УДК 004

В.Шевченко, магістр гр. КН-21М-1,4,
Центральноукраїнський національний технічний університет

ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ДОСТУПУ ДО ХМАРНИХ СЕРВІСІВ З ВИКОРИСТАННЯМ ТЕХНОЛОГІЇ PKI

У статті розроблено програмне забезпечення, яке призначено для системи доступу до хмарних сервісів з використанням технології PKI. Метою розробки є дослідження та програмна реалізація системи доступу до хмарних сервісів з використанням технології PKI. Об'єктом дослідження є процес доступу до хмарних сервісів з використанням технології PKI. Предметом дослідження є методи доступу до хмарних сервісів з використанням технології PKI. Методи дослідження базуються на методах захисту інформації та хмарних обчислень, методах математичної статистики, методах розробки програмного забезпечення. Результат роботи – програмна реалізація системи доступу до хмарних сервісів з використанням технології PKI. В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.
комп'ютерні науки, хмарні сервіси, PKI

Постановка проблеми. Сьогодні більшість компаній так чи інакше використовують Інтернет, і із цим зв'язані проблеми захисту віддалених і мобільних користувачів інформаційних систем компанії, захисту корпоративних хмарних сервісів (інтранет-сайту й будь-якого додатка компанії, що працює по http протоколу). Інтернет – це зона підвищеного ризику, відповідно, потрібні спеціальні засоби захисту при роботі віддалених користувачів з WEB-додатками по SSL-протоколу. Таким чином, підсистема захисту WEB-ресурсів вирішує наступні задачі:

- Забезпечення єдиного інтерфейсу до додатків;
- інтегрований контроль доступу до корпоративних хмарних сервісів;
- захист клієнтських браузерів;
- захист хмарних сервісів.

Існує багато технологій захисту хмарних сервісів. У магістерському проекті пропонується система захисту основана на використанні протоколів SSL/TLS, які побудовані з використання інфраструктури відкритих ключів (PKI).

У протоколі SSL/TLS використовується ряд симметричних алгоритмів, асиметричних алгоритмів та геш-функцій. Тому одним із завдань, які потрібно вирішити у даному магістерському проекті є вибір того, або іншого алгоритму, які використовуються на різних етапах побудови системи захисту доступу до хмарних сервісів.

Технологія PKI дозволяє перевіряти й засвідчувати дійсність користувача. PKI забезпечує єдину ідентифікацію, автентифікацію й авторизацію користувачів системи, додатків і процесів і разом із цим гарантує доступність, цілісність і конфіденційність інформації.

Аналіз останніх досліджень і публікацій. При аналізі останніх досліджень і публікацій [1-10] було виявлено певні прогалини у забезпеченні системи доступу до хмарних сервісів з використанням технології PKI.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи доступу до хмарних сервісів з використанням технології PKI.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Огляд існуючих систем доступу до хмарних сервісів з використанням технології PKI.

– Дослідження системи доступу до хмарних сервісів з використанням технології PKI.

– Програмна реалізація системи доступу до хмарних сервісів з використанням технології PKI.

Об'єктом дослідження є процес доступу до хмарних сервісів з використанням технології PKI.

Предметом дослідження є методи доступу до хмарних сервісів з використанням технології PKI.

Методи дослідження базуються на методах захисту інформації та хмарних обчислень, методах математичної статистики, методах розробки програмного забезпечення.

Виклад основного матеріалу. Опис SSL/TLS

SSL – криптографічний протокол, що забезпечує безпечну передачу даних по мережі Інтернет. При його використанні створюється захищене з'єднання між клієнтом і сервером. SSL споконвічно розроблений компанією Netscape Communications. Згодом на підставі протоколу SSL 3.0 був розроблений і прийнятий стандарт RFC, що одержав ім'я TLS.

TLS – криптографічний протокол, що забезпечує захищену передачу даних між вузлами в мережі Інтернет. TLS-протокол заснований на Netscape SSL-протоколі версії 3.0 і складається із двох частин – TLS Record Protocol і TLS Handshake Protocol. Розходження між SSL 3.0 і TLS 1.0 незначні, тому далі в тексті термін «SSL» буде відноситися до них обох. TLS Working Group, заснована в 1996 році, продовжує працювати над протоколом.

Опис

TLS надає можливості автентифікації й безпечної передачі даних через Інтернет з використанням криптографічних засобів. Часто відбувається лише автентифікація сервера, у той час як клієнт залишається неавтентифікованим. Для взаємної автентифікації кожна зі сторін повинна підтримувати інфраструктуру відкритого ключа (PKI), що дозволяє захистити клієнт-серверні додатки від перехоплення повідомлень, редагування існуючих повідомлень і створення підроблених.

SSL містить у собі три основних фази:

–Діалог між сторонами, метою якого є вибір алгоритму шифрування.

–Обмін ключами на основі криптосистем з відкритим ключем або автентифікація на основі сертифікатів.

–Передача даних, шифруємих за допомогою симетричних алгоритмів шифрування.

Алгоритм процедури встановлення з'єднання по протоколу TLS handshake

Клієнт і сервер, що працюють по TLS, установлюють з'єднання, використовуючи процедуру handshake ("рукостискання"). Протягом цього handshake, клієнт і сервер приймають угоду щодо параметрів, використовуваних для встановлення захищеного з'єднання. Послідовність дій при встановленні TLS з'єднання:

– клієнт підключається до TLS – підтримуваного сервера й запитує захищене з'єднання;

– клієнт надає список підтримуваних алгоритмів шифрування й геш-функцій;

– сервер вибирає зі списку, наданого клієнтом, найбільш стійкі алгоритми, які також підтримуються сервером, і повідомляє про свій вибір клієнтові;

– сервер відправляє клієнтові цифровий сертифікат для власної ідентифікації. Звичайно цифровий сертифікат містить ім'я сервера, ім'я довіреного центра сертифікації й відкритий ключ сервера;

– клієнт може зв'язатися із сервером довіреного центра сертифікації й підтвердити автентичність переданого сертифіката до початку передачі даних;

– для того щоб згенерувати ключ сесії для захищеного з'єднання, клієнт шифрує випадково згенеровану цифрову послідовність відкритим ключем сервера й посилає результат на сервер. З огляду на специфіку алгоритму асиметричного шифрування,

використовуваного для встановлення з'єднання, тільки сервер може розшифрувати отриману послідовність, використовуючи свій закритий ключ.

Handshake у деталях

Відповідно до протоколу TLS додатки обмінюються записами, інкапсулюючими (що зберігають усередині себе) інформацію, яка повинна бути передана. Кожен із записів може бути стисла, доповнена, зашифрована або ідентифікована MAC залежно від поточного стану з'єднання (стану протоколу). Кожний запис в TLS містить наступні поля: content type (визначає тип умісту запису), поле, що вказує довжину пакета, і поле, що вказує версію протоколу TLS.

Коли з'єднання тільки встановлюється, взаємодія йде по протоколу TLS handshake, content type якого 22.

Нижче описаний простий приклад установлення з'єднання:

1. Клієнт посилає повідомлення **ClientHello**, указуючи найбільш останню версію підтримуваного TLS протоколу, випадкове число й список підтримуваних методів шифрування й стиски, що підходять для роботи з TLS.
2. Сервер відповідає повідомленням ServerHello, що містить: обрану сервером версію протоколу, випадкове число, послане клієнтом, що підходить алгоритм шифрування й стиски зі списку наданого клієнтом.
3. Сервер посилає повідомлення **Certificate**, що містить цифровий сертифікат сервера (залежно від алгоритму шифрування цей етап може бути пропущений)
4. Сервер може запросити сертифікат у клієнта, у такому випадку з'єднання буде взаємно автентифіковано.
5. Сервер відсилає повідомлення **ServerHelloDone**, що ідентифікує закінчення handshake.
6. Клієнт відповідає повідомленням **ClientKeyExchange**, що містить PreMasterSecret відкритий ключ, або нічого (знову ж залежить від алгоритму шифрування).
7. Клієнт і сервер, використовуючи PreMasterSecret ключ і випадково згенеровані числа, обчислюють загальний секретний ключ. Вся інша інформація про ключ буде отримана із загального секретного ключа (і згенерованих клієнтом і сервером випадкових значень).
8. Клієнт посилає **ChangeCipherSpec** повідомлення, що вказує на те, що вся наступна інформація буде зашифрована встановленим у процесі handshake алгоритмом, використовуючи загальний секретний ключ. Це повідомлення рівня записів і тому має тип 20, а не 22.
9. Клієнт посилає повідомлення **Finished**, що містить геш і MAC, згенеровані на основі попередніх повідомлень handshake.
10. Сервер намагається розшифрувати Finished-повідомлення клієнта й перевірити геш і MAC. Якщо процес розшифровки або перевірки не вдається, handshake вважається невдалим і з'єднання повинне бути обірване.
11. Сервер посилає **ChangeCipherSpec** і зашифроване **Finished** повідомлення й у свою чергу клієнт теж виконує розшифровку й перевірку.

Із цього моменту handshake уважається завершеним, протокол установленим. Весь наступний уміст пакетів іде з типом 23, а всі дані будуть зашифровані.

Алгоритми, що використовуються в TLS

У даній поточній версії протоколу доступні наступні алгоритми:

– Для обміну ключами й перевірки їхньої дійсності застосовуються комбінації алгоритмів: RSA (асиметричний шифр), Diffie-Hellman (DH) (безпечний обмін ключами), DSA (алгоритм цифрового підпису) і алгоритми технології Fortezza.

– Для симетричного шифрування: RC2, RC4, IDEA, DES, Triple DES або AES;

– Для геш-функцій: MD5 або SHA.

Алгоритми можуть доповняться залежно від версії протоколу.

На рисунку 1 наведено архітектуру SSL/TLS.

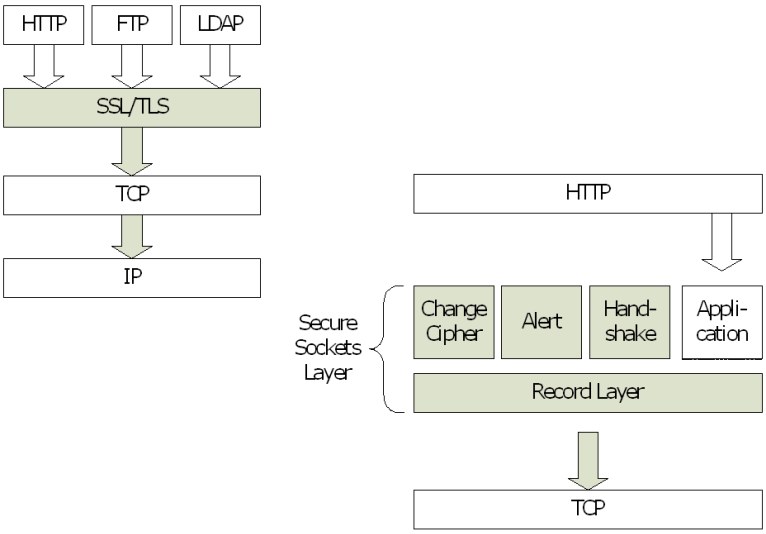

Рисунок 1 – Архітектура SSL/TLS

Як ми бачимо в архітектурі задіяні наступні протоколи:

– HTTP – протокол прикладного рівня передачі даних у першу чергу у вигляді текстових повідомлень. Основою HTTP є технологія «клієнт-сервер», тобто передбачається існування споживачів (клієнтів), які ініціюють з'єднання й надсилають запит, і постачальників (серверів), які очікують з'єднання для одержання запиту, роблять необхідні дії й повертають обернено повідомлення з результатом. HTTP використовується також у якості «транспорту» для інших протоколів прикладного рівня, таких як SOAP. Основним об'єктом маніпуляції в HTTP є ресурс, на який вказує URI (Uniform Resource Identifier) у запиті клієнта. Звичайно такими ресурсами є файли, що зберігаються на сервері, але ними можуть бути логічні об'єкти або щось абстрактне. Особливістю протоколу HTTP є можливість указати в запиті й відповіді спосіб подання того самого ресурсу по різних параметрах: формату, кодуванню, мові й т.д. Саме завдяки можливості вказівки способу кодування повідомлення клієнт і сервер можуть обмінюватися двійковими даними, хоча даний протокол є текстовим. HTTP – протокол прикладного рівня, аналогічними йому є FTP і SMTP. Обмін повідомленнями йде за звичайною схемою «запит-відповідь». Для ідентифікації ресурсів HTTP використовує глобальні URI. На відміну від багатьох інших протоколів, HTTP не зберігає свого стану. Це означає відсутність збереження проміжного стану між парами «запит-відповідь». Компоненти, що використовують HTTP, можуть самостійно здійснювати збереження інформації про стан, пов'язаної з останніми запитами й відповідями. Браузер, що посилає запити, може відслідковувати затримки відповідей. Сервер може зберігати IP-адреси й заголовки запитів останніх клієнтів. Однак сам протокол не обізнаний про попередні запити й відповіді, у ньому не передбачена внутрішня підтримка стану, до нього не пред'являються такі вимоги.

– FTP – протокол, призначений для передачі файлів у комп'ютерних мережах. FTP дозволяє підключатися до серверів FTP, переглядати вміст каталогів і завантажувати файли із сервера або на сервер; крім того, можливий режим передачі файлів між серверами. Протокол FTP відноситься до протоколів прикладного рівня й для передачі даних використовує транспортний протокол TCP. Команди й дані, на відміну від більшості інших протоколів передаються по різних портах. Порт 20 використовується для передачі даних, порт 21 для передачі команд. Протокол не шифрується, при автентифікації передає логін і пароль відкритим текстом. Якщо зловмисник перебуває в одному сегменті мережі з користувачем FTP, те, використовуючи сніффер, він може перехопити логин і пароль користувача, або, при наявності спеціального ПЗ, одержувати передані по FTP файли без авторизації. Щоб запобігти перехопленню трафіку, необхідно використовувати протокол

шифрування даних SSL, що підтримується багатьма сучасними FTP-серверами й деякими FTP-Клієнтами.

– LDAP – це мережний протокол для доступу до служби каталогів X.500, розроблений IETF як полегшений варіант розробленого ITU-T протоколу DAP. LDAP – відносно простий протокол, що використовує TCP/IP і дозволяє робити операції автентифікації (bind), пошуку (search) і порівняння (compare), а також операції додавання, зміни або видалення записів. Звичайно LDAP-сервер приймає вхідні з'єднання на порт 389 по протоколах TCP або UDP. Для LDAP-сеансів, інкапсульованих в SSL, звичайно використовується порт 636. Усякий запис у каталозі LDAP складається з одного або декількох атрибутів і має унікальне ім'я (DN). Унікальне ім'я складається з одного або декількох відносних унікальних імен (RDN), розділених комою. На одному рівні каталогу не може існувати двох записів з однаковими відносними унікальними іменами. У силу такої структури унікального ім'я запису в каталозі LDAP можна легко представити у вигляді дерева. Запис може складатися тільки з тих атрибутів, які визначені в описі класу запису (object class), які, у свою чергу, об'єднані в схеми (schema). У схемі визначено, які атрибути є для даного класу обов'язковими, а які – необов'язковими. Також схема визначає тип і правила порівняння атрибутів. Кожний атрибут запису може зберігати кілька значень.

– TCP – один з основних мережних протоколів Internet, призначений для керування передачею даних у мережах і під мережах TCP/IP. Виконує функції протоколу транспортного рівня спрощеної моделі OSI. IP-ідентифікатор – 6. TCP – це транспортний механізм, що надає потік даних, з попередньою установкою з'єднання, за рахунок цього даючий впевненість у вірогідності одержуваних даних, здійснює повторний запит даних у випадку втрати даних і усуває дублювання при одержанні двох копій одного пакета. На відміну від UDP, гарантує, що додаток одержить дані точно в такій же послідовності, у якій вони були відправлені, і без втрат. Реалізація TCP як правило, убудована в ядро системи, хоча є й реалізації TCP у контексті додатка. TCP часто позначають «TCP/IP». Коли здійснюється передача від комп'ютера до комп'ютера через Internet, TCP працює на верхньому рівні між двома кінцевими системами, наприклад, інтернет-браузер і інтернет-сервер. Також TCP здійснює надійну передачу потоку байт від однієї програми на деякому комп'ютері в іншу програму на іншому комп'ютері. Програми для електронної пошти й обміну файлами використовують TCP. TCP контролює довжину повідомлення, швидкість обміну повідомленням, мережний трафік.

– IP – маршрутизуємий мережний протокол, основа стека протоколів TCP/IP. Протокол IP використовується для негарантованої доставки даних (поділюваних на так звані пакети) від одного вузла мережі до іншого. Це означає, що на рівні цього протоколу (третій рівень мережної моделі OSI) не дається гарантій надійної доставки пакета до адресата. Зокрема, пакети можуть прийти не в тому порядку, у якому були відправлені, продублюватися (коли приходять дві копії одного пакета; у реальності це буває вкрай рідко), виявитися ушкодженими (звичайно ушкоджені пакети знищуються) або не прибути зовсім. Гарантії безпомилкової доставки пакетів дають протоколи більш високого (транспортного) рівня мережної моделі OSI – наприклад, TCP – які IP використовують як транспорт. У сучасній мережі Інтернет використовується IP четвертої версії, також відомий як IPv4. У протоколі IP цієї версії кожному вузлу мережі відноситься у відповідність IP-адреса довжиною 4 октети (іноді говорять «байта», маючи на увазі розповсюджений восьмибітовий мінімальний адресуємий фрагмент пам'яті ЕОМ). При цьому комп'ютери в підмережах поєднуються загальними початковими бітами адреси. Кількість цих біт, загальна для даної підмережі, називається маскою підмережі (раніше використовувалося ділення простору адрес по класах – A, B, C; клас мережі визначався діапазоном значень старшого октету й визначав число адресуємих вузлів у даній мережі, зараз використовується безкласова адресація). У поточний час вводиться в експлуатацію шоста версія протоколу – IPv6, що дозволяє адресувати значно більшу кількість вузлів, чим IPv4. Ця версія відрізняється підвищеною розрядністю адреси, убудованою можливістю шифрування й деяких інших

особливостей. Перехід з IPv4 на IPv6 пов'язаний із трудомісткою роботою операторів зв'язку й виробників програмного забезпечення й не може бути виконаний одномоментно.

Опис алгоритму шифрування AES

Шифрування

Для AES довжина input (блоку вхідних даних) і State (стану) постійна й дорівнює 128 бітів, а довжина шифроключа **K** становить 128, 192, або 256 бітів. Для позначення обраних довжин input, State і Cipher Key у байтах використовується нотація $Nb = 4$ для input і State, $Nk = 4, 6, 8$ для Cipher Key відповідно для різних довжин ключів.

На початку шифрування input копіюється в масив State за правилом $s[r,c] = \text{in}[r + 4c]$, для $0 \le r \le 4$ і $0 \le c < Nb$. Після цього до State застосовується процедура AddRoundKey() і потім State проходить через процедуру трансформації(раунд) 10, 12, або 14 разів (залежно від довжини ключа), при цьому треба врахувати, що останній раунд трохи відрізняється від попередніх. У підсумку, після завершення останнього раунду трансформації, State копіюється в output за правилом $\text{out}[r + 4c] = s[r,c]$, для $0 \le r \le 4$ і $0 \le c < Nb$.

SubBytes()

У процедурі SubBytes, кожний байт в state заміняється відповідним елементом у фіксованої 8-бітній таблиці пошуку, S; $b_{ij} = S(a_{ij})$.

Процедура SubBytes() обробляє кожний байт стану, незалежно роблячи нелінійну заміну байтів використовуючи таблицю замін (S-box). Така операція забезпечує нелінійність алгоритму шифрування. Побудова S-box складається із двох кроків. По-перше, виробляється взяття оберненого числа в $GF(2^8)$. По-друге, до кожного байта b з яких складається S-box застосовується наступна операція:

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

де $0 \le i \le 8$, і де b_i є i-ий біт b, а c_i – i-ий байт c = {63} або {01100011}. У такий спосіб забезпечується захист від атак заснованих на простих алгебраїчних властивостях.

У процедурі ShiftRows, байти в кожному рядку state циклічно зсувається вліво. Розмір зсуву байтів кожного рядка залежить від його номера.

ShiftRows працює з рядками State. При цій трансформації рядка стани циклічно зсувається на r байт по горизонталі, залежно від номера рядка. Для нульового рядка $r = 0$, для першого рядка $r = 1$, і т.д. У такий спосіб кожний стовпчик вихідного стану після застосування процедури ShiftRows складається з байтів з кожного стовпчика початкового стану. Для алгоритму Rijndael паттерн зсуву рядків для 128 і 192-бітних рядків однаковий. Однак для блоку розміром 256 бітів відрізняється від попередніх тим, що 2, 3, і 4-ий рядки зміщаються на 1, 3, і 4 байти відповідно.

MixColumns()

У процедурі MixColumns, кожний стовпчик стану перемножується з фіксованим багаточленом $c(x)$.

У процедурі MixColumns, чотири байти кожного стовпчика State змішуються використовуючи для цього оборотну лінійну трансформацію. MixColumns обробляє стан по стовпчиках, трактуючи кожен з них як поліном четвертого ступеня. Над цими поліномами виробляється множення в $GF(2^8)$ по модулю $x^4 + 1$ на фіксований багаточлен $c(x) = 3x^3 + x^2 + x + 2$. Разом з ShiftRows, MixColumns вносить diffusion у шифр.

AddRoundKey()

У процедурі AddRoundKey, кожний байт стану поєднується з RoundKey використовуючи XORoperation(). У процедурі AddRoundKey, RoundKey кожного раунду поєднується з State. Для кожного раунду Roundkey виходить із CipherKey використовуючи процедуру KeyExpansion; кожний RoundKey такого ж розміру, що й State. Процедура робить побітовий XOR кожного байта State з кожним байтом RoundKey.

KeyExpansion()

AES алгоритм використовуючи процедуру KeyExpansion() і подаючи в неї Cipher Key, K, одержує ключі для всіх раундів. Усього вона одержує $Nb*(Nr + 1)$ слів: с початку для алгоритму потрібен набір з Nb слів, і кожному з Nr раундів потрібно Nb ключових наборів даних. Отриманий масив ключів для раундів позначається як $w[i]$, $0 \le i < Nb*(Nr + 1)$. Алгоритм KeyExpansion() показаний у псевдокоді нижче

Функція SubWord() бере чотирьохбайтне вхідне слово й застосовує *S*-box до кожного із чотирьох байтів, те, що вийшло подається на вихід. На вхід процедури RotWord() подається слово $[a_0,a_1,a_2,a_3]$ яке вона циклічно переставляє й повертає $[a_1,a_2,a_3,a_0]$. Масив слів, слів постійний для даного раунду, $Rcon[i]$, містить значення $[x^{i-1},00,00,00]$, де $x = \{02\}$, а x^{i-1} є ступенем x в $GF(2^8)$ (i починається з 1).

Перші Nk слів розширеного ключа заповнені Cipher Key. У кожне наступне слово, $w[i]$, кладе значення отримане при операції XOR $w[i\text{-}1]$ і $w[i - Nk]$. Для слів, позиція яких кратна Nk, перед XOR'ом до $w[i\text{-}1]$ застосовується трасформація, за якою слідує XOR з константою раунду $Rcon[i]$. Зазначена вище трансформація складається із циклічного зсуву байтів у слові(RotWord()), за якою слідує процедура SubWord() – теж саме, що й SubBytes(), тільки вхідні й вхідні дані будуть розміром у слово.

Важливо помітити, що процедура KeyExpansion() для 256 бітного Cipher Key не набагато відрізняється від тих, які застосовуються для 128 і 192 бітних шифроключів. Якщо $Nk = 8$ й $i - 4$ кратно Nk, то SubWord() застосовується до $w[i\text{-}1]$ до XOR'a.

Опис протоколу обміну ключами Діффі-Хеллмана

Ціль алгоритму полягає в тому, щоб два учасники могли безпечно обмінятися ключем, що надалі може використовуватися в якому-небудь алгоритмі симетричного шифрування. Сам алгоритм Діффі-Хеллмана може застосовуватися тільки для обміну ключами. Алгоритм заснований на труднощі обчислень дискретних логарифмів. Дискретний логарифм визначається в такий спосіб. Уводиться поняття примітивного кореня простого числа Q як числа, чиї ступені створюють всі цілі від 1 до $Q - 1$. Це означає, що якщо A є примітивним коренем простого числа Q, тоді числа: $A \bmod Q$, $A^2 \bmod Q$, . . . , $A^{Q-1} \bmod Q$, є різними й складаються із цілих від 1 до $Q - 1$ з деякими перестановками. У цьому випадку для будь-якого цілого $Y < Q$ і примітивного кореня A простого числа Q можна знайти єдину експоненту X, таку, що:

$$Y = A^X \bmod Q, \text{ де } 0 \le X \le (Q - 1).$$

Експонента X називається дискретним логарифмом, або індексом Y, по підставі $A \bmod Q$. Це позначається як:

$$\text{ind}_{A, Q}(Y).$$

Тепер опишемо алгоритм обміну ключів Діффі-Хеллмана.

Загальновідомі елементи

Q – просте число

A – $A < Q$ і A є примітивним коренем Q

Створення пари ключів клієнтом

Вибір випадкового числа X_i (*закритий ключ*): $X_i < Q$

Обчислення числа Y_i (*відкритий ключ*): $Y_i = A^{Xi} \bmod Q$

Створення відкритого ключа сервером

Вибір випадкового числа X_j (*закритий ключ*): $X_j < Q$

Обчислення випадкового числа Y_j (*відкритий ключ*): $Y_j = A^{Xj} \bmod Q$

Створення спільного секретного ключа клієнтом

$$K = (Y_j)^{Xi} \bmod Q.$$

Створення спільного секретного ключа сервером

$$K = (Y_i)^{Xj} \bmod Q.$$

Розпишемо алгоритм більш докладно. Передбачається, що існують два відомих усім числа: просте число Q і ціле A, що є примітивним коренем Q. Тепер припустимо, що клієнт та сервер хочуть обмінятися ключем для алгоритму симетричного шифрування. Клієнт

вибирає випадкове число $X_i < Q$ і обчислює $Y_i = A^{Xi} \bmod Q$. Аналогічно сервер незалежно вибирає випадкове ціле число $X_j < Q$ і обчислює $Y_j = A^{Xj} \bmod Q$. Кожна сторона тримає значення X у секреті й робить значення Y доступним для іншої сторони. Тепер клієнт обчислює ключ як $K = (Y_j)^{Xi} \bmod Q$, і сервер обчислює ключ як $K = (Y_i)^{Xj} \bmod Q$. У результаті обоє одержать те саме значення:

$$K = (Y_j)^{Xi} \bmod Q = (A^{Xj} \bmod Q)^{Xi} \bmod Q = (A^{Xj})^{Xi} \bmod Q =$$

(за правилами модульної арифметики)

$$= A^{Xj\,Xi} \bmod Q = (A^{Xj})^{Xj} \bmod Q = (A^{Xi} \bmod Q)^{Xj} \bmod Q = (Y_i)^{Xj} \bmod Q$$

Таким чином, дві сторони обмінялися секретним ключем. Так як X_i і X_j є закритими, зловмисник може одержати тільки наступні значення: Q, A, Y_i і Y_j. Тобто дві взаємодіючі сторони по відкритому каналу змогли поширити секретний ключ не розкриваючи його третій стороні.

Для обчислення ключа атакуючий повинен зламати дискретний логарифм, тобто обчислити: $X_j = \mathrm{ind}_{a,\,q}(Y_j)$.

Безпека обміну ключа в алгоритмі Діффі-Хеллмана випливає з того факту, що, хоча відносно легко обчислити експоненти за модулем простого числа, дуже важко обчислити дискретні логарифми. Для великих простих чисел задача вважається нерозв'язною.

Варто помітити, що даний алгоритм уразливий для атак типу "in-the-middle". Якщо зловмисник може здійснити активну атаку, тобто має можливість не тільки перехоплювати повідомлення, але й заміняти їх іншими, він може перехопити відкриті ключі учасників Y_i і Y_j , створити свою пару відкритого й закритого ключа (X_{on}, Y_{on}) і послати кожному з учасників свій відкритий ключ. Після цього кожний учасник обчислить ключ, що буде спільним із зловмисником, а не з іншим учасником. Якщо немає контролю цілісності, то учасники не зможуть виявити подібну підміну.

Розробка структурної схеми

Розроблене програмне забезпечення представляє із себе набір компонентів призначених для забезпечення політики безпеки як у вже існуючих, так і в створюваних мережних інформаційних системах.

Розроблене програмне забезпечення дозволяє забезпечити захист переданої по мережі інформації, строгу взаємну автентифікацію користувачів і серверів, гнучке розмежування доступу. Для реалізації цих функцій у системі використовуються SSL/TLS протоколи й X.509 цифрові сертифікати, тобто універсальні, що стали стандартом де-факто, механізми, підтримувані практично всіма розповсюдженими Веб-агентами.

За допомогою розробленого програмного забезпечення легко забезпечуються вимоги по інформаційній безпеці, запропоновані різними Інтернет додатками, такими як сервера платіжних систем, інтернет-магазини, багатопрофільні корпоративні Веб-сервера, що містять інформацію з різним рівнем конфіденційності, B2B системи, системи захищеного документообігу, обміну електронною поштою й багато які інші.

На рисунку 2 представлена структурна схема розробленої системи. На цій схемі введені наступні позначення:

- ЕЦП – електроний цифровий підпис;
- ЦС – цифровий сертифікат;
- PKI – інфраструктура відкритих ключів;
- DVCS – Data Validation and Certification Server Protocols – протокол підтвердження даних та сертифікації серверу;
- OCSP – Online Certificate Status Protocol – онлайн протокол статусу сертфікату;
- TSP – Time-Stamp Protocol – протокол часових міток;
- TLS – криптографічний протокол, що забезпечує захищену передачу даних між вузлами в мережі Інтернет;
- RFC – документ, у якому описується той або інший стандарт.

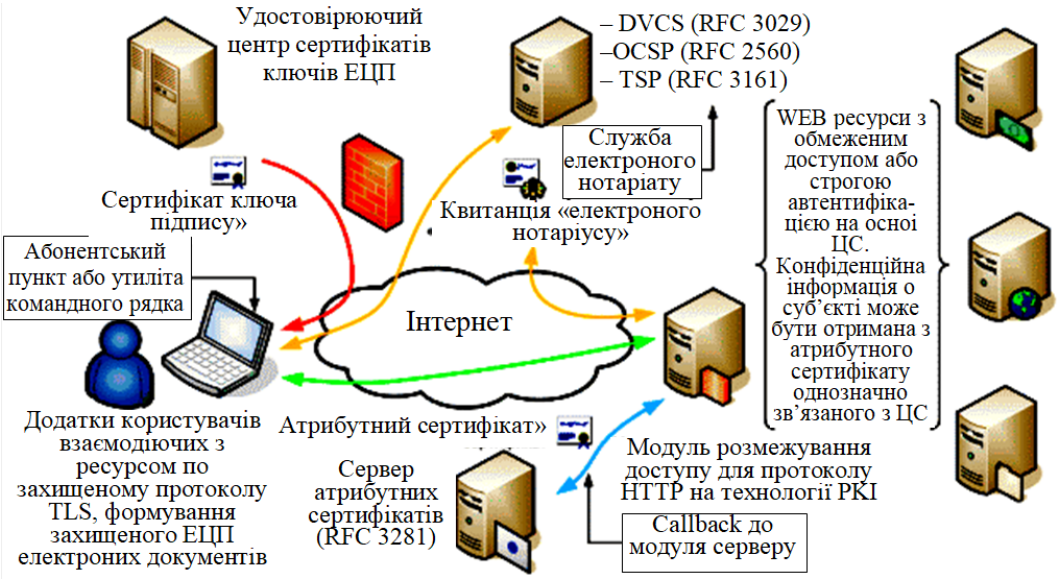

Рисунок 2 – Структурна схема

Розглянемо більш детально складові структурної схеми, та принцип роботи системи.

Центр, що засвідчує, сертифікатів ключів підпису

Центр, що засвідчує, сертифікатів ключів підпису (удостовірюючий центр – УЦ) є повністю вітчизняною розробкою:

– відповідає вимогам Доктрини інформаційної безпеки в плані заміщення імпортних технічних і програмних засобів у українських інформаційних системах;

– завдяки відомості всіх криптографічних процедур в ізольований криптографічний PKSC#11 токен, в УЦ можуть використовуватися як вітчизняні криптографічні алгоритми, так і закордонні;

– використовувані вітчизняні криптографічні механізми й протоколи опираються на інтернет драфти, розроблені вітчизняними компаніями, а отже, рішення сумісні з рішеннями інших вітчизняних виробників;

– технічна реалізація заснована на сучасних керівних документах, стандартах і міжнародних рекомендаціях, що дозволяє:

1) в одному технічному рішенні підтримувати невиразно велике число зовсім ізольованих, у тому числі з різними криптографічними алгоритмами, видавців;

2) підтримуються механізми кросування видавців (аж до рівня мостового УЦ), у тому числі й зовнішніх, для утворення єдиних зон обігу захищених документів;

3) для систем наближених до On-line передбачене поширення відновлень списків відкликаних сертифікатів (delta CRL);

4) компоненти самого УЦ для побудови ланцюжків сертифікації використовують внутрішній сервіс OCSP, що може бути оформлений як корпоративний зі складу служби "електронного нотаріату";

5) до складу системи входить власна служба роздачі міток часу, з механізмами підстроювання під зовнішні еталони;

6) відповідно до RFC 3039 Internet X.509 Public Key Infrastructure. Qualified Certificate Profile у сертифікат може бути введений серійний номер імені, тим самим вирішена "колізія імен" – проблема «однофамільців»;

7) реєстр крім самого сертифіката користувача може містити додаткову інформацію про суб'єкта, включаючи графічні елементи (фотографії, відбитки пальців і т.п.);

8) всі інформаційні блоки при транспортуванні й зберіганні захищені електронними цифровими підписами, що забезпечує цілісність, авторство й невідрікаємість і спрощує процедури розбору конфліктних ситуацій.

Центр, що засвідчує, сертифікатів ключів підпису (УЦ) є основою комп'ютерних систем захищеного документообігу на технології відкритого розподілу ключів (Public Key Infrastructure (PKI)). Технічна реалізація Центра, що засвідчує, відповідає вимогам Закону України "Про електронний цифровий підпис" за умови використання в ЦУ сертифікованих ДССЗЗІ засобів електронного цифрового підпису. УЦ, може виступати як ключовий компонентом для різного типу прикладних захищених систем корпоративного рівня (захищений документообіг, Інтернет-банкінг, білінгові системи, електронна комерція (B2C, B2B), Інтернет процесінг і.т.п.).

Сертифікат X.509 v3

Сертифікат X.509 v3 визначається в такий спосіб. Для обчислення підпису дані, які повинні бути підписані, представляються з використанням ASN.1 однозначних правил подання (DER).

Поля сертифіката. Сертифікат є послідовність трьох обов'язкових полів: tbsCertificate, signatureAlgorithm і signatureValue.

−tbsCertificate. Поле містить імена суб'єкта й випускаючого, відкритий ключ, пов'язаний із суб'єктом, період дійсності й іншу пов'язану із цим сертифікатом інформацію. Поля докладно описані далі; tbsCertificate звичайно включають розширення, які теж будуть описані нижче.

−signatureAlgorithm. Поле signatureAlgorithm містить ідентифікатор криптографічного алгоритму, використовуваного УЦ для підписування даного сертифіката. Існують стандартні алгоритми, які повинні підтримуватися всіма реалізаціями, але конкретна реалізація може підтримувати й інші алгоритми.

Ідентифікатор алгоритму визначається наступної ASN.1-структурою:

```
AlgorithmIdentifier ::= SEQUENCE  {
  algorithm  OBJECT IDENTIFIER,
  parameters ANY DEFINED BY algorithm OPTIONAL
}
```

Ідентифікатор алгоритму використовується для визначення криптографічного алгоритму. Компонент OBJECT IDENTIFIER ідентифікує алгоритм (такий як DSA з SHA-1). Компоненти поля параметрів змінюються відповідно до зазначеного алгоритму. Поле повинне містити той же самий ідентифікатор алгоритму, що й поле підпису в tbsCertificate.

−signatureValue. Поле signatureValue містить цифровий підпис, обчислений для поля tbsCertificate, записаному в DER-поданні ASN.1. Це означає, що поле tbsCertificate, представлене як ASN.1 DER, використовується як вхід у функцію підпису. Отримане значення підпису представлене як BIT STRING і включено в поле підпису. Деталі даного процесу можуть відрізнятися для кожного конкретного алгоритму підпису. Створенням даного підпису УЦ підтверджує дійсність інформації в поле tbsCertificate. Зокрема, УЦ підтверджує зв'язок між матеріалом відкритого ключа й суб'єктом сертифіката.

−TBSCertificate. Послідовність TBSCertificate містить інформацію, пов'язану із суб'єктом сертифіката й УЦ, що випустив сертифікат. Кожний TBSCertificate містить імена суб'єкта й випускаючого, відкритий ключ, пов'язаний із суб'єктом, період дійсності, номер версії й серійний номер сертифіката; деякі поля можуть (але це не обов'язково) містити унікальний ідентифікатор. Розглянемо синтаксис і семантику таких полів. TBSCertificate звичайно включає розширення. Розглянемо також найбільше часто використовувані в Internet розширення.

−Version. Дане поле описує версію подання сертифіката. Якщо використовуються розширення, то версія повинна бути 3 (значення – 2). Якщо розширення не зазначені, але UniqueIdentifier представлений, версія може бути 2 (значення – 1); але версія може бути й 3. Якщо представлені тільки базові поля, версія може бути 1 (значення в сертифікаті опущене як значення за замовчуванням); але версія може бути 2 або 3. Реалізації повинні бути готові приймати будь-яку версію сертифіката. Як мінімум конформні реалізації повинні розпізнавати версію 3 сертифікатів.

–Serial number. Серійний номер повинен бути позитивним цілим, призначуваним УЦ для кожного сертифіката. Він повинен бути унікальним для кожного сертифіката, випущеного даним УЦ. Таким чином, ім'я що выпустили й серійний номер однозначно визначають сертифікат. Cas повинні забезпечувати, щоб серійні номери були ненегативними цілими. Уважається, що серійні номери можуть мати довжину до 20 октетів.

–Signature. Дане поле містить ідентифікатор алгоритму, використовуваного УЦ для підписування сертифіката.

–Дане поле повинне містити той же самий ідентифікатор алгоритму, що й поле signatureAlgorithm в Certificate. Зміст необов'язкового поля параметрів залежить від конкретного алгоритму.

–Issuer. Поле issuer ідентифікує того, хто підписав і випустив сертифікат. Поле issuer повинне містити непусте унікальне ім'я (DN). Ім'я визначається у відповідності з наступної ASN.1 структурою:

```
Name ::= CHOICE { RDNSequence }
RDNSequence ::= SEQUENCE OF
   RelativeDistinguishedName
RelativeDistinguishedName ::=
   SET OF AttributeTypeAndValue
AttributeTypeAndValue ::= SEQUENCE {
   type   AttributeType,
   value  AttributeValue
}
AttributeType ::= OBJECT IDENTIFIER
AttributeValue ::= ANY DEFINED BY
   AttributeType
```

Ім'я описує ієрархічне ім'я, що складається з атрибутів, таких, наприклад, як назва країни, і відповідних значень, таких як RU. Тип компонента AttributeValue визначається значенням AttributeType. Стандарт X.509 не обмежує набір типів атрибутів, які можуть з'явитися в ім'ї. Проте, стандартом рекомендується підтримувати наступні типи атрибутів в іменах випускаючі й суб'єкта:

- Країна.
- Організація.
- Організаційна одиниця.
- Позначення унікального імені.
- Назва штату або регіону.
- Загальноприйняте ім'я (наприклад, Іванов Іван).
- Серійний номер.

Додатково можуть бути присутнім деякі інші типи атрибутів в іменах випускаючі й суб'єкта, наприклад:

- Локалізація.
- Заголовок.
- По батькові.
- Призначене ім'я.
- Ініціали.
- Псевдонім.
- Спеціальна назва (наприклад, "Jr.", " 3-ій" або "IV").

Також може бути присутнім атрибут domainComponent. DNS надає собою ієрархічну систему позначення ресурсів. Даний атрибут надає зручний механізм для організацій, які хочуть використовувати унікальні DN імена паралельно зі своїми DNS-Іменами. Це не заміняє dNSName компонент альтернативного поля ім'я. Стандарт не вимагає конвертувати такі імена в DNS-Імена. Сторона, що перевіряє, повинна обробляти поля унікального ім'я

випускаючого й унікального ім'я суб'єкта для одержання ланцюжка імен при перевірці *дійсності сертифікаційного* шляху. Ланцюжок імен виходить у випадку відповідності унікального ім'я випускаючого в першому сертифікаті ім'я суб'єкта в сертифікаті УЦ.

Служба "Електронного нотаріату"

"Електронний нотаріус" (ЕН) може бути як додатковим сервісом в комп'ютерній системі, що виконує функції Центра, що засвідчує (УЦ) сертифікатів ключів підпису в якості стандартизованогоRFC 3029 Internet X.509 Public Key Infrastructure Data Validation and Certification Server Protocols (DVCS). RFC 2560 Online Certificate Status Protocol – OCSP. RFC 3161 Time-Stamp Protocol (TSP)) технічного рішення «Про Електронний цифровий підпис», так і як самостійний програмний комплекс, і реалізовувати разову або абонентську послугу з перевірки й сертифікації інформації, перевірки сертифікатів і виробленню квитанції, що містять «штамп» часу. В ЕН зведені служби, технічна реалізація яких стандартизована міжнародними рекомендаціями, по фактах усіляких перевірок, підтверджень і вироблення «штампа часу» для зовнішніх компонентів інфраструктури відкритих ключів.

"Електронний нотаріус" виконує наступні функції:

–Посвідчення факту володіння інформацією з або без її подання сервісу.

–Перевірка дійсності ЕЦП.

–Перевірка дійсності сертифіката відкритого ключа (для компонентів DVCS або OCSP).

–Вироблення квитанції, що містить «штамп» часу (TSP).

Задачі, у рішенні яких, може бути використана Служба "Електронного нотаріату":

1. Створення єдиного домена захищеного електронного документообігу, у тому числі побудованому на несумісних між собою засобах криптографічного захисту інформації, але маючих сертифікат ДССЗЗІ на засоби криптографічного захисту інформації для гетерогенних програмно-апаратних платформ.

2. Одержання штампа «дійсного часу для даної PKI системи» на завіреному електронному документі. Досить важливо (для попередження шахрайських дій або колізій) при завіренні електронного документа коректно вказувати дату підписання, однак проставляння дійсної дати цілком є відповідальністю підписується стороны, що. ЕН у цьому випадку є «третьою» стороною – довіреному арбітром, що фіксує факт наявності дійсної ЕЦП на конкретний момент часу. Даний сервіс іноді називають Time Stamping. Наявність «третьої» незалежної сторони може виявитися корисною щоб зафіксувати певний етап у технологічному ланцюжку документообігу (якийсь документ пройшов яку те стадію свого формування), наприклад, на конкретний момент часу податкова декларація завірена й доставлена від платника податків в інспекцію. У більше широкому змісті ЕН може бути використаний абстрактною прикладною системою як джерело TSP міток «еталонного часу».

3. Тривале архівне зберігання електронних документів. ЕЦП на електронному документі має «строк життя», що, зокрема визначається «строком життя» персонального сертифіката, який бере участь у формуванні ЕЦП. Через багато причин цей час досить обмежений, що не дозволяє будувати повноцінну систему документообігу, включаючи такий важливий компонент як архівне зберігання. Наявність DVC квитанцій по перевірці ЕЦП дозволяє робити висновки про дійсність ЕЦП уже після витікання часу дійсності сертифіката, що брав участь у виробленні даної ЕЦП. Дана властивість пояснюється тим, що сертифікат ЕН (DVCS), яким завірена квитанція, більш тривала й існують механізми пролонгації квитанцій (випуск квитанції на квитанцію).

4. Організація перевірки ЕЦП «третьою» стороною для користувачів дозволяє перевести сам факт перевірки із площини криптографічних обчислень на сертифікованих серверах захисту інформації в площину організації довіреної доставки квитанцій із сервера ЕН, що в багатьох випадках значно технологичніше. Ступінь «доручення» доставки квитанцій не регламентується законом «Про ЕЦП» і цілком визначається специфікою комп'ютерної системи, у якій використовуються завірені документи. Способів доставки досить багато: від доставки квитанції кур'єрською службою, підтвердженням по телефоні,

порівнянням самих файлів – квитанцій отриманих по мережі й з репозиторія ЕН (DVCS), до організації захищених сегментів мережі, на підтвердження що мають, наприклад, атестат ДССЗЗІ.

5. Покладання на сервіс ЕН функцію перевірки дійсності якогось цифрового сертифіката істотно спрощує комп'ютерну систему, у якій циркулюють завірені електронні документи. Сама по собі процедура перевірки сертифіката досить трудомістка, необхідно побудувати ланцюжок перевірки кінцевого сертифіката з перевіркою всіх проміжних кореневих сертифікатів, визначити місце розповсюдження, одержати й обробити списки відкликаних сертифікатів і т.п.

6. Даний сервіс може бути досить корисний для комп'ютерних систем (КС), у яких використовується факт володіння користувачем якоюсь інформацією без її опублікування. Наприклад, КС проведення різних тендерів, регламент яких визначає, що до певного строку ніхто не повинен мати доступ до конкурсного матеріалу (за винятком коротких анотацій) і тільки по настанню часу початку конкурсу «конверти» повинні бути розкриті. Для таких систем учасники представляють квитанції на істинність ЕЦП конкурсного матеріалу без фактичної передачі самого матеріалу до моменту настання конкурсу. Істинність представленого в наслідку матеріалу підтверджено ЕЦП зі складу DVC квитанції. У цьому випадку захист конкурсного матеріалу покладає на самих конкурсантів – самих зацікавлених у захисті даних осіб і повністю знімає ризик шахрайства в системі.

Служба атрибутування (реалізація заснована на RFC 3281) вирішує дві задачі:

– Дозволяє здійснити криптографічний зв'язок сертифіката ключа підпису з додатковою інформацією, захищеної ЕЦП, що визначає роль власника сертифіката в КС, наприклад, для цілей розмежування доступу, розміщення персональної інформації, розміщення інформації уточнюючого повноваження й т.п.

– Дозволяє здійснити криптографічний зв'язок між абстрактним блоком даних і додатковою інформацією (метаданими), наприклад, такий атрибутний контейнер можна асоціювати з електронним документом разом з метаданими при міжсистемному (міжвідомчому) інформаційному обміні. Додатково, використовувана технологія дозволяє (ЕЦП у вигляді CMS або PKCS#7, або «підпис із розширеними даними для перевірки» по ETSI TS 101 733 не може це забезпечити) ввести поняття строку дійсності документа, включаючи механізми екстреного визнання розміщеної в контейнері інформації недійсною. Дана унікальна властивість може бути використана при випуску електронних дозволів, наприклад, імпортно-карантинні дозволи, ліцензії (у тому числі й на програмне забезпечення) і т.п.

Висновки. У статті наведені теоретичне узагальнення й рішення наукового завдання дослідження методів доступу до хмарних сервісів з використанням технології PKI. Рішення даного завдання полягало у вирішенні наступних задач: Був проведений огляд існуючих систем доступу до хмарних сервісів з використанням технології PKI; Досліджена система доступу до хмарних сервісів з використанням технології PKI; На основі отриманих результатів досліджень створена програмна реалізація системи доступу до хмарних сервісів з використанням технології PKI; Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання доступу до хмарних сервісів з використанням технології PKI; Проведено аналіз предметної галузі в ході якого були виявлені об'єкти, взаємодія яких носить істотний характер для функціональної діяльності предметної галузі, і їхні основні характеристики; побудована алгоритм і вибраний середовище розробки.

Список літератури

1. О.А. Смірнов, П.С. Усік, «дослідження перспектив використання технологічних рішень в мережах 5g» у Кібербезпека та інформаційні технології: монографія. – Х. : ТОВ «ДІСА ПЛЮС», 2020.С. 122-135.

2. Смірнов О.А., Дрєєва Г.М., «Метод генерування фрактального трафіку за допомогою моделі генератора на графі» у Інформаційна безпека та інформаційні технології: монографія / за заг. ред. В. С. Пономаренка. – Х. : Вид. Рожко С.Г. 2019. С. 123-139.
3. Смирнов А.А., Коваленко А.В. Комплекс математических моделей технологии тестирования web-приложений. Інформаційні технології: сучасний стан та перспективи: монографія / За загальною редакцією В.С. Пономаренка. – Х.: ТОВ «ДІСА ПЛЮС», 2018. – 461 с.
4. Смирнов А.А., Коваленко А.В. Разработка метода управления рисками разработки программного обеспечения. Інформаційні технології: проблеми та перспективи: монографія / За загальною редакцією В.С. Пономаренка. – Х.: Видавець Рожко С.Г., 2017. – 447 с.
5. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Смірнов С.А., Поліщук Л.І., «Дослідження стійкості до диференціального криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» Системи управління, навігації та зв'язку, 2022, № 3(69). С. 93-98. 2022.
6. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Поліщук Л.І., Смірнов С.А. «Дослідження статистичної стійкості та швидкісних характеристик запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» Вісник Хмельницького національного університету. Серія: «Технічні науки», № 2 (307). С. 46-52. 2022.
7. Смірнов О.А., Смірнова Т.В., Константинова Л.В., Смірнов С.А., Якименко Н.М., «Дослідження стійкості до лінійного криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» Системи управління, навігації та зв'язку, 2022, № 1(67). С. 84-89.
8. Смірнов О.А., Смірнова Т.В., Буравченко К.О., Кравченко С.С., Горбов В.О., «Хмарна система підтримки прийняття рішень технологічного процесу відновлення поверхонь конструкцій і деталей машин». Сучасні інформаційні системи. 2021. Т. 5, № 4. С. 79-95
9. Смирнов А., Кузнецов А., Кузнецова Т. «Шумоподобные дискретные сигналы для асинхронных систем кодового разделения радиоканалов». Радиотехника, № 2(205), 175–183. 2021.
10. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New Technique for Hiding Data in Cover Images Using Adaptively Generated Pseudorandom Sequences». CEUR Workshop Proceedings Volume 2732, 2020, Pages 214-227.
11. Smirnov, O., Neskorodieva, T., Fedorov, E., Rudakov, K., Neskorodieva, A. «Method Detection Audit Data Anomalies on Basis Restricted Cauchy Machine» CEUR Workshop Proceedings, Volume 3187, 2022, pp. 1-12. (Scopus).
12. Smirnov O., Smirnova T., Anas M. Al-Oraiqat, Drieiev O., Polishchuk L., Sheroz Khan, Yassin M. Y. Hasan, Aladdein M. Amro, Hazim S. AlRawashdeh «Method for Determining Treated Metal Surface Quality Using Computer Vision Technology». Sensors (Basel, Switzerland) Volume 22, Issue 16, 6223, 2022. (Scopus).
13. Smirnov, O., Lakhno, V., Akhmetov, B., Chubaievskyi, V., Khorolska, K., Bebeshko, B. «Selection of a Rational Composition of İnformation Protection Means Using a Genetic Algorithm». In: Rajakumar, G., Du, KL., Vuppalapati, C., Beligiannis, G.N. (eds) Intelligent Communication Technologies and Virtual Mobile Networks. Lecture Notes on Data Engineering and Communications Technologies, vol 131. 2023. Springer, Singapore. pp. 21-34. (Scopus).
14. Smirnov O.A., Al-Oraiqat A.M., Ulichev O.S., Meleshko Ye.V., Al-Rawashdeh H.S., Polishchuk L.I. «Modeling strategies for information influence dissemination in social networks». Journal of Ambient Intelligence and Humanized Computing Volume 13, Issue 5. Springer, Cham. 2022, pp. 2463-2477. (Scopus).
15. Smirnov O., Kuznetsov A., Kryvinska N., Kiian A., Kuznetsova K. «Full Non-Binary Constant-Weight Codes». SN Computer Science, Vol 2, 337, 2021. https://doi.org/10.1007/s42979-021-00739-w (Scopus).
16. Smirnov O., Kovalenko O., Kovalenko A., Kavun S. «Quantitative Risk Assessment Method Development in the Context of the SDLC-model». 2021 IEEE 8th International Conference on Problems of Infocommunications, Science and Technology (PIC S&T), 2021, pp. 203-208, doi: 10.1109/PICST54195.2021.9772143 (Scopus).
17. Smirnov O., Neskorodieva T., Fedorov E., Rymar P. «Neural Network Modeling Method of Transformations Data of Audit Production with Returnable Waste». CEUR Workshop Proceedings Volume 3101, 2021, Pages 192-207. (Scopus).
18. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova K. «Data hiding scheme based on spread sequence addressing». CEUR Workshop Proceedings Volume 2805, 2020, Pages 44-58. (Scopus).
19. Smirnov, O., Kuznetsov, A., Potii, O., Poluyanenko, N., Stelnyk, I., Mialkovsky, D. «Combining and filtering functions in the framework of nonlinear-feedback shift register». International Journal of Computing; 2020, Volume 19, Issue 2 – Research Institute for Intelligent Computer Systems – 2020. – P. 247-256. (Scopus).
20. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova T. «Non-binary constant weight coding technique». CEUR Workshop Proceedings. Volume 2740, 2020, Pages 102-114. (Scopus).