РОЗДІЛ ІІ. КОМП'ЮТЕРНІ НАУКИ

УДК 004.75
DOI https://doi.org/10.26661/2413-6549-2020-1-17

РЕАЛІЗАЦІЯ ЗАПЕРЕЧУВАНОГО ШИФРУВАННЯ ДАНИХ НА БАЗІ РОЗПОДІЛЕНИХ ОБЧИСЛЕНЬ

Гальченко А. В.
аспірант кафедри програмної інженерії
Запорізький національний університет
вул. Жуковського, 66, Запоріжжя, Україна
orcid.org/0000-0002-2258-9755
andream1993@ukr.net

Чопоров С. В.
доктор технічних наук, доцент,
професор кафедри програмної інженерії
Запорізький національний університет
вул. Жуковського, 66, Запоріжжя, Україна
orcid.org/0000-0001-5932-952X

Ключові слова: *балансування навантаження, віртуальні машини, витік даних, заперечуване шифрування, несанкціонований доступ, обчислення, примушування, продуктивність, розподілені системи, шифрування, часові характеристики.*

Інформаційний простір розширюється досить швидко. Щодня користувачі створюють та обробляють велику кількість даних із використанням автоматизованих систем і мобільних пристроїв. Тобто контроль за поширенням даних, не кажучи про їх захист, стає все складнішим. Саме цим користуються треті особи (зловмисники). Вони отримують доступ до різноманітної інформації та використовують її в особистих цілях. Досить часто їх вигода є наслідком збитків, які несуть власники вказаної інформації. Для захисту інформації в більшості випадків використовують криптографічні засоби захисту. Їх застосування дає змогу запобігти витоку даних у разі отримання зловмисником доступу до інформаційної системи. Однак на практиці третіми особами використовуються різноманітні способи несанкціонованого доступу як до інформації, так і до ключів для захисту даних. Оскільки засоби криптографічного перетворення інформації поступово втрачають свою надійність, розроблення новітніх методів інформаційної безпеки є перспективним напрямом досліджень. Одним із них є розроблення механізмів заперечуваного шифрування даних. Алгоритми перетворення інформації на їх основі дають змогу захистити дані від витоку протягом усього часу їх існування. Також використання зазначених алгоритмів дає можливість зменшити ймовірність застосування атак на основі примусу користувачів інформаційних систем. Разом із тим установлено, що, незважаючи на велике різноманіття рішень, вищевказані механізми досі не використовуються на практиці. Основною причиною цього є правові й технічні аспекти, які полягають у необхідності використання значних обчислювальних ресурсів. Головною метою дослідження є створення моделі розподілених обчислень, яка буде використана для покращення ефективності механізмів заперечуваного шифрування. Об'єктом дослідження є сучасні архітектури, моделі та принципи побудови розподілених систем обробки даних, що дають змогу об'єднати для сумісної роботи мільйони комп'ютерів по всьому світу. Предметом дослідження є алгоритми заперечуваного шифрування та їх реалізації на базі розподілених обчислень. За результатами дослідження авторами запропоновано декілька схем розподілених систем залежно від

технічних можливостей користувачів і методів розподілу задач. Ефективність вказаних схем перевірена в ході обчислювальних експериментів, результатом яких є покращення часових характеристик алгоритмів заперечуваного шифрування даних. Проведене дослідження дало змогу на практиці підвищити продуктивність виконання алгоритмів заперечуваного шифрування за допомогою наявних способів і схем реалізації розподіленої обробки даних між групою мережевих комп'ютерів. Наукова новизна запропонованого підходу полягає в застосуванні засобів розподіленої обробки даних для реалізації механізмів заперечуваного шифрування даних. Практична значимість результатів полягає в отриманні підвищення продуктивності для наявних алгоритмів заперечуваного шифрування, що в перспективі дасть змогу їх практичного застосування.

THE DENIABLE ENCRYPTION BASED ON THE DISTRIBUTED CALCULATIONS

Halchenko A. V.
Postgraduate Student at the Department of Software Engineering
Zaporizhzhia National University
Zhukovskogo str., 66, Zaporizhzhia, Ukraine
orcid.org/0000-0002-2258-9755
andream1993@ukr.net

Choporov S. V.
Doctor of Sciences in Engineering, Associate Professor,
Professor at the Department of Software Engineering
Zaporizhzhia National University
Zhukovskogo str., 66, Zaporizhzhia, Ukraine
orcid.org/0000-0001-5932-952X
s.choporoff@znu.edu.ua

Key words: *coercion, computation, data leakage, denied encryption, distributed systems, encryption, load balancing, performance, time characteristics, unauthorized access, virtual machines.*

The cyberspace expands fast. A plenty of information is being created and processed by information systems and users' devices every day. The data spread control becomes more difficult. It's the best time to steal, collect and make a profit from users sensitive data. A lot of hacker techniques are used by them. The users are low skills and not ready to protect their data. General, the cryptographic security methods are the most used for sensitive data protection. It protects data from unauthorized access. But there is not any way to protect data from leaking. Also, the cryptography keys information is the another profit for intruders. Its allows them to use other systems for spy, information systems recognizing, use and sale their resources for another users. Also, most information security rules are broken by inside users. That's why the nowadays information security tools based on the cryptography systems lose its reliability. The innovated information security tools development is the top priority. It has to increase information systems protection independent of users' behavior. The deniable encryption schemes is one of the available ways. Its protection is based on the public key cryptography schemes. They prevent the data leaking inside and outside of information systems. It also prevents from the information security policy breaking by users. But they are not applied, because of their low productivity. It confines their applying on practice. The adaptive models' development based on the distributed computing for the deniable encryption schemes practice applying is the main investigation purpose. The modern architectures and models of the distributed data processing systems is the object of this research. The productivity increasing of deniable encryption algorithms by the distributed systems' implementation is the subject of this paper. As a result, the several effective models based on the distributed systems have been implemented into the low productivity deniable encryption algorithm and been tested. The effective load balance method has been found. The highest productivity has been reached. The base deniable encryption algorithm has not been modified. The experiment results have been compared with the other authors' investigations. The development of the effective distributed data processing model and it's applying in the deniable encryption algorithm is the scientific novelty of the proposed approach. The productivity increasing of the deniable encryption algorithm is the practice result.

Вступ. Сучасне інформаційне поле досягло небачених розмірів, не лише дані про навколишнє середовище потребують систематичної обробки і зберігання. За даними [1], до 2020 року розмір інформаційного простору досягне 40 ЗБ(без урахування даних на пристроях). Для обробки цієї інформації створено велику кількість алгоритмів, які застосовуються для вирішення різноманітних завдань. Попри те що персональні комп'ютери, мобільні пристрої, промислові й мережеві автоматизовані системи обробки даних набули значних обчислювальних потужностей, їх кількість недостатня для обробки, накопичення та захисту вищевказаної інформації. Для розв'язання цієї проблеми створили й активно використовують розподілені системи обробки даних [2], які не лише дають змогу об'єднати потужності вказаних пристроїв, а й забезпечують балансування навантаження на них для отримання результатів за оптимальний час.

Згідно з твердженнями Ендрю Таненбаума [3], єдиного трактування розподіленої системи не існує. У більшості випадків до них зараховують групи комп'ютерів, у яких відмова одного з них не призводить до зупинки обчислювальної системи, а задачі розподіляються між іншими комп'ютерами. Але вказане не є основним призначенням розподілених обчислень, лише окремою вимогою, яка висувається до них. Більш вузькою і валідною характеристикою вказаних систем є набір незалежних комп'ютерів, які з'єднані каналами зв'язку й утворюють єдине ціле. На тепер існують різні реалізації розподілених систем, які включають поділ обчислень на мережевому, транспортному та прикладному рівнях моделі OSI.

Постановка проблеми. Алгоритми заперечуваного шифрування даних є засобами захисту нового покоління. Їх використання дає можливість забезпечити конфіденційність даних до витоку та після нього. Також вони дають змогу запобігти можливості застосування примусу до користувачів з метою отримання закритих даних. Основна проблема практичного застосування заперечуваного шифрування – це низька швидкість перетворення інформації [4]. З огляду на розмір даних, у сучасних обставинах указане є суттєвим недоліком.

Установлено, що раніше для розв'язання подібних проблем здійснювали шляхом унесення відповідних змін у базові алгоритми перетворення даних, використання симетричних криптографічних систем за рахунок маніпуляцій обчислювальним процесом [4–6].

У результаті розроблено низку власних рішень для підвищення ефективності використання вказаних алгоритмів на локальних пристроях [7; 8]. Отримані результати близькі до сучасних систем криптографічного перетворення даних, однак обчислювальні ресурси локальних пристроїв є обмеженими. Тому подальші дослідження щодо практичного застосування цих алгоритмів на програмному рівні полягають у розподілі обчислень між кількома пристроями, об'єднаними в локальну мережу.

Метою дослідження є розроблення моделі розподілених обчислень для реалізації механізмів заперечуваного шифрування даних. Подібний підхід повинен забезпечити гнучкість управління процедурами перетворення даних алгоритмів заперечуваного шифрування та, як наслідок, покращити їх часові характеристики.

Виклад основного матеріалу.

1. Огляд моделей розподілених обчислень

Під системою, яка реалізує розподіл обробки даних, у науковій літературі розуміють таку, структурні й функціональні частини якої можуть бути розміщені одночасно на декількох вузлах. Незважаючи на відмінності в побудові та складність структури, в основі розподілених систем лежить використання моделі «клієнт-сервер» [2; 3; 6; 9].

Практичне застосування вказаного підходу потребує маніпуляції процесом виконання завдань. У зв'язку з цим виділяють синхронну й асинхронну моделі «клієнт-сервер». Алгоритм роботи першої ґрунтується на очікуванні клієнтом завершення обробки свого запиту на стороні серверу. Остання ж дає клієнту змогу продовжити виконання інших завдань до завершення обробки його запиту на сервері. Разом із тим під час використання цієї моделі важливим завданням є організація взаємодії програмного забезпечення, яке розміщене на вузлах та утворює розподілену систему.

За результатами аналізу вихідної моделі можна виділити взаємодію програмного забезпечення вузлів на таких рівнях:

1) інтерфейс користувача (відповідає за взаємодію оператора з системою);

2) логіка програмного забезпечення (відповідає за валідне виконання закладених у програму алгоритмів);

3) доступ до бази даних (відповідає за управління інформацією на вузлах).

Трапляється, що користувачі, які працюють у неоднорідних системах, потребують доступу до даних в інших систем. У зв'язку з цим обчислювальні системи почали розділяти між групами комп'ютерів з єдиним сервером, який здійснює контроль за доступом інформації на обчислювальних вузлах. До того ж блок логіки програмного забезпечення може бути розміщений як на клієнті, так і на сервері (або розділений між ними).

На рис. 1 наведено приклади сучасної розподіленої системи, у якій кожний вузол виконує окремі

частини вихідного завдання внаслідок розподілу логіки програмного забезпечення між вузлами, та системи на базі багатоланкової архітектури (зі зростанням) «у ширину»:

Особливістю останньої системи є неможливість прямого доступу до інформації з боку користувачів. Вищевказані системи є найбільш поширеними й дають можливість об'єднувати мільйони комп'ютерів, незважаючи на географічне положення.

Незважаючи на відмінності, до всіх розподілених систем висувається низка вимог [3]:

1) відкритість (ґрунтується на використанні загальнодоступних протоколів і стандартів, створенні специфікації для залучення незалежних розробників до створення системи);

2) масштабованість (дає змогу додавати нові вузли для збільшення продуктивності й зменшення навантаження на вихідну систему);

3) підтримка логічної цілісності даних (гарантує валідність і повноту обробки даних);

4) стійкість (забезпечує дублювання модулів з метою перерозподілу завдань між вузлами);

5) безпека (гарантує можливість доступу до системи лише авторизованих вузлів і користувачів, захист тимчасових даних від модифікації та витоку);

6) ефективність (ґрунтується на мінімізації витрат від реалізації системи).

Вищевказані положення є обов'язковими до виконання при побудові розподілених систем. Незважаючи на це, в окремих випадках дозволяється знехтувати виконанням деяких вимог на користь продуктивності системи. Прикладом указаного є ієрархічна мережа DNS-серверів, у якій розробники знехтували вимогами безпеки.

2. Огляд методів балансування навантаження

З метою отримання кращих показників продуктивності внаслідок масштабування системи виникає необхідність використовувати алгоритми балансування навантаження (управління розподілом завдань між вузлами) [6; 9].

Узагальненим прикладом їх роботи є розподіл завдань між вузлами, які натепер найменш завантажені. Такий підхід до організації обчислювального процесу дає можливість збільшити продуктивність обчислень, регулювати навантаження на вузлах і підвищити стійкість системи до відмов загалом.

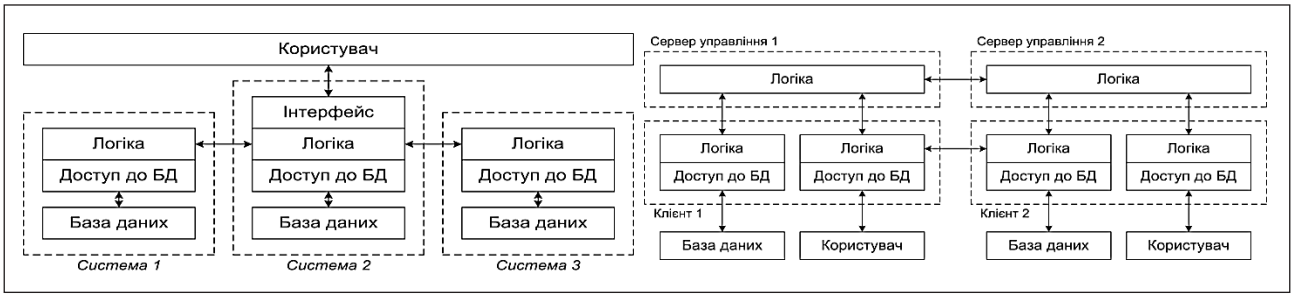
Для досягнення кращих показників продуктивності обчислень на етапі проектування розподілених систем розробники реалізують механізми балансування навантаження на мережевому, транспортному та прикладному рівнях моделі OSI.

Балансування навантаження на мережевому рівні ґрунтується на використанні набору фізичних серверів, які забезпечують стійкість системи до відмов шляхом їх дублювання процедур між групою вузлів. Указаний підхід потребує значних фінансових витрат, але забезпечує високі показники ефективності системи. Технологія балансування навантаження на цьому рівні реалізує управління розподілом завдань за рахунок вирівнювання по DNS, NLB-кластеру та IP-адрес.

На транспортному рівні поділ навантаження між групою вузлів реалізовано за рахунок застосування певного алгоритму (пошук серверів по колу або пошук найменш завантаженого серверу), який виконується за окремим вузлом (подібний до проксі-серверу). Подібний підхід є більш простим та ефективним, а також перешкоджає прямий доступ користувачів до серверу шляхом комутації запитів і відповідей між ними.

Технологія розподілу завдань на прикладному рівні досить подібна до попередньої. Однак у її основі лежить використання механізмів аналізу контексту запитів. Завдяки такому підходу до вирішення завдання запити клієнтів перенаправляються виключно на ті вузли, які відповідають за виконання певних завдань.

Вищевказані підходи до регулювання навантаження на мережеві вузли реалізовані у вигляді алгоритмів балансування навантаження [6]. Для досягнення кращих показників продуктивності необхідно обрати відповідний алгоритм згідно з такими вимогами:



а б

Рис. 1. Приклад сучасних системи обробки даних:
а – розподілена система; б – мережа прямого обміну даними

1) справедливість (ґрунтується на динамічній оцінці завантаження вузлів і використовується для прогнозування піків навантаження);

2) раціональність (дає змогу досягти максимально рівномірного поділу навантаження між вузлами);

3) швидкість (гарантує безперешкодну та швидку обробку даних).

У таблиці 1 наведено порівняльну характеристику відомих алгоритмів балансування навантаження щодо виконання вищевказаних критеріїв вибору.

3. Тестові моделі обчислень

У роботі автори дослідили можливість використання вищевказаних положень і виконали їх імплементацію у вихідну модель обчислень. Указане дало змогу реалізувати поділ обчислень тестового алгоритму заперечуваного шифрування даних між групою вузлів. З метою перевірки роботоспроможності цього підходу та пошуку оптимального варіанта організації обчислень синтезовані такі моделі розподілених обчислень (рис. 2): «клієнт-клієнт», «клієнт-сервер», «клієнт-сервер-клієнт».

Таблиця 1

Аналіз методів балансування навантаження

№ з/п	Критерії порівняння	Методи балансування навантаження			
		Round Robin	**Weighted Round Robin**	**Least Connections**	**Sticky Sessions**
1	Справедливість	Черга запитів			
2	Раціональність	Послідовний розподіл запитів між вузлами	Пріоритет мають більш потужні вузли	Пріоритет у вузла з меншим навантаженням	Послідовний розподіл для визначених IP-адрес
3	Швидкість	Залежить від продуктивності вузлів	Залежить від кількості потужних вузлів	Ефективна швидкість при використанні однакових вузлів	Витрати на зміну основного вузла

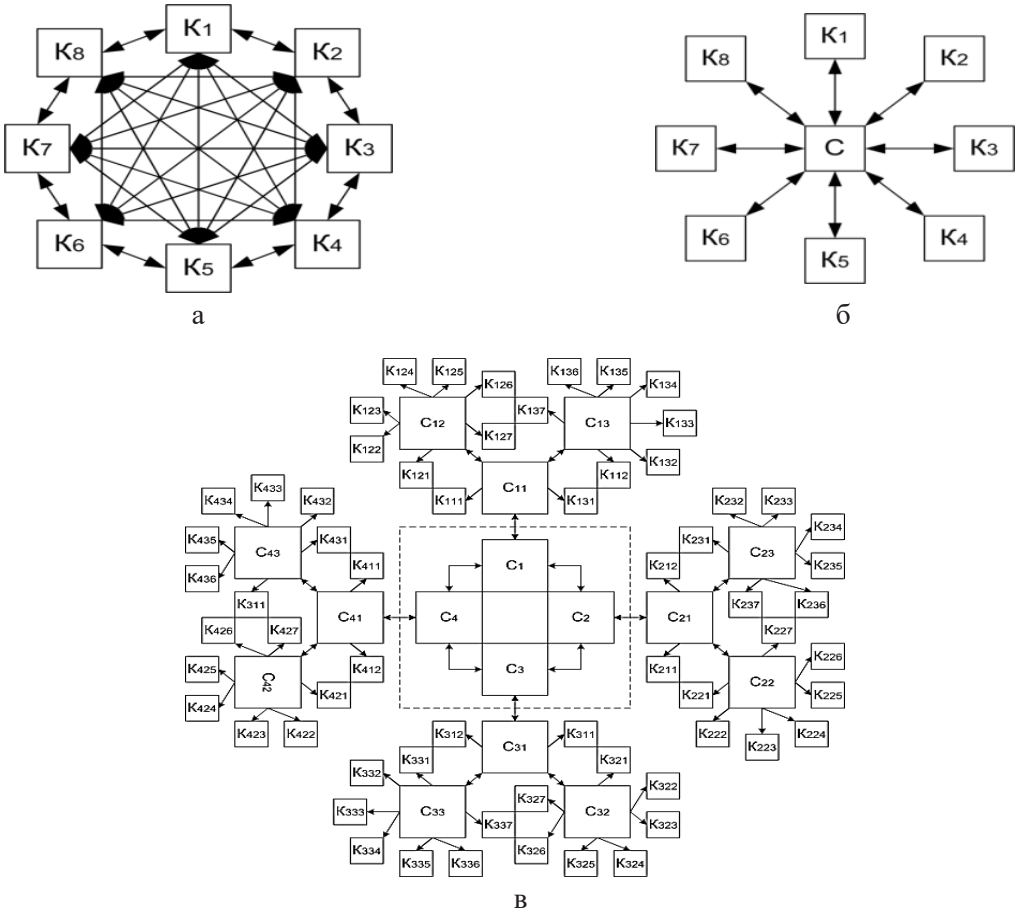


Рис. 2. Схеми розподілу обчислень:
а – «клієнт-клієнт», б – «клієнт-сервер», в – «клієнт-сервер-клієнт»

Модель, представлена на рис. 2а, ґрунтується на обміні даними виключно між клієнтами. Вузол, який ініціює обробку даних, виконує функції управління обчислювальним процесом. Реалізація вказаної моделі дає змогу зменшити витрати на обмін даними між вузлами, що можуть вплинути на кінцевий час обробки даних.

Обробка даних згідно з моделлю «клієнт-сервер» (рис. 2б) потребує виділення окремого вузла, який буде виконувати функції обробки даних або перерозподіл запитів клієнта між групою обчислювальних серверів. Указана модель є досить ефективною при використанні малопотужних вузлів у системі. Однак сучасні комп'ютери здатні виконувати більшу кількість завдань з обробки даних, зокрема виконання важких обчислень з використанням спеціалізованого програмного забезпечення. Отже, використання цієї моделі обчислень є доцільним у разі наявності в системі окремих вузлів із потужними обчислювальними ресурсами.

Реалізація моделі обчислень на рис. 2в доцільна лише в разі можливості масштабування системи «у ширину».

4. Розроблення алгоритму балансування навантаження

З метою підвищення ефективності обчислень авторами запропоновано алгоритм балансування навантаження, який ґрунтується на статичних методах розподілу обчислень (із повним попереднім розподілом між вузлами). Механізм його роботи зводиться до рівномірного розподілу навантаження між усіма обчислювальними вузлами (подібний до Round Robin [6]). В узагальненому вигляді цей алгоритм включає такі кроки:

1) Визначити перелік ***PCL***-вузлів, які є учасниками розподіленої обчислювальної системи на період сканування.

2) Визначити вільні ресурси, якими володіють ***PCL***-вузли на період опитування ($CPUs$, V_{RAM} та V_{HDD}).

3) Виконати поділ даних на базові блоки, розмір яких залежить від кінцевого обчислювального алгоритму обробки даних (алгоритм поділу даних [8] наведений нижче).

4) Виконати потокову відправку/отримання блоків з даними на/з ***PCL***-вузлів. У разі використання динамічного поділу обчислень виконати пакетну відправку даних.

5) У разів появи в системі нових ***PCL***-вузлів або появи додаткових ресурсів на раніше відкинутих вузлах виконуються кроки 2 і 4.

5. Розроблення алгоритмів перетворення даних

Вищевказані рішення застосовані під час побудови обчислювальної моделі (рис. 2а), до складу якої входять такі процедури: пошук клієнтів, поділ, відправка й отримання, шифрування та дешифрування даних, контроль доступу. Попри вказане, залежно від обраної схеми обчислень (рис. 2б, 2в) приведені алгоритми можуть бути використані на всіх вузлах. Це зумовить лише зміщення центру управління обчисленнями.

Алгоритм пошуку вузлів потребує виконання таких кроків:

1) Визначити перелік вузлів підключених до мережі PCL_{act} .

2) Сформувати перелік PCL_{DE} -вузлів шляхом фільтрування PCL_{act} -клієнтів за наявністю встановлених DE -агентів.

3) Виконати запит технічних характеристик PCL_{DE} -вузлів, зокрема інформації про процесори $CPUs$, розмір оперативної пам'яті V_{RAM} і розмір вільного місця накопичувача V_{HDD} .

4) Сформувати перелік PCL -вузлів шляхом фільтрування PCL_{DE} -вузлів за критеріями: $CPUs > 1$, $V_{RAM} \geq 2$ ГБ та $V_{HDD} \geq 20$ ГБ.

5) Сформувати остаточний перелік PCL -вузлів у форматі: $IP \# CPUs \# V_{RAM} \# V_{HDD}$.

На підставі кількості вузлів, які можуть узяти участь в обчисленнях, програма може надати рекомендацію для користувача щодо доцільності вибору однієї з вищевказаних моделей обробки даних.

Поділ даних може бути реалізований як на сервері, так і на клієнтах [8]. Він реалізує підготовку даних для подальшого балансування навантаження в ході обчислювального процесу. Його алгоритм включає кроки:

1) Визначити розміри відкритих і закритих даних ($\|F_M\|$ і $\|F_T\|$), а також різницю між ними $dF_S = \|F_M\| - \|F_T\|$.

2) Визначити розмір блоку з даними згідно з розмірами ключа $\boldsymbol{K}_S$ шифрування та службових даних $\boldsymbol{ADI}_S$ за допомогою виразу $\boldsymbol{B}_S = \boldsymbol{K}_S - \boldsymbol{ADI}_S$.

3) Визначити максимальний розмір вихідних даних за допомогою наступного виразу

$$D_{\max} = f_{\max}\left(\|F_M\|, \|F_T\|\right) + K_S - f_{\max}\left(\|F_M\|, \|F_T\|\right) \left(\mathrm{mod}\, B_S\right).$$

4) Визначити початкову кількість блоків $\boldsymbol{BQ}$ за допомогою розміру блоку даних $\boldsymbol{B}_S$ і відношення

$$1 < BQ \leq \left\lceil \frac{D_{max}}{B_S} \right\rceil .$$

5) Розділити файли з даними F_M та F_T на фрагменти більш меншого розміру

$$FOD_{1...PCL} = \left\{ F_{M\underbrace{\lfloor i...i+1 \rfloor \cdot BQ \cdot B_S}_{PCL}} \; ; \quad F_{T\underbrace{\lfloor i...i+1 \rfloor \cdot BQ \cdot B_S}_{PCL}} \right\}.$$

Процедури інформаційного обміну є базовими для вищевказаних моделей і розподілених обчислень загалом. Вони включають алгоритми відправки даних та отримання відповідних результатів обчислень.

Алгоритм відправки даних на PCL_i -вузли включає такі кроки:

1) Формування переліку PCL -вузлів.

2) Формування захищеного каналу обміну даними між вузлами за допомогою OT_n^1 -протоколу [10], який ґрунтується на використанні схеми RSA.

3) Поділ вихідних даних на PCL -частин.

4) Відправка $\boldsymbol{FOD}_i$-фрагменту вихідних даних і ключової інформації на PCL_i -вузол.

Отримання результатів обчислень (відповідь PCL_i -вузла) включає такі кроки:

1) Формування захищеного каналу обміну даними між вузлами за допомогою OT_n^1 -протоколу, який ґрунтується на використанні схеми RSA.

2) Відправка $\boldsymbol{FOD}_i$-результату обчислень головному вузлу.

Алгоритм відновлення даних передбачає формування тимчасового файлу для зберігання фрагментів інформації. Він виконується виключно на сервері та включає виконання додаткових кроків [8]:

1) Визначити максимальний розмір даних

$$SMO = 2 \cdot k \cdot K_{\mathrm{S}}.$$

2) Визначити максимальну кількість блоків

$$BQ_{max} = \frac{SMO}{2 \cdot BQ}.$$

3) Згрупувати блоки даних і зберегти в окремі файли за допомогою виразу

$$FOD_V = \left\{ C_{V,1\ldots BQ_{max}} \;;\; D_{V,1\ldots BQ_{max}} \right\}.$$

4) Сформувати кінцевий файл з даними $\{F_C;F_D\}$ за допомогою виразу

$$\{F_C;F_D\} = FOD_1 \| FOD_2 \| FOD_3 \| \ldots \| FOD_{PCL}.$$

Залежності від обраного алгоритму заперечуваного шифрування вказаний нижче порядок дій може бути змінений. Приклад адаптованого блокового алгоритму шифрування даних, який ґрунтується на використанні алгоритму заперечуваного шифрування з відкритим ключем розширеної криптографічної схеми Рабіна [4; 7], включає виконання таких кроків:

1) Сформувати два великі прості числа p та q.

2) Сформувати ключі шифрування $N=p \cdot q$ та дешифрування $(p; q)$.

3) Зчитування блоків даних $F_{M_{1\ldots n}}$ та $F_{T_{1\ldots n}}$ і їх попередня обробка.

4) Обчислення випадкових значень

$$R_{i,1\ldots n} : \{r_{i,1}, r_{i,2}, r_{i,3}, \ldots, r_{i,n}\}$$

для блоків закритих даних за допомогою виразу $r_i = (N - T_i)^2 \bmod N$.

5) Обчислення значення блоків шифрограми $C_i =: \{A_i; B_i\}$ за допомогою виразів $A_i = 2 \cdot r_{\mathrm{i}} - M_i (\bmod N)$ та $B_i = r_{\mathrm{i}} \cdot (r_{\mathrm{i}} - M_i) (\bmod N)$.

6) Зберегти отримані значення у файл із шифрограмою F_C.

7) Повторювати виконання кроків 1–4 до завершення обробки блоків.

Дешифрування даних умовно поділено на 2 етапи [4; 7]. Перший із них передбачає відновлення відкритих даних і включає такі кроки:

1) Зчитати блоки даних $F_{C_{1\ldots n}}$. з шифрограми та виконати їх попередню обробку.

2) Обчислити корені рівняння за складним модулем $x_2 + A_i \cdot x - B_i = 0 (modN)$ для кожного блоку шифрограми. Складність його рішення ґрунтується на неможливості швидкого вирішення порівнянь другої степені за простим модулем: $r_i = \sqrt{x_i} (\bmod p)$ та $r_i = \sqrt{x_i} (modq)$ [11].

3) Обчислити варіанти вихідних відкритих даних за допомогою

$$M_{1\ldots n,1\ldots 4} = 2 \cdot A_i - r_{i,1\ldots 4} (modN).$$

4) Виконати зворотну обробку блоків з даними, а результат зберегти у файл із дешифрованими даними F'_M.

5) Повторювати виконання кроків 1–4 до завершення обробки блоків.

Останній алгоритм передбачає дешифрування закритих даних і виконується у тому разі, якщо головний клієнт був авторизованим у системі. Він потребує виконання таких кроків:

1) Дешифрування відкритих даних згідно з алгоритмом, який наведений вище. У результаті отримано корені $R_{i,1\ldots n} : \{r_{i,1}, r_{i,2}, r_{i,3}, \ldots, r_{i,n}\}$.

2) Обчислення коренів порівнянь другої степені за простим модулем: $z_i = \sqrt{r_i} (modp)$ та $z_i = \sqrt{r_i} (modq)$ [11].

3) Обчислення варіантів вихідних даних за допомогою виразу $T_{1\ldots n} = N - z_{i,1\ldots 4} (modN)$.

Алгоритм, який відповідає за контроль доступу, забезпечує виконання вимог безпеки. Він також забезпечує моніторинг активності каналів зв'язку між головним та обчислювальними вузлами (для моделі на рис. 2а). У разі відсутності зв'язку хоча б з одним з обчислювачів дані, які знаходяться на них, повинні бути остаточно знищені. Указаний підхід має забезпечити захист фрагментів з даними в разі втручання зловмисників на один із вузлів. Недоліком цього підходу є можливість зростання часу виконання обчислень або відмови від використання системи загалом (у разі нестабільного зв'язку між вузлами).

Реалізації вищевказаних моделей обчислень мають свої особливості й недоліки. Зокрема, використання систем (рис. 2б і 2в) передбачає появу додаткових витрат часу (затримок) під час обміну даними між клієнтом і сервером. Крім того, у разі використання групи серверів з'являється можливість утрати результатів обчислень через порушення зв'язку між локальними клієнтами й серверами. Попри це, у загальному випадку алгоритм

функціонування серверів однаковий і включає такі кроки:

1) Отримання завдання на обробку даних від PCL_{DE}-клієнта.

2) Сканування мережі на предмет активних PCL_{DE}-клієнтів згідно з вищевказаним алгоритмом пошуку.

3) Завантаження даних від PCL_{DE}-клієнта на сервер.

4) Балансування навантаження за допомогою одного з вищевказаних методів.

5) Відправка завдань на PCL_{DE}-обчислювачі й очікування результатів обчислень.

6) Повернення результатів обчислень PCL_{DE}-обчислювачів на сервер.

7) Закриття каналів зв'язку з PCL_{DE}-обчислювачами, які завершили обробку даних.

8) Повернення оброблених даних на вихідний PCL_{DE}-клієнт.

Попри те яка з вищевказаних схем розподілення обчислень буде застосована, кожний вузол має бути оснащений мінімальним набором програмного забезпечення, що включає мережевий агент (для обміну даними по мережі), блок управління (ядро), блок даних (для організації доступу до бази даних і зберігання тимчасових даних у ній), обчислювальний блок (реалізує логіку обробки даних, яка виконується розподіленою мережею). Водночас алгоритм роботи PCL_{DE}-обчислювача включає такий порядок дій:

1) Очікування запиту на обробку даних від серверу управління.

2) Обробка даних: попередня обробка (підготовка), шифрування, дешифрування даних згідно з вищевказаними алгоритмами. При цьому ефективна обробка даних забезпечується лише з використанням багатопотокових обчислень.

3) Підготовка результатів обчислень і їх відправка на сервер управління.

4) Повторення кроків 1–3 до вимкнення агента або клієнта загалом.

6. Обчислювальні експерименти

З метою апробації запропонованих рішень автори провели експерименти. Разом із тим через обмеженість у кількості технічних засобів заплановано проведення досліджень лише перших двох схем обчислень. Але у зв'язку з тим що вони подібні між собою, за винятком серверу управління обчисленнями, пріоритетним стало дослідження лише першої схеми розподіленої обробки даних. Для оцінювання другої необхідно лише врахувати втрати часу, які з'являються при обміні даними між клієнтами та сервером управління.

Експериментальний стенд включає таке програмне й апаратне забезпечення (рис. 3):

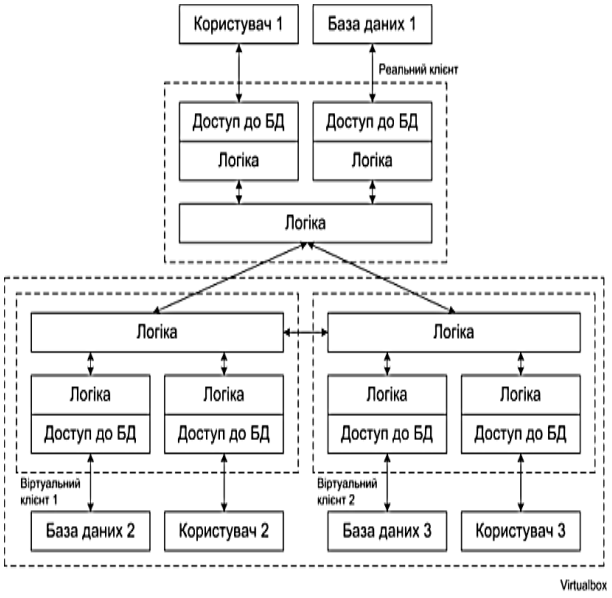


Рис. 3. Структурна схема експериментального стенду

Таблиця 2

Результати обчислювальних експериментів

Критерії	Час шифрування даних, с. (P_ENC)			Час дешифрування даних, с.					
				відкритих (P_DEC)			закритих (S_DEC)		
Кількість вузлів	1	2	3	1	2	3	1	2	3
1	4,71	2,35	1,57	163,30	81,65	54,43	309,98	154,99	103,33
2	4,39	2,19	1,46	162,60	81,30	54,20	321,36	160,68	107,12
3	4,25	2,13	1,42	136,19	68,10	45,40	317,68	158,84	105,89
4	4,06	2,03	1,35	130,24	65,12	43,41	309,36	154,68	103,12
5	4,57	2,29	1,52	150,54	75,27	50,18	265,57	132,79	88,52
6	4,37	2,19	1,46	154,58	77,29	51,53	318,98	159,49	106,33
7	4,23	2,12	1,41	150,96	75,48	50,32	317,91	158,96	105,97
8	4,18	2,09	1,39	151,75	75,88	50,58	309,03	154,52	103,01
9	4,36	2,18	1,45	135,23	67,62	45,08	306,36	153,18	102,12
10	4,27	2,13	1,42	150,18	75,09	50,06	295,74	147,87	98,58

ЦП Intel(R) Core(TM) i5-8250, ОЗП 4 ГБ (по 2 ГБ для віртуальних машин), HDD 100 ГБ (по 40 ГБ для віртуальних машин); ОС Windows 10, середовище програмування PythonIDLE 3.7.3.

Технічним обмеженням указаного експерименту є обмеженість обчислювальних ресурсів. У ролі активного вузла використано реальне робоче місце, а пасивними є 2 віртуальні машини.

Як тестові дані використано фрагменти бази даних юридичних осіб у форматі XML і фрагмент текстового файлу з даними для доступу до особистих кабінетів сервісу GMAIL.

Для виконання обчислень використано ключ шифрування розміром 1024 біти, як найбільш оптимальний з погляду безпеки та продуктивності.

Результати. За результатами проведених експериментів побудовано модель розподіленого шифрування даних, яка реалізує механізми заперечуваного шифрування. Результати експериментів викладено в таблиці 2 та на рис. 4.

Обговорення. Результати експериментів (таблиця 2) демонструють можливість поділу обчислень між групою тестових вузлів без порушення логічної цілісності даних на етапах шифрування та дешифрування даних.

Також, згідно з отриманими часовими характеристиками, можна зробити висновок, що поділ обчислень між близькими за технічними характеристиками вузлами зумовлює приріст швидкості виконання процедури шифрування та дешифрування даних (рис. 4).

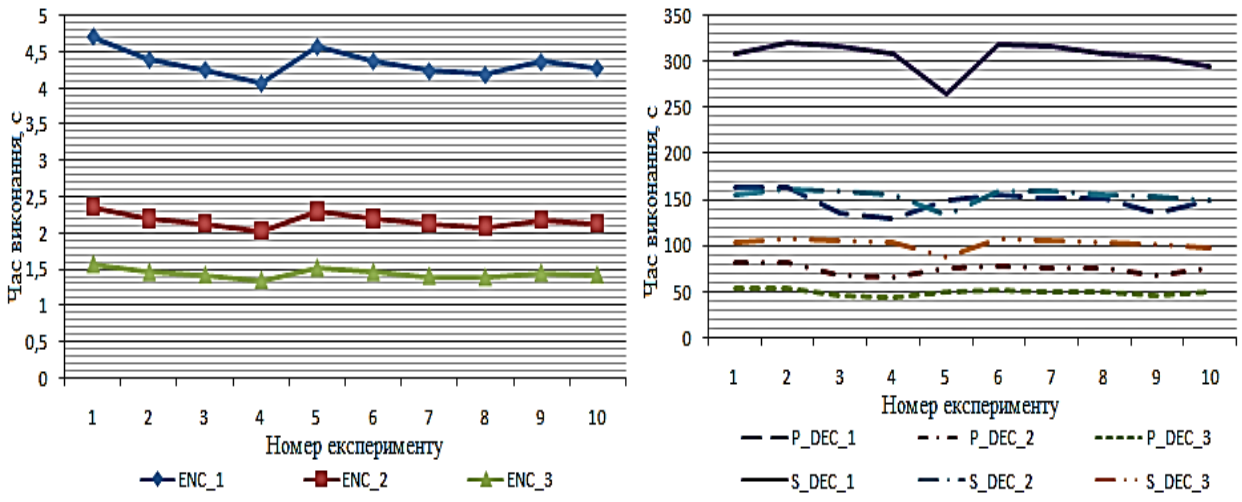
Однак прискорення в процедурі шифрування не є суттєвим і при незначному розмірі вихідних даних може бути виконане на локальному робочому місці за прийнятний час. Разом із тим спостерігається суттєве прискорення обчислень у процедурі дешифрування даних, що є цільовим завданням поділу обчислень.

При проведенні вказаного дослідження автори обмежилися тестуванням однієї (базової схеми обчислень), оскільки результати тестування останніх зводяться до оцінювання часу на передачу даних між вузлами мережі. У разі використання даних малого розміру це може призвести до появи суттєвих витрат часу щодо їх розподілу між клієнтами.

Висновки. У розглянутій роботі вивчено основні методи й підходи, які використовуються для побудови розподілених систем. Ураховуючи особливості перетворень даних, які лежать в основі механізмів заперечуваного шифрування, запропоновано реалізувати розподіл обчислень між групою комп'ютерів, об'єднаних у локальну мережу. Виконано побудову та програмну реалізацію запропонованих моделей, що виконують розподілене шифрування даних. Ефективність запропонованого підходу доведена в ході експериментів. Показано, як можна виконати перетворення даних з використанням указаних алгоритмів за оптимальний час, і визначено мінімальні технічні характеристики вузлів для виконання обчислень.

Науковою новизною вказаної роботи є використання розподіленої обробки даних в алгоритмах заперечуваного шифрування, які є досить вимогливими до обчислювальних ресурсів.

Практичне значення роботи полягає в розробленні моделі розподіленого шифрування даних, у якій можуть бути наявні механізми заперечуваного шифрування. Також використання цього підходу зумовило зростання продуктивності вихідного алгоритму шифрування за рахунок масштабування моделі обчислень.



а б

Рис. 4. Графік витрат часу на виконання обчислень: а – шифрування, б – дешифрування

Напрями подальших досліджень пов’язані з розробленням ефективних апаратних рішень, застосування яких призведе до вирішення проблем з продуктивністю в окремих блоках обчислень алгоритмів заперечуваного шифрування та появи можливості їх практичного застосування в промислових масштабах, розробленням і пошуком рішень для ефективної реалізації механізмів заперечуваного шифрування на базі сучасних симетричних криптографічних схем перетворення даних.

ЛІТЕРАТУРА

1. Рытенкова О. Рост объема информации – реалии цифровой вселенной. *Технологии и средства связи.* 2013. № 1 (94). С. 24–25.
2. Бабичев С.Л., Коньков К.А. Распределенные системы: учебное пособие для вузов. Москва : Юрайт, 2019. 507 с.
3. Таненбаум Э.С., Стеен В.М. Распределенные системы. Принципы и парадигмы. Санкт-Петербург : Питер, 2003. 877 с.
4. Молдовян Н.А., Вайчикаускас М.А. Расширение криптосхемы Рабина: алгоритм отрицаемого шифрования по открытому ключу. *Вопросы защиты информации.* 2014. № 2. С. 12–16.
5. Молдовян Н.А., Биричевский А.Р., Мондикова Я.А. Отрицаемое шифрование на основе блочных шифров. *Информационно-управляющие системы.* 2014. № 5. С. 80–86.
6. Балансировка нагрузки в облачных вычислениях / Е.Н. Десятирикова и др. *Весник ВГУ.* 2017. № 3. С. 103–109.
7. Гальченко А.В., Чопоров С.В. Заперечуване шифрування на основі застосування підходу гібридних криптографічних систем. *Радіоелектроніка, інформатика, управління.* 2019. № 1. С. 178–191.
8. Гальченко А.В., Чопоров С.В. Використання методу розділяй та володарюй в алгоритмах заперечуваного шифрування (неопублікована).
9. Бугеря А.Б., Ким Е.С., Соловьев М.А. Распараллеливание реализаций сугубо последовательных алгоритмов. *Труды ИСП РАН.* 2018. Т. 30. № 2. С. 25–44.
10. Chou T., Orlandi C. The Simplest Protocol for Oblivious Transfer. *LATINCRYPT 2015:* 4th International Conference on Cryptology and Information Security in Latin America, Guadalajara, 23-26 Aug. 2015. Guadalajara, 2015. P. 40–58.
11. Schlage-Puchta JC. On Shank's algorithm for modular square roots. *Applied Mathematics E-Notes.* 2005. Vol. 5. P. 84–88.

REFERENCES

1. Rytenkova O. (ed.) (2013) Rost ob"ema informatsii – realii tsifrovoy vselennoy [Growth of information volume - digital universe reality]. *Tekhnologii i sredstva svyazi,* vol. 1, no. 94, pp. 24–25.
2. Babichev S. L., Kon'kov K. A. (2019) Raspredelennye sistemy: uchebnoe posobie dlya vuzov [Distibuted systems]. M.: Yurayt (in Russian).
3. Tanenbaum E. S., Steen V. M. (2003) Raspredelennye sistemy. Printsipy i paradigmy [Distibuted systems. Principles and paradigms]. SaintPetersburg: Piter (in Russian).
4. Moldovyan N. A., Vaychikauskas M. A. (2014) Rasshirenie kriptoskhemy Rabina: algoritm otritsaemogo shifrovaniya po otkrytomu klyuchu [The Rabin's cryptography scheme extending: public key deniable encryption algorithm]. *Voprosy zashchity informatsii,* vol. 2, pp. 12–16.
5. Moldovyan N. A., Birichevskiy A. R., Mondikova Ya. A. (2014) Otritsaemoe shifrovanie na osnove blochnykh shifrov [Deniable encryption based on the block ciphers]. *Informatsionno-upravlyayushchie sistemy*, vol. 5, pp. 80–86.
6. Desyatirikova E. N., Khadzh A. M., Khodar A., Al'kadi U., Radzhab Kh. (2017)Balansirovka nagruzki v oblachnykh vychisleniyakh[Load balancing in cloud computing]. *Vesnik VGU*,vol. 3,pp. 103–109.
7. Halchenko A. V., Choporov S. V. (2019) Zaperechuvane shyfruvannia na osnovi zastosuvannia pidkhodu hibrydnykh kryptohrafichnykh system [Deniable encryption based on hybrid cryptographic systems using]. *Radioelektronika, informatyka, upravlinnia*, vol. 1, pp. 178–191.
8. Halchenko A. V., Choporov S. V. (2020) Vykorystannia metodu rozdiliai ta volodariui v alhorytmakh zaperechuvanoho shyfruvannia [The divide and conquer method in the deniable encryption algorithms] (unpublished).
9. Bugerya A. B., Kim E. S., Solov'ev M. A. (2018) Rasparallelivanie realizatsiy sugubo posledovatel'nykh algoritmov [The distributed implementation of sequential algorithms]. *Trudy ISP RAN*, vol. 30, no. 2, pp. 25–44.
10. Chou T., Orlandi C. (2015) The Simplest Protocol for Oblivious Transfer. Proceedings of the *LATINCRYPT 2015: 4th International Conference on Cryptology and Information Security in Latin America, (Mexico, Guadalajara, August 23–26, 2015)* (eds. Lauter K., Rodríguez-Henríquez F.), Berlin: Springer, pp. 40–58.
11. Schlage-Puchta J. (2005). On Shank's algorithm for modular square roots. *Applied Mathematics E-Notes*, vol. 5, pp. 84–88.