МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ПЗВО «МІЖНАРОДНИЙ КЛАСИЧНИЙ УНІВЕРСИТЕТ
імені ПИЛИПА ОРЛИКА»
Економіко-технологічний факультет
Кафедра інженерних технологій

Кваліфікаційна робота
на здобуття освітнього ступеня магістра
за освітньою програмою «Комп'ютерна інженерія»
зі спеціальності 123 «Комп'ютерна інженерія»

на тему: **«ЗАСОБИ ЗАХИСТУ КРИТИЧНОЇ ІНФОРМАЦІЇ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ СИСТЕМ»**

Виконала:
здобувач ІІ курсу, групи КІ -20-24
Насипайко Оксана Сергіївна

Керівник:
к.т.н., доцент кафедри інженерних технологій
Гайша Олександр Олександрович

Миколаїв – 2024

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

HDD	–	Hard Disk Drive
RBA	–	Remote Biometrical Assurance
SAML	–	Security Assertion Markup Language
SNMP	–	Simple Network Management Protocol
WAN	–	Wide Area Network
ЗІ	–	Захист інформації
ІКМ	–	Інформаційно-комунікаційні мережі
ІКС	–	Інформаційно-комунікаційні системи
ІКТ	–	Інформаційно-комунікаційні технології
ІТ	–	Інформаційні технології
ІТР	–	Інженерно-технічні робітники
ОС	–	Операційна система
ПЗ	–	Програмне забезпечення
ПК	–	Персональний комп’ютер
СЗІ	–	Система захисту інформації
СКД	–	Система контролю доступу
ТЗІ		Технічний захист інформації

ВСТУП

Актуальність. Напередодні 2020-х років уся планета виявилася обплетеною Всесвітнім павутинням Internet, інфраструктура якого утворюється шляхом об’єднання величезних регіональних мереж трансконтинентальними кабелями, а також численними каналами меншої перепускної здатності. Будь-які сучасні інформаційно-комунікаційні мережі (далі – ІКМ) можуть надавати своїм абонентам різноманітні, як завгодно складні, сервіси, однак, зазвичай, усі вони передбачають взаємодію двох абонентів між собою. Навіть, якщо разом одночасно спілкуються більше, ніж два абоненти (наприклад, як у телеконференціях із багатьма учасниками), все одно насправді за цією взаємодію приховується велика кількість взаємодій усіх абонентів поодинці із одним центральним сервером (тобто і тут мова іде про сукупність бінарних взаємодій типу «абонент-абонент»).

Важливо, що у переважній більшості усіх випадків взаємодії через ІКМ один абонент має виділене становище, може надавати у користування іншим абонентам свої ресурси (інформаційні, дискові, обчислювальні, алгоритмічні, і т.п.) і називається сервером, а інший підключається до нього і називається клієнтом. Під час підключення клієнта до сервера через ІКМ практично завжди виникає проблема санкціонованості цього процесу, адже абсолютно не будь-який клієнт може отримувати доступ до кожного ресурсу, який присутній в Інтернет (чи в ІКМ менших масштабів, зокрема регіональних – Wide Area Network, WAN). Коротко цю проблему можна охарактеризувати, як необхідність зведення відповідної системи захисту інформації (далі – СЗІ) в ІКМ, заснованої на певних протоколах автентифікації, що, відповідно, є актуальною задачею сучасної галузі кібербезпеки.

Відомі підходи до вирішення поставленої задачі. Очевидно, описана проблема виникла майже одразу із появою технології «клієнт-сервер» у територіально розподіленому варіанті її застосування (тобто фактично – з появою перших ІКМ). За цей час розроблено чимало способів її вирішення, в

результаті чого виникли як відомі надійні протоколи автентифікації типу Kerberos та OpenID Connect, так і поширені ненадійні механізми Cookie сесій, а також менш поширені у практичному використанні сертифікати X.509, можливості спеціальної мови SAML (Security Assertion Markup Language), а також механізм Secure SNMP (Simple Network Management Protocol) з використанням цифрового підпису.

Визначальною рисою усіх згаданих підходів є їх заснованість на знанні (інформації парольного характеру), або на атрибутах (деякі спеціальні файли складно, або неможливо сформувати вручну; вони повинні бути в наявності у законного користувача, тому можуть бути віднесені до атрибутів, хоча і нематеріального характеру). В той же час, у цих протоколах автентифікації відсутня прив’язка до біометричної інформації авторизованого користувача, що у деяких випадках може бути цілком неприйнятним через особливості системи захисту. Таким чином, актуальною є конкретна задача створення протоколу автентифікації, який включає біометричну інформацію клієнта, а отже дозволяє серверу більш надійним чином здійснювати його автентифікацію.

Таким чином, **метою роботи** є підвищення ступеня захисту ресурсів інформаційно-комунікаційних мереж за рахунок розробки та впровадження удосконаленого протоколу автентифікації.

Для досягнення поставленої мети слід вирішити наступні задачі дослідження:

- проаналізувати особливості існуючих протоколів автентифікації (як мережних, так і локальних);

- обрати один із протоколів, що може бути удосконалений, зокрема в частині використання біометричної інформації клієнта для його автентифікації;

- розробити удосконалений протокол автентифікації, який можна застосовувати в ІКМ;

- впровадити даний новий протокол у робочому програмному продукті та провести його тестування;

- проаналізувати результати дослідження, зробити висновки по роботі та оцінити перспективи її розвитку.

Об’єктом дослідження є процес автентифікації клієнта в інформаційно-комунікаційних мережах.

Предметом дослідження є алгоритми протоколів автентифікації клієнта, що можуть застосовуватися в ІКМ.

В роботі застосовуються **методи** криптографічних перетворень та біометричних досліджень (математичної статистики), а також елементи кластерного аналізу. Використано об’єктно-орієнтовану технологію програмування.

Наукова новизна полягає у тому, що вперше створено протокол RBA автентифікації клієнта в інформаційно-комунікаційних мережах на базі його біометричної інформації, що дозволяє підвищити ступінь захисту відповідних мережних ресурсів (в першу чергу за рахунок зведення додаткової ланки захисту).

Практичне значення роботи полягає у тому, що на базі розробленого протоколу створено працюючий програмний продукт, за допомогою якого можна виконувати віддалену автентифікацію клієнта за його біометричними показниками.

В **перспективі** до протоколу можна додати можливості по автентифікації сервера, які відсутні у версії протоколу RBA 1.0, щоби надати гарантій безпечного з’єднання і клієнтові також (зокрема, уникнути фішингу – підробки серверного ресурсу).

РОЗДІЛ 1. АНАЛІЗ СУЧАСНОГО СТАНУ ПРОБЛЕМИ ЗІ В ІКМ НА БАЗІ ПРОТОКОЛІВ АВТЕНТИФІКАЦІЇ

1.1. Загальний опис проблеми ЗІ в ІКМ

Інформаційно-комунікаційні мережі в першу чергу призначені для обміну та зберігання інформації. Як відомо, інформація, що у тій, чи іншій мірі підлягає захисту, може бути класифікована за ступенем секретності наступним чином:

- цілком таємна;
- таємна;
- для службового користування;
- відкрита.

Однак, для цілей даного дослідження більший інтерес представляє характеристика власника інформації, що захищається. Тут можна виділити наступні класи поділу:

- державна таємниця (що циркулює у державних установах, організаціях, підприємствах і безпосередньо відноситься або є спорідненою до сфери їх діяльності);
- службового характеру, причому така, що використовується на приватних підприємствах;
- особистого характеру, тобто секретна інформація, яка породжена або належить приватним особам.

Питання захисту державної таємниці в цілому відносяться до компетенції спеціальних структур (наприклад, Служби Безпеки України) і мають вирішуватися професіоналами найвищого рівня, причому з можливістю залучення необхідних, досить потужних матеріальних ресурсів. Взагалі кажучи, ресурси, що можуть бути відведені для забезпечення захисту державної таємниці, теоретично можуть бути як завгодно великими. В принципі тут може

навіть порушуватися відомий принцип галузі захисту інформації, за яким витрати на виконання захисту інформації не повинні перевищувати вартості самої інформації. Така ситуація може бути обумовлена політичними рішеннями, небажанням окремих високих посадовців нести навіть і незначні іміджеві втрати, іншими особистими причинами людей, в руках яких сконцентрована значна влада. Беручи до уваги ці обставини, розглядати у даній роботі захист державної таємниці не будемо.

З іншого боку, особиста інформація, яка є важливою для окремих приватних осіб, зазвичай не є досить цінною для широкої спільноти зловмисників (хакерів), що прагнули б нею заволодіти. Відповідно, зводити спеціальні СЗІ у цьому випадку м'яко кажучи не доцільно, тому у даній роботі захист персональних даних також не розглядається.

Таким чином, предметом захисту у даному дослідженні цілком обґрунтовано виступає комерційна інформація, яка є продуктом життєдіяльності приватних підприємств (широкого профіля).

Методологічно одним із перших кроків при вирішенні будь-якої задачі захисту інформації є аналіз заданої системи на предмет наявності об'єктів захисту, зокрема, для даної роботи, пошук слід проводити у складі інформаційно-комунікаційних мереж. Відповідно, навіть, до самої назви систем – інформаційно-комунікаційні – очевидно, усі їх об'єкти захисту можна розбити на два великих класи – рис. 1.1:

- сховища даних, або засоби збереження *інформації* (цим терміном назвемо усі пасивні сутності, місця, де можуть зберігатися дані, що підлягають захисту; це, наприклад, файли, робочі місця, письмові столи, і т.п., залежно від рівня абстракції, що прийнятий на даному етапі аналізу);

- канали зв'язку, або засоби *комунікації* (усі зв'язки між суб'єктами та об'єктами системи у різних комбінаціях; це, наприклад, провідний телефон в кабінеті директора підприємства, провідний телефон секретаря, ADSL-з'єднання у відділі технічних працівників, Ethernet-з'єднання, і т.п.).

Рис. 1.1. Найбільш укрупнена класифікація об’єктів захисту в ІКМ.

Обидва види об’єктів захисту з рис. 1.1 зручно унаочнюються на схемі інформаційних потоків системи: адже вхід та вихід кожного потоку може бути сховищем даних, а сам потік безпосередньо має реалізуватися якимсь каналом зв’язку.

Таким чином, для можливості виконання пошуку загроз (що є наступним кроком при зведенні СЗІ) для об’єктів захисту інформації обраної системи, необхідно провести аналіз існуючих (та, можливо, нових, які необхідно утворити шляхом впровадження системи захисту інформації) інформаційних потоків, тобто створити інформаційну модель підприємства. Для цього необхідно виконати аналіз наявних бізнес-процесів підприємства, існуючих виробничих відносин на ньому та загальної методики його роботи. Оскільки усі підприємства мають власні особливості, а саме, профіль діяльності, кількість та якість працівників, наявні матеріальні активи, і т.д., і т.п., то зробити точну загальну модель, звичайно, неможливо. Однак, можна розглянути орієнтовну узагальнену структуру, що включає в себе різноманітні основні елементи діяльності та складу приватних підприємств: купівлю/продаж, виробництво, сервісні функції, функціонування відділу інформаційних технологій, функціонування відділу головного інженеру (питання електрики, будівельні питання, пожежна безпека, механічні операції, тощо), і т.д. Таку узагальнену

модель приватного підприємства та взаємодії між його різними структурними ланками, зобразимо у вигляді схеми на рис. 1.2.

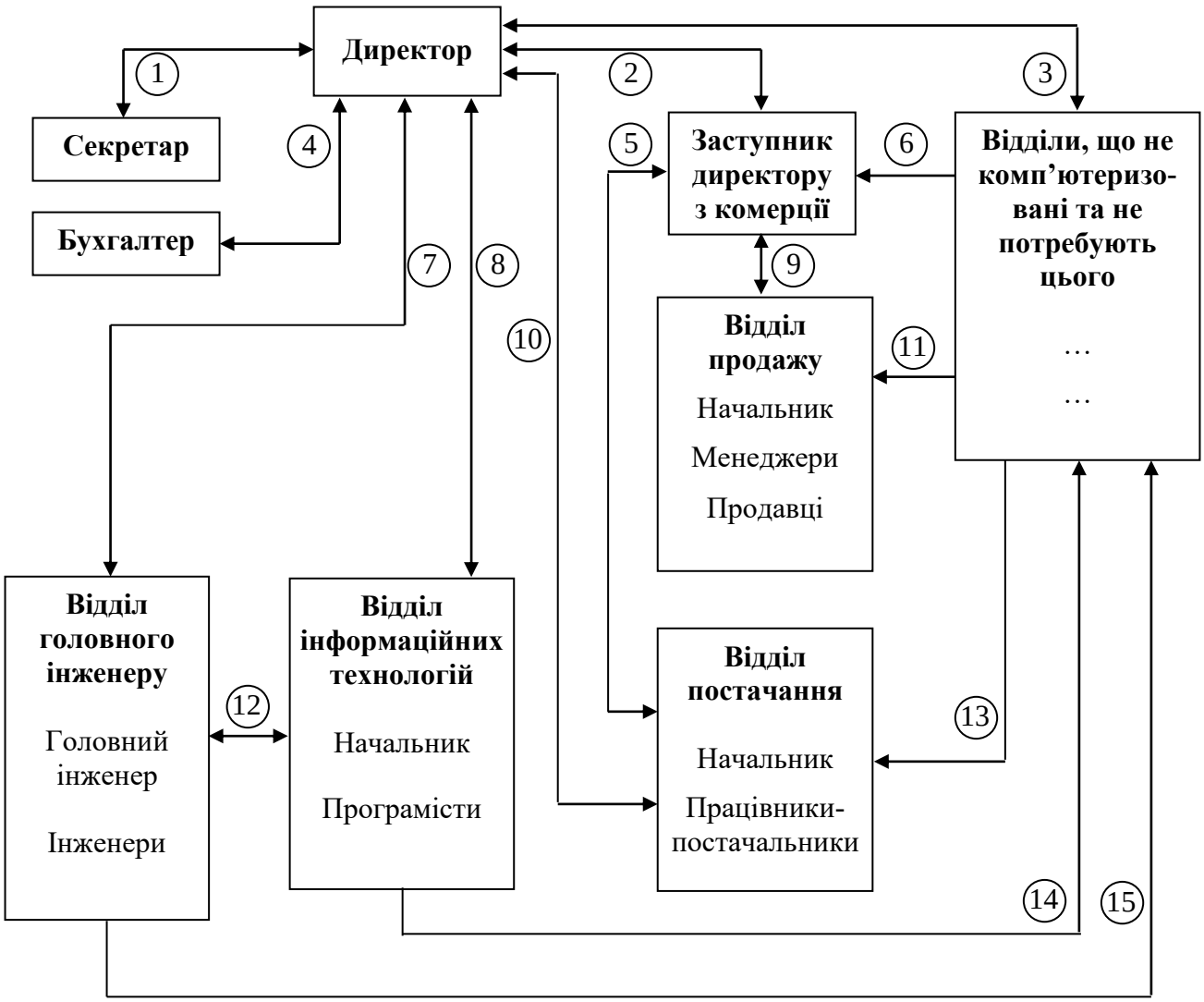


Рис. 1.2. Універсальна інформаційна модель приватного підприємства у вигляді схеми інформаційних потоків.

Для повного і правильного розуміння структури і значення кожного інформаційного потоку необхідно розшифрувати його семантичний зміст, що виконано в наступному тексті.

Дамо розшифровку наведених на рис. 1.2 позначень інформаційних потоків:

1 – інформація від секретаря про запити зовнішніх суб’єктів до директора; зворотний потік вказівок про поточні завдання; фізична суть інформації: мовна голосова;

2 – інформація про важливі замовлення, проблемні ситуації, звітування про хід комерційної діяльності виробництва; зворотний потік наказів з підвищення рівня комерційної діяльності підприємства;

3 – будь-яка безпосередня інформація від працівників до директора; зворотній потік заохочень та стягнень, відповідь на запити робітників;

4 – інформація про фінансовий стан підприємства; зворотний потік наказів про проплати, розподіл та управління коштами;

5 – інформація до заступника директору про необхідні закупки товарів та комплектуючих, запит фінансів; зворотний потік рекомендацій про оптимальний розподіл коштів на закупки, управління закупками;

6 – інформація від працівників про необхідність великих придбань, ремонту обладнання, вирішення довільних виробничих питань, суперечок;

7 – інформація від інженерно-технічних робітників (далі – ІТР) про нові перспективні розробки, удосконалення наявної продукції, вирішення принципових питань з проектування виробничого процесу (сюди ж включаються питання про апаратні засоби захисту критичної інформації приватного підприємства); зворотній потік даних про необхідність нових розробок та удосконалення старих, спрямовування у потрібному напрямі розробок продукції (стратегічне управління);

8 – інформація про нові інформаційні технології, які знаходяться на різних стадіях впровадження на підприємстві, в т.ч. ще тільки потребують впровадження (сюди ж включаються програмні засоби захисту критичної інформації приватного підприємства); зворотний потік інформації рекомендаційного характеру з питань ІТ;

9 – інформація відділу продаж про хід продажу, запит готової продукції, аналітична інформація різного характеру; зворотний потік інформації про

ціноутворення, планування рекламних подій, підвищення обізнаності потенційних покупців про продукцію підприємства;

10 – зведена інформація відділу постачання про проведені та плановані закупки, про необхідні великі закупівлі, відомості про контрагентів-постачальників; зворотний потік відомостей про пріоритетні місця закупок, управління процесом купівлі;

11 – інформація про хід виробництва готової продукції;

12 – обмін технічними відомостями про інформаційно-комунікаційні технології (ІКТ) та споріднені процеси;

13 – запит потрібних для нормального протікання виробничого процесу товарів працівниками;

14 – надання робочої документації по створенню продукції, яка містить елементи ІКТ;

15 – надання усієї робочої документації, консультації робітників, проведення навчання та роз'яснень.

Переважна більшість інформаційних потоків мають бути захищеними у тій, чи іншій мірі, причому засоби захисту потоків приватного підприємства, що циркулюють в інформаційно-комунікаційних мережах, будуть розглянуті нижче, тому тут розглянемо лише потоки, що є некомп'ютеризованими, аби до них більше не повертатися - табл. 1.1.

На основі аналізу табл. 1.1 обираємо інформаційні потоки, що слід комп'ютеризувати максимальним чином та впровадити засоби захисту програмного або апаратного характеру (зокрема, протокол автентифікації), що буде розглянуто у наступних розділах:

2, 4, 5, 7, 8, 9, 10, 12.

Табл. 1.1 – Матриця інформаційних потоків, що відображує їх можливу фізичну природу.

№ потоку	Мовна голосова	Мовна текстова	Графічна	Таблична	Відео
1	розмова телефонна, усна	SMS, записки	факс		
2	розмова телефонна, усна	записки	факс	звіти	
3	розмова усна	заяви		табель	
4	розмова телефонна, усна	рахунки	факс	звіти	
5	розмова телефонна, усна	службові записки		звіти відомість	
6	розмова усна				
7	розмова усна	службові записки, заяви	креслення	специфікації, відомості	моделювання, записи випробувань
8	розмова усна	службові записки, заяви, тексти програм	креслення, схеми алгоритмів	специфікації програм	моделювання, записи випробувань
9	розмова усна, телефонна	SMS, розпоря-дження	рекламні зображення	прайс-листи, звіти	рекламні ролики
10	розмова усна, телефонна	накази, службові записки, заяви	фотографії	відомості товарів	
11	розмова телефонна, усна	описи особливос-тей продукції	пояснюючі фотографії		відеозаписи роботи продукції
12	розмова усна	текстові документи	креслення, схеми	таблиці команд і станів	
13	розмова усна	службові записки, замовлення		перелік закупних товарів	
14	розмова усна	специфікації електроніки	схеми підклю-чення	таблиці електричних контактів	
15	розмова усна	текстові описи, технологія	креслення, схеми розміщення	специфікації	навчальні матеріали

При обранні вказаних восьми потоків були прийняті наступні допущення:

- у всіх можливих випадках усну розмову можна замінити (повністю чи частково) на електронні комп’ютерні засоби зв’язку;

- усі потоки, що пов’язані із відділами, які не підлягатимуть комп’ютеризації, не слід розглядати в контексті проблеми захисту інформації приватного підприємства в ІКМ, адже у цих відділах власне відсутні засоби інформаційно-комунікаційного характеру;

- деякі інформаційні потоки не можна комп’ютеризувати з нетехнічних причин (наприклад, усну або телефонну розмову секретаря і директора не можна комп’ютеризувати через необхідність збереження певної субординації у їхніх відносинах).

В цілому, оскільки сама комп’ютеризація інформаційних потоків не є основною темою даної роботи, то після обрання інформаційних потоків, які підлягають комп’ютеризації, вважатимемо, що вона вже зроблена і будемо у наступному розглядати для цих потоків ті ж протоколи захисту, що й для інших інформаційних потоків, які з самого початку забезпечувалися засобами ІКТ.

Наступним кроком при зведенні СЗІ є виконання аналізу загроз, які існують для даної системи. Як відомо, усі загрози поділяються на три класи, в залежності від тієї якості інформації, на порушення якої вони напрямлені. Так, розрізняють три основні якості інформації, що є важливими для галузі кібербезпеки:

- цілісність;

- конфіденційність;

- доступність.

Розглянемо докладніше ці три варіанти порушення критичних властивостей інформації приватного підприємства, що циркулює в ІКМ.

Загрози цілісності звичайно виникають у наступних випадках:

- відмова певного обладнання, що пов’язане із обробкою чи передачею даних;

- виконання людиною-оператором помилок («людський фактор»);

- деструктивна дія програм-вірусів;

- злочинне пошкодження інформації людиною-зловмисником.

Частою і дуже неприємною проблемою як персональних настільних комп’ютерів, так і ноутбуків, а іноді, навіть і серверів, є відмова жорстких дисків, виготовлених по типу HDD (Hard Disk Drive), у яких над тонкими магнітними пластинами, виконаними з феромагнетику літає (в прямому сенсі, дуже швидко переміщуючись поруч із пластинами) голівка, що читає та пише інформацію. Фактично, HDD є єдиною критичною частиною комп’ютера, у якій здійснюється механічний рух (вентилятори систем охолодження можна не рахувати через їх дуже низьку вартість та легкість заміни), а як відомо, електронні компоненти на порядки надійніше, ніж механічні. В цілому, відновити дані зі зламаного жорсткого диску можна тільки, якщо причиною є вихід з ладу електронного контролеру HDD, а якщо причина – у механічному пошкодженні пластин, то відновити інформацію взагалі буде неможливим (принаймні традиційним обладнанням сервісних центрів, без залучення напівміфічних «шпіонських» технологій). Отже, загроза втрати даних є цілком реальною і для її уникнення обов’язковим має бути резервування інформації, причому особливо критичної – навіть більш, ніж двократне (зазвичай виконується за розкладом, спеціальними утилітами).

Помилки людини є ще однією важливою загрозою цілісності, що може мати суттєві наслідки, причому мова йде саме про ненавмисні дії (злочинні наміри розглянемо пізніше). Імовірність реалізації цієї загрози напряму залежить від кваліфікації користувачів, що здійснюють обробку інформації: чим вона вища, тем менше можливість несвідомої модифікації даних. Очевидним шляхом нейтралізації цієї загрози є наскрізний процес підвищення кваліфікації персоналу шляхом проведення тренінгів, курсів, окремих занять.

Ще однією проблемою цілісності даних є робота вірусів типу вандалів, які повністю знищують, або вносять невиправні зміни в інформації з метою простого завдання шкоди усім користувачам, компаніям та комп’ютерам підряд. Очевидним шляхом подолання такої загрози є використання

перевірених (тобто таких, що використовуються відповідно до їх ліцензії) програм-антивірусів.

Також можливе пошкодження інформації людиною-зловмисником, який може бути як внутрішнім (незадоволеним або підкупленим, підмовленим) працівником компанії, так і зовнішнім (хакери, працівники фірм-конкурентів, і т.п.). Для уникнення несанкціонованого доступу (НСД) типу «запис/видалення інформації» слід проводити заходи з захисту інформації загального характеру, впроваджувати спеціальні апаратні та програмні засоби захисного характеру.

Для уникнення описаних нештатних ситуацій необхідно розробити і проводити комплекс заходів, направлених на збереження цілісності даних.

Наступним класом загроз є загрози доступності, які існують, коли:

- в мережі існує значне перевантаження каналів (наприклад, як наслідок DoS/DDoS-атаки, або у зв’язку із відмовою певної частини перепускних магістралей);

- тимчасово відмовив носій, на якому зберігаються дані, як частий випадок – недоступний (вимкнений) комп’ютер із необхідними даними;

- важливі дані стерто завдяки халатності, злого замислу, вірусом, тощо (загроза аналогічна по своїй суті модифікації).

Традиційно проблеми з доступністю інформації розглядаються в контексті атак на відмову у обслуговуванні, що є актуальним для приватного підприємства за умови, що у нього є сайт, на якому зберігається критична інформація. В цьому випадку, DoS-атака, а точніше розподілена DDoS, становить реальну загрозу, зважаючи на те, що сайти приватних підприємств зазвичай не розміщуються на високопродуктивних серверах і ризик виходу з ладу всього інформаційного ресурсу при такій атаці є надзвичайно високим.

Для зменшення імовірності успішної DDoS-атаки треба намагатися якомога сильніше оптимізувати серверне ПЗ, особливо, якщо сайт є активним і використовує певний серверний код (на зразок PHP). При цьому усі складові сайту (в першу чергу, скрипти) мають бути оптимізовані, оскільки навіть різниця в 1 мілісекунду на одному запиті, що здається мізерною, при сотні

тисяч запитів виростає у додаткові хвилинні затримки. Також суворому налаштуванню підлягають параметри самого веб-сервера, особливо в частині відсікання підозрілих, нестандартно сформованих запитів.

Нарешті, загрози конфіденційності з'являються, якщо суб'єкт має доступ до інформації, яка вимагає вищого рівня секретності, ніж той, який початково був наданий суб'єктові (тобто поточний рівень доступу вище за документально дозволений). Частинним випадком такої ситуації є проникнення в мережу зовнішнього зловмисника (початковий рівень доступу – нульовий, тому будь-який доступ є незаконним і виникає загроза конфіденційності інформації). Цей варіант загрози приймаємо за основний у подальшому матеріалі, і саме боротьба із ним буде основною метою подальшого дослідження.

Існуючі загрози зведемо в таблицю 1.2.

Табл. 1.2. Матриця загроз локальної мережі приватного підприємства.

Небезпечний об'єкт, процес	Цілісність	Конфіденційність	Доступність
Внутрішній зловмисник	1.1. Помилкова модифікація даних працівником фірми. 1.2. Навмисна модифікація даних працівником фірми.	2.1. Отримання доступу до інформації більш високого рівня доступу.	3.1. Помилкове видалення інформації. 3.2. Навмисне видалення інформації.
Зовнішній зловмисник	1.3. Зміна даних через відкриті для повного доступу ресурси.	2.2. Несанкціонований доступ в локальну мережу підприємства. 2.3. Прослуховування трафіку мережі.	3.3. Організація відмов у обслуговуванні (DoS-атак).
Вірус	1.4. Модифікація файлів.	2.3. Ведення скритного нагляду за користувачем. 2.4. Перехоплення введених паролів. 2.5. Відправлення в мережу файлів з інформацією.	3.4. Організація розподілених відмов у обслуговуванні (DDoS-атак). 3.5. Видалення даних. 3.6. Руйнація даних на жорсткому диску.
Відмова апаратної частини	1.5. Некоректний запис у файл з даними.		3.7. Відсутність доступу до інформації, розміщеної на носієві, який відмовив.

Таким чином, у підрозділі розглянуто типи інформації, що підлягає захисту в ІКМ, та за основний об'єкт, що захищається, обрано службову інформацію приватних підприємств. Далі проаналізовано інформаційні потоки приватного підприємства, що підлягають захисту в ІКМ засобами програмного

та апаратного характеру. Також виконано аналіз загроз та за основну прийнято загрозу несанкціонованого доступу клієнтів до серверної інформації з більш високим рівнем доступу, ніж є у клієнта (за умовчанням мається на увазі доступ на читання). Далі розглянемо, якими існуючими засобами цей захист може забезпечуватися.

1.2. Аналітичний огляд науково-технічних джерел, присвячених проблемі ЗІ в ІКМ на базі протоколів автентифікації

Як зазначалося у вступі, на сьогоднішній день існує певна кількість протоколів автентифікації, що мають забезпечувати захист інформації при організації зв’язку типу «клієнт-сервер» в ІКМ; розглянемо їх докладніше.

Так, головним засобом вирішення окреслених проблем автентифікації у мережі є протокол Kerberos, що є конкретною реалізацією протоколу Нідхема-Шредера, і описаний у багатьох джерелах. Kerberos – це програмний продукт, розроблений в середині 1980-х років у Масачусетському технологічному інституті і з тих пір він піддався ряду принципових змін. Клієнтські компоненти Kerberos присутні в більшості сучасних операційних систем (ОС).

Цей протокол призначений для вирішення наступного завдання. В наявності відкрита (незахищена) мережа, у вузлах якої зосереджені суб'єкти – користувачі, а також клієнтські і серверні програмні компоненти. Кожен суб'єкт має секретний ключ. Щоб суб'єкт *C* міг довести свою справжність суб'єкту *S* (без цього *S* не стане обслуговувати *C*), він повинен не тільки назвати себе (ідентифікація), але і продемонструвати знання секретного ключа (автентифікація). *C* не може просто надіслати *S* свій секретний ключ, по-перше, тому, що мережа відкрита (доступна для пасивного і активного прослуховування), а, по-друге, тому, що *S* не знає (і не повинен знати) секретний ключ *C*. В системі потрібно використовувати менш прямолінійний спосіб демонстрації клієнтом знання свого секретного ключа для сервера.

Для цього система Kerberos включає довірену третю сторону (тобто сторону, якій довіряють усі), що володіє секретними ключами суб'єктів, які обслуговуються, і допомагає їм у попарній перевірці автентичності. Схема застосування всього протоколу наведена на рис. 1.3.

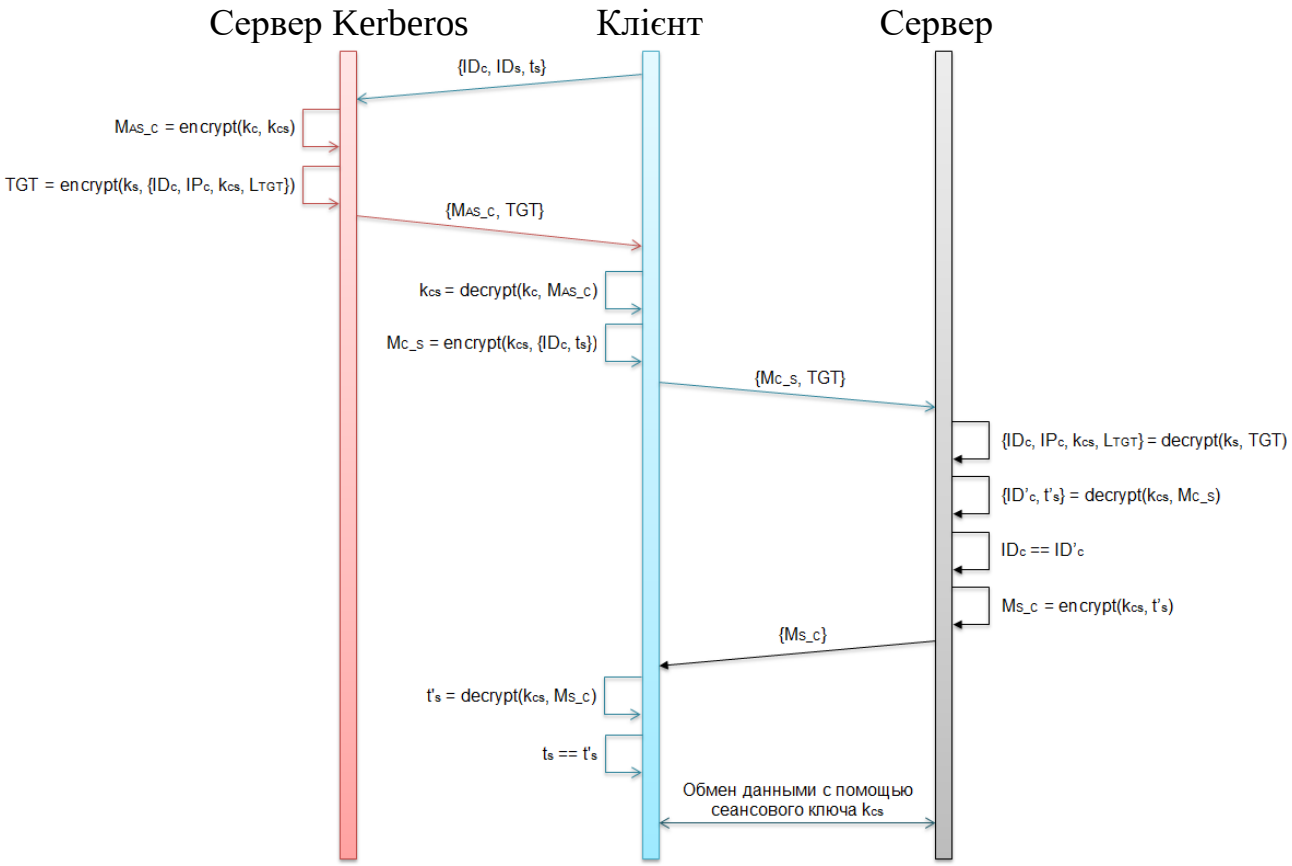


Рис. 1.3. Схема використання протоколу Kerberos для автентифікації у ІКМ.

Спрощено послідовність ідентифікації / автентифікації за допомогою Kerberos версії 5.0 на основі специфікації RFC 1510 "Request for Comments: 1510. The Kerberos Network Authentication Service (V5)" виглядає наступним чином:

1) Клієнт посилає серверу Kerberos запит, який містить свій ідентифікатор IDc і ідентифікатор сервера даних (запитуваної послуги) IDs, а також мітку часу ts.

2) Сервер Kerberos виконує наступні дії:

- по мітці часу ts перевіряє синхронізацію свого годинника з годинником клієнта;

- формує повідомлення MAS_C - зашифровує сеансовий ключ kcs ключем клієнта kc;

- формує квиток TGT - зашифровує ідентифікатор клієнта IDc, його IP-адресу IPc, сеансовий ключ kcs і час життя квитка (сеансового ключа) LTGT ключем сервера ks;

- відсилає повідомлення MAS_C і квиток TGT клієнту.

3) Клієнт виконує наступні дії:

- розшифровує повідомлення MAS_C за допомогою свого ключа kc для отримання сеансового ключа kcs;

- формує повідомлення MC_S - зашифровує свій ідентифікатор IDc і мітку часу ts сеансовим ключем kcs;

- відсилає повідомлення MC_S і квиток TGT сервера.

4) Сервер виконує наступні дії:

- розшифровує квиток TGT за допомогою свого ключа ks для отримання ідентифікатора клієнта IDc, його IP-адреси IPc, сеансового ключа kcs і часу життя квитка LTGT;

- розшифровує повідомлення MC_S за допомогою сеансового ключа kcs для отримання ідентифікатора клієнта ID'c і мітки часу t's;

- для автентифікації клієнта виконує порівняння ідентифікаторів IDc і ID'c, ойтриманих, відповідно, з квитка TGT і повідомлення MC_S;

- формує повідомлення MS_C - зашифровує мітку часу t's сеансовим ключем kcs;

- відсилає повідомлення MS_C клієнту.

5) Клієнт прикінцево:

- розшифровує повідомлення MS_C за допомогою сеансового ключа kcs для отримання мітки часу t's;

- для автентифікації сервера виконує порівняння міток часу ts і t's.

6) Обмін даними між клієнтом і сервером виконується за допомогою сеансового ключа kcs протягом часу життя квитка (сеансового ключа) LTGT.

Як видно з даного протоколу, крім ідентифікації / автентифікації, паралельно вирішується питання з обміном сеансовим ключем.

Ярким недоліком розглянутого підходу є відсутність автентифікації самого користувача. Фактично автентифікації підлягає набір даних (ключів), яким може заволодіти зловмисник, а потім, як результат, і отримати доступ до віддалених ресурсів серверу. Також недоліком може виявитися наявність третьої сторони (серверу Kerberos), якщо він буде підроблений.

Набагато більш «молода» надбудова OpenID Connect над протоколом OAuth 2.0 має абсолютно той же самий недолік. OAuth – це відкритий протокол (схема) авторизації, що дозволяє надати третій стороні обмежений доступ до захищених ресурсів користувача без необхідності передавати їй (третій стороні) логін і пароль [8-9]. Робота над протоколом почалася в листопаді 2006 року, а остання версія OAuth 1.0 була затверджена 4 грудня 2007 року. Як подальший розвиток в 2010 році з'явився протокол OAuth 2.0, остання версія якого в якості в RFC 6749 опублікована в жовтні 2012 року.

OpenID Connect - відкритий стандарт децентралізованої системи автентифікації, що надає користувачеві можливість створити єдиний обліковий запис для автентифікації на безлічі не пов'язаних один з одним Інтернет-ресурсів, використовуючи послуги третіх осіб. Базовою функцією OpenID є надання портативного, клієнт-орієнтованого, цифрового ідентифікатора для вільного і децентралізованого використання. Таким чином, дійсно доступ забезпечується за допомогою цифрового атрибуту – ідентифікатора (спеціального файлу), яким може заволодіти зловмисник, і, отже, отримати доступ до віддалених ресурсів користувача.

Інші протоколи автентифікації в ІКМ (тобто такі, що використовуються у віддаленому режимі) мають аналогічний недолік: вони не беруть до уваги біометричні показники клієнта, хоча очевидно, що така можливість безперечно є наявною.

1.3. Постановка задачі дослідження

Беручи до уваги усі особливості інформації, наведеної у попередньому розділі, можна сформулювати наступну задачу дослідження: розробити протокол автентифікації в інформаційно-комунікаційних мережах та систему захисту інформації на основі його, що використовували би у якості об’єкта контролю біометричні показники особи, яка потребує надання доступу. При цьому мають бути вирішені наступні питання:

- який протокол автентифікації брати за основу;
- в якій саме частині вносити зміни в обраний протокол автентифікації;
- які саме біометричні показники повинні підлягати контролю;
- яким способом здійснювати порівняння поточного образу клієнта та еталону;
- як реалізувати конкретну СЗІ на базі розробленого протоколу;
- якими є результати роботи створеної СЗІ.

Результатом роботи має бути завершений програмний продукт у вигляді розподіленого програмного комплексу, який забезпечує виконання автентифікації за розробленим протоколом.

Відповіді на ці запитання у комплексі визначають особливості даного дослідження і будуть надані та обґрунтовані в наступних розділах.

РОЗДІЛ 2. АНАЛІЗ ТА ОБГРУНТУВАННЯ ВИБОРУ ЗАСОБІВ ТА ТЕХНОЛОГІЙ, ЩО МОЖУТЬ БУТИ ЗАСТОСОВАНІ ДЛЯ ЗІ В ІКМ З ВИКОРИСТАННЯМ ПРОТОКОЛІВ АВТЕНТИФІКАЦІЇ

2.1. Аналіз існуючих протоколів автентифікації

Недоліком існуючих протоколів автентифікації в ІКМ у попередньому розділі було названо відсутність урахування біометричних параметрів суб'єкта, що авторизується. Відповідно, для початку слід розглянути протоколи біометричної автентифікації, що виконується локально (на тому ж комп'ютері, доступ до якого контролюється).

Найбільш укрупнено усі протоколи (способи) біометричної автентифікації діляться на статичні (по фізіологічним характеристикам тіла людини) та динамічні (по оцінці поведінки людини у певних, стандартизованих ситуаціях). Розглянемо їх особливості докладніше.

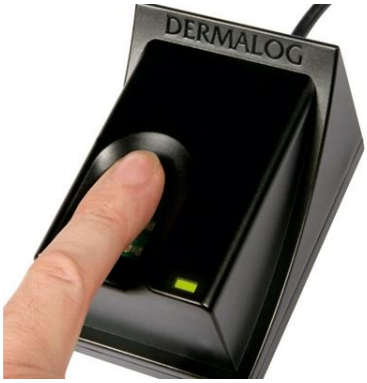
Почнемо з автентифікації по фізіологічним характеристикам людини, які змінюються дуже повільно (істотні зміни видно на інтервалах часу, що значно перевищують час між послідовними автентифікації), або взагалі не підлягають змінам.

В першу чергу, під такий тип автентифікації підпадає використання відбитків пальців людини, унікальність яких використовується вже більше ста років в криміналістиці (метод, названий «дактилоскопія»). Згідно з різними оцінками, ймовірність збігу відбитків пальців у двох різних людей становить від 10^{-10} (тобто у двох осіб з не менше ніж 10 млрд. людей) до 10^{-4} (у двох із 10 тис.). Навіть при розгляді самої песимістичної оцінки, даний рівень традиційно вважається достатнім для використання відбитків пальців у якості 100% -ного доказу в суді.

Для цілей даної роботи інтерес представляє можливість розширення протоколу доступу до ресурсів локального ПК на основі такої і всіх наступних

біометричних систем, до контролю доступу віддалено. Очевидним недоліком саме цього варіанту СКД є необхідність наявності у кожного клієнта спеціалізованого апаратного рішення - сканера відбитків пальців - рис. 2.1.

а)



б)

в)

г)

Рис. 2.1. Варіанти сканерів відбитків пальців: *а* - стаціонарний для обліку робочого часу співробітників, *б* - портативний для ПК, *в* - у мобільному телефоні, *г* - комбінований з атрибутною СКД (по електронній карті).

У деяких електронних цифрових пристроях виробники вводять сканер відбитків пальців як додаткову опцію, яка підвищує конкурентоспроможність продукту на ринку - рис. 2.1., в. Однак, якщо потрібно впровадити такий спосіб

автентифікації на основі вже існуючого апаратного забезпечення (а не такого, що тільки буде придбане в майбутньому), то ймовірність відсутності таких датчиків в наявній (старій) техніці буде близька до 100%. Відповідно, необхідні витрати на їх придбання і впровадження, навчання персоналу, обслуговування, супровід, і т.п., тобто тут повною мірою виявляються недоліки статичних біометричних систем, а саме їх вартісні характеристики.

Ще сильніше ситуація з фінансами ускладнюється при бажанні використовувати сканери райдужної оболонки ока, які є значно більш дорогими пристроями, ніж сканери відбитків пальців. Цей спосіб автентифікації заснований на унікальності картини райдужних оболонок ока у кожної людини. В цілому, багато «місць» на тілі людини мають унікальну картину тих, чи інших елементів, які можна використовувати для автентифікації: це і особливості шкірних покривів, розподіл жирової тканини, параметри кісткового скелета - все, що повільно змінюється, відносно легко детектується, має задовільну стабільність на малих часових інтервалах для однієї конкретної людини, може бути використано в якості контрольованої характеристики в біометричних СКД. Звичайно, в залежності від складності процесу контролю таких різноманітних характеристик буде змінюватися і вартість необхідного апаратного забезпечення.

Так, досить дорогі сканери форми руки, голови, обличчя, тому що для адекватної роботи вони повинні містити кілька камер і складне математичне забезпечення для обробки (суміщення) просторово рознесених зображень.

Висока вартість (особливо при великій кількості абонентських пунктів СКД або, наприклад, необхідності забезпечення їх мобільності) не є єдиним недоліком статичних систем. У деяких випадках існують принципові перешкоди організаційного характеру для впровадження статичних біометричних систем, в той час, як динамічні системи можуть бути легко впроваджені. Така ситуація якраз реалізується, наприклад, при необхідності введення контролю доступу до ресурсів інформаційно-комунікаційної мережі, що має загальновідомий вихід у всесвітнє павутиння Інтернет.

У таких випадках традиційним підходом є використання парольного захисту, однак, при вході в систему, на додаток до парольного захисту або замість нього, легко можна ініціювати виконання користувачем деяких дій, і за особливостями самого процесу їх виконання людиною проводити процес автентифікації (тобто оцінювати поведінку користувача, манеру його дій, також і числові характеристики, що характерно для поведінкових біометричних СКД). Таким чином, при наявності великої необхідності, біометричні системи можуть впроваджуватися без будь-якої попередньої підготовки, причому мова йде саме про поведінкові (динамічні) СКД. Ця ситуація є надзвичайно зручною для організації протоколу автентифікації в ІКМ, що і буде виконано у подальшому, (але спочатку треба буде докладно розглянути принципи дії поведінкових біометричних систем).

Отже, як висновок можна сказати, що фізіологічні біометричні СКД мають наступні недоліки:

- висока вартість початкових одноразових витрат на реалізацію системи;
- висока вартість супроводу системи;
- слабка гнучкість, неможливість віддаленої автентифікації без попереднього створення апаратного абонентського пункту.

З огляду на ці недоліки, однозначно можна сказати, що в даному дослідженні використовувати статичні біометричні СКД недоцільно, тому розглянемо другий клас таких систем - побудовані на контролюванні поведінки людини.

Загальновідомо, що кожна людина в процесі життєдіяльності набуває все більшої кількості звичок. Фактично, кожен новий вид діяльності, призначений для виконання людиною, перший раз проводиться якимсь певним способом і, якщо індивід в цілому задоволений результатом, далі цей спосіб або манера виконання закріплюється і переходить в звичний стиль поведінки. У той же час, переважна більшість дій (крім, можливо, найпростіших) мають різні варіанти або способи виконання та характеризуються різними значеннями своїх числових параметрів. Це стосується і складових комплексних процесів,

зокрема, пов'язаних з вищої нервової діяльністю (як-то звичка мислити певним чином, наприклад, песимістично або оптимістично; намагатися знайти рішення проблеми самостійно або відразу звертатися за допомогою до кого-небудь, і т.д.), і навіть простих моторних дій, пов'язаних зі звичайною життєдіяльністю. Так, наприклад, ніхто не стане заперечувати, що кожна людина має свої особливості ходьби, які простими словами називають «хода», оригінальними особливостями написання тексту (свій «почерк»), і т.д.

В цілому, можна сказати, що будь-який вид діяльності людини (від досить простих до найскладніших), який раніше багато разів нею виконувався, має свої стійкі особливості, які можуть бути використані для контролю в біометричних СКД.

Слід зазначити деякі особливості самих поведінкових СКД, пов'язані з варіативністю виконання тієї чи іншої контрольованої дії при різних умовах, а саме, необхідно:

- проаналізувати особливості варіативності виконання потенційно контрольованої дії однією людиною в різні моменти часу (тобто у випадку, що найбільш часто зустрічається, для контролю якоїсь числової характеристики обраної дії, потрібно якісно або кількісно оцінити дисперсію, що отримується на основі обробки значень цієї характеристики, виміряних в різних експериментах з одним і тим же користувачем);

- розібрати особливості варіативності ознаки в межах групи людей (тобто проаналізувати варіативність виконання потенційно контрольованої дії різними людьми). Для цього можна оцінити внутрішньогрупову дисперсію, що отримується на основі обробки середніх значень цієї характеристики, що приписуються кожному користувачеві окремо, і отриманих для кожного користувача в результаті окремої серії експериментів.

Отже, в першу чергу, розглянемо варіативність для однієї людини, і ця індивідуальна варіативність, в свою чергу, також має кілька аспектів.

По-перше, деякі дії, які в рамках даного дослідження доречно буде назвати надзвичайно простими, можуть у зв'язку з самою своєю природою мати

малу варіативність (причому сюди ж слід віднести і групову варіативність). Наприклад, така дія як плескання в долоні, хоча і може виконуватися досить різноманітними способами з точки зору взаємного просторового положення долонь, сили удару, швидкості руху рук, і т.д., проте така його характеристика, як тривалість самого звуку (інтервал часу в мілісекундах від моменту реєстрації звукового сигналу мікрофоном і до його падіння нижче певного процентного значення, припустимо, 1%) не буде мати значної варіативності в зв'язку з самою природою процесу.

По-друге, крім самої природи процесу, варіативність також може бути обмежена просторовими, тимчасовими або іншими рамками. Наприклад, в деяких автоматизованих системах час введення інформації обмежений і по його закінченні нормальна робота користувача переривається (наприклад, задається питання, чи тут знаходиться людина). Така поведінка системи не дозволяє використовувати таку характеристику, як час введення даних в систему, для ефективної поведінкової ідентифікації окремих категорій громадян, таких як глибокі пенсіонери, яким часто для роботи в системі потрібний значний час, що суттєво перевищує середні значення. Ще одним прикладом обмежувальних рамок може служити використання незвичайних засобів введення-виведення для контролю, наприклад, почерку людини. Так, пропозиція написати текст спеціальною ручкою на скляному сенсорному екрані викличе у багатьох користувачів уповільнення всього процесу письма, і, як наслідок, більш старанне виведення окремих букв, що в цілому знижує варіативність (внутрішньогрупову - у більшій мірі, але також і індивідуальну).

По-третє, багато (якщо не всі) поведінкові характеристики можуть залежати від настрою (в меншій мірі), а також, що більш істотно - від втоми (найбільш важливий аспект), нездужання, хвороби, травм. Однак, принципових перешкод до впровадження поведінкових систем в реальних СКД зазначені зауваження не вносять, тому що втома може бути врахована в роботі системи якимось окремим показником (коефіцієнтом), що може змінюватися протягом робочого дня.

По-четверте, деякі процеси мають дуже погану повторюваність, наприклад, не можна контролювати силу удару м'яча по невеликій мішені, так як по ній ще потрібно потрапити, а при попаданні удар може бути не центральним (як необхідно для вимірювання сили удару), а дотичним (і тоді, при малій зміні траєкторії, зафіксована датчиком сила може бути дуже і дуже мала, що абсолютно не відповідатиме реальній силі, з якою був кинутий м'яч, тобто поведінці користувача). Однак, такі характеристики все одно можна успішно використовувати для контролю доступу, якщо тільки в якості контрольованої величини вибрати не миттєве (одноразово зняте) значення такої характеристики, а середнє значення на великому числі випробувань. Практично це буде виглядати наступним чином:

- людина кидає м'яч по мішені 10 разів і кожного разу датчиком фіксується сила удару (в деяких випадках навіть і дотичного, навіть і невдалого, тобто без попадання, коли сила в результаті дорівнює нулю);

- середнє значення сили, що спостерігається за цими 10 ударами записується в файл-еталон;

- потім, при необхідності авторизуватися в системі, людині знову пропонується N разів виконати вказану дію;

- розраховується середнє значення по числу всіх спроб;

- виконується безпосередньо порівняння двох середніх і робиться висновок про успішну авторизацію або відмову в доступі.

Порівнянню у даному випадку підлягають значення, усереднені по N спробам одного користувача.

Ще одне, надзвичайно важливе зауваження, що стосується поведінкових систем, полягає в тому, що, звичайно ж, кожна людина є живою системою, що пристосовується, тому, якщо змінюються зовнішні умови, то і поведінка також підлягає зміні. Однак, ключовим моментом тут є те, що ці зміни все одно мають еволюційний (поступовий), а не вибуховий характер.

Наприклад, якщо людині навіть наказати найсуворішим чином почати писати іншим почерком, повністю задовольнити таку вимогу миттєво вона не

зможе. Можливо, відразу можна буде поміняти величину букв, або навіть їх нахил, однак більш глибинні параметри написання окремих символів (як-то різноманітні завитки, початки і закінчення при написанні літер) все одно будуть вказувати на авторство рукописного тексту. Також іноді обов'язково будуть проскакувати і старі варіанти написання тексту, що буде також свідчити про його первісне авторство. Таким чином, навіть при цілеспрямованій примусовій зміні, поведінка людини не може змінитися відразу, тим більше вона не може занадто змінитися у користувача, який навпаки бажає авторизуватися в системі і проходить процес автентифікації.

Отже, єдиною важливою умовою успішної роботи СКД, заснованої на контролі поведінки, пов'язаною з мінливістю цієї самої поведінки, є необхідність малості проміжку часу між двома послідовними автентифікації (не більше кількох місяців, а краще - кілька тижнів) і періодичне оновлення еталонних характеристик (наприклад, раз на місяць - при інтенсивній роботі в системі). В ідеалі необхідно, щоб таке оновлення відбувалося прозоро для користувача, тобто шляхом пасивного спостереження якоюсь автоматизованою частиною біометричної СКД, відповідальною за оновлення еталонної інформації, за його рутинними робочими процесами. Якщо таку прозору поведінку СКД реалізувати неможливо, то шляхом впровадження організаційних заходів щодо захисту інформації (тобто, наприклад, за розкладом, закріпленим спеціальною інструкцією по роботі в системі), слід періодично «знімати» з користувача системи поточні значення його контрольованих поведінкових характеристик.

Другий аспект роботи поведінкових систем заснований на потенційній варіативності обраної характеристики всередині групи, тобто для різних особистостей. Як уже зазначалося вище, деякі характеристики не можуть сильно варіюватися в зв'язку з самою своєю природою, що стосується і індивідуальної, і внутрішньогрупової варіативності. Наприклад, час виконання стрибка на місці (від моменту відриву ніг до моменту торкання з поверхнею) не може збільшуватися для різних людей у великій мірі, тому що ніхто з людей не

вміє літати. Це ще один приклад характеристики, яка має обмежену самою своєю суттю варіативність.

З огляду на вищесказане, можна стверджувати, що завдання фахівця із захисту інформації при зведенні протоколу роботи динамічної біометричної СКД полягає у відборі комплексу таких поведінкових характеристик і їх параметрів, які найкращим чином підходять для цілей захисту інформації, а точніше мають наступні властивості:

- володіють достатньою стабільністю своїх числових оцінок (оцінки) для даного індивідуума (іншими словами у таких характеристик повинна бути мала індивідуальна варіативність);

- велика групова варіативність (тобто іншими словами, значення даної характеристики для різних людей повинні бути такими, щоб мати можливість відрізнятися один від одного, причому чим у більш значній мірі, тим краще);

- не дуже швидка мінливість середнього значення характеристики в часі (інтервал часу, за який відбуваються значні зміни середнього значення, повинен бути хоча б в рази більше інтервалу між двома послідовними авторизаціями);

- принципово хороша повторюваність результатів вимірювань характеристики у одного індивідуума (скупченість результатів біля середнього значення, дисперсія мала), через що контролю підлягає саме виміряне значення;

- принципово погана повторюваність результатів вимірювань характеристики у одній групі (великий розкид значень навколо середнього, велика дисперсія).

Слід зазначити, що застосування автентифікації за результатами вимірювань всього лише однієї характеристики не доцільно, тому що має малу ефективність. Набагато кращі результати дає перевірка відразу декількох характеристик, і чим більше їх підлягає контролю, тим ефективніша така система. В цьому випадку нівелюється можливість простого збігу значень вимірювання якої-небудь однієї характеристики, тому що береться до уваги цілий їх ансамбль. У цьому випадку кожен користувач після зняття еталонної інформації буде характеризуватися для системи не одним числом, але *n*-мірним

(по числу контрольованих величин) вектором. Наочно такий вектор може бути представлений стовпчатою діаграмою, дуже зручною для поверхневого візуального аналізу.

Розглянемо докладніше приклад, в якому для аналізу особистості користувача біометричної СКД вибрано чотири характеристики, що виражаються числовими величинами в межах (0; 200). З трьох різних людей були зняті еталонні характеристики, в результаті чого отримана табл. 2.1.

Табл. 2.1. Приклад еталонних даних для 3 людей по 4 характеристикам.

	Хар-ка 1	Хар-ка 2	Хар-ка 3	Хар-ка 4
Користувач1	10	22	115	48
Користувач2	28	23	56	17
Користувач3	34	27	55	62

Дані із табл.2.1 можуть бути наочно представлені наборами стовпчикових діаграм - як на рис. 2.2.

Проаналізуємо рис. 2.2: на ньому крім еталонних значень всіх контрольованих характеристик (пронумеровані цифрами від 1 до 4) для трьох користувачів (Еталон1, Еталон2, Еталон3), також зображений «знімок» людини, яка бажає пройти авторизацію в даний момент. Елементарний візуальний аналіз дозволяє вирішити задачу ідентифікації, якби вона була актуальна (таке завдання могло би виникнути, якби точно було відомо, що авторизуватися в системі може тільки хтось один із тих трьох людей, що в ній зареєстровані, тобто з яких були зняті еталонні характеристики). Дійсно очевидною є схожість людини, що авторизується, і Еталона1, інша справа, що дана схожість має отримати числове математичне підтвердження. Таким чином, задача автентифікації не може бути вирішена приблизно, а вимагає суворого математичного розрахунку.

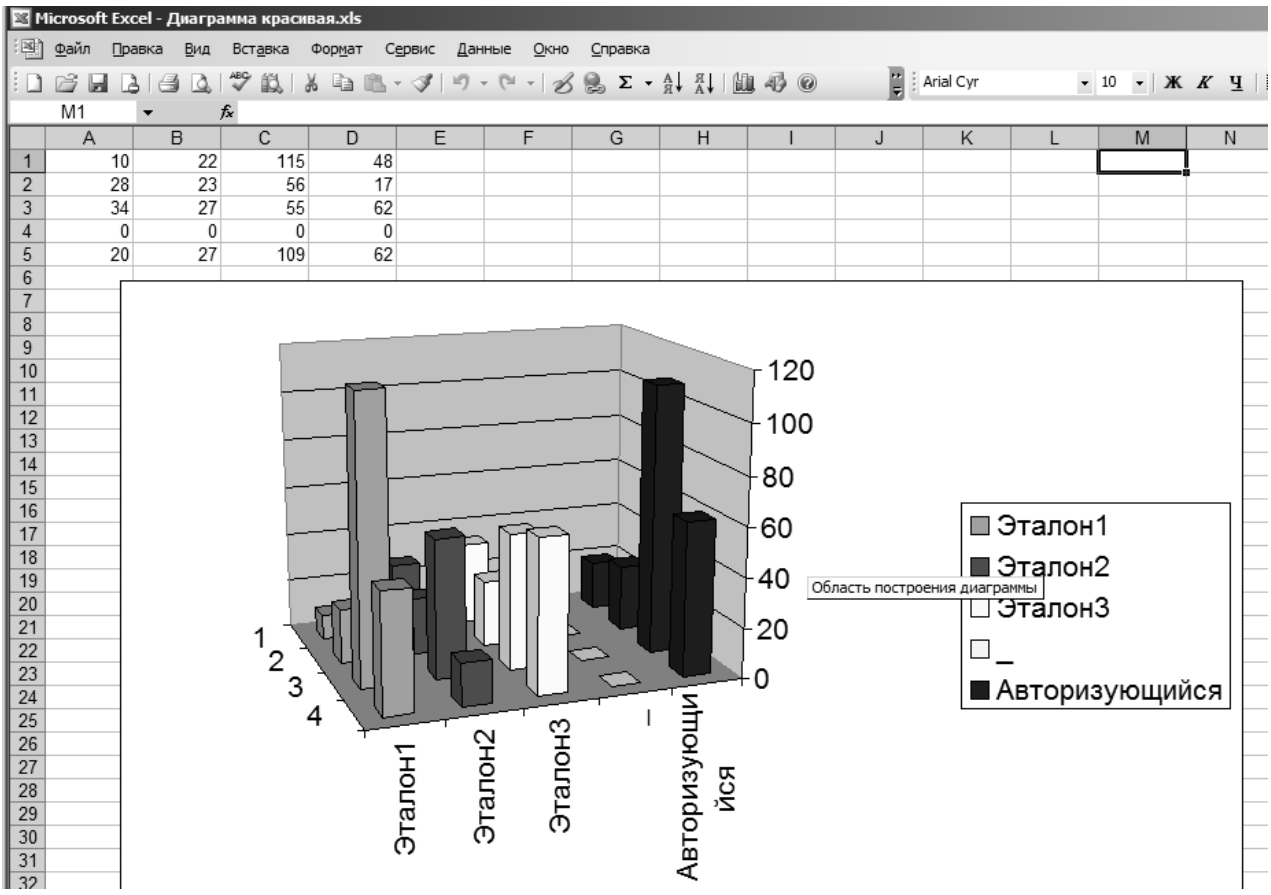


Рис. 2.2. Наочне представлення еталонної інформації, при якому кожна людина характеризується своїм набором чотирьох характеристик.

Беручи до уваги усю вищесказану інформацію, можна сформулювати наступне питання: які ж характеристики можуть задовольняти зазначеним теоретичним вимогам, і які реально застосовуються в біометричних СКД? В першу чергу це можуть бути характеристики роботи користувача з пристроями введення-виведення, а точніше - саме введення інформації в ПК. І найцікавішим (комплексним і складним) з таких пристроїв є клавіатура комп'ютера. Серед показників роботи з клавіатурою можна виділити наступні:

- середня швидкість набору тексту;

- відносна кількість помилок при наборі тексту;

- середній час утримання різних окремих кнопок при наборі тексту;

- середній час набору деяких поширених біграм (двосимвольних послідовностей).

Що стосується миші, як пристрою введення інформації, то тут контролю можуть підлягати:

- максимальна швидкість переміщення миші;

- середній час наведення курсору миші на невеликі об'єкти перед клацанням;

- середній інтервал часу між двома послідовними натисканнями лівої кнопки миші при подвійному натисканні;

- середній інтервал часу між викликом контекстного меню за допомогою правої кнопки миші і вибором будь-якого пункту меню;

- і т.д.

Слід сказати, що можна запропонувати і більшу кількість різних показників, однак деякі з них зручніші, інші - ні. Кожен такий показник можна характеризувати цілим комплексом властивостей, серед яких можна назвати наступні, важливі для окресленої предметної області і частково вже висвітлені вище:

- стійкість значень показника при однаковому стані користувача;

- незначний розкид значень показника при зміні стану людини;

- простота визначення показника з технічної точки зору.

Цим вимогам повністю задовольняють три показника з тільки що розглянутих: середній час утримання деяких кнопок, що часто набираються, середній час набору деяких поширених біграм, відносна кількість помилок при наборі тексту. Зазначені показники в цілому можна охарактеризувати словами «клавіатурний почерк»; розглянемо його докладніше, як показник особистості користувача, що підходить для задачі його розпізнавання.

Завдання набору тексту зустрічається при роботі на комп'ютері досить часто, незалежно від характеру програм, з якими найчастіше має справу користувач. Найбільш повно цей вид діяльності проявляється при роботі в текстових редакторах типу Microsoft Word. Однак не слід думати, що набір тексту обмежений тільки цим додатком, адже насправді такий вид роботи зустрічається повсюдно:

- при обміні повідомленнями в соціальних мережах та Інтернет-пейджерах (типу Viber або ICQ)

- при пошуку інформації, коли спочатку завжди слід ввести її текстовий опис;

- при внесенні інформації в бази даних або електронні таблиці;

- навіть при тривіальному вході на сайт, що вимагає авторизації, необхідно набрати на клавіатурі свій логін і пароль, а це вже цілком конкретний набір символів (хоча і не дуже великий).

Грубо кажучи, можна сказати, що набір тексту - одна з основних функцій персонального комп'ютера. І для нас важливо, що кожен користувач має свій неповторний більш-менш стабільний клавіатурний почерк. Його стабільність порушується хіба що при зміні стану користувача: втоми, нервозності і т.д. Таким чином, цей показник ідеально підходить для цілей даного дослідження.

Формалізуючи поняття клавіатурного почерку, можна сказати, що це набір середніх значень інтервалів часу утримань окремих клавіш, а також інтервалів між двома послідовними натисканнями на різні пари клавіш. Під клавішами маються на увазі кнопки, на які нанесені всі літери українського або англійського тексту (або того, з яким доводиться працювати людині). Такі інтервали для звичайних користувачів (що мають певний досвід роботи на клавіатурі) вимірюються в мілісекундах. Надалі будемо розглядати український алфавіт, як такий, що є досить поширеним в повсякденному застосуванні у досить великій частині Інтернету. Таким чином, клавіатурний почерк в першому наближенні є сукупністю двох комплексних математичних об'єктів:

- матриці набору біграм;

- вектора утримань окремих клавіш.

Матриця набору біграм має вигляд $\boldsymbol{T}_{32x32}$ (по числу букв алфавіту), де t_{ij} - інтервал часу, який проходить з моменту набору i-тої літери алфавіту до натискання наступної за нею j-ої літери (в тому випадку, коли ці літери йдуть послідовно одна за одною - тобто утворюють біграми):

$$\boldsymbol{T}_{32x32} = \begin{pmatrix} 0 & 123 & 222 & ... & 291 \\ 90 & 0 & 346 & ... & 123 \\ 113 & 198 & 0 & ... & 73 \\ ... & ... & ... & ... & ... \\ 140 & 229 & 304 & ... & 0 \end{pmatrix} \qquad (2.1)$$

Розглянемо матрицю (2.1), згідно з якою, наприклад, t_{12} = 123, і це означає, що середній час між натисканням клавіші «а» (перша буква алфавіту, тому що перший індекс дорівнює 1 і «б» (друга буква алфавіту, так як другий індекс дорівнює 2) становить 123 мс. Далі, наприклад $t_{1\ 32}$ = 291, тобто середній час між натисканням на клавішу «а», а потім відразу на «я» становить 291 мс.

Спочатку матриця ***T*** заповнюється в перший раз: при створенні профілю користувача і (бажано) коли він перебував в хорошому робочому настрої. При цьому йому пропонується виконати набір великого текстового фрагменту (як мінімум одна, а краще декілька сторінок). Оскільки цей процес виконується лише одноразово, то відповідні великі витрати часу на його виконання можна вважати обгрунтованими підвищеною точністю визначення почерку ***T***.

Далі кожен раз при необхідності оцінки стану користувача слід попросити його набрати на клавіатурі який-небудь текст поменше (припустимо з одного-двох абзаців). При цьому формується матриця ***T'*** поточних значень часів набору біграм. Дві матриці ***T'*** і ***T*** порівнюються і по «відстані» між ними визначається ступінь відхилення поточного користувача від еталонного. Якщо дане відхилення перевищує певний поріг, автентифікацію не можна вважати успішною.

Відстань між матрицями може оцінюватися різними способами, наприклад як відстань в 1024-вимірному (32x32 = 1024) евклідовому просторі (евклідова норма):

$$\Delta = \sqrt{\sum_{i=1}^{32}\sum_{j=1}^{32}\left(t'_{ij} - t_{ij}\right)^2}\ . \qquad (2.2)$$

Основний недолік (2.2) - отримання норми в абсолютних одиницях. Це означає, що більш рідкісні біграми, на набір яких, отже, витрачається більше часу, можуть давати велику різницю і сильніше впливати на загальний результат, ніж ті, що більш часто зустрічаються при наборі (отже, які користувач набирає швидше, з меншими t_{ij}, і, отже, з меншими абсолютними відмінностями). Така ситуація звичайно ж неприпустима, і самий простий і зручний спосіб уникнути її - перейти до підсумовування відносних величин:

$$\Delta = \sqrt{\sum_{i=1}^{32}\sum_{j=1}^{32}\left(\frac{t'_{ij} - t_{ij}}{t_{ij}}\right)^2} \quad . \qquad (2.3)$$

Перевага формули (2.3) полягає в тому, що результат не залежить ані від розмірності величин, які підсумовуються (хоч це в даному прикладі і не дуже важливо, так як підсумовування підлягають тільки інтервали часу, однак при урахуванні інших за своєю природою чинників, це стане надзвичайно важливим), ані від їх співвідношення. Тому (2.3) являє більш адекватну оцінку «відстані» між двома матрицями клавіатурного почерку - поточної ***T'*** і еталонної ***T***.

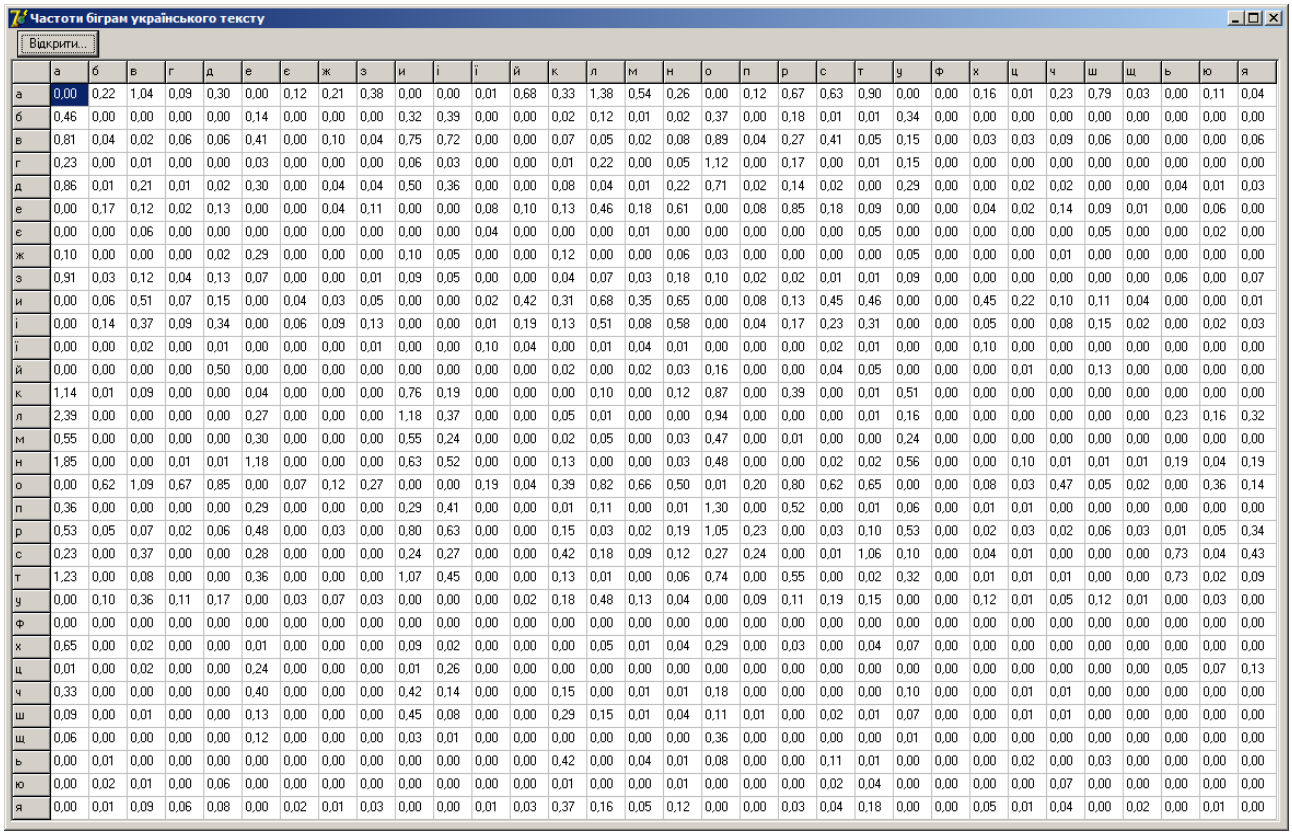
Цікавим є питання про величину тексту, що підлягає набору, при створенні еталону і при поточній перевірці. З точки зору статистики кількість текстової інформації для набору при створенні еталону ***T*** може бути точно пораховано згідно з критерієм Стьюдента, однак, з огляду на те, що число біграм дуже велике (32x32 = 1024), то і обсяг тексту, який потрібно набрати при високому рівні значущості (наприклад, для перевірки гіпотези з ймовірністю 0,99) буде досить великим. В реальній практиці не вийде змусити користувача набирати такий великий текст. Тому, в даному випадку слід керуватися критерієм розумної достатності, а не строгими математичними критеріями. Таким чином, зупиняємося на 1-2 сторінках тексту при знятті еталону та 0,5 стр. при поточній перевірці. Крім того, наведені нижче міркування

дозволяють значно звузити кількість вимірюваних величин, що, загалом, піднімає точність статистичної обробки при малому обсязі вибірки.

Дійсно, виходячи з логічного аналізу тексту українською мовою, слід зазначити, що деякі біграми не можуть зустрічатися в ньому в принципі (наприклад, «ИИ», «ЩЩ», «АЬ» і багато інших), а ще більше число біграм зустрічається досить рідко. Біграм, що зустрічаються більш-менш часто, насправді не так вже й багато. Це важливо з тих міркувань, що показує інформаційну надмірність матеріалу матриці $\boldsymbol{T}_{32x32}$.

Для аналізу стану користувача за допомогою клавіатурного почерку слід попросити його набрати на клавіатурі деякий текст. Для набору хоч якої-небудь правдоподібної статистики цей текст не повинен бути занадто малий, але, навіть якщо це буде великий абзац, біграми, які рідко зустрічаються, навряд чи взагалі там будуть присутні. Це означає, що без втрати точності оцінки клавіатурного почерку можна замість заповнення і порівняння всієї повної матриці $\boldsymbol{T}$, оперувати з деяким вектором, який буде складатися з N часів набору деяких фіксованих біграм. Логічно вибрати N біграм, які найбільш часто зустрічаються в текстах українською мовою.

Для визначення кількості і конкретного типу біграм, які найбільш часто зустрічаються в українських текстах, реалізуємо невелику допоміжну програму, яка аналізує текстові файли. Програму напишемо в поширеному середовищі програмування Delphi. Єдиною вхідною інформацією для програми буде файл з текстом українською мовою обсягом не менше 100 Кб. Результатом роботи програми буде таблиця частоти біграм в цьому тексті – рис. 2.3.



	а	б	в	г	д	е	є	ж	з	и	і	ї	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ю	я
а	0,00	0,22	1,04	0,09	0,30	0,00	0,12	0,21	0,38	0,00	0,00	0,01	0,68	0,33	1,38	0,54	0,26	0,00	0,12	0,67	0,63	0,90	0,00	0,00	0,16	0,01	0,23	0,79	0,03	0,00	0,11	0,04
б	0,46	0,00	0,00	0,00	0,00	0,14	0,00	0,00	0,00	0,32	0,39	0,00	0,00	0,02	0,12	0,01	0,02	0,37	0,00	0,18	0,01	0,01	0,34	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
в	0,81	0,04	0,02	0,06	0,06	0,41	0,00	0,10	0,04	0,75	0,72	0,00	0,00	0,07	0,05	0,02	0,08	0,89	0,04	0,27	0,41	0,05	0,15	0,00	0,03	0,03	0,09	0,06	0,00	0,00	0,00	0,06
г	0,23	0,00	0,01	0,00	0,00	0,03	0,00	0,00	0,00	0,06	0,03	0,00	0,00	0,01	0,22	0,00	0,05	1,12	0,00	0,17	0,00	0,01	0,15	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
д	0,86	0,01	0,21	0,01	0,02	0,30	0,00	0,04	0,04	0,50	0,36	0,00	0,00	0,08	0,04	0,01	0,22	0,71	0,02	0,14	0,02	0,00	0,29	0,00	0,00	0,02	0,02	0,00	0,00	0,04	0,01	0,03
е	0,00	0,17	0,12	0,02	0,13	0,00	0,00	0,04	0,11	0,00	0,00	0,08	0,10	0,13	0,46	0,18	0,61	0,00	0,08	0,85	0,18	0,09	0,00	0,00	0,04	0,02	0,14	0,09	0,01	0,00	0,06	0,00
є	0,00	0,00	0,06	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,04	0,00	0,00	0,00	0,01	0,00	0,00	0,00	0,00	0,00	0,05	0,00	0,00	0,00	0,00	0,00	0,05	0,00	0,00	0,02	0,00
ж	0,10	0,00	0,00	0,00	0,02	0,29	0,00	0,00	0,00	0,10	0,05	0,00	0,00	0,12	0,00	0,00	0,06	0,03	0,00	0,00	0,00	0,00	0,05	0,00	0,00	0,00	0,01	0,00	0,00	0,00	0,00	0,00
з	0,91	0,03	0,12	0,04	0,13	0,07	0,00	0,00	0,01	0,09	0,05	0,00	0,00	0,04	0,07	0,03	0,18	0,10	0,02	0,02	0,01	0,01	0,09	0,00	0,00	0,00	0,00	0,00	0,00	0,06	0,00	0,07
и	0,00	0,06	0,51	0,07	0,15	0,00	0,04	0,03	0,05	0,00	0,00	0,02	0,42	0,31	0,68	0,35	0,65	0,00	0,08	0,13	0,45	0,46	0,00	0,00	0,45	0,22	0,10	0,11	0,04	0,00	0,00	0,01
і	0,00	0,14	0,37	0,09	0,34	0,00	0,06	0,09	0,13	0,00	0,00	0,01	0,19	0,13	0,51	0,08	0,58	0,00	0,04	0,17	0,23	0,31	0,00	0,00	0,05	0,00	0,08	0,15	0,02	0,00	0,02	0,03
ї	0,00	0,00	0,02	0,00	0,01	0,00	0,00	0,00	0,01	0,00	0,00	0,10	0,04	0,00	0,01	0,04	0,01	0,00	0,00	0,00	0,02	0,01	0,00	0,00	0,10	0,00	0,00	0,00	0,00	0,00	0,00	0,00
й	0,00	0,00	0,00	0,00	0,50	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,02	0,00	0,02	0,03	0,16	0,00	0,00	0,04	0,05	0,00	0,00	0,00	0,01	0,00	0,13	0,00	0,00	0,00	0,00
к	1,14	0,01	0,09	0,00	0,00	0,04	0,00	0,00	0,00	0,76	0,19	0,00	0,00	0,00	0,10	0,00	0,12	0,87	0,00	0,39	0,00	0,01	0,51	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
л	2,39	0,00	0,00	0,00	0,00	0,27	0,00	0,00	0,00	1,18	0,37	0,00	0,00	0,05	0,01	0,00	0,00	0,94	0,00	0,00	0,00	0,01	0,16	0,00	0,00	0,00	0,00	0,00	0,00	0,23	0,16	0,32
м	0,55	0,00	0,00	0,00	0,00	0,30	0,00	0,00	0,00	0,55	0,24	0,00	0,00	0,02	0,05	0,00	0,03	0,47	0,00	0,01	0,00	0,00	0,24	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
н	1,85	0,00	0,00	0,01	0,01	1,18	0,00	0,00	0,00	0,63	0,52	0,00	0,00	0,13	0,00	0,00	0,03	0,48	0,00	0,00	0,02	0,02	0,56	0,00	0,00	0,10	0,01	0,01	0,01	0,19	0,04	0,19
о	0,00	0,62	1,09	0,67	0,85	0,00	0,07	0,12	0,27	0,00	0,00	0,19	0,04	0,39	0,82	0,66	0,50	0,01	0,20	0,80	0,62	0,65	0,00	0,00	0,08	0,03	0,47	0,05	0,02	0,00	0,36	0,14
п	0,36	0,00	0,00	0,00	0,00	0,29	0,00	0,00	0,00	0,29	0,41	0,00	0,00	0,01	0,11	0,00	0,01	1,30	0,00	0,52	0,00	0,01	0,06	0,00	0,01	0,01	0,00	0,00	0,00	0,00	0,00	0,00
р	0,53	0,05	0,07	0,02	0,06	0,48	0,00	0,03	0,00	0,80	0,63	0,00	0,00	0,15	0,03	0,02	0,19	1,05	0,23	0,00	0,03	0,10	0,53	0,00	0,02	0,03	0,02	0,06	0,03	0,01	0,05	0,34
с	0,23	0,00	0,37	0,00	0,00	0,28	0,00	0,00	0,00	0,24	0,27	0,00	0,00	0,42	0,18	0,09	0,12	0,27	0,24	0,00	0,01	1,06	0,10	0,00	0,04	0,01	0,00	0,00	0,00	0,73	0,04	0,43
т	1,23	0,00	0,08	0,00	0,00	0,36	0,00	0,00	0,00	1,07	0,45	0,00	0,00	0,13	0,01	0,00	0,06	0,74	0,00	0,55	0,00	0,02	0,32	0,00	0,01	0,01	0,01	0,00	0,00	0,73	0,02	0,09
у	0,00	0,10	0,36	0,11	0,17	0,00	0,03	0,07	0,03	0,00	0,00	0,00	0,02	0,18	0,48	0,13	0,04	0,00	0,09	0,11	0,19	0,15	0,00	0,00	0,12	0,01	0,05	0,12	0,01	0,00	0,03	0,00
ф	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
х	0,65	0,00	0,02	0,00	0,00	0,01	0,00	0,00	0,00	0,09	0,02	0,00	0,00	0,00	0,05	0,01	0,04	0,29	0,00	0,03	0,00	0,04	0,07	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
ц	0,01	0,00	0,02	0,00	0,00	0,24	0,00	0,00	0,00	0,01	0,26	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,05	0,07	0,13
ч	0,33	0,00	0,00	0,00	0,00	0,40	0,00	0,00	0,00	0,42	0,14	0,00	0,00	0,15	0,00	0,01	0,01	0,18	0,00	0,00	0,00	0,00	0,10	0,00	0,00	0,01	0,01	0,00	0,00	0,00	0,00	0,00
ш	0,09	0,00	0,01	0,00	0,00	0,13	0,00	0,00	0,00	0,45	0,08	0,00	0,00	0,29	0,15	0,01	0,04	0,11	0,01	0,00	0,02	0,01	0,07	0,00	0,00	0,01	0,01	0,00	0,00	0,00	0,00	0,00
щ	0,06	0,00	0,00	0,00	0,00	0,12	0,00	0,00	0,00	0,03	0,01	0,00	0,00	0,00	0,00	0,00	0,00	0,36	0,00	0,00	0,00	0,00	0,01	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
ь	0,00	0,01	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,42	0,00	0,04	0,01	0,08	0,00	0,00	0,11	0,01	0,00	0,00	0,00	0,02	0,00	0,03	0,00	0,00	0,00	0,00
ю	0,00	0,02	0,01	0,00	0,06	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,01	0,00	0,00	0,01	0,00	0,00	0,00	0,02	0,04	0,00	0,00	0,00	0,00	0,07	0,00	0,00	0,00	0,00	0,00
я	0,00	0,01	0,09	0,06	0,08	0,00	0,02	0,01	0,03	0,00	0,00	0,01	0,03	0,37	0,16	0,05	0,12	0,00	0,00	0,03	0,04	0,18	0,00	0,00	0,05	0,01	0,04	0,00	0,02	0,00	0,01	0,00

Рис. 2.3. Вікно допоміжної програми, написаної для аналізу частоти біграм, що зустрічаються в українському тексті.

Висхідний текст програми наведено в лістингу 2.1.

Лістинг 2.1.

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics,
Controls, Forms,
  Dialogs, StdCtrls, Grids, StrUtils;
type
  TForm1 = class(TForm)
    OpenDialog1: TOpenDialog;
    StringGrid1: TStringGrid;
    Button1: TButton;
    procedure Button1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
const N=32;
var
  Form1: TForm1;
```

```
    alf:string;

    implementation
    {$R *.dfm}
    procedure TForm1.Button1Click(Sender: TObject);
    var f:TextFile;
    bi:array[1..N+1,1..N+1]of real;
    prevs,nexts:char;
    i,j,all:integer;
    begin
      all:=0;
      if(OpenDialog1.Execute)then
        begin
          for i:=0 to N do
            for j:=0 to N do
              bi[i,j]:=0;
          AssignFile(f,OpenDialog1.FileName);
          Reset(f);
          Read(f,prevs);
          while(not Eof(f))do
            begin
              Read(f,nexts);
              i:=Pos(prevs,alf);
              j:=Pos(nexts,alf);
              if(i<>0)and(j<>0)then
                begin
                  bi[i,j]:=bi[i,j]+1;
                  all:=all+1;
                end;
              prevs:=nexts;
            end;
          CloseFile(f);
          for i:=1 to N do
            for j:=1 to N do

StringGrid1.Cells[j,i]:=FloatToStrF(bi[i,j]/all*100,ffFixed,5,2);
        end;
    end;

    procedure TForm1.FormCreate(Sender: TObject);
    var i:integer;
    begin
      alf:='абвгдеєжзиіїйклмнопрстуфхцчшщьюя';
      for i:=1 to N do
        begin
          StringGrid1.Cells[0,i]:=alf[i];
          StringGrid1.Cells[i,0]:=alf[i];
        end;
    end;

    end.
```

В отриманій матриці частот виберемо 10 біграм, які зустрічаються в тексті найбільш часто – табл. 2.2 (аналіз виконано на прикладі твору «Кайдашева сім'я» загальним обсягом близько 270 Кб).

Табл.2.2. Частоти 10 самих поширених біграм українського тексту.

Біграма	ЛА	НА	АЛ	ПО	ТА	НЕ	ЛИ	ОВ	ТИ	СТ
Відносна частота, %	2,39	1,85	1,38	1,30	1,23	1,18	1,18	1,09	1,07	1,06

Аналізувати будемо середній час набору цих $N = 10$ біграм за допомогою вектора:

$$\vec{t} = \{t_{ЛА}, t_{НА}, t_{АЛ}, t_{ПО}, t_{ТА}, t_{НЕ}, t_{ЛИ}, t_{ОВ}, t_{ТИ}, t_{СТ}\}.$$

За методикою зазначеної вище, спочатку необхідно один раз сформувати еталонний вектор, а потім при кожній необхідності перевірки стану користувача слід формувати поточний вектор. Відстань між векторами будемо шукати за допомогою міри Евкліда, згідно з формулою, аналогічною (2.3):

$$\Delta = \sqrt{\sum_{i=1}^{N} \left(\frac{t'_i - t_i}{t_i} \right)^2}. \qquad (2.4)$$

Величина Δ визначає розбіжність між еталоном, знятим в безпечному середовищі, і поточним користувачем.

Для практичної реалізації можна запропонувати формулу, по суті подібну (2.4), однак більш просту для програмування. Для спрощення в чисельнику (2.4) можна складати НЕ квадрати з подальшим знаходженням кореня, а модулі різниць $t_i' - t_i$. Ділити модулі таких різниць також слід не на квадрати, а саму величину t_i:

$$\Delta = \sum_{i=1}^{N} \frac{|t'_i - t_i|}{t_i}. \qquad (2.5)$$

Також, можна шукати не суму, а середнє арифметичне відносних нев'язок, позбавляючись, таким чином, від залежності величини Δ від кількості аналізованих величин N:

$$\Delta = \frac{1}{N}\sum_{i=1}^{N}\frac{\left|t'_i - t_i\right|}{t_i}. \qquad (2.6)$$

В результаті обчислень по (2.6) отримаємо число у відносних одиницях (якщо помножити на 100% - то у процентах), яке показує, на скільки відсотків в середньому відрізняється еталонний користувач від поточного, що бажає пройти авторизацію (по 10 показникам - часам набору обраних біграм).

Крім розглянутих характеристик клавіатурного почерку, пов'язаних з набором біграм, також будемо враховувати час набору однієї літери δt_i, тобто різницю між моментом відпускання кнопки і моментом її натискання:

$$\delta t_i = t_{\text{відп } i} - t_{\text{нат } i}, \qquad (2.7)$$

где $t_{\text{відп } i}$ – момент часу, коли клавіша поточної *i*-тої літери була відпущена;

$t_{\text{нат } i}$ – момент часу, коли клавіша поточної *i*-тої літери була натиснута.

Усі такі часи δt_i утримання *i*-тої літери усереднюються за час набору авторизаційного тексту і потім порівнюються зі значеннями δt_i', які зберігаються в еталонному файлі відповідного користувача.

Тут, аналогічно попередньому комплексу характеристик, пов'язаних з набором біграм, виникає питання контролю всіх або тільки деяких з усіх доступних елементів такого типу. Іншими словами: чи контролювати утримання всіх без винятку букв, або вибрати тільки деякі, які найбільш часто зустрічаються. З огляду на те, що деякі букви типу "ф", "щ", "ш", "ц", "х" зустрічаються досить рідко, недоцільно контролювати будь-які характеристики, з ними пов'язані, з огляду на велику імовірність повної відсутності таких букв у тексті для набору. У якості найбільш статистично значущих можна вибрати літери, що утворюють вищенаведені біграми (табл. 2.2), що програмно реалізується у вигляді змінної, що містить весь «алфавіт», з якого скомбіновані ці біграми:

```
alf="авеилностп";
```

Оскільки букв в цьому «алфавіті» досить багато (також 10), то з метою спрощення програмної реалізації ці ж самі букви і можна контролювати на предмет часу утримання окремих клавіш.

На підставі часів (2.7) будемо формувати вираз виду (2.6), але не для часів набору біграм, а для часів утримання обраних букв:

$$\delta\Delta = \frac{1}{N}\sum_{i=1}^{N}\frac{\left|\delta t'_i - \delta t_i\right|}{\delta t_i}. \qquad (2.8)$$

Таким чином, на підставі часів утримання обраних 10 букв формується ще одне середнє арифметичне нев’язок (2.8).

І ще однією досить важливою характеристикою клавіатурного почерку є інтенсивність здійснення помилок, друкарських помилок, тобто невірного набору. Так, загальновідомо, що будь-який оператор технічного пристрою (в т.ч. і звичайний пересічний користувач ПК) в процесі роботи може робити помилки, що, зокрема, залежить від його ступеня володіння ПК в цілому. Втім іноді люди, що дуже повільно набирають текст, роблять це занадто ретельно і практично не роблять помилок, правда, набираючи при цьому всього декілька букв за хвилину. Але і в цьому випадку інтенсивність виникнення помилок можна використовувати як елемент автентифікації, важливо, щоб цей показник контролювався в комплексі з раніше розглянутими, і тоді він може дати досить корисний вклад до всієї методики оцінки згідно біометричного протоколу.

При комп'ютерному наборі кількість помилок можна оцінювати за двома напрямками:

- помилки, які виникли в процесі набору, але були помічені і виправлені;

- помилки, які виникли в процесі набору і не були помічені або були помічені, але з якоїсь причини не були виправлені.

Помилки другого типу контролювати автоматично досить складно з алгоритмічної точки зору, тому в подальшому будемо говорити тільки про помилки першого типу, тобто такі, які виправлялися відразу ж під час набору тексту користувачем. Виправляти помилки можна різними способами, однак найбільш традиційним при послідовному наборі тексту є натискання клавіші

BackSpace кілька разів, щоб стерти останні, набрані невірно, символи і дістатися до помилки.

Отже, кількість виправлених помилок може бути оцінений по натиснень на кнопку Backspace. В цьому випадку потрібно враховувати тільки одне з декількох поспіль натискань кнопки повернення, що випливає з таких міркувань. Середній користувач набирає текст не по одному символу, а групами. Якщо в групі символів є хоча б одна помилка, користувач знайде її не відразу, а тільки після набору всієї групи, коли подивиться на екран. Як вже зазначалося раніше, зазвичай помилка виправляється відразу, шляхом натискання кілька разів клавіші Backspace для того, щоб повернутися до символу, який містить помилку. Таким чином, всі ці численні натискання викликані однією помилкою, і тому натискання Backspace, які йдуть поспіль, повинні асоціюватися з однією помилкою. Ще один варіант виправлення помилки - стрілками «вліво» - «вправо» на клавіатурі встановити курсор прямо перед неправильним символом і натиснути Backspace один раз. При цьому одній помилці відповідає одне натискання кнопки повернення.

Таким чином, без особливого збитку для точності можна вважати, що для однієї помилки, допущеної при комп'ютерному наборі тексту, відповідає від 1 до M поспіль натискань на кнопку Backspace (при такому підході ми нехтуємо поправками помилок за допомогою клавіші Delete, яка, з огляду на її відокремлене становище на клавіатурі - на відміну від кнопки BackSpace, яка інтегрована в блок буквених клавіш, використовується зазвичай для редагування вже набраного тексту, але не для виправлення помилок при безпосередньому наборі).

При реалізації алгоритму будемо збільшувати лічильник виправлених помилок E на одиницю в тому випадку, якщо натиснута кнопка Backspace, і безпосередньо перед нею була натиснута інша кнопка. Якщо натиснута кнопка Backspace, але попередня натиснута клавіша також Backspace, то лічильник помилок збільшувати не потрібно, тому що, ймовірно, йде повернення до одного конкретного неправильно введеному символу. Абсолютне число

помилок *E* потрібно поділити на загальне число набраних символів *All*, щоб перейти до відносного числа помилок, яке і можна порівнювати з еталоном:

$$R = \frac{E}{All}.$$

В результаті обчислення кількості помилок отримаємо число *R*, яке також знімається під час первинного набору тексту (тоді ж, коли формується і еталон набору біграм і утримань окремих клавіш). Як результат порівняння будемо приймати відносне збільшення числа помилок за вказаним показником:

$$\Delta_R = \frac{|R' - R|}{R}. \qquad (2.9)$$

Формулу (2.9) слід об'єднати з формулами (2.6) і (2.8) так, щоб інтегрувати всі фактори, які враховуються (час набору біграм, час утримання окремих клавіш, відносне число помилок - все разом це і буде формувати клавіатурний почерк в рамках даного дослідження):

$$\Delta_{res} = \frac{\Delta + \delta\Delta + \Delta_R}{3} = \frac{1}{3}\left(\frac{1}{N}\sum_{i=1}^{N}\frac{|t'_i - t_i|}{t_i} + \frac{1}{N}\sum_{i=1}^{N}\frac{|\delta t'_i - \delta t_i|}{\delta t_i} + \frac{|R' - R|}{R} \right). \qquad (2.10)$$

У (2.10) три різних за характером нев’язки усереднюються (щоб не відхилятися від середніх, об'єктивних величин), хоча теоретично ці три групи показників можуть бути по-різному важливі для виконання процесу автентифікації. Таким чином, замість формули (2.10) в перспективі можна використовувати формулу виду:

$$\Delta_{res} = a_1\Delta + a_2\delta\Delta + a_3\Delta_R, \qquad (2.11)$$

де a_i - вагові коефіцієнти розглянутих ознак, які можуть бути використані для автентифікації по клавіатурному почерку. Для визначення цих вагових коефіцієнтів потрібне окреме об'ємне дослідження, тому в даній роботі обмежимося залежністю (2.10), хоча перспективною є і формула (2.11).

Отже, візьмемо за основу (2.10) при реалізації протоколу автентифікації в ІКМ біометричним методом, а саме, на базі аналізу клавіатурного почерку користувача.

2.2. Дослідження можливості удосконалення обраного протоколу автентифікації в ІКМ

Розглянутий у попередньому розділі спосіб біометричної автентифікації працює на локальному ПК та потребує удосконалення для впровадження у розподілених системах типу «клієнт-сервер», що відображене на рис. 2.4.

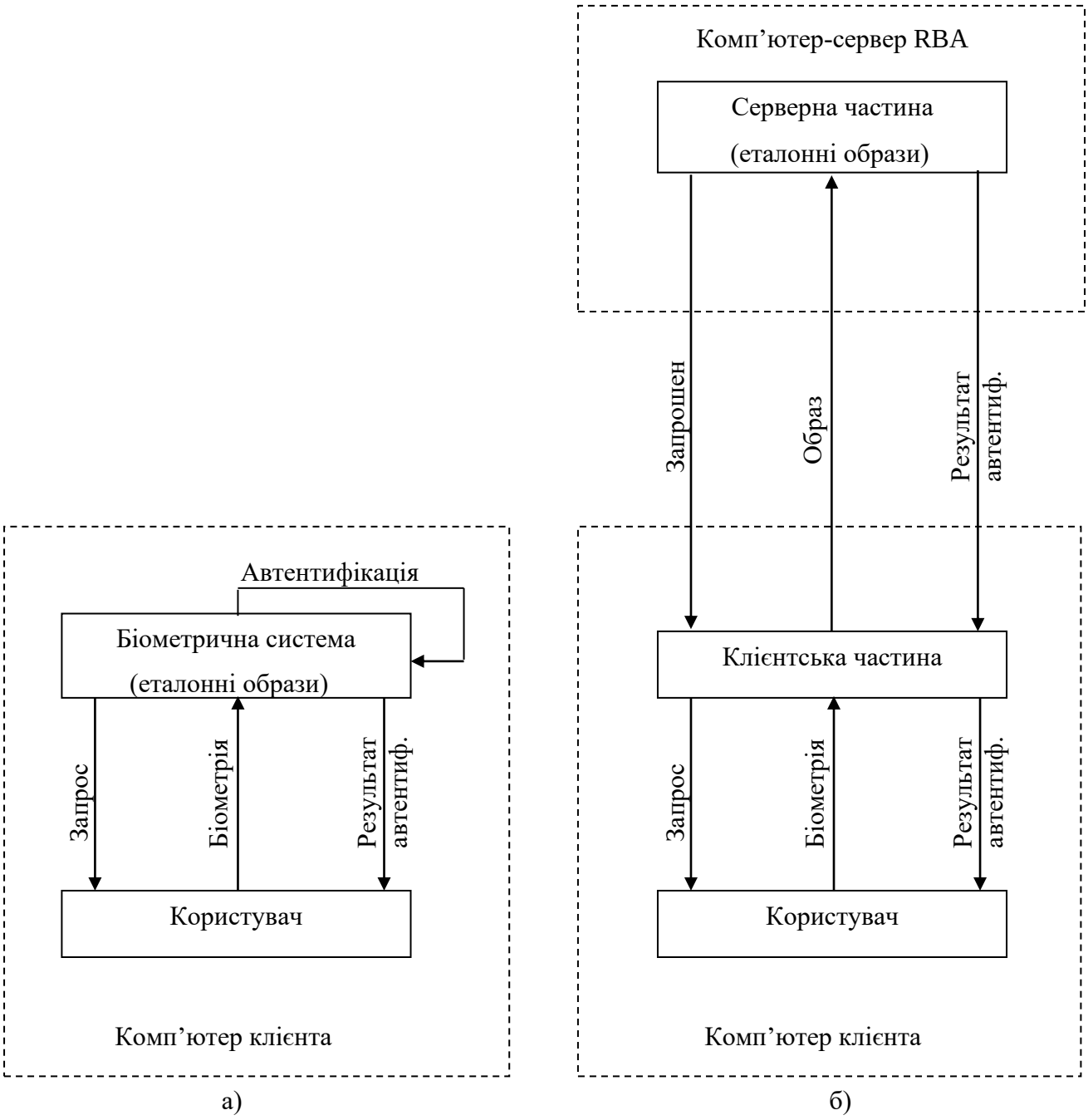


Рис. 2.4. Впровадження розподіленого режиму у протоколі біометричної автентифікації RBA.

У локальній схемі застосування біометричних протоколів автентифікації (рис. 2.4, *а*) еталонні образи зберігаються на тому ж комп'ютері, де і працює клієнт. Це значно зменшує стійкість системи, адже порівняно просто ці еталонні образи можуть бути відкриті, проаналізовані та імітовані. Якщо ж еталони зберігаються на віддаленому комп'ютері, як запропоновано у розробленому протоколі RBA, то отримати до них доступ можна лише зламавши сервер, який є іншою ланкою захисту (а не нашою розробкою RBA). Відповідно, стійкість самого запропонованого протоколу у цій частині (доступність еталонів, загроза їх аналізу) є забезпеченою.

Також недоліком локальної схеми є можливість внесення змін логічного типу у саму процедуру автентифікації, адже код, що їй відповідає, також знаходиться на комп'ютері клієнта. Отже, цей код може бути модифікований, причому за тією ж схемою, за якою традиційно зловмисники вносять зміни до пропрієтарного програмного забезпечення, щоби несанкціоновано його використовувати. А саме способом, при якому якийсь один висококваліфікований зловмисник пише програму-патч, що вносить зміни до платного програмного забезпечення на двійковому рівні (на рівні окремих команд асемблера, тобто машинних кодів), а потім її поширює, в результаті чого і усі, навіть малокваліфіковані зловмисники, можуть модифікувати ПЗ на тих комп'ютерах, де їм потрібно обійти систему захисту. Зазвичай модифікація зводиться до пошуку у дизасембльованому двійковому коді останнього умовного оператора, після якого виконується вердикт, чи законною є робота програми на даному комп'ютері (або, відповідно, чи пройдена автентифікація за локальним біометричним протоколом), та подальшої зміни цього оператора на логічно протилежний (якщо це JZ, то ставиться JNZ; якщо JE – ставиться JNE, і т.д.).

Таким чином, при розміщенні виконуваного коду, який відповідає за власне біометричну автентифікацію, на комп'ютері клієнта, завжди існує імовірність внесення змін у логіку роботи відповідної програми (шляхом завантаження програми-патчу, або ручного виправлення необхідних команд

асемблера у програмі-дизасемблері). На жаль при цьому навіть самі хитромудрі алгоритми біометричної автентифікації стають безсилими перед грубою фізичною модифікацією коду.

Запропонована схема протоколу RBA дозволяє позбутися і цієї загрози, адже згідно схеми рис. 2.4, *б* уся процедура автентифікації алгоритмічно проводиться на сервері, а клієнтові лише надсилається повідомлення, чи була автентифікація успішною.

І ще однією загрозою, яка існує при локальному застосуванні протоколу біометричної автентифікації, але нейтралізується у запропонованій схемі RBA, є підробка самого повідомлення про успішну аутентифікацію. Дійсно, зловмисник може відтрасувати процес проведення автентифікації і встановити спосіб передачі повідомлення про її успіх від однієї частини програмного продукту до іншої. Стандартними способами такої передачі, що можуть бути реалізовані програмістом, та незаконно використані хакером, є:

- виклик якоїсь функції із одним аргументом, якщо автентифікація успішна, та з іншим, якщо навпаки. При цьому зловмисник може написати власний код, що викликатиме потрібну функцію із «успішним» аргументом, а роботу біометричної СКД взагалі вимкнути;

- встановлення значення якоїсь змінної (наприклад, встановлення Auth=true), що також може бути здійснено хакером самостійно, без успішної роботи системи біометричної автентифікації: йому, наприклад, достатньо подивитися адресу у пам’яті, де зберігається змінна Auth і вручну, чи за допомогою програми-патча пам’яті, змінити вміст цієї комірки;

- інші варіанти, які в принципі також можуть імітуватися зловмисником і без роботи підсистеми біометричної автентифікації.

Усі ці способи дадуть свій результат, оскільки, зокрема, ресурси, що захищаються, знаходяться на тому ж комп’ютері, де завантажується і клієнтська частина (в одній оперативній пам’яті, на одному жорсткому диску, і т.п.).

У запропонованому протоколі віддаленої біометричної автентифікації така ситуація із підробкою способу сигналізації, що автентифікація була успішною, є безглуздою, оскільки усі бажані ресурси знаходяться на віддаленому комп’ютері, отже підробка повідомлення від нього до клієнта не внесе змін на самому сервері, і, отже, не надасть доступу до його ресурсів (лише користувач зможе побачити повідомлення, що авторизація була успішною, але подальшого реального доступу до ресурсів серверу у нього не буде).

Таким чином, перевагами запропонованого протоколу віддаленої біометричної автентифікації є наступні:

- підвищений захист еталонної інформації, яка розміщена на сервері, а не у клієнта;

- жодне підконтрольне клієнту програмне забезпечення доступу до еталонної інформації не має;

- програмний код, що реалізує безпосередньо процедуру біометричної автентифікації весь повністю розміщений на сервері, а клієнт до нього доступу не має, отже нейтралізується загроза злочинної модифікації цього коду для винесення потрібного зловмисникові позитивного рішення про автентифікацію;

- функція клієнта полягає тільки у збиранні поведінкової біометричної інформації (причому якою саме вона повинна бути зловмисник не може знати) та отриманні від сервера вердикту про результат автентифікації (який відображується людині-користувачу);

- підробка повідомлення про успішну автентифікацію не має сенсу, оскільки не призведе до отримання доступу до ресурсів серверу, а лише введе користувача клієнтської частини в оману (покаже, що авторизація була успішною, але реального доступу не буде).

Таким чином, запропонований протокол віддаленої біометричної автентифікації є надзвичайно ефективним і нейтралізує чимало загроз, які існують при локальному режимі застосування, а тому можна переходити до питань, пов’язаних із його програмною реалізацією.

2.3. Обґрунтування вибору засобів розробки для реалізації удосконаленого протоколу автентифікації в ІКМ

Перед виконанням будь-якої програмної реалізації слід визначитися із кількома концептуальними питаннями, пов’язаними із нею, а саме:

- яку технологію програмування найкраще застосувати для реалізації даного алгоритму при даному комплексі умов;
- яку мову програмування, що підтримує обрану технологію програмування, доцільніше всього застосувати для програмної реалізації у наявних умовах;
- яке середовище розробки (або комплекс простих окремих засобів розробки) краще всього застосувати для даної програмної реалізації.

Тільки отримавши обґрунтовані відповіді на ці запитання, можна переходити безпосередньо до етапу програмування.

2.3.1. Вибір технології розробки з урахуванням особливостей предметної галузі.

Отже, першочерговим питанням, яке постає перед розробниками практично будь-якого програмного забезпечення, є вибір моделі або технології його розробки. Такими, що реально широко використовуються на сьогоднішній день у виробничій практиці, є технології структурного (процедурного) та об’єктно-орієнтованого програмування. Кожна з них має свої особливості, переваги і недоліки, які розглянемо докладніше.

Структурне, або як його ще називають, процедурне програмування засноване на використанні окремих структурних блоків - в першу чергу, підпрограм (процедур і функцій).

Історично перші комп’ютерні програми були відносно простими і мали пакетний режим роботи: отримуючи на вхід якусь інформацію (можливо, навіть на перфокарті) вони виконували певний обсяг операцій з обробки цих даних і видавали результат. При цьому існувала сувора функціональна залежність виходу від входу – рис. 2.5.

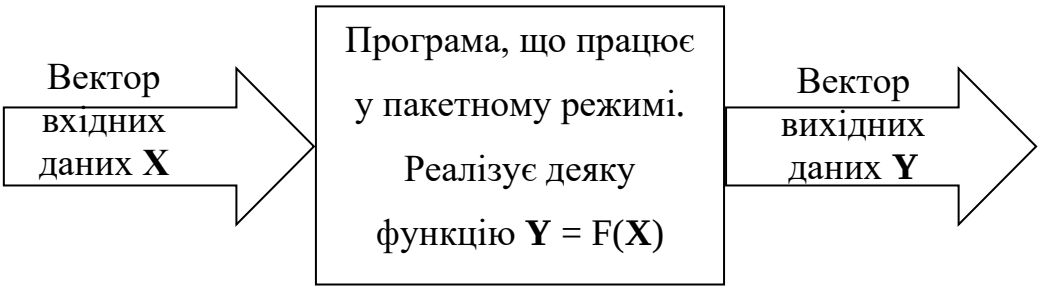


Рис. 2.5. Схема роботи пакетної програми.

Незважаючи на відсутність явного зв’язку функціональності і структури, пакетні програми в основному мали просту лінійну послідовність виконання. Тобто в них були практично відсутні будь-які підпрограми, принаймні, використання підпрограм не було важливим елементом самої методики програмування.

В міру ускладнення функціональності програмного забезпечення, змінювалася і його внутрішня структура: поступово розвинулася інтерактивна модель взаємодії користувача і програми – рис. 2.6. Програми стали запитувати інформацію і активно реагувати на дії людини. Ускладнення функціональності привело до відповідного ускладнення програмного коду, який, в першу чергу, став досить обширним у розмірах. Обширним для того, щоби середньостатистична людина-програміст без всяких спеціальних хитрувань швидко і легко розібралася з незнайомим кодом. Розміщувати код у простій лінійній послідовності без виділення великих блоків стало незручно, в першу чергу, для ефективного розуміння цього коду.

Тут слід зазначити психологічні особливості сприйняття людиною складних «великих» завдань. Неструктуроване «велике» завдання (наприклад, написання дипломної роботи) зазвичай викликає певний психологічний ступор і, як результат, повну неможливість поступово розібратися з ним. Людині зручно розбити проблему на не надто велику (зазвичай до десятка, а краще 3-4) кількість завдань (наприклад, розділів у дипломній роботі), не замислюючись про реалізацію кожного з них. Коли є ясність і розуміння проблеми на найвищому рівні абстракції, слід приступати до деталізації підзадач, кожну з

яких слід розбити на окремі «підпідзадачі» тобто підзадачі нижчого рівня, більш дрібні. Уже після такого розбиття слід аналізувати всі перераховані підзадачі. На певному етапі зупиняються і виконують не розбиття чергової підзадачі на більш дрібні, а безпосередню її реалізацію в програмних кодах.

Рис. 2.6. Схема роботи програми, що активно взаємодіє з користувачем.

Отже, можна сказати, що розвиток методик програмування відбувався паралельно з розвитком призначеного для користувача інтерфейсу: і грубо кажучи, розвиненому інтерфейсу командного рядка відповідає парадигма структурного програмування.

Говорячи більш формалізовано, структурне програмування має на увазі побудову програми відповідно до трьох основних принципів: слідування, розгалуження, повторення.

Слідування означає, що оператори і блоки програмного коду слідують і виконуються один за іншим. Розгалуження реалізується різними умовними операторами типу if, і дозволяє вибирати один з декількох подальших варіантів виконання програми. Повторення зазвичай відносять на рахунок циклів (багаторазових повторів однієї і тієї ж ділянки коду, що йдуть підряд), хоча цей же принцип можна віднести і до підпрограм.

Взагалі ж, структурне програмування іноді називають застарілою методикою програмування, на зміну якій разом з віконним інтерфейсом

прийшло об’єктно-орієнтоване програмування (деякі сучасні мови програмування загального призначення навіть не дозволяють створити структурну програму, тільки об’єктно-орієнтовану – як, наприклад, Visual C# чи Java). Проте, при створенні невеликих програм (наприклад, до 10000 рядків коду і без передбачуваного розширення) застосування цієї методики програмування однозначно більш виправдано і відповідний код набагато краще сприймається, ніж його об’єктно-орієнтований варіант.

Суть же методології об’єктно-орієнтованого програмування полягає в тому, що система розглядається, як сукупність окремих сутностей - об’єктів, які мають набір якихось своїх внутрішніх параметрів - властивостей, а також можуть взаємодіяти між собою за допомогою деяких дій - викликів методів (або трохи більше непрямим чином - шляхом надсилання повідомлень, оброблюваних методами об’єктів; для цього необхідна присутність активної сутності, яка роздає повідомлення адресатам, як, наприклад, менеджер вікон в ОС Windows).

Якщо говорити про програмний код, то для того, щоб оперувати об’єктом, його спочатку потрібно створити. Об’єкти створюються як змінні, у яких типом виступає клас об’єкта. Клас - це просто опис, які властивості можуть мати об’єкти такого типу (тобто яку інформацію вони можуть зберігати), і які у них є методи (тобто які дії вони можуть виконувати). Об’єкт - це набір значень, чому саме рівні властивості даного об’єкта (свої методи кожен об’єкт отримує від свого класу, тобто методи однакові у всіх об’єктів, що належать даному класу).

Для чого потрібен цей специфічний підхід, адже самі по собі об’єкти не додають нічого корисного (навпаки, введення об’єктів ускладнює програму, вносить в неї нові сутності)? Виявляється, реалізовуючи всі сутності, необхідні, згідно з алгоритмом, для роботи програми, у вигляді класів і об’єктів, ми спрощуємо її розуміння для самих себе. Саме тому ОО-підхід рекомендується до застосування для великих проектів (більше десятків тисяч рядків коду), коли утримувати «в голові» всю систему цілком стає важко. Можна сказати, що

розбиття програми на об’єкти і проектування їх класів наближає розуміння предметної області до звичного людського образу мислення (в разі «великих» проектів). Людина мислить класами, об’єктами і зв’язками між ними.

Порівняльна складність проектів, що мають однакову функціональність, але побудованих по-різному (згідно структурному і об’єктно-орієнтованого підходів до програмування), як функція їх обсягу показана на рис. 2.7. З графіка рис. 2.7 слід, що, якщо потрібно реалізувати продукт з невеликою функціональністю (тобто кількість рядків коду, що її реалізують буде очевидно невеликою, порядку кількох тисяч рядків), то це краще робити без застосування класів, так як вони будуть тільки ускладнювати всю справу. Якщо ж програма має більш-менш значну функціональність, а значить, реалізується хоча б кількома тисячами рядків коду, то вже є сенс замислюватися про застосування об’єктно-орієнтованого підходу. Однозначно будуватися за принципами ООП повинні програми, які мають 10000 рядків коду і більш.

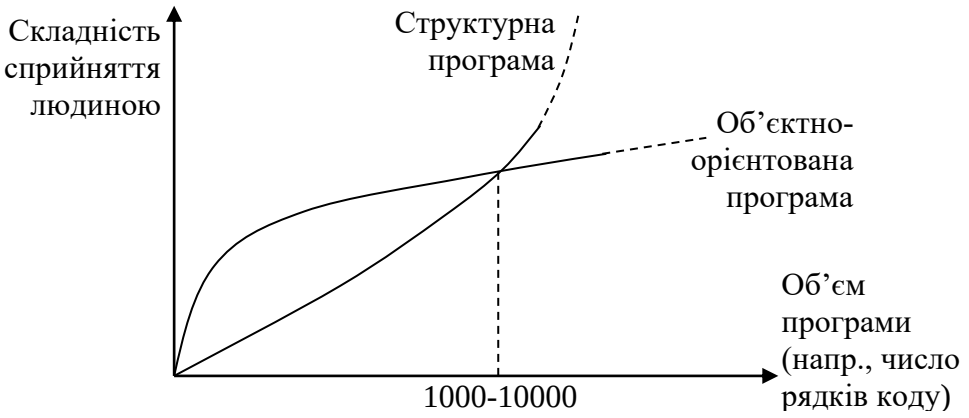


Рис. 2.7. Порівняльна складність висхідного тексту двох програм, що мають однакову функціональність, але реалізованих по-різному: згідно об’єктно-орієнтованому та структурному підходам.

Відзначимо, що часто крім розглянутих міркувань, також на вибір методики програмування впливають інші чинники, наприклад, можливість майбутнього розширення функціональності, створення якомога більш зрозумілого коду (для роботи над проектом цілої команди, а не одного програміста), або просто побажання замовника застосувати найбільш сучасний

підхід до програмування (чи навпаки, дозвіл на використання вікових, перевірених часом технологій).

Крім розбиття (декомпозиції) всієї предметної області на об’єкти (класи) і співвідношення між ними, також ОО-підхід має на увазі дотримання трьох основних його принципів: інкапсуляція, наслідування, поліморфізм.

Під інкапсуляцією мається на увазі об’єднання даних (значення властивостей класу у деякого конкретного об’єкта) та засобів їх обробки (методи класу). Це знову ж таки зручно психологічно, так як дозволяє реалізовувати окремі завершені сутності - класи, які самі обробляють свої дані. Звернення до об’єктів цих класів відбувається за допомогою методів, що утворюють інтерфейс класу.

Наслідування дуже корисно, тому що дозволяє сильно скоротити обсяги повторюваного коду (до чого потрібно завжди прагнути при розробці будь-якого програмного забезпечення). Згідно з цим принципом виділяється клас, який має загальний набір властивостей і методів для декількох більш розширених класів. Цей клас оголошується батьком, базовим класом для декількох похідних від нього (нащадків, спадкоємців). Всі класи-нащадки успадковують від базового всі його властивості та методи, але до цього ще мають свої власні оригінальні властивості і / або методи.

Наприклад, клас Студент є похідним від класу Людина, тому що кожна людина має властивість Ім’я, Прізвище, метод Відпочити(). Однак у Студента є свої специфічні властивості і / або методи, які як раз і відрізняють його від просто Людини: СереднійБал, НомерЗаліковки, ЗдатиЕкзамен(), і т.д.

При наслідуванні іноді методи батьківського і похідного класу мають однакове призначення, але реалізовуються по-різному. Такі методи називаються перевантаженими. Наприклад, метод Відпочити() у класу Людина реалізується як відпочинок на дивані, а у класу Студент - як похід в клуб. При цьому ще раз підкреслимо, що призначення методу в обох випадках одне і те саме.

Поліморфізм є можливістю деякої функції приймати об’єкти як батьківського, так і похідних класів, і вміти викликати перевантажені методи саме того класу, об’єкт якого був переданий в функцію. Слід сказати, що це досить специфічна можливість і загалом багато програмістів використовують ОО-підхід і без звернення до поліморфізму.

Нехай, наприклад, у програмі є функція ПровестиВихідні(), припустимо яка не належить якомусь класу (хоча це не принципово). Нехай аргументом цієї функції є об’єкт класу Людина. Тоді в неї можна передавати об’єкти всіх похідних від Людини класів: Студент, Службовець, Пенсіонер, і т.д., тому що всі вони є Людиною (спадкоємці цього класу). Ясно, що ця функція повинна включати різні дії: ПрибратиКвартиру(), ПітиНаРинок(), і в тому числі Відпочити(). Так ось поліморфізм дозволяє всередині цієї функції просто вказати назву методу Відпочити(), не вказуючи якого саме класу він повинен бути викликаний, а вже в процесі виконання програми, якщо в функцію переданий об’єкт класу Студент, то викликається саме його метод Відпочити(), а якщо переданий об’єкт класу Пенсіонер, то автоматично викликається саме його метод Відпочити(), і т.д. Кажуть, що функція ПровестиВихідні() - поліморфна, і вона є такою завдяки тому, що реалізує принцип поліморфізму.

Важливими поняттями в ООП також є: статичні члени класу, абстрактні методи і класи, дружба функцій і класів, і т.д.

Грунтуючись на перерахованих особливостях двох існуючих методів програмування, вибираємо об’єктно-орієнтований підхід, як більш сучасний, що відповідає середнім масштабам проектованого ПЗ, а також вимогам до його складності.

2.3.2. Вибір мови програмування

На сьогоднішній день існує чимало мов програмування, які підтримують об’єктно-орієнтовану парадигму (у більш-менш повній мірі, тобто з поліморфізмом та іншими, більш сучасними нововведеннями галузі об’єктно-орієнтованого аналізу). Із популярних на сьогоднішній день мов можна назвати C++, Java, C#.

Перший варіант, мова C++, є надзвичайно потужною, яка має стабільну популярність вище середнього рівня, але і досить складною для сприйняття; фактично, це інструмент для професіоналів найвищого рівня. В той же час на сьогоднішній день більш популярними серед широкої спільноти програмістів середнього рівня є мови Java та C#. Мова Delphi (або точніше – середовище розробки, оскільки у ньому використовується мова програмування Object Pascal), не зважаючи на свої зручність та простоту (а, можливо, навпаки, через них), стабільно не має широкої популярності серед професійних розробників програмного забезпечення, а використовується в основному для малих проектів та навчальних цілей.

Таким чином, при реальному процесі вибору мови програмування для професійної розробки на сьогоднішній день слід розглядати дві укрупнених альтернативи: мова C++ з одного боку та одна з мов – Java та C# – з іншого. Дуже багато полеміки було присвячено порівнянню цих двох останніх мов, але беззаперечним фактом є значна перевага продукту від Microsoft над конкурентом у частині зведення користувацького інтерфейсу. У цьому питанні середовище Visual Studio надає практично той же рівень зручності, що й Delphi, але використовує популярні мови, а не Паскаль, до якого традиційно укорінилося ставлення як до навчальної мови програмування для новачків. Отже, зважаючи на необхідність створення програмного продукту для операційної системи Windows, доцільніше використовувати C#, а не Java.

Якщо ж порівнювати C++ та C#, то перша мова є більш пристосованою для системних задач (як от організація зв’язку по TCP-IP, численні математичні операції, відслідковування подій засобів введення-виведення, а саме, клавіатури, і т.д.). Таким чином, кінцево обираємо мову програмування C++, що найкраще відповідає технічному завданню на розробку в рамках даного дослідження. Коротко розглянемо особливості цієї мови програмування.

Якщо коротко говорити про характеристики мови програмування C++, на якій була розроблена програмна система, то слід сказати, що вона є високорівневою компільованою мовою загального призначення зі строгою

типізацією, яка підходить для створення самих різних додатків, починаючи від настільних програм, до веб-додатків і, навіть, мобільних застосувань. Беручи до уваги те, що на C/C++ можна впроваджувати асемблерні вставки, можна сказати, що цією мовою можна реалізувати практично усе, тобто будь-який програмний продукт. Таким чином, суттєвим плюсом мови C/C++ є її універсальність в частині створення додатків для різноманітних цільових платформ.

Також мова C/C++ стала настільки поширеною, що саме від неї пішли майже усі інші поширені мови програмування (зокрема, ті ж самі Java та C#), які так і називають: «сі-подібні».

Також слід зазначити, що на сьогоднішній день C/C++ є однією із найпопулярніших і найпоширеніших мов у світі, а професійні програмісти цією мовою отримують чи не найвищі заробітні плати в галузі IT.

Суттєвою перевагою мови C++ є наявність стандартної бібліотеки шаблонів (STL), у якій зведені разом усі популярні структури даних (створені у вигляді окремих класів), а також реалізовані базові алгоритми роботи із цими структурами. Зокрема, як відомо, основними алгоритмами є пошук та сортування – ці, та багато інших операцій, присутні для усіх структур даних бібліотеки STL. Серед таких структур можна назвати:

- масив (array);
- вектор (vector);
- список (list);
- дека, або черга (deque);
- хеш (hash) та ін.

Беручи до уваги усі плюси мови C/C++ вона прийнята за основну для виконання програмної реалізації у даному дослідженні.

2.3.3. Вибір середовища розробки

Перейдемо до розгляду конкретних програмних засобів, які можуть використовуватися для процесу розробки на C++. Основним із них, що

дозволяє реалізовувати ОО-проекти цією мовою, можна вважати середовище розробки Microsoft Visual Studio (у ньому нас цікавитиме можливість створення C++ проекту). Microsoft Visual Studio - це серія продуктів фірми Майкрософт, які включають інтегроване середовище розробки програмного забезпечення та ряд інших інструментальних засобів. Ці продукти дозволяють розробляти як консольні програми, так і програми з графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms, а також веб-сайти, веб-застосунки, веб-служби як в рідному, так і в керованому кодах для всіх платформ, що підтримуються Microsoft Windows, Windows Mobile, Windows Phone, Windows CE, .NET Framework, .NET Compact Framework та Microsoft Silverlight. Важливим фактом про це професійне середовище розробки є те, що у нього присутні безкоштовні версії, що дуже зручно для законної розробки у ньому некомерційних програмних продуктів, зокрема таких, що виконуються в рамках магістерських досліджень. В той же час недоліком середовища є якраз його направленість на надмірно професійну розробку: воно перевантажене непотрібними опціями, про свідчить хоча би розмір цієї програми, яка у повній версії наближається до 60 Гб (!).

Ще одним середовищем розробки для мови C/C++ є Borland C++ Builder (з 2008 року – просто C++ Builder). Це середовище є дуже зручним, але через слабку маркетингову політику фірми-розробника (Embarcadero Technologies) значна частка програмістів вважає, що даний продукт загинув ще у 00-х роках XXI ст. Насправді цей програмний продукт успішно розвивається і остання його версія датується 01.11.2019 р. Характерною особливістю цього середовища розробки (як і усіх програмних продуктів, що розроблялися компанією Borland) була і є надзвичайна простота у вирішенні деяких складних (але рутинних) питань, які встають перед будь-якими розробниками мовою C++. Наприклад, за рахунок використання власного типу String надзвичайно спрощена робота з текстовими рядками, у порівнянні з класичним C/C++. Головним же плюсом цього середовища розробки є величезна автоматизація проблеми створення ефективного інтерфейсу користувача програми.

Програмісту тут достатньо просто перетаскувати візуальні компоненти на форму (заготовку вікна програми) і задавати їх властивості та дії у візуальному режимі. Це набагато простіше, ніж використовувати одну із бібліотек Microsoft для зведення користувацького інтерфейсу (як, наприклад, MFC, або ATL), і як результат економить велику кількість часу на зведення інтерфейсу.

Також можливим є використання одного із компіляторів мови C/C++, що підтримуються open-source спільнотою програмістів, наприклад GCC, або DJ++ Compiler, однак ці продукти (як і велика кількість інших продуктів, пов’язаних з цим об’єднанням) страждають складністю роботи з ними, незручностями, які у комплексі спричинюють суттєві втрати робочого часу на виконання рутинних дій, які у тому ж C++ Builder виконуються за лічені хвилини, чи навіть швидше).

Таким чином, зважаючи на усі переваги, у якості основного інструменту для подальшої розробки обираємо інтегроване середовище розробки C++ Builder.

Розділ 3. ПРОЄКТНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ УДОСКОНАЛЕНОГО ПРОТОКОЛУ АВТЕНТИФІКАЦІЇ В ІКМ

3.1. Особливості алгоритмічних складових удосконаленого протоколу автентифікації в ІКМ

В запропонованому у попередньому розділі рішенні (рис. 2.4) змінено саму послідовність процесу виконання автентифікації (тобто фактично – протокол її проведення).

Алгоритм автентифікації у новому варіанті виглядає наступним чином:

1) користувач, якому потрібний доступ до ресурсів віддаленого серверу, завантажує на своєму комп'ютері програму-клієнт (клієнтську частину програмного комплексу, що реалізує розроблений протокол RBA);

2) у своїй клієнтській частині він вводить свій логін, під яким раніше він реєструвався на цільовому сервері;

3) після введення логіну, користувач набирає у нижньому текстовому полі довільний текст українською мовою;

4) якщо користувач не знає, який текст набирати, то у якості орієнтиру він може завантажити у верхнє текстове поле будь-який текстовий документ українською мовою, для чого натискає кнопку «Обрати текст…». Даний пункт не є обов'язковим, якщо користувач набирає текст по пам'яті, або із друкованої книжки, що є у нього поруч із комп'ютером;

5) після введення достатньої кількості символів (у програмі це обмеження виставлене на 200 літер), стає активною кнопка «ГОТОВО», при натисненні на яку програма-клієнт пробує приєднатися до сервера;

6) якщо спроба під'єднання була невдалою, або користувач вирішив перервати процес набоур тексту з інших причин, то він може натиснути кнопку «Почати заново», в результаті чого усі його показники обтуляться і він зможе розпочати весь процес автентфиікації заново;

7) Сервер весь час очікує під'єднань клієнтів і при приєднанні нового з них, приймає від нього спочатку його логін (рядок із 128 байтів), а потім 10

чисел типу float (тобто всього 60 байтів), які відповідають часам набору 10 найпоширеніших біграм, а потім ще 10 чисел типу float, які рівні середнім часам утримання 10 найбільш уживаних літер українського алфавіту;

8) після отримання імені та поточного образу користувача сервер виконує пошук еталонного файлу із вказаним ім’ям. Якщо такий файл не знайдено, клієнтові передається відповідь про неуспішну авторизацію, а також робиться відповідний запис у єдиному текстовому полі прогарми-серверу;

9) якщо передане клієнтом ім’я користувача зареєстроване на сервері (що проявляється у наявності на ньому текстового файлу з таким же самим ім’ям), із цього файлу зчитуються еталонні характеристики і за формулою (2.10) здійснюється розрахунок середнього відхилення характеристик поточного користувача від відповідного еталону;

10) отримане середнє відхилення порівнюється із граничним значенням (прийнято на рівні 26 %) і робиться висновок про результат автентифікації;

11) отриманий результат надсилається клієнтській частині, яка відображує його своєму користувачеві.

За таким алгоритмом функціонує весь розподілений комплекс, що реалізує протокол віддаленої біометричної автентифікації в інформаційно-комунікаційних мережах, програмну реалізацію якого слід розглянути докладно у наступному викладі.

3.2. Особливості програмної реалізації окремих складових удосконаленого протоколу автентифікації в ІКМ

Як було розроблено у попередньому розділі, програмне забезпечення складається із двох складових серверної – рис. 3.1 та клієнтської – рис. 3.2.

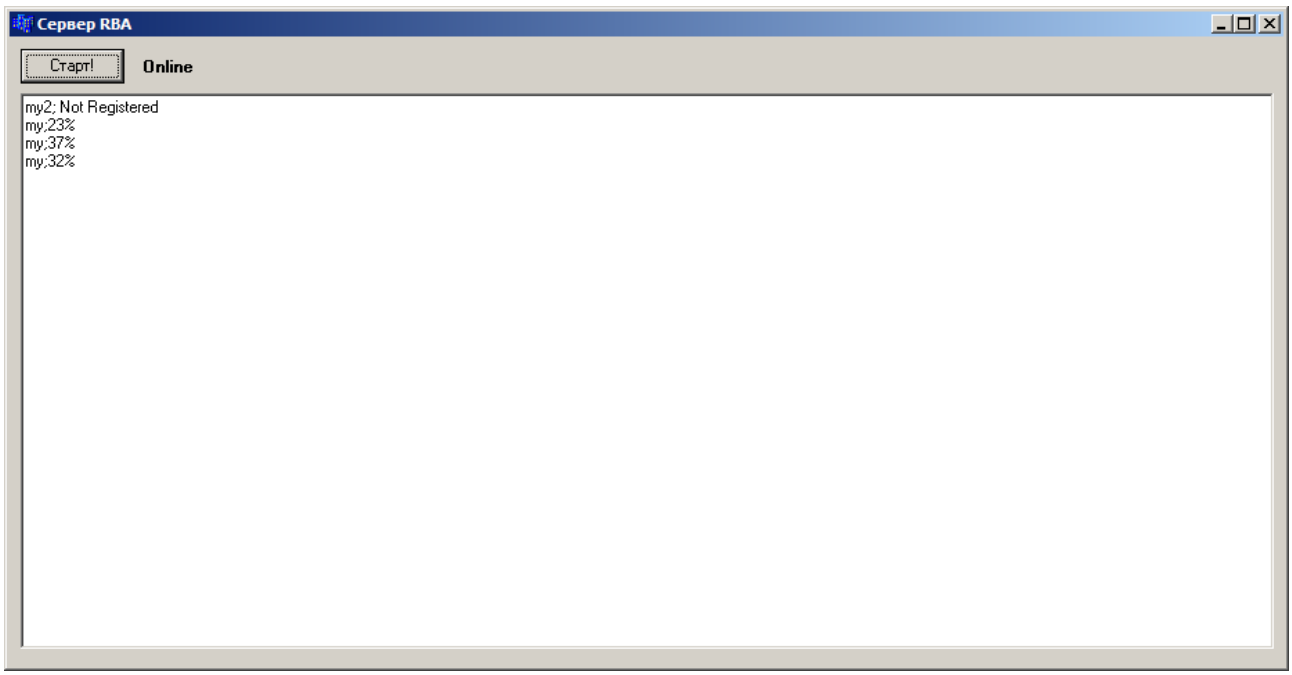


Рис. 3.1. Загальний вигляд вікна серверної частини розробленого програмного комплексу, що реалізує протокол RBA.

Слід відмітити, що програмний комплекс, реалізований у даній роботі, носить більше демонстраційний характер і не інтегрований у жодне реальне середовище із ресурсами, що захищаються. У більшій мірі це зроблено для наголошення на його універсальності, тобто можливості його застосування практично у будь-якій системі «клієнт-сервер». По-друге, така stand-alone реалізація ще й займає значно менше часу на розробку при повній демонстрації ефективності та роботоспроможності запропонованого протоколу віддаленої біометричної автентфиікації RBA. Таким чином, автор притримувався відомого у проектуванні принципу KISS, який вимагає дотримання якомога більшої простоти при виконанні поставленої задачі (тобто у роботу не слід вносити зайвих ускладнень, якими є прив’язки до конкретних програмних продуктів; тут показана лише ефективність самого протоколу, що розроблявся).

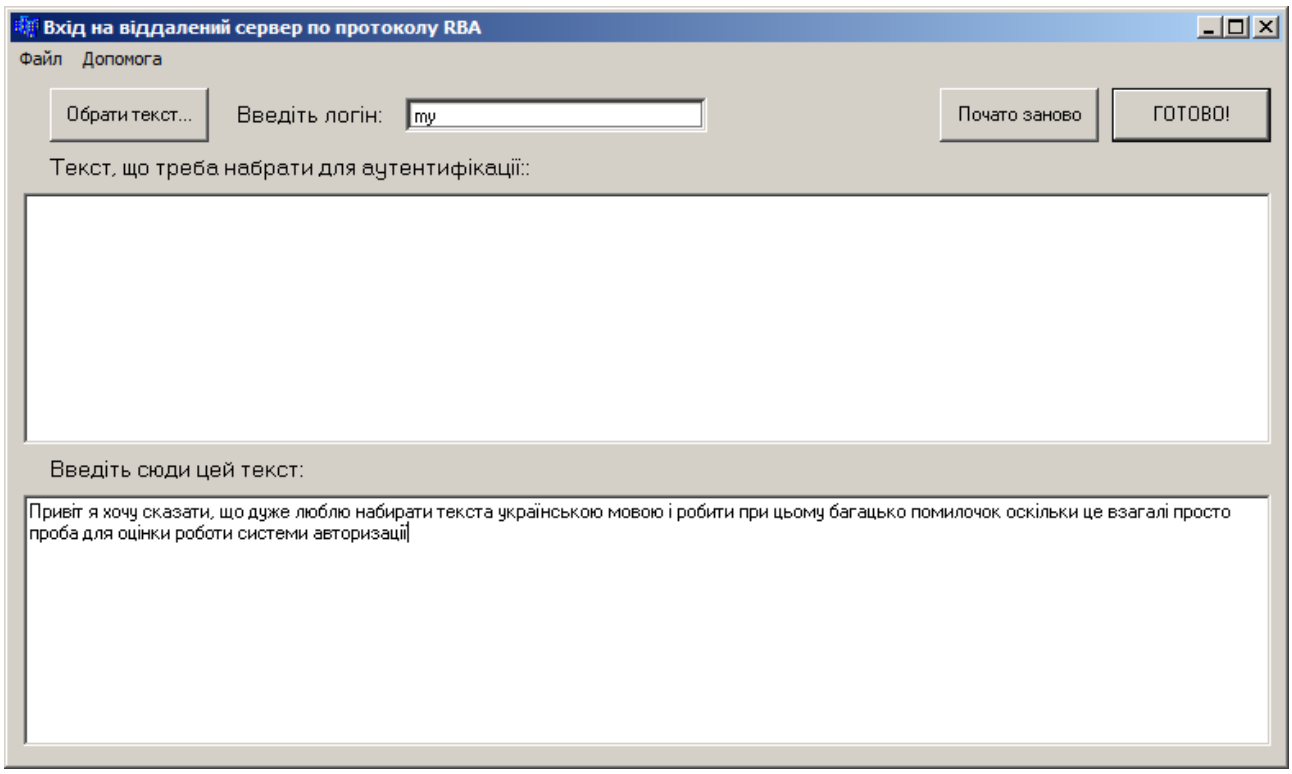


Рис. 3.2. Загальний вигляд вікна клієнтської частини розробленого програмного комплексу, що реалізує протокол RBA.

3.3. Загальна характеристика отриманого програмного продукту, опис інтерфейсу користувача, інструкція по експлуатації

На рис. 3.1 видно, що інтерфейс серверної частини достатньо простий і має фактично лише кнопку для запуску сервера (якщо він не запущений, а клієнт пробує під'єднатися до нього, то виникає помилка, показана на рис. 3.3). При виникненні такої помилки слід впевнитися, що програма-сервер завантажена (працює) і у верхній її частині відображений online статус. Якщо з'єднання немає, то слід натиснути ще раз кнопку «СТАРТ!», а потім ще раз спробувати підключитися клієнтом до сервера.

Слід відмітити, що клієнт та сервер використовують порти TCP-IP з номерами 1234 та 1235, які є вільними від використання стандартними компонентами Windows, але, як і завжди при використанні TCP-IP, потрібно слідкувати за тим, щоби ці порти не використовувалися іншими програмами (на зразок мережних комп'ютерних ігор).

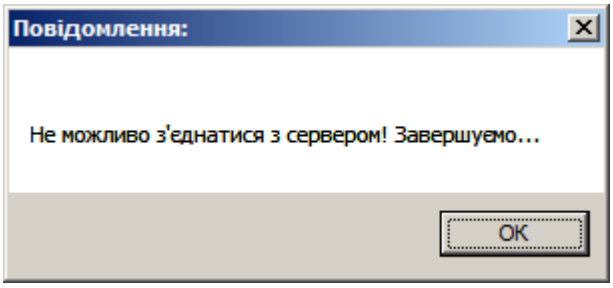


Рис. 3.3. Вікно про помилку, що виникає при спробі під’єднання за розробленим протоколом RBA до неактивного серверу.

Коли сервер знаходиться у робочому режимі, то у великому текстовому полі відображаються результати автентифікацій, які бувають трьох типів (рис. 3.1):

– успішні, коли у відповідному рядку відображується логін користувача, що успішно пройшов автентифікацію, а також відсоток відмінності його від свого еталону (причому цей відсоток менше граничного значення, яке у даній роботі прийнято на рівні 26 %);

– неуспішні, коли відображується логін користувача та велике значення відмінності його поточного образу від еталонного (тобто коли відсоток більше 26 %);

– спроби під’єднання користувачів із недійсними логінами, коли відображується названий зловмисником логін та підпис Not Registered.

Якщо клієнт вказав вірний логін, але не пройшов біометричну автентифікацію, то йому відображується вікно про провал цього процесу, яке зображено на рис. 3.4.

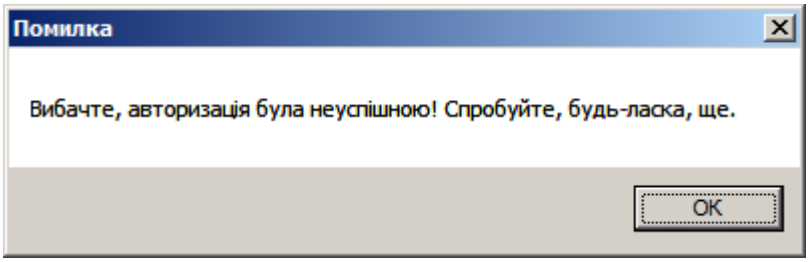


Рис. 3.4. Вікно про помилку, що виникає при спробі неуспішній автентифікації по розробленому протоколу RBA існуючого клієнта.

Якщо клієнт вказав вірний логін і пройшов біометричну автентифікацію, то йому відображується вікно про успіх цього процесу, яке зображено на рис. 3.5. За умови інтеграції розробленого програмного комплексу в реальну систему «клієнт-сервер», після цього клієнтові надається доступ до бажаних ресурсів сервера.

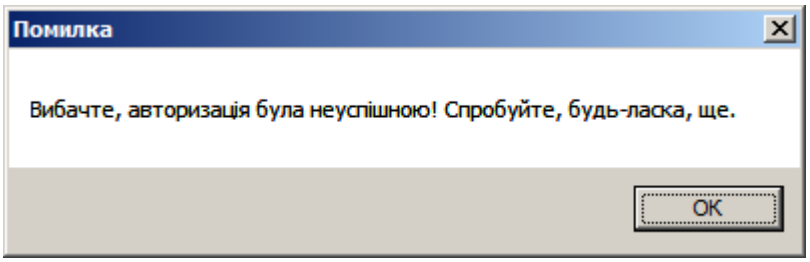


Рис. 3.5. Вікно, що виникає при успішній автентифікації по розробленому протоколу RBA існуючого клієнта.

3.4. Результати тестування розробленого програмного продукту

Тестування розподіленого програмного комплексу складалося із двох частин: оцінки роботи клієнтської та серверної частин. Встановлено, що в обох випадках робота програмного продукту є стабільною і не викликає помилок операційної системи.

Окрім тестування на стабільність роботи, звичайно, було виконано тестування на ефективність досягнення цілей роботи, що можна оцінити за допомогою спеціальних показників біометричних систем контролю доступу.

Як відомо, біометричні системи мають свої специфічні показники ефективності, які в закордонній літературі називають абревіатурами FRR та FAR, а у вітчизняній – помилкою першого роду та другого роду відповідно (як відомо, під нульовою гіпотезою завжди мається на увазі найбільш природний стан речей і, якщо мова йде про авторизацію, то нормальним є те, що її проходить законний користувач).

Помилка типу FRR (False Rejection Rate), або як її ще називають помилка першого роду («хибна тривога»), є відхиленням авторизації законного

користувача, образ якого насправді знаходиться серед еталонів на сервері під відповідним ім’ям, яке даний користувач назвав під час ідентифікації. Сам показник FRR надає імовірність такої ситуації, яка при взятті до уваги частотного змісту імовірності рівна (при завданні у відсотках) кількості осіб, яким відмовлено в авторизації, із кожної сотні таких, що намагаються її пройти.

Помилка типу FAR (False Acceptance Rate), або помилка другого роду виникає, коли система пропускає зловмисника, якому слід було відмовити в авторизації. Відповідне число (виражене у відсотках) рівне кількості разів, при яких зловмисник успішно авторизується в системі, із кожної сотні таких спроб.

В результаті аналізу створеного програмного продукту встановлено, що при указанні у (2.10) порогового значення автентифікації $\Delta = 0{,}26 = 26\%$ показник хибних відмов законним користувачам складає $FRR < 5\%$, а імовірність успішної авторизації зловмисника складає $FAR < 10\ \%$.

Як висновок можна сказати, що розроблений протокол RBA може застосовуватися для захисту шляхом розподіленої біометричної (поведінкової) автентифікації комерційної інформації середнього рівня конфіденційності (максимум – із грифом «Для службового користування»).

ВИСНОВКИ

В роботі виконано розробку та реалізації удосконаленого протоколу автентифікації для захисту інформації у інформаційно-комунікаційних мережах, тобто у розподіленому режимі. Рішення реалізовано мовою C/C++ програмно у вигляді комплексу, який працює за технологією «клієнт-сервер», тобто містить дві частини:

- клієнтська збирає поведінкові характеристики клієнта (А саме особливості його клавіатурного почерку) та надсилає на сервер;

- серверна частина порівнює отриманий образ із еталонами, що на ній зберігаються і надсилає клієнтові висновок – успішна авторизація, чи ні.

Дана робота реалізована у вигляді окремого продукту, який надає користувачеві клієнтської частини вердикт «так», чи «ні», але може бути інтегрована у будь-яку більш складну СЗІ.

В роботі проаналізовано особливості українського тексту на предмет використання його для розпізнавання клавіатурного почерку, розроблено спеціальний програмний продукт для відповідних статистичних досліджень та визначено найуживаніші біграми.

Робота являє собою завершене дослідження і може використовуватися на практиці, а також – як основа для подальших досліджень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ:

1. Богуш В. М., Довидьков О. А., Кривуца В. Г. Теоретичні основи захищених інформаційних технологій. К., 2010.
2. Вишня В. Б. Інформаційні системи та технології: підруч. Запоріжжя: ЗНУ, 2021.
3. Вишня В. Б., Ісмайлов К. Ю., Краснобрижий І. В. Інформаційні технології: підруч. Дніпро: ДДУВС, 2021.
4. Грайворонський М. В., Новіков О. М. Безпека інформаційно-комунікаційних систем. К., 2009.
5. Гуржій А. М., Возненко Л. І., Поворознюк Н. І., Самсонов В. В. Основи інформаційних технологій: навч. посіб. Київ: Літера ЛТД, 2023.
6. Комп'ютерні технології криптографічного захисту інформації на спеціальних цифрових носіях / В.К. Задірака, А.М. Кудін, В. О. Людвиченко, О. С. Олексюк. К., 2007.
7. Кравченко І. В., Микитенко В. І. Інформаційні технології: підруч. Київ: КПІ ім. Ігоря Сікорського, 2022.
8. Основи інформаційної безпеки / В.І. Андреєв, В.О. Хорошко, В.С. Чередниченко, М.Є. Шелест. К., 2009.
9. Turban E., Pollard C., Wood G. Information Technology for Management: Driving Digital Transformation to Increase Local and Global Performance, Growth and Sustainability. Hoboken: Wiley, 2020.
10. Stair R., Reynolds G. Principles of Information Systems. 13th ed. Boston: Cengage Learning, 2018.
11. Gallaugher J. Information Systems: A Manager's Guide to Harnessing Technology. 9th ed. Boston: FlatWorld, 2021.
12. Yates S. J., Rice R. E. (eds.) The Oxford Handbook of Digital Technology and Society. Oxford: Oxford University Press, 2020.
13. Khan W. Z., Ahmed E., Hakak S., Yaqoob I., Din I. U. Edge computing: A survey. Future Generation Computer Systems. 2019. 97:219-235.

14. Matheu S. N., García E., Hernández-Ramos J. L., Skarmeta A., Baldini G. A Survey of Cybersecurity Certification for the Internet of Things. ACM Computing Surveys. 2020. 53(6):131. DOI:10.1145/3410160.
15. Bertolino A., De Angelis G., Di Nitto E., Ferreira M. A Systematic Review on Cloud Testing. CM Computing Surveys. 2019. 52(5):87. DOI:10.1145/3331447.
16. Xu J., Xiong H., Chen X., Li J., Yang J. A Survey of Blockchain Consensus Protocols: Taxonomy, Performance, and Security. ACM Computing Surveys. 2023. 55(13s):269. DOI:10.1145/3579845.
17. Mosenia A., Sur-Kolay S., Raghunathan A., Jha N. B. The Internet of Things security: a survey encompassing enabling technologies, protocols, and applications. Computers & Security. 2021. 102494.
18. Sadhu P. K., Pattanayak B. K., Sahoo S. K. Internet of Things: Security and Solutions Survey. Sensors. 2022. 22(19):7433. DOI:10.3390/s22197433.
19. Zou J., Ye X., Choo K. K. R., Xiao L. Integrated Blockchain and Cloud Computing Systems: A Systematic Survey, Solutions, and Challenges. ACM Computing Surveys. 2021. 54(8):160. DOI:10.1145/3466658.
20. IEEE Digital Reality Initiative. Digital Transformation – White Paper. Piscataway: IEEE, 2019.
21. Proceedings of the IEEE. Special Issue on Edge Computing. 2019. Vol. 107, No. 8.
22. Azad N., Shahin M., Liang P., Babar M. A. DevOps critical success factors – A systematic literature review // Information and Software Technology. 2023. 160:107198.

ДОДАТОК А. Висхідний код серверної частини розробленого програмного комплексу.

```
//---------------------------------------------------------------------
-----------
#include <stdio.h>
#include <io.h>
#include <stdlib.h>
#include <winsock.h>
#include <math.h>
#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
//---------------------------------------------------------------------
-----------
#pragma package(smart_init)
#pragma resource "*.dfm"
#define SRV_PORT 1234

TForm1 *Form1;
float t[10],et[10];
float tlet[10],etlet[10];

//---------------------------------------------------------------------
-----------
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
}

void Error(int ErrNo)
{
char mess[127];
switch(ErrNo)
     {
     case 0:
          strcpy(mess,"\nCannot    initialize    WinSock    system!
Exiting...");
          break;
     case 1:
          strcpy(mess,"\nBad WinSock version! Exiting...");
          break;
     case 2:
          strcpy(mess,"\nCannot create socket! Exiting...");
          break;
     case 3:
          strcpy(mess,"\nCannot bind socket! Exiting...");
          break;
     case 4:
          strcpy(mess,"\nCannot    connect    to    the    server!
Exiting...");
          break;
```

```
     case 5:
          strcpy(mess,"\nExiting...");
          break;
     case 6:
          strcpy(mess,"\nCannot resolve host name! Exiting...");
          break;
     case 7:
          strcpy(mess,"\nCannot receive data! Exiting...");
          break;
     case 8:
          strcpy(mess,"\nCannot send data! Exiting...");
          break;
     }
MessageBox(NULL,mess,"Помилка!",0);
WSACleanup();
ExitProcess(0);
return;
}



//---------------------------------------------------------------------
-----------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  int i,sockaddr_inlength,numb;
  float sum,NumErr1=0,eErr1,eps,numeps;
  unsigned int s,s_new;
  struct sockaddr_in sin,from_sin;
  char local[]="127.0.0.1";
  char chlogin[128],res;
  WSADATA wsadata;
  String login;

  if(WSAStartup(MAKEWORD(1,1),&wsadata))
      Error(0);
  if(wsadata.wVersion!=MAKEWORD(1,1))
      Error(1);
  if((s=socket(AF_INET,SOCK_STREAM,0))==SOCKET_ERROR) //-1
      Error(2);
  memset((char*)&sin,0,sizeof(sin));
  sin.sin_family=AF_INET;
  sin.sin_addr.s_addr=inet_addr(local);
  sin.sin_port=SRV_PORT;

  if(bind(s,(struct sockaddr*)&sin,sizeof(sin))==SOCKET_ERROR)
      Error(3);
  if(listen(s,3))
      Error(4);
  Label1->Caption="Online";

  sockaddr_inlength=sizeof(from_sin);
  sum=i=0;
```

```
  while(i++==0)
      {
      do
          s_new=accept(s,(struct                sockaddr*)&from_sin,
&sockaddr_inlength);
      while(s_new<=0);
      numb=recv(s_new,(char*)chlogin,128,0);
      if(!numb)Error(7);
      numb=recv(s_new,(char*)t,sizeof(float)*10,0);
      if(!numb)Error(7);
      numb=recv(s_new,(char*)tlet,sizeof(float)*10,0);
      if(!numb)Error(7);
      login=chlogin;

      if(access((login+".txt").c_str(),0)==0)
      {
        FILE *f;
        char str[128];
        f=fopen((login+".txt").c_str(),"r");
        for(i=0;i<10;i++)
        {
            fgets(str,128,f);
            et[i]=atof(str);
            fgets(str,128,f);
            etlet[i]=atof(str);
        }
        fclose(f);

        eps=0;
        numeps=0;
        for(i=0;i<10;i++)
        {
            if((t[i]!=0)&&(et[i]!=0))
            {
                eps=eps+fabs(et[i]-t[i])/t[i];
                numeps++;
            }
            if((tlet[i]!=0)&&(etlet[i]!=0))
            {
                eps=eps+fabs(etlet[i]-tlet[i])/tlet[i];
                numeps++;
            }
        }
        if(NumErr1!=0)
        {
            eps=eps+fabs(eErr1-NumErr1)/NumErr1;
            numeps++;
        }
        eps/=numeps;
        Memo1->Lines-
>Add(login+";"+itoa(floor(eps*100),str,10)+"%");
        if(eps>0.26)
```

```
          res=0;
        else
          res=1;
      }
      else
      {
        res=0;
        Memo1->Lines->Add(login+"; Not Registered");
      }
      numb=send(s_new,(char*)&res,1,0);
      if(!numb)Error(6);

      closesocket(s_new);
      }
  closesocket(s);

 // Error(5);
  return;

}
//---------------------------------------------------------------------
-----------
```

ДОДАТОК Б. Висхідний код клієнтської частини розробленого програмного комплексу.

```
//---------------------------------------------------------------
-----------
#include <winsock.h>
#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
#include "Unit2.h"
//---------------------------------------------------------------
-----------
#pragma package(smart_init)
#pragma resource "*.dfm"

#define SRV_PORT 1234
#define CLNT_PORT 1235


String alf;
String bigrami[10];
float t[10];         //массив для общего времени всех наборов i-той
биграммы
int numbi[10]; //массив для количества раз, сколько была набрана
i-тая биграмма
                              //потом поделим t[i] на numbi[i] и
получим среднее время набора биграммы
int      numsym[10];//коды      букв,      образующих      биграммы
а,в,е,и,к,н,о,р,с,т
int CanAnalyze;
short int prevc,curc,NumPrinted,all; //Word
float NumErr1, NumErr2; //количество ошибок на единицу набранного
текста
int prevt,curt,curup;
char cursym,prevsym,cursymup;
float tlet[10];     //массив для общего времени всех наборов i-той
буквы
int numlet[10];//массив для количества раз, сколько была набрана
i-тая буква
float rest[10],reslet[10];

void Error(int ErrNo)
{
char mess[127];
switch(ErrNo)
     {
     case 0:
          strcpy(mess,"\nНе можливо ініціалізувати систему WinSock!
Завершуємо...");
          break;
     case 1:
          strcpy(mess,"\nНе та версія WinSock! Завершуємо...");
```

```
          break;
     case 2:
          strcpy(mess,"\nНе      можливо      створити      сокет!
Завершуємо...");
          break;
     case 3:
          strcpy(mess,"\nНе      можливо      зв'язати      сокет!
Завершуємо...");
          break;
     case 4:
          strcpy(mess,"\nНе   можливо   з'єднатися   з   сервером!
Завершуємо...");
          break;
     case 5:
          strcpy(mess,"\Завершуємо...");
          break;
     case 6:
          strcpy(mess,"\nНе   можливо   розпізнати   ім'я   хоста!
Завершуємо...");
          break;
     case 7:
          strcpy(mess,"\nНе можливо отримати дані! Завершуємо...");
          break;
     case 8:
          strcpy(mess,"\nНе      можливо      надіслати      дані!
Завершуємо...");
          break;
     }
MessageBox(NULL,mess,"Повідомлення:",0);
WSACleanup();
//ExitProcess(0);
}

TForm1 *Form1;
//---------------------------------------------------------------
-----------
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
}
//---------------------------------------------------------------
-----------

void __fastcall TForm1::N2Click(TObject *Sender)
{
    Form1->Close();
}
//---------------------------------------------------------------
-----------
void __fastcall TForm1::N4Click(TObject *Sender)
{
    Form2->ShowModal();
}
```

```
//---------------------------------------------------------------
-----------
void __fastcall TForm1::FormCreate(TObject *Sender)
{
  int i;
  alf="авеилностп";
  bigrami[0]="ЛА";
  bigrami[1]="НА";
  bigrami[2]="АЛ";
  bigrami[3]="ПО";
  bigrami[4]="ТА";
  bigrami[5]="НЕ";
  bigrami[6]="ЛИ";
  bigrami[7]="ОВ";
  bigrami[8]="ТИ";
  bigrami[9]="СТ";

  numsym[0]=70;
  numsym[1]=68;
  numsym[2]=84;
  numsym[3]=66;
  numsym[4]=75;
  numsym[5]=89;
  numsym[6]=74;
  numsym[7]=67;
  numsym[8]=78;
  numsym[9]=71;
  prevc=0;
  for(i=0;i<10;i++)
    t[i]=0;
}
//---------------------------------------------------------------
-----------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    TStringList *SL=new TStringList();
    int i;
    if(OpenDialog1->Execute())
    {
        Memo1->Clear();
        //SL=TStringList->Create();
        SL->LoadFromFile(OpenDialog1->FileName);
        for(i=0;i<SL->Count;i++)
            Memo1->Lines->Add(SL->Strings[i]);
        SL->Free();
    }
}
//---------------------------------------------------------------
-----------
void __fastcall TForm1::Memo2KeyDown(TObject *Sender, WORD &Key,
      TShiftState Shift)
{
  int i,j;
```

```
  String bigr;
  curt=GetTickCount();
  bigr="00";
  //if(not CanAnalyze)then exit;
  for(i=0;i<10;i++)
    if(Key==numsym[i])
    {
        cursym=alf[i+1];
        if(prevsym!='0')
        {
            bigr[1]=prevsym;
            bigr[2]=cursym;
            for(j=0;j<10;j++)
              if(AnsiCompareText(bigr,bigrami[j])==0)
              {
                  t[j]=t[j]+curt-prevt;
                  numbi[j]++;
                  break;
              }
        }
        all++;
        break;
    }
    else
      cursym='0';
  if((Key==8)&&(prevc!=8))
      NumErr1++;
  prevsym=cursym;
  prevc=Key;
  prevt=curt;
}


void __fastcall TForm1::Memo2KeyPress(TObject *Sender, char &Key)
{
  if (Memo2->Text.Length()>100)
    Button4->Enabled=true;
  else
    Button4->Enabled=false;
}
//---------------------------------------------------------------
-----------

void __fastcall TForm1::Button2Click(TObject *Sender)
{
  int i;
  for(i=0;i<10;i++)
  {
      t[i]=0;
      numbi[i]=0;
      tlet[i]=0;
      numlet[i]=0;
  }
```

```
  Memo1->Clear();
  Memo2->Clear();
  Edit1->Clear();
}
//---------------------------------------------------------------------
-----------

void __fastcall TForm1::Memo2KeyUp(TObject *Sender, WORD &Key,
      TShiftState Shift)
{
  int i;
  curup=GetTickCount();
  for(i=0;i<10;i++)
    if(Key==numsym[i])
    {
        cursymup=alf[i+1];
        if(cursym==cursymup)
        {
            tlet[i]=tlet[i]+curup-curt;
            numlet[i]=numlet[i]+1;
        }
        break;
    }
}
//---------------------------------------------------------------------
-----------

void __fastcall TForm1::Button4Click(TObject *Sender)
{


int s, min=-1,max=-1,numb,i;
struct sockaddr_in clnt_sin,srv_sin;
char local[]="127.0.0.1";
char res=-1,chlogin[128];
long double sum=0;
WSADATA wsadata;

if(WSAStartup(MAKEWORD(1,1),&wsadata)) //Version == 257
     Error(0);
if(wsadata.wVersion!=MAKEWORD(1,1))
     Error(1);
if((s=socket(AF_INET,SOCK_STREAM,0))==SOCKET_ERROR) //-1
     Error(2);
memset((char*)&clnt_sin,0,sizeof(clnt_sin));
clnt_sin.sin_family=AF_INET;
clnt_sin.sin_addr.s_addr=inet_addr(local);
clnt_sin.sin_port=htons(CLNT_PORT);
if(bind(s,(struct
sockaddr*)&clnt_sin,sizeof(clnt_sin))==SOCKET_ERROR)
     Error(3);

memset((char*)&srv_sin,0,sizeof(srv_sin));
```

```
srv_sin.sin_family=AF_INET;
srv_sin.sin_addr.s_addr=inet_addr(local);
srv_sin.sin_port=SRV_PORT;

if(connect(s,(struct                sockaddr*)&srv_sin,sizeof(struct
sockaddr))==SOCKET_ERROR)
     Error(4);

for(i=0;i<10;i++)
{
    if(numbi[i])rest[i]=t[i]/numbi[i];
    if(numlet[i])reslet[i]=tlet[i]/numlet[i];
}
strcpy(chlogin,Edit1->Text.c_str());
numb=send(s,(char*)chlogin,128,0);
if(!numb)Error(8);
numb=send(s,(char*)rest,sizeof(float)*10,0);
if(!numb)Error(8);
numb=send(s,(char*)reslet,sizeof(float)*10,0);
if(!numb)Error(8);

numb=recv(s,(char*)&res,1,0);
if(!numb)Error(7);

if(res==1)
    MessageBox(NULL,"Вітаємо,    Ви    успішно    авторизувалися    в
системі!","Вітаємо",0);

if(res==0)
    MessageBox(NULL,"Вибачте,    авторизація    була    неуспішною!
Спробуйте, будь-ласка, ще.","Помилка",0);

closesocket(s);
}
//---------------------------------------------------------------------
-----------
```