальні слова також починають з’являтися, коли ви робите деякі заміни. Наприклад, "tKe" може часто з'являтися після заміни "t" і "e". Дуже ймовірно, що це "the", дуже поширене слово в англійській мові.

Процес частотного аналізу використовує різноманітні тонкі властивості мови, і з цієї причини майже неможливо доручити комп’ютеру виконувати всю роботу. Неминуче в цьому процесі необхідна участь людини, щоб приймати обґрунтовані рішення про те, які букви замінити.

2.2 Аналіз забезпечення цілісності інформації механізмами криптографічного перетворення за критерієм «стійкість – складність».

Криптографія — це практика та вивчення безпечного зв'язку в присутності третіх сторін.[1] У минулому криптографія стосувалася переважно шифрування. Шифрування — це процес перетворення звичайної текстової інформації на зашифрований текст. Зворотним є розшифрування. Шифрування — це механізм, який робить інформацію конфіденційною для будь-кого, крім розшукуваних одержувачів. Шифр — це пара алгоритмів, яка створює шифрування та дешифрування. Робота шифру залежить від алгоритму та ключа. Ключ - таємниця, яку знають комуніканти. Крім того, існує два типи шифрування за використовуваними ключами:

2.2.1 Алгоритми симетричного ключа

Алгоритми симетричного ключа (криптографія із закритим ключем): той самий ключ використовується для шифрування та дешифрування. (AES, DES тощо) (AWS KMS використовує Symmetric Key Encryption для виконання шифрування та дешифрування цифрових даних)

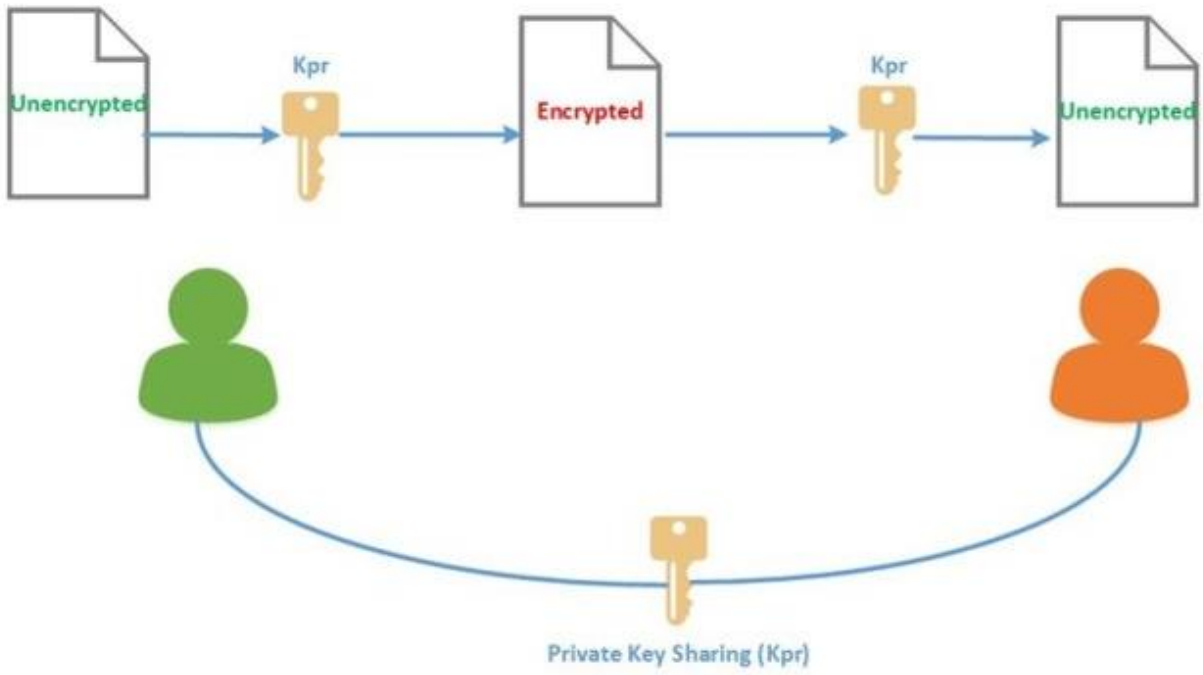


Рис 111

Симетричні ключові шифри, реалізовані як блокові або потокові шифри за типом вхідних даних. Блоковий шифр шифрує введення у вигляді блоків

відкритого тексту на відміну від окремих символів, форма введення, яку використовує потоковий шифр.

Блокові шифри : шифрують блок даних фіксованого розміру. (DES, AES тощо)

Потокові шифри : шифрують безперервні потоки даних. (RC4 тощо)

Ентропія вимірює, наскільки випадковими є дані при використанні в криптографії.

2.2.2 Алгоритми з асиметричним ключем

Алгоритмииз асиметричним ключем (криптографія з відкритим ключем): два різні ключі (приватний і відкритий), що використовуються для шифрування та дешифрування. (RSA, еліптична крива тощо) (пари ключів AWS EC2 використовують асиметричне шифрування.)

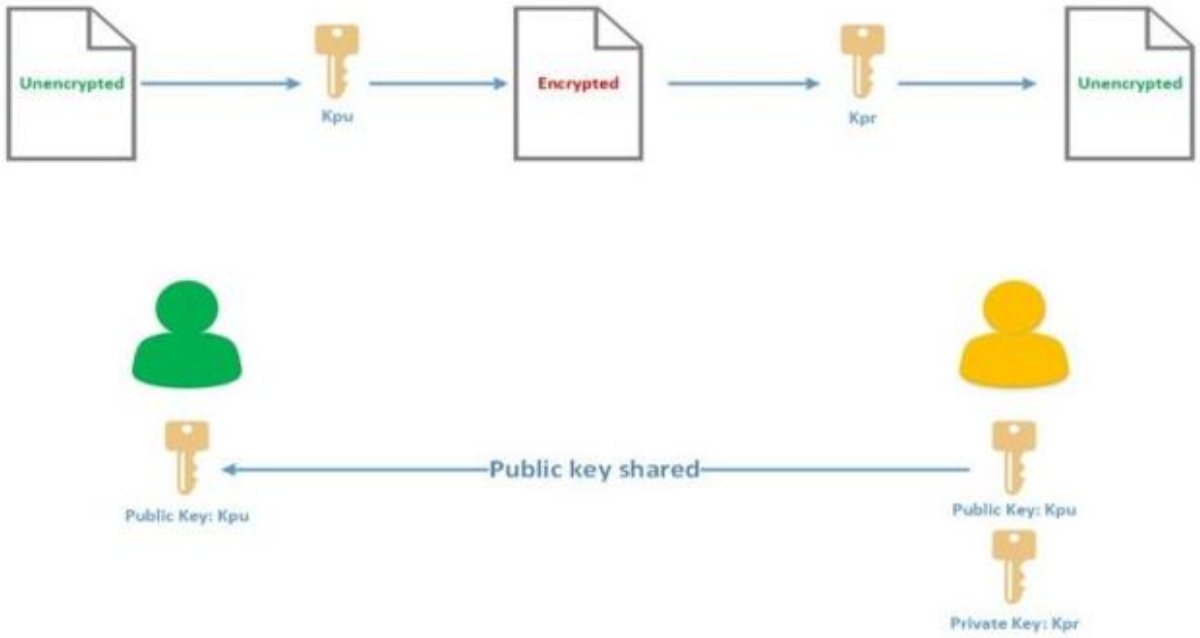
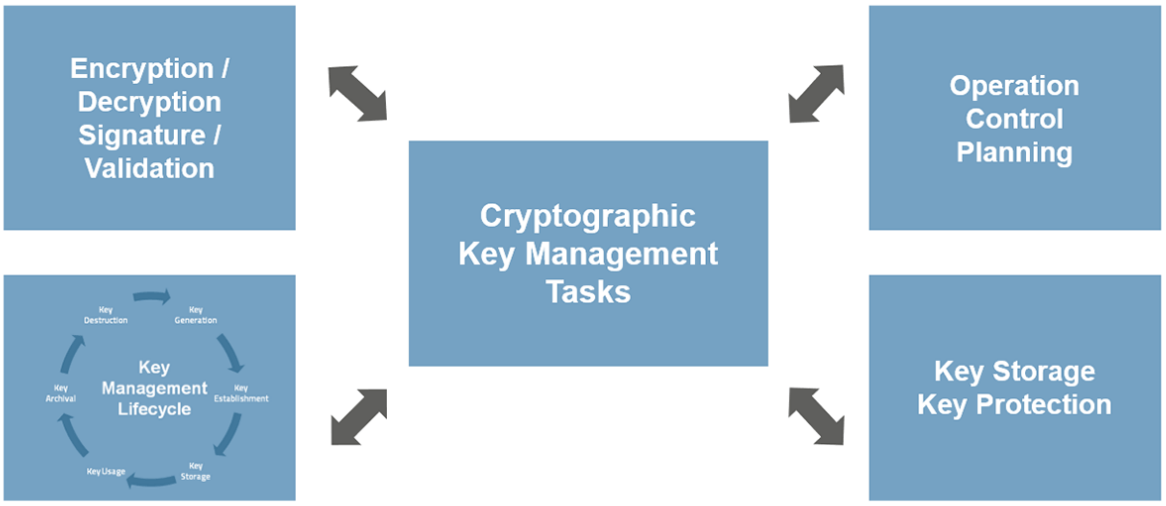


Рис 111

Маніпулювання даними в симетричних системах відбувається швидше, ніж в асиметричних системах, оскільки вони зазвичай використовують меншу довжину ключа. Асиметричні системи використовують відкритий ключ для шифрування повідомлення та закритий ключ для його дешифрування. Однак використання асиметричних систем підвищує безпеку зв'язку; він сильно споживає ресурси ЦП.

2.2.3 Керування ключовою системою

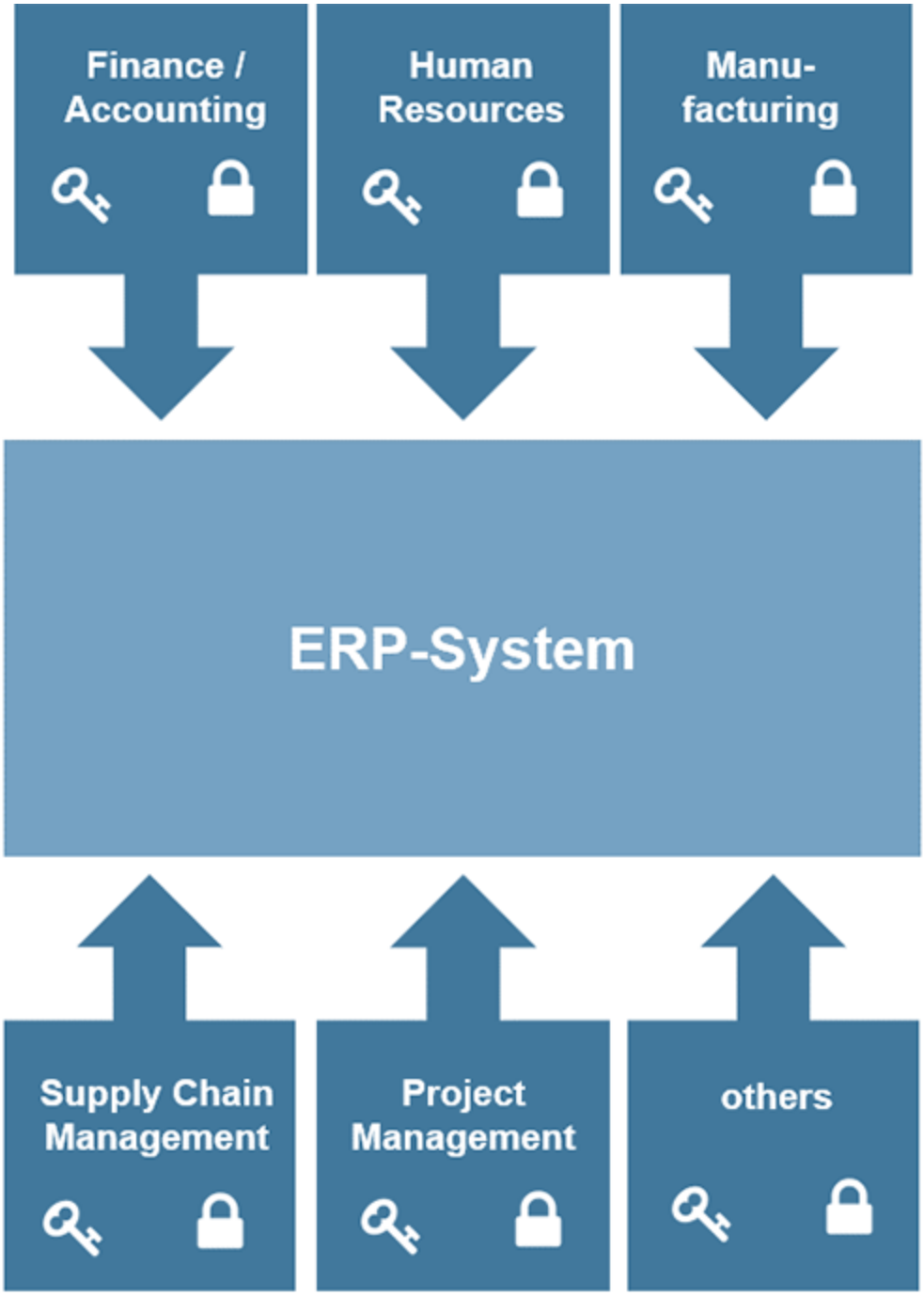
Ефективна система керування криптографічними ключами (KMS) повинна мати можливість централізовано підтримувати всі IT-процеси компанії, пов'язані з безпекою. В Інтернеті речей керування великою кількістю індивідуальних криптографічних ключів у виробництві та на сайті клієнта стане головним завданням.



Завдання системи керування криптографічним ключем

Відповідна система керування криптографічними ключами (система керування ключами шифрування або шифрування) має виходити далеко за межі звичайного менеджера паролів. Усі процеси, пов'язані з IT-безпекою в компанії, потребують про активної підтримки безпеки за допомогою централізованого підходу.

Основними елементами процесів безпеки є ключі та сертифікати. Протягом усього життєвого циклу керування ключами система керування ключами шифрування має зберігати та керувати великою кількістю «секретів» (ключі SSH, ключі API, сертифікати тощо). Важливо, щоб організації мали огляд у будь-який момент часу, як ключі та сертифікати використовуються в їхній мережі. Багато компаній використовують велику кількість ключів і сертифікатів без централізованого контролю. Наприклад, невідомо, хто має доступ до яких ключів, або немає спеціального керування ролями та правами.



Кожен відділ керує своїми власними ключами та сертифікатами.

Якщо ключі та сертифікати не захищені належним чином, компанія стає вразливою до атак. Важливо, щоб організації розуміли, які ключі та сертифікати використовуються в мережі. Їм потрібно знати, хто має до них доступ, і контролювати, як і коли вони використовуються. Також важливо мати огляд крипто-алгоритмів, які використовуються в організації, щоб мати можливість ідентифікувати поточні небезпечні алгоритми.

2.2.4 Проблеми поширення сертифікатів відкритих ключів

У напрямку криптографії з відкритим ключем необхідно знати відкритий ключ кожного користувача в мережі, щоб легко здійснювати шифрування та дешифрування. У цьому контексті існує декілька схем, запропонованих у «Криптографії та мережевій безпеці Вільяма Столлінгса», як-от публічне оголошення відкритого ключа, загальнодоступний каталог, авторитет відкритого ключа та центр сертифікації. Для генерації секретного ключа також доступне використання відкритого ключа. У цьому документі ми реалізуємо концепцію центру сертифікації з відкритим ключем без центру сертифікації (PKAwCA) і центру сертифікації (CA). Основна мета — представити порівняльну діаграму між обома схемами, що розглядаються щодо безпеки. Є дві звичайні схеми для надання відкритого ключа в мережі.

2.2.5 Створення пакету документів

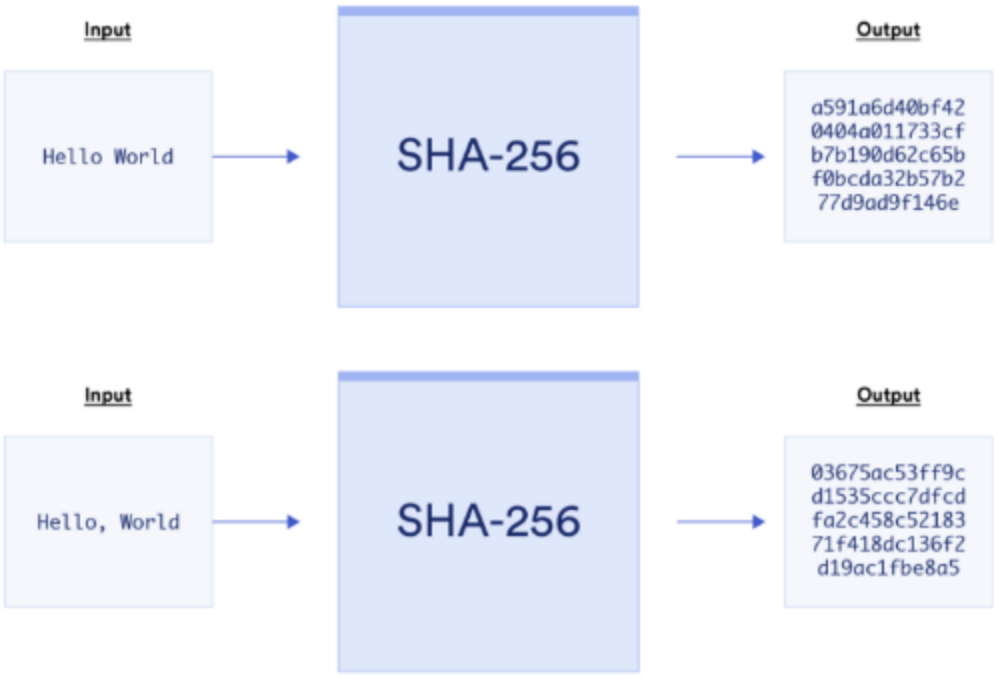
Організація системи електронного документообігу не зводиться до установки програмного забезпечення. Значно більш складним і трудомістким процесом (принаймні, на початковому етапі) є підготовка документів, що докладно описує всі процедури функціонування системи, а також навчання співробітників, що будуть забезпечувати її роботу. Спрощує ситуацію те, що зразки подібних документів вже існують і можна замовити розробку всього пакета компанії, що має досвід успішного застосування ЕЦП. Ідеально, якщо ці документи пройшли «перевірку боєм», тобто на їхній основі розглядався конфлікт у суді. Адміністрацію системи можна організувати на базі сторонньої фірми, що розташовує відповідними службами, кваліфікованими співробітниками, необхідними комплектами договорів, визначеним досвідом обслуговування таких систем. Ризик розкриття конфіденційної інформації при цьому відсутній, оскільки секретними ключами учасників адміністрація не володіє - вона оперує тільки довідниками відкритих ключів. Важливо, щоб генерація ключів (включаючи секретні) проводилася уповноваженими співробітниками учасників (нехай і на території ліцензованої адміністрації.

2.2.6 Криптографічні хеш-функції

Хеш-функції перетворюють дані будь-якої довжини на фіксовану довжину біта, яка називається хешем. Хеш – це унікальний ідентифікатор даних, схожий на відбиток пальця, по суті, перевіряючи початковий набір даних. Хешування даних широко використовується для індексування даних та ефективного отримання їх з бази даних. Він також використовується для безпечного зберігання даних, наприклад, у випадку веб-сайтів, які зберігають лише солоний хеш паролів користувачів, щоб запобігти витоку паролів у разі порушення бази даних.

Хеш-функції не використовують ключі і засновані на односторонніх функціях, що гарантує, що вони стійкі до попередніх зображень – для того, хто не знає оригінального повідомлення, практично неможливо обчислити його тільки з хешу. Єдиним реальним методом отримання вихідного повідомлення є груба сила, тобто вгадування всіх можливих входів, які в разі хешів є необмеженим набором можливих комбінацій символів. Алгоритми хешування також, як правило, розроблені, щоб бути стійкими до змови – це майже неможливо для двох різних входів, щоб генерувати один і той же хеш. Ці властивості роблять так, що зміна чогось у вході, наприклад, велика літера або додавання розділового знака, призведе до зовсім іншого і, здавалося б, випадкового хешу.

Найпопулярнішим алгоритмом безпечного хешування на сьогоднішній день є SHA-256, який створює рядки довжиною 256 біт (тобто 256 послідовних 1s і 0s). 256-бітні хеші часто представлені в шістнадцятковому форматі рядків, тому вони можуть бути довжиною 32 або 64 символи. SHA-256 існує як частина набору хеш-функцій SHA-2, який замінив тепер криптографічно несправний SHA-1. SHA-3 також був представлений в 2015 році на основі криптографічного примітиву Keccak.



Зміна одного символу в хеші, наприклад, додавання коми в другому вході, повністю змінить результат.

Криптографічні хеш-функції є третім типом криптографічного алгоритму. Повідомлення будь-якої довжини приймається як вхідні дані та виводиться у короткий хеш фіксованої довжини. (MD5, SHA тощо) Це математичний алгоритм, який відображає дані довільного розміру в бітовий рядок фіксованого розміру (хеш) і розроблений як одностороння функція, яку неможливо інвертувати. Перевірка цілісності — це механізм перевірки того, чи інформація не змінилася. Для підтвердження цілісності створюється відбиток (також званий хеш або дайджест) інформації. Відбиток, створений за допомогою алгоритму, який створює коротший бітовий рядок з інформації.

Рис 111

Випадковий рядок, або сіль, додається до пароля (щоб зробити пароль більш безпечним), а потім хешується. Це запобігає атакам райдужних таблиць. Засіб слід використовувати з криптографічно безпечним генератором псевдовипадкових чисел, також відомим як CSPRNG. Солі повинні мати високу ентропію. Однак RSA використовував стандарт Dual_EC_DRBG для CSPRNG, який, як було показано, не є криптографічно безпечним і, як вважають, має клептографічний бекдор NSA. Backdoor був підтверджений у 2013 році, і RSA Security отримала 10 мільйонів доларів від АНБ для цього.

Криптографічну хеш-функцію або блоковий шифр можна багаторазово застосовувати в циклі. Сучасні функції виведення ключів на основі пароля, такі як PBKDF2, використовують криптографічний хеш, такий як SHA-2, довший соль (наприклад, 64 біти) і велику кількість ітерацій (десятки або сотні тисяч).

2.2.7 Автентифіковане шифрування

Автентифіковане шифрування забезпечує конфіденційність, цілісність даних і гарантії автентичності зашифрованих даних . Автентифіковане шифрування може бути створено узагальнено шляхом поєднання схеми шифрування такоду автентифікації повідомлення (MAC). Наприклад, AWS KMS Encrypt API приймає відкритий текст, ідентифікатор головного ключа клієнта (CMK) і контекст шифрування (контекстшифрування— це набір несекретних пар ключ-значення, які ви можете передати в AWS KMS, коли викликаєтеEncrypt,Decrypt,ReEncrypt,GenerateDataKey, абоGenerateDataKeyWithoutPlaintextAPI.) і повертає зашифрований текст. Контекст шифрування представляє додаткові автентифіковані дані (AAD). Процес шифрування використовує AAD лише для створення тегу автентифікації. Тег додається до вихідного зашифрованого тексту та використовується як вхідні дані для процесу дешифрування. Це означає, що контекст шифрування, який ви надаєте Decrypt API, має збігатися з контекстом шифрування, який ви надаєте Encrypt API. Інакше теги шифрування та дешифрування не збігатимуться, і процес дешифрування не створить відкритий текст. Крім того, якщо будь-який із параметрів було підроблено — зокрема, якщо зашифрований текст було змінено

— тег автентифікації не обчислюватиме те саме значення, що й під час шифрування. Процес дешифрування не вдасться, і зашифрований текст не буде розшифровано.

2.2.8 Цифровий підпис

Цифровий підпис — це математична схема для демонстрації автентичності цифрових повідомлень або документів. Дійсний цифровий підпис забезпечує цілісність інформації (за допомогою хеш-алгоритму), щоб переконатися, що повідомлення не змінено, повідомлення створено відправником (автентифікація), і відправник не може заперечити, що надіслав повідомлення (невідмовність). Цифровий підпис має бути автентичним, таким, що не підлягає підробці, багаторазовому використанню, незмінним і безвідкличним. Коли все це майно зібрано, можна перевірити справжність і цілісність інформації.

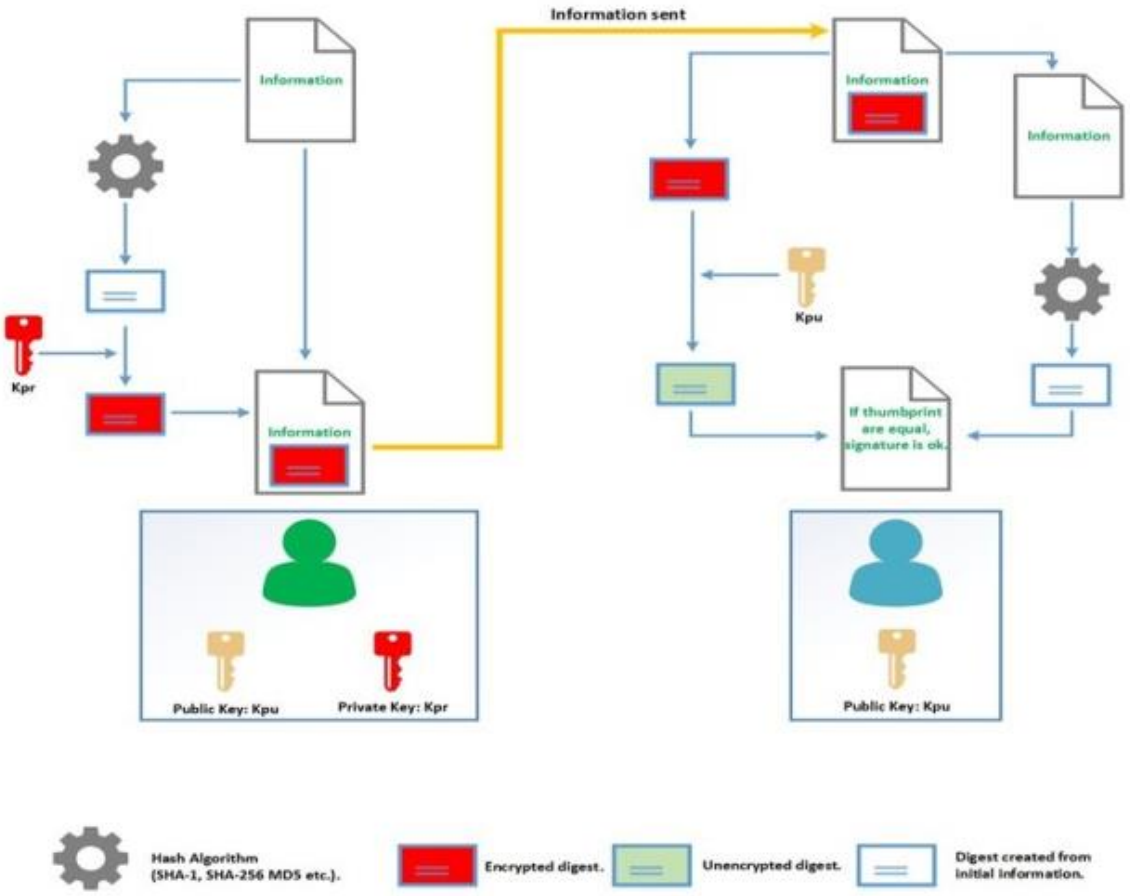


Рис 111

Операція підпису заснована на асиметричній криптографії. Спочатку створюється дайджест початкової інформації, а останній шифрується закритим

ключем. Ця операція називається підписом.

Щоб перевірити підпис, одержувач витягує зашифрований дайджест із повідомлення та використовує свій відкритий ключ, щоб розшифрувати його. Далі одержувач створює дайджест із отриманої інформації та порівнює його з попередньо незашифрованим дайджестом. Це процес перевірки підпису.

Хороший спосіб запам'ятати, коли використовується закритий ключ, - це знати, яка інформація важлива в кожній операції. У процесі підписання критична інформація є дайджестом, тому для підпису використовується закритий ключ. У процесі шифрування важлива інформація шифрується, тому для розшифровки використовується закритий ключ.

Кожен алгоритм шифрування має переваги та зручності, тому Modern Encryption поєднує як симетричні, так і асиметричні методи. Сучасний алгоритм використовує сеансовий ключ (тимчасовий ключ) для шифрування інформації за допомогою симетричної криптографії. Далі сеансовий ключ шифрується відкритим ключем одержувача. Щоб розшифрувати інформацію, одержувач спочатку розшифровує сеансовий ключ своїм закритим ключем і розшифровує інформацію сеансовим ключем.

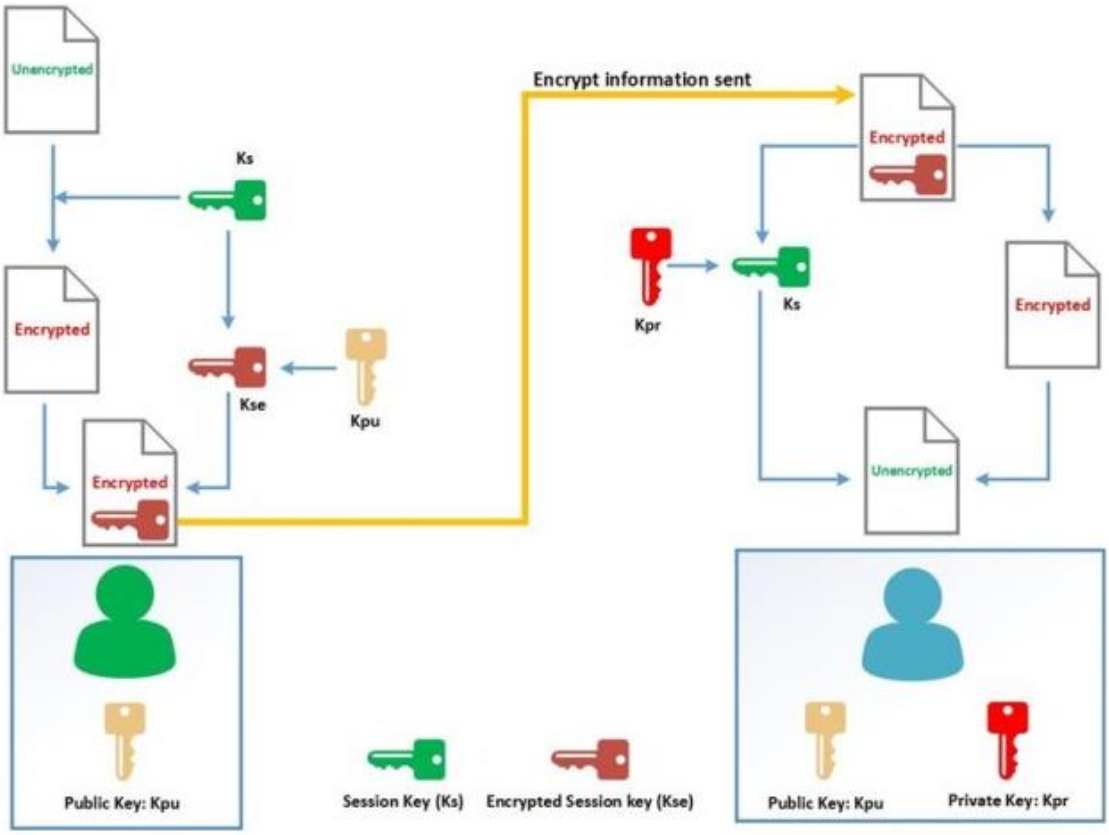


Рис 111

На стороні відправника виконано такі дії:

1. Створений тимчасовий ключ, який називається сеансовим ключем (Ks);

2. Інформація, зашифрована сеансовим ключем (Ks);

3. (Ks) Зашифровано відкритим ключем (Kpu) одержувача. Цей ключ називається Kse;

4. Kse додано до зашифрованого інформаційного файлу. Цей файл надіслано одержувачу.

На стороні одержувача виконано наведену нижче дію.

Зашифрована інформація та Kse розділені;

Ключ Kse розшифровується за допомогою закритого ключа (Kpr) одержувача та стає Ks;

Документ не зашифрований Kc.

Операція шифрування та цифрового підпису

Тепер, коли ми знаємо про шифрування, хеш-алгоритм і підпис, давайте подивимось, як ці елементи взаємодіють разом, щоб зробити інформацію

конфіденційною, автентичною та чесною.

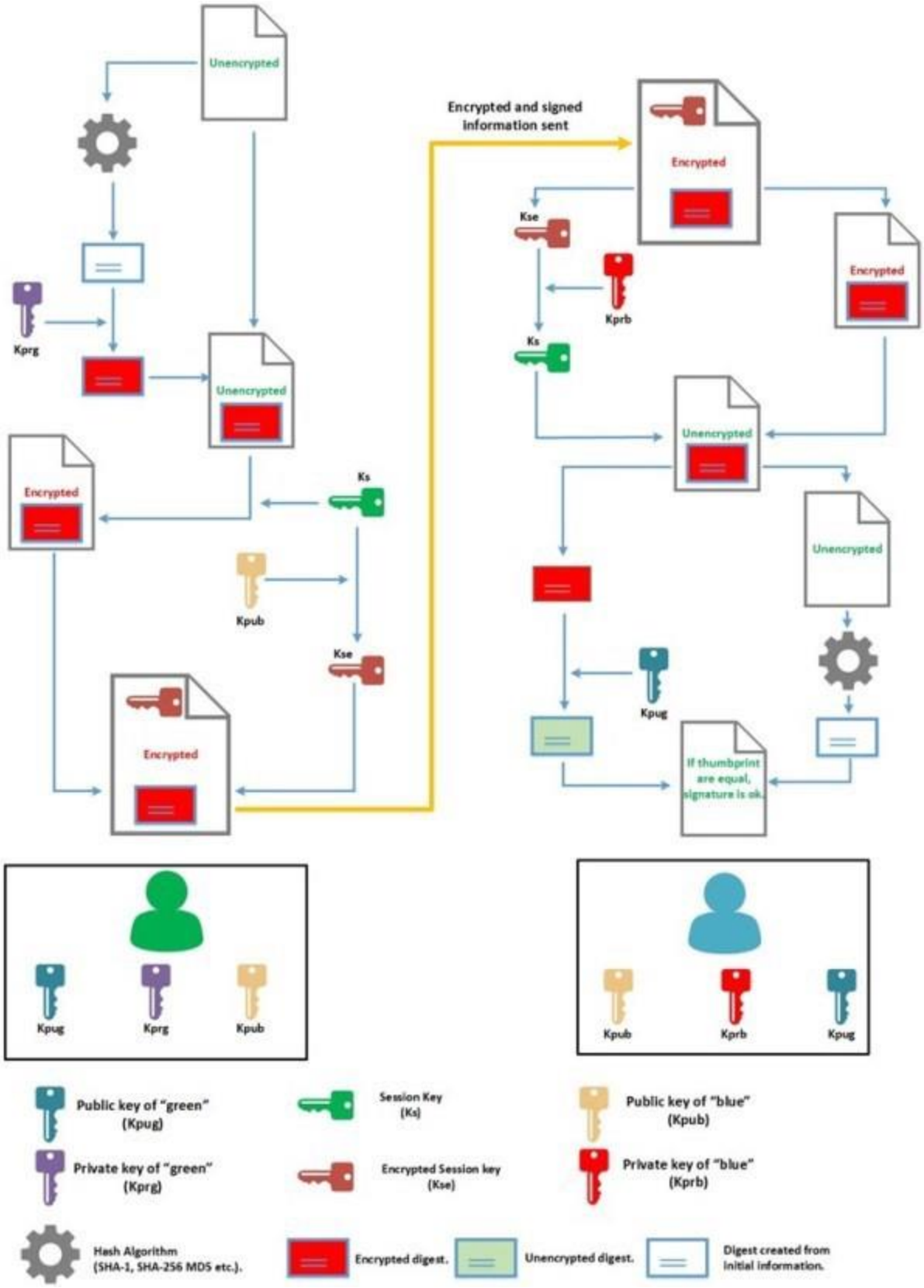


Рис 111

Коли підпис і шифрування використовуються разом, процес підписання виконується першим. Виконано наступні дії:

1. З вихідної інформації створюється дайджест;

2. Цей відбиток зашифровано закритим ключем (Kprg);

3. Відбиток додається до вихідної інформації (у тому ж файлі);

4. Створюється тимчасовий сеансовий ключ (Ks). Він використовуватиметься для шифрування початкової інформації;

5. Сеансовий ключ зашифровано (Kse) відкритим ключем одержувача (Kpub);

6. Kse додано до зашифрованого інформаційного файлу. Отже, цей файл містить зашифровану інформацію, Kse та підпис.

Коли одержувач отримує файл від видавця, він починає розшифровувати файл, а потім перевіряє підпис:

1. Одержувач витягує Kse з отриманого файлу. Цей ключ розшифровується за допомогою закритого ключа (Kprb), щоб отримати ключ сеансу (Ks);

2. Ks використовується для розшифровки інформації;

3. Наступний одержувач витягує зашифрований відбиток;

4. Відкритий ключ (Kpug) використовується для розшифровки відбитка;

5. Одночасно одержувач створює дайджест із раніше незашифрованої інформації;

6. На завершення одержувач порівнює незашифрований відбиток із дайджестом, створеним із незашифрованої інформації. Якщо вони збігаються, підпис перевірено.

2.2.9 Алгоритм Рівест-Шаміра-Адлемана (RSA)

RSA розшифровується як Rivest-Shamir-Adleman. Це криптосистема, яка використовується для безпечної передачі даних. В алгоритмі RSA ключ шифрування є відкритим, але ключ дешифрування є закритим. Цей алгоритм заснований на математичному факті, що розкласти добуток двох великих простих чисел нелегко. Його розробили Рон Рівест, Аді Шамір і Леонард Адлеман у 1977 році.

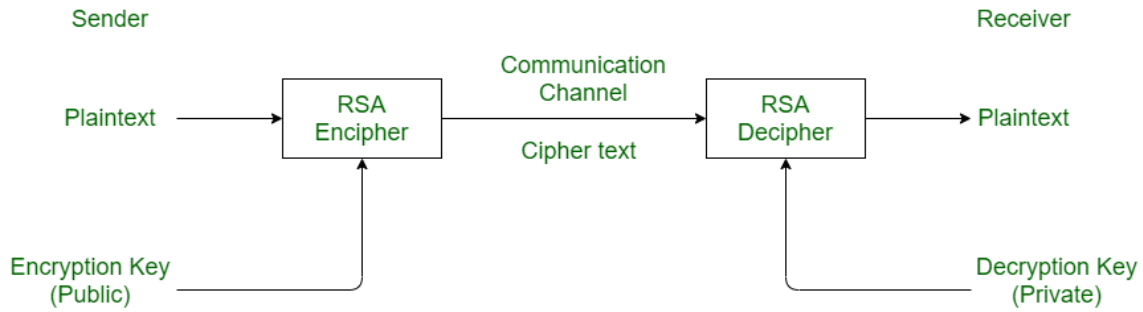


Рис 111

RSA це алгоритм криптосистеми. Він використовується для безпечної передачі даних. Він був розроблений в 1977 році.

Його розробили Рон Рівест, Аді Шамір і Леонард Адлеман. Він використовує математичну концепцію факторизації добутку двох великих простих чисел.

Це повільніше в генерації ключів. Це швидше, ніж DSA у шифруванні. Він повільніше розшифровується. Він найкраще підходить для перевірки та шифрування.

Алгоритм RSA є асиметричним алгоритмом шифрування. Асиметричний насправді означає, що він працює з двома різними ключами, тобто відкритим ключем і закритим ключем. Як видно з назви, відкритий ключ надається кожному, а закритий ключ залишається закритим.

Приклад асиметричної криптографії:

- Клієнт (наприклад, браузер) надсилає свій відкритий ключ на сервер і запитує деякі дані.
- Сервер шифрує дані за допомогою відкритого ключа клієнта та надсилає зашифровані дані.
- Клієнт отримує ці дані та розшифровує їх.
- Оскільки це асиметрично, ніхто інший, крім браузера, не може розшифрувати дані, навіть якщо третя сторона має відкритий ключ браузера.

RSA базується на тому, що велике ціле число важко розкласти на множники. Відкритий ключ складається з двох чисел, де одне число є множенням двох

великих простих чисел. І закритий ключ також походить від тих самих двох простих чисел. Отже, якщо хтось може розкласти велике число на множники, приватний ключ буде зламано. Тому міцність шифрування повністю залежить від розміру ключа, і якщо ми подвоюємо або потроїмо розмір ключа, міцність шифрування зростає експоненціально. Ключі RSA зазвичай можуть мати довжину 1024 або 2048 біт, але експерти вважають, що 1024-бітні ключі можуть бути зламані найближчим часом. Але поки це здається нездійсненним завданням. механізм, алгоритмому RSA: >> Генерація відкритого ключа:

```
Select two prime no's. Suppose P = 53 and Q = 59.
Now First part of the Public key  : n = P*Q = 3127.
 We also need a small exponent say e :
But e Must be
An integer.
Not be a factor of n.
1 < e < Φ(n) [Φ(n) is discussed below],
Let us now consider it to be equal to 3.
    Our Public Key is made of n and e
```

Рис 111

```
We need to calculate Φ(n) :
Such that Φ(n) = (P-1)(Q-1)
      so,  Φ(n) = 3016
    Now calculate Private Key, d :
d = (k*Φ(n) + 1) / e for some integer k
For k = 2, value of d is 2011.
```

>> Генерація закритого ключа:

Тепер ми готові з нашим відкритим ключем (n = 3127 і e = 3) і закритим ключем (d = 2011). Тепер ми зашифруємо «HI»:

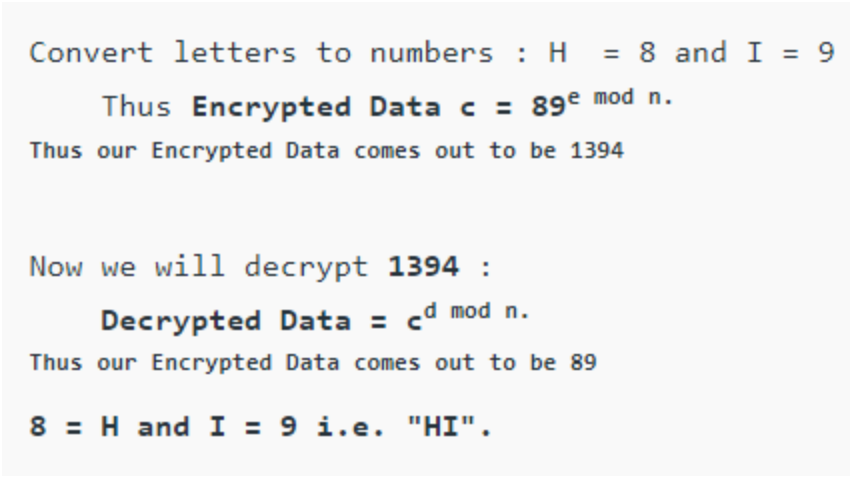


Рис 111

Нижче наведено реалізацію алгоритму RSA для

Шифрування та дешифрування малих цифрових значень:

```
/* пакет будь-який //не пишіть тут назву пакета */
import java.io.*;
import java.math.*;
import java.util.*;
/*
* Програма Java для асиметричного криптографічного алгоритму RSA.
* Для демонстрації значення є
* порівняно невеликий порівняно з практичним застосуванням*/
public class GFG {
public static double gcd(double a, double h)
{
        /*
                * Ця функція повертає gcd або найбільше загальне
                * дільник
                */
        double temp;
        while (true) {
        temp = a % h;
        if (temp == 0)
```

```
            return h;
        a = h;
        h = temp;
        }
}
public static void main(String[] args)
{
        double p = 3;
        double q = 7;
        // Stores the first part of public key:
        double n = p * q;
        // Finding the other part of public key.
        // double e stands for encrypt
        double e = 2;
        double phi = (p - 1) * (q - 1);
        while (e < phi) {
        /*
                    * e має бути співпростим з phi і
                    * менше ніж phi.
                    */
        if (gcd(e, phi) == 1)
            break;
        else
            e++;
        }
        int k = 2; // A constant value
        double d = (1 + (k * phi)) / e;
        // Message to be encrypted
        double msg = 12;
        System.out.println("Message data = " + msg);
```

```
        // Encryption c = (msg ^ e) % n
        double c = Math.pow(msg, e);
        c = c % n;
        System.out.println("Encrypted data = " + c);
        // Decryption m = (c ^ d) % n
        double m = Math.pow(c, d);
        m = m % n;
        System.out.println("Original Message Sent = " + m);
}
}
```

Вихід

Initial message:

Test Message

The encoded message(encrypted by public key)

8633128871359515934139274349128871359513595830518790128887

The decoded message(decrypted by private key)

Test Message

2.2.10 Алгоритм цифрового підпису (DSA)

DSA означає алгоритм цифрового підпису. Використовується для цифрового підпису та його перевірки. Він заснований на математичній концепції модульного піднесення до степеня та дискретного логарифма. Він був розроблений Національним інститутом стандартів і технологій (NIST) у 1991 році.

Він включає чотири операції:

- Генерація ключів
- Розподіл ключів
- Підписання

- Перевірка підпису

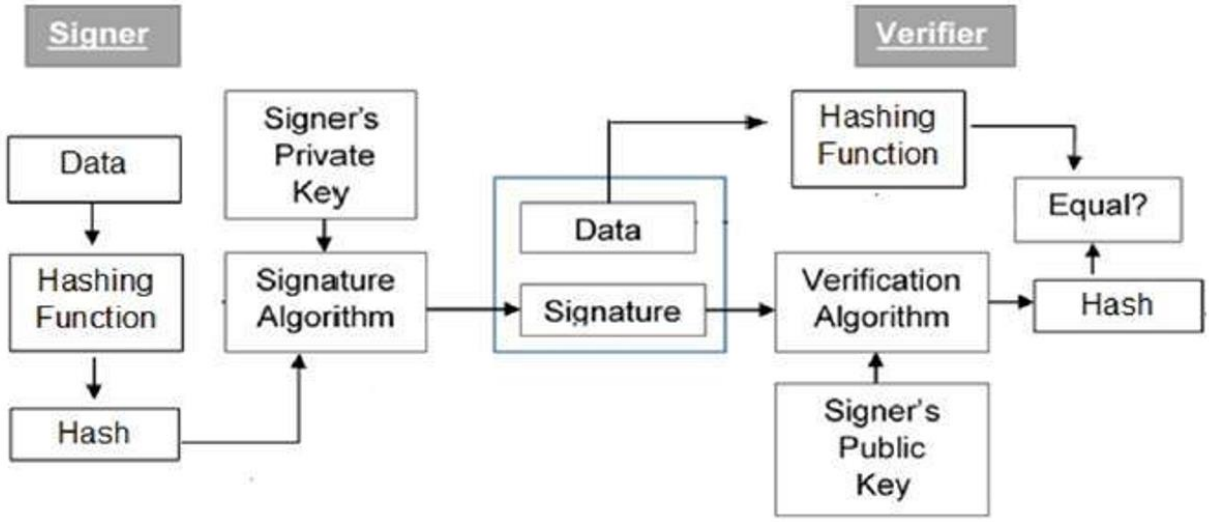


Рис 111

DSA це алгоритм цифрового підпису. Використовується для цифрового підпису та його перевірки. При цьому він був розроблений в 1991 році. Він був розроблений Національним інститутом стандартів і технологій (NIST). Він використовує модульне піднесення до степеня та дискретний логарифм. Хоча він швидше генерує ключі порівняно з RSA. Хоча він повільніший у шифруванні. При цьому він швидше розшифровується. Він найкраще підходить для входу та розшифровки.

2.2.11 Алгоритм секретного обміну Шаміра

Криптографія - це техніка захисту інформації та зв'язку за допомогою кодів, щоб тільки та особа, для якої призначена інформація, могла її зрозуміти та обробити. Таким чином запобігає несанкціонований доступ до інформації. Префікс «crypt» означає «прихований», а суфікс graphy означає «написання». У цій статті обговорюється тип криптографічного методу, алгоритм обміну секретами Шаміра.

Алгоритм Shamir’s Secret Sharing: Shamir’s Secret Sharing – це алгоритм у криптографії, створений Аді Шаміром. Основна мета цього алгоритму - розділити секрет, який потрібно зашифрувати, на різні унікальні частини.

- Скажімо, S — секрет, який ми хочемо закодувати.
- Він розділений на N частин: S1, S2, S3, …., Sn.
- Після його поділу користувач вибирає число K, щоб розшифрувати частини та знайти оригінальний секрет.
- Його вибрано таким чином, що якщо ми знаємо менше ніж K частин, то ми не зможемо знайти секрет S (тобто секрет S не можна реконструювати за допомогою (K – 1) частин або менше.
- Якщо ми знаємо K або більше частин із S1, S2, S3, …., Sn, тоді ми можемо легко обчислити/відновити наш секретний код S. Це умовно називається (K, N) пороговою схемою.

Основна ідея алгоритму обміну секретами Шаміра полягає в тому, що для заданих K точок ми можемо знайти поліноміальне рівняння зі ступенем (K – 1). Для заданих двох точок (x1, y1) і (x2, y2) можна знайти лінійний поліном ax + by = c. Подібним чином для заданих трьох точок ми можемо знайти квадратний многочлен $ax^2 + bx + cy = d$.

Отже, ідея полягає в тому, щоб побудувати поліном зі ступенем (K – 1), щоб постійний член був секретним кодом, а решта чисел були випадковими, і цей постійний член можна знайти, використовуючи будь-які K точок із N точок, згенерованих з цей поліном за допомогою базисного полінома Лагранжа.

Наприклад: Нехай секретний код S = 65, N = 4, K = 2.

1. Спочатку, щоб зашифрувати секретний код, будуємо поліном ступеня (K – 1).

2. Отже, нехай многочлен y = a + bx. Тут постійна частина «a» є нашим секретним кодом.

3. Нехай b будь-яке випадкове число, скажімо, b = 15.

4. Тому для цього многочлена y = 65 + 15x ми генеруємо з нього N = 4 точки.

5. Нехай ці 4 точки будуть (1, 80), (2, 95), (3, 110), (4, 125). Зрозуміло, що ми можемо створити початковий поліном з будь-яких двох із цих 4 точок, і в отриманому поліномі постійний член a є необхідним секретним кодом.

Щоб відновити заданий поліном назад, використовується поліном базису Лагранжа.

Основна концепція полінома Лагранжа полягає в тому, щоб спочатку сформувати тотожності Лагранжа, а підсумовування цих тотожностей дає нам потрібну функцію, яку нам потрібно знайти із заданих точок. Наступні рівняння показують, як їх обчислити:

$$l_i = \frac{x - x_0}{x_i - x_0} \times \ldots \times \frac{x - x_{i-1}}{x_i - x_{i-1}} \times \frac{x - x_{i+1}}{x_i - x_{i+1}} \times \ldots \times \frac{x - x_{k-1}}{x_i - x_{k-1}}$$

$$f(x) = \sum_{i=0}^{K-1} y_i l_i(x)$$

Наприклад, скажімо, ми хочемо обчислити задане рівняння, використовуючи K = 2 точки з вибраних чотирьох точок (1, 80), (3, 110). На наступній ілюстрації показано покрокове рішення для цього:

$$l_0 = \frac{x - x_1}{x_0 - x_1} = \frac{x - 3}{1 - 3}$$

$$l_1 = \frac{x - x_0}{x_1 - x_0} = \frac{x - 1}{3 - 1}$$

$$f(x) = y_0 l_0 + y_1 l_1$$

$$f(x) = 80\left(\frac{x - 3}{-2}\right) + 110\left(\frac{x - 1}{2}\right)$$

$$f(x) = -40x + 120 + 55x - 55$$

$$f(x) = 15x + 65$$

Нижче наведено реалізацію вищезазначеного підходу:

```
// C++ реалізація shamir
// алгоритм обміну секретами

#include <bits/stdc++.h>
using namespace std;

// Функція для обчислення значення
// y = poly[0] + x*poly[1] + x^2*poly[2] + ...
```

```
int calculate_Y(int x, vector<int>& poly)
{
    // Initializing y
    int y = 0;
    int temp = 1;

    // Iterating through the array
    for (auto coeff : poly) {

        // Computing the value of y
        y = (y + (coeff * temp));
        temp = (temp * x);
    }
    return y;
}

// Функція виконання секрету
// алгоритм спільного використання та кодування
// заданий секрет
void secret_sharing(int S, vector<pair<int, int> >& points,
                    int N, int K)
{
    // Вектор для зберігання полінома
    // коефіцієнт K-1 ступеня
    vector<int> poly(K);

// Випадково обираємо K - 1 чисел, але
// не нуль і poly[0] є секретом
// створити поліном для цього
    poly[0] = S;

    for (int j = 1; j < K; ++j) {
        int p = 0;
        while (p == 0)

            // Щоб зберегти випадкові значення
            // в діапазоні не надто високо
            // беремо мод з a
            // просте число близько 1000
            p = (rand() % 997);

        // Це зроблено для того, щоб ми цього не зробили
        // створити поліном, що складається
        // з нулів.
        poly[j] = p;
```

```
    }

    // Створення N точок із
    // поліном, який ми створили
    for (int j = 1; j <= N; ++j) {
        int x = j;
            int y = calculate_Y(x, poly);

        // Бали, створені під час обміну
        points[j - 1] = { x, y };
    }
}

// Ця структура використовується для дробу
// частина обробки множення
// і додавання дробу
struct fraction {
    int num, den;

    // Дріб складається з а
    // чисельник і знаменник
    fraction(int n, int d)
    {
        num = n, den = d;
    }

    // Якщо дробу немає
    // у скороченому вигляді
    // зменшити діленням
    // їх з їх GCD
    void reduce_fraction(fraction& f)
    {
        int gcd = __gcd(f.num, f.den);
        f.num /= gcd, f.den /= gcd;
    }

    // Виконання множення на
    // частка
    fraction operator*(fraction f)
    {
        fraction temp(num * f.num, den * f.den);
        reduce_fraction(temp);
        return temp;
    }
```

```
    // Виконання додавання на
    // частка
    fraction operator+(fraction f)
    {
        fraction temp(num * f.den + den * f.num,
                      den * f.den);

        reduce_fraction(temp);
        return temp;
    }
};

// Функція генерації секрету
// назад від заданих точок
// Ця функція використовуватиме базисний поліном Лагранжа
// Замість пошуку повного многочлена
// Нам потрібен лише poly[0] як секретний код,
// таким чином ми можемо позбутися x термінів
int Generate_Secret(int x[], int y[], int M)
{

    fraction ans(0, 1);

    // Цикл для повторення заданого
    // бали
    for (int i = 0; i < M; ++i) {

        // Initializing the fraction
        fraction l(y[i], 1);
        for (int j = 0; j < M; ++j) {

            // Computing the lagrange terms
            if (i != j) {
                fraction temp(-x[j], x[i] - x[j]);
                l = l * temp;
            }
        }
        ans = ans + l;
    }

    // Повернути секрет
    return ans.num;
}

// Функція для кодування та декодування
```

```
// надається секрет за допомогою вищезазначеного
// визначені функції
void operation(int S, int N, int K)
{

    // Вектор для зберігання точок
    vector<pair<int, int> > points(N);

    // Sharing of secret Code in N parts
    secret_sharing(S, points, N, K);

    cout << "Secret is divided to " << N
         << " Parts - " << endl;

    for (int i = 0; i < N; ++i) {
        cout << points[i].first << " "
             << points[i].second << endl;
    }

    cout << "We can generate Secret from any of "
         << K << " Parts" << endl;

    // Введіть будь-які M точок із них
    // щоб повернути наш секретний код.
    int M = 2;

    // M може бути більше або дорівнювати порогу, але
    // для цього прикладу ми беремо поріг
    if (M < K) {
        cout << "Points are less than threshold "
             << K << " Points Required" << endl;
    }

    int* x = new int[M];
    int* y = new int[M];

   // Введіть M балів, ви отримаєте секрет
     // Нехай ці точки є першими M точками від
     // N точок, якими ми поділилися вище
     // We can take any M points

    for (int i = 0; i < M; ++i) {
        x[i] = points[i].first;
        y[i] = points[i].second;
    }
```

```
    // Знову повертаємо наш результат.
    cout << "Our Secret Code is : "
         << Generate_Secret(x, y, M) << endl;
}

// Код драйвера
int main()
{
    int S = 65;
    int N = 4;
    int K = 2;

    operation(S, N, K);
    return 0;
}
```

Вихід:

```
Secret is divided to 4 Parts -
1 602
2 1139
3 1676
4 2213
We can generate Secret from any of 2 Parts
Our Secret Code is : 65
```

2.3 Аналіз електронних цифрових підписів в інфраструктурі корпоративної інформаційної системи.

Питання безпеки Системи управління документами, як правило, забезпечують належну безпеку та контроль доступу до документів і можливість змінювати документи. Однак, коли документ залишає цей документ у безпеці навколишнє середовище вона піддається потенційним змінам. З будь-яким цифровим документом його легко змінити вміст або копіювання зображень і створення реалістичних, але підроблених і шахрайських документів. Незахищені документи у форматі PDF, XML, Office та електронні документи не надають одержувачу можливості визначення того, чи є документ справжнім, ким були укладач і затверджувачі, чи є вони уповноважений випускати його або чи був він змінений з моменту його випуску. Просте вбудоване Зображення підпису Squiggle нічого не робить для захисту документа від майбутніх змін.

Загалом люди розуміють поняття цифрових підписів, однак більшість не знає, що таке робить його дійсним або який рівень безпеки необхідний для їх використання. Переважна більшість людей знайомі з електронним підписом на планшетних пристроях для прийняття доставок, однак висока цінність або високий ризик договори всередині компанії або з третіми особами потребують узгодження підписами, які можуть бути підтверджені як автентичні, так і такі, що гарантують, що нічого не можна буде пізніше змінити без порушення підпису.

На щастя, хмарні бізнес-сервіси дозволяють легко створювати цифрові підписи та перевіряти їх у будь-який пізніший час. Цифрові підписи надають вагомі докази підписання та затвердження (неповернення), а також автентифікацію (цілісність) документа, і вони блокують документ проти несанкціоновані зміни. Довгострокові підписи гарантують, що ці докази залишаються дійсними для багатьох років у майбутнє та вбудовує довірений час підписання у вигляді позначки часу в межах документ. Цей документ має такий довгостроковий підпис на останній сторінці, який можна перевірити продуктом Adobe Reader. Захищені журнали аудиту та інші функції підзвітності також

надайте вагомі докази дій користувача щодо підписання.

У США Управління з санітарного нагляду за якістю харчових продуктів і медикаментів також вимагає цифрових підписів під високою довірою документи, включаючи дозволи на ліки та рецепти контрольованих речовин, чому? Бо вони є єдиною формою підпису, яка захищає дані. Цифровий підпис у Європі директива визначає як кваліфіковані, так і розширені цифрові підписи як засоби виробництва юридичних вагові докази, які є електронним еквівалентом чорнильного підпису. Кваліфіковані підписи забезпечити золотий стандарт довіри, оскільки кінцеві користувачі повинні пройти суворий Відомий Ваш Процес реєстрації клієнтів. Розширені підписи також прийнятні для багатьох підприємств заявок, але, як правило, мають менш формальний процес реєстрації.

В Європі Директива ЄС з ПДВ спрямована на запобігання шахрайству з ПДВ шляхом зменшення ризиків змінені або фіктивні документи. Директива вимагає, щоб: "Рахунки-фактури, відправлені електронними засобами приймається державами-членами за умови, що справжність походження та цілісності вміст гарантований». Єдиний стандартний і сумісний спосіб зробити це - використовувати цифрових підписів і ЄС рекомендує використовувати кваліфікований електронний підпис для підтвердження особистість укладача.

Директива також вимагає, щоб «Достовірність походження та цілісності змісту рахунки-фактури, а також їх читабельність повинні бути гарантовані протягом усього терміну зберігання». Цей це, мабуть, найважливіший аспект, оскільки документи повинні зберігатися протягом багатьох років і бути здатний показати, що вони незмінні та оригінальні, це досить важке прохання для будь-якого бізнесу - якщо тільки Звичайно, ви захищаєте свої рахунки-фактури за допомогою довгострокових цифрових підписів. Ascertia надала широкому спектру організацій рішення для підписання рахунків-фактур, рішення для захисту платежів, дані або рішення для підписання документів і затвердження робочого процесу документів.

Найбільш ефективним способом задоволення цих вимог є використання галузевого стандарту цифрового Підписи. Вони пов'язують особу підписувача зі

змістом документа таким чином, щоб чітко показує, що документ автентичний і що документ незмінний. Цифрові підписи надання довірчих послуг, які можуть залишатися ефективними протягом багатьох років, достатніми для всіх фінансових документи (існують спеціальні варіанти доказування понад 20+ років). Передова електронна підпис забезпечує автентичність та цілісність, і, як правило, довірений час затвердження також прив'язаний до документа. Поєднайте ці підписи з документами формату PDF/A (ISO 19005) і тепер ви можете гарантувати, що документ може відображатися правильно без вмісту або зміни в макеті, і що підпис(и) все ще дійсні протягом багатьох років у майбутньому.

Видимі цифрові підписи забезпечують ефективний спосіб зіставлення паперових підписів мокрими чорнилами з електронний світ. Невидимі підписи за своєю природою менш очевидні, і все ж вони перешкоджають будь-яким несанкціоновані зміни. Захист, який пропонують криптографічні процеси, надзвичайно надійний і використовує численні міжнародні стандарти для широкої оперативної сумісності. Без цього Форма захисту важливих бізнес-даних відкрита для зловживань, маніпуляцій, шахрайства, відмови та Крадіжки. Документ із цифровим підписом дає змогу одержувачу:

- Підтвердити, хто підписав документ, коли вони його підписали, та їх поточний статус довіри (шляхом перевірки та перевірки цифрового сертифіката підписувачів)
- Для підтвердження того, що документ не був змінений ні зараз, ні пізніше, будь то тижні, місяці або навіть років (шляхом перевірки цифрового підпису) Щоб підтвердити, що укладач мав намір затвердити його - це не проект непідписаного документа і що автор не може відмовити у його затвердженні (шляхом перевірки підпису)
- для економії часу та значних щомісячних витрат на друк рахунків, паперу, поштових витрат та архівування/зберігання (увімкнувши безпаперові робочі процеси)
- для задоволення потреб різних законодавчих та регуляторних

режимів

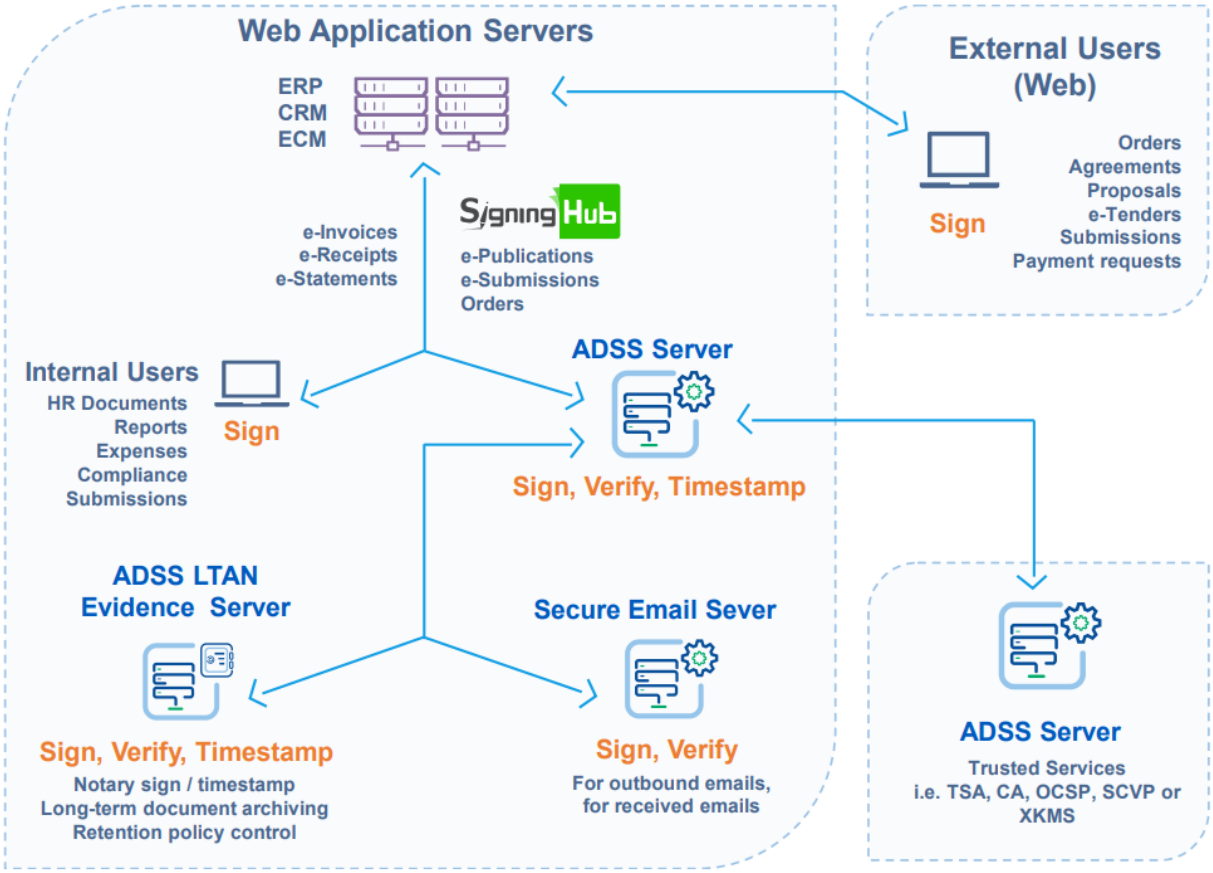


Рис 111

Довіра до цифрових підписів документів виводиться за допомогою ряду елементів:

- надійність емітента сертифіката та якість його політики та процедури при роботі з реєстрацією користувача та підтвердженням особи (тобто кваліфікованими Підписи сертифікатів у порівнянні з розширеними цифровими підписами)
- Контроль управління безпекою, який регулює процес створення цифрового підпису і спосіб, яким доступ до цього може бути авторизований, а також журнали подій та транзакцій перевірені
- Спосіб, у який цифровий підпис може бути перевірений та перевірений незалежними третіми Програмне забезпечення для партій (наприклад.PDF читацькі продукти), а також постачальники послуг

- Використання довірених третіх сторін для надання послуг з позначки часу, підписання або перевірки самостійно заявляти про довіру (часто пропонуючи відповідальність за ці послуги)

Пропоновані довірчі послуги підходять для всіх важливих ділових документів, включаючи:

- Ті, що розміщені в системах управління документами
- Усі електронні рахунки-фактури та всі інші видані та архівні фінансові документи
- подання регуляторних документів на фінансовому, фармацевтичному та інших ринках
- Фінансові інструменти, документи на страховий поліс, кредитні договори
- Документи центральних та місцевих органів влади, медичні записи, бібліотечні системи
- Інженерія, архітектура, планування, науково-дослідні креслення
- Політики, процедури, внутрішній контроль та документи про відповідність
- ERP, CRM, HR та інші системи документообігу

Зниження витрат. При включенні електронного підпису в бізнес зниження витрат чітко проявляється при покупці паперу, ручок, принтерів, тонера, папок і т.д., що передбачає велику економію накладних витрат компанії, що дозволяє вкладати ці кошти в інші ключові бізнес-процеси.

Оптимальний документообіг. Ефективність документообігу значно зростає з впровадженням електронного підпису. З одного боку, скорочується час, необхідний для відправки і роздруківки документів і пошуку в фізичних архівах. З іншого боку, компанії стають більш організованими. Оскільки всі документи

зберігаються в хмарі, доступ до них легше, що дозволяє будь-якому користувачеві шукати та отримувати документацію з будь-якого пристрою в будь-який час.

Більша мобільність. Для доступу до вмісту документа, підписаного електронним підписом, потрібно тільки один пристрій, так як місце і час вільно вибираються. Таким чином, адміністрування електронного документа набагато практичніше, ніж паперового документа. Наприклад, якщо підпис паперовий і є проблеми з втратою документів, процес потрібно проводити з самого початку, змушуючи особу, відповідальну за підпис, приходити особисто, серед іншого.

Підвищена безпека. Одним з найбільш актуальних аспектів електронного підпису є неможливість іншого користувача змінити або відредагувати його. На відміну від паперової, електронний підпис повністю захищена; тому це надійний ресурс у повному обсязі.

Підвищення продуктивності компанії. Оптимальне управління документацією організації призводить до підвищення продуктивності праці всіх її колективів. За рахунок скорочення етапів і етапів виконання і отримання запитів завдяки електронному підписанню прискорюються внутрішні і зовнішні процеси і завдання компанії. Таким чином, кращі результати виходять за більш короткі проміжки часу.

Покращений користувацький досвід. Клієнт завжди хоче отримати найкращий досвід, іншими словами, чим менше часу вам доведеться витратити на будь-яку дію з компанією і чим це легше, тим більше задоволення ви отримаєте. За допомогою електронних підписів компанії, які використовують цю технологію, пропонують краще обслуговування клієнтів і, як наслідок, привертають увагу більшої кількості клієнтів, а також лояльність клієнтів. Для клієнта це означає підвищення зручності і безпеки у всіх угодах.

Утилізація паперу. Електронний підпис є одним з варіантів вирішення проблеми надмірного паперу в компанії. Завдяки цьому методу цифрової сертифікації паперові процеси стають стійкими електронними процесами, допомагаючи навколишньому середовищу з перших рук.

Розділ 3. Розробка рекомендацій щодо вдосконалення методів застосування електронних цифрових підписів на об’єктах інформаційної діяльності.

3.1 Можливості цифрового підпису у форматі XML.

Специфікація підпису XML - "Цифровий підпис XML" (XMLDS) - яка була розроблена спільними зусиллями організації W3C і IETF, має статус Рекомендації W3C. Цей документ [7] визначає правила обробки і синтаксис, призначені для упакування в XML-формат даних про цілісність повідомлення, його автентифікації і користувальницької автентифікації.

Java XML Digital Signature API — це стандартний Java API для створення та перевірки XML-підписів. Цей API було визначено в рамках процесу спільноти Java як JSR 105.

Підписи XML можна застосовувати до даних будь-якого типу, XML або двійкових . Отриманий підпис представлений у форматі XML. XML-підпис можна використовувати для захисту ваших даних і забезпечення цілісності даних, автентифікації повідомлень і автентифікації підписувача.

У процесі обробки цифрового підпису для XML-документів передбачається процедура їх розподілення, у відповідності з методом, що визначається стандартом Canonical XML. У стандарті пропонується метод асоціювання з інформаційним ресурсом деякого ключа, що підписується. При цьому не визначається, яким образом такі ключі асоціюються з особами або організаціями, який зміст мають дані, до яких відноситься цифровий підпис.

API розроблено для підтримки всіх необхідних або рекомендованих функцій Рекомендації W3C щодо синтаксису та обробки XML-підпису. API є розширюваним і підключається та базується на архітектурі постачальника послуг криптографії Java. API призначений для двох типів розробників:

- Розробники, які хочуть використовувати XML Digital Signature API для створення та перевірки XML-підписів;
- Розробники, які хочуть створити конкретну реалізацію API цифрового підпису XML і зареєструвати його як криптографічний сервіс постачальника JCA.

3.2 Приклад підпису XML

Ви можете використовувати XML-підпис для підпису будь-яких довільних даних, будь то XML або двійкові. Дані ідентифікуються за допомогою URI в одному або кількох елементах Reference. XML-підписи описані в одній або кількох із трьох форм: відокремлена, огортаюча або охоплена. Відокремлений підпис над даними, які є зовнішніми або поза самим елементом підпису. Конвертні підписи — це підписи над даними, які знаходяться всередині елемента підпису, а конвертований підпис — це підпис, який міститься всередині даних, які він підписує.

XML-підпис у конверті, згенерований поверх вмісту XML-документа. Кореневий елемент Envelop містить Signature елемент:

```
<Envelope xmlns="urn:envelope">
  <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
    <!-- ... -->
  </Signature>
</Envelope>
```

Рис 111

Цей Signature елемент було вставлено всередину вмісту, який він підписує, що робить його підписом у конверті. Потрібний SignedInfo елемент містить фактично підписану інформацію:

```
<Envelope xmlns="urn:envelope">
  <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
    <SignedInfo>
      <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC
          -xml-c14n-20010315#WithComments"/>
      <SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig
          -more#rsa-sha256"/>
      <Reference URI="">
        <Transforms>
          <Transform Algorithm="http://www.w3.org/2000/09
              /xmldsig#enveloped-signature"/>
        </Transforms>
        <DigestMethod Algorithm="http://www.w3.org/2001/04
            /xmlenc#sha256"/>
        <DigestValue>/juoQ4bDxElf1M+KJauO20euW+QAvvPP0nDCruCQooM
            =</DigestValue>
      </Reference>
    </SignedInfo>
    <!-- ... -->
  </Signature>
</Envelope>
```

Рис 111

Потрібний CanonicalizationMethod елемент визначає алгоритм, який використовується для канонізації SignedInfoелемента перед його підписанням або перевіркою. Канонізація — це процес перетворення вмісту XML у канонічну форму з урахуванням змін, які можуть зробити підпис над цими даними недійсним. Канонізація необхідна через природу XML і спосіб його аналізу різними процесорами та посередниками, які можуть змінити дані таким чином, що підпис більше не є дійсним, але підписані дані залишаються логічно еквівалентними.

Необхідний SignatureMethod елемент визначає алгоритм цифрового підпису, який використовується для створення підпису, у цьому випадку RSA з SHA-256.

Один або кілька Reference елементів ідентифікують дані, які перетравлюються. Кожен Reference елемент ідентифікує дані через URI. У цьому прикладі значенням URI є порожній рядок (""), який вказує на корінь документа. Необов'язковий Transformselement містить список з одного або кількох Transform елементів, кожен з яких описує алгоритм перетворення, що використовується для перетворення даних перед тим, як їх переварити. У цьому прикладі є один елемент Transform для алгоритму конвертованого перетворення. Конвертне перетворення потрібне для конвертованих підписів, щоб сам елемент підпису було видалено перед обчисленням значення підпису. Необхідний DigestMethod елемент визначає алгоритм, який використовується для аналізу даних, у даному випадку SHA-256. Нарешті необхідне DigestValue елемент містить фактичне переварене значення в кодуванні base64.

Потрібний SignatureValue елемент містить значення підпису в кодуванні base64 для підпису над SignedInfo елементом.

Додатковий KeyInfo елемент містить інформацію про ключ, необхідний для перевірки підпису:

```
<KeyInfo>
    <KeyValue>
      <RSAKeyValue>
        <Modulus>
9hSmAKw/4TTw/1l1u1pYzdFm6lOjRB/5NfdGWl/fB8iAa/tiK0f1u
    /VWoK6SMtogYgSDKqQThbAu
9dy9rRnOWRGY2He1JtpOvGh0WCmIFUEs2P22HvEf+JGKVEpkoP4hv53ucT69T+7nKGK3
    /bjxgp+T
C7fbnVj651+jAHuDFlC8Txt1R8ZymfN5cUeHIH96dvNFrtai/uwZDbVMfhV9chL
    //+Vyhx4O5nHv
jfS+0So9Qi52YAbEyLu6+BLdu8wnMWapC88CfXsRwrpx8b6aCU0e6QSZyOvdgXWz3
    +9ifVTBDIxE
kjhL5OASx0qjvc+dPUOMvq7fJE05RRZLyb0YJw==
        </Modulus>
        <Exponent>AQAB</Exponent>
      </RSAKeyValue>
    </KeyValue>
  </KeyInfo>
```

Рис 111

Цей KeyInfo елемент містить KeyValue елемент, який, у свою чергу, містить RSAKeyValue елемент, що складається з відкритого ключа, необхідного для перевірки підпису. KeyInfo може містити різний вміст, наприклад сертифікати X.509 та ідентифікатори ключів PGP. Щоб отримати додаткові відомості про різні типи , перегляньте розділ « KeyInfo Елемент у синтаксисі та обробці підпису XML ».KeyInfo.

3.3 Режим безпечної перевірки підпису XML

Захищений режим перевірки XML-підпису може захистити вас від XML-підписів, які можуть містити потенційно ворожі конструкції, які можуть викликати відмову в обслуговуванні або інші типи проблем безпеки.

Режим безпечної перевірки підпису XML увімкнено за замовчуванням.

За потреби та на власний ризик ви можете вимкнути безпечний режим перевірки XML-підпису, установивши для org.jcp.xml.dsig.secureValidation властивості Boolean.FALSE метод DOMValidateContext.setProperty().

Коли ввімкнено безпечний режим перевірки XML-підпису, XML-підписи обробляються більш безпечно. Обмеження встановлюються для різних конструкцій підпису XML, щоб уникнути таких умов, як атаки на відмову в обслуговуванні. За замовчуванням застосовуються такі обмеження:

- Забороняє використання перетворень XSLT
- Обмежує кількість SignedInfo або Manifest Reference елементів до 30 або менше
- Обмежує кількість Reference трансформацій до 5 або менше
- Забороняє використання підписів MD5 або SHA-1 або алгоритмів MAC MD5
- Забезпечує Reference унікальність ідентифікаторів, щоб запобігти атакам із загортанням підпису
- Забороняє ReferenceURI типу http, https або file
- Не дозволяє RetrievalMethod елементу посилатися на

інший RetrievalMethod елемент

- Забороняє ключі RSA або DSA менше 1024 біт
- Забороняє ключі EC менше 224 біт

Крім того, ви можете використовувати jdk.xml.dsig.secureValidationPolicy властивість безпеки, щоб контролювати та точно налаштовувати обмеження, перелічені раніше, або додавати додаткові обмеження. Щоб отримати додаткові відомості, перегляньте визначення цієї властивості безпеки у java.security файлі.

3.3 Перевірка XML-підпису

У цьому прикладі показано, як перевірити XML-підпис за допомогою Java XML Digital Signature API. У цьому прикладі використовується DOM (the Document Object Model) для аналізу XML-документа, що містить елемент Signature, і реалізації DOM для перевірки підпису.

Створення екземпляра документа, що містить підпис. Спочатку ми використовуємо JAXPDocumentBuilderFactory щоб проаналізувати XML-документ, що містить Signature. Програма отримує стандартну реалізацію для DocumentBuilderFactory за допомогою виклику наступного рядка коду:

```
DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
```

Рис 111

Ми також повинні зробити простір імен фабрики обізнаним:

```
dbf.setNamespaceAware(true);
```

Рис 111

Далі ми використовуємо фабрику, щоб отримати екземпляр DocumentBuilder, який використовується для аналізу документа:

```
Document doc = null;
try (FileInputStream fis = new FileInputStream(args[0])) {
    doc = dbf.newDocumentBuilder().parse(fis);
}
```

Визначення елемента підпису, який потрібно перевірити

Нам потрібно вказати Signature елемент, який ми хочемо перевірити, оскільки в документі може бути більше одного. Ми використовуємо метод DOMDocument.getElementsByTagNameNS, передаючи йому URI простору імен підпису XML і назву тегу Signature елемент, як показано:

```
NodeList nl =
    doc.getElementsByTagNameNS(XMLSignature.XMLNS, "Signature");
if (nl.getLength() == 0) {
    throw new Exception("Cannot find Signature element");
}
```

Рис 111

Це повертає список усіхSignature елементи в документі. У цьому прикладі лише один Signature елемент.

Створення контексту перевірки

Ми створюємоXMLValidateContext екземпляр, що містить вхідні параметри для перевірки підпису. Оскільки ми використовуємо DOM, ми створюємо екземпляр DOMValidateContext екземпляр (підклас XMLValidateContext), і передайте йому два параметри, KeyValueKeySelectori посилання на елемент Signature, який потрібно перевірити (це перший запис у NodeList ми створили раніше):

```
DOMValidateContext valContext = new DOMValidateContext
    (new KeyValueKeySelector(), nl.item(0));
```

Рис 111 TheKeyValueKeySelector

Демаршалінг XML-підпису

Витягуємо вміст Signature елемент в XMLSignature об'єкт. Цей процес називається демаршалінгом. TheSignature елемент демаршалюється за допомогою XMLSignatureFactory об'єкт. Програма може отримати реалізацію DOMXMLSignatureFactoryза допомогою виклику наступного рядка коду:

```
XMLSignatureFactory fac = XMLSignatureFactory.getInstance("DOM");
```

Рис 111

Потім ми викликаємо unmarshalXMLSignature метод заводу демаршалувати XMLSignaturei передати йому контекст перевірки, який ми створили раніше:

```
XMLSignature signature = fac.unmarshalXMLSignature(valContext);
```

Рис 111

Тепер ми готові перевірити підпис. Ми робимо це, викликаючи метод validate XMLSignaturei передайте йому контекст перевірки таким чином:

```
boolean coreValidity = signature.validate(valContext);
```

Рис 111

Метод validate повертає «true», якщо підпис успішно перевірено відповідно до основних правил перевірки в рекомендаціях W3C XML Signature Recommendation, і false в іншому випадку.

Що робити, якщо XML-підпис не перевіряється?

Якщо XMLSignature.validate метод повертає false, ми можемо спробувати звузити причину невдачі. Основна перевірка XML-підпису складається з двох етапів:

- Перевірка підпису (криптографічна перевірка підпису)
- Перевірка посилання (перевірка дайджесту кожного посилання в підписі)

Кожен етап має бути успішним, щоб підпис був дійсним. Щоб перевірити, чи підпис не вдалося перевірити криптографічно, ми можемо перевірити статус таким чином:

```
boolean sv = signature.getSignatureValue().validate(valContext);
System.out.println("signature validation status: " + sv);
```

Рис 111

Ми також можемо переглянути посилання та перевірити статус перевірки кожного з них, як показано нижче:

```
Iterator<Reference> i =
    signature.getSignedInfo().getReferences().iterator();
for (int j=0; i.hasNext(); j++) {
    boolean refValid = i.next().validate(valContext);
    System.out.println("ref["+j+"] validity status: " + refValid);
}
```

Рис 111

3.4 Використання KeySelectors

KeySelectors використовуються для пошуку та вибору ключів, необхідних для перевірки XMLSignature. Раніше, коли ми створили DOMValidateContext об'єкт, ми передали KeyValueKeySelector об'єкт як перший аргумент:

```
DOMValidateContext valContext = new DOMValidateContext
    (new KeyValueKeySelector(), nl.item(0));
```

Рис 111

Крім того, ми могли б пройти PublicKey як перший аргумент, якщо ми вже знаємо, який ключ потрібен для перевірки підпису. Однак ми часто не знаємо.

Клас KeyValueKeySelector є конкретною реалізацією абстрактного KeySelector клас. Реалізація KeyValueKeySelector намагається знайти відповідний ключ перевірки, використовуючи дані, що містяться в KeyValue елементи в KeyInfo елемент XMLSignature. Він не визначає, чи є ключ надійним. Це дуже просто KeySelector реалізація, призначена для ілюстрації, а не для використання в реальному світі. Більш практичний приклад KeySelectorце той,

який шукає KeyStore для надійних ключів, які збігаютьсяX509Data інформацію (наприклад, X509SubjectName, X509IssuerSerial, X509SKI, або X509Certificate елементів), що містяться в KeyInfo.

3.5 Приклад GenEnveloped

Зразок програми згенерує конвертований підпис документа у файлі envelope.xml та збереже його у файлі envelopedSignature.xml в поточному робочому каталозі.

```
<Envelope xmlns="urn:envelope">
</Envelope>
```

Рис 111

3.6 Створення підпису XML

У цьому прикладі показано, як створити XML-підпис за допомогою XML Digital Signature API. Якщо говорити точніше, у цьому прикладі створюється XML-підпис у конверті XML-документа. Конвертований підпис — це підпис, який міститься всередині вмісту, який він підписує. У цьому прикладі використовується DOM (об'єктна модель документа) для аналізу XML-документа, який потрібно підписати, і реалізація DOM для генерації кінцевого підпису.

Базові знання XML-підписів та їхніх різних компонентів корисні для розуміння цього розділу. Для отримання додаткової інформації.

По-перше, ми використовуємо JAXPDocumentBuilderFactory щоб розібрати XML-документ, який ми хочемо підписати. Програма отримує стандартну реалізацію для DocumentBuilderFactory за допомогою виклику наступного рядка коду:

```
DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
```

Рис 111

Ми також повинні зробити простір імен фабрики обізнаним:

```
dbf.setNamespaceAware(true);
```

Рис 111

Далі ми використовуємо фабрику, щоб отримати екземпляр DocumentBuilder, який використовується для аналізу документа:

```
Document doc = null;
try (FileInputStream fis = new FileInputStream(args[0])) {
    doc = dbf.newDocumentBuilder().parse(fis);
}
```

Рис 111

3.7 Створення пари відкритих ключів

Ми генеруємо пару відкритих ключів. Далі в цьому прикладі ми будемо використовувати закритий ключ для створення підпису. Ми створюємо пару ключів за допомогою KeyPairGenerator. У цьому прикладі ми створимо RSAKeyPair довжиною 2048 байт:

```
KeyPairGenerator kpg = KeyPairGenerator.getInstance("RSA");
kpg.initialize(2048);
KeyPair kp = kpg.generateKeyPair();
```

Рис 111

На практиці закритий ключ зазвичай попередньо генерується та зберігається в KeyStore файл із відповідним сертифікатом відкритого ключа.

3.8 Створення контексту підпису

Ми створюємо XMLSignContext містить вхідні параметри для генерації підпису. Оскільки ми використовуємо DOM, ми створюємо екземпляр DOMSignContext(підклас XMLSignContext) і передайте йому два параметри:

приватний ключ, який використовуватиметься для підпису документа, і корінь документа, який потрібно підписати:

```
DOMSignContext dsc = new DOMSignContext
    (kp.getPrivate(), doc.getDocumentElement());
```

Рис 111

3.9 Складання підпису XML

Ми збираємо різні частини елемента Signature вXMLSignature об'єкт. Усі ці об’єкти створюються та збираються за допомогою XMLSignatureFactory об'єкт. Додаток отримує реалізацію DOMXMLSignatureFactory за допомогою виклику наступного рядка коду:

```
XMLSignatureFactory fac = XMLSignatureFactory.getInstance("DOM");
```

Рис 111

Потім ми викликаємо різні фабричні методи для створення різних частинXMLSignatureоб'єкт. Ми створюємо aReferenceоб'єкт, передаючи йому наступне:

- URI об’єкта, який потрібно підписати (ми вказуємо URI "", що означає корінь документа.)
- TheDigestMethod(ми використовуємо SHA256)
- СамотнійTransform, в конверті Transform, який потрібен для підписів у конвертах, щоб сам підпис було видалено перед обчисленням значення підпису

```
Reference ref = fac.newReference
    ("", fac.newDigestMethod(DigestMethod.SHA256, null),
     List.of
      (fac.newTransform
        (Transform.ENVELOPED, (TransformParameterSpec) null)),
     null, null);
```

Рис 111

Далі ми створюємоSignedInfoоб'єкт, який є фактично підписаним об'єктом. При створенні SignedInfo, ми передаємо як параметри:

- TheCanonicalizationMethod(ми використовуємо включно та зберігаємо коментарі)
- TheSignatureMethod(ми використовуємо RSA)
- Список References(в даному випадку тільки один)

```
SignedInfo si = fac.newSignedInfo
        (fac.newCanonicalizationMethod
         (CanonicalizationMethod.INCLUSIVE_WITH_COMMENTS,
          (C14NMethodParameterSpec) null),
         fac.newSignatureMethod("http://www.w3.org/2001/04/xmldsig
             -more#rsa-sha256", null),
         List.of(ref));
```

Рис 111

Далі ми створюємо необов’язковий KeyInfoоб'єкт, який містить інформацію, яка дозволяє одержувачу знайти ключ, необхідний для перевірки підпису. У цьому прикладі ми додаємо KeyValue об'єкт, що містить відкритий ключ. Створювати KeyInfoі його різні підтипи, ми використовуємо KeyInfoFactory об'єкт, який можна отримати, звернувшись до getKeyInfoFactory метод XMLSignatureFactory наступним чином:

```
KeyInfoFactory kif = fac.getKeyInfoFactory();
```

Рис 111

Потім ми використовуємо KeyInfoFactory створити KeyValuei додайте його до KeyInfo об'єкт:

```
KeyValue kv = kif.newKeyValue(kp.getPublic());
KeyInfo ki = kif.newKeyInfo(List.of(kv));
```

Рис 111

Нарешті ми створюємо XMLSignature об'єкт, передаючи як параметри SignedInfoiKeyInfo об'єкти, які ми створили раніше:

```
XMLSignature signature = fac.newXMLSignature(si, ki);
```

Рис 111

Зауважте, що ми фактично ще не згенерували підпис; ми зробимо це на наступному кроці.

3.10 Створення підпису XML

Тепер ми готові згенерувати підпис, що ми робимо, викликаючи метод sign на XMLSignaturei передайте йому контекст підписання наступним чином:

```
signature.sign(dsc);
```

Рис 111

Отриманий документ тепер містить підпис, який було вставлено як останній дочірній елемент кореневого елемента.

3.11 Друк або відображення отриманого документа

Ви можете використати такий код, щоб надрукувати отриманий підписаний документ у файл або стандартний вихід:

```
OutputStream os;
if (args.length > 1) {
   os = new FileOutputStream(args[1]);
} else {
   os = System.out;
}

TransformerFactory tf = TransformerFactory.newInstance();
Transformer trans = tf.newTransformer();
trans.transform(new DOMSource(doc), new StreamResult(os));
```

Рис 111

3.12 Використання в захисті Web-сервісів

Виникає питання, яким образом брандмауер XML використовує підписи і шифрування XML для захисту SOAP-серверів? Адже незважаючи на те, що можливості цих двох технологій були продемонстровані на численних прикладах, необхідно з'ясувати, як застосовувати ці дві специфікації при використанні брандмауера XML для захисту SOAP-серверів, особливо якщо врахувати жодна з них не є специфічною для SOAP. Спробуємо зрозуміти, чому вся інформація, що стосується підписи, була поміщена в заголовок SOAP, а не в тіло SOAP [3].

Брандмауер XML обробляє XML-запити та відповіді через HTTP або HTTPS.

Брандмауер XML використовує єдиний протокол і містить політику обробки з набором правил запитів, відповідей, двосторонніх і помилок. Його конфігурація визначає пари хост-порт, що прослуховують, і загальний захист від загроз.

Ця служба може обробляти XML-документи всіх типів, включаючи повідомлення у форматі SOAP, документи JSON і необроблені (текстові або двійкові) документи. За допомогою політики обробки служба може застосовувати різні дії обробки запиту та відповіді, незалежно від формату.

Ви можете налаштувати брандмауер XML для проксі віддалених служб. Загальною особливістю брандмауера XML є петлева служба. Віддалений сервер є невизначеним. У цьому випадку сервіс сам генерує відповідь на оброблений запит клієнта. Ця функція корисна під час розробки, коли віддалений сервер недоступний.

Брандмауер XML реалізує специфічні для сайту практики безпеки. Ви можете швидко налаштувати цю службу, але ви також можете визначити складні методи безпеки, які використовують всю потужність GatewayScript і XSLT.

Фільтрація трафіку. Сервіс забезпечує швидкісну фільтрацію вхідних документів XML або SOAP. Документи можна фільтрувати на кількох рівнях стека протоколів, щоб включити: вміст повідомлення на рівні поля XML, конверт SOAP, IP-адресу, ім'я хоста, номер порту, розмір корисного навантаження та інші характеристики трафіку.

Підтвердження та верифікація даних. Сервіс забезпечує легітимність і структуру документів, надаючи перевірку XML-схеми на швидкості з'єднання вхідних і вихідних документів. Політики, які налаштовуються користувачем, захищають від атак DoS, переповнення буфера та інших розпізнаних уразливостей, які можуть бути результатом шкідливих або неправильно сформованих документів XML.

Брандмауер XML надає жорсткий цифровий підпис на основі стандартів і можливості перевірки підпису на швидкості з'єднання.

Повідомлення та криптографія на рівні поля

Служба підтримує надійне 128-бітне шифрування, яке можна застосувати на рівні повідомлень або на рівні поля XML. На рівні повідомлення ви можете зашифрувати та розшифрувати весь документ. На рівні полів можна шифрувати та розшифровувати лише конфіденційні поля даних.

Контроль доступу до веб-служб XML Сервіс підтримує різні методи автентифікації та авторизації веб-сервісів.

Віртуалізація сервісу. Сервіс забезпечує політики перезапису URL-адрес, високопродуктивні перетворення XSL і маршрутизацію на основі XML і SOAP. Маршрутизація відображає клієнтські запити на набір захищених віддалених ресурсів.

Найбільш популярним і широко використовуваним маркером безпеки є пара логін-пароль, подібна до тієї, яку ви використовуєте під час перевірки електронної пошти.

Пара логін-пароль — це зрозумілий для людини маркер безпеки. Є деякі

маркери безпеки, які мають двійкову форму (і тому не обов'язково читаються людиною). Такі токени називаються бінарними токенами безпеки. Наприклад, сертифікат X509 (широко популярний формат цифрових сертифікатів, розроблений ITU-T) є двійковим маркером безпеки.

Атрибут ValueTypeелемента wsse:BinarySecurityTokenв лістингу 9 повідомляє, який тип бінарного маркера безпеки загорнуто в цей BinarySecurityToken елемент. У лістингу 9ValueType атрибут містить wsse:X509v3значення, яке ідентифікує сертифікати X509 .

Атрибут EncodingTypeелемента wsse:BinarySecurityTokenповідомляє кодування двійкового маркера безпеки. Як уже пояснювалося, двійкові дані неможливо загорнути у формат XML як такий. Тому ми повинні закодувати двійкові дані (зазвичай у вигляді послідовності значень, закодованих за базою 64), перш ніж загорнути в XML. Сертифікат X509 загорнутий всередину wsse:BinarySecurityTokenелемента як вміст елемента.

Елемент ds:Signatureтакий самий, як той, який ми обговорювали в розділі про підписи XML. Зверніть увагу на дві важливі речі:

Подивіться на URIатрибут Reference елемента. Його значення (#myDiscountRequestBody) є ідентифікатором фрагмента, який вказує на елемент SOAP:Body. Це означає, що елемент SOAP:Body є тим, який ми підписали та загорнули підпис у теги XMLDS.

По-друге, також подивіться, що ds:KeyInfoмістить елемент. Це wsse:SecurityTokenReferenceелемент. Елемент wsse:SecurityTokenReferenceмістить посилання на маркери безпеки. У нашому випадку він має дочірній елемент під назвою wsse:Reference, атрибут URI якого вказує на wsse:BinarySecurityTokenелемент, розглянутий у пункті 3 вище. Це означає, що відкритий ключ у сертифікаті X509 (який wsse:BinarySecurityTokenобгортає елемент) використовуватиметься для перевірки підпису.

Це дуже простий приклад представлення підписаних повідомлень WSS. У третій і четвертій частинах цієї серії докладніше буде розглянуто безпеку на

основі XML, різні типи маркерів безпеки, які ми можемо використовувати з WSS, і використання шифрування XML у повідомленнях WSS.

ВИСНОВКИ

Спочатку розроблено та розглянуто для забезпечення автентифікації та цілісності функції, цифрові підписи в даний час вивчаються по відношенню до електронних документів, щоб їх можна було вважати еквівалентними рукописним підписам, застосованим на паперові документи. Тим не менш, стандартизований формат, який буде використовуватися спеціально для електронних документів представництво ще не вказано. Кожна система управління документами є безкоштовною вибрати будь-який формат електронних документів, який відповідає його вимогам (наприклад, ASCII, Word, PDF, двійковий). Поки що деякі рішення для систем управління документами були знайдені, але жодного з них було розроблено, щоб вважати безпеку важливою вимогою та увімкнути цифрові підписання та просте керування електронними документами. Можливим вирішенням цієї проблеми є наш сейф система документообігу XML Digital Signature API.

Проект XML Digital Signature API реалізував захищений XML система управління документами, яку можна легко адаптовані до різних типів сценаріїв реального світу де електронні документи повинні мати цифровий підпис. Один з основними вимогами на етапі проектування був аналіз і виберіть формат для цифрового підпису та розробити та реалізувати формат електронного документа. Це документ описує найбільш широко використовувані формати для підпис криптографічних конвертів і описує представлення e-doc в XML Digital Signature API на основі XML та Підпис XML.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. XML – Signature Syntax and Processing (W3C Consortium Feb 12, 2002) http://www.w3.org/TR/xmldsig-core/
2. XML – Signature Interoperability (W3C Consortium May 2002) http://www.w3.org/Signature/2001/04/05-xmldsig-interop.html
3. Namespace in XML (W3C Consortium January 14, 1999) http://www.w3.org/TR/REC-xml-names/
4. XML Electronic Signatures (Institute for Applied Information Processing and
5. Communication Sept 24, 2001) http://www.nanobiz.com/xmlsig/docs/iaik/iaik.htm
6. XML Namespaces by Example (XML.COM January, 1999) http://www.xml.com/lpt/a/1999/01/namespaces.html
7. Canonical XML (W3C Consortium March 15, 2001) http://www.w3.org/TR/2001/REC-xml-c14n-20010315
8. Digest Value for DOM (IBM April 2000) http://www.ietf.org/rfc/rfc2803.txt
9. Secured hash standard (Federal Information Processing Standards April 17, 1995) http://csrc.nist.gov/publications/fips/fips180-1/fip180-1.txt
10. Digital Signature Standard (Federal Information Processing Standards January 27, 2000) http://csrc.nist.gov/publications/fips/fips186-2/fips186-2.pdf
11. RSA Cryptography Standard (RSA Laboratory Oct 1998) http://www.ietf.org/rfc/rfc2437.txt
12. IBM XML Security Suite (IBM April 26, 1999) http://www.alphaworks.ibm.com/tech/xmlsecuritysuite
13. VeriSign XML Key Management (VeriSign October 2001) http://www.verisign.com/resources/wp/xml/keyManagement.pdf
14. Bray, T., Paoli, J. and C.M. Sperberg-McQueen (1998). Extensible Markup Language (XML) 1.0, W3C Recommendation, available at http://www.w3.org/TR/1998/REC-xml19980210
15. XPATH. (1999). The W3C, Recommendation, XML Path Language 1.0, 16 November 1999, available at http://www.w3.org/TR/xpath XMLSchema.

(2001).

16. XML Schema 1.1, World Wide Web Consortium (W3C) Recommendation, available at http://www.w3.org/XML/Schema
17. XSLT. (2002). XSL Transformations 2.0, The W3C, Recommendation, available at http://www.w3c.org/ Style/XSL
18. XMLDSIG. (2000). XML Signature, W3C & IETF, Candidate Recommendation, October 2000, available at http://www.w3.org/Signature
19. ITU-T Recommendation X.509-ISO/IEC 9594-8. (2000). Information Technology – Open Systems Interconnection - The Directory: Public Key and Attribute Certificate Frameworks.
20. Housley, R., Ford, W., Polk, W., and D. Solo (2002). Internet X.509 Public Key Infrastructure Certificate and CRL Profile, IETF, RFC-3280. SSL. (1996). The Secure Sockets Layer
21. (SSL) Protocol - Version 3.0, available at http://home.netscape.com/eng/ssl3/ssl-toc.html Adams, C., Cain, P., Pinkas, D., and R. Zuccherato (2001). Internet X.509 Public Key Infrastructure Time Stamp Protocol (TSP), IETF, RFC-3161.
22. AIDA. (1999). Advanced Interactive Digital Administration Project, http://aida.infonova.at. Berbecaru, D., Lioy, A., and M. Marian (2001). A Framework for Secure Digital Administration, Proceedings of EuroWeb2001 Conference, pp: 119-133.