ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«УЖГОРОДСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ»
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Кваліфікаційна наукова
праця на правах рукопису

ПРИГАРА МИХАЙЛО ПЕТРОВИЧ

УДК 004.056.5

ДИСЕРТАЦІЯ
ЗАХИЩЕНА СИСТЕМА ТЕХНІЧНОЇ ПІДТРИМКИ ПРОЦЕСІВ ДИСТАНЦІЙНОГО ВОЛЕВИЯВЛЕННЯ

05.13.21 «Системи захисту інформації»
12 – Інформаційні технології

Подається на здобуття наукового ступеня кандидата технічних наук

Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ М.П. Пригара

Науковий керівник - кандидат технічних наук,
доцент Вишняков Володимир Михайлович

Київ – 2018

АНОТАЦІЯ

Пригара М.П. Захищена система технічної підтримки процесів дистанційного волевиявлення. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня кандидата технічних наук (доктора філософії) за спеціальністю 05.13.21 «Системи захисту інформації» (12 – Інформаційні технології) – Ужгородський національний університет ДВНЗ. Національний авіаційний університет МОН України, Київ, 2018.

Дисертаційна робота присвячена технічному захисту інформації (ТЗІ) у рамках системи дистанційного волевиявлення (СДВ), що гарантує збереження таємниці голосів та забезпечує свободу волевиявлення в умовах адміністративного тиску. Виявлено «слабкі місця» у захисті існуючих СДВ, зокрема специфічні загрози для інформації, відсутність протидії котрим підриває довіру громадян до її «чесної» роботи. Визначено можливості та шляхи нейтралізації цих «слабких місць».

Побудовано модель СДВ, у якій завдяки введенню необмеженої кількості користувачів з правами доступу на ознайомлення з усіма файлами і процесами на сервері, але без права на будь-яку модифікацію, забезпечена можливість виявлення всіх порушень політики безпеки щодо цілісності результатів волевиявлення та конфіденційності голосів в умовах недовіри до всіх без винятку осіб, що беруть участь у розробці, створенні та обслуговуванні СДВ. Запропоновано удосконалений метод захищеного обміну даними через Інтернет, який завдяки використанню випадкових бітових послідовностей та сумісного застосування шифру Вернама і алгоритму Диффі-Геллмана з параметрами, що гарантують стійкий захист даних, і завдяки збереженню відстані єдиності згідно з К. Шенноном, забезпечують формально обґрунтовану неможливість порушення конфіденційності даних в каналі зв’язку.

Ключові слова: системи дистанційного волевиявлення, технічний захист інформації, гарантована контрольованість програмного середовища, досконала стійкість щодо порушень конфіденційності та цілісності даних.

Список публікацій здобувача

1.В.М. Вишняков, М.П. Пригара, О.В. Воронін, «Відкрита система таємного голосування», Управління розвитком складних систем, 2014, №20, С. 110 – 115. http://urss.knuba.edu.ua/files/zbirnyk-20/22.pdf *Особистий внесок здобувача: здобувачу належить провідна ідея формування відкритих систем із загальним контролем.*

2.С.В. Бронин, Пригара М. П, «Подбор оптимального набора строительных обьектов при планировании инвестиционной деятельности в строительстве», «Будівельне виробництво» No 55 2013р. http://ndibv.kiev.ua/wp-content/uploads/2016/04/ЗМІСТ-_Будівельне-виробництво_-№55-2013р.-.pdf *Особистий внесок здобувача: здобувачу належить адаптація економічних методів планування інвестиційної діяльності для будь-яких об'єктів на прикладі будівництва.*

3.В.М. Чуприн, В.М.Вишняков, М.П. Пригара, «Генерування випадкових чисел штатними засобами хостів мережі Інтернет», Захист інформації. – 2016. – Т. 18, №4 – С. 323-335. *Особистий внесок здобувача: здобувачу належить провідна ідея формування дійсно випадкових чисел.*

4.В.М. Чуприн, В.М. Вишняков, М.П. Пригара, «Метод протидії незаконному впливу на виборців у системі Інтернет голосування», Безпека інформації. – 2017. – Т. 23, №1 – С. 7-14. *Особистий внесок здобувача: здобувачу належить розробка методу емуляції правильної роботи системи при ознаках впливу на виборця.*

5.В.М. Чуприн, В.М.Вишняков, М.П. Пригара, «Захист операційного середовища систем інтернет голосування», ЗАХИСТ ІНФОРМАЦІЇ. – Т. 19, №1 – С. 56-66. *Особистий внесок здобувача: здобувачу належить провідна ідея ізольованого ядра безпеки.*

*6.*М. П. Пригара, «Використання метода Форда-Фалкерсона для визначення перевантажених маршрутів та ресурсів комп'ютерних мереж», Новітні комп'ютерні технології. – Кривий Ріг : ДВНЗ «Криворізький національний університет», 2013. – Випуск XI. – С. 185-187.

7. М.П. Пригара, «МЕТОДИ ОПТИМАЛЬНОГО ВИБОРУ РІВНЯ ЗАХИСТУ КОМП'ЮТЕРНИХ СИСТЕМ», Наукова конференція молодих вчених, аспірантів і студентів КНУБА: тези доповідей. – в 2-х частинах. – Ч.1. – К.: КНУБА, 2011. – 212 с. http://science.knuba.edu.ua/source/archive/molodi_vcheni/molodi_vcheni_tezy_2011-1.pdf

8. М.П. Пригара, Т.О. Чайковська, «ПАТТЕРН «COMPOSITE», Збірник тез студентських доповідей : КНУБА, 2012.-222 с. http://science.knuba.edu.ua/source/archive/npk/npk_tezy_2012.pdf

9. В. М. Вишняков, М. П. Пригара, Д. М. Тарасюк, «Багаторівневий захист даних в системах розв'язання складних задач на суперкомп'ютері», НОВІТНІ КОМП'ЮТЕРНІ ТЕХНОЛОГІЇ Матеріали IX Міжнародної науково-технічної конференції NOCOTE'2011 - Київ–Севастополь, 13–16 вересня 2011 р. – К. : Мін- регіон України. С. – 36-38.

10. В.М. Вишняков, М.П Пригара, «Забезпечення свободи волевиявлення в системі Інтернет-голосування (ІГ)», Матеріали 4-ї Міжнародної наукової конферен- ції ICS-2015 «Інформація, комунікація, суспільство 2015» , С. 124 – 125. Режим доступу: World Wide Web. – URL: http://ena.lp.edu.ua: 8080/xmlui/bitstream/ handle/ntb/33187/055-124-125.pdf? sequence= 1&is Allowed=y

ABSTRACT

Prygara M. Secure system of technical support of processes of remote expression of will – Manuscript.

Thesis for a Candidate of Technical Sciences degree in specialty 05.13.21 – «Information Security Systems». – National Aviation University, Kyiv, 2018.

The dissertation is devoted to the technical protection of information within the

remote voting system (RVS), which guarantees secrecy of votes and ensures freedom of expression of will in conditions of administrative pressure. "Weaknesses" are identified in the protection of existing RVS, including specific threats to information, the lack of counteraction which undermines the trust of citizens in its "honest" work. The possibilities and ways of neutralizing these "weaknesses" are determined.

The model of the RVS was constructed, which, by introducing an unlimited number of users with access rights to reviewing all files and processes on the server, but without the right of any modification, was able to detect all violations of the security policy regarding the integrity of the results of the expression of will and the confidentiality of votes in conditions of distrust to all, without exception, persons involved in the development, creation and servicing of the RVS. An improved method of secure data exchange over the Internet is proposed, which, due to the use of random bit sequences and the coherent application of the Vername cipher and the Diffie-Hellman algorithm with parameters that guarantee the stable data protection, and, due to the preservation of the distance of unity according to K. Shannon, provide a formally grounded impossibility of violation confidentiality of data in the communication channel.

Keywords: controlled from distance voting systems over the Internet, technical priv, methods of providing of the assured testability of software environment, confidentiality and integrity of data that is passed by communication channels.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

СДВ – система дистанційного волевиявлення;

СДГ – система дистанційного голосування;

ДРВ – державний реєстр виборців;

ДВ – дистанційне волевиявлення;

ДС – функціональна послуга безпеки стійкість до відмов;

КВ – функціональна послуга безпеки конфіденційність при обміні;

НК – функціональна послуга безпеки достовірний канал;

НР – функціональна послуга безпеки реєстрація;

НСД – несанкціонований доступ;

ОС – операційна система;

ПБ – послуга безпеки;

ТЗІ – технічний захист інформації;

ФПЗ – функціональний профіль захисту;

ЦВК – центральна виборча комісія;

ID – ідентифікаційні дані виборця;

MITM – Man In The Middle (атака посередника);

NR – номер обраного виборцем режиму (1-голосування, 2-отримання довідок);

PW – пароль виборця;

A – випадкова степінь примітивного елементу поля Галуа, що отримана на боці клієнта;

B – випадкова степінь примітивного елементу поля Галуа, що отримана на сервері;

C – ключ для шифрування ендоморфним шифром Вернама;

E – відсоток статистичної похибки;

F_n – необхідна частота генерації випадкових бітів, Гц;

I_i – кількість інформації для ідентифікації виборця, біт;

I_a – кількість інформації для автентифікації виборця, біт;

I_v – кількість інформації про волевиявлення виборця, біт;

K – оцінка кількості виборців, які можуть одночасно обслуговуватись сервером виборчої дільниці;

$|K|$ – мінімально можлива кількість бітових ключів шифру, що є необхідною для користування ендоморфним шифром Вернама;

k_L – кількість можливих фальшивих ключів при розшифруванні тексту довжиною L;

L_0 – мінімальна довжина шифрованого тексту для однозначного встановлення істинного ключа шифру;

N – загальний обсяг сукупності, з якого береться вибірка;

n – розмір вибірки;

P_0 – імовірність обслуговування запиту виборця без очікування в черзі;

p – імовірність події, яку ми досліджуємо у відсотках;

τ – тривалість сеансу обслуговування виборця, с.

v – кількість інтервалів часу на обслуговування запитів одного виборця;

w – кількість пауз між періодами обслуговування запитів від одного виборця;

λ – середнє значення інтенсивності потоку запитів;

X – примітивний елемент поля Галуа $GF(2^{503})$;

z – коефіцієнт, що залежить від відсотку довіри;

ВСТУП

Актуальність теми. Одним з важливих напрямків розвитку сучасного суспільства є впровадження та вдосконалення засобів електронної демократії. Поняття електронної демократії охоплює процеси об’єктивного інформування суспільства, прийняття спільних рішень на основі свободи волевиявлення щодо тих чи інших аспектів суспільного життя та спільне контролювання діяльності органів державної влади з використанням мережних комп'ютерних технологій. Такі процеси можуть мати місце на різних рівнях організації суспільства від окремих підприємств або установ до загальнонаціональних і міжнародних. Важлива роль в системах електронної демократії належить процесу прийняття рішень шляхом дистанційного волевявлення з використанням інформаційно-телекомунікаційних мереж, зокрема Інтернет. Дистанційне волевиявлення (ДВ) – це зручний для учасників виборчих або конкурсних процедур спосіб відправлення своєї думки з будь-якого місця, де є доступ до мережі Інтернет, коли в якості пристрою для цього може бути використаний мобільний телефон, планшет, комп’ютер або, навіть, телевізор з функцією *SmartTV*.

Засоби захисту інформації, що функціонують у складі сучасних систем технічної підтримки процесів дистанційного волевиявлення (надалі - СДВ), не надають упевненості у тому, що специфічні загрози для інформації, виникнення котрих підриває довіру громадян до збереження таємниці та об’єктивності результатів волевиявлення, нейтралізовані.

У випадку ДВ виборець знаходиться поза меж контрольованої зони домену безпеки (тобто, поза меж виборчої дільниці), що суттєво збільшує можливості зловмисників щодо порушень свободи волевиявлення, зокрема змушування виборців здійснювати акти волевиявлення, що не відповідають їхній волі. Окрім того, обмін конфіденційними даними між виборцями та сервером дистанційного волевиявлення здійснюється, головним чином, через відкриті канали передачі інформації, що також збільшує можливості зловмисників щодо порушень конфіденційності, цілісності та доступності конфіденційних даних виборців. Існуючі СДВ, судячи з публікацій [1-3], не

мають ефективних механізмів протидії зазначеним вище загрозам і, от же, не вирішують проблему недовіри виборців, що ставить під сумнів доцільність їхнього застосування на практиці. От же, наукові дослідження, що спрямовані на вирішення цієї проблеми, є актуальними для суспільства.

Основний недолік сучасних СДВ полягає у відсутності дієвих механізмів всебічного повноцінного контролю над засобами самої системи з боку суспільства у цілому. Через це не може бути забезпечена повна гарантованість збереження таємниці голосів та захист виборців від адміністративного тиску. Виборець не є упевненим, що результати голосування не будуть спотворені, а персональні дані не розголошені.

Виконані у даній роботі дослідження СДВ (див. розділ 1) показують, що корені обґрунтованої недовіри громадян до «чесності» роботи СДВ слід шукати у недосконалості методів та засобів технічного захисту інформації (ТЗІ), що знайшли застосування в існуючих СДВ. У даній роботі зроблена спроба вирішити проблему недовіри методами, що лежать у сфері ТЗІ.

Дистанційне волевиявлення з застосуванням Інтернету надає суттєві переваги щодо зручності, мобільності та економії часу, але суттєвим стримуючим фактором на шляху його впровадження є недовіра спільноти через невпевненість у тому, що результати волевиявлення не будуть замінені, а таємницю голосів буде збережено. Більше ніж десять років в багатьох країнах світу в тій чи іншій мірі дистанційні засоби волевиявлення впроваджуються в системах голосування, але у жодному з цих випадків не надається доказів повної гарантії щодо неможливості підробки результатів і абсолютного збереження таємниці голосів. В Україні значна частина громадян не упевнена, що організатори СДВ в змозі запобігти проявам різного роду зловживань, а також моральному або іншому тиску по відношенню до учасників процесу волевиявлення. Тому наукові дослідження щодо можливості створення захищеної системи технічної підтримки процесів ДВ, що спрямовані на усунення перелічених недоліків СДВ, є актуальними.

Мета роботи

Мета даної роботи полягає в тому, щоб забезпечити беззаперечні гарантії неможливості виникнення порушень цілісності результатів волевиявлення та конфіденційності персональних даних голосуючих за умов повної недовіри до усіх без винятку учасників процесу ДВ, що усуває будь-які підстави для недовіри з боку голосуючих щодо можливості реалізації вказаних порушень.

Задачі дослідження

1. Здійснити аналіз характеристик існуючих СДВ з метою виявлення загроз, які підривають довіру громадян до результатів волевиявлення та збереження таємниці голосів, і визначити профіль захищеності інформації, що гарантує відсутність підстав для недовіри щодо точності результатів волевиявлення та/або збереження таємниці голосів.

2. Побудувати модель СДВ, що реалізує визначений профіль захищеності інформації під час обробки на сервері за рахунок використання методу спостереження в режимі реального часу за станом сервера з боку необмеженого кола будь-яких користувачів Інтернету в умовах недовіри до всіх без винятку осіб, що приймають участь у розробці, створенні та обслуговуванні СДВ.

3. Розробити метод досконало захищеного обміну даними через Інтернет між клієнтами та сервером СДВ, що забезпечує формально обґрунтовану неможливість порушень конфіденційності та цілісності даних в каналі зв’язку, з використанням необхідного для цього методу отримання випадкових бітових послідовностей в умовах типового клієнтського обладнання доступу до Інтернету без додаткових програмних або апаратних засобів.

4. Створити та протестувати програмне забезпечення, що реалізує розроблену модель та запропоновані методи захисту і спостереження. Оцінити показники якості функціонування цієї моделі.

Об’єктом дослідження є процеси захисту та технічної підтримки процедур ДВ з використанням мережі Інтернет.

Предметом дослідження є моделі, методи та засоби захищеної системи технічної підтримки процесів ДВ.

Методи дослідження

Виявлення «слабких місць» у захисті існуючих СДВ здійснено з використанням методів побудови комплексних систем захисту, що знайшли своє відображення у чинних нормативних документах з ТЗІ. Розробка методів, що гарантують контрольованість середовища функціонування СДВ, виконана на основі результатів теорії побудови обчислювальних середовищ. Розробка методів забезпечення гарантованої конфіденційності та цілісності даних, що передаються каналами зв'язку, заснована на теорії криптографічних систем, у т.ч. теорії секретного зв'язку К. Шеннона. Синтез джерела дійсно (а не псевдо) випадкових послідовностей здійснено на основі результатів математичного моделювання пакетного трафіка. Статистичні параметри побудованої СДВ оцінювались з використанням результатів теорії телетрафіка.

Наукова новизна одержаних результатів

1. Вперше побудовано **модель СДВ**, у якій за допомогою введення необмеженої кількості користувачів мережі з правами доступу виключно на ознайомлення з усіма файлами і процесами на сервері, але без прав на будь-яку модифікацію, можливе виявлення всіх порушень політики безпеки щодо цілісності результатів волевиявлення та конфіденційності голосів в умовах недовіри до всіх без винятку осіб, що беруть участь у розробці, створенні та обслуговуванні СДВ.

2. Дістав подальший розвиток **метод дистанційного спостереження** за роботою СДВ, використання якого за рахунок виконання визначеної послідовності контрольних дій та за умов повністю відкритого програмного забезпечення, унеможливлює виникнення непомічених порушень прийнятої політики безпеки, що усуває підстави для недовіри до СДВ.

3. Удосконалено **метод захищеного обміну даними через Інтернет**, який завдяки використанню випадкових бітових послідовностей та сумісного застосування шифру Вернама і алгоритму Диффі-Геллмана із параметрами, що забезпечують стійкий захист даних, і завдяки збереженню відстані єдиності згідно з К. Шенноном, формально обґрунтовують неможливість порушення

конфіденційності даних у каналі зв’язку.

4. Удосконалено **метод отримання випадкових бітових послідовностей,** який завдяки комбінованому використанню природної нестабільності кварцових резонаторів, що входять до складу типового клієнтського обладнання, та непередбачуваності моментів появи та тривалості переривань, що виникають під час обробки випадкових мережевих запитів, забезпечує можливість коректної реалізації шифру Вернама в сукупності з алгоритмом Диффі-Геллмана. Метод дозволяє на типовому клієнтському обладнанні без додаткових програмних або апаратних засобів реалізувати запропонований удосконалений метод захищеного обміну даними через Інтернет.

Практичне значення одержаних результатів

1. Використання побудованої моделі СДВ в сукупності з розробленими методами надає можливість гарантувати відсутність порушень таємниці голосів та істинність результатів волевиявлення під час проведення виборів, конкурсів та опитувань в умовах повної недовіри до всіх без винятку учасників процесу волевиявлення.

2. Використання запропонованого методу дистанційного спостереження за роботою сервера СДВ з боку необмеженої кількості користувачів мережі Інтернет за рахунок усунення підстав для недовіри щодо істинності результатів волевиявлення та збереження таємниці голосів стимулює до участі в масових опитуваннях, виборах та референдумах.

3. Побудована модель СДВ за рахунок можливості приховування інформації про результати особистого волевиявлення від зловмисників усуває доцільність незаконного впливу на виборців методами підкупу, залякування або силового тиску.

4. Результати роботи впроваджено у НАУ, КНУБА, Національному університеті ім. Т.Г. Шевченка та в комп’ютерній мережі Державного науково-дослідного інституту автоматизованих систем в будівництві (Акт впровадження

від 19.04.2018 р.), де встановлено відповідне програмне забезпечення для визначення суспільних думок, проведення референдумів, здійснення конкурсних та виборчих процедур.

Провідна наукова ідея

В роботі [4] відомий криптолог Брюс Шнайер запропонував в системах електронних виборів використовувати відкрите ПЗ, що сприяє підвищенню довіри виборців до електронного голосування. Але програмне забезпечення може бути в якійсь неконтрольований момент замінено на інше і не завжди таку заміну можливо виявити. Для унеможливлення зловживань подібного типу треба забезпечити постійний контроль за функціонуванням системи. Незважаючи на те, що ми не довіряємо кожному окремому учаснику виборчого процесу, ми маємо довіряти спільноті виборців у цілому. Тому технологія функціонування системи повинна забезпечити повноцінний дистанційний контроль з боку будь-якої особи, у т.ч. громадських контролерів, за усіма без винятку програмно-технічними засобами системи від моменту запуску до завершення виборчої кампанії. При цьому кожен спостерігач повинен мати можливість контролювати усі файли і процеси на тому сервері, де відбувається підрахунок голосів. Кількість спостерігачів не повинна обмежуватись.

Провідною науковою ідеєю, що набула розвитку у даній роботі, є забезпечення можливості повного дистанційного контролю за процесом волевиявлення будь-якими особами, незалежно від ролі, яку вони відіграють у цьому процесі. Необхідно створити просту і доступну методику дистанційного контролю середовища функціонування системи волевиявлення з будь-яких термінальних вузлів Інтернет, за допомогою якої отримувані результати перевірок у реальному часі мають виявляти будь-які фальсифікації виборчого процесу. У разі виявлення будь-якого відхилення (це може бути наявність розбіжності між файлом на сервері та еталоном або введення нештатної команди адміністратором сервера) спостерігач повинен мати можливість відправити повідомлення до відповідного контролюючого органу про факт порушення.

Слід зауважити, що відкритість (контрольованість) системи аж ніяк не повинна порушувати таємницю голосів виборців. Захищеність інформації у СДВ від порушень її конфіденційності та цілісності, у т.ч. і при обміні даними через відкрите середовище Інтернет, має не тільки зберігатися, а навіть підсилюватися до рівня надання стовідсоткових гарантій щодо неможливості дізнаватись, хто як голосував.

Якщо спиратися на вищезазначені твердження, то з'являється можливість подолати обґрунтовану недовіру виборців до СДВ шляхом забезпечення стовідсоткової гарантії: 1) повної контрольованості середовища функціонування СДВ з боку будь-яких осіб, незалежно від їхньої кількості, соціального статусу та місця перебування; 2) абсолютної захищеності інформації у СДВ від порушень її конфіденційності та цілісності, зокрема шляхом коректного використання досконалих шифрів.

Суттєва увага у даній роботі приділена забезпеченню захисту від спотворень результатів волевиявлення. Таке забезпечення має здійснюватися з урахуванням того, що неможливо гарантувати чесне ставлення до виконання функцій управління системою з боку обслуговуючого персоналу. Цілком можливо, що цей персонал у тій чи іншій мірі буде зацікавлений в отриманні результату підрахунку голосів на чиюсь користь. Фактично серед учасників виборчого процесу може не існувати осіб, які були б зацікавлені в отриманні об'єктивних результатів волевиявлення. Тому в умовах повної недовіри до всіх без винятку учасників виборчого процесу необхідно створювати такі засоби та умови, які б унеможливлювали будь-яке втручання у виборчий процес. При цьому також необхідно враховувати можливість впливу за допомогою підкупу, погроз або залякування з боку зацікавлених осіб на персонал, що обслуговує систему, або на самих виборців. Усім переліченим загрозам, які здатні спотворювати результати волевиявлення, необхідно протиставити досконало стійкі засоби захисту та надати стовідсоткові гарантії коректного їхнього функціонування в реальних умовах використання.

Зв'язок роботи з науковими та учбовими програмами, планами, темами

Висвітлені в дисертації наукові результати отримано, здебільшого, в рамках науково-дослідної роботи, яка була виконана Київським національним університетом будівництва і архітектури (КНУБА) на замовлення Державного НДІ автоматизованих систем у будівництві Мінрегіон України (НДІАСБ), що здійснює свою діяльність у сфері створення комп'ютерних систем для потреб будівельної галузі, та „Укртелеком" (Договір про НДР №1036-Х15/80С321-2731). Автор дисертації був виконавцем цих робіт. Отримані результати використовуються у навчальному процесі НАУ при викладанні навчальної дисципліни «Стратегії обслуговування телекомунікаційних мереж».

Особистий внесок здобувача

Основні положення і результати дисертаційної роботи отримані автором самостійно, обмежуються обсягом тих результатів наукової діяльності, які відображені у цій роботі. Із робіт, що опубліковані у співавторстві, у дисертаційній роботі використовуються результати, що отримані особисто здобувачем.

Апробація результатів дисертації

Результати досліджень дисертаційної роботи доповідались, обговорювались і отримали позитивну оцінку на наступних конференціях:

1. IX Міжнародна науково-технічна конференція «Новітні комп'ютерні технології» NOCOTE'2011 (м. Севастополь, 2011).

2. X Міжнародна науково-технічна конференція «Новітні комп'ютерні технології» NOCOTE'2012 (м. Севастополь, 2012).

3. Наукова конференція молодих вчених, аспірантів і студентів КНУБА (м.Київ 2011)

4. 72-га науково-практична конференція КНУБА (м. Київ, 2011).

5. 73-тя науково-практична конференція КНУБА (м. Київ, 2012).

6. 75-та науково-практична конференція КНУБА (м. Київ, 2014).

7. XI Міжнародна науково-технічна конференція «Новітні комп'ютерні

технології» NOCOTE’2013 (м. Севастополь, 2013).

8. Міжнародної наукової конференції ICS-2015 «Інформація, комунікація, суспільство 2015» (м.Львів, 2015)

9. 71-ша підсумкова наукова конференція професорсько-викладацького складу ДВНЗ «Ужгородський національний університет» (м. Ужгород, 2017).

Достовірність результатів. Для перевірки достовірності одержаних результатів проведено повне натурне моделювання і випробування усіх програмно-технічних засобів в умовах реальних опитувань серед студентів провідних вищих навчальних закладів України.

Публікації. За результатами виконаних досліджень опубліковано 10 наукових робіт, з яких 6 статтей у фахових науково-технічних спеціалізованих виданнях та 4 тези доповідей на науково-технічних конференціях.

Структура та обсяг роботи. Дисертаційна робота складається зі вступу, чотирьох розділів, висновків по кожному розділу та загальних висновків по роботі в цілому, списку використаних літературних джерел (84 найменування), 2 додатки. Повний обсяг дисертації - 226 сторінок (без анотації), у тому числі 154 сторінок основного тексту, 28 рисунків, 10 таблиць.

РОЗДІЛ 1
АНАЛІЗ СИСТЕМ ТЕХНІЧНОЇ ПІДТРИМКИ ПРОЦЕСІВ ДИСТАНЦІЙНОГО ВОЛЕВИЯВЛЕННЯ З ПОЗИЦІЙ ЗАХИСТУ ІНФОРМАЦІЇ

1.1. Основні характеристики систем технічної підтримки процесів дистанційного волевиявлення з позицій захисту інформації

Об’єктом даного дослідження є процеси дистанційного волевиявлення з використанням інформаційно-телекомунікаційних мереж, зокрема мережі Інтернет, під час проведення, перш за все, загальнонаціональних виборів, але також і різноманітних референдумів, опитувань, конкурсів, експертних оцінювань тощо. Під волевиявленням розуміється акт групового прийняття рішення щодо вибору κ варіантів із n можливих, де $k \leq n$, в умовах, коли необхідно зберегти таємницю індивідуального вибору кожного із членів групи, а процес волевиявлення може породжувати проблемну ситуацію – недовіру до результатів волевиявлення. Ці процеси у даній роботі розглядаються як процеси визначення результатів колективного волевиявлення шляхом дистанційного голосування через Інтернет. З технічної точки зору вони підтримуються програмно-апаратними засобами систем дистанційного волевиявлення, зокрема систем дистанційного загальнонаціонального голосування (СДГ), що знайшли застосування для вирішення політичних проблем суспільства.

Дистанційне голосування – зручний для учасників виборчих (конкурсних) процедур спосіб волевиявлення з будь-якого місця розташування, де є доступ до мережі Інтернет, коли в якості термінального пристрою для здійснення волевиявлення може бути використаний мобільний телефон, планшет, комп’ютер або, навіть, телевізор з функцією *SmartTV*. Саме завдяки зручному користувацькому інтерфейсу СДВ (у т.ч., СДГ) набувають усе більш широкого використання.

1.1.1. Фундаментальні принципи функціонування систем дистанційного голосування у демократичних країнах

Легальність функціонування СДГ демократичних країн базується на принципах, що витікають із Міжнародного пакту про політичні права [5], що

конкретизує загальні права людини, які задекларовані ООН у 1948 р. Відповідно до пакту кожна людина має право брати участь в управлінні своєю країною через вільно обраних представників, голосувати на справжніх періодичних виборах, які проводяться на основі загального і рівного виборчого права при таємному голосуванні [5]. Таємне голосування гарантує також стаття 71 Конституції України, а також відповідні Закони України «Про вибори Президента України», «Про вибори народних депутатів України» та ін. Порушення таємниці голосування службовою особою тягне кримінальну відповідальність (стаття 159 Кримінального Кодексу України) [6-8].

В умовах вільного вибору виключається можливість будь-якого примусу щодо здійснення акту голосування. Процедура голосування має унеможливлювати виявлення того, як саме голосував той чи інший виборець, або чи мав він взагалі намір голосувати. При цьому жоден виборець не може бути примушений, навіть в суді, визнати, як він голосував.

Значну роль при проведенні виборів відіграє принцип гласності. Це означає, що необхідно інформувати виборців про роботу СДВ. Виборці повинні розуміти особливості системи, що використовується, і ставитися до неї з довірою. Внаслідок територіального розмежування Інтернет-терміналів, з яких голосують виборці, істотно погіршується будь-який контроль за процесом голосування з боку громадськості, преси, спостерігачів, тому інформуванню про хід виборів приділяється особлива увага.

1.1.2. Основні характеристики існуючих систем дистанційного голосування у демократичних країнах

Розглянемо характеристики загальнонаціональних СДГ, що використовують у виборчих процесах країн із «зрілою» демократією.

Провісником вторгнення Інтернет-технологій у сферу громадсько-політичних відносин став бурхливий розвиток мережних послуг, завдяки якому стало можливим розраховуватися кредитними картками, робити замовлення в Інтернет-магазинах, користуватися сервісами Інтернет-банкінгу. Позитивний досвід користування мережними послугами, перевірений на десятках і сотнях

мільйонів транзакцій, надав упевненості у доцільності застосування Інтернету і для підтримки інших юридично обумовлених функцій. На сьогодні Інтернет став основною платформою для електронної підтримки функцій органів державної влади. Останні дослідження [9] показують, що соціальні ЗМІ і урядові ресурсні сайти та портали є популярними у мережі, хоча існує значна неоднаковість щодо їхньої присутності та впливовості в залежності від політики органів та рівня влади. Наприклад, у Великобританії та Австралії налічується більш ніж 100 000 сайтів, що забезпечують таке сучасне явище, яке позначається терміном Інтернет-демократія. Електронний уряд визначається (Європейською комісією) як використання інформаційних та комунікаційних технологій у державному управлінні у поєднанні з організаційними змінами, щоб поліпшити державні послуги, демократичні процеси та зміцнити підтримку державної політики [10]. Інтернет призвів до еволюційних, а не революційних змін в політиці, він сьогодні продовжує змінювати демократію в позитивну сторону і заохочувати всіх громадян бути повноправними учасниками у формуванні державної політики [11].

Існує світовий досвід впровадження СДГ через Інтернет для забезпечення загальнодержавних виборів. Естонія стала першою країною у світі, яка стала на цей шлях. На сьогоднішній день близько третини її виборців голосують через Інтернет [12]. Інтернет-голосування стає усе більш популярним серед виборців, Наприклад, за даними звіту про голосування на парламентських виборах у Норвегії у 2013 році [13] до 37% бюлетенів було прийнято через Інтернет (проти 26% у 2011 році). Інтернет-виборці зазначали, що їм було легко і зручно при голосуванні через Інтернет, і це основна причина їхнього вибору цього способу голосування. Наразі у Америці йде підготовка до Інтернет-голосування на виборах президента у 2016 році [14].

Слід вказати на дві додаткові причини, що сприяють поширенню СДГ. По-перше, це зростаючий глобальний рух людей. Зокрема, наявність значної кількості виборців за кордоном, наприклад, військових. У зв’язку з цим Конгресом США були прийняті зміни до законодавства щодо заочного

голосування, які надали поштовху для розвитку технологій дистанційного голосування [15]. На сьогодні 66% громадян, які проживають за кордоном, а зареєстровані у муніципалітетах, голосують за допомогою мережі Інтернет. По-друге, це значне поширення мобільних пристроїв та бездротових технологій доступу до Інтернет, що зумовило можливість їх застосування у процесі дистанційного голосування. На сьогодні розроблено специфікації та вимоги до мобільних підсистем, зокрема на платформі Android [16], що надало можливість доступу до СДГ широкому колу користувачів.

Досвід застосування існуючих СДГ показав, що усі вони, у цілому, задовольняють виборців за характеристиками швидкості доступу. Зокрема, порівняльне тестування найбільш поширених на сьогодні систем таємного голосування (Baseline Data for Helios, Pret a Voter та Scantegrity II) показали [17], що хоч швидкості роботи систем, що порівнювались, відрізнялись одна від одної майже втричі, тим не менш, вони задовольняли користувачів.

Нами було проаналізовано опубліковані офіційні звіти про результати застосування виборчих Інтернет-технологій, зокрема у Норвегії [13], на виборах у США серед військових за кордоном [18] та на парламентських Інтернет-виборах в Естонії [19]. Досліджена також користувацька "зручність" інтерфейсів різних систем Інтернет-голосування [20] та окремі вимоги до них [21-22]. Загальний висновок: СДГ країн із сталими демократичними традиціями функціонують у повній відповідності із вище зазначеними принципами, а «чесність» проведення виборів, істинність отриманих результатів волевиявлення, за винятком окремих локальних форс-мажорних випадків, не викликають нарікань з боку суспільства.

1.1.3. Основні характеристики існуючих систем дистанційного волевиявлення в Україні

В Україні в контексті євроінтеграції та впровадження міжнародних стандартів розроблено законопроекти у рамках реалізації проекту «Цифрова Україна», які є першим кроком на шляху побудови юридичного фундаменту для впровадження сучасних інформаційних технологій у державному секторі та

надання цифрових сервісів громадянам та бізнесу [23]. Вони регламентують та врегульовують прийом електронних звернень від громадян у цифровій формі без жодних підписів і технічних обмежень, а також забезпечують прозорість роботи державного апарату, оприлюднення структурованих даних органів влади, інформації про їх діяльність, формування державного бюджету тощо.

В Україні також створюються системи Інтернет-голосування [24]: Facebook-опитування, в якому проголосували десятки тисяч користувачів; онлайн-голосування проекту "Просвіта" [25]; інтерактивна карта Міністерства внутрішніх справ "Вибори" [26], за допомогою якої можна отримати інформацію про виборчі правопорушення та побачити реакцію міліції на них; проект «Опір» [27] для моніторингу прозорості виборів, що підтримує мобільні термінали Android та iOS для повідомлення про порушення під час голосування; інструменти Інтернет-демократії "Опора" [28], що містять інтерактивну карту з даними про хід голосування, спеціальний навчально-практичний посібник для журналістів і спостерігачів на виборах, електронні бланки скарг і форми повідомлень про порушення; АС "eCast" [29], яка інтегрує центр обробки SMS з Web-доступом до результатів, що дозволяє оперативно збирати та зводити дані від спостерігачів з усіх ДВК і, завдяки автоматизації цього процесу, оновлювати дані в режимі онлайн; сайт електронних петицій на сторінці Офіційного Інтернет-представництва Президента України [30].

За підтримки Фонду сприяння демократії Посольства США в Україні був реалізований проект, за яким для популяризації системи проводилось учбово-тренувальне онлайн-голосування "День виборів Президента" [31]. На жаль, розробники цієї системи Інтернет-голосування не дуже професійно підійшли до захисту від зловмисників, тому Інтернет-вибори були зірвані.

Для підвищення довіри до результатів виборів в НАНУ [32] розробляються технічні рішення щодо можливості перевірки виборцем врахування його волевиявлення шляхом порівняння цифрового коду, в якому зашифровані номери кандидатів, за які він проголосував з кодами, наведеними

на сайті виборчої комісії. Цифровий код, що сформований системою, є секретним і відомий тільки виборцю.

З’являються державні Інтернет-ресурси, що є необхідними для підприємницької діяльності та приватного сектору. Наприклад реєстри документів дозвільного та декларативного характеру у будівництві [33]. Розроблюються Інтернет-системи аудиту та моніторингу [34], які охоплюють термінали по всій країні.

Проте СДГ поки що не знайшли широкого застосування в Україні. Причини такого стану речей розглянуто далі.

1.1.4. Аналіз стану захищеності інформації в існуючих СДГ

Перші системи електронної реєстрації учасників виборів, підрахунку голосів та формування результатів виборів почали створюватись у 80-х роках без приділення належної уваги до вирішення завдань інформаційної безпеки. Тому їх застосування на практиці виявили численні прояви нібито випадкових помилок поряд із звинуваченнями у підтасовуванні результатів. (Однак, межа між випадковістю і злим умислом є дуже розпливчастою.) Аналіз цих систем з позицій ТЗІ виявив серйозні недоліки у їхній побудові, зокрема існування можливостей для зловживання комп'ютерними системами - особливо з боку обслуговуючого персоналу [35]. Тому у 1993 році П.Нейманом вперше були сформульовані критерії інформаційної безпеки для систем електронного голосування. У подальшому ці критерії були удосконалені та узагальнені [36]. Наразі розроблено ряд систем, які відповідають вказаним критеріям, наприклад, система Sensus [37] або електронна система таємного голосування Civitas [38], для якої на основі криптографічного аналізу була доведена стійкість захисту.

Проблеми інформаційної безпеки, що стоять перед будь-яким розподіленим застосуванням, збільшуються, коли мова йде про вирішення завдань, що мають важливе значення для демократичного суспільства. Загальнонаціональний виборчий процес забезпечує вирішення саме таких важливих завдань. Засоби ТЗІ у СДГ повинні унеможливлювати порушення процесу голосування або ставити під сумнів легітимність результатів.

Судячи з підсумків Шостої міжнародної конференції з електронного голосування EVOTE2014 [39], яка проходила у місті Брегенц (Австрія) з 28 до 31 жовтня 2014 року, вирішальним фактором щодо успіхів електронного голосування є впевненість виборців у незаангажованості виборчого процесу і їхня довіра до коректності реалізації нових інформаційно-телекомунікаційних технологій. Така довіра та упевненість у більшості виборців демократичних країн, у цілому, існує. Тому СДГ у цих країнах широко застосовуються. Проте з урахуванням високої міри важливості результатів виборів для суспільства є і супротивники впровадження технологій дистанційного доступу у процес голосування. Аналізу можливих негативних наслідків від дистанційного голосування присвячена робота італійця Е. Ломбардії [40]. В якості головного аргументу висувається припущення про неможливість створення системи дистанційного голосування, яку можна було б проконтролювати на 100%, внаслідок чого у критичних ситуаціях виникає недовіра виборців до подібних систем. Крім того, висловлюється сумнів з приводу забезпечення таємниці голосу через складність для розуміння виборцями криптографічних методів захисту. Інша аргументація пов'язана з можливими фальсифікаціями шляхом втручання у роботу сервера обліку голосів.

1.2. Недоліки систем технічної підтримки процесів дистанційного волевиявлення з позицій захисту інформації

СДВ у демократичних країнах будуються, як правило, у відповідності із архітектурою «клієнт/сервер», для інформаційних ресурсів котрих притаманні загрози несанкціонованого доступу (НСД) до серверних даних, а також порушення конфіденційності та цілісності даних під час їхнього передавання через незахищене середовище. У рамках інформаційно-телекомунікаційних систем, що побудовані згідно цієї архітектури, функціонують достатньо досконалі системи захисту інформації, які, у цілому, задовольняють потреби

власників та користувачів цих систем. Це твердження є справедливим і для СДВ демократичних країн.

Проте в країнах, що не мають стійких демократичних традицій, процедура голосування через Інтернет повинна враховувати специфічні особливості менталітету виборців і конкретні умови здійснення виборчого процесу. Зокрема, враховувати тотальну недовіру основної маси виборців до будь-яких суб'єктів, які яким-небудь чином пов'язані з організацією виборів, або можливість підкупу, адміністративного або навіть силового тиску на виборців. Важливою особливістю систем загального голосування є наявність труднощів у формуванні груп організаторів і адміністраторів (менеджерів) СДГ, які були б об'єктивно зацікавлені у достовірності результату підрахунку голосів, оскільки при загальному голосуванні у кожного представника цієї групи може існувати зацікавленість в тому, щоб результат голосування співпадав з його власним бажанням. Цим системи загального голосування відрізняються від більшості інших систем, власники яких мають можливість звернутися до послуг об'єктивно незацікавлених менеджерів. У захисті системи загального голосування від фальсифікацій може бути зацікавлено тільки суспільство виборців у цілому, але до кожного окремого громадянина або будь-яких груп громадян є підстави відноситися з недовірою. Не можна також довіряти найманому обслуговуючому персоналу, оскільки на нього можуть робити вплив зацікавлені групи виборців. Довірою в частині захисту системи від фальсифікацій може володіти тільки суспільство виборців в цілому. Звідсіля витікає думка, що єдиним методом, який потенційно може гарантувати відсутність спотворень у результатах волевиявлення, є масовий вичерпний контроль самими виборцями усіх процесів діючої системи у реальному часі без яких-небудь обмежень на кількість контролерів. Проте завдання реалізації такого контролю ускладнюється тим, що при цьому необхідно зберегти в таємниці персональні дані і голоси виборців.

От же, з точки зору ТЗІ модель загроз для інформації у СДВ має враховувати специфічні загрози, що безпосередньо спричиняють недовіру

виборців. Звідсіля випливає завдання: доповнити відому модель загроз для інформації у структурах типу "клієнт/сервер" підмножиною загроз, які відбивають специфіку голосування через Інтернет в країнах з недорозвиненими демократичними традиціями. Наш аналіз [41] вказує на наступну підмножину загроз та потенційно можливі методи їхньої нейтралізації.

1. **Загроза реєстрації фіктивних виборців**

На підготовчій стадії виборчого процесу має місце загроза, яку, якщо не прийняти відповідних заходів, легко реалізувати при дистанційному голосуванні. Цією загрозою є реєстрація фіктивних виборців. Вірогідних зловмисників слід шукати, передусім, серед працівників, які готують списки виборців для дистанційного голосування і заносять їхні персональні дані у базу даних. Процедура складання списків виборців у нашому випадку вимагає точної реєстрації дій персоналу для можливого подальшого притягнення до відповідальності за помилки або зловживання. Проте виявити фальсифікацію на етапі складання списків виборців дуже складно, оскільки відрізнити дійсний запис від фіктивного без детального аналізу таких списків практично неможливо. При цьому потрібно мати упевненість, що аналіз виконується неупередженими особами.

Можливий метод протидії. Виявити фіктивних виборців можна тільки за допомогою детального аналізу списку виборців. Такий аналіз доцільно здійснювати силами самих виборців – громадських контролерів за допомогою структурованих списків з підсумками. Наприклад, для виборів у рамках підприємства досить опублікувати кількість виборців по кожному структурному підрозділу. При цьому комп'ютерний файл з базою даних має бути відкритим для перегляду. Конфіденційні дані у цьому файлі (наприклад, паролі виборців) мають бути надійно зашифровані. А відкритий параметр, що має аналізуватися - загальна кількість виборців. За наявності фальсифікацій ця кількість перевищить суму за списками підрозділів. От же, громадські контролери легко зможуть виявити приписки у підрозділах і фіктивні підрозділи. Для загально національних виборів за списком достатньо буде

підсумувати кількість виборців у розрізі квартир, будинків, вулиць і виборчій дільниці у цілому. При цьому виборцям не буде складно виявити «зайвих» мешканців у своїй квартирі або «зайвий» будинок на своїй вулиці. Зрозуміло, що слід провести попередню роз'яснювальну роботу серед виборців, щоб вони не були байдужими і здійснювали відповідний контроль списків виборців.

2. Загроза некоректної роботи закритого серверного програмного забезпечення

Найбільш небезпечним джерелом загроз являється сервер, який отримує і підраховує голоси виборців. Розробники програмного забезпечення (ПЗ) у минулому дотримувалися принципу закритості текстів програм, мотивуючи це необхідністю надійного захисту системи від зловмисників.

Метод протидії. Брюс Шнаєр у роботі [4] вказав на уразливість цього принципу. Нині серверне ПЗ для СДВ у демократичних країнах відкрито публікують і обговорюють.

3. Загроза використання фальшивого серверного програмного забезпечення

Існують сумніви, що опубліковане ПЗ буде замінено на підробку. Щоб переконатися у непідробності ПЗ, слід надати усім виборцям право контролювати усі файли і процеси на сервері у реальному часі, що і запропоновано у роботі [41]. Такий підхід значно ускладнює завдання для зловмисників, але, все ж, не виключає можливостей для втручання у процес підрахунку голосів або підміни результатів голосування. Умови для реалізації цієї загрози:

- наявність «слабкого місця» у серверній ОС з точки зору контрольованості дій адміністратора;

- недосконаліть технології контролю об'єктів та процесів на сервері;

- наявність «слабкого місця» у прикладному ПЗ.

Можливий метод протидії. Необхідно забезпечити виконання основної вимоги до ОС - контрольованість усіх без виключення файлів на сервері, а також дій персоналу з установки і запуску програм. Для спрощення контролю

слід видалити файли, у яких немає необхідності. Сучасні ОС з кожним роком стають складнішими із-за всілякого сервісу, що розширюється. У нашому випадку ускладнення ОС є небажаним. (Це добре узгоджується з думкою Брюса Шнайера [4] з приводу розробки систем електронного голосування, де він призвав не витрачати гроші на ускладнення систем, а, навпроти, стати на шлях їх спрощення.) У системі дистанційного голосування слід використати ОС, в якій реалізовані можливості її контролю з боку рядових виборців. Така ОС має забезпечити виконання наступного ряду функцій.

1) Створення користувача з правами громадського контролера, якому дозволено переглядати і копіювати усі файли сервера, а також виконувати безпечні команди для контролю дій адміністратора. Кількість таких користувачів має бути необмежена, а процедура контролю - досить простою і доступною.

2) Установка стандартних мовних засобів програмування для забезпечення функціонування прикладного ПЗ.

3) Створення користувача (адміністратора сервера) з обмеженими правами, якому дозволено копіювати файли прикладного ПЗ тільки у наперед визначену директорію і здійснювати тільки їхній запуск на виконання.

4) Видалення користувача - головного адміністратора (типу *root*) з повними (надлишковими у даному випадку) правами.

5) Установка і запуск прикладного ПЗ.

6) Забезпечення безперервної роботи прикладних програм в автоматичному режимі на протязі усієї виборчої кампанії аж до моменту фізичного відключення сервера. (Зрозуміло, що для цього необхідно забезпечити експлуатаційну надійність роботи сервера по критерію коефіцієнта готовності не гірше 0,9999.)

4. Загроза заміни прикладного ПЗ (перший варіант)

Найбільш небезпечна загроза для виборчої спільноти - підміна прикладного ПЗ. При цьому порушник може імітувати процес нормального голосування для виборців і, у той же час, як завгодно міняти результати

підрахунку голосів. Крім того, порушник зможе отримати можливість дізнатися, хто як голосував, тобто порушувати таємницю голосування. Реалізація цієї загрози пов'язана з порушенням прийнятої послідовності дій адміністратора сервера. Щоб реалізувати цю загрозу, треба у копійованому на сервер ПЗ підмінити основну програму, яку відразу після підміни запустити на виконання, а після цього скопіювати на її місце штатну програму.

Можливий метод протидії. Щоб полегшити можливість виявлення громадськими контролерами такого порушення слід передбачити обов’язкову паузу між моментом закінчення копіювання ПЗ і дозволеним моментом запуску програми на виконання. Тоді у громадських контролерів буде досить часу для верифікації програм, щоб переконатися у відсутності загрози. У таких умовах реалізація цієї загрози, у т.ч. знищення доказів нелегального втручання в роботу сервера працюючим під громадським контролем адміністратором, уявляється неможливою.

5. Загроза заміни прикладного ПЗ (другий варіант)

Інший варіант реалізації цієї загрози полягає у заміні сервера. Під час такої заміни усі сеанси зв'язку користувачів будуть перервані, що, безумовно, повинно насторожити контролерів, але їм треба буде довести факт порушення, а для цього слід знайти відмінність у файлах або інших характеристиках штатного і заміненого серверів.

Проаналізуємо технологію реалізації цієї загрози і можливість видалення усіх доказів при заміні штатного сервера на якій-небудь інший. Називатимемо ці сервери першим і другим, відповідно. У цьому випадку модель порушника можна представити таким чином. Другий сервер, що призначений для фальсифікації голосування, треба готувати синхронно з першим в частині перших трьох функцій ОС, перелічених вище.

З першого сервера на другий слід скопіювати файли з журналами відвідувань і інші файли, в яких відбита поточна робота сервера. Оскільки неможливо забезпечити точність синхронізації в межах однієї секунди, а будь-яка розбіжність часу створення якого-небудь файлу більша, ніж на секунду,

легко виявляється при контролі списку файлів (і, отже, може бути пред'явлена в якості доказу підміни сервера), то необхідно на штатному сервері відразу після його відключення виконати спеціальну програму зчитування даних про час створення усіх файлів і перенести ці дані другий сервер.

Далі слід видалити користувача з повними правами і запустити нештатну програму, яка повинна мати те ж ім'я, що і штатна. Після запуску нештатної програми слід замінити файл з нештатною програмою на штатний файл. Тепер можна підключити другий сервер до мережі замість першого. Таким чином, замість штатної програми користувачі спілкуватимуться з програмою зловмисника і майже усі сліди, що свідчать про загрозу, будуть знищені.

Метод протидії. Звернемо увагу на сліди, які неможливо знищити, і вкажемо ознаки, на які слід звернути увагу, для виявлення загрози. У різних ОС необхідні ознаки можуть відрізнятися, але для вирішення нашого завдання вибір ліг на OpenBSD як найкращу ОС за критеріями максимальної захищеності і простоти (обґрунтування – у розділі 2), то вкажемо таку ознаку для цієї системи.

Щоб виявити загрозу досить мати хоч би одну ознаку, яку неможливо завуалювати. Такою ознакою є результат виконання команди *ps aux*. Ця команда виводить коротку довідку про усі працюючі процеси. Серед процесів є такі, які запускаються на початку роботи системи і зберігаються до її відключення. Кожному процесу система привласнює ідентифікатор *PID*. Оскільки ідентифікатори привласнюються випадковим чином, то практично неймовірно, щоб вони співпали при різних запусках системи. Варіанти, що пов'язані з підробкою ОС, не розглядаються, оскільки при технології контролю, описаній нижче, така підробка не є можливою. Тому достатньо на початку роботи системи зафіксувати результат виконання команди *ps aux* і, якщо ідентифікатори процесів, які стартували при запуску системи, стануть іншими, то це і буде доказом наявності можливої загрози.

Критичний інтервал, під час якого порушник може підмінити сервер, не залежить від виду тієї або іншої ОС. Початком інтервалу є момент копіювання

файлів прикладних програм, а кінцем - момент запуску прикладної програми. Якщо в цьому інтервалі не спостерігалося розриву зв'язку із сервером одночасно в усіх контролерах, то це означає, що сервер не підмінявся.

6. Загроза підміни результатів голосування у процесі розсилки

Надійний захист таємниці голосів виборців і точність їх обліку на сервері можуть бути забезпечені шляхом обробки голосів у внутрішніх змінних виконуваної програми, оскільки до цих змінних ніякий зовнішній доступ не є можливим. Але після завершення підрахунку голосів, коли усі конфіденційні дані оброблені і знищені, а отримані результати голосування слід оприлюднити, необхідно вжити особливі заходи для унеможливлення підміни результатів у процесі розсилки, оскільки саме у цей момент зловмисник, що має доступ до сервера, використовуючи відому технологію проміжного сервера, може підмінити розсилку.

Можливий метод протидії. Для виключення цієї загрози можна надати можливість виборцям отримувати результати безпосередньо із сервера у захищеному вигляді, наприклад використовуючи той же метод захисту, що і при передаванні даних щодо волевиявлення. Тоді кожен виборець зможе звірити результати, що отримані у захищеному виді, з опублікованими у пресі і переконатися у відсутності розбіжностей.

7. Загроза примусу виборців віддавати свій голос усупереч їх власній думці

Існує група загроз, що пов'язані безпосередньо з процесом дистанційного голосування, оскільки виборці у цьому випадку у момент голосування знаходяться поза меж доменів безпеки. Наприклад, примус виборців віддавати свій голос усупереч їх власній думці, використовуючи підкуп, залякування або інші методи впливу. Такі незаконні дії можуть суттєво вплинути на результати голосування, особливо в кризовий період.

Метод протидії. Для боротьби з такими загрозами при дистанційному голосуванні можна використати метод заміни пароля, при якому тільки той, що здійснює акт голосування, буде обізнаний про справжній результат свого

голосування, а ті, хто безпосередньо візуально спостерігають за усіма його діями, не зможуть відрізнити факт дійсного голосування від імітації. Для протидії цим загрозам слід створювати такі умови, щоб виборець мав змогу виявити дійсно своє волевиявлення, не зважаючи на підкуп, залякування або інші засоби зовнішнього впливу.

8. Підробка результату підрахунку голосів

Остання група загроз, де зловживання є найбільш небезпечними, бо їх вплив може повністю зруйнувати картину волевиявлення, є підробка результату підрахунку голосів. Для усунення можливості загроз цього типу необхідно, з одного боку, надійно ізолювати програму підрахунку голосів від будь-якого несанкціонованого втручання в процес її роботи, а, з другого боку, треба забезпечити достовірний вивід результатів підрахунку. Крім того, у разі необхідності підсумовувати результати голосування з різних дільниць, треба унеможливити будь-яку підробку цього процесу шляхом надання йому повної прозорості і контрольованості для широкого кола виборців.

9. **Порушення цілісності, конфіденційності або доступності щодо процесів обміну даними між виборцями і сервером підрахунку голосів через загальнодоступні канали мережі Інтернет. Перехоплення автентифікаційних даних виборців (наприклад, паролів) для підміни голосуючої особи або блокування доступу виборців до сервера.**

Названі загрози не є специфічними саме для СДВ. Специфічними мають бути умови їхньої нейтралізації, а саме: для отримання довіри виборців гарантованість захисту даних при обміні через незахищене середовище Інтернет має бути абсолютною. Методів протидії цим загрозам існує досить багато і тут задача захисту полягає у виборі серед всього цього різноманіття найбільш придатних і ефективних для реалізації саме для умов функціонування СДВ.

Слід зауважити, що саме впровадження дистанційного голосування вже і є потужним засобом унеможливлення цілої низки методів фальсифікації, які можливо реалізувати тільки за умов використання паперових бюлетенів. До

таких методів слід віднести вкидання заповнених пачок бюлетенів, вибіркове псування бюлетенів членами виборчих комісій під час підрахунку, а на останок, друк бюлетенів у двох примірниках з метою підміни дійсних пачок бюлетенів на підроблені разом з підміною протоколу виборчої комісії про результати голосування.

1.3. Аналіз методів захисту інформації в операційному середовищі сервера дистанційного голосування

Структура сучасних розподілених комп'ютерних систем, що засновані на використанні ресурсів Інтернет, є досить складною та розгалуженою, що надає широкі можливості для зловмисників щодо пошуку шляхів реалізації планів порушення безпеки інформації. Міжнародний досвід експлуатації таких систем, у складі котрих зазвичай функціонують розвинені засоби захисту інформації, свідчить про те, що гарантувати абсолютний (тобто, сто відсотковий) захист від зловмисників у більшості випадків практично неможливо. Щодо систем електронного голосування Брюс Шнаєр [4] пропонує шлях підвищення захищеності інформації, що полягає у максимально можливому спрощенні таких систем, зокрема їхнього ПЗ. Але різноманіття нових функцій сучасних ОС, кількість яких з кожним роком невпинно збільшується, стимулює ускладнення ПЗ з метою надання додаткових можливостей та зручностей користувачам. З позицій ТЗІ таке ускладнення пов'язано з можливістю появи нових «слабких місць» у захисті, які можуть довгий час залишатись непоміченими і, отже, незахищеними. Пропозиції щодо спрощення комп'ютерних систем з метою забезпечення більш надійного захисту інформації поки що не знаходять підтримки серед розробників, скоріш за все, через необхідність відмови від певного ряду зручностей, до яких вже звикли як самі розробники, так і користувачі.

Проте, як свідчать результати підрозділу 1.3, задача створення досконало захищеної СДГ є досить чітко функціонально обмеженою, то є доцільним проаналізувати можливі шляхи її розв'язання.

По-перше, для розв'язання даної задачі слід обрати відкриту і надійно захищену ОС, у якій був би мінімум необхідних функцій. Така система може бути утворена шляхом блокування зайвих функцій у вже існуючій ОС.

По-друге, прикладне ПЗ повинно бути відкритим і достатньо простим, щоб нескладно було упевнитись в його безпечності та надійності функціонування.

По-третє, усі дії персоналу, що пов'язані з роботою прикладного ПЗ, повинні бути повністю контрольованими із точок віддаленого доступу, щоб можна було гарантувати неможливість часткової або повної заміни будь-якого штатного файлу з програмами або даними.

Сумлінність виконання функцій адміністративного контролю роботи СДГ не викликає довіри з боку суспільства. Об'єктивним може бути тільки всеохоплюючий та безперервний контроль з боку громади у цілому без обмежень на кількість контролюючих осіб. Тільки за умов повного виконання усіх перелічених вище вимог може йти мова про досконало надійний захист інформації в операційному середовищі СДГ.

Інформацію, що потребує захисту від порушень конфіденційності та цілісності, доцільно розділити на такі два види:

- паролі виборців (або інша інформація для автентифікації), які зберігаються у зашифрованому вигляді і завантажуються з файлу даних про виборців;
- інформація про волевиявлення виборців, яка надходить в процесі здійснення актів голосування у зашифрованому вигляді безпосередньо до програми підрахунку голосів через інтерфейс віддаленого зв'язку.

Метод шифрування паролів повинен бути достатньо надійним, щоб витік зашифрованих паролів не міг призвести до розкриття самого пароля за той проміжок часу, у який можна скористатись паролем для фальшування голосу

виборця. Оскільки ПЗ системи є відкритим, то програма шифрування паролів буде відома усім, включаючи зловмисника. Знаючи зашифрований пароль і маючи програму шифрування можна знайти дійсний пароль, наприклад, за допомогою паралельних обчислень. На цю процедуру треба часу тим більше, чим довшими будуть паролі і чим довше буде виконуватись шифрування. Враховуючи усі ці обставини, можна досягти стану, при якому витік зашифрованих паролів не буде загрозою для системи голосування.

Голоси виборців у зашифрованому вигляді потрапляють безпосередньо до діючої програми підрахунку голосів, де вони розшифруються і до моменту зарахування повинні зберігатись в оперативній пам’яті у вигляді значень змінних цієї програми. Після здійснення підрахунку зберігати голоси з точки зору забезпечення секретності не є доцільним. Однак у відомих сучасних СДГ ці голоси зберігають у пам’яті сервера для можливості перевірки самим виборцем правильності зарахування його голосу. У разі, якщо цю перевірку зробить хтось інший, крім самого виборця, то це буде фактом порушення права на збереження таємниці голосів. Тому зберігання голосів виборців після їх зарахування є негативним фактором, що сприяє можливості порушення прав виборців. З нашої точки зору зберігання результатів персонального вибору після закінчення підрахунку не є доцільним. Так само, як вкинувши паперовий бюлетень в урну, виборець вже не має можливості перевірити як він проголосував. Необхідність в збереженні голосів виборців в естонській програмі [41] пояснюється тим, що через низьку швидкість обробки електронних голосів було прийнято рішення збільшити час на електронне голосування і надати можливість виборцям змінювати свій голос до певного моменту. Але Естонія належить до демократичних країн. Ментальність її громадян дозволяє припустити наявність вказаного вище «слабкого місця» у захисті.

У нашому випадку для забезпечення гарантій захисту доцільно використати той факт, що в існуючих ОС не має доступу до оперативної пам’яті діючої програми з боку інших програм. Обмін даними між програмами

реалізується тільки через файли. Але не можна виключати можливість проникнення до оперативної пам’яті діючої програми за допомогою позаштатних засобів. Надійним методом захисту від такої можливості є повний контроль засобів ОС.

Таким чином можна вважати, що за умов відкритості ОС та прикладного ПЗ існує можливість забезпечення стовідсоткових гарантій досконалості захисту інформації в операційному середовищі СДГ.

1.4. Аналіз методів, що забезпечують захищений обмін даними через незахищені канали в системах дистанційного волевиявлення

Найбільш поширеним способом обміну даними в СДГ між клієнтською частиною програмного забезпечення, що реалізує функції інтерфейсу для спілкування з виборцем, та серверною програмою, яка займається підрахунком голосів, є спосіб *end-to-end* (скорочено *E2E*). Цей спосіб покладено в основу СДГ, що планується використати на президентських виборах 2016 року у США [14]. Перевага даного підходу полягає у тому, що конфіденційна інформація про волевиявлення виборця, яка кодується у комп’ютері виборця, не може бути декодована ніде, крім програми підрахунку голосів. Забезпечення досконалої секретності передавання даних між виборцем і сервером СДГ за таких умов повністю залежить від досконалості обраних засобів криптографічного захисту.

Відомо, що єдиним методом, який забезпечує абсолютний захист від компрометації інформації є шифр Вернама, який називають *one-time pad* (одноразовий блокнот) [43]. Складність реалізації цього методу полягає у необхідності формування по-справжньому випадкової (не псевдовипадкової) послідовності символів, яка повинна не бути відомою нікому, крім відправника та одержувача інформації. Довжина цієї послідовності повинна бути не меншою за довжину повідомлення, яке треба пересилати у зашифрованому вигляді. Тільки за цих умов може бути гарантована повна секретність передавання даних. Про важливість необхідності гарантування збереження

таємниці голосів під час дистанційного голосування свідчить заборона урядом Швейцарії у 2015 році у дев’яти з 27 кантонів засобів для електронного голосування (розробка компанії *Unisys*, яка базується у США) через виявлену під час аудиту можливість витоку конфіденційної інформації про волевиявлення виборців [44].

Проаналізуємо методи, які б дозволяли забезпечити виконання вимог щодо реалізації схеми Вернама для систем дистанційного голосування. Сам алгоритм Вернама є одним з найпростіших алгоритмів захисту. Він полягає в додаванні за модулем два до кожного біту даних випадкового біту. У такому вигляді дані можна передавати без жодного ризику щодо витоку інформації, бо єдиним методом розшифрування є таке саме додавання послідовності випадкових бітів, які нікому невідомі, окрім відправника і одержувача інформації. Таким чином, для забезпечення абсолютних гарантій захисту залишається створити наступне.

1) Обрати метод отримання дійсно випадкових бітових послідовностей.

2) Забезпечити, щоб випадкові бітові послідовності були однаковими у виборця і на сервері підрахунку голосів.

3) Забезпечити умови, щоб ці однакові випадкові бітові послідовності не могли стати відомими ніякій третій стороні.

Розглянемо питання генерування випадкових бітових послідовностей. Такі послідовності досить просто генерувати на кожному комп’ютері , але необхідно дотримуватись умов щодо забезпечення необхідної швидкості генерування випадкових бітових послідовностей. У СДГ оцінку необхідної частоти генерування можна отримати шляхом ділення кількості інформації, яку необхідно шифрувати у кожному сеансі голосування на тривалість обслуговування цього сеансу, що може бути представлено виразом

$$F_n = \frac{I_i + I_a + I_v}{\tau}, \tag{1.1}$$

де F_n – необхідна частота генерації випадкових бітів, Гц;

I_i – кількість інформації для ідентифікації виборця, біт;

I_a – кількість інформації для автентифікації виборця, біт;

I_v – кількість інформації про волевиявлення виборця, біт;

τ – тривалість сеансу обслуговування виборця, с.

Оскільки генерувати випадкові біти з меншою частотою простіше, ніж з високою, нас буде цікавити тільки верхня оцінка частоти. Тому ми будемо завищувати оцінки кількості інформації у виразі (1.1) та занижувати оцінку тривалості сеансу обслуговування виборця. Обираючи довжини паролю та ідентифікатору по 20 байт або по 160 біт, а кількість бюлетенів для одночасного голосування оберемо 10 з 256 варіантами вибору у кожному бюлетені, то загальна кількість інформації, що підлягає захисту у сеансі обслуговування виборця не перевищуватиме 250 бітів. Для тривалості сеансу 5 секунд отримаємо необхідну швидкість генерації випадкових бітів 50 Гц.

Для такої швидкості генерації випадкових бітів цілком задовільний результат можемо отримати за допомогою методу, який рекомендовано у стандарті України для криптографічного захисту інформації [45]. Цей метод полягає в тому, щоб у деякі моменти часу обирати біти, які відповідають мінімальному розряду комп'ютерного таймера. Цей таймер лічить кількість мілісекунд від нуля годин 1 січня 1970 року. У разі високої частоти моментів вибору значень таймера можемо отримати незадовільний результат через те, що таймер може не встигати змінювати своє значення. Задовільний результат будемо отримаємо у разі випадкових моментів з частотою значно нижчою за 1 кГц. Зауважимо, що тактовий генератор, який забезпечує роботу процесора, і генератор таймера є різними, кожен з яких має свою нестабільність. Ця непередбачувана нестабільність є фактором, який сприяє отриманню дійсно випадкової бітової послідовності.

Послідовності випадкових бітів, які можуть бути утворені на комп'ютері виборця і сервері необхідно перетворити на однакові. При цьому необхідно забезпечити неможливість їх розкриття.

Оскільки формування дійсно випадкових послідовностей може бути реалізовано в межах ресурсів кожної діючої програми з використанням тільки внутрішніх змінних, то значення цих послідовностей будуть недоступними для інших програм, бо засобами ОС кожній програмі на час виконання надається окрема ділянка оперативної пам'яті, до якої інші програми не мають доступу. Для повної гарантії неможливості розкриття значень внутрішніх змінних у системах дистанційного голосування рекомендовано використовувати відкрите програмне забезпечення, що ліквідує можливість сумніву у вірності роботи програм [4].

Залишається перетворити дві різні випадкові послідовності в однакові, що може бути реалізовано з використанням відкритого каналу зв'язку за допомогою алгоритму Диффі-Геллмана [46]. Для цього слід виконати наступні дії

$$A=q^a \bmod P,$$
$$B=q^b \bmod P, \qquad (1.2)$$

де q – твірний елемент мультиплікативної групи;

P – кількість елементів цієї мультиплікативної групи;

a, b – випадкові числа, що утворені у вигляді бітових послідовностей з боку клієнта та сервера, відповідно.

Отриманими значеннями A та B клієнт та сервер повинні обмінятись з використанням відкритого каналу зв'язку. Знаючи параметри q, P, A та B за умов достатньо великих P неможливо розкрити значення a та b, без чого не можна отримати ключову інформацію.

Коли комп'ютер виборця, де було утворено число a, приймає число B, знаходять число Z з виразу

$$Z=B^a \bmod P, \tag{1.3}$$

а на сервері, де було утворено число b, знаходять число K з виразу

$$K=A^b \bmod P. \tag{1.4}$$

Легко впевнитись, що $Z=K$, бо

$$Z=q^{ba} \bmod P,$$
$$K=q^{ab} \bmod P. \tag{1.5}$$

Після такого обміну буде створено можливості для подальшого обміну секретною інформацією з використанням відкритого каналу зв’язку.

Наявність розглянутих методів свідчить про можливість забезпечення повної секретності під час передавання даних через незахищені канали мережі Інтернет в системах дистанційного волевиявлення.

1.5. Постановка завдань досліджень

Основний недолік сучасних СДВ полягає у відсутності дієвих механізмів контролю над засобами самої системи з боку спільноти виборців у цілому, а також стовідсоткових гарантій забезпечення конфіденційності та цілісності інформації в каналах обміну даними. Через це не може буде забезпечена абсолютна гарантованість збереження таємниці голосування та свободи волевиявлення в умовах адміністративного тиску. Виборець не є упевненим, що результати голосування не будуть спотворені, а персональні дані не розголошені. Тому метою даної роботи є розробка наукових та технічних рішень у сфері захисту інформації, які б забезпечили беззаперечні гарантії неможливості виникнення порушень цілісності результатів волевиявлення та конфіденційності персональних даних голосуючих за умов повної недовіри до

усіх без винятку учасників процесу ДВ, що усуває будь-які підстави для недовіри з боку голосуючих щодо можливості реалізації вказаних порушень, а також можливої спроби тиску на виборців і інших учасників цих процедур, зокрема членів конкурсних (виборчих) комісій методами підкупу, шантажу, залякуванню і т.п.

Описані в цьому розділі «слабкі місця» у захисті систем ДВ, через які можливе виникнення специфічних загроз для інформації, відсутність протидії котрим підриває довіру громадян до того, що результати голосування будуть об'єктивно відображати істинну волю виборців, а також наукові завдання щодо нейтралізації цих загроз представлено у табл. 1.1.

Таблиця 1.1

Загрози, що впливають на рівень довіри до коректної роботи СДВ

№ з.п.	Визначення загрози	Наукові завдання з нейтралізації визначених загроз
1	Реєстрація фіктивних виборців	1. Розробити **модель СДВ**, що гарантує цілісність результатів волевиявлення та збереження таємниці голосів в умовах недовіри до всіх без винятку осіб, що приймають участь у розробці, створенні та обслуговуванні СДВ. 2. Розробити **метод спостереження** в реальному часі за станом сервера і діями адміністратора СДВ з боку необмеженого кола будь-яких користувачів Інтернету, що виключає можливість виникнення непомічених порушень прийнятої політики безпеки та усуває підстави для недовіри до СДВ .
2	Заміна системного ПЗ сервера на нештатне	
3	Модифікація штатного системного ПЗ сервера	
4	Виконання позаштатної команди управління сервером	
5	Підробка прикладного ПЗ сервера	
6	Нелегальна фізична підміна сервера	
7	Доповнення серверного обладнання нештатними засобами з метою реалізації *MITM* (атаки посередника)	
8	Підміна результатів голосування у процесі розсилки	
9	Підробка результату підрахунку голосів	
10	Примус виборців віддавати свій голос усупереч їх власному бажанню	Розробити метод нейтралізації спроб здійснення будь-яких видів тиску на учасників процедур волевиявлення.
11	Порушення цілісності та (або) конфіденційності інформації при обміні даними через Інтернет	1. Розробити **метод досконало захищеного обміну даними**, що гарантує цілісність та конфіденційність інформації при обміні даними через Інтернет. 2. Розробити **метод отримання чисто випадкових бітових послідовностей**, засобами виключно типового клієнтського обладнання масового виробництва.
12	Перехоплення автентифікаційних даних виборців з метою підміни голосуючої особи	

Слід зауважити, що загрози порушення доступності ресурсів СДВ можуть призвести до зриву процедур волевиявлення, але не позначаються на рівні довіри громадян до коректної роботи СДВ.

Технологічний цикл функціонування СДВ з визначеними періодами часу виконання окремих процедур та позначеними моментами найбільш важливих подій показано на рис. 1.1.

Рис. 1.1. Технологічний цикл функціонування СДВ, де ПБД ДВ – період заповнення бази даних претендентів на дистанційне волевиявлення, а ПП ДВ – період підготовки до дистанційного волевиявлення (завантаження електронних бюлетенів після остаточного коригування)

Таким чином для реалізації мети даної роботи слід виконані наступні завдання:

1. Розробити **модель СДВ**, що гарантує цілісність результатів волевиявлення та збереження таємниці голосів в умовах недовіри до всіх без винятку осіб, що приймають участь у розробці, створенні та обслуговуванні СДВ.

2. Розробити **метод спостереження** в реальному часі за станом сервера і діями адміністратора СДВ з боку необмеженого кола будь-яких користувачів Інтернету, що виключає можливість виникнення непомічених порушень прийнятої політики безпеки та усуває підстави для недовіри до СДВ.

3. Розробити **метод досконало захищеного обміну даними**, що гарантує цілісність та конфіденційність інформації при обміні даними через Інтернет.

4. Розробити **метод отримання чисто випадкових бітових послідовностей**, засобами виключно типового клієнтського обладнання масового виробництва.

5. Розробити метод нейтралізації спроб здійснення будь-яких видів тиску на учасників процедур волевиявлення.

Матеріали щодо виконання вищеназваних завдань містяться у наступних трьох розділах роботи.

Висновки до першого розділу

1.1. Світовий досвід впровадження у суспільно-політичні відносини систем ДВ свідчить про те, що ці системи у порівнянні із традиційними забезпечують принципово нову якість обслуговування учасників процесу волевиявлення - можливість здійснювати акт волевиявлення, користуючись широкодоступними засобами зв’язку, із будь-якого місця розташування у зручний для будь-якої особи момент часу. Окрім того, забезпечується індивідуальний контроль з боку виборця правильності фіксації даних щодо його вибору, більш висока швидкість і точність підрахунку голосів. У той же час, технічні завдання щодо захисту даних волевиявлення ускладнюються, оскільки ці дані мають транспортуватися через відкриті, тобто незахищені, канали зв’язку, деякі елементи виборчого процесу втрачають прозорість, коли вони реалізовані у мультисервісному середовищі функціонування сучасних інформаційних технологій, а на виборців під час волевиявлення може чинитися безпосередній силовий тиск. Особливу увагу вище зазначеним факторам слід приділити під час побудови СДВ у країнах, де існує суттєва корупційна складова в роботі суспільно-політичних конструкцій, що породжує тотальну недовіру до коректності побудови цих конструкцій з боку громадян. Для подолання проблеми недовіри при побудові СДВ в таких умовах необхідно

забезпечити абсолютну конфіденційність особових даних виборців та цілісність результатів волевиявлення, а також повну відкритість та контрольованість середовища дистанційного волевиявлення. Тільки за цих умов зникнуть підстави для сумнівів щодо об’єктивності результатів волевиявлення.

1.2. Аналіз публікацій показав, що існуючі методи криптографічного захисту дозволяють забезпечити гарантовану конфіденційність та цілісність інформації під час її зберігання на серверах або під час обміну даними через незахищене середовище. Проте, поки що, ці методи в існуючих системах ДВ у достатній мірі не реалізовані. Проблема недовіри до систем ДВ з боку виборців, поки що, не знята, що вказує на доцільність розробки методів забезпечення гарантованої конфіденційності та цілісності інформації у системах ДВ.

1.3. Міжнародний досвід експлуатації існуючих захищених мережевих комп’ютерних систем свідчить про те, що гарантувати абсолютний захист від проникнення зловмисників до конфіденційних даних у таких системах практично неможливо. Але для СДВ, функції яких є досить чітко обмеженими, за умов відкритості операційної системи та прикладного програмного забезпечення може існувати можливість абсолютно досконалого захисту інформації (згідно термінології К. Шеннона) в серверному операційному середовищі, якщо забезпечити повний контроль середовища функціонування СДВ. Отже, розробка методів забезпечення повної відкритості (прозорості) та контрольованості середовища функціонування СДВ є актуальним завданням.

1.4. Головною метою створення методів захисту результатів дистанційного волевиявлення є забезпечення гарантованої захищеності даних від спотворення результатів волевиявлення на всіх етапах виборчого процесу від реєстрації виборців до отримання остаточних результатів їхнього вибору. Тільки таким чином можливо подолати недовіру виборців щодо доцільності використання технологій дистанційного волевиявлення. Тому слід визначити умови, за яких реалізація гарантованого захисту інформації у СДВ є принципово можливою.

1.5. Виявлено «слабкі місця» у захисті існуючих СДВ, зокрема специфічні

загрози для інформації, відсутність протидії котрим підриває довіру громадян до того, що результати голосування будуть об’єктивно відображати істинну волю виборців. Визначено завдання щодо нейтралізації цих загроз.

РОЗДІЛ 2
МЕТОДИ ЗАБЕЗПЕЧЕННЯ ГАРАНТОВАНОЇ КОНТРОЛЬОВАНОСТІ ПРОЦЕСІВ ДИСТАНЦІЙНОГО ВОЛЕВИЯВЛЕННЯ

2.1. Основний концептуальний принцип побудови захищених систем технічної підтримки процесів дистанційного волевиявлення

Загальнонаціональні системи дистанційного волевиявлення (СДВ), як показує практика [3, 42], завжди створюються для одночасного функціонування з існуючими системами, котрі передбачають необхідність фізичної присутності виборців на виборчих дільницях в момент голосування. Тому при побудові захищеної СДВ слід враховувати необхідність сумісного функціонування створюваної СДВ із існуючою протягом невизначеної тривалості часу. Через це деякі суттєві параметри існуючої системи волевиявлення мають залишатися незмінними при побудові нової. У даному випадку створювану СДВ побудуємо таким чином, щоб вона була здатна функціонувати сумісно із існуючою виборчою системою України.

Виборча система України складається з наступної ієрархії виборчих комісій (див. рис.2.1):

1) центральна виборча комісія;

2) обласні, міські (інших адміністративних одиниць) виборчі комісії (частина з котрих функціонує на постійній основі);

3) окружні виборчі комісії (у 225 одномандатних округах, функціонують на постійній основі);

4) територіальні виборчі комісії (станом на 26.10.2015 у 669 регіонах);

5) дільничні виборчі комісії (у 33544 виборчих дільницях, функціонують у період проведення виборчих кампаній), кожна з яких обслуговує до 2500 виборців.

Отже, існуюча виборча система України має ієрархічну організаційну структуру. Так що, необхідно визначити функціональність створюваної СДВ у розрізі ієрархічних рівнів існуючої в Україні системи проведення виборів.

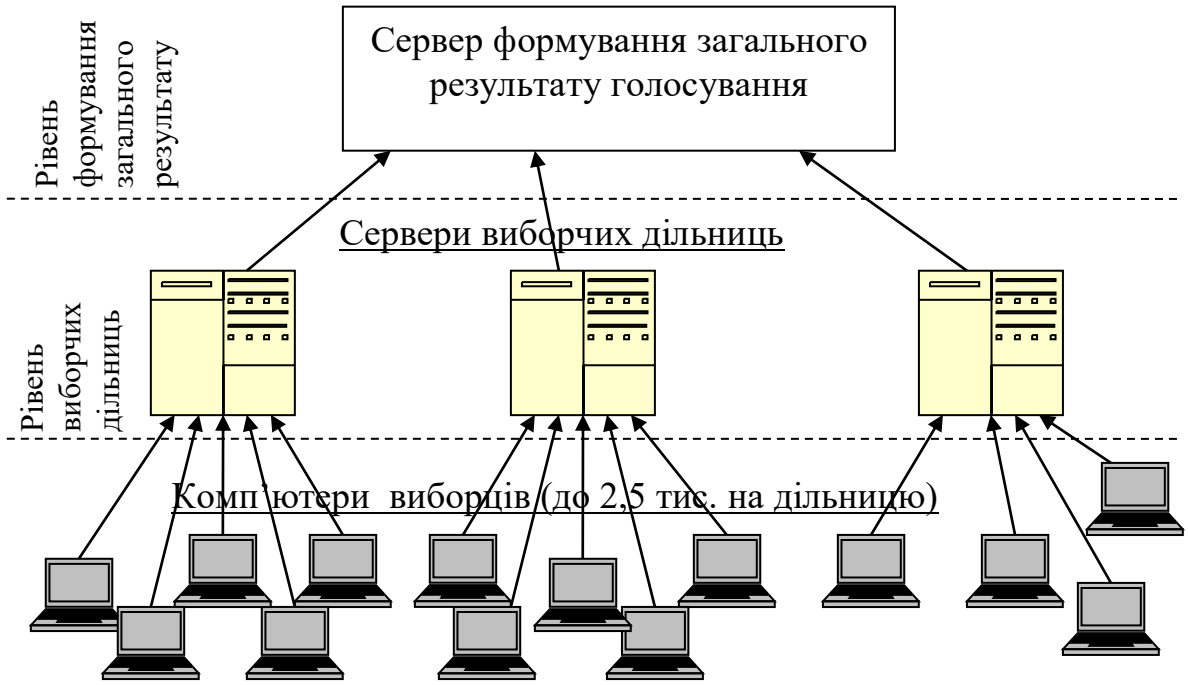

Рис. 2.1. Організаційна структура захищеної СДВ в Україні

У даному випадку підлягають узгодженню із існуючою системою дві наступні функції:

1) первинний підрахунок голосів, який відбувається на рівні виборчих дільниць;

2) формування загального результату голосування, яке відбувається на вищих ієрархічних рівнях даної організаційної структури.

Оскільки на всіх вищих рівнях виконується тільки функція додавання, то доцільно, у даному випадку, усі вищі рівні розглядати, як один рівень формування загального результату волевиявлення.

Найбільш небезпечним місцем, с точки зору можливості створення загроз, є сервер виборчої дільниці. Метою спроб реалізації цих загроз може бути втручання у процес підрахунку голосів для його фальсифікації та/або розкриття даних щодо того, хто і як голосував. В якості суб'єктів, що мають найбільші можливості щодо реалізації цих загроз, перш за все, розглядається персонал виборчих дільниць.

Перелік загроз, які можуть бути реалізовані персоналом на сервері виборчої дільниці, разом з методами протидії цим загрозам представлений у табл. 2.1.

Таблиця 2.1

Загрози, які можуть бути реалізовані персоналом, що обслуговує сервер виборчої дільниці

№	Опис шляху реалізації загрози	Метод протидії для спостерігача
1	Заміна операційної системи сервера	Порівняння файлів ОС з відомими штатними
2	Заміна файлів операційної системи або занесення нештатних файлів	Порівняння каталогів і файлів ОС зі штатними
3	Виконання позаштатної команди управління сервером	Контроль за виконанням команд управління
4	Заміна серверного обладнання	Контроль параметрів активних процесів ОС
5	Доповнення серверного обладнання нештатними засобами з метою реалізації *MITM* (атаки посередника)	Контроль характеристик трафіку
6	Завантаження нештатної програми	Порівняння тексту програм з відомим штатним
7	Доповнення штатної програми нештатними командами	Порівняння тексту програм з відомим штатним
8	Несвоєчасне виконання штатних дій	Перевірка дій за розкладом

Крім протидії загрозам, які перелічені у табл.2.1, необхідно також забезпечити захист від загроз, що можуть бути реалізованими поза меж домену безпеки сервера виборчої дільниці. Перелік цих загроз разом з методами протидії представлений у табл. 2.2.

Перелік інформаційних потоків і інших об’єктів, які підлягають захисту у створюваній СДВ, надано у табл. 2.3.

Таблиця 2.2

Загрози, які можуть бути реалізовані поза сервером виборчої дільниці

№	Опис загрози	Метод протидії
1	Перехоплення даних під час передавання	Створення захищеного каналу (криптографічний захист даних)
2	Заміна даних під час передавання	Використання протоколу, який забезпечує цілісність даних
3	Проникнення до серверу через засоби дистанційного доступу	Створення інтерфейсів, які від віддалених користувачів не сприймають нештатні запити
4	Заміна даних про результат голосування	Порівняння даних з довідками, отриманими через захищений канал

Таблиця 2.3

Інформаційні потоки та об’єкти, які потребують захисту в СДВ

№	Опис об’єкту захисту	Потрібна властивість
1	Потік даних від комп’ютера виборця до сервера виборчої дільниці	Конфіденційність, цілісність та доступність
2	Усі без винятку файли на сервері виборчої дільниці у реальному часі виборчого процесу	Цілісність та контрольованість
3	Команди управління сервером виборчої дільниці (від адміністратора сервера)	Цілісність та контрольованість
4	Файли, які завантажуються на сервер виборчої дільниці (адміністратором сервера)	Цілісність та контрольованість
5	Дані, що пов’язані зі процесом підрахунку голосів, на сервері виборчої дільниці	Конфіденційність та цілісність
6	Потік даних від сервера виборчої дільниці до сервера формування загального результату голосування	Цілісність та контрольованість
7	Файл з даними про результати голосування на сервері формування загального результату	Цілісність та контрольованість

Основний принцип побудови захищених систем технічної підтримки процесів ДВ, який реалізовано у даній роботі, полягає у наступному. З метою досягнення основної мети даної роботи, що спрямована на забезпечення гарантованого захисту інформації і усунення підстав для можливої недовіри до об’єктивності результатів голосування, СДВ створюється таким чином, щоб виборцям була надана можливість повного і безперервного контролю у реальному часі подій, що відбуваються на сервері виборчої дільниці. При цьому виборці можуть контролювати цілісність усіх без винятку файлів програмного забезпечення і спостерігати за виконанням штатних дій обслуговуючим персоналом, протягом усього періоду роботи серверів виборчих дільниць.

Слід зауважити, що забезпечення відкритості та контрольованості середовища функціонування СДВ має не призводити до зменшення ефективності захисту від порушень конфіденційності та цілісності інформації щодо результатів волевиявлення. Ефективність цього захисту має не тільки зберігатися, а навіть збільшуватися шляхом забезпечення абсолютної неможливості фальсифікації результатів волевиявлення або визначення, хто як голосував, зокрема шляхом використання досконало стійких методів захисту

інформації. у т.ч. в умовах екстремального насильницького впливу, фізичного або морального, на виборця безпосередньо під час здійснення акту волевиявлення.

З точки зору ТЗІ функціональний профіль захищеності [47] створюваної СДВ має містити у своєму складі, поряд з іншим, підмножину послуг безпеки (ПБ) [48], що забезпечують абсолютну (повну) відкритість та контрольованість середовища функціонування цієї системи. Специфікації ПБ, що забезпечують контрольованість середовища, мають відображати не тільки особливості реалізації функцій спостереження за подіями у середовищі СДВ, але і забезпечувати можливість контролю усіх без винятку об’єктів та дій суб’єктів виборчого процесу з боку будь-яких без винятку зацікавлених осіб у реальному часі функціонування сервера. Такий контроль, на відміну від пасивного спостереження, передбачає активні дії спостерігачів.

Підкреслимо, що через прийнятий за основу побудови захищеної СДВ принцип недовіри до усіх без винятку учасників виборчого процесу, включаючи персонал, який обслуговує систему, можливість виконання функцій контролю повинні мати усі без винятку учасники виборчого процесу, а також будь-які інші особи, які з тих чи інших причин зацікавлені в отриманні об’єктивних результатів волевиявлення, зокрема громадські контролери. Відсутність обмежень у доступі до процедур контролю є основною умовою подолання будь-якої недовіри до створюваної СДВ. Через таку особливість виникає ряд вимог щодо вибору операційної системи (ОС) для об’єктів інформаційної діяльності, що підтримують виборчий процес, та засобів розробки прикладного програмного забезпечення СДВ.

2.2. Вимоги щодо забезпечення гарантованої контрольованості процесів дистанційного волевиявлення

2.2.1. Визначення поняття абсолютно відкритого середовища СДВ

Під поняттям абсолютно відкритого середовища СДВ будемо розуміти середовище функціонування глобальної комп'ютерної мережі, що базується на використанні ресурсів Інтернет і задовольняє наступним вимогам:

1) усе без винятку програмне забезпечення (ПЗ) є відкритим і доступним без будь-яких обмежень для перевірок і випробувань;

2) апаратні засоби є стандартними із відомою структурою побудови за архітектурними, топологічними та функціональними ознаками;

3) існує необмежена кількість реально функціонуючих точок дистанційного доступу через Інтернет з використанням відкритого стандартного мережного протоколу до усіх без виключення серверів виборчих дільниць для необмеженої кількості користувачів ресурсів СДВ;

4) усі без винятку користувачі мають право контролю змісту усіх без винятку комп'ютерних файлів, що встановлені на серверах виборчих дільниць (можливо, крім тої обмеженої кількості закритих для читання файлів ядра операційної системи, зміст яких щодо можливостей модифікацій чи підмін не викликає сумнівів);

5) усі без винятку користувачі, зокрема громадські контролери, мають право контролю відсутності подій перезапуску ОС від моменту їхнього запуску, що встановлені на усіх активних макроелементах СДВ, до моменту фізичного виключення серверів дистанційного волевиявлення (тобто, після закінчення виборчої кампанії);

6) усі без винятку користувачі мають право і можливість здійснювати активний контроль (за допомогою спеціальних команд) усіх процесів, які знаходяться у стадії виконання.

2.2.2. Критерії, яким має відповідати ОС виборчої дільниці

1) Операційна система (ОС) на сервері виборчої дільниці повинна бути відкритою і простою для забезпечення можливості її детальної перевірки будь-якою зацікавленою особою, що має стандартний рівень знань спеціаліста з комп'ютерних технологій.

2) Специфікації ОС, у т.ч. вихідні коди комп'ютерних програм, мають бути опублікованими та доступними в Інтернеті для встановлення на будь-яких хостах з метою забезпечення можливості порівняння файлів серверної ОС із файлами ОС, що встановлені на комп'ютерах користувачів Інтернет.

3) Функціональність ОС повинна забезпечувати гарантовано досконалий захист від несанкціонованого доступу (НСД) з тим, щоб адміністратор сервера не мав можливості перекладати свою відповідальність на формально невизначену особу.

4) ОС повинна забезпечувати доступ до сервера кожному виборцю для перевірки усіх файлів і процесів без можливості заподіяння будь-якої шкоди нормальній роботі сервера.

2.2.3. Критерії, яким мають відповідати засоби програмування

1) Мінімальна кількість та максимальні простота і популярність обраних мов програмування для створення прикладних програм (ПП), з метою спрощення перевірки ПП з боку громадських контролерів.

2) Можливість функціонування ПП на різних програмно-апаратних платформах з метою зняття зайвих обмежень під час перевірок ПП будь-якими особами на будь-якому обладнанні.

3) Повна відкритість інструментальних програмних засобів, які обрані для створення і забезпечення функціонування ПП на сервері виборчої дільниці.

2.2.4. Вимоги щодо забезпечення повної контрольованості ОС

Зрозуміло, що існують певні обмеження при побудові повністю контрольованого середовища функціонування СДВ. В першу чергу, це стосується ОС, яка повинна задовольняти наступним вимогам:

1) забезпечувати абсолютний захист від НСД в період виконання заданої прикладної задачі;

2) забезпечувати надання спеціальних прав користувачам для контролю усіх файлів, як самої ОС, так і всіх інших файлів у її середовищі;

3) унеможливлювати створення скритих файлів;

4) реалізовувати функціональність команди, за допомогою якої користувачі мали б можливість впевнитись, що робота СДВ не переривалась у часі на протязі тривалості виборчої кампанії (доступ до цієї команди має бути необмеженим);

5) реалізовувати функціональність команди, за допомогою якої користувачі мали б можливість безперешкодної перевірки у реальному часі стану виконання усіх без винятку процесів і команд управління (доступ до цієї команди має бути не обмеженим);

6) унеможливлювати виникнення будь-якого шкідливого впливу на роботу СДВ в умовах необмеженого доступу до команд контролю параметрів, що визначають стан роботи системи;

7) реалізовувати можливість встановлення на загальнодоступних комп'ютерах виборців, що підключені до Інтернет, безпосередньо із мережі необхідної ОС та прикладного ПЗ СДВ з функціями контролю середовища функціонування;

Кількість файлів СДВ має бути якомога меншою, щоб не було ускладнень під час перевірок через велику кількість файлів.

Система повинна легко і швидко встановлюватись, щоб не утворювати будь-які зайві труднощі під час її перевірки.

2.2.5. Вимоги щодо забезпечення відкритості та контрольованості прикладного ПЗ

1) Прикладне ПЗ, що встановлене на сервері виборчої дільниці, повинно бути відкритим і простим, з метою забезпечення можливості його детальної перевірки будь-якою зацікавленою особою, що має стандартний рівень знань спеціаліста з комп'ютерних технологій.

2) Специфікації прикладного ПЗ, у т.ч. вихідні коди комп'ютерних програм, мають бути опублікованими та доступними в Інтернеті із можливістю безпосереднього встановлення цих програм на будь-яких хостах з метою забезпечення можливості порівняння файлів серверного прикладного ПЗ із

файлами прикладного ПЗ, що встановлено на комп’ютерах користувачів Інтернет.

3) Мови програмування для створення цього забезпечення повинні бути широко відомими і якомога більш популярними.

4) Кількість мов програмування повинна бути мінімальною.

5) Тексти програм повинні бути чіткими і зрозумілими.

6) Кількість програмних модулів повинна бути мінімальною.

7) Бажано, щоб серверна програма розміщувалась у єдиному модулі, з метою спрощення контролю за її роботою.

8) Файли даних повинні мати якомога простішу структуру з метою спрощення процедури їх перевірки.

9) Конфіденційні дані у файлах повинні бути зашифровані таким чином, щоб їх дешифрування протягом визначеного періоду було практично неможливим.

10) Обробка конфіденційних даних повинна відбуватись виключно в оперативній пам’яті діючих програм, після чого конфіденційні дані повинні знищуватись.

11) Інтерфейси для обміну даними зі зовнішніми програмно-апаратними засобами повинні унеможливлювати проникнення до системи будь-якої інформації зловмисного характеру.

12) Мають бути створені умови для внесення пропозицій з метою вдосконалення роботи прикладного ПЗ. Надання таких можливостей, безумовно, буде сприяти досягненню головної мети створення відкритих комп’ютерних систем, а саме, – усуненню будь-яких підстав для недовіри цим системам.

2.2.6. Вимоги щодо забезпечення відкритості технології обробки інформації у СДВ

Технологія обробки інформації у СДВ повинна бути такою, щоб сприяти контролю за роботою системи і полегшити виявлення можливих відхилень від

штатного режиму функціонування СДВ. Ця технологія має відповідати наступним вимогам.

1) Повинні бути чітко визначені моменти або інтервали часу щодо дій, які будуть виконуватись у системі, за умов відсутності відхилень від режиму штатної роботи. До таких дій слід віднести команди управління роботою системи, завантаження файлів та запуск програм. Для спрощення контролю за роботою системи у реальному часі треба надати контролюючим особам вільний доступ до розкладу цих дій.

2) Мають бути розроблені конкретні рекомендації щодо контролювання роботи системи у реальному часі із прив'язкою до розкладу штатних дій.

3) Бажано, щоб було створено спеціальний ресурс в мережі Інтернет, на якому була б розміщена вся необхідна інформація щодо контролю за роботою системи і надана можливість оперативного звернення до відповідальних осіб для інформування про виявлені нештатні дії.

2.3. Метод забезпечення гарантованої контрольованості процесів дистанційного волевиявлення

Перед пошуком шляхів забезпечення повної відкритості та контрольованості середовища функціонування СДВ доцільно визначитися із базовими поняттями.

З точки зору ТЗІ середовище функціонування СДВ являє собою (N+1) доменів безпеки (див.рис.2.1), де N – кількість виборчих дільниць, що об'єднані у зіркоподібну структуру (у центрі зірки сервер формування загального результату). Необхідно забезпечити повну відкритість та контрольованість усіх процесів та процедур впродовж усієї виборчої кампанії, що мають здійснюватися у межах кожного із доменів безпеки. Типовим макроелементом середовища СДВ є комп'ютерне середовище відповідного сервера.

2.3.1. Середовище функціонування сервера СДВ

Структуру середовища функціонування сервера СДВ доцільно представити у вигляді ієрархії рівнів, що показана на рис.2.2 [49],

Рис. 2.2. Структура середовища функціонування сервера СДВ

а вибір характеристик цієї структури здійснювати у наступній послідовності: операційна система (ОС); системне програмне забезпечення (СПЗ); прикладне програмне забезпечення (ППЗ).

2.3.2. Вибір та встановлення операційної системи на сервері СДВ

Із вимог до ОС щодо повної відкритості її специфікацій та середовища її функціонування, які сформульовані у підрозділі 2.2, витікає необхідність контролю усіх без винятку файлів і процесів. Тому доцільно мінімізувати кількість функцій, що виконує ОС. Дійсно, щоб забезпечити контрольованість ОС в умовах обмеженого часу, вкрай важливо мінімізувати кількість файлів, що підлягають контролю. Перш за все, слід позбутися функцій, які непотрібні для вирішення завдань, що покладені на СДВ. Бажано залишити мінімум функцій ОС, без яких неможливо розв'язання задач СДВ. Проте розвиток сучасних ОС, як показує практика, спрямований на розширення їхніх функціональних можливостей [50], через що вони все більш складнішають. Контроль середовищ

функціонування таких ОС уявляється вельми утрудненим. Для СДВ, враховуючи їхню важливу роль для суспільства, доцільно було б створити спеціальну ОС, яка б повністю задовольняла зазначеним вище вимогам, але це пов'язано із значними витратами часу і ресурсів. У нашому випадку доцільно обмежитися вибором прийнятної ОС серед множини вже існуючих ОС, а виявлені «слабкі місця» у її захисті намагатися нейтралізувати за рахунок адміністративно-організаційних заходів.

У першому розділі було зроблено аналіз існуючих відкритих ОС і пошук серед них найбільш придатної для нашого випадку. Критерії пошуку – мінімально можлива функціональність і максимально можлива захищеність від НСД. Як результат, було обрано операційну систему *OpenBSD* у мінімальній конфігурації. Характеристики цієї системи представлені у таблиці 2.4.

Таблиця 2.4

Характеристики ОС *OpenBSD* у мінімальній конфігурації

Назва характеристики або команди	**Значення характеристики**
Доступ до вихідних текстів ОС	Повний
Захист від несанкціонованого доступу	За певних умов абсолютно досконалий
Створення користувача-контролера	Можливо
Створення скритих файлів	Неможливо
Команда для контролю незмінності ОС	*top* (будуть незмінні 20 *PID*)
Команда контролю стану процесів	*ps aux*
Створення загроз діями контролера	Неможливо
Обмеження прав адміністратора	Можливо
Блокувння користувача з повними правами	Можливо

Виявлене «слабке місце» цієї ОС: хоч більша частина характеристик даної ОС відповідає названим вище вимогам, проте деякі файли ОС *OpenBSD* неможливо скопіювати для перевірки, що не у повній мірі відповідає цим вимогам. Оскільки рейтинг і відгуки користувачів *OpenBSD* щодо захищеності системи від НСД є високими, то не існує приводу для недовіри цій ОС. Але через те, що ми не довіряємо персоналу, котрий може підмінити або модифікувати ОС, є необхідність контролю цілісності встановленої ОС.

Процедуру такого контролю краще за все виконати одразу після етапу встановлення ОС, бо з кожним наступним етапом кількість файлів буде збільшуватись і процес контролю буде ускладнюватись.

Процес встановлення ОС починається із створення на CD *ISO*-образу ОС за допомогою будь-якого комп'ютера, який підключений до Інтернету і має стандартні засоби для створення *ISO*-образів. Після встановлення ОС адміністратор повинен створити користувача із правами громадського контролера і зробити паузу, щоб контролери за цей час встигли виконати перевірку справжності ОС. Кожен з контролерів повинен встановити таку ж ОС на своєму комп'ютері для порівняння між собою файлів на двох комп'ютерах.

2.3.3. Доведення можливості реалізації достовірного контролю ОС *OpenBSD* з використанням засобів самої ОС.

Перевірки, які може виконати контролер для визначення справжності встановленої на сервері ОС, полягають у порівнянні множини характеристик файлів ОС у дисковій файловій системі і множини реакцій на деякі команди між двома операційними системами, які встановлені, з одного боку, на комп'ютері контролера, а з другого – на віддаленому сервері.

Множина $\boldsymbol{F}$ характеристик кожного з файлів ОС у дисковій системі складається з елементів $\{ f_1, \ldots, f_6\}$, яким відповідають наступні значення:

f_1 – ім'я файлу,

f_2 – місце файлу у дереві каталогів,

f_3 – розмір файлу у байтах, (2.1)

f_4 – час останнього корегування файлу,

f_5 – права доступу до файлу,

f_6 – повний зміст файлу.

Позначимо множину множин характеристик файлів, які підлягають порівнянню, на сервері контролера $\boldsymbol{F}_k = \{ F_{k1}, \ldots, F_{kn} \}$, а на віддаленому сервері – $\boldsymbol{F}_s = \{ F_{s1}, \ldots, F_{sn} \}$. Оскільки справжність $\boldsymbol{F}_k$ не підлягає сумніву, то перша умова визначення справжності серверної ОС виглядатиме так:

$$F_k \Leftrightarrow F_s . \tag{2.2}$$

Другою умовою для визначення справжності серверної ОС є коректність результатів перевірки реакцій сервера на команди контролера. Для кожної з команд множина характеристик, по яким оцінюватиметься коректність реакції сервера, буде різна. Повна множина $\boldsymbol{R}$ цих характеристик по всім командам контролю даної ОС складається з елементів { r_1, ..., r_8}, яким відповідають наступні значення:

r_1 – *IP*-адреса та *TCP*-порт з'єднання з боку клієнта (команда *netstat*),

r_2 – *IP*-адреса та *TCP*-порт з'єднання з боку сервера (команда *netstat*),

r_3 – значення часу встановлення з'єднання з точністю до хвилини (команда *ps aux*),

r_4 – ідентифікаційні дані користувача (команди *top*, *ps aux*),

r_5 – назва команди користувача (команди *top*, *ps aux*),

r_6 – момент початку виконання команди з точністю до хвилини (команда *ps aux*),

r_7 – ідентифікатори активних процесів сервера (команди *top*, *ps aux*),

r_8 – моменти початку активних процесів сервера (команди *top*, *ps aux*).

Позначимо множину значень характеристик реакцій сервера, які підлягають перевірці, на боці контролера $\boldsymbol{R}_k$ = { r_{k1} , … , r_{km} }, а на віддаленому сервері – $\boldsymbol{R}_s$ = { r_{s1} , … , r_{sm} }. Умова визначення справжності серверної ОС по цих характеристиках виглядатиме так:

$$R_k \Leftrightarrow R_s . \tag{2.3}$$

Рішення про справжність серверної ОС прийматиметься тільки у разі істинності наступного предикату:

$$\left(F_k \Leftrightarrow F_s\right) \wedge \left(R_k \Leftrightarrow R_s\right). \tag{2.4}$$

Проаналізуємо структуру системи, яку можуть створити зловмисники, чия задача полягає в тому, щоб непомітно для контролерів, порушити цілісність даних про результати волевиявлення. Для збереження абсолютної точності результатів перевірок слід встановити на якомусь комп'ютері штатну ОС, яка буде дійсно надавати вірні відповіді на всі запити контролерів. Вносити зміни в цю ОС не дозволяється, бо контроль будь-які зміни може виявити. Тому все, що залишається зловмисникам, – це встановити додаткове обладнання, яке може підключатись паралельно або послідовно (за принципом *MITM*) зі штатним сервером, як показано на рис. 2.3.

Задача обладнання, що підключено послідовно (див. рис. 2.3), полягає у розподілі звернень від користувачів на два наступні потоки:

потік, що позначений цифрою 1, відправляти звернення контролерів;

потік, що позначений цифрою 2, відправляти звернення виборців.

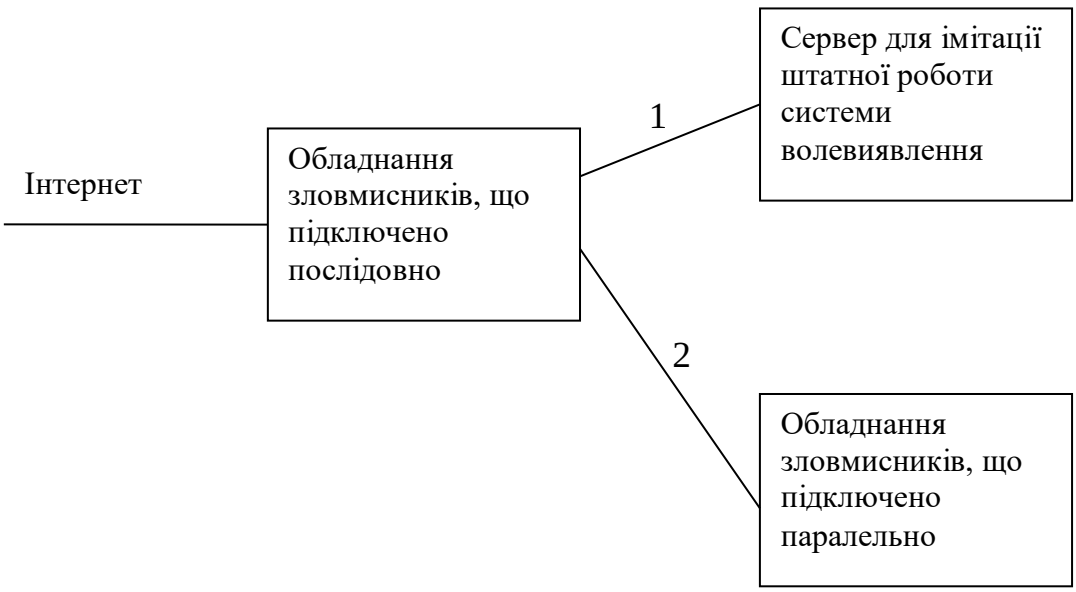

Рис. 2.3. Схема підключення обладнання для здійснення спроб підробки результатів волевиявлення

Слід зауважити, що відрізняти звернення контролерів, які завжди відправляються на *TCP* порт 22, від звернень виборців не є утрудненим. Через дану схему контролери будуть перевіряти сервер, на якому все буде точно відповідати штатному сценарію, а виборці голосуватимуть на паралельному

комп’ютері із підробленою серверною програмою, яка може зараховувати їх голоси так, як треба зловмисникам, а ще й розкривати таємницю голосів. Невдача такого задуму для зловмисників полягає в тому, що протокол *TCP* не дозволяє встановлювати з’єднання одразу з двома серверами, особливо у захищеному режимі.

Розкриття даної загрози потребує перевірки потоку даних до сервера за допомогою команди *netstat –p tcp*. При цьому можливі такі два варіанти:

1) Контролер під час уведення виборцями паролів для голосування не виявив на сервері відповідних потоків даних від виборців. Це одразу свідчить про наявність загрози.

2) Контролер виявив, що реакція сервера під час уведення виборцями паролів для голосування відрізняється від нормальної. Це означає, що зловмисники роблять спробу імітації запитів виборців до сервера. На цьому можливості зловмисників закінчуються. Відповіді сервера контролюються командою *netstat* в середовищі самого сервера через захищений протокол *SSH*.

Слід зауважити, що зловмисники можуть підробляти запити до сервера, але відповіді сервера підробити неможливо. Через те, що паролі виборців надійно захищені, а відповіді сервера на вірні і невірні запити суттєво відрізняються, бо так закладено в прикладному ПЗ, ніяких реальних шансів створення непомічених загроз у зловмисників не залишається.

Дану загрозу слід виявити в період введення паролів для голосування, що унеможливить підробку або розкриття голосів виборців.

2.3.3. Характеристики виборчих процедур з позицій ТЗІ

Технологія дистанційного волевиявлення складається з ряду процедур із різними вимогами щодо функціональних послуг захисту інформації. Характеристики цих процедур з позицій ТЗІ, починаючи від формування електронних списків виборців у розрізі кожної виборчої дільниці і до отримання остаточного результату підрахунку голосів, надано у таблиці 2.5.

Таблиця 2.5

Функціональний профіль захищеності інформації СДВ

Назва процесу	Назва дії, виконавець	Необхідність у послугах захисту	Специфікації послуг рівня Г7
Очна перевірка і реєстрація претендентів на дистанційне волевиявлення	Занесення даних претендента у базу, реєстратор	Ідентифікація та антентифікація отримувача	НП-1 НИ-1
	Введення пароля, претендент	Ідентифікація та антентифікація відправника	НА-1 НИ-1
Підготовка сервера СДВ	Завантаження і запуск ОС, адміністратор	Не впливають на рівень довіри з боку суспільства	Набір послуг захисту від НСД до підсистеми керування СДВ
	Завантаження і запуск ППЗ, адміністратор		
Очна перевірка особи голосуючого і введення пароля для голосування	Перевірка особи, реєстратор	Однонаправлений достовірний канал Одиночна ідентифікація та автентифікація	НК-1 НИ-2
	Введення пароля, голосуючий		
Підготовка до волевиявлення	Завантаження бюлетенів, адміністратор	Не впливають на рівень довіри з боку суспільства	Набір послуг захисту від НСД до підсистеми керування СДВ
Дистанційне волевиявлення	Акт волевиявлення, голосуючий	Ідентифікація і автентифікація при обміні Абсолютна конфіденційність при обміні	НВ-2 КВ-4
	Отримання довідок про кількість голосуючих, голосуючий	Не впливають на рівень довіри з боку суспільства	Набір послуг захисту від НСД до підсистеми керування СДВ
	Підрахунок голосів, сервер		
Вивід результатів волевиявлення	Вивід результату, сервер		
Масове спостереження	Перевірка справжності ОС та ПЗ, спостерігач (будь-хто)	Послуги спостереженості: 1) Самотестування у реальному часі (за запитом спостерігача); 2) Розподіл обов'язків на підставі привілеїв; 3) Цілісність КЗЗ з функціями диспетчера доступу	НТ-3 НО-3 НЦ-3
	Перевірка справжності сервера, спостерігач		
	Перевірка дій адміністратора, спостерігач		

Коди функціональних послуг захисту, що наведені в таблиці 2.5, згідно специфікаціям НД ТЗІ [48] мають наступні значення:

НП – Послуга автентифікації отримувача.

НИ – Послуга ідентифікації і автентифікації користувача.

НА – Послуга автентифікації відправника.

НК – Послуга достовірний канал.

НВ – Послуга ідентифікації і автентифікації при обміні.

КВ – Послуга конфіденційності при обміні дозволяє забезпечити захист об'єктів від несанкціонованого ознайомлення з інформацією під час їх експорту/імпорту через незахищене середовище.

НТ – Послуга само тестування КЗЗ.

НО – Послуга щодо розподілу обов’язків користувачів.

НЦ – Послуга щодо забезпечення цілісності КЗЗ.

ДС – Послуга стійкість до відмов гарантує доступність об'єктів після відмови компонента КС. Повинні бути чітко вказані рівні відмов, при перевищенні яких відмови призводять до зниження характеристик обслуговування або недоступності послуги. КЗЗ повинен бути спроможний повідомити адміністратора про відмову будь-якого захищеного компонента.

Під послугою масового спостереження мається на увазі контроль, при якому не накладаються обмеження на кількість контролерів. У нашому випадку число виборців на дільниці не може перевищити 2,5 тисячі осіб, що у певній мірі обумовлює можливу кількість контролерів.

У таблиці 2.5 не показані послуги, які в даній роботі не обираються, бо ці послуги повинні вже мати місце в системі, на зразку якої, як показано у підрозділі 2.1, розглядаються засоби дистанційного волевиявлення. Додаткових вимог щодо рівнів захисту і видів цих послуг не висувається, бо фактично нова система буде являти собою частину старої і список виборців, що голосують дистанційно буде частиною загального списку виборців. На результати голосування будуть однаково впливати помилки в кожному з цих списків. Найбільш імовірно, що обидва списки у періоди між виборами будуть

зберігатись в одному й тому ж місці, яке обов’язково буде забезпечене засобами захисту інформації.

Тільки в період виборів з'являється інформація про волевиявлення. Саме ця інформація потребує додаткових засобів захисту. До цих засобів за старою технологією можна віднести прозорі урни, камери відео-спостереження і неупереджених спостерігачів.

У дистанційній технології волевиявлення замість прозорих урн ми пропонуємо абсолютно відкрите середовище для підрахунку голосів, а замість спостерігачів, – контрольованість за допомогою засобів масового контролю.

2.3.4. Захист даних для дистанційного волевиявлення у періоди між виборами

Особисті дані про виборців, які прийняли рішення користуватись дистанційним волевиявленням, можуть зберігатись на дисках довгий час. Ці дані у значній мірі дублюють дані Державного реєстру виборців, але мають деякі додаткові атрибути і іншу структуру представлення. Цими додатковими атрибутами є ідентифікатор виборця, за який ми обрали номер паспорта із його серією і пароль виборця, який зберігається у зашифрованому вигляді. Вибір методу шифрування паролів повинен забезпечувати неможливість розкриття кожного з паролів за період виборчої кампанії (2-3 місяці), що робить недоцільним спробу такого розкриття. Особливість структури представлення цих даних полягає в тому, що для кожної виборчої дільниці створюється окремий файл бази даних, у якому слід передбачити можливість виконання реєстратором таких дій:

1) Реєстрація нового виборця.
2) Перереєстрація із зміною паролю.
3) Видалення даних через відмову виборця.
4) Видалення даних через зміну стану виборця.

Кожна дія реєстратора повинна виконуватись на підставі письмового документу (заяви виборця або офіційної довідки) і реєструватись у журналі. Паролі виборці повинні вводити самостійно і зберігати їх у таємниці.

2.3.5. Захист даних для дистанційного волевиявлення у передвиборчий період

У цей період дії, що пов’язані з реєстрацією виборців, припиняються до закінчення виборів. На серверах виборчих дільниць встановлюється операційна система і виконується наступна послідовність дій:

1) Створення користувача із правами громадського контролера, що дозволяє починати масовий контроль за роботою сервера.

2) Створення спеціальної директорії і користувача із правами роботи виключно у межах цієї директорії. В подальшому цей користувач буде єдиним виконавцем дій щодо адміністрування сервера.

3) Встановлення у цю спеціальну директорію пакетів ППЗ.

4) Видалення користувача з повними правами. З цього моменту всі дії щодо створення або корегування файлів поза межами даної директорії стають неможливими.

5) Заповнення спеціальної директорії файлами прикладного ПЗ, включаючи файл з базою даних виборців.

6) Запуск програми управління роботою сервера. З цього моменту дана програма буде автоматично вмикати і вимикати режими роботи сервера точно за розкладом. Ця програма буде єдиною діючою на сервері програмою до завершення виборчого процесу.

Спочатку програма сервера завантажує в оперативну пам’ять з файлу цілком базу даних про виборців і встановлює режим введення паролів для голосування. Ніяких звернень до файлів програма більше не виконуватиме. З цього моменту оперативна пам'ять програми стає саме тим абсолютно захищеним середовищем СДВ, у якому буде зберігатись і оброблятись вся конфіденційна інформація. Доступ до цього середовища можливий виключно через внутрішній інтерфейс самої програми для виборців і спеціальних користувачів для виконання тільки дозволених дій. Ніякого іншого доступу до цього середовища не існує, тому паролі для голосування, які зберігатимуться там, шифрувати не потрібно.

Введення виборцями паролів для голосування повинно відбуватись у спеціально обладнаному приміщенні (після очної перевірки їх осіб) протягом чітко визначеного періоду (приблизно 10-30 днів). Після завершення цього періоду програма формує файл з ознаками виборців, які ввели пароль і отримали право на дистанційне голосування. Ці ознаки потрібні для формування списку на заборону видачі паперових бюлетенів. Між періодом введення паролів для голосування і періодом голосування необхідна пауза тривалістю близько однієї доби для корегування списку виборців з врахуванням того, що частина виборців голосуватиме дистанційно. В період цієї паузи треба також занести на сервер файл з електронними бюлетенями для голосування, бо зміни можуть вноситись в бюлетені майже до початку процедури голосування.

Часову діаграму процесу обслуговування сервера виборчої дільниці показано на рис. 2.4.

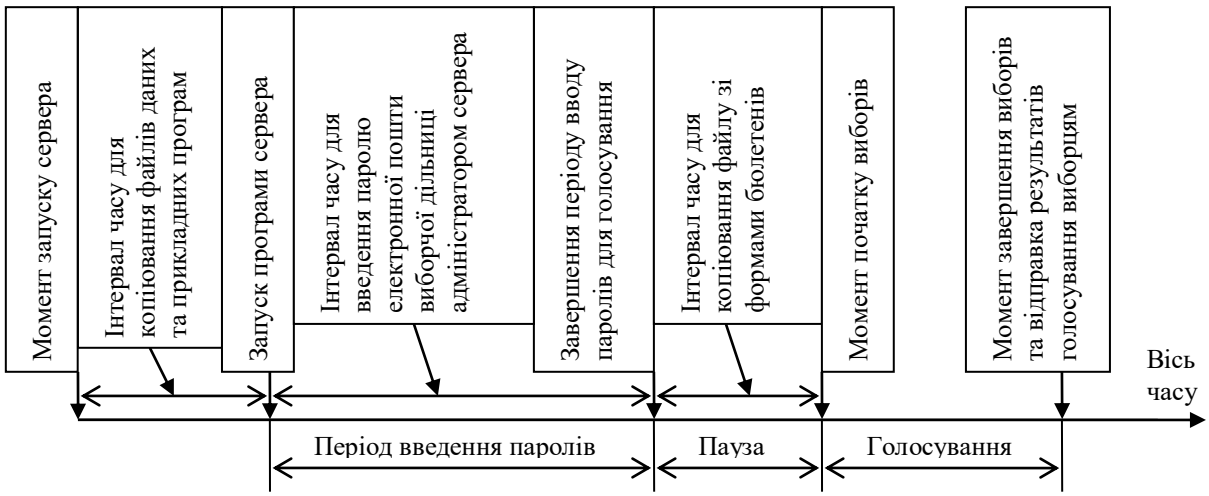

Рис. 2.4. Часова діаграма функціонування сервера виборчої дільниці у системі дистанційного голосування

2.3.6. Захист даних дистанційного волевиявлення в період виборів

Концептуальна модель захисту інформації в СДВ у період виборів, яка пропонується в даному розділі, представлена на рис. 2.5.

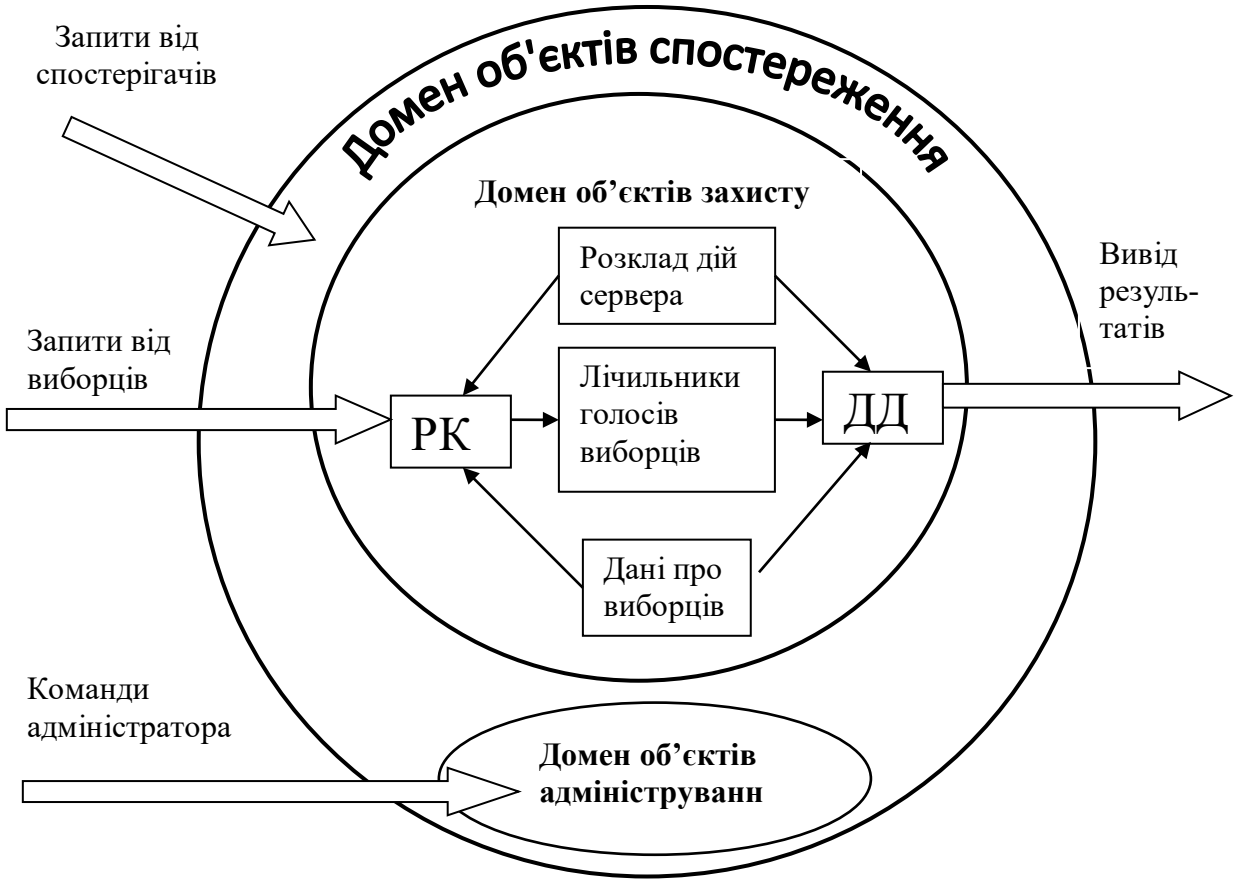

Рис.2.5. Концептуальна модель захисту інформації в СДВ

На рисунку 2.5 прийнято такі скорочені позначення функціональних блоків:

РК – блок розшифровки та контролю запитів виборців;

ДД – блок дозволу доступу до довідок та результатів волевиявлення.

Домен об'єктів адміністрування включає всі файли в директорії адміністратора. Домен об'єктів спостереження включає всі файли сервера, а також результати дії команд контролю за роботою сервера. Домен об'єктів захисту включає всі дані, що знаходяться в межах оперативної пам'яті, яка виділена для процесу виконання прикладної програми.

Згідно даної моделі функціональність СДВ має забезпечувати можливість: 1) логічної ізоляції обчислювальних процесів прикладної програми; 2) імпорту у реальному часі даних, що є об'єктами захисту в обчислювальні процеси прикладної програми; 3) створення умов для використання шифру Вернама для захисту даних, що імпортуються в домен

об’єктів захисту; 4) контроль за роботою сервера з боку необмеженого кола будь-яких користувачів Інтернету.

Поверхня внутрішнього шару ізолює множину активних обчислювальних процесів, що є об’єктами захисту, від інших обчислювальних процесів. Тобто, простір внутрішнього шару – це операційне середовище прикладної програми, яка підтримує виконання запрограмованих процедур волевиявлення. Зовнішній прошарок – це середовище ПЗ сервера, яке є відкритим для спостереження з боку широкого колу будь-яких осіб, які мають доступ до Інтернету. Особисті дані голосуючих у зашифрованому вигляді доставляються в оперативну пам’ять прикладної програми через контрольовану точку доступу. Використовуються досконало стійкі шифри для обміну даними та практично стійкі шифри для взаємної автентифікації сторін. Розклад дій сервера, закладений в прикладній програмі забезпечує визначення моментів включення та відключення дозволу на обробку запитів від виборців і дозволу на вивід довідок та результатів волевиявлення. Критичні дані перебувають виключно в оперативній пам’яті діючої програми. Вивід цих даних у вигляді результату підрахунків стає можливим тільки після закінчення процесу голосування.

Функція масового спостереження забезпечує неможливість реалізації усіх без винятку дій, які здатні вплинути на підрахунок голосів сервером, а саме:

1) Підробка або заміна штатної операційної системи.
2) Підробка або заміна пакетів програмного забезпечення.
3) Підробка або заміна прикладного програмного забезпечення.
4) Заміна сервера в цілому.
5) Будь-яке нештатне проникнення до оперативної пам'яті програми сервера.

Операційна система сервера виборчої дільниці (після певних процедур налаштування) повинна дозволяти виконувати користувачам ті, і тільки ті дії, що є елементами множини Q, де Q являє собою об’єднання множин дій голосуючих виборців, адміністратора сервера та контролерів, які складають повну групу можливих дій користувачів.

$$Q = V \cup A \cup K, \tag{2.5}$$

де V – множина дій голосуючих виборців (ця множина у разі потреби може доповнюватись діями осіб для виконання спеціалізованих наперед відомих дій, що повинні бути відображені у заздалегідь відкритій прикладній програмі);

A – множина можливих (штатних і нештатних) дій адміністратора сервера;

K – множина дій контролюючих осіб.

Слід зауважити, що під терміном дії користувачів, які показані на рис. 2.6, ми розуміємо виключно успішно проведені транзакції звернень до сервера.

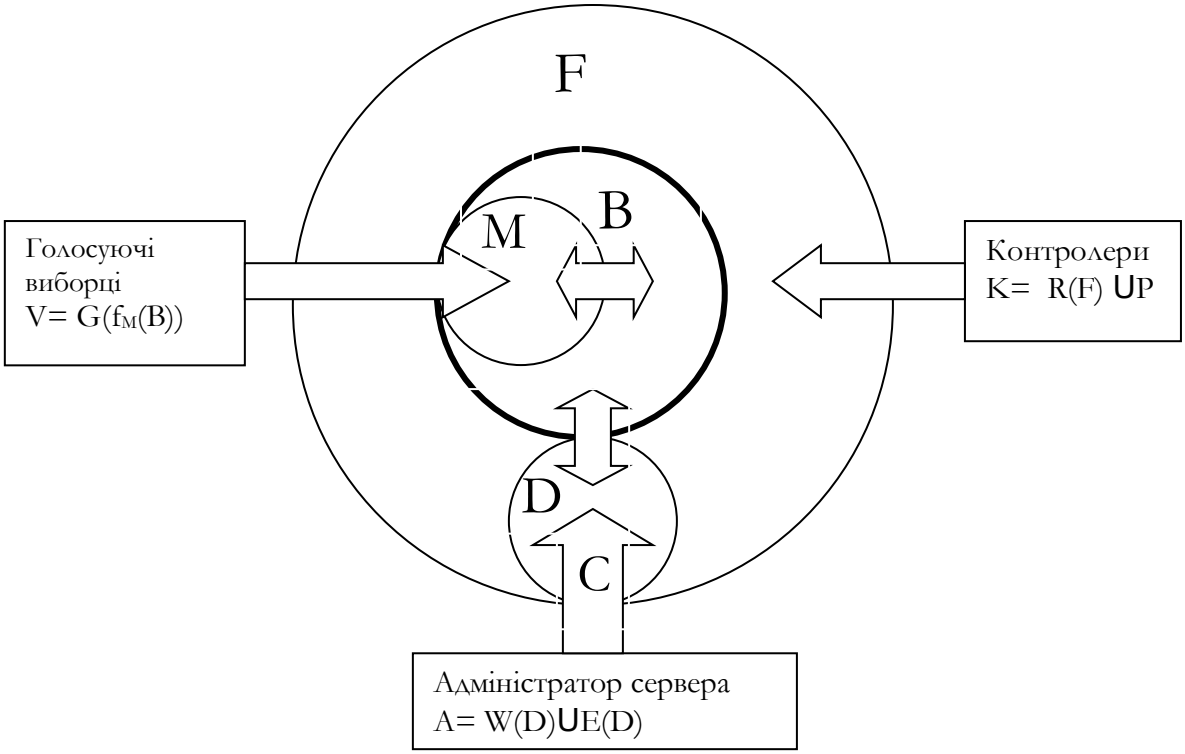

Рис.2.6. Модель взаємодії користувачів із сервером СДВ

Повна множина об’єктів, над якими можуть виконуватись дії користувачів складається з наступних множин:

F – множина всіх даних, що розміщенні у файловій системі сервера, включаючи файли з програмами готовими до виконання, а також з історією команд адміністратора;

$C-$ множина відображень команд адміністратора сервера, при чому $C \subset F$, $f : C \to A$, де f – функція відображення;

$D-$ множина файлів у тій директорії, до якої має доступ адміністратор, при чому $D \subset F$;

$B-$ множина даних в оперативній пам'яті прикладної програми сервера, при чому $B \not\subset F$;

$M-$ множина даних для моніторингу звернень виборців (ці дані використовує прикладна програма для авторизації голосуючих виборців), при чому $M \subset B$ (множина M у разі необхідності може доповнюватись діями осіб для виконання спеціалізованих наперед відомих процедур, що повинні бути відображені у заздалегідь відкритій прикладній програмі).

Множини дій користувачів над переліченими об'єктами описують наступні вирази:

$$V = \{G_1(f_M(B)), ..., G_i(f_M(B)), ..., G_n(f_M(B))\}, \quad (2.6)$$

де G_i- функція, яка відповідає і-тому варіанту запиту виборця до сервера, $i = \overline{1, n}$;

$n-$ кількість варіантів запитів виборця до сервера (наприклад: голосування, отримання довідки про хід голосування, тощо);

f_M – функція моніторингу звернень голосуючих виборців до сервера,

$$A = W(D) \cup E(D), \quad (2.7)$$

де $W-$ функція, яка відповідає множині дій адміністратора (команді запису) для приєднання файлів до множини D;

E – функція, яка відповідає діям адміністратора (команді) щодо запуску на виконання файлів (програм) з множини D;

$$K = R(F) \cup P, \quad (2.8)$$

де R – функція, яка відповідає множині дій щодо доступу контролерів для ознайомлення з об’єктами множини F, при чому $C \subset F$, $D \subset F$;

P – множина дій контролера (команд) щодо перевірки статусу процесів на сервері та отримання інших відомостей, які можуть свідчити про порушення політики безпеки.

Єдиний користувач, який має можливість виконання небезпечних дій на сервері, це – адміністратор сервера, бо будь-які дії виборців і контролерів не здатні утворити загрозу штатній роботі сервера. Адміністратору дозволено виконувати тільки дві дії, а саме, заносити файли в свою директорію і запускати на виконання файли з цієї директорії. При цьому, будь-яка нештатна дія адміністратора може бути зафіксована контролерами. Не існує таких дій, які можна було б приховати від контролерів.

Оперативна пам'ять серверної програми є зоною абсолютно захищеною від будь-якого проникнення з боку інших процесів, що забезпечено засобами операційної системи. Розклад дій сервера, закладений в програмі, забезпечує точність моментів включення та відключення дозволу на обробку запитів від виборців і дозволу на вивід результатів волевиявлення. Моменти дій у цьому розкладі формуються від таймера, який рахує системний час (кількість мілісекунд від нуля годин 1 січня 1980 року), що не залежить від комп’ютерного календаря і годинника, які можуть бути не точними.

Захист даних у процесі підрахунку голосів (див. об’єкт 5 у табл. 2.3) від будь-якого втручання забезпечується тим, що ці дані перебувають весь час виключно в оперативній пам’яті діючої програми, куди під час виконання цієї програми не існує доступу. Вивід цих даних у вигляді результату підрахунків стає можливим тільки після закінчення процесу голосування, що має бути передбачено у штатній програмі роботи сервера.

Для створення захищеного каналу обміну даними через Інтернет між виборцем та сервером (щоб протидіяти загрозам 1 та 4, які описані у табл. 2.2), як обґрунтовано далі у розділі 3, слід використати досконало стійкий шифр

Вернама у сукупності з алгоритмом Диффі-Геллмана і генератором дійсно випадкових чисел.

Ті умови, які створені для виборців у захищеній системі дистанційного волевиявлення, дозволяють гарантувати наступне:

1) Програма для голосування електронними бюлетенями, яку виборець отримає у вигляді відповіді на свій запит до сервера виборчої дільниці, не може бути підроблена. За наявності будь-яких сумнівів її можна перевірити засобами браузера, порівнюючи з еталоном.

2) У каналі зв'язку між виборцем і сервером конфіденційна інформація виборця, завдяки криптографічному захисту, не може бути розкрита або підроблена.

3) На сервері виборчої дільниці неможливо підробити або розкрити волевиявлення виборця.

Оскільки виборець може сам обирати місце для голосування, то єдине, що може знадобитись з метою захисту його права на свободу волевиявлення, – це усунення можливості будь-якого впливу на його власне рішення з боку інших осіб. Відомо чимало випадків впливу на виборців підкупом або залякуванням. За умов дистанційного голосування можна утворювати імітацію процесу голосування для уведення в оману зловмисника, що безпосередньо на візуальному рівні контролює дії виборця. Ця імітація не впливатиме на підрахунок голосів, а діалог з виборцем слід побудувати так, щоб ніяким чином не можна було відрізнити імітацію від справжнього голосування. Оскільки пароль для голосування нікому не відомий, крім самого виборця, то тільки він сам зможе відрізнити справжнє голосування від імітації.

2.4. Метод забезпечення гарантованої контрольованості процесів обробки результатів волевиявлення

Вважаючи те, що базовим принципом побудови даної системи захисту інформації є повна недовіра до всіх учасників виборчого процесу, то всі

відповідальні процедури виборчого процесу підлягають масовому контролю з боку виборчої спільноти в цілому. Головна вимога до процесу підрахунку голосів на всіх рівнях ієрархії виборчої системи полягає у його повній відкритості для кожного виборця з можливістю простої перевірки. Процедура підрахунку повинна бути представлена у такому вигляді, щоб не залишалось жодного завуальованого етапу. Такий підрахунок можна буде реалізувати на основі комп'ютерних даних, що будуть отримані з серверів виборчих дільниць. Значимість результатів буде зростати разом з кількістю виборців, які приймуть участь у дистанційному голосуванні. Оскільки доступ до мережі Інтернет є у всіх населених пунктах України, то дистанційним методом можна буде скористатись і безпосередньо у пунктах голосування кожної виборчої дільниці за наявності там комп'ютера з доступом до мережі. Для того, щоб результати електронного підрахунку відображали дійсну картину волевиявлення виборців, необхідно мати значний відсоток учасників дистанційного голосування. Значення похибки при цьому можна оцінити за допомогою методів математичної статистики [51] з наступного виразу

$$E = z\sqrt{\frac{(1-p)p}{n}}\sqrt{\frac{N-n}{N-1}}, \tag{2.9}$$

де E – відсоток статистичної похибки;

z – коефіцієнт, що залежить від відсотку довіри;

p – імовірність події, яку ми досліджуємо;

n – розмір вибірки;

N – загальний обсяг сукупності, з якого береться вибірка.

Знаючи, що загальна кількість виборців, які прийняли участь у виборах в останні роки в Україні дорівнює близько 18 млн. осіб, а кількість округів дорівнює 225, то можемо за допомогою виразу (2.9) розрахувати процент статистичної похибки в межах одномандатного округу, де кількість виборців приблизно дорівнює 18000000/225=80000. Результати розрахунків для p = 50,

що відповідає найгіршому значенню оцінки, і z = 1,96, що відповідає 95% довіри результату, та z = 2,58, що відповідає 99% довіри, наведені у табл.2.6.

Таблиця 2.6

Відсоток статистичної похибки оцінки результатів виборів від відсотка виборців, що голосують дистанційно

$\frac{n}{N}100$	2,5	5	10	20	30	40	50	60	70	80
95% довіри	2,16	1,51	1,04	0,69	0,53	0,42	0,35	0,28	0,23	0,17
99% довіри	2,85	1,95	1,37	0,91	0,70	0,56	0,46	0,37	0,30	0,23

Слід зауважити, що результати, які надані у табл. 2.6, відповідають категорії виборців, які є користувачами мережі Інтернет. Цей недолік не здається суттєвим, бо відсоток користувачів Інтернету швидко зростає. В Україні на кінець 2015 року було близько 60%, що свідчить про суттєве зростання, бо у 2013 році цей показник був тільки 41,8%.

Остаточні офіційні результати по кожному голосуванню надаються виборчими комісіями різного рівня в залежності від масштабу виборів. Тому для виявлення можливих розбіжностей між електронним і офіційним підрахунками, необхідно, щоб рівні електронних підрахунків співпадали з рівнями офіційних.

Оскільки результати голосування не є конфіденційними, то підлягають захисту тільки їх цілісність і доступність. Саме доступність результатів для кожного виборця і можливість простої перевірки є запорукою збереження цілісності. Важливо також, щоб у разі виявлення будь-якої неточності, виборці мали можливість відкрито через спеціальний сайт звернутись до відповідального органу для усунення виявленої неточності.

Зрозуміло, що для масштабних виборів (зокрема, на рівні держави), була б не простою задача перевірки підсумків для десятків тисяч дільниць, але оскільки у цих випадках ЦВК формує офіційні результати по всіх одномандатних округах, то процедура перевірки значно спрощується. На

кожному одномандатному окрузі буває від 100 до 240 виборчих дільниць, що без будь-яких ускладнень дозволяє реалізувати перевірку підсумку за допомогою широко відомих засобів типу електронних таблиць. Підсумок по округах перевіряти не обов'язково, бо він в офіційному документі ЦВК роздруковується разом з таблицею даних по всіх одномандатних округах, тому, у разі перевірених даних по кожному з округів, цей підсумок завжди буде вірним (не підлягає сумніву, що його перевіряють на комп'ютері).

Залишається пояснити технологію збору даних від серверів виборчих дільниць і формування електронної таблиці, яку міг би будь-хто завантажити на свій комп'ютер для перевірки. Для цього потрібні сервери, кількість яких буде відповідати кількості одномандатних округів у випадку виборів на рівні держави або кількості варіантів бюлетенів у випадку місцевих виборів. На кожному такому сервері необхідно встановити програму збору даних про результати підрахунку голосів від серверів виборчих дільниць по конкретному варіанту електронного бюлетеня. На цьому сервері повинна бути таблиця адрес усіх тих серверів виборчих дільниць, де відбувалося голосування по даному варіанту електронного бюлетеня. Цей сервер, який будемо називати сервером формування загального результату голосування, повинен послідовно опитати усі сервери виборчих дільниць, де відбувалося голосування по конкретному варіанту електронного бюлетеня, і занести результати, що отримані від дільничних серверів, у відповідні рядки електронної таблиці формування загального результату голосування. Після завершення процедури опитування усіх серверів виборчих дільниць (згідно з таблицею їхніх адрес) у підсумковому рядку і буде сформовано загальний результат голосування по конкретному бюлетеню. Схему технологічного процесу формування загального результату голосування зображено на рис.2.6.

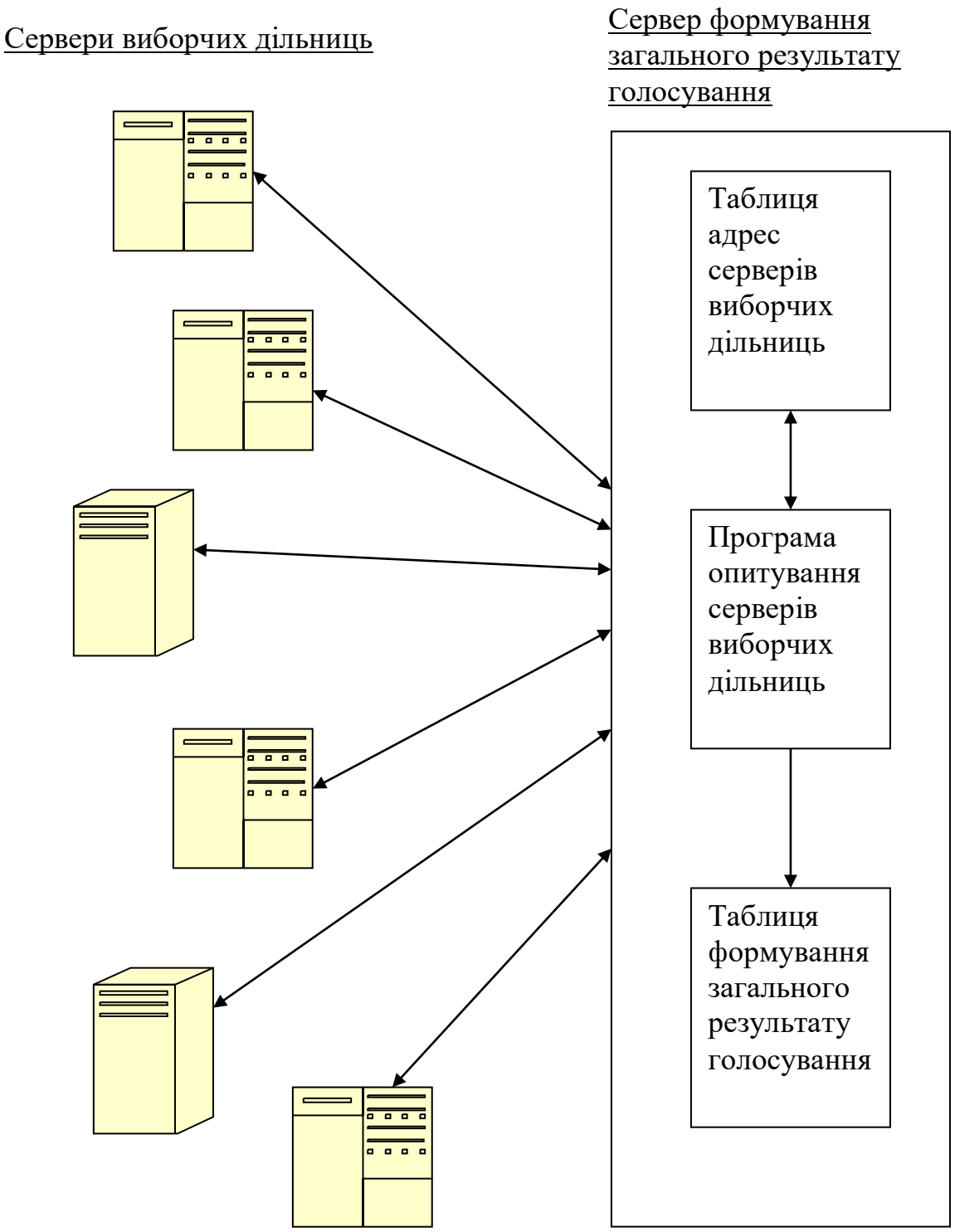

Рис.2.6. Схема технологічного процесу формування загального результату голосування

Розглянемо більш детально принципи побудови програми опитування серверів виборчих дільниць. Відомо, що найбільш достовірні результати підрахунку голосів сервером виборчої дільниці знаходяться в оперативній

пам’яті діючої програми *SVD.js*, яку докладно описано в розділі 4. Доступ до цих результатів можна отримати виключно через захищений інтерфейс, яким користуються усі клієнти-виборці. Для того, щоб забезпечити доступ до сервера виборчої дільниці через цей самий захищений інтерфейс від сервера формування загального результату голосування, необхідно в програмі *SVD.js* створити ще одного особливого користувача по аналогії, як це було зроблено для адміністратора сервера виборчої дільниці (див. підрозділ 4.4).

Обмін даними між сервером формування загального результату голосування і сервером виборчої дільниці представлено на рис.2.7.

Спеціальному користувачу, роль якого буде виконувати сервер формування загального результату голосування, можна надати право доступу без пароля, оскільки в період, коли буде відбуватись цей доступ, голосування вже завершиться і ніяких впливів на результат підрахунку голосів утворити буде неможливо.

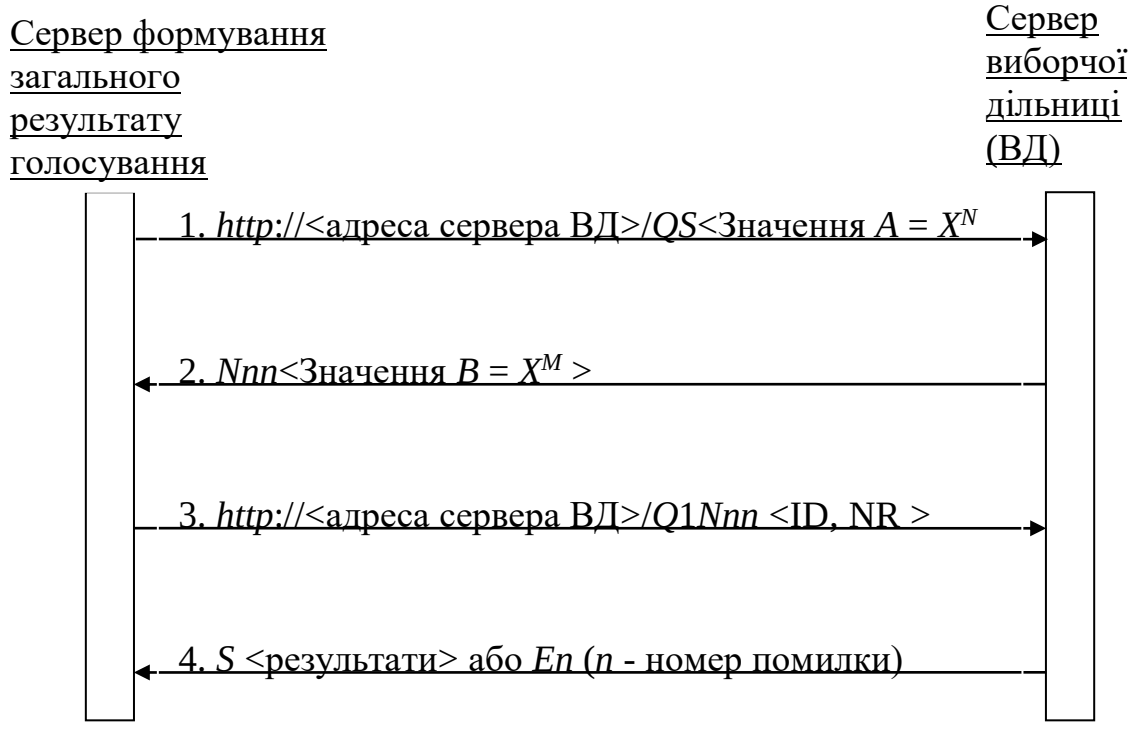

Рис.2.7. Протокол обміну даними між сервером формування загального результату голосування і сервером виборчої дільниці

У разі виникнення будь-якої аварійної ситуації під час спроби отримання результатів голосування, сервер буде повторювати запити, починаючи від першої дії за протоколом, що представлений на рисунку 2.7. До завершення опитування усіх серверів виборчих дільниць, адреси яких надані у відповідній таблиці на сервері формування загального результату голосування (див. рис. 2.6), дозвіл на коригування даних у таблиці формування загального результату голосування повинен бути забороненим.

Для побудови програми опитування серверів виборчих дільниць на сервері формування загального результату голосування можна скористатись тими самими програмними модулями, які описані в розділі 4 на мові *JavaScript*. Єдиний модуль, який необхідний для цієї програми і якого немає серед модулів, описаних в розділі 4, це модуль, що формує запити від сервера формування загального результату голосування до серверів виборчих дільниць. Запити виконуються в циклі по всім адресам, що надані в таблиці адрес серверів виборчих дільниці, показаній на рис. 2.6. Фрагмент програми, що реалізує ці запити показаний на рис. 2.8.

```
var http = require('http');
var options =
{
 host: ADRESA, // Адреса серверу виборчої дільниці
 port: PORT,  // Номер порту
 method: "GET",
 path: TR   // Рядок символів запиту (/QS . . .)
};
callback = function(response)
{
 RT= '';  // Рядок для занесення відповіді від серверу виборчої дільниці
 response.on('data', function (chunk) {RT += chunk;});
 response.on('end', OBROBKA() ); // Звернення до програми обробки відповіді
}
http.request(options, callback).end(); // Завершення процедури обміну даними
```

Рис. 2.8. Фрагмент програми на мові *JavaScript*, що реалізує запити від сервера формування загального результату голосування до серверів виборчих дільниць

Програму опитування серверів виборчих дільниць треба запускати тільки після завершення процесу голосування, бо до того моменту її запити, хоч і не будуть обслуговуватись, але будуть утворювати зайве навантаження на сервер,

що негативно вплине на якість обслуговування виборців. Файл з електронною таблицею формування загального результату голосування краще побудувати за стандартом *dBase*, який сприймається усіма сучасними програмами для роботи з електронними таблицями. Форму таблиці для перевірки загального результату голосування представлено у вигляді таблиці 2.7.

Таблиця 2.7

Форма таблиці для перевірки загального результату голосування по одному бюлетеню

Дільниця	№1	№2	№3	№4	№5	№6	№7	№8
80345	5	103	734	56	67	2	1	0
80346	3	547	468	68	45	3	0	2
…….								
80987	4	657	968	18	15	5	1	1
Всього	45	65047	92435	4537	2343	124	12	5

Незважаючи на те, що кількість рядків у таблиці найчастіше буде від 100 до 200 рядків, процедура перевірки у кожного виборця відніме не більше однієї хвилини, бо виборцю треба буде тільки відшукати рядок з номером своєї виборчої дільниці і порівняти його дані з тими, що він отримав від сервера своєї виборчої дільниці. Дані повинні співпадати. Крім цього, можна засобами своєї електронної таблиці зробити ще один контрольний підсумок і порівняти його із наданим. Після цього треба зберегти таблицю у своєму комп’ютері для порівняння результатів з офіційними. Для повної гарантії точності загального підрахунку достатньо, щоб хоч по одному виборцю з кожної дільниці виконали таку перевірку. Описана процедура перевірки на стільки прозора і проста, що не вимагає додаткових роз’яснень з приводу забезпечення за її допомогою абсолютної точності загального підрахунку голосів. Через високу надійність розглянутого методу контролю, сервер формування загального результату голосування не потребує використання якихось спеціальних засобів захисту, крім традиційних.

В момент завершення періоду голосування в оперативній пам'яті сервера буде сформований остаточний результат підрахунку голосів для даної виборчої дільниці. Цей результат не є конфіденційним, тому необхідно захищати тільки

його цілісність і доступність. Для оприлюднення даного результату можна автоматично розіслати його кожному виборцю за допомогою електронної пошти, крім того є можливість у кожного виборця отримати цей самий результат через захищений канал безпосередньо від сервера. Для формування загального підсумку результатів голосування потрібні додаткові сервери, кількість яких відповідає кількості потрібних результатів, бо для різного масштабу виборів підсумки формуються по різним групам виборчих дільниць. Принцип функціонування цих серверів полягає в тому, що кожен з цих серверів повинен опитати певну кількість серверів виборчих дільниць, де відбувалось голосування з визначеним типом бюлетеня. Від усіх опитаних серверів потрібно обов'язково отримати результати підрахунку голосів по даному типу бюлетеня. Отримані результати формуються у вигляді електронної таблиці, рядки якої відповідають виборчим дільницям, а у підсумковому рядку буде загальний результат голосування по конкретному бюлетеню.

Захист даних, що обробляються в цьому процесі, полягає в наданні можливості масового контролю сформованої таблиці. Достатньо, щоб хоч по одному виборцю з кожної дільниці перевірили точність результатів у рядку по своїй дільниці, а також, щоб хтось із них, перевірив на своєму комп'ютері загальну кількість рядків і підсумок, для гарантування абсолютної точності формування загального результату.

Висновки до другого розділу

2.1. Дистанційні технології волевиявлення з використанням Інтернету, у порівнянні з традиційними, здатні покращити як конфіденційність, так і цілісність інформації за рахунок використання криптографічних алгоритмів. Крім того, стає доступною участь у місцевих виборах, навіть під час перебування за кордоном. Але існує суттєвий недолік систем дистанційного волевиявлення, що полягає в неможливості спостерігати з боку спільноти за всіма процесами голосування через закритість комп'ютерного середовища, де

вони відбуваються. Цей недолік породжує не тільки недовіру, але й протидію, впровадженню СДВ.

2.2. З метою усунення цього суттєвого недоліку СДВ, у даній роботі пропонується забезпечити повну відкритість і контрольованість середовища, у якому відбуваються усі ті процеси, які можуть викликати сумніви с точки зору порушення конфіденційності або цілісності інформації. При цьому не накладаються обмеження на кількість контролюючих осіб.

2.3. Сформульовані вимоги до операційної системи, до засобів розробки і принципів побудови прикладного програмного забезпечення, а також до технології функціонування засобів захисту інформації у системі дистанційного волевиявлення щодо забезпечення повної відкритості і контрольованості середовища. Виконання цих вимог створює умови гарантування захищеності інформації у системах дистанційного волевиявлення.

2.4. Розроблено модель СДВ, що представлена у вигляді кореспондованої сукупності концептуальної моделі захисту інформації в СДВ та моделі взаємодії користувачів із сервером СДВ, яка за умов відкритості ПЗ дозволяє забезпечити: 1) досконалий захист критичних даних як при зберіганні на сервері, так і при обміні через середовище Інтернет; 2) гарантовану конфіденційність та відсутність фальсифікацій волевиявлення за умови повної недовіри до усіх без виключень учасників процесу волевиявлення; 3) можливість застосування методів протидії незаконному впливу на виборців. Побудовано моделі процесів для усіх можливих варіантів нападу на інформацію щодо кожного періоду функціонування системи дистанційного волевиявлення і запропоновано методи протидії усім тим загрозам, які можуть пошкодити інформацію в результаті такого нападу.

РОЗДІЛ 3
МЕТОДИ ЗАБЕЗПЕЧЕННЯ ГАРАНТОВАНОЇ КОНФІДЕНЦІЙНОСТІ ТА ЦІЛІСНОСТІ ІНФОРМАЦІЇ У ПРОЦЕСІ ДИСТАНЦІЙНОГО ВОЛЕВИЯВЛЕННЯ

3.1. Метод протидії загрозам, що виникають під час обслуговування сервера дистанційного волевиявлення

Важливий елемент СДВ – сервер, на якому відбувається підрахунок голосів. У загальнонаціональній СДГ ці сервери відповідають виборчим дільницям, котрих в Україні, згідно даним ЦВК, нараховується близько 30-ти тисяч, у кожній з яких може обслуговуватись до 2,5 тисяч виборців. Задачі, вирішення яких забезпечує сервер виборчої дільниці, полягають у наступному:

1) ідентифікація та автентифікація виборців з метою отримання дозволу на дистанційне голосування;

2) формування списку виборців, що прийматимуть участь у дистанційному голосуванні;

3) підтримка діалогу з виборцями під час дистанційного голосування за умов забезпечення повної конфіденційності їх волевиявлення;

4) оперативний вивід довідок про кількість учасників голосування;

5) підрахунок голосів виборців, який повинен зберігатись у таємниці до моменту закінчення голосування;

6) вивід результатів підрахунку голосів.

Забезпечення гарантованої конфіденційності та цілісності інформації у СДВ безпосередньо залежить від ефективності протидії загрозам, що виникають під час використання та обслуговування сервера. Результати аналізу загроз інформаційним ресурсам СДВ (див. розділ 1.2) свідчать, що сервер виборчої дільниці в існуючих умовах його застосування є потенційно уразливим елементом СДВ з точки зору інформаційної безпеки. В першу чергу, це стосується програмного забезпечення (ПЗ) сервера. У розділі 1.2 показано,

що під час обслуговування сервера СДВ виникають загрози, засоби протидії яким не забезпечують гарантовану конфіденційність та цілісність інформації у СДВ. Тому у даному підрозділі поставлено та вирішено завдання створення методу протидії загрозам порушення конфіденційності та цілісності, що забезпечують гарантованість захисту.

3.1.1. Постановка завдання

Розробити метод протидії загрозам порушення конфіденційності та цілісності інформації у середовищі функціонування сервера СДВ за умов повної недовіри до будь-яких осіб незалежно від їхніх прав доступу до серверу та статусу у суспільстві. Визначити умови обслуговування цього сервера, за яких забезпечується абсолютна конфіденційність та цілісність інформації у середовищі функціонування СДВ.

3.1.2. Загальні міркування щодо вирішення завдання

Існуючі методи та засоби протидії порушенням конфіденційності та цілісності середовища функціонування об’єктів інформаційної діяльності з використанням мережі Інтернет не забезпечують можливість контролю дій адміністраторів цих об’єктів з боку сторонніх осіб. У той же час в якості основного результату даного дослідження передбачається створення методів і процедур дистанційного волевиявлення, реалізація котрих на практиці гарантують об’єктивність і свободу дистанційного волевиявлення в умовах повної недовіри будь-яким адміністраторам чи власникам програмно-апаратних засобів СДВ та телекомунікаційної інфраструктури. Отже, розроблюваний метод має, перш за все, забезпечувати протидію загрозам нештатного впливу з боку будь-яких зацікавлених осіб, в першу чергу адміністраторів сервера, на процес його штатного функціонування.

Загроза нештатного впливу адміністратора сервера на процес його штатного функціонування пов’язана з наступним. Вважаючи, що адміністратор сервера, маючи безпосередній доступ до сервера і будучи прихильником якого-небудь конкретного варіанту голосування, може спробувати нелегальним чином вплинути на процес роботи сервера, з метою підробки результату

голосування на користь бажаного варіанту. Через те, що адміністратор має практично необмежені можливості виконання будь яких дій на сервері, загроза від його дій може повністю зруйнувати картину волевиявлення виборців. Без подолання цієї загрози неможливо досягти повної довіри виборців до системи дистанційного голосування. Тому необхідно не тільки подолати цю загрозу, але й обрати такий шлях її подолання, щоб кожен виборець міг самостійно переконатись у неможливості реалізації подібної загрози.

3.1.3. Загальна схема рішення

1) Здійснити цілеспрямований вибір операційної системи (ОС).

2) Створити та оприлюднити розклад дій персоналу під час проведення виборчої кампанії.

3) Розробити механізм доступу через Інтернет до програмних засобів контролю дій персоналу.

4) Розробити механізм контролю актуального складу компонентів ОС на сервері.

5) Розробити механізм контролю стану активних процесів на сервері.

6) Розробити механізм контролю складу актуальних процесів, що відбуваються на сервері.

7) Розробити механізм фіксації та нейтралізації загрози розміщення зловмисником на сервері позаштатних програмних файлів і їхньої відправки на виконання.

8) Виконати схематичне представлення метода забезпечення гарантованої конфіденційності та цілісності інформації у середовищі функціонування сервера СДВ.

3.1.4. Цілеспрямований вибір операційної системи

Можливості реалізації зазначеної вище загрози на сервері, перш за все, можуть бути закладені в операційній системі (ОС). Тому питання вибору ОС для сервера СДГ з точки зору ТЗІ є актуальним. В нашому випадку ОС повинна бути повністю відкритою, надійно захищеною від несанкціонованого доступу

(НСД) і гарантовано унеможливлювати виконання будь-яких нештатних дій під час адміністрування сервера.

Виходячи з цих міркувань, під час вибору серверної ОС мають задовольнятися наступні вимоги.

1) Специфікації обраної ОС, у т.ч. вихідні тексти програм, мають бути відкритими та доступними для ознайомлення, а логічна структура самої ОС має відповідати загальновідомим принципам, щоб забезпечити можливість її оперативної верифікації з боку будь-якої зацікавленої особи із середовища програмістів.

2) Має бути забезпечена можливість встановлення цієї ОС без будь-яких ускладнень на будь-якому комп'ютері, що підключений безпосередньо до Інтернет.

3) Засоби ОС повинні забезпечувати абсолютно досконалий захист від НСД до ресурсів сервера.

4) Засоби ОС повинні забезпечувати можливість перманентної пасивної перевірки через Інтернет з боку будь-якої зацікавленої особи усіх файлів і процесів, що задіяні на сервері, але не уможливлювати здійснення будь-яких інших дій, що не передбачені діючими правилами розмежування доступу.

Усім цим вимогам відповідає ОС *OpenBSD* у мінімальній конфігурації. Тому одною з основних умов забезпечення протидії загрозам нештатного впливу адміністратора сервера на процес його штатного функціонування є його оснащення операційною системою *OpenBSD*.

Слід зауважити, що обрана ОС *OpenBSD* цілком заслуговує на довіру, бо за 18 років її експлуатації було виявлено тільки два слабких місця [53], які були знешкоджено. Тому можна бути впевненим, що ніяка інша особа, крім адміністратора сервера не зможе виконувати ніякі небезпечні дії на сервері.

3.1.5. Створення та оприлюднення розкладу дій персоналу

Обравши абсолютно досконалу ОС з точки зору захищеності від НСД, необхідно ще довести кожному виборцю, що ця ОС на протязі всієї виборчої кампанії функціонує у штатному складі і що у ній нічого не змінено, зокрема

нічого не виключено і не доповнено. Для забезпечення абсолютної довіри виборців до серверної ОС і всіх інших програмних засобів, встановлених на сервері, необхідно оприлюднити розклад дій персоналу щодо адміністрування сервера в період підготовки і проведення виборів з точними моментами часу виконання кожної дії. Здійснення будь-яких дій з боку будь-яких осіб, у т.ч. з боку персоналу, поза цього розкладу має бути категорично заборонено. Реакції на будь-які нештатні події, навіть форс-мажорні, мають виконуватися тільки у рамках оприлюдненого розкладу. За цих умов ризики виникнення форс-мажорних обставин мають бути мінімізовані. Передбачається, що демократичне суспільство за форс-мажорних обставин скоріш погодиться із втратою засобів СДВ, ніж з порушенням розкладу дій персоналу.

3.1.6. Забезпечення доступу через Інтернет до програмних засобів контролю дій персоналу

Виборці повинні мати можливість заздалегідь ознайомитись з повним комплектом програмного забезпечення (ПЗ) сервера і отримати інструкцію щодо контролю за роботою сервера. Усе ПЗ має бути таким, щоб його на своєму комп'ютері міг встановити і випробувати кожен бажаючий. Інформацію, яку необхідно мати для такого випробування, надано у таблиці 3.1.

Таблиця 3.1

Склад програмного забезпечення сервера виборчої дільниці і інструкцій щодо його контролю

Найменування ресурсу	**Мережева адреса ресурсу**
Операційна система *OpenBSD*	*http://www.openbsd.org*
Інструкція щодо встановлення операційної системи *OpenBSD*	*http://www.lissyara.su/articles/openbsd/syntony/installation_openbsd_4.2/*
Мова прикладного програмного забезпечення *JavaScript* та інструкція щодо встановлення	*https://nodejs.org*
Файл прикладної програми сервера на мові *JavaScript*	<адреса сайту виборчої дільниці> /SVD.js
Програма для контролю процесів на сервері *PuTTY*	*http://www.putty.org*
Програма для контролю файлів на сервері *WinSCP*	*https://winscp.net/eng/docs/lang:ru*
Інструкція для контролю функціонування сервера	<адреса сайту виборчої дільниці> /INSTRUKC.doc
Розклад дій персоналу щодо адміністрування сервера в період проведення виборів	<адреса сайту виборчої дільниці>

3.1.7. Контроль актуального складу компонентів ОС на сервері

Для реалізації перевірки відсутності будь-яких модифікацій встановленої на сервері ОС *OpenBSD* кожен бажаючий має встановити на своєму комп'ютері таку ж ОС. Така можливість має бути забезпечена шляхом публікації специфікацій встановленої ОС. Після цього, знайшовши в опублікованому розкладі дій персоналу час початку перевірки і, маючи на своєму комп'ютері програму *WinSCP* для контролю файлів, порівняти файли ОС на сервері з файлами ОС на своєму комп'ютері. Слід зауважити, що для цієї перевірки зручно скористатись двома комп'ютерами, на одному з яких ОС *OpenBSD*, а на другому – ОС *Windows* із програмою *WinSCP*. Вигляд вікна програми *WinSCP* під час перевірки файлів показано на рис. 3.1.

Для того, щоб встановити зв'язок зі сервером виборчої дільниці, треба в програмі *WinSCP* вказати наступні три параметри, які повинні бути надані на сайті виборчої дільниці:

Host name – *IP*-адреса сервера виборчої дільниці (91.198.50.7 на рис. 3.1).

User name – ідентифікатор користувача (один для усіх суб'єктів, що здійснюють контроль).

Password – пароль користувача (єдиний для усіх контролерів).

З метою спрощення процедури контролю можна видалити на сервері файли, які непотрібні для даної прикладної задачі, бо чим менше файлів, тим легше буде контролювати сервер.

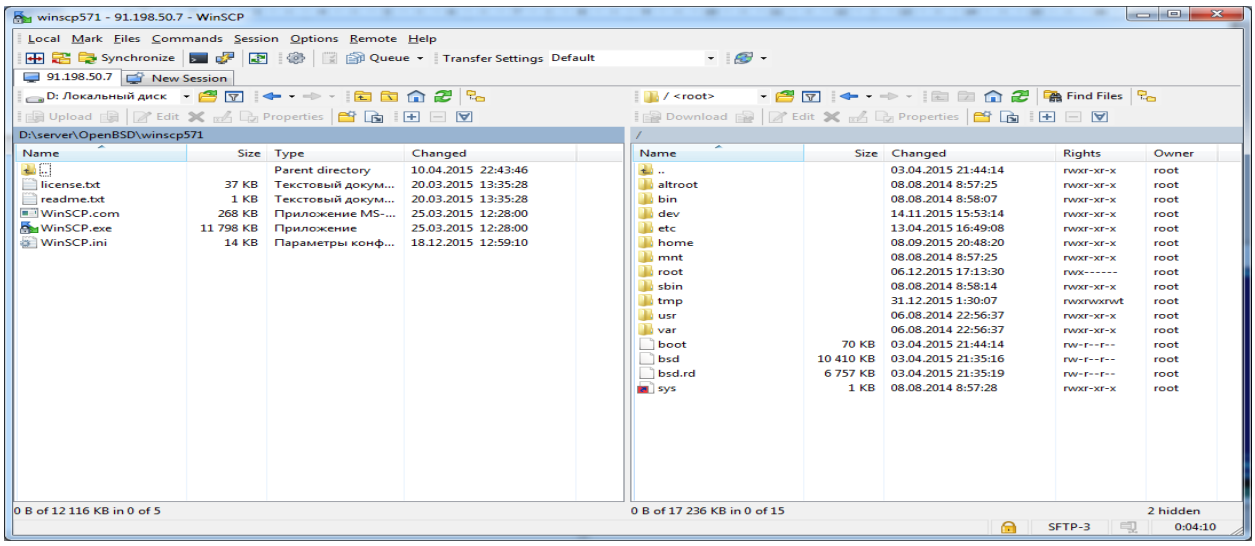

Рис. 3.1. Інтерфейс програми *WinSCP*

3.1.8. Контроль стану активних процесів на сервері

Другою важливою перевіркою є контроль стану активних процесів, які діють на сервері. Для цього можна скористатись програмою *PuTTY*, яка дозволяє отримати можливість виконання команд на сервері. Зрозуміло, що серед цих команд для користувачів-контролерів будуть дозволені тільки ті, які потрібні для виконання функцій контролю. На рис. 3.2 показано результат виконання на сервері команди *ps aux*.

3.1.9. Контроль складу актуальних процесів, що відбуваються на сервері Аналіз складу актуальних процесів, які знаходяться в активному стані в момент запуску сервера, показує, що їх кількість обмежена і дорівнює 20. Тому об'єктами контролю у даній опції перевірки є дані про ці 20 процесів. Слід упевнитись, що протягом усього періоду функціонування сервера ідентифікатори цих процесів, які наведені у стовбці *PID*, не змінюються. Моменти запуску цих процесів відображені у стовбці *STARTED*.

На рис. 3.2 показані дані про сервер, який було запущено 14 листопада 2015 року, тому у стовбці *STARTED* ми бачимо значення 14*Nov*15.

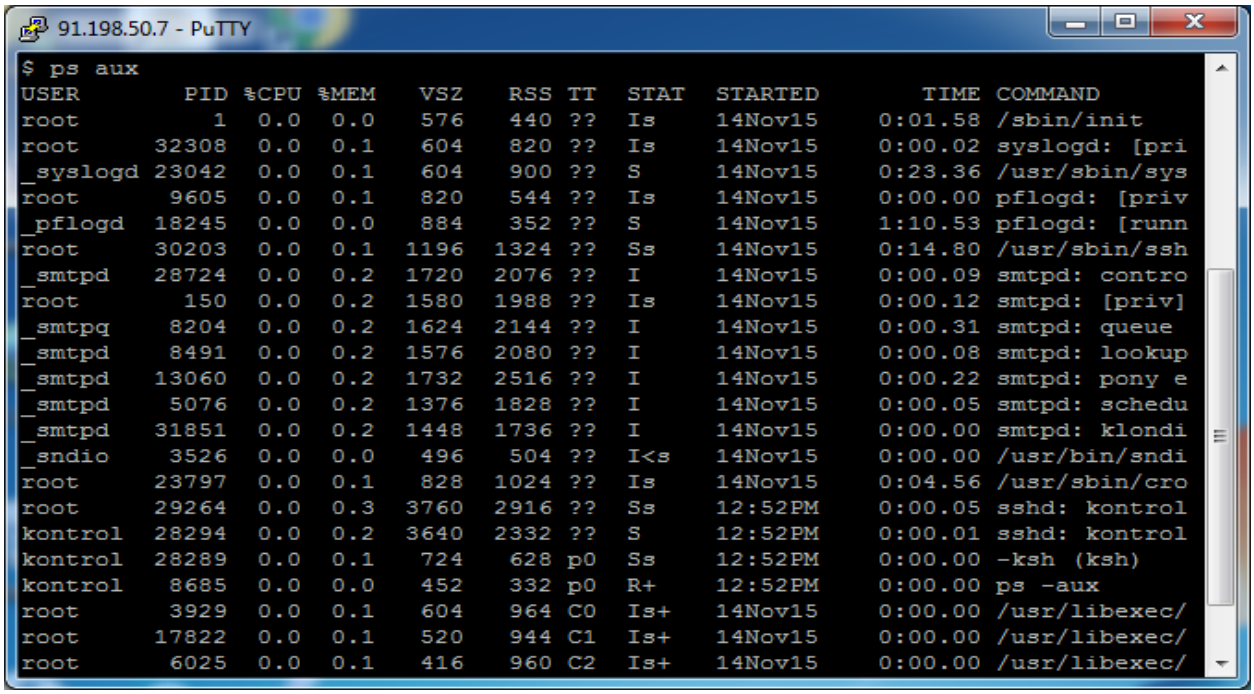

Рис. 3.2. Результат виконання команди *ps aux*, яка дозволяє перевірити стан діючих програм на сервері

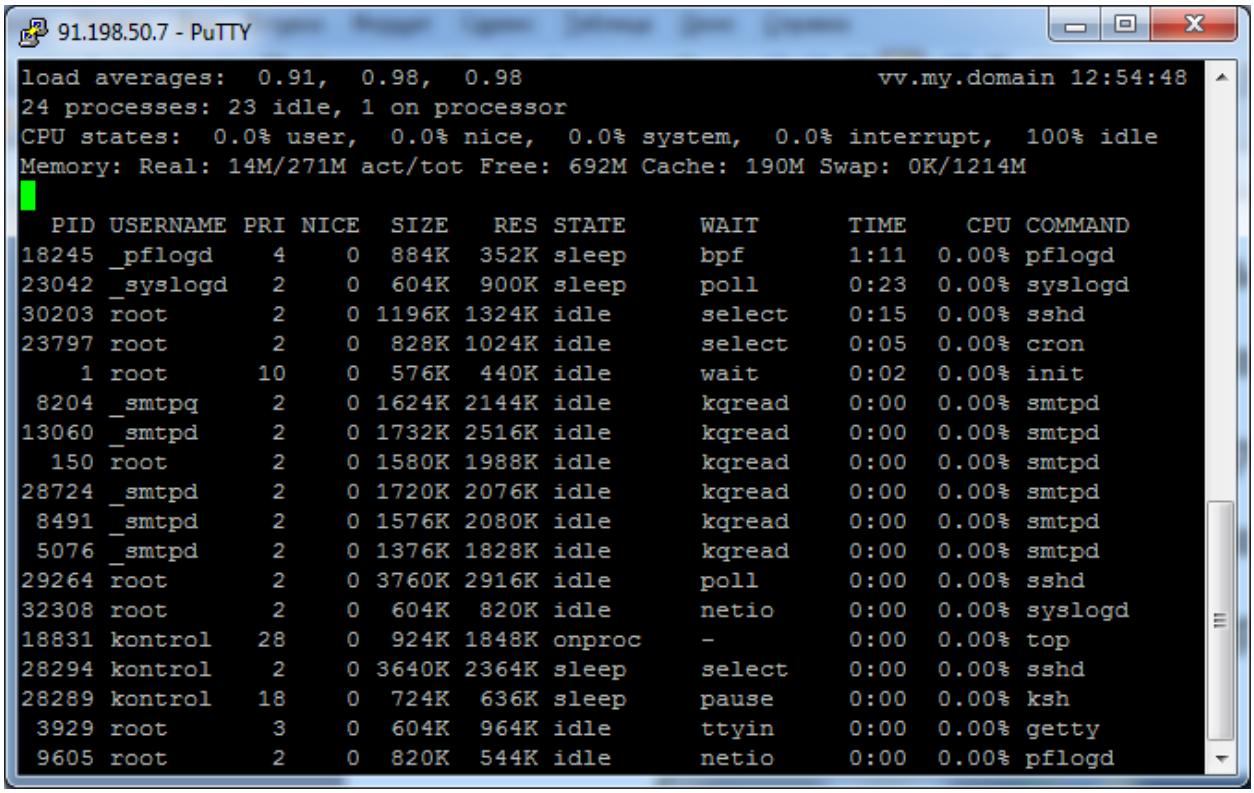

Рис. 3.3. Результат дії команди *top*

Результат дії команди *top*, яка також надає довідку про процеси, що відбуваються на сервері, показано на рис.3.3.

Команда *top* відрізняється від команди *ps aux* тим, що вона працює безперервно, поновлюючи дані на екрані кожні 5 секунд. Дія цієї команди припиняється за допомогою комбінації клавіш *Ctrl* і *C*. На рис. 3.2 і 3.3 ми бачимо процеси контролера, які мають параметр *USERNAME* (або скорочено просто *USER*) *kontrol*. Їх є декілька для кожного користувача, бо один процес підтримує зв'язок між клієнтом і сервером через захищений протокол *SSH*, другий процес відстежує всі дії клієнта, дозволяючи йому виконувати виключно дозволені дії, та процес виконання команди *ps aux* або *top*. Решта (20 процесів) є процеси операційної системи, які існують протягом усього часу роботи сервера.

3.1.10. Фіксація та нейтралізація загрози розміщення зловмисником на сервері позаштатних програмних файлів і їхньої відправки на виконання

Відомо, що для реалізації будь-якої загрози зловмиснику необхідно розмістити на сервері позаштатний програмний файл і відправити його на виконання. При цьому обов’язково на сервері з’являться нештатні процеси, як під час появи програмного файлу, так і після його запуску. Все це може бути зафіксовано спостерігачами. Крім того, позаштатний процес (так само, як і всі інші процеси) не може впливати на функціонуючі програми, а може тільки змінювати файли, що зберігаються на носіях. Оскільки протягом всього часу функціонування сервера виборчої дільниці, діє тільки одна прикладна програма обміну даними з виборцями і підрахунку голосів, яка взагалі не звертається до файлів, то подібні дії зловмисника не можуть зашкодити її роботі.

3.1.11. Схематичне представлення методу

Розроблений метод доцільно представити у вигляді логічної моделі технологічного процесу обслуговування і захисту від загроз сервера виборчої дільниці. Схематичне представлення метода забезпечення абсолютної конфіденційності та цілісності середовища функціонування сервера СДВ надано на рис. 3.4.

На рис. 3.4 представлено дві нові дії адміністратора, які вимагають пояснення. Створюючи користувача *admin* з обмеженими повноваженнями і блокуючи користувача *root* з повними повноваженнями, адміністратор сам позбавляє себе можливості створення нових користувачів. Після цього він залишає собі тільки ті можливості управління сервером, які необхідні для виконання відомих конкретних функцій. Таке рішення значно спрощує задачі контролерів, бо їм залишається перевіряти тільки директорію *home/admin.* За цих умов на сервері до кінця виборчої кампанії буде єдиний користувач *admin,* який може встановлювати і запускати програми, бо інших користувачів створювати вже неможливо. Тому слідкувати за процесами на сервері стає значно простіше. Решта користувачів з ім'ям *kontrol* принципово не можуть утворити загрозу через обмеження повноважень, тому на їх процеси можна не звертати увагу.

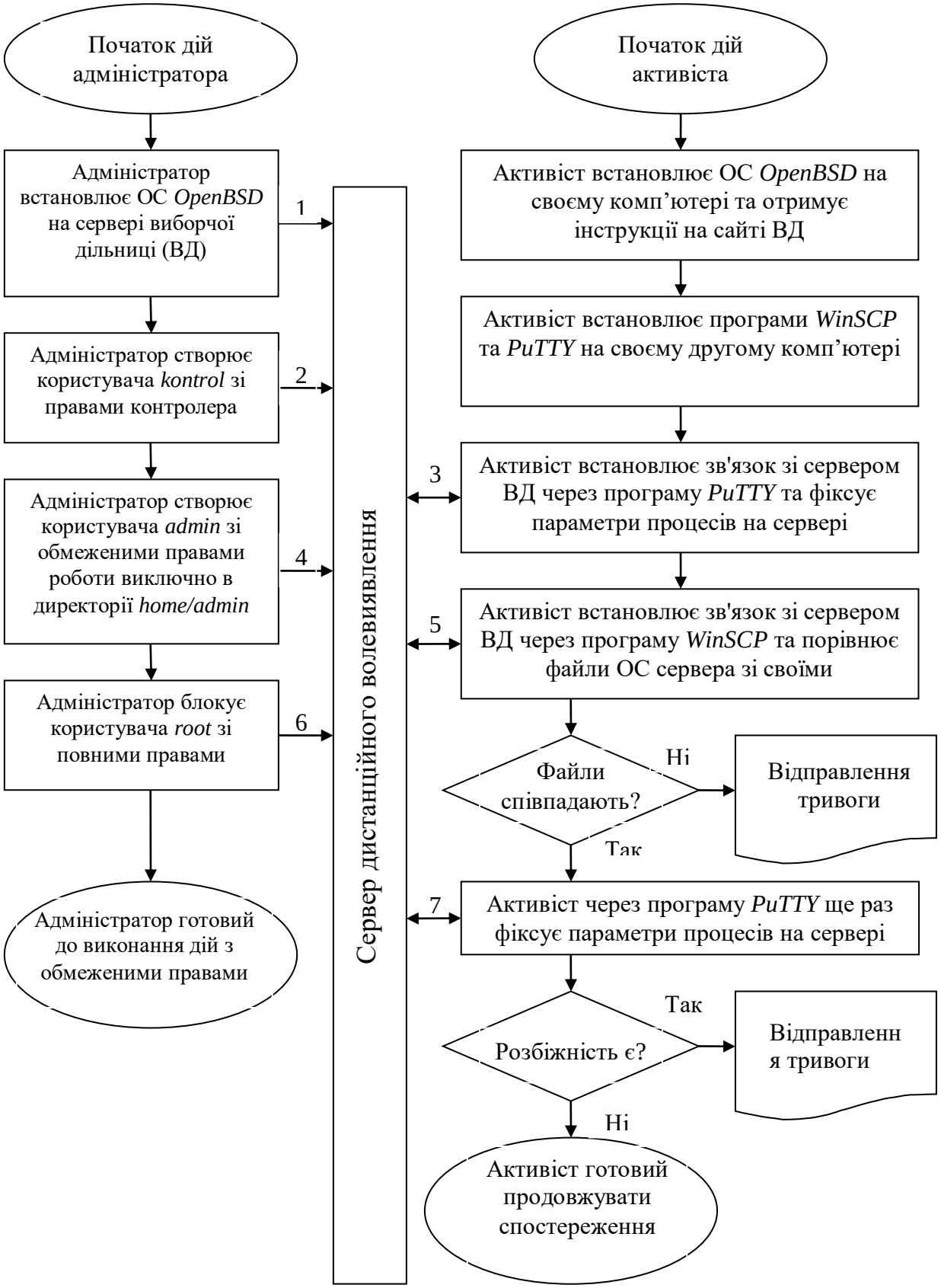

Рис. 3.4. Метод спостереження за функціонуванням СДВ

Нові процеси з параметром *USERNAME root* будуть з'являтись тільки одночасно зі новими процесами користувачів *kontrol* або *admin*, бо їх буде автоматично створювати ОС у відповідь на звернення осіб-користувачів. Сама по собі операційна система не може бути ініціатором будь-яких нових процесів.

3.2. Метод досконало захищеного обміну даними між клієнтами та сервером системи дистанційного волевиявлення

Канал обміну інформацією між клієнтом та сервером у будь-якій СДВ, як показано у розділі 1, є потенційним джерелом виникнення загроз порушення конфіденційності та (або) цілісності даних, що транспортуються через цей канал, і тому слід здійснити заходи щодо його захисту, зокрема розробити специфікації протоколу захищеного обміну даними через цей канал. У відповідальних випадках, наприклад у разі створення загальнонаціональної системи дистанційного голосування (СДГ), основна вимога до розроблюваного протоколу (див. розділ 3.1) – забезпечити абсолютно стійкий захист канальних даних. Результати розробки такого протоколу представлено у даному підрозділі.

3.2.1. Постановка завдання

Розробити протокол захищеного обміну даними через Інтернет між клієнтами та сервером СДВ, що забезпечує згідно К. Шеннону [43] абсолютно стійкий захист канальних даних від порушень їхньої конфіденційності та цілісності, і конкретизувати умови використання цього протоколу у загальнонаціональній СДГ. Абсолютна стійкість захисту канальних даних має бути забезпечена як на стадії обміну ключами шифру, так і на стадії обміну конфіденційною інформацією під час здійснення волевиявлення.

3.2.2. Загальна схема рішення

Протокол розроблено у класі симетричних криптографічних систем з використанням одного із варіантів ендоморфного шифру [54, 55]. Сеанс зв’язку

між клієнтом та сервером, що має відбуватися під час процесу волевиявлення, розбивається на дві стадії. На першій стадії здійснюється обмін даними для визначення та синхронізації ключів шифру з використанням алгоритму Диффі-Геллмана [46]. На другій стадії здійснюється безпосередній обмін конфіденційною інформацією, що генерується під час виконання акту дистанційного волевиявлення через відкритий канал Інтернет на основі методу одноразових шифрувальних блокнотів (*one-time-pad*) [43], тобто шляхом використання досконало стійкого шифру Вернама, що забезпечує за певних нижче визначених умов абсолютно стійкий захист даних під час їхнього передавання. Обидва названі методи (шифри) є широко відомими, принципи реалізації та основні характеристики котрих висвітлені у розділі 1. Вибір алгоритму Диффі-Геллмана для обміну ключовою інформацією та шифру Вернама для обміну конфіденційними даними обумовлений факторами, що висвітлені у розділі 1. Проте для коректного їхнього використання у загальнонаціональній СДГ необхідно конкретизувати діапазони оптимальних значень параметрів цих методів, визначити обмежувальні умови застосування та розробити методи і засоби забезпечення виконання цих умов.

У даному випадку розробка протоколу передбачає необхідність визначення:

1) необхідної тривалості обслуговування запиту (фактично, тривалості сеансу зв’язку між клієнтом і сервером) за різних значень якості обслуговування (що визначається, у даному випадку, ймовірною тривалістю очікування клієнтів у чергах до серверу);

2) довжини ключів шифру, що напряму пов’язано із вибором методу шифрування та визначенням кількості конфіденційних даних, що мають бути передані через канал Інтернет на протязі одного сеансу зв’язку;

3) параметрів алгоритму Диффі-Геллмана, зокрема варіанту мультиплікативної групи для реалізації цього алгоритму, за яких злом системи захисту в період проведення виборчої кампанії є практично гарантовано неможливим;

4) методу отримання дійсно (а не псевдо) випадкових бітових послідовностей, реалізація котрого є необхідною умовою коректного функціонування досконало секретної системи, що реалізована з використанням шифру Вернама, а також необхідною умовою коректної реалізації алгоритму Диффі-Геллмана.

3.2.3. Визначення оптимального значення тривалості сеансу зв'язку між клієнтом і сервером за різних значень якості обслуговування

Вихідні дані: 1) максимальна кількість виборців, що обслуговуються одним сервером на виборчій дільниці в Україні – 2500 чоловік; 2) максимальна тривалість часу голосування – 12 годин; 3) потік запитів виборців до сервера - однорідний та стаціонарний без післядії.

Метод визначення. Визначимо залежність імовірності часу очікування клієнта у черзі до ресурсів сервера від витрат часу на обслуговування одного виборця. Якщо потік запитів виборців до сервера буде однорідним та стаціонарним без післядії, то імовірність потрапляння k запитів в інтервал часу τ буде відповідати закону Пуассона [56]. Для випадку, коли максимальна кількість виборців однієї дільниці (2500 чоловік) голосуватиме дистанційно, середній інтервал часу між запитами буде ≈17с або де середнє значення інтенсивності потоку запитів λ=1/17[1/с]. СДГ має забезпечувати можливість здійснити акт волевиявлення кожному виборцю. Отже, проміжок часу, що припадає у цьому випадку на обслуговування кожного запиту сервером, повинен бути меншим за 17с. Змінюючи інтервал обслуговування від 1с до 12с, визначимо значення імовірності того, що виборцю необхідно буде очікувати у черзі до сервера від 0 до T секунд. Позначивши тривалість інтервалу обслуговування запиту виборця сервером через τ, можемо знайти значення такої імовірності як події потрапляння k запитів в інтервал часу τ, при цьому мінімальне значення T буде дорівнювати $k\times\tau$, а максимальне – $(k+1)\times\tau$. Інтенсивність обслуговування запитів буде дорівнювати $\mu=1/\tau$, а імовірність того, що сервер буде вільний $P_0=\mu/(\lambda+\mu)$. Значення імовірності обслуговування

запиту виборця без очікування P_0 в залежності від часу обслуговування запиту сервером τ надані у табл.3.1.

Таблиця 3.1

Залежність імовірності обслуговування виборця без очікування в черзі від інтервалу обслуговування запитів

τ,с	1	2	3	4	5	6	7	8	9	10	11
P_0	0.945	0.896	0.852	0.812	0.775	0.742	0.712	0.683	0.657	0.633	0.611

Значення імовірності обслуговування запиту виборця з очікуванням P_T в залежності від часу обслуговування запиту сервером τ надані у табл.3.2.

Таблиця 3.2

Залежність імовірності обслуговування виборця з очікуванням в черзі від інтервалу обслуговування запитів

τ,с	1	2	3	4	5	6	7	8	9	10	11
P_τ	0.052	0.092	0.127	0.153	0.176	0.191	0.206	0.217	0.225	0.236	0.239
$P_{2\tau}$	0.003	0.011	0.018	0.028	0.039	0.049	0.058	0.068	0.078	0.086	0.092
$P_{3\tau}$	-	0.001	0.003	0.006	0.008	0.013	0.017	0.022	0.026	0.031	0.036
$P_{4\tau}$	-	-	-	0.001	0.002	0.003	0.005	0.007	0.009	0.012	0.014
$P_{5\tau}$	-	-	-	-	-	0.001	0.002	0.002	0.003	0.004	0.005

Дані табл.3.2 свідчать: збільшення інтервалу обслуговування на величину більшу, ніж *6с* , призводить до того, що кожний п’ятий виборець за вище визначених умов повинен буде очікувати своєї черги до сервера від 7с до 14с , а кожний сімнадцятий – від 14с до 21с.

Проте не всі виборці будуть брати участь у голосуванні, тим більш, не всі будуть користуватись дистанційним способом доступу до сервера. Тому наведені дані відповідають найгіршому, з точки зору якості обслуговування, варіанту.

Наведений розрахунок дозволяє визначитись з вибором оптимального значення тривалості сеансу зв’язку між клієнтом і сервером для забезпечення

конкретного рівня якості обслуговування клієнтів сервера під час голосування з урахуванням конкретних цілей побудови та умов використання СДВ. Також цей розрахунок буде корисним для вибору необхідної кількості серверного обладнання, бо у ряді випадків можна буде декілька виборчих дільниць обслуговувати за допомогою одного сервера.

3.2.4. Вибір довжини ключа шифру

Вихідна передумова. Забезпечення необхідного значення відстані єдиності [55, стор.86] L_0, тобто сумарна довжина бітових послідовностей переданих текстів через канал Інтернет $I_\sum$ на протязі одного сеансу зв'язку має бути не більшою, ніж довжина ключової інформації $2^{|K|}$, де $|K|$ - мінімально можлива кількість бітових ключів шифру, що є необхідною для користування ендоморфним шифром Вернама у режимі забезпечення досконалої стійкості.

Основна перевага ендоморфних шифрів, у т.ч. шифру Вернама, полягає у можливості їхнього використання у режимі так званої досконало секретної системи,коли відомі із криптографії показники стійкості не застосовуються, оскільки вважається, що такі шифри за певних умов є абсолютно стійкими.

Умови забезпечення режиму досконалої стійкості. В якості основного параметра ефективності захисту при використання шифру Вернама розглядається так звана відстань єдиності [43]. Якщо вважати, що шифр – це сукупність множини відкритих текстів X, множини ключів K, множини криптограм Y, ключі шифру є рівно ймовірними, а передані тексти осмисленими, то натуральне число L_0, для якого очікувана кількість фальшивих ключів дорівнює нулю, визначає відстань єдиності (або точку єдиності) [55, стор. 87]. За своєю суттю L_0 – це мінімальна довжина шифрованого тексту, що є необхідним для однозначного встановлення істинного ключа шифру. Але у нашому випадку важливим є те, що при сумарній довжині переданих через канал даних, меншій або рівній L_0, істинний ключ шифру може бути лише один.

При виборі даного класу шифрів крипто аналіз зводиться до наступного. Нехай осмислені тексти повідомлень у сумі на протязі одного сеансу зв'язку мають довжину L, записані природною мовою з надлишковістю D в абетці A та

складаються із m букв. Тоді, відповідно до теореми про оцінку середньої кількості ключів [55], маємо:

$$|K| / (k_L + 1) \le m^{LD}, \tag{3.1}$$

де k_L – кількість можливих фальшивих ключів при розшифруванні тексту довжиною L.

При $k_L = 0$ маємо

$$|K| \le m^{LD}. \tag{3.2}$$

Звідки

$$L \ge \log_2|K| / D \log_2 m. \tag{3.3}$$

Так що, відстань єдиності – це не тільки міра довжини криптограми, яка потрібна для здійснення компрометації шифру, але і обмежувальна умова отримання єдиності результату крипто аналізу. Тобто, якщо виконується умова (3.3), то будемо мати лише один істинний ключ шифру. Але згідно теоремі Шеннона [43]: щоб шифр, для якого виконується умова $|X| = |Y| = |K|$, був досконало стійким, необхідно і достатньо, аби розподіл ймовірностей вибору ключів був рівномірним та виконувалась умова (3.3).

Теорема Шеннона [55, стор. 70] доводить, що за вищезазначених умов шифр Вернама за модулем m є досконало стійким. Схема реалізації цього шифру полягає у наступному.

Нехай знаки відкритого тексту, криптограми і ключа шифру отримують свої значення з кільця залишків Z_m, а довжини ключа і криптограми дорівнюють довжині n відкритого тексту. Тоді рівняння шифрування можливо представити як

$$y_i \equiv (x_i + k_i) \bmod m_i , \;\; i = 1, 2, \ldots , n . \tag{3.4}$$

Рівняння (3.4) визначає процедуру шифрування n-грами відкритого тексту $(x_1, x_2, \ldots , x_n)$ на ключі (гамі) $(k_1, k_2, \ldots , k_n)$, у результаті якої утворюється криптограма $(y_1, y_2, \ldots , y_n)$.

Вище названа теорема Шеннона доводить, що для будь-якого невідомого відкритого тексту усі криптограми є рівноймовірними. А це означає, що при додаванні до будь-якого невідомого відкритого тексту за формулою (3.4) дійсно випадкової ключової послідовності (у тому ж форматі даних) завжди отримується дійсно випадкова послідовність символів. Так що, зламати шифр Вернама за визначених умов (зокрема, якщо на кожен сеанс зв'язку генеруються нові ключі шифру) не допоможуть ніякі обчислювальні потужності.

Визначення потрібної довжини ключів шифру Вернама. Потрібна довжина ключів шифру напряму пов'язана із кількістю конфіденційних даних, що мають бути передані через канал Інтернет на протязі одного сеансу зв'язку [43]. У свою чергу, збільшення кількості переданих даних, які потребують захисту, пов'язано з необхідністю збільшення потрібних комп'ютерних ресурсів сервера (в першу чергу, комп'ютерного часу), що мають бути використані для їхньої обробки. Тому визначимо значення часу τ, яке є необхідним для забезпечення роботи досконало секретної системи захисту.

Кількість даних, які потребують захисту на протязі одного сеансу обміну між виборцем і сервером, може бути знайдена з виразу

$$I_{\Sigma} = I_i + I_a + I_v + I_c, \tag{3.5}$$

де I_i – кількість інформації, що виділена для ідентифікації виборця, біт;

I_a – кількість інформації, що виділена для автентифікації виборця, біт;

I_v – кількість інформації, що необхідна для відображення результатів волевиявлення виборця, біт;

I_c – кількість інформації, що виділена для контролю цілісності даних, біт.

Для забезпечення необхідного значення відстані єдиності необхідно, щоб

$$I_{\sum} \le 2^{|K|}, \tag{3.6}$$

де $|K|$ - мінімально можлива кількість бітових ключів шифру Вернама

Приклад. Якщо обрати довжини паролів та ідентифікаторів розміром по 20 байт (або по 160 біт) кожний, а кількість видів бюлетенів, що мають бути використані в одному сеансі голосування, - десять видів з 256-ма варіантами вибору у кожному, то загальна кількість інформації, що підлягає захисту в одному сеансі зв'язку між клієнтом та сервером, буде дорівнювати 240 бітів. Якщо додати два байти для контролю цілісності, то отримаємо $I_{\Sigma} = 256$ біт. Це означає, що у цьому випадку для забезпечення досконалої секретності системи захисту з використанням шифру Вернама довжину ключа для шифрування або розшифрування необхідно обирати не менше, ніж 256 біт.

Залишається з'ясувати значення необхідної довжини ключа, для гарантованого забезпечення достатнього рівня захисту під час обміну даними за алгоритмом Диффі-Геллмана. Однак обмін даними через канал згідно цього алгоритму здійснюється у відкритому режимі, а вибір його параметрів виконується за іншими принципами, що викладені у наступному підрозділі.

3.2.5. Визначення параметрів алгоритму Диффі-Геллмана

Згідно результатів аналізу умов використання алгоритму Диффі-Геллмана (див. підрозділ 1.2) у даному випадку мова йде про вибір мультиплікативної групи над полями Галуа $GF(2^n)$, де n – безпечне просте число (*SPN*, *Safe prime number*), за яких витрати комп'ютерного часу на розв'язання задачі дискретного логарифмування повинні перевищувати реальні можливості зловмисників.

Вихідні умови:

1) Витрати комп'ютерного часу на розв'язання задачі дискретного логарифмування, з метою розкриття конфіденційної інформації, мають гарантовано перевищувати тривалість проведення виборчої кампанії.

2) Середня тривалість часу обслуговування клієнта, тобто значення інтервалу τ, для недопущення затримок в обслуговуванні виборців має обиратися у межах кількох секунд згідно даним табл. 3.2.

3) Довжина ключа повинна перевищувати 256 біт, що пов’язано з необхідністю дотримання відстані одиничності з урахуванням вище визначеної кількості даних, які підлягають захисту під час їхнього передавання через канал Інтернет.

4) Витрати часу на генерацію ключів повинні бути мінімізовані (до 10 с), щоб для кожного сеансу зв’язку можна було б генерувати нові ключі.

Захищеність алгоритму Диффі-Геллмана забезпечується складністю розв’язання задачі дискретного логарифмування. Над цією задачею працюють фахівці, які регулярно публікують результати своїх досліджень. Тому для оцінки рівня захисту даних, в залежності від вибору тих чи інших мультиплікативних груп, можна скористатись цими публікаціями [58- 65].

Відомо, що безпечні прості числа повинні задовольняти виразу

$$n = 2p + 1, \text{ де } p - \text{просте число.} \qquad (3.7)$$

Один з останніх результатів розв’язання задачі для $p=1279$, що відображений у роботі [60], можна вважати найбільшим успіхом, але хоч 1279 є простим числом, але воно не відповідає умові *SPN*, тобто не є безпечним. Для отримання цього результату було використано супер-комп’ютер із кількома сотнями ядер і витрачено близько 4,5 тисяч ядро годин. Інший ефективний результат [65]: для найбільшого значення p = 9234 було витрачено 400 тисяч ядро годин, але 9234 взагалі не є простим числом.

Аналіз публікацій щодо останніх досягнень дискретного логарифмування показав, що ця задача для випадку мультиплікативних груп над полями Галуа $GF(2^n)$, де n – безпечне просте число (*Safe prime number*), яке перевищує 500, поки що не розв’язана.

Найменшим безпечним простим числом n, що перевищує 500 і задовольняє виразу (3.5), є число 503. Це число і обрано у даній роботі в якості ключового параметру розроблюваного протоколу обміну даними між клієнтом і сервером для загальнонаціональної СДГ. При n = 503 у даному випадку

забезпечується найбільша швидкість шифрування. Цей вибір пояснюється тим, що обираючи більші значення безпечних простих чисел, ряд яких (за умов $n > 500$) має наступний вигляд:

503, 563, 587, 719, 839, 863, 887, 983, 1019, 1087, …, буде, з одного боку, ускладнюватись задача розкриття шифру, що має позитивне значення на випадок появи більш досконалих методів дискретного логарифмування, але, з другого боку, буде потрібно більше витрачати часу на шифрування, що негативно впливає на якість обслуговування виборців. Для визначення конкретних значень часу шифрування для різних значень n необхідно провести експерименти з використанням реальних сучасних комп'ютерних засобів, яким присвячено наступний розділ 4 даної роботи.

3.2.6. Метод отримання дійсно випадкових бітових послідовностей

У підрозділі 3.2.4 було показано, що умовою збереження режиму досконалої стійкості при використанні шифру Вернама є додавання згідно рівняння (3.5) до відкритої послідовності символів дійсно випадкової послідовності ключа шифру. Так що усі можливі спроби зламу цього шифру будуть пов'язані із спробами компрометації методу отримання дійсно випадкових послідовностей символів. З іншого боку, для програмної реалізації алгоритму Диффі-Геллмана в умовах використання процесора з одним ядром (котрий, як мінімум, завжди має бути у розпорядженні будь-якого користувача СДВ) необхідно мати можливість отримувати дійсно (а не псевдо) випадкові бітові послідовності, а швидкість генерування дійсно випадкових бітових послідовностей має бути узгоджена із швидкістю знаходження степені над полем $GF(2^n)$, де n=503.

У даній роботі для генерації дійсно випадкових чисел пропонується використати таке природне явище як нестабільність частоти кварцових резонаторів, що є невід'ємними частинами будь-якого комп'ютера. У кожному комп'ютері масоого виробництва є два незалежних кварцових резонатори, один з яких має частоту 32,768 кГц і використовується для таймера, а другий - з частотою не менше, ніж 14318,18 кГц, – для формування тактових сигналів

процесора. Обидва ці генератори мають нестабільність від 10 до 100 *ppm* (*parts per million*) або 10^{-5} – 10^{-4}. Якби ці генератори були синхронізовані між собою, то в будь-якому довільним чином обраному часовому інтервалі, який вимірюється цілою кількістю імпульсів першого генератора, була б завжди однакова кількість імпульсів другого генератора (що має більшу частоту генерування імпульсів). Однак генератори, що функціонують у складі кожного комп'ютера, не синхронізовані між собою, а їх сумарна нестабільність є сприяючим фактором для отримання дійсно випадкових бітів. Оскільки однією з відомих причин нестабільності частоти кварцових резонаторів є фазовий джиттер [75], який породжується впливом на кристал кварцу природного іонізуючого випромінювання, а саме нейтронного та γ-випромінювання, то ця нестабільність має характер білого шуму і не може бути передбачуваною. Ця нестабільность влаштовує, як виробників, так і користувачів комп'ютерної техніки, бо вона практично не впливає на якість роботи комп'ютерів, а те, як її можна використовувати для отримання випадкових бітових послідовностей показано на рис. 3.5.

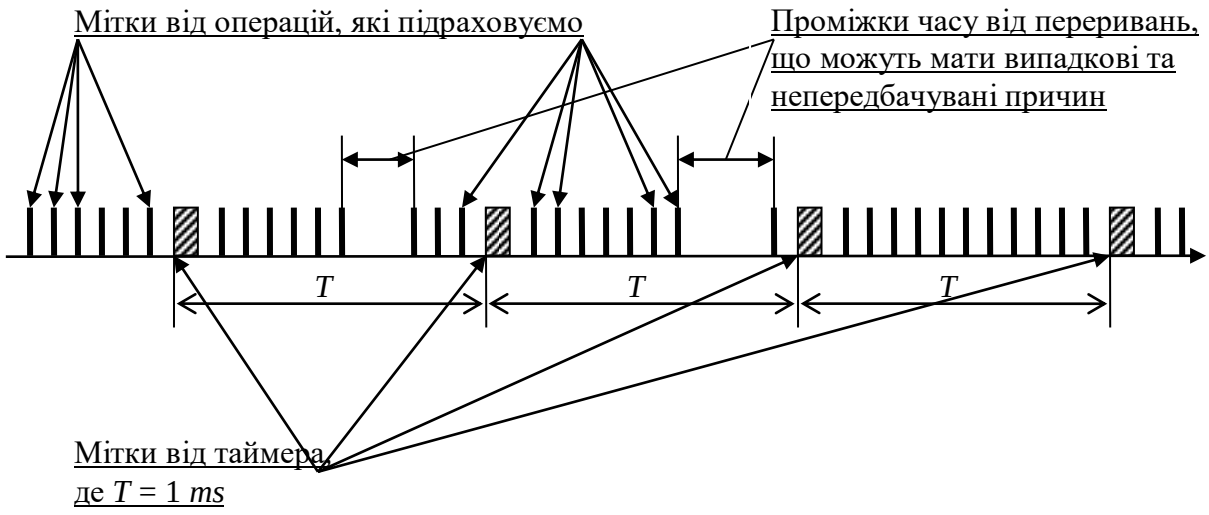

Рис. 3.5. Метод отримання дійсно випадкових бітових послідовностей

Ідея побудови генератора дійсно випадкових бітових послідовностей полягає в тому, що у разі взаємної нестабільності генераторів неможливо

передбачити кількість операцій, що будуть знаходитися у межах обраного часового інтервалу [76]. Кількість нарахованих операцій, що виконуються процесором комп'ютера в інтервалах тривалістю T залежить від значення тактової частоти і може змінюватись від сотень до тисяч. Випадковий характер цієї кількості підсилюється наявністю випадкових проміжків від переривань процесу підрахунку Моменти появи і тривалість цих переривань є непередбачуваними, бо вони обумовлені запитами з мережі Інтернет. Відомо, що потік цих запитів має фрактальний характер, функція розподілу котрого не є відомою. Зокрема в якості моделі потоку запитів розглянуто напівнескінчений відрізок стаціонарного випадкового процесу X дискретного аргументу (часу) $t=0,1,...,k, \ldots$, тобто часовий ряд $\{X_k;\ k=0;1;2;...\}$, де k - поточний номер часового інтервалу усереднення процесу X. Тоді точкове значення k-го відліку часового ряду $\{X_k^{(\tau)};\ k=0;1;2;\ \ldots\}$ при моделюванні потоку запитів інтерпретується як кількість запитів x_k^{τ} , що надійшли у вузол обробки даних протягом k-го інтервалу часу тривалістю τ. Тобто, у даному випадку τ - це інтервал усереднення запитів у потоці. Якщо ряд $\{X_k^{(\tau)};\ k=0;1;2;\ \ldots\}$ унормувати відносно τ, то отримаємо ряд $\{I_k^{(\tau)};\ k=0;1;2;\ \ldots\}$, в якому k -й компонент визначає поточну інтенсивність запитів на k -ому кроці його усереднення. Кількість запитів, що надійшли у вузол обробки даних протягом k-го інтервалу часу тривалістю τ, дорівнює максимально можливому значенню індексу i_{max} , що задовольняє нерівності

$$\tau \geq sum\ \Delta\tau_{k,i} = \Delta\tau_{k,1} + \Delta\tau_{k,2} + \ldots + \Delta\tau_{k,imax}, \qquad (3.8)$$

де $\Delta\tau_{k,i}$ - проміжок часу між сусідніми запитами у потоці, $i = 0,1,2, \ldots$ - поточний номер цього проміжку, а k - поточний номер часового інтервалу усереднення процесу $\{X_k^{(\tau)};\ k=0;1;2;\ \ldots\}$.

Отже, k-й компонент ряду $\{I_k^{(\tau)};\ k=0;1;2;\ \ldots\}$, що визначає поточну інтенсивність запитів на k-ому кроці його усереднення, визначено як

$$I_k^{\{\tau\}} = x_k^{\tau}/\tau. \quad (3.9)$$

Використання частотного методу криптоаналізу втрачає сенс, якщо упевнитись, що потік запитів має ознаки фрактального процесу. Таку упевненість отримуємо шляхом оцінювання значень параметра Херста за індексом дисперсії. IDC визначається як відношення дисперсії кількості оброблених запитів на заданому часовому інтервалі T до математичного очікування цієї величини:

$F(T) = \frac{Var[N(T)]}{E[N(T)]}$, де $N(T)$ – кількість запитів на інтервалі T. (3.10)

Для самоподібних процесів натуральний логарифм $F(T)$-1 як функція від натурального логарифма інтервалу T лінійно зростає, оскільки:

$$\ln[F(T)-1] = (2H-1)\ln T + y, \text{ де } y = \ln\left[\frac{2K}{\alpha(1-\alpha)} M_{\Gamma}(\alpha) B^{-\alpha/2}\right],$$

$$M_{\Gamma}(x) = \frac{\Gamma(1+x/2)\Gamma(1-x)}{\Gamma(1-x/2)}. \quad (3.11)$$

Оцінювання параметру Херста може бути здійснено по кутовому коефіцієнту нахилу цієї прямої лінії.

Докладний опис програмної реалізації генератора дійсно випадкових бітів, який побудований на основі цієї ідеї, надано у розділі 4.3 даної роботи.

3.2.7. Обмін даними між клієнтом і сервером

Метод захищеного обміну даними між клієнтами та сервером системи дистанційного волевиявлення являє собою надбудову над стандартним протоколом прикладного рівня *HTTP*. Схема обміну даними між виборцем і сервером, що здійснюється з використанням цього методу, зображена на рис.3.6.

Даний метод гарантує захист канальних даних як на стадії обміну ключами шифру, так і на стадії обміну конфіденційною інформацією під час здійснення волевиявлення.

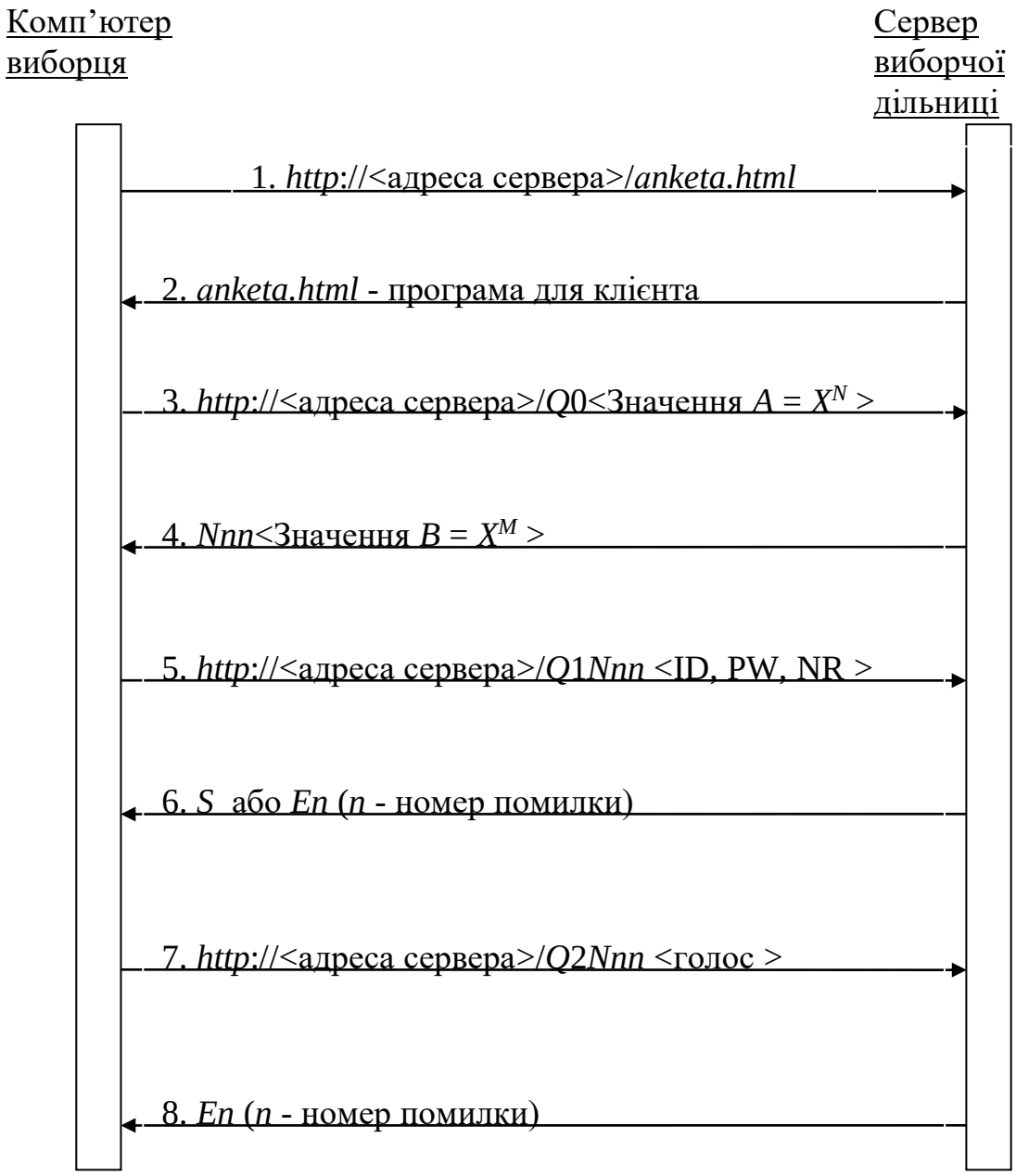

Рис. 3.6. Метод захищеного обміну даними між виборцем і сервером під час голосування

На рис. 3.6 дві початкові дії (верхні два рядка) являють собою стандартну процедуру отримання виборцем від сервера за протоколом *HTTP* у відкритому режимі файлу *anketa.html*. Після цього на екрані виборця з'явиться анкета, у яку він повинен занести свої паспортні дані і пароль (приклад форми цієї анкети показано на рис. 4.9). Одночасно із заповненням анкети виборцем на його комп'ютері починає роботу програма, яку вбудовано у файл *anketa.html*. Ця програма (приблизно за одну секунду) формує дійсно випадкову послідовність

N довжиною 503 бітів, знаходить степінь примітивного елементу *X* над полем $GF(2^{503})$, а результат $A = X^N$ пересилає на сервер (дія 3 на рис. 3.5) слідом за комбінацією символів *Q*0, які є ознакою відправки числа *A*.

Сервер, отримавши число *A*, формує дійсно випадкову послідовність *M* довжиною 503 бітів, знаходить число $B = X^M$ і відправляє його клієнту після символів *Nnn*, де замість *nn* ставиться умовний номер клієнта від 00 до 99 (дія 4 на рис.3.6).

Програма сервера може одночасно обслуговувати до 100 клієнтів, тому на час обслуговування кожному клієнту надається умовний номер, який ніяк не пов'язаний з ідентифікаційними даними. Після виконання дії 4, програма сервера знаходить ключ *C* з виразу

$$C = A^M = X^{NM}, \tag{3.10}$$

а клієнтська програма знаходить цей самий ключ з виразу

$$C = B^N = X^{MN}. \tag{3.11}$$

Після цього, маючи однакові дійсно випадкові ключі, клієнт і сервер здійснюють обмін конфіденційними даними із використанням шифру Вернама. Наступний запит клієнта, що починається з символів *Q*1, являє собою зашифровані паспортні дані, пароль і номер режиму роботи. Номер режиму 0 означає голосування, а номери 1, 2 та 3 – отримання різних довідок. Відповідь сервера на цей запит може починатись з букви *E,* після якої буде номер помилки, або однієї букви *S,* коли помилок не знайдено. В усіх випадках клієнт отримає на екрані відповідне текстове повідомлення. Відправка на сервер зашифрованого результату голосування виборця починається з символів *Q*2, після чого відповідь сервера буде відображено на екрані клієнта у вигляді текстового повідомлення. Варіанти цих повідомлень показані на рисунку 4.12.

Даний протокол реалізовано у вигляді програми на мові *JavaScript*, текст якої надано у додатку і описано у розділі 4.3.

3.3. Метод протидії загрозам, що пов'язані із зловживаннями під час заповнення бази даних сервера

Процедура заповнення бази даних сервера, що містить реєстр виборців, пов'язана з можливістю реалізації двох видів загроз. По-перше, можливість створення фіктивних виборців. По-друге, можливість здійснення спроб дешифрувати паролі виборців з боку персоналу, який має доступ до файлу бази даних сервера.

3.3.1. Загрози створення фіктивних виборців

В умовах Інтернет-голосування загроза, що пов'язана із появою в електронному списку виборців для дистанційного голосування фіктивних записів, набуває особливого значення, бо голосування через Інтернет позбавляє можливості візуального контролю за діями виборців. В Україні створено і діє електронний Державний реєстр виборців (ДРВ). Для реалізації даної загрози зловмисник буде намагатися занести в електронний реєстр якнайбільшу кількість фіктивних виборців. Маючи достатньо велику кількість фіктивних виборців, зловмисник може у суттєвій мірі спотворити результати дійсного волевиявлення.

3.3.2. Метод протидії

1. Має бути дотримана умова реєстрації виборців для дистанційного голосування тільки у разі їх наявності у ДРВ.

Тоді для реєстрації фіктивного виборця треба буде ще й заносити його у ДРВ, що ускладнює процедуру реалізації даної загрози. Слід зауважити, що для отримання права на дистанційне голосування за технологією, яка запропонована у даній роботі і докладно пояснена у розділі 4, кожен виборець повинен пройти очну перевірку з пред'явленням паспорту не менше, ніж двічі. Перша така перевірка відбувається під час занесення виборця в базу даних для

дистанційного голосування (див. рисунок 4.5), а друга – в передвиборчий період для отримання спеціального паролю (див. рисунок 4.8). Наявність двох перевірок ускладнює можливість зловживань, але не ліквідує дану загрозу.

2. ПЗ сервера має забезпечувати можливість оперативної та безперешкодної перевірки через Інтернет з боку будь-яких зацікавлених осіб кількості зареєстрованих виборців у розрізі найменувань вулиць, позначень будинків та квартир.

Прийнятий для даної розробки принцип повної недовіри до будь-яких суб'єктів виборчого процесу, включаючи офіційний обслуговуючий персонал, передбачає необхідність створення засобу боротьби із загрозами створення фіктивних виборців з боку виборчої спільноти у цілому. Безперешкодний контроль у режимі реального часу з боку суспільно активних громадян якраз і є таким засобом.

Перевірка списку голосуючих на наявність фіктивних записів не є простою процедурою, бо не існує такої ознаки, яка б дозволяла будь-кому легко виявити приписки. Тому важливо довести можливість цієї перевірки до кожного виборця. Саме в умовах дистанційного голосування через Інтернет з'являється можливість за допомогою програмних засобів реалізувати таку перевірку. Для цього у ПЗ сервера передбачена можливість отримання довідок про кількість голосуючих з підсумками у розрізі назв вулиць, номерів будинків та квартир. За допомогою цих довідок можна контролювати, чи не з'явились в межах виборчої дільниці невідомі вулиці, будинки чи квартири, а також впевнитись, що кількість виборців у квартирах відповідає дійсності. Для виборів в межах установ можуть бути надані списки виборців у розрізі підрозділів. Об'єктивну перевірку таких списків можуть зробити будь-які зацікавлені особи або групи осіб.

3.3.3. Приклад реалізації метода протидії створенню фіктивних виборців

ПЗ сервера виборчої дільниці формує такі три варіанти довідок:

1) кількість виборців, що голосують дистанційно у розрізі усіх вулиць у межах виборчої дільниці;

2) кількість виборців, що голосують дистанційно, по будинках у межах даної вулиці;

3) кількість виборців, що голосують дистанційно, по квартирах у межах даного будинку.

В цих довідках не розкриваються ніякі персональні дані виборців, але вони дозволяють виявити можливі приписки, особливо великого масштабу. Слід зауважити, що тільки приписки значного масштабу, можуть суттєво вплинути на результат волевиявлення.

Текст довідок програма відправляє виборцям на електронну пошту, яку виборці обов'язково повинні вказати під час реєстрації.

Фрагмент тексту серверної програми на мові *JavaScript* для формування однієї із цих довідок представлено на рис. 3.7.

3.3.4. Загроза здійснення спроб дешифрувати паролі виборців з боку персоналу, який має доступ до файлу бази даних сервера

Крім дописування фіктивних виборців, персонал, який має доступ до файлу бази даних, може спробувати дешифрувати паролі виборців. Це надає можливість зловмисникам зробити спробу введення паролів для голосування замість тих виборців, які ще не встигли проголосувати. При цьому будуть виникати непорозуміння, якщо такі виборці спробують проголосувати традиційним методом, бо їх буде викреслено зі списку на отримання паперових бюлетенів. Проте в деяких випадках виникнення таких непорозумінь йде на користь зловмисникам.

3.3.5. Метод протидії

Проаналізуємо можливість реалізації даної загрози, беручи до уваги, що через відкритість ПЗ, що використовується, є можливість скористатись для дешифрування паролів тією ж програмою, яка використовується для їхнього шифрування. Беручи до уваги, що стандартні методи зберігання паролів мають широко відомі слабкі місця, а в Інтернеті існують готові програми для розкриття таких паролів за лічені секунди, то програма, що реалізує функцію зберігання паролів, має використовувати той самий блок, що обчислює

значення степені примітивного елементу поля Галуа $GF(2^{503})$ для захисту даних під час передачі.

У цьому випадку неможливо буде розкривати паролі зазначеним вище методом.

```
{ // Вивід довідок
var SUBJ; // Тема листа
var TEXT='Кількість виборців, що голосують дистанційно \r\n';
var KZ,KZV; // Кількість зареєстрованих виборців, ... всього
var KG,KGV; // Кількість виборців, що проголосували, ... всього
 switch (NR[SNL])
 {
 case 1: SUBJ='Довідка по дільниці';
 TEXT=TEXT+'    по дільниці '+DILN+'\r\n'+
'------------------------------------------------------------------------------------- \r\n'+
':  Назва вулиці   : Зареєстровано : Проголосували \r\n' +
'------------------------------------------------------------------------------------- \r\n';
 KZ=KZV=KG=KGV=0;
 for (j=0; j<KVUL; j++) // Цикл по вулицям
   {   S2=NU2[j]
       for (i=0; i<Kstr; i++) // Цикл по всім рядкам БД
   {
          if (String.fromCharCode(SFILE[i*Lstr+1])=='0')
              { // Обираємо тільки дійсних виборців
               if (String.fromCharCode(SFILE[i*Lstr+119])==S2[0] &&
                 String.fromCharCode(SFILE[i*Lstr+120])==S2[1])
                { // Обираємо тільки j-ту вулицю
                 KZ++;
                 if (GOLOSF[i]==1) KG++;
                }
              }
         }
       TEXT=TEXT+VUL[j]+'\t\t\t'+KZ+'\t\t\t'+KG+'\r\n';
       KZV=KZV+KZ; KGV=KGV+KG; KZ=KG=0;
}
TEXT=TEXT+
'------------------------------------------------------------------------------------- \r\n'+
' Усього'+'\t\t\t\t'+KZV+'\t\t\t'+KGV+'\r\n';
 break;
 case 2: SUBJ='Довідка по вулиці';
 TEXT=TEXT+'    по вулиці'+VUL[KODVUL]+'\r\n'+
'------------------------------------------------------------------------------------- \r\n'+
':  Номер будинку   : Зареєстровано : Проголосували \r\n' +
'------------------------------------------------------------------------------------- \r\n';
```

Рис. 3.7. Фрагмент серверної програми для відправлення довідок на електронну пошту виборця

Дійсно, оскільки про дискретне логарифмування над таким полем інструкцій немає, то можна скористатись відомим методом автоматичного перебору різних варіантів паролю. Для прискорення процесу пошуку паролів можна скористатись паралельним або хмарним обчисленням. Якщо одне таке обчислення потребує близько секунди, то для перебору паролів мінімальної довжини у 10 байт треба буде витратити приблизно 10^8 ядро годин, або на 100 тисячах комп'ютерів – 1000 годин (40 діб). Так що, для паролів максимальної

довжини (16 байт) в сучасних умовах неможливо розкривати паролі цим методом.

Програми на мові *JavaScript* для захисту паролів представлена на рис. 4.3.

3.4. Метод протидії загрозам, що пов'язані із протиправним впливом на суб'єктів процесу дистанційного волевиявлення

3.4.1. Загрози, що пов'язані із протиправним впливом на суб'єктів процесу дистанційного волевиявлення

Трапляються випадки впливу на виборців методами морального або іншого тиску, які можуть призвести до того, що замість власного рішення виборець під час голосування буде вимушений підтримати таке, яке не співпадає з його власним. Подібні випадки здатні вплинути на результат виборів, спотворюючи справжність виявлення суспільної думки, що заважає процесу нормального розвитку демократичного суспільства.

3.4.2. Метод протидії

Досвід вивчення цієї загрози свідчить про те, що зловмисник обов'язково хоче дізнатись справжній результат голосування виборця. Для цього виборців примушували вкидати вже заповнені бюлетені, отримані від зловмисників, а свої порожні повертати за винагороду. Інших виборців примушували сфотографувати бюлетені після заповнення.

Головна задача у боротьбі з даною загрозою в умовах дистанційного голосування полягає в унеможливленні дізнаватись про справжній результат голосування, навіть, якщо виборець віддасть ідентифікаційні дані і пароль зловмиснику, щоб той міг проголосувати замість нього.

Вважаючи, що усе ПЗ СДГ є повністю відкритим, а методи боротьби із загрозами не можуть бути у повній мірі замасковані від зловмисників, то єдина надійна протидія даній загрозі полягає у повному приховуванні інформації, яка потрібна зловмиснику. При цьому можна також використовувати дезінформацію. Важливо, щоб не було жодної ознаки не тільки про результат

голосування, але і про той факт, що голосування відбувалось. Знання пароля для голосування є суттєвою перевагою виборця над зловмисником, бо зловмисник не може дізнатись пароль ніяк, окрім як від самого виборця [77].

Головна ідея, яку було покладено в розробку методу захисту від загроз подібного типу, полягає в тому, щоб система надавала виборцю абсолютно ідентичні повідомлення у разі голосування, як з вірним паролем, так і з деякою множиною невірних паролів, наприклад, тих, що мають ту ж саму довжину, що й вірний, або починаються з того ж символу. При цьому зарахувати система повинна тільки один результат з вірним паролем.

Опис механізму реалізації цього методу, а також програмної реалізації СДГ, яка побудована з використанням цього методу, надано у підрозділі 4.5.

Висновки до третього розділу

3.1. У системах дистанційного голосування з використанням мережі Інтернет, які побудовані за принципом встановлення безпосереднього захищеного зв'язку між комп'ютером виборця і сервером підрахунку голосів, існує ряд специфічних загроз, які можуть суттєво вплинути на результат волевиявлення виборців аж до повного спотворення дійсної інформації. Найбільш небезпечними щодо спотворення результатів голосування є загрози, що пов'язані із можливими зловживаннями під час обслуговування серверів виборчих дільниць, на яких відбувається зарахування голосів, з боку персоналу, який може бути зацікавленим у спотворенні результатів виборів.

Вперше запропоновано метод протидії загрозам штатній роботі сервера, що базуються на спостереженні за станом активних процесів (діючих програм) і усіх файлів на сервері з боку необмеженої кількості зацікавлених осіб, у т.ч. громадських контролерів, за умов повної відкритості усіх програмних засобів і публікації розкладу дій персоналу, що адмініструє сервер. Даний метод унеможливлює створення непомічених загроз на сервері СДВ за умов використання виключно відкритого ПЗ, склад та специфікації котрого є

опублікованими напередодні виборчої кампанії і, отже, є відомими широкому загалу. Метод створює умови, за яких будь-яке зловживання щодо роботи сервера не може залишитись непоміченим. Будь яка спроба скористатись нештатною програмою або підмінити сам сервер буде зафіксована та задокументована будь-якою зацікавленою особою незалежно від її статусу або ролі у виборчому процесі.

3.2. Дістав подальший розвиток метод досконало захищеного обміну даними через Інтернет між клієнтами та сервером СДВ у класі симетричних систем з використанням ендоморфного шифру, що забезпечує досконало стійкий захист канальних даних від порушень їхньої конфіденційності та цілісності. Конкретизовано умови використання цього методу у загальнонаціональній системі дистанційного голосування.

Надана оцінка необхідної тривалості обслуговування запиту клієнта в залежності від значення припустимої тривалості очікування клієнтів у чергах до серверу. Показано, що збільшення інтервалу обслуговування клієнта на величину більшу, ніж 6с , призводить до того, що кожний п’ятий виборець буде очікувати своєї черги до сервера від 7с до 14с, а кожний сімнадцятий – від 14с до 21с.

Доведено, що для забезпечення гарантованої захищеності обміну даними через Інтернет у СДВ доцільно використати досконало стійкий шифр Вернама, коректність застосування котрого ппотребує випадкових послідовностей бітів. Визначено умови забезпечення стійкості при обміні даними через Інтернет. Визначена потрібна довжина ключової послідовності для шифру Вернама.

Для забезпечення захисту ключової інформації при дистанційному доступі виборця до сервера через мережу Інтернет застосовано алгоритм Диффі-Геллмана. Визначено необхідні значення параметрів цього алгоритму, за яких потенційно можливі витрати комп’ютерного часу на розв’язання задачі дискретного логарифмування перевищують тривалість виборчої кампанії. Зокрема, обрана для реалізації мультиплікативна група над полями Галуа $GF(2^n)$, де n – безпечне просте число (SPN), що дорівнює 503.

Розроблено метод отримання випадкових бітових послідовностей на комп’ютерах масового виробництва без будь-яких додаткових програмно-апаратних засобів. Цей метод базується на використанні такого природного явища як нестабільність частоти кварцових резонаторів, що є невід’ємними частинами будь-якого комп’ютера. Реалізація такого методу є необхідною умовою для коректного функціонування досконало стійкої системи захисту, що реалізована з використанням шифру Вернама.

3.3. Розроблено метод протидії загрозам, що виникають внаслідок зловживаннь під час заповнення електронного реєстру виборців.

Протидія загрозі створення фіктивних виборців забезпечена шляхом створення засобів оперативної та безперешкодної перевірки через Інтернет з боку громадських контролерів кількості зареєстрованих виборців у розрізі найменувань вулиць, позначень будинків та квартир. Надано можливість кожному виборцю отримати довідки про кількість голосуючих по вулицям в межах виборчої дільниці, по будинкам в межах вулиці та по квартирах в межах будинку, що унеможливлює дописування фіктивних виборців в значних масштабах.

Протидія загрозі здійснення спроб дешифрувати паролі виборців з боку персоналу, який має доступ до файлу бази даних сервера, забезпечена шляхом використання у програмному модулі, що реалізує функцію зберігання паролів, блоку, що обчислює значення степені примітивного елементу поля Галуа $GF(2^{503})$. Доведено надійність зберігання паролів завдяки обраному методу шифрування.

3.4. Запропоновано метод протидії загрозам, що пов’язані із силовим або психологічним впливом на виборців під час голосування, які можуть спотворювати справжність результатів виявлення. Сутність цього методу полягає у повному приховуванні інформації, яка потрібна зловмиснику, що робить недоцільними подібні загрози і дозволяє виборцям реалізувати своє право на власне волевиявлення.

РОЗДІЛ 4

ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ЗАХИЩЕНОЇ СИСТЕМИ ТЕХНІЧНОЇ ПІДТРИМКИ ПРОЦЕСІВ ДИСТАНЦІЙНОГО ВОЛЕВИЯВЛЕННЯ

4.1. Завдання експериментального дослідження захищеної системи технічної підтримки процесів дистанційного волевиявлення

Для практичного підтвердження можливостей досягнення результатів, що отримані в роботі, проведено натурне моделювання усіх засобів СДВ, у т.ч. засобів захисту інформації, в умовах, що імітують процес загальнонаціональних виборів в Україні. Задачею досліджень є доведення того, що комплекс програмно-технічних засобів СДВ, створений на основі запропонованих в роботі методів, здатен реалізувати наступні вимоги:

1) Гарантований захист інформації на сервері виборчої дільниці від будь-якого несанкціонованого втручання, як з боку сторонніх осіб, так і з боку штатного персоналу, який володіє знаннями про всі без винятку засоби захисту, включаючи автентифікаційні дані, і має повний доступ до всіх програмно-технічних засобів СДВ.

2) Гарантований захист інформації від витоку та від спотворень під час обміну даними між сервером і клієнтами (виборцями) у відкритих каналах мережі Інтернет в умовах, коли задіяні методи захисту є відомими зловмиснику.

3) Захист інформації клієнтів під час обміну даними із сервером від можливого впливу з боку ПЗ СДВ.

4) Гарантована захищеність ОС сервера від порушення її цілісності в умовах, коли надана можливість контролю її стану протягом всього часу функціонування СДВ будь-якому суб’єкту, що знаходиться поза меж домену безпеки.

5) Зручний механізм перевірки прийнятих розробниками СДВ технічних рішень з боку широкого кола зацікавлених осіб.

6) Інваріантність прикладного ПЗ щодо різних програмно-апаратних платформ (з метою зняття зайвих обмежень під час перевірок функціонування ПЗ на будь-якому обладнанні).

7) Наявність засобів контролю щодо виявлення фіктивних виборців у базі даних СДВ.

8) Наявність механізмів, що забезпечують можливість здійснювати акти волевиявлення за власним розсудом в умовах безпосереднього тиску з боку зловмисників.

9) Наявність механізмів протидії перехопленню голосів виборців іншими особами та можливості проголосувати кожному з виборців більше одного разу.

10) Гарантоване збереження таємниці голосів виборців.

11) Неможливість будь-якого втручання в процес підрахунку голосів.

Експериментальному підтвердженню можливості виконання всіх цих вимог присвячено даний розділ.

4.2. Базові програмні рішення та засоби програмування

У розділі 2 показано, що більшість можливостей реалізації інформаційних загроз на сервері СДГ пов’язана з особливостями побудови ОС. Для спрощення процедури контролю бажано створити спеціальну ОС, але тоді тільки поліпшаться умови контролювання, захищеність же від визначених вище загроз при цьому не підвищиться. Тому для сервера СДГ була обрана ОС, у якій визначені вище (у розділі 1) загрози неможливо реалізувати. У нашому випадку це - операційна система *OpenBSD* у мінімальній конфігурації, оскільки серед ОС з відкритим програмним кодом вона є найбільш простою і надійно захищеною. Чим менше файлів на сервері, тим легше контролювати його стан. Тому цю мінімальну конфігурацію ОС *OpenBSD* ще зменшили шляхом видалення файлів, які непотрібні під час функціонування сервера СДГ.

Функції контролю за станом сервера полягають в перевірці наявності файлів ОС (команди *ls* та *cd*) і активності процесів (команди *ps aux* та *top*). Як показано у розділі 2, таких перевірок достатньо, щоб впевнитись у відсутності загроз, що пов'язані з порушеннями порядку обслуговування сервера. При цьому користувачів-контролерів з обмеженими правами, які зможуть відслідковувати стан файлів та процесів на сервері, в системі *OpenBSD* може бути необмежена кількість.

Але навіть у випадку оптимально обраної з точки зору ТЗІ ОС необхідно ще бути впевненим в тому, що зловмисник в процесі виборчої кампанії не замінить її на який-небудь фальсифікат (див. розділ 1). Тому у даному випадку ПЗ побудовано таким чином, щоб сервер СДГ виконував тільки одну прикладну програму.

В сучасному арсеналі програмних засобів для роботи в комп'ютерних мережах є досконало безпечні засоби, які позбавлені можливості нанесення будь-якої шкоди інформаційним ресурсам комп'ютерів. До таких засобів належить протокол передавання гіпертексту *http* за умов, що всі файли, які будуть передаватись, являють собою тексти на мові *HTML* з можливим їх доповненням програмами на мові *JavaScript* [66]. Поєднання двох комп'ютерних мов *HTML* та *JavaScript* набуло значного поширення у всесвітній мережі через таку важливу властивість, як гарантування безпеки для клієнтських комп'ютерів. Усі сучасні засоби доступу до прикладних ресурсів мережі Інтернет (браузери) мають вбудовані засоби виконання *JavaScript* програм. Але, оскільки через різні перехресні посилання, якими завжди переповнений контент Інтернету, існує можливість з безпечного ресурсу перейти на небезпечний, то для гарантування безпечного користування необхідно робити перевірку коду сторінки. Таку перевірку з боку клієнтів СДГ зробити досить легко, бо коди всіх комп'ютерних програм даної системи є у відкритому вигляді, тому код сторінки є завжди відомим. Обираючи для створення прикладного ПЗ для СДГ мову *JavaScript* , ми одночасно отримуємо наступні три важливі властивості:

- виборцям достатньо мати звичайний доступ до мережі Інтернет;
- програмне забезпечення СДГ не може заподіяти шкоду клієнтам;
- вибір єдиної мови *JavaScript* для клієнтських і серверних програм спрощує процеси програмування і ознайомлення з програмами.

Слід зауважити, що мова *JavaScript* досить проста і дуже поширена, а також не потребує ніяких проміжних файлових перетворень, тобто файл з текстом програми на мові *JavaScript* можна відправляти прямо на виконання. Крім того, такий метод спілкування між сервером і клієнтом є безпечним також і для сервера, бо через інтерфейс передавання гіпертексту на сервер не може потрапити жоден неконтрольований байт [67].

Для використання мови *JavaScript* не тільки для клієнтських програм, а також і як мову програмування для серверних програм, необхідно додаткове програмне забезпечення, а саме пакет *Node.js*, який є відкритим і таким, що може бути встановленим на різних комп'ютерних платформах. Це означає, що все розроблене з використанням пакету *Node.js* прикладне ПЗ може перевірятись і дороблятись на довільному сучасному комп'ютері під довільною ОС.

Таким чином, усіх перелічених вище рішень щодо вибору ОС, мов та засобів програмування цілком достатньо для створення прикладного ПЗ СДГ. У розділі 2 показано, що на сервері СДГ, з метою забезпечення захисту інформації, всі команди, які можна було б використати для утворення загроз, підлягають блокуванню. От же, на сервері не має можливостей створювати і корегувати файли. Усі файли ПЗ потрапляють на сервер тільки через єдиний інтерфейс під контролем громадських спостерігачів. Кожен файл при цьому перевіряється шляхом порівняння із заздалегідь перевіреним еталоном. Зрозуміло, що через такі обмеження, створювати ПЗ безпосередньо на самому сервері не є можливим. Для розробки ПЗ мають використовуватися будь-які інші комп'ютери.

4.3. Реалізація захищеного обміну даними між клієнтами і сервером

Важливою вимогою до обміну даними між сервером виборчої дільниці та виборцями є захист інформації від порушень її конфіденційності та цілісності. Такому захисту підлягають ідентифікаційні дані виборців, а також дані про їх волевиявлення. Для цього (див. розділ 3), перед кожним початком сеансу обміну даними між клієнтом і сервером, прикладне ПЗ формує дійсно (а не псевдо) випадкові послідовності бітів, як на боці клієнта, так і на боці сервера. Відомо, що додавання за модулем 2 до кожного інформаційного біта випадкових бітів унеможливлює дешифрування інформації під час передавання, бо єдиним методом, що дозволяє отримати вихідну інформацію в такому випадку, є додавання тієї самої випадкової бітової послідовності. Цей метод захисту має назву шифру Вернама. З урахуванням вищезазначеного, у даній роботі побудова захищеного каналу для обміну даними між клієнтом і сервером здійснена на основі розв’язання двох наступних задач:

1) Генерація чисто випадкових послідовностей бітів для кожного сеансу зв’язку, як на комп’ютері виборця, так і на сервері.

2) Відкритий обмін службовими даними між клієнтом і сервером з метою отримання однакових випадкових послідовностей бітів на обох сторонах обміну даними за умов збереження в таємниці від третіх сторін отриманих випадкових послідовностей.

За стандартом криптографічного захисту інформації [68] для утворення дійсно випадкових бітів рекомендовано обирати значення молодшого біту таймера в комп’ютері. Але враховуючи те, що значення цього біту змінюється кожної мілісекунди, то, фактично, тільки перший біт буде дійсно випадковим, а значення всіх наступних бітів буде залежати від частоти опитувань. У разі стабільної частоти опитувань послідовність бітів взагалі не буде випадковою, а для частот вище за 1 кГц значення біта не будуть змінюватись в інтервалах тривалістю 1 мс. Таким чином, щоб отримати за цим методом дійсно випадкову

послідовність бітів, необхідно генерувати послідовність інтервалів випадкової довжини.

Розв’язання другої задачі здійснювалось за відомим алгоритмом Диффі-Геллмана. Для реалізації цього алгоритму необхідно підготувати дані у вигляді таблиці степенів примітивного елементу поля Галуа і при цьому кількість рядків цієї таблиці має співпадати з необхідною кількістю випадкових бітів. Тому в цикл заповнення таблиці вбудовано генератор випадкових бітів. Інтервали часу для обчислення різних рядків таблиці не можуть бути однаковими через різну кількість операцій під час множення різних елементів поля Галуа. Крім того, має місце нестабільність кварцевих резонаторів, один з яких обслуговує таймер, а другий обслуговує процесор. Ці резонатори не синхронізовані між собою. Їх сумарна нестабільність є сприяючим фактором для отримання дійсно випадкових бітів від таймера. Оскільки комп’ютерний час для розрахунку усіх значень рядків таблиці складає близько однієї секунди, то за цей час процесор буде неодноразово перериватись для обслуговування різних процесів, що також сприяє невизначеності довжини інтервалів. Останнє, що зроблено для підсилення випадковості довжини інтервалів, це додавання до отриманого випадкового біту за модулем два бінарного значення функції *Math.random*, яка генерує випадкові числа в діапазоні від 0 до 1. Таким чином, випадковий характер послідовності бітів у даному випадку обумовлений впливом на значення кожного з бітів п’ятьох незалежних один від одного факторів, що є цілком достатнім для визнання отриманої послідовності бітів дійсно випадковою [69].

Фрагменти розробленої у рамках даної роботи програми на мові *JavaScript*, що реалізують даний цикл для випадку поля Галуа $GF(2^{503})$, представлені на рисунках 4.1. та 4.2.

```
// Всі елементи поля Галуа мають вигляд масивів з 503 чисел 1 або 0.
// M1[1] – молодший біт, M1[503] – старший біт, M1[0] - резерв
var M1 = [504];  // M1[1] – молодший біт, M1[503] – старший біт, M1[0] - резерв
var M2 = [504];  // M1[], M2[] - множники
var R = [504];  // R[] - результат множення
// Множення виконуємо за правилом поліномів, де M1[1] – коефіцієнт до X^0,
// M1[2] – коефіцієнт до X, M1[3] – коефіцієнт до X^2, M1[4] – коефіцієнт до X^3
// M1[503] – коефіцієнт до X^502,
function MULT()  // Функція множення елементів поля Галуа GF(2^503)
{         // Для скорочення користуємось поліномом X^503=X^3+1
 var i,j,r,r1,r2,r3; // Змінні, що означають номери елементів
  for (i=1;i<=503;i++) R[i]=0; // Заносимо у результат нулі для початку
  for (i=1;i<=503;i++) // Починаємо зовнішній цикл множення поліномів по елементах M1[]
  if (M1[i]==1) // Беремо тільки одиниці з M1[], бо множення на нуль дає нуль
  {
    for (j=1;j<=503;j++) // Починаємо внутрішній цикл множення по елементах M2[]
    if (M2[j]==1) // Беремо тільки одиниці з M2[], бо множення на нуль дає нуль
    {
    r=i+j-1; // Обчислюємо номер елемента масиву R[] для занесення результату множення
      if (r>503) // У разі, коли номер элементу вийшов за межі масиву R[],
      {                 // будемо користуватись поліномом X^503=X^3+1 для заміни
       r=r-503;
       if (r>=501) // У разі, коли після заміни теж вийшли за межі масиву R[],
        {    // будемо ще раз користуватись поліномом X^503=X^3+1
        r=r-501;
        r1=1+r; r2=4+r; r3=501+r; // r1, r2, r3 - номери елементів R[], до яких
                    // треба додати 1 за модулем 2
        if (R[r3]==0) R[r3]=1; else R[r3]=0; // Додавання 1 за модулем 2 у R[r3]
        }
      else {r1=r; r2=r+3;} // r1, r2 - номери елементів R[], до яких
                 // треба додати 1 за модулем 2
      if (R[r1]==0) R[r1]=1; else R[r1]=0; // Додавання 1 за модулем 2 у R[r1]
      if (R[r2]==0) R[r2]=1; else R[r2]=0; // Додавання 1 за модулем 2 у R[r2]
      }
      else {if (R[r]==0) R[r]=1; else R[r]=0;} // Додавання 1 за модулем 2 у R[r]
    }
  }
} // End of function MULT()
```

Рис.4.1. Програма, що реалізує функцію множення елементів поля Галуа

Розглянутий вище цикл виконується на боці клієнта, після чого кожен раз обчислюється значення випадкового степеню на полем Галуа з виразу [70]

$$A = X^N, \tag{4.1}$$

де X – примітивний елемент поля Галуа;

N – випадкове число, що відповідає випадковій послідовності бітів.

Фрагменти розроблених у даній роботі програм на мові *JavaScript*, що реалізують обчислення за виразом (4.1) та відправлення на сервер результату цього обчислення, представлені на рисунках 4.3 та 4.4 відповідно.

```
var M = new Array(504); // Масив M для таблиці степенів примітивного елементу поля Галуа
for(var i=0; i<504; i++) M[i] = new Array(504); // Всі елементи поля Галуа мають вигляд
// масивів з 503 чисел 1 або 0. M[1] – молодший бит, M[503] – старший біт, M[0] - резерв
var N = [504]; // Масив N для занесення випадкових бітів
var T1 = new Date(); // Поточне значення часу занесли в T1
var TB = T1.getTime(); // TB – кількість мілісекунд від 00:00 01.01.1970
var TD = TB % 2;      // TD = 0 для парного значення TB, TD = 1 для непарного TB
if (TD>0) N[1]=1; else N[1]=0; // Заповнюємо перший випадковий біт N[1]
for (var i=1;i<=503;i++) M[1][i] = 0; // Заносимо нулі в перший рядок таблиці степенів
M[1][2]=1; // на місце другого біту занесли 1. Це й буде примітивний елемент поля Галуа
for (var I=2;I<=503;I++) // Початок циклу заповнення масивів M та N
{ for (var J=1;J<=503;J++)
  M1[J]=M2[J]=M[I-1][J]; MULT(); // Множимо значення попереднього рядку само на себе
 // Кожен рядок таблиці є квадратом попереднього рядку,
 // де M[номер рядка таблиці][номер біта в рядку]
  for (var j=1;j<=503;j++) M[I][j]=R[j]; // Занесли результат множення в черговий рядок
 // Починаємо обчислення чергового випадкового біту
  T1 = new Date();  // Поточне значення часу занесли в T1
  TB = T1.getTime(); // TB – кількість мілісекунд від 00:00 01.01.1970
  TD = TB % 2; // TD = 0 для парного значення TB, TD = 1 для непарного TB
  if (TD>0) N[I]=1; else N[I]=0; // Заповнюємо черговий випадковий біт N[I]
  var NN = Math.random(); // NN – випадкове число від 0 до 1
  if (NN>0.5) N[I]=N[I]+1; if (N[I]>1) N[I]=0; // Остаточно заповнили біт N[I]
} // Кінец циклу заповнення масиву степенями (1, 2, 4, 8, 16,...) примітивного елементу
 // поля Галуа та масиву N чисто випадковими бітами
```

Рис.4.2. Програма, що реалізує цикл заповнення таблиці степенів примітивного елементу поля Галуа та масиву випадкових бітів

```
var A = [504]; // A[] - результат випадкової степені примітивного елементу
 // Обчислення A[]=X[]^N[], де X[]-примітивний елемент поля Галуа;
 // N[]- випадкова послідовність бітів
  for (var i=1;i<=503;i++) A[i]=0; A[1]=1; // Занесли в A[] примітивний елемент
  for (var J=1;J<=503;J++) // Починаємо зовнішній цикл обчислення степені
    if (N[J]==1) // Беремо тільки одиниці з N[], бо множення на нуль дає
    {
      for (I=1;I<=503;I++) // Починаємо внутрішній цикл, в якому
      {          // готуємо множники
       M1[I]= M[J][I]; // У M1[] заносимо потрібну степінь примітивного елементу
       M2[I]=A[I];   // У M2[] заносимо результат попереднього множення
      }
      MULT(); // Функція множення
      for (I=1;I<=503;I++)
       A[I]=R[I]; Результат занесли у A[]
    } // Кінець обчислення степені
```

Рис. 4.3. Програма реалізації обчислення степені елементу поля Галуа

```
var TR='/Q0'; // Рядок для відправлення даних на сервер
var RT='/'; // Рядок для занесення відповіді від сервера
var B = [504]; // B[] - випадкова степінь примітивного елементу від сервера
        for (var i=1;i<=503;i++)
        { // Переносимо значення степені A[] в рядок TR для відправлення на сервер
        if (A[i]==0) TR=TR+'0'; else TR=TR+'1';
        }
var xmlhttp = getXmlHttp(); // Асинхронно відправляємо на сервер рядок TR,
xmlhttp.open('GET',TR,true);  // Стандартна функція для обміну даними
xmlhttp.onreadystatechange = function()
       {if (xmlhttp.readyState == 4) // 4 - означає завершення виконання команди
        {if(xmlhttp.status == 200) // 200 - означає успішність виконання
                {RT=xmlhttp.responseText; // Відповідь сервера заносимо у рядок RT
                      if (RT[0]=='E') {alert("Лінію зайнято"); FNEW=1;}
                for (i=1;i<=503;i++) B[i]=RT[i-1]; // Цикл заповнення значення B[]
            }
         else {alert("Лінію зайнято"); FNEW=1;}
        }
       };
       xmlhttp.send(null);
```

Рис.4.4. Програма, що реалізує обмін даними між клієнтом і сервером

Значення *B*, яке отримує клієнт в результаті обміну даними з сервером за допомогою програми, що представлена на рис.4.4, розраховується на боці сервера у той самий спосіб, що і на боці клієнта розраховується значення *A*, а саме з виразу, аналогічному (4.1):

$$B = X^M, \tag{4.2}$$

де X – примітивний елемент поля Галуа;

M – випадкове число, що отримане на сервері.

На сервері діють такі ж самі програми, як і на боці клієнта, що показані на рисунках 4.1, 4.2 і 4.3. Позначення випадкового числа N у виразі (4.2) замінено на M для того, щоб відрізняти одне від одного випадкові числа, що отримані на боці клієнта і на сервері. Для обміну даними з клієнтами на сервері використовується стандартна функція пакету *Node.js*

http.createServer(*function* (*req, res*) { *pp* }).*listen*(*port*);

де прийняті такі позначення:

req – дані, які клієнт відправляє на сервер (запит),

res – дані, які сервер відправляє клієнту (відповідь),

pp – прикладна програма обробки даних,

port – номер порту для обміну даними за протоколом *TCP*.

Після того, як сервер отримає від клієнта значення *A*, а клієнт отримає від сервера значення *B*, кожен з них виконує операцію знаходження степені згідно відомому алгоритму Диффі-Геллмана, а саме на кожному боці обчислюється значення ключа *C* :

$$C = A^M = B^N, \tag{4.3}$$

де A^M – ключ, що обчислюється на сервері;

B^N – ключ, що обчислюється на боці клієнта.

Легко впевнитись в тому, що $A^M = B^N$, бо $A^M = X^{NM}$, а $B^N = X^{MN}$. Таким чином, на кожному боці буде отримано однакове значення ключа *C*, що необхідно і достатньо для реалізації гарантованого захисту даних під час передавання з використанням шифру Вернама.

Для обчислення ключа *C* використовується така сама програма знаходження степені, яку показано на рис.4.3, але замість примітивного елементу *X* там береться значення *A* на сервері і значення *B* на боці клієнта.

Всі дані, що підлягають захисту, на боці передавача перетворюються в послідовність чисел 1 або 0 і додаються за модулем 2 до випадкових бітів ключа *C*. На боці одержувача робиться зворотне перетворення за допомогою тих самих випадкових бітів ключа *C*.

Для компрометації такого методу захисту слід перехопити значення *A* або *B*, які пересилаються у відкритому вигляді, і знайти число *N* або *M*. Така процедура вимагає дискретного логарифмування, що, як відомо, для великих розмірів поля Галуа потребує значних витрат комп'ютерного часу, що вимірюються роками.

Під час експериментів у даній роботі випробовувались різні розміри полів Галуа. Для поля $GF(2^{503})$ процедура обчислення степені віднімає часу близько

секунди, а для поля $GF(2^{743})$ ця процедура віднімає близько 14 секунд на звичайному сучасному комп'ютері. Зрозуміло, що покращення захисту пов'язано з додатковими витратами часу на шифрування. Для кожної конкретної СДГ може бути обрано розмір поля Галуа в залежності від необхідного рівня захисту даних.

Таким чином, отримані результати доводять, що ПЗ СДГ дозволяє досягти під час передавання даних той рівень захисту, який унеможливить протягом певного періоду часу розкрити зміст цих даних через надзвичайну складність розв'язання задачі дискретного логарифмування над полями Галуа великого розміру. Оскільки цей період часу можна обрати більшим за період зацікавленості у такому розкритті, то будь-які спроби даного розкриття, враховуючи його трудомісткість, будуть недоцільними.

4.4. Реалізація методів захисту даних під час їхньої обробки засобами прикладного програмного забезпечення

Оскільки обробка даних прикладним ПЗ, як на боці клієнта, так і на сервері, відбувається без звернення до будь-яких файлів протягом всього періоду обслуговування виборів (див. розділ 2), то, якщо використовувати лише штатні програмні засоби, неможливо розкрити зміст даних, що обробляються, бо в ОС загального користування не передбачено засобів для такого розкриття. Тобто, тільки сама прикладна програма може розкрити значення своїх змінних, а інших шляхів доступу до цих значень не передбачено. Якщо на сервері, завдяки запропонованій технології всебічного контролю, можна гарантувати відсутність позаштатних програмних засобів, то на боці клієнта таких гарантій не існує. Але для клієнта його власні дані і його голос не є таємницею, а наявність будь-яких позаштатних програм на його комп'ютері не надає можливості розкрити голоси і персональні дані інших виборців, бо для кожного сеансу зв'язку з виборцем, як показано в розділі 3, встановлюється окремий захищений канал передачі даних. Тому питання захисту даних на комп'ютерах

клієнтів є справою кожного виборця. Для збереження власних конфіденційних даних, виборці повинні впевнитись у відсутності на тих комп’ютерах, якими вини користуються для голосування, позаштатних програмних засобів. Проаналізуємо докладніше прикладне ПЗ СДГ. Це забезпечення складається з двох основних частин, які відповідають двом періодам функціонування СДГ.

1) Підготовка бази даних (БД) про виборців (або їх реєстрація) для дистанційного голосування, яка проходить між періодами виборів.

2) Обслуговування виборів, до якого належать остаточна перевірка готовності виборців до дистанційного голосування, саме дистанційне голосування та вивід результатів підрахунку голосів.

Кожному із цих двох періодів відповідає окрема частина ПЗ. Зв'язок між цими частинами відбувається виключно через єдиний файл, що являє собою БД учасників дистанційного голосування.

ПЗ для підготовки БД складається з трьох наступних файлів:

SBD.js – програма на мові *JavaScript*, яка забезпечує функціонування сервера БД;

BASE.html – текст на мові *HTML* зі вбудованою клієнтською програмою на мові *JavaScript*, яка підтримує діалог клієнта-реєстратора зі сервером для занесення даних в базу;

VYBORCI.DBT – файл з базою даних за структурою стандарту *dBase*, який є результатом діяльності цього періоду функціонування СДГ.

Усі ці три файли встановлюються на сервері підготовки БД в робочу директорію, де також повинен бути встановлений пакет `node-static`, який входить до складу засобів *Node.js*. Операційна система цього серверу може бути обрана якою завгодно з міркувань зручності та підтримки засобів *Node.js*. Сервер БД активізується за допомогою команди `node` *SBD*. Після цього клієнт-реєстратор БД може встановити зв'язок з сервером із свого комп’ютера через будь-який браузер, вводячи запит за стандартною формою

<ім'я або адреса сервера >[:<номер порта>]/*BASE.html*

У відповідь на цей запит реєстратор БД отримає файл *BASE.html*, з програмою для занесення даних виборця в діалоговому режимі через форму, яку представлено на рисунку 4.5.

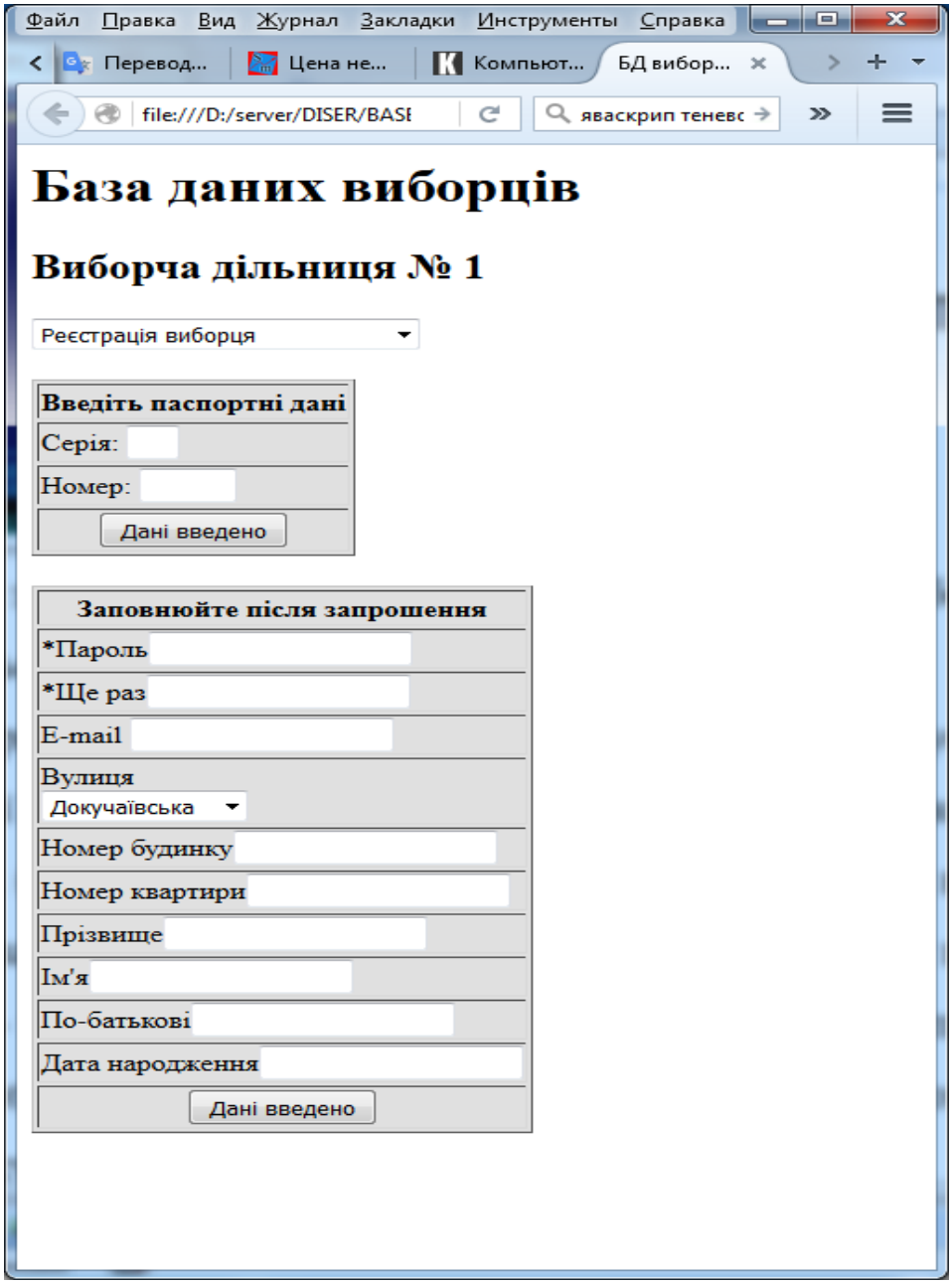

Рис.4.5. Форма для занесення у базу даних про виборця

Обмін даними між реєстратором і сервером БД відбувається за таким же сценарієм, як описаний у підрозділі 4.3. Тому зловмисник, що буде прослуховувати дані під час передавання, не буде мати шансів на отримання інформації про виборців. Для унеможливлення розкриття паролів, вони у файлі

VYBORCI.DBT зберігаються у зашифрованому вигляді. Алгоритм шифрування використано той самий, що і для передавання ключів, який описано виразом (4.1). Фрагмент програми для перетворення паролю за цим алгоритмом представлено на рисунку 4.6.

```
 // Перетворюємо пароль для зберігання у 72-байтовий рядок
 // Знаходимо степінь примітивного елементу поля Галуа,
 // де за показник степені візьмемо пароль (112 біт)
       // Переносимо розшифрований пароль (112 біт) з O[] у N[]
       for (i=1;i<=112;i++) N[i]=O[NC1+i-1+(k-16)*7];
           for (i=113;i<=503;i++) N[i]=0; // решту бітів у N[] заповнюємо нулями
// Заносимо в A[] примітивний елемент поля Галуа
  for (i=1;i<=504;i++) A[i]=0; A[1]=1; // Занесли в A[] примітивний елемент
  for (var J=1;J<=503;J++) // Починаємо зовнішній цикл обчислення степені
    if (N[J]==1) // Беремо тільки одиниці з N[], бо множення на нуль дає
    {
      for (I=1;I<=503;I++) // Починаємо внутрішній цикл, в якому
      {          // готуємо множники
       M1[I]= M[J][I]; // У M1[] заносимо потрібну степінь примітивного елементу
       M2[I]=A[I];   // У M2[] заносимо результат попереднього множення
      }
      MULT(); // Функція множення
      for (I=1;I<=503;I++)
       A[I]=R[I]; Результат занесли у A[]
    } // Кінець обчислення степені
// Зашифрований пароль з A[] переносимо в PW (по 7 біт в байт)
       PW='';
       for (j=0;j<72;j++) // Цикл по байтам PW
       {e=1+j*7;// Обчислюємо номер початкового біту для чергового байту PW
       var w=0; // Числове значення байту
       if (A[e]==1)  w=w+1;
       if (A[e+1]==1) w=w+2;
       if (A[e+2]==1) w=w+4;
       if (A[e+3]==1) w=w+8;
       if (A[e+4]==1) w=w+16;
       if (A[e+5]==1) w=w+32;
       if (A[e+6]==1) w=w+64;
       PW=PW+String.fromCharCode(w);}
// Зашифрований пароль перенесено в рядок PW (72 байти) для зберігання
```

Рис.4.6. Фрагмент програми шифрування пароля для зберігання у БД

Надійність захисту паролів від розкриття під час зберігання у БД пояснюється складністю задачі дискретного логарифмування (так само, як і надійність захисту ключів у підрозділі 4.3). Для неможливості розкриття паролю реєстратором під час введення, клавіатуру передають виборцю, щоб він самостійно двічі ввів свій пароль. При цьому, на екрані всі символи паролів автоматично замінюються однаковими чорними крапками.

Вимоги щодо захисту інформації під час періоду підготовки БД виборців є менш жорсткими, ніж для періоду голосування. Більшість даних цієї бази не є конфіденційними, їх роздруковують з файлу *VYBORCI.DBT* у вигляді списку

для перевірки. Подібні перевірки завжди відбувалися незалежно від методів голосування. Під час перевірок виявляються різні неточності, в тому числі можливі дописування фіктивних виборців.

Захисту від розкриття підлягають тільки паролі виборців. Вище показано, що паролі є захищеними на всіх етапах створення БД, починаючи від введення, і далі, під час передавання до сервера і зберігання у файлі *VYBORCI.DBT*.

Відповідальність за вірність занесення даних в БД несе реєстратор. Для виявлення можливих помилок в його роботі, в кожному рядку БД фіксується дата і час занесення даних. Це надає змогу встановити особу реєстратора, який припустив помилку. Реєстратору надається право запуску і припинення роботи програми *SBD.js* за допомогою стандартних засобів управління сервером в дистанційному режимі. Під час роботи програми *SBD.js* ніякі треті особи не зможуть отримати доступ до сервера БД, бо в програмі *SBD.js* передбачено тільки одноосібний режим роботи. У разі спроби підключення ще одного реєстратора буде видано повідомлення «Сервер зайнятий».

ПЗ періоду обслуговування виборів складається з наступних чотирьох файлів:

SVD.js – програма на мові *JavaScript*, яка забезпечує функціонування сервера виборчої дільниці від початку перевірки готовності виборців для участі у виборах (під час цієї перевірки виборці повинні ввести пароль для голосування) до завершення видачі результатів підрахунку голосів;

CPW.html – текст на мові *HTML* зі вбудованою клієнтською програмою на мові *JavaScript*, яка підтримує діалог виборця зі сервером щодо занесення паролю для голосування;

anketa.html – текст на мові *HTML* зі вбудованою клієнтською програмою на мові *JavaScript*, яка підтримує діалог виборця зі сервером під час голосування і отримання довідок про кількість виборців і результатів голосування;

VYBORCI.DBT – файл з базою даних.

Усі ці чотири файли на сервер виборчої дільниці в директорію *home/admin* у визначений момент часу заносить адміністратор сервера під наглядом виборців-контролерів, як описано у підрозділі 4.2. У цій директорії він також повинен встановити пакети `node-static` (команда `npm install node-static`) та `nodemailer` (команда `npm install nodemailer`), що входять до складу засобів *Node.js*. Після паузи, яка потрібна для проведення перевірки змісту усіх перелічених файлів контролерами, в чітко визначений момент часу адміністратор повинен виконати запуск програми *SVD.js* за допомогою команди `node` *SVD*. Цей момент є початком періоду остаточної перевірки готовності виборців для дистанційного голосування та введення виборцями спеціальних паролів для голосування. Від цього моменту програма *SVD.js* буде автоматично підключати та відключати ті чи інші функціональні можливості щодо обслуговування запитів виборців та дій адміністратора відповідно до заданих періодів часу у годинах, які вказані в початковій частині файлу *SVD.js*, що показано на рис. 4.7.

Першим режимом, що встановлюється одразу після запуску програми *SVD.js*, є режим введення паролів, інтервал часу для якого у фрагменті, що представлений на рисунку 4.7, дорівнює 240 годин (10 діб). В цей період виборці повинні з’являтись до пункту реєстрації з паспортом для засвідчення особи. У цих пунктах їм буде надана можливість встановити зв'язок зі сервером виборчої дільниці. Для цього зв'язку вони можуть використати власні пристрої (смартфони, планшети і т. ін. з Wi-Fi доступом до мережі) або комп’ютери пункту реєстрації.

```
 // SVD Server Vyborchoi Dilnici Ver. 3 June 2015
//////////// Параметри, що потребують налаштування ////////////
var EMAILSERV="gmail"; // Ім'я сервісу для електонної пошти виборчої дільниці
var EMAILNAME="vybir800876@gmail.com"; // Електонна пошта виборчої дільниці
var EMAILPASS="xxxx";// Пароль ел. пошти виборчої дільниці (замінюється адміністратором)
var TCPW=240; // Тривалість періоду введення паролю для голосування виборцями(годин)
var TPAUSE=24;// Тривалість паузи для друку списку голосуючих та завантаження бюлетенів
var TGOL=12; // Тривалість періоду для голосування (годин)
var DILN='№ 800876'; // Назва виборчої дільниці
var KVUL=3; // Кількість вулиць на виборчій дільниці
var VUL = [3]; // Назви вулиць
  VUL[0]='Волгоградська';
  VUL[1]='Івана Неходи';
  VUL[2]='Городня';////////////// Кінець параметрів, що потребують налаштування ////////////
var FLAG_END=0; // Ознака завершення виборів (1-вибори завершено)
var NPERIOD=1; // Встановили номер періоду (1-введення паролів, 2-пауза, 3-голосуваннння)
var NAMEHTML="/CPW.html"; // Назва клієнтської програми (встановлюємо для 1-го періоду)
// NAMEHTML="/anketa.html"; Це буде встановлено для 3-го периоду (NPERIOD=3)
// Підключаємо бібліотеки node.js
var http = require('http');
var url = require('url');
var fs = require('fs');
var static = require('node-static');
var querystring = require('querystring');
var nodemailer = require('nodemailer');
var file = new static.Server('.');
```

Рис.4.7. Початковий фрагмент тексту програми *SVD.js*

Головна мета даного періоду полягає у захисті від спроб фальсифікації осіб виборців. Поки що не існує по-справжньому достовірних дистанційних методів встановлення особи. У разі їх появи можна буде обійтись без особистого прибуття виборців до пунктів реєстрації. Після встановлення зв’язку із сервером виборець повинен ввести пароль для голосування в діалоговому режимі через форму, яку представлено на рисунку 4.8.

Адміністратор в цей період також повинен ввести пароль, але не для голосування, а для доступу до електронної пошти виборчої дільниці. Цей пароль він повинен утримувати в таємниці до моменту фізичного відключення сервера. Необхідність електронної пошти на сервері виборчої дільниці пов’язана з тим, що з цієї поштової адреси сервер автоматично відправляє виборцям числові коди для голосування, а також довідки про кількість і активність виборців і про результати голосування.

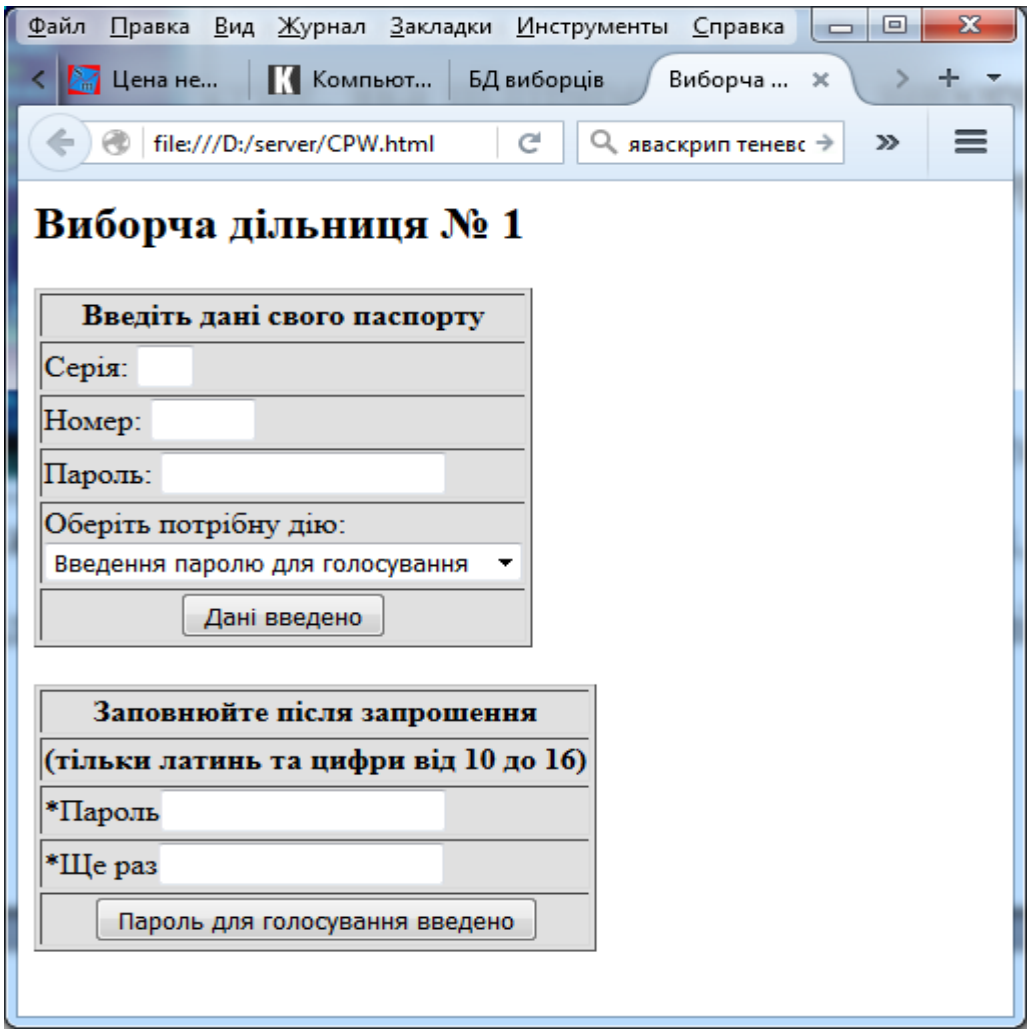

Рис.4.8. Форма для введення паролю для голосування

Враховуючи, що всі файли є відкритими, а знаючи пароль, зловмисники зможуть втручатись в роботу електронної пошти сервера, тому цей пароль не можна вписати в текст програми, як зроблено, наприклад, з назвами вулиць. Для введення цього паролю за адміністратором закріплено неіснуючий номер паспорту 000000 з пропусками замість серії. За допомогою цього номеру він може утворювати захищений канал с сервером, як і звичайний виборець, але без права голосу.

Захист паролів для голосування в сервері забезпечується завдяки тому, що ніякого зовнішнього доступу до символьного масиву, де зберігаються ці паролі, не існує. Значення цих паролів відоме виключно їх власникам і програмі, що безперервно буде діяти на сервері до моменту фізичного

відключення. Ці паролі можна використати тільки один раз для голосування, після чого вони повністю втрачають свою цінність.

Тривалість паузи між періодом введення паролів і голосуванням у фрагменті програми, що показаний на рис. 4.7, встановлена 24 години. Ця пауза потрібна для того, щоб співробітники виборчої дільниці встигли викреслити чи помітити виборців, що голосують дистанційно, у загальних дільничних списках. Оскільки кількість виборців на дільниці в Україні може бути до 2500, то така процедура потребує деяких витрат часу.

Для забезпечення захисту від підробки списку виборців, які ввели пароль для голосування, цей список формується серверною програмою у вигляді файлу *REGISTR.TXT* і розміщується в директорії *home/admin,* що постійно контролюється виборцями. У файлі *REGISTR.TXT* знаходиться один текстовий рядок з одиниць і нулів, де одиницями позначено отримання права на дистанційне голосування. Кожен символ цього рядку відповідає рядку БД з файлу *VYBORCI.DBT*. За допомогою файлів *REGISTR.TXT* і *VYBORCI.DBT,* використовуючи відомі стандартні засоби, наприклад *Microsoft Excel*, можна отримати і проконтролювати даний список. Зауважимо, що через відсутність виборця у списку, він зможе проголосувати двічі, скориставшись ще й паперовим бюлетенем. Такі випадки були зафіксовані в Естонії.

Період голосування є найбільш відповідальним і захищеним від загроз, які б сприяли розкриттю таємниці голосування або спотворенню результатів волевиявлення.

Для здійснення процедури дистанційного голосування, виборець повинен зі свого браузера відправити запит на сервер виборчої дільниці за стандартною формою:

<ім'я або адреса сервера >[:<номер порта>]/ *anketa.html*

У відповідь на цей запит він отримає файл *anketa.html* з програмою для голосування. При цьому на екрані з’явиться форма схожа за типом на таку, як представлена на рис.4.9.

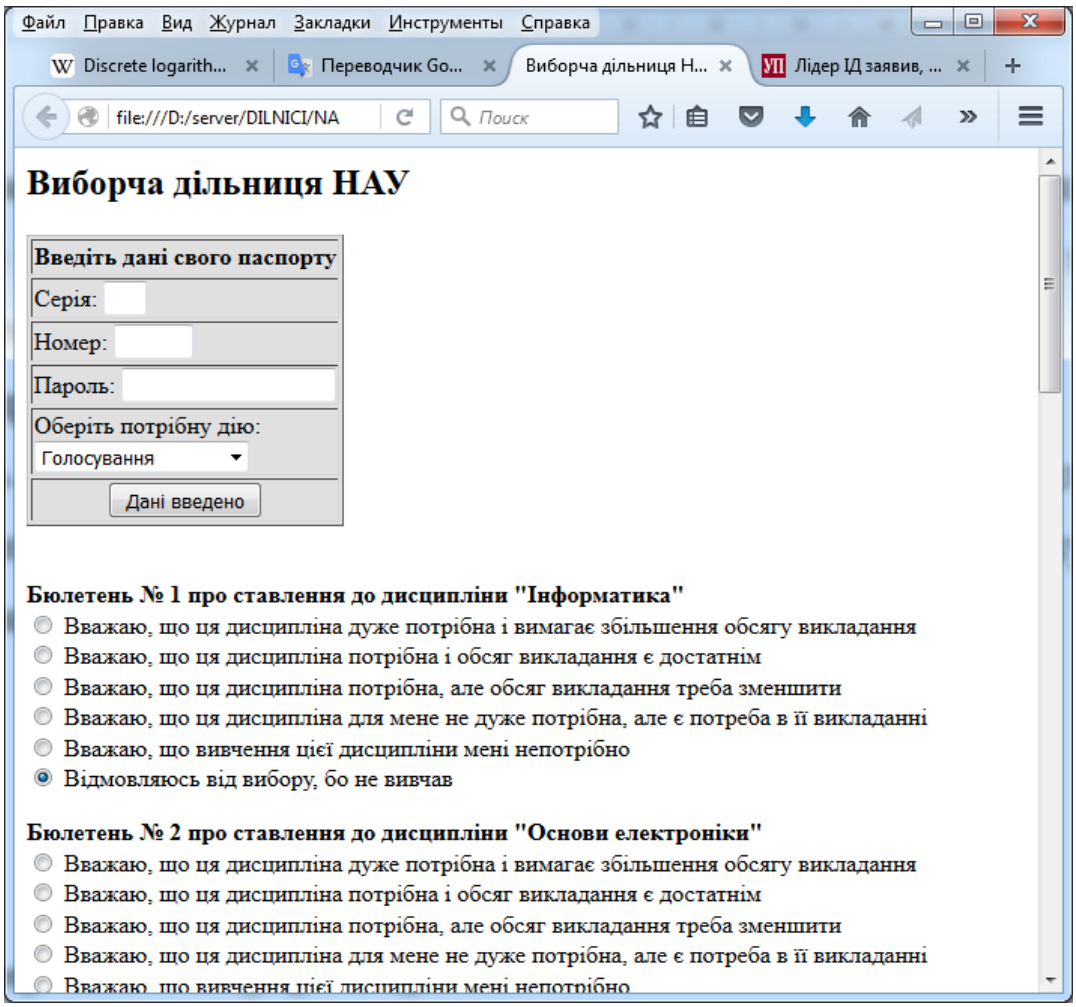

Рис.4.9. Форма для голосування під час експериментального опитування студентів Національного авіаційного університету

Для спрощення процедури доступу виборців до сервера своєї виборчої дільниці під час експерименту було створено допоміжний ресурс, який дозволив виборцям потрапити на сервер для голосування через єдиний сайт за допомогою меню, що представлено на рис.4.10.

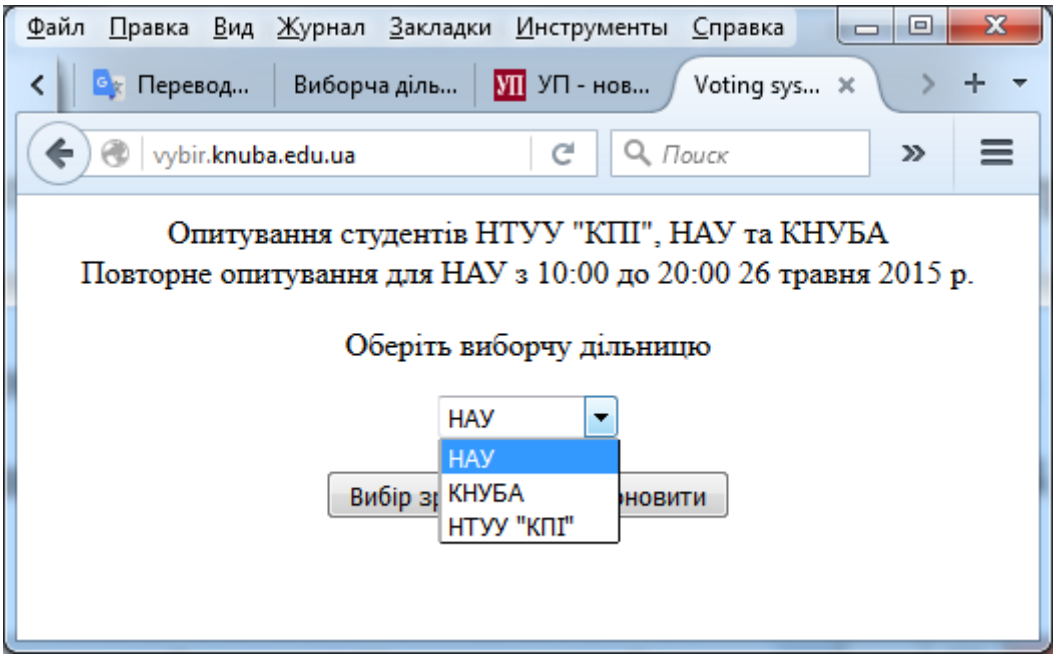

Рис. 4.10. Допоміжне меню для доступу на сервер виборчої дільниці

Надамо короткий перелік методів захисту даних на етапі голосування, які були докладно описані вище у даному розділі.

По-перше, одразу після звернення виборця до сервера виборчої дільниці автоматично утворюється надійно захищений від витоку інформації канал зв’язку між клієнтом і сервером.

По-друге, неможливо дізнатись пароль для голосування, оскільки він зберігається в оперативній пам’яті діючої програми на сервері. Доступ до цієї пам’яті має виключно сама програма.

По-третє, процес підрахунку голосів виборців відбувається виключно в оперативній пам’яті діючої програми на сервері, тому втрутитись у цей процес неможливо.

Крім цих методів захисту, з метою підвищення рівня впевненості в тому, що голосує дійсно сам виборець, а не інша особа, яка якимось чином дізналась пароль, наприклад, підглянувши записи виборця, сервер відправляє виборцю на електронну пошту або на мобільний телефон числовий код з шести випадкових цифр у формі, що представлена на рис. 4.11.

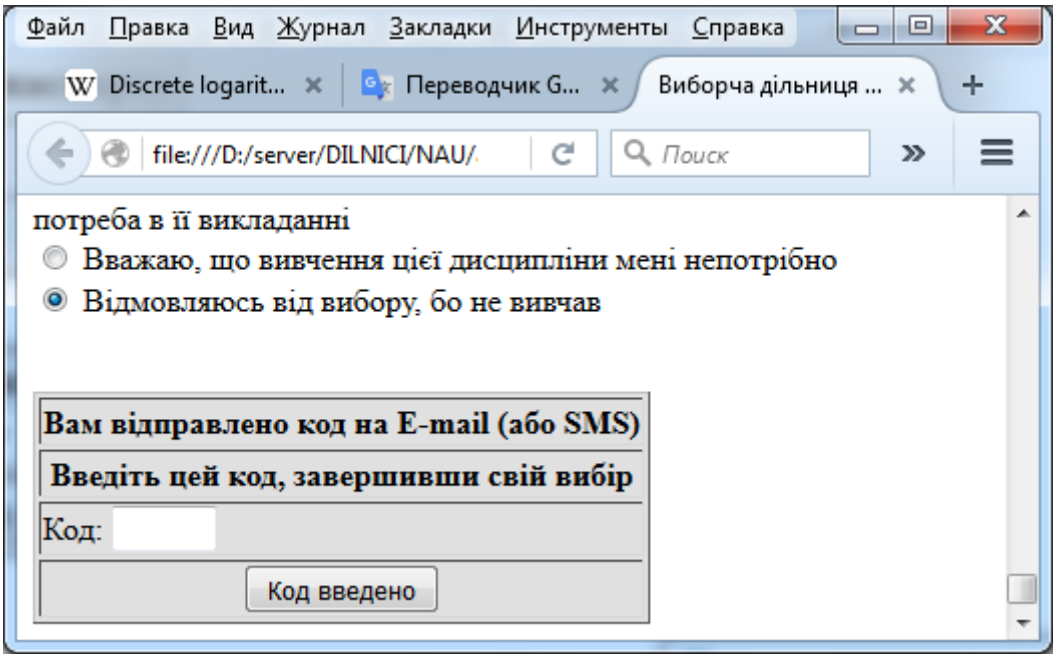

Рис.4.11. Форма для введення коду після завершення процедури вибору

Голос виборця зараховується тільки після перевірки вірності коду. Для впевненості у зарахуванні голосу, виборцю відправляється повідомлення про зарахування його голосу. Варіанти повідомлень показані на рис.4.12.

```
switch (RT[1])
      {
      case '0': alert("Ваш голос прийнято і враховано."); break;
      case '1': alert("Голос відхилено через помилку коду."); break;
      case '2': alert("Ваше право голосу вже використано."); break;
      case '4': alert("Відведений час вичерпано."); break;
      }
```

Рис.4.12. Фрагмент програми, що інформує виборця про результат його голосування

Головний принцип захисту інформації під час обробки прикладним ПЗ базується на тому, що доступ до оперативних даних діючої програми може здійснюватись виключно через інтерфейси цієї програми. Ніякої іншої можливості доступу до цих даних не існує. Оскільки все без винятку ПЗ сервера є відкритим і всі дії, щодо управління сервером контролюються, то будь-яку підміну або модифікацію програмних засобів реалізувати неможливо.

Відомо, що всі комп'ютерні програми можуть виконувати тільки ті дії, які в них запрограмовані. І у тому випадку, коли програма безперервно працює і неможливо контролювати стан її внутрішніх оперативних даних, не підлягає сумніву, що програма ніяких не запрограмованих дій виконувати не буде. В нашому випадку всі можливі дії програмних засобів легко передбачити завдяки їх простоті і відкритості. На цьому базується можливість повної довіри до даної СДГ.

4.5. Метод забезпечення доступності засобів дистанційного волевиявлення

4.5.1. Задачі та критерії протидії загрозам доступності

Важливою вимогою до розробки СДВ є забезпечення зручності у користуванні для виборців. Зафіксовані випадки, відмови від електронного голосування через складність інтерфейсу [20]. Хоч головним фактором, який стримує процес впровадження засобів дистанційного голосування, вважається недовіра суспільства, але на другому місці виступає недостатня зручність у користуванні. Зрозуміло, що є і третій фактор, який неможливо подолати технічними засобами, – це низький рівень розвитку демократичної свідомості. Саме цей третій фактор і є причиною появи більшості тих загроз, боротьбі з якими присвячено дану роботу.

Методи, які запропоновані для СДВ у даній роботі, дозволяє позбутись можливості порушень цілісності даних на всіх етапах виборчого процесу, починаючи від голосування і підрахунку голосів в межах виборчої дільниці, і до отримання загального результату у вигляді підсумку по будь-якій кількості дільниць в залежності від масштабу виборів. Або, іншими словами, унеможливлюють фальсифікацію результатів волевиявлення на всіх рівнях виборчого процесу. Запропоновані в роботі методи роблять неможливим розкриття конфіденційних даних виборців. Але важливо також створити умови, щоб якомога більше виборців мали можливість користуватись засобами СДВ.

Треба зробити дистанційне голосування доступним для виборців. З технічної точки зору критерії доступності передбачають наступне [52]:

- забезпечення доступу користувачів до потрібних інформаційних ресурсів без зайвих ускладнень;

- мінімізація відмов в обслуговуванні;

- швидке відновлення роботи після збоїв;

- гаряча заміна обладнання у разі виходу з ладу.

Зрозуміло, що не можна, з метою покращення доступності, зменшувати рівень якості забезпечення цілісності і конфіденційності інформації. Тому неможливо позбутись необхідності відповідального ставлення виборців до паролів, які слід не забувати і зберігати у таємниці. Це є дійсно тим єдиним ускладненням у порівнянні із традиційним методом голосування. Паролів неможливо позбутись через вимоги збереження конфіденційності. Але, дивлячись на сучасні темпи зростання кількості користувачів Інтернету, де до паролів поступово звикають, можна цей недолік вважати несуттєвим.

Розглянемо методи покращення доступності виборців до необхідних інформаційних ресурсів в процесі дистанційного голосування і контролю за діями персоналу для виявлення можливих порушень інформаційної безпеки.

4.5.2. Забезпечення доступу до потрібних інформаційних ресурсів

Забезпечення зручного доступу до сервера виборчої дільниці полягає у зменшенні зусиль, які виборцю треба витрачати для пошуку своєї серед десятків тисяч інших дільниць. Для цього можна використати відомий метод пошуку на фрагментах карти місцевості. Через велику кількість можливих варіантів, пошук краще розподілити на 2-3 етапи з різними масштабами карти. Останній етап зручно буде реалізувати на сервері формування загального результату голосування, де має місце таблиця адрес серверів виборчих дільниць (див. рис. 2.6), дані з якої і є результатом або кінцевою точкою даного пошуку. Альтернативою цього варіанту пошуку може бути пошук через назву міста або селища і вулиці, як це вже реалізовано на сайті Державного реєстру виборців за адресою *https://www.drv.gov.ua/portal/!cm_core.cm_index*.

4.5.3. Мінімізація затримки на очікування відповідей від сервера

Щоб мінімізувати затримку на очікування відповідей від сервера виборчої дільниці на запити виборців під час голосування, в програмі сервера виборчої дільниці *SVD.js* передбачено можливість підтримки діалогу зі 100 виборцями одночасно. Таку кількість виборців для одночасного обслуговування обрано виходячи з експериментальної оцінки відношення сумарної тривалості пауз P_i, які виникають між інтервалами τ_j, що відповідають часу обробки запитів від одного виборця, до суми цих інтервалів, а саме

$$K = \frac{\sum_{i=1}^{w} P_i}{\sum_{j=1}^{v} \tau_j}, \tag{4.4}$$

де K – оцінка кількості виборців, які можуть одночасно обслуговуватись сервером виборчої дільниці;

w – кількість пауз між періодами обслуговування запитів від одного виборця;

v – кількість інтервалів часу на обслуговування запитів одного виборця.

Часову діаграму процесу обслуговування виборця сервером виборчої дільниці показано на рис. 4.13.

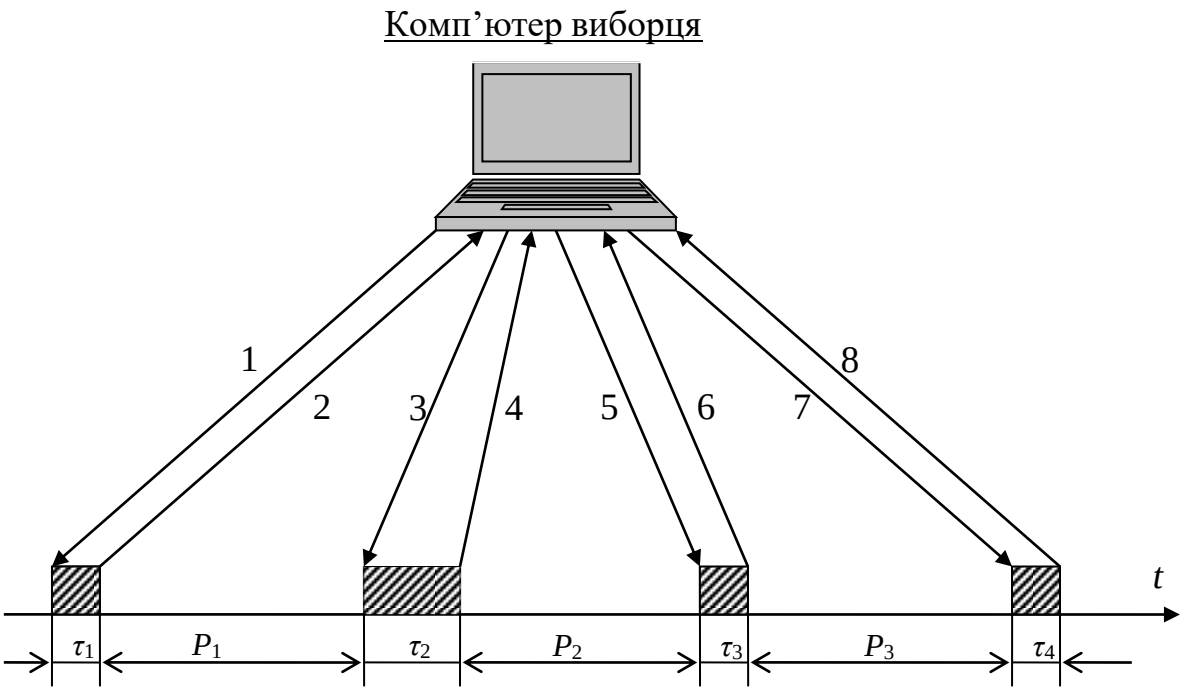

Рис. 4.13. Часова діаграма процесу обслуговування виборця під час голосування

Номери запитів від комп’ютера виборця і номери відповідей сервера, відповідають номерам дій, які описані у протоколі обміну даними між виборцем і сервером виборчої дільниці у розділі 3 (див. рис. 3.6). Через те, що сумарна тривалість пауз під час обслуговування запитів виборця приблизно у 100 разів більша за суму періодів обробки цих запитів, а серверна програма у періоди пауз може обслуговувати запити від інших виборців (до 99), то час очікування відповіді сервера у переважній більшості випадків не перевищує 2-3 секунди.

4.6. Реалізація захисту прав виборців на власне волевиявлення за умов підкупу або примушування

Загроза підкупу або примушування виборців призводить до того, що замість власного рішення вони підтримують чуже, яке не співпадає з їх власним. Такі випадки здатні вплинути на результат виборів, спотворюючи

справжність виявлення суспільної думки, що заважає процесу нормального розвитку демократичного суспільства. Досвід вивчення цієї загрози показує, що зловмисник кожного разу відшукує якусь можливість дізнатись про те, як проголосував виборець. Для цього виборці вкидали вже заповнені бюлетені, отримані від зловмисників, а свої порожні повертали за винагороду. Інші виборці, на вимогу зловмисника, фотографували бюлетені після заповнення.

В умовах дистанційного голосування існують дві наступні можливості дізнання про те, як проголосував виборець.

По-перше, спостереження за діями виборця в процесі голосування.

По-друге, дізнавшись у виборця його ідентифікаційні дані і пароль, проголосувати замість нього.

Для захисту від даної загрози в умовах дистанційного голосування треба побудувати систему так, щоб зловмисник не міг дізнатись справжній результат голосування. Відомі методи боротьби з даною загрозою, що описані, наприклад, в [71], полягають в тому, щоб виборець мав два варіанти доступу до сервера, один з яких – справжній, а другий для імітації процесу голосування. Але, у разі повної відкритості програмного забезпечення, всі особливості побудови системи захисту будуть відомі зловмиснику і він може вимагати від виборця проголосувати з обома варіантами доступу. Для того, щоб зловмиснику були невідомі засоби захисту, необхідно деяку частину програмного забезпечення зробити закритою, але навіть часткова закритість програмного коду, є цілком зрозумілою підставою для недовіри усій системі, а втрата довіри виборців є абсолютно неприйнятною.

Вважаючи, що у повністю відкритій системі неможливо створити такий секретний канал зв’язку між сервером і виборцем, про який зловмисник не міг би дізнатись, з метою надійного захисту від даної загрози ми вимушені обмежити виборця в можливості отримання інформації про голосування.

Під час даного експериментального дослідження було перевірено два варіанти протидії загрозам підкупу або примушування виборців. За основу

побудови системи захисту в обох варіантах прийнято той факт, що виборець знає пароль для голосування, а зловмисник – не знає.

Першій варіант захисту, який надає можливість проголосувати під наглядом зловмисника без можливості розкриття факту імітації, полягає в заміні повідомлення "Ваше право голосу вже використано" на повідомлення "Ваш голос прийнято і враховано." При цьому сервер налаштовується таким чином, щоб голос зараховувався тільки з вірним паролем і тільки один раз. Заміна правильного повідомлення на дезінформацію може ввести в оману не тільки зловмисника, але й самого виборця, якщо в нього є сумніви у вірності свого паролю. Те, що виборець через цю дезінформацію повинен буде з більшою відповідальністю ставитись до процедури введення паролю, є розплатою за захист від підкупу. Повідомлення про невірний пароль відправляється тільки у разі невірної довжини пароля, а в інших випадках дозволяється проголосувати, після чого відправляється код повідомлення "Ваш голос прийнято і враховано".

Під час більш глибокого експериментального дослідження першого варіанту захисту, виходячи з того, що зловмисник може мати високий рівень кваліфікованості і необмежені можливості впливу на виборця було виявлено наступні слабкі місця.

1) Неможливо віддати зловмиснику ідентифікаційні дані і пароль, щоб він самостійно проголосував замість виборця, бо тоді треба ще й передати пароль до електронної пошти для отримання числового коду. Заволодівши доступом до електронної пошти, зловмисник може змінити пароль і тоді виборець не зможе голосувати.

2) Зловмисник може дізнатись про факт голосування за допомогою довідки про активність виборців. Якщо в довідці вказано, що виборець вже проголосував, то зловмисник одразу розкриє спробу імітації. У разі ж відсутності активності виборця, зловмисник може обрати момент голосування близьким до завершення виборів і прослідкувати, щоб виборець більше не голосував.

Для ліквідації першого з цих двох слабких місць можна відмовитись від відправлення цифрового коду на електронну пошту або через СМС. Таке спрощення процедури голосування послаблює захист від крадіжки пароля і через це вимагає від виборця більшої відповідальності щодо збереження таємниці пароля. Методів розкриття паролю для голосування без участі виборця в умовах обмеження в часі не існує. Точніше, для цього треба знайти дискретний логарифм над елементами поля Галуа $GF(2^{503})$, що потребує багато часу і комп'ютерних ресурсів.

Друге слабке місце ліквідується шляхом заборони надавати у довідках інформацію про активність виборців до закінчення виборів, залишивши у довідках тільки ознаку про введення паролю для голосування. При цьому активністю слід вважати не тільки дійсне голосування, але і його імітацію.

Таким чином, за рахунок обмеження надання інформації виборцям під час голосування, та користуючись дезінформацією, у системі з повністю відкритим програмним забезпеченням можуть бути створені умови, за яких зловмисник не зможе відрізнити справжнє голосування від імітації, що призводить до недоцільності підкупу або примусу виборців.

Зрозуміло, що подібні загрози пов'язані з бідністю та низьким рівнем суспільної свідомості громадян, тому найкращими методами їх подолання є підвищення рівня життя та сприяння розвитку свідомості членів суспільства.

Висновки до четвертого розділу

4.1. Для практичного підтвердження можливостей досягнення результатів, що отримані в роботі, проведено натурне моделювання усіх засобів СДВ, у т.ч. засобів захисту інформації, в умовах, що імітують процес загальнонаціональних виборів в Україні. Задачею досліджень є доведення того, що комплекс програмно-технічних засобів СДВ, створений на основі запропонованих в роботі методів, здатен протистояти загрозам інформації, реалізація котрих викликає недовіру громадян до СДВ.

4.2. Показана функціональність обраної ОС, мов та засобів програмування, яка забезпечує повну контрольованість середовища функціонування СДВ. Забезпечено блокування усіх команд, які можна було б використати для утворення загроз. На сервері СДВ не має можливостей створювати і корегувати будь-які файли. Усі файли ПЗ потрапляють на сервер тільки через єдиний інтерфейс під контролем громадських спостерігачів. Кожен файл при цьому перевіряється шляхом порівняння із заздалегідь перевіреним еталоном. Через такі обмеження, створювати ПЗ безпосередньо на самому сервері не є можливим.

4.3. Розкрито програмні механізми реалізації розробленого протоколу обміну даними між клієнтами і сервером СДВ. Показано, яким чином у СДВ побудовано захищений канал обміну даними. Описано механізм генерування дійсно випадкових послідовностей бітів для кожного сеансу зв’язку, а також механізм програмної реалізації криптографічного алгоритму Диффі-Геллмана. Надано відповідні фрагменти розроблених комп’ютерних програм на мові *JavaScript*.

4.4. Розкрито програмні механізми реалізації розроблених методів захисту даних під час їхньої обробки засобами прикладного ПЗ. Показано, що доступ до оперативних даних діючої прикладної програми може здійснюватись виключно через інтерфейси цієї програми. Ніякої іншої можливості доступу до цих даних не існує. Неможливо дізнатись пароль для голосування, оскільки він зберігається в оперативній пам’яті діючої програми на сервері. Доступ до цієї пам’яті має виключно сама програма. Процес підрахунку голосів виборців відбувається виключно в оперативній пам’яті діючої програми на сервері, тому втрутитись у цей процес неможливо.

Оскільки все без винятку ПЗ сервера є відкритим і всі дії, щодо управління сервером контролюються, то будь-яку підміну або модифікацію програмних засобів реалізувати неможливо.

Одразу після звернення виборця до сервера виборчої дільниці автоматично утворюється надійно захищений від витоку інформації канал зв’язку між клієнтом і сервером.

З метою підвищення рівня впевненості в тому, що голосує дійсно сам виборець, а не інша особа, яка якимось чином дізналась пароль, наприклад, підглянувши записи виборця, сервер відправляє виборцю на електронну пошту або на мобільний телефон числовий код з шести випадкових цифр.

4.5. З метою мінімізації затримки очікування виборцями відповідей від сервера виборчої дільниці передбачена можливість підтримки діалогу зі 100 виборцями одночасно. Таку кількість виборців для одночасного обслуговування обрано виходячи з експериментальної оцінки відношення сумарної тривалості пауз, які виникають між інтервалами, що відповідають часу обробки запитів від одного виборця, до суми цих інтервалів.

4.6. Розкрито механізми реалізації методу захисту прав виборців на власне волевиявлення за умов підкупу або примушування. Під час даного експериментального дослідження було перевірено два варіанти протидії загрозам підкупу або примушування виборців. За основу побудови системи захисту в обох варіантах прийнято той факт, що виборець знає пароль для голосування, а зловмисник – не знає. Аналіз цих варіантів показав, що за рахунок обмеження надання довідкової інформації виборцям (з боку сервера) під час голосування, та користування певними прийомами дезінформації зловмисників, у системі з повністю відкритим ПЗ можуть бути створені умови, за яких зловмисник не зможе відрізнити справжнє голосування від імітації, що призводить до недоцільності підкупу або примусу виборців.

ОСНОВНІ РЕЗУЛЬТАТИ ТА ВИСНОВКИ

У дисертаційній роботі, відповідно до поставленої мети, розв’язано актуальну науково-технічну задачу гарантування неможливості виникнення порушень цілісності результатів та конфіденційності персональних даних в системах дистанційного волевиявлення, що усуває будь-які підстави для недовіри з боку голосуючих щодо можливості реалізації вказаних порушень.

У процесі виконання дисертаційної роботи отримані такі основні результати:

1. Здійснено аналіз характеристик існуючих СДВ. Показано, що основний недолік цих систем - відсутність дієвих механізмів контролю функціонування СДВ із боку суспільства. Через це не може буде забезпечена довіра людей щодо збереження таємниці голосів та результатів волевиявлення. Виявлено «слабкі місця» у захисті існуючих СДВ, зокрема виявлено загрози, відсутність протидії яким підриває довіру громадян до СДВ. Визначено профіль захищеності інформації, що забезпечує беззаперечні гарантії неможливості порушень таємниці голосів та результатів волевиявлення.

2. Побудовано модель захищеної СДВ у вигляді кореспондованої сукупності концептуальної моделі захисту інформації в СДВ та моделі взаємодії користувачів із сервером СДВ. Згідно з цією моделлю за допомогою засобів ОС сервера здійснюється логічна ізоляція процесів прикладної програми, досконало стійкий захист даних виборців в каналі доступу до критичних даних прикладної програми та безперервний контроль усіх ресурсів сервера (файлів та процесів) з боку необмеженого кола осіб. Модель реалізує визначений профіль захищеності засобами типового комп’ютерного обладнання з гарантованістю на рівні Г7 в умовах повної недовіри до всіх без винятків учасників процесу волевиявлення.

3. Запропоновано метод дистанційного спостереження за файлами, процесами, подіями на сервері СДВ з метою протидії загрозам штатній роботі сервера СДВ. Метод базується на спостереженні за станом активних процесів

(діючих програм) і всіх файлів на сервері з боку необмеженої кількості будь-яких користувачів Інтернету за умов повної відкритості ПЗ і наперед відомого розкладу дій адміністратора. Даний метод унеможливлює створення непомічених загроз на сервері.

4. Розроблено метод досконало захищеного обміну даними через Інтернет між клієнтами та сервером ДВ, що забезпечує стійкий захист даних від порушень конфіденційності та цілісності. Доведено, що для забезпечення гарантованої захищеності обміну даними через Інтернет доцільно використати досконало стійкий шифр Вернама. Визначено умови забезпечення режиму досконалої стійкості цього шифру. Для захисту ключової інформації застосовано алгоритм Диффі-Геллмана. Для реалізації алгоритму обрано мультиплікативну групу над полями Галуа $GF(2^n)$, де n – безпечне просте число (*SPN*), що може дорівнювати числам 503, 563, 587, 719, які відповідають виразу $2p+1$, де p - просте число.

5. Розроблено метод отримання випадкових бітових послідовностей. Для отримання дійсно випадкових бітових послідовностей запропоновано використати: 1) випадковий та статистично не прогнозований характер потоку запитів до комп’ютера, що включений у мережу Інтернет; 2) випадковий характер нестабільності частоти кварцових резонаторів, що є у складі комп’ютера. Метод дозволяє забезпечити: 1) коректне функціонування досконало стійкої системи захисту, що реалізована з використанням шифру Вернама; 2) коректну реалізацію алгоритму Диффі-Геллмана. Реалізація методу може бути здійснена засобами типового комп’ютера, що важливо для широкого впровадження СДВ.

6. Запропоновано метод протидії загрозам, що пов’язані із силовим або психологічним впливом на виборців. Сутність цього методу полягає у наданні технічної можливості виборцям у разі необхідності імітувати процес волевиявлення та приховувати інформацію, яка потрібна зловмиснику, що робить недоцільною реалізацію подібних загроз.

7. Проведено комп’ютерне моделювання роботи СДВ згідно із

запропонованою моделлю, побудованою з використанням розроблених методів ТЗІ. Показано, що за підтримки одночасного діалогу із 100 користувачами час очікування відповіді сервера не перевищує 2-3 секунди.

8. Надана оцінка часу очікування обслуговування запитів клієнтів до сервера. Показано, що збільшення інтервалу обслуговування клієнта на величину більшу ніж 6с , призводить до того, що кожний п’ятий клієнт буде очікувати своєї черги до сервера від 7с до 14с, а кожний сімнадцятий – від 14с до 21с.

9. Представлені результати забезпечують досягнення мети дослідження - забезпечити беззаперечні гарантії неможливості виникнення порушень конфіденційності персональних даних користувачів та результатів їхньої обробки в СДВ в умовах повної недовіри до всіх без винятку учасників процесу волевиявлення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Lombardi E. Electronic Vote & Democracy. URL: http://www.electronic-vote.org (дата звернення: 15.11.2017).

2. Schneier B. Unusual Electronic Voting Machine Threat Model. URL: https://www.schneier.com/blog/archives/2014/05/unusual_electro.html (дата звернення: 10.01.2017).

3. Lessons from the EVOTE 2014 International Conference, November 27, 2014. URL: http://e-lected.blogspot.com (дата звернення: 15.11.2017).

4. Schneier B. What's Wrong With Electronic Voting Machines? URL: https://www.schneier.com/essays/archives/2004/11/whats_wrong_with_ele.html (дата звернення: 10.01.2017).

5. Міжнародний пакт про громадянські і політичні права (ратифіковано Указом Президії Верховної Ради Української РСР від 19.10.73 № 2148-VIII).

6. Закон України «Про вибори Президента України». *Відомості Верховної Ради України*. 1999. № 14. Ст. 81.

7. Закон України «Про вибори народних депутатів України», *Відомості Верховної Ради України*. 2012. № 10-11. Ст.73.

8. Закон України «Про місцеві вибори». *Відомості Верховної Ради України*. 2015. № 37-38. Ст.366.

9. Henman P., Ackland R., Graham T. Community Structure in e-Government Hyperlink Networks. *14th European Conference on e-Government (ECEG)*. Brasov (Romania), 2014. Pp. 135 – 143.

10. Kalvet1 T. Innovation: a factor explaining e-government success in Estonia. *Electronic Government*: International Journal, 2012, Vol. 9, No. 2, pp. 142 – 157.

11. Margolis M., Moreno-Riaño G. The Prospect of Internet Democracy. *Surrey, UK*: Ashgate Publishing Company Brookfield (USA), 2009. 200 p.

12. Springall D., Finkenauer T., Durumeric Z. Security Analysis of the Estonian Internet Voting System. *Proceedings of the ACM SIGSAC* Conference on Computer and Communications Security (CCS '14), 2014, pp. 703 – 715.

13. Segaard S.B., Christensen D.A., Jo Saglie B.F. Internettvalg: Hva gjor og mener velgerne? Rapport 2014:07. Oslo, Institutt for samfunnsforskning, 2014. 146 s.

14. The future of Internet voting in the US, October 22, 2015. URL: http://e-lected.blogspot.com (дата звернення: 15.11.2017).

15. Smith C.M. Convenience Voting and Technology: The Case of Military and Overseas Voters. New York, Palgrave Macmillan, 2014. 240 p.

16. Ali Mohammed A., Timour R.A. Efficient E-voting Android Based System. International Journal of Advanced Research in Computer Science and Software Engineering, 2013, Vol. 3, Issue 11. Pp. 43 – 48.

17. Acemyan C.Z., Kortum P., Byrne M.D., Wallach D.S. Usability of Voter Verifiable, End-to-end Voting Systems: Baseline Data for Helios, Prêt à Voter, and Scantegrity II. USENIX Journal of Election Technology and Systems (JETS), 2014, Vol. 2, No 3, pp. 26 – 56.

18. Smith C.M. Convenience Voting and Technology: The Case of Military and Overseas Voters. New York, Palgrave Macmillan, 2014. 240 p.

19. Springall D., Finkenauer T., Durumeric Z. Security Analysis of the Estonian Internet Voting System. Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS '14), 2014, pp. 703 – 715.

20. Acemyan C.Z., Kortum P., Byrne M.D., Wallach D.S. Usability of Voter Verifiable, End-to-end Voting Systems: Baseline Data for Helios, Prêt à Voter, and Scantegrity II. USENIX Journal of Election Technology and Systems (JETS), 2014, Vol. 2, No 3, pp. 26 – 56.

21. Buchmann J., Neumann S., Volkamer M. Tauglichkeit von Common Criteria-Schutzprofilen für Internetwahlen in Deutschland. Datenschutz und

Datensicherheit-DuD. Springer Fachmedien Wiesbaden, 2014, Vol. 38, Issue 2, pp. 98 – 102.

22. Ali Mohammed A., Timour R.A. Efficient E-voting Android Based System. International Journal of Advanced Research in Computer Science and Software Engineering, 2013, Vol. 3, Issue 11, Pp. 43 – 48.

23. Модуль інтернет-голосування. URL: https://arma.gov.ua/modul-inet-vote (дата звернення: 05.10.2017).

24. Спосіб електронного голосування: пат. 39582 Україна: МПК G07C 13/00. № 200900174; заявл. 10.01.2009; опубл. 25.02.2009, Бюл. № 4, 4 с.

25. Вибори до Верховної Ради: 5 інтернет-проектів у допомогу виборцям. URL: http://basicgroup.ua/vybory-do-verkhovnoi-rady-5-internet-proektiv-u-dopomogu-vybortsiam (дата звернення: 05.10.2017).

26. Інтерактивна карта виборчих порушень – статистика і узагальнення тенденцій. URL: http://maidanua.org/vybory2012/stats/ (дата звернення: 15.11.2017).

27. У Рівному презентовано новий проект Регламенту міської ради URL: http://opora.rv.ua/novini/1399-u-rivnomu-prezentuvaly-proekt-rehlamentu-miskoi-rady-2.html. (дата звернення: 05.10.2017).

28. Моніторинг виборів до місцевих рад URL: http://tusovka.kr.ua/news/2015/09/23/monitoring-viboriv-do-mistsevih-rad-gromada-na-storozhi-zakonu (дата звернення: 05.10.2017).

29. eCast – супроводження дня голосування URL: http://basicgroup.ua/ecast (дата звернення: 05.10.2017).

30. Електронні петиції. Офіційне Інтернет-представництво Президента України URL: https://petition.preside (дата звернення: 05.10.2017).

31. Інтернет-вибори Президента України зірвано URL: http://politiko.ua/blogpost17340 (дата звернення: 05.10.2017).

32. Спосіб електронного голосування: пат. 91920 Україна: МПК G07C 13/00. № 200900175; заявл. 10.01.2009; опубл. 10.09.2010, Бюл. № 17, 4 с.

33. Гірник Д.А., Сергієнко О.В. Створення АС комп'ютерної бази даних єдиного реєстру документів дозвільного та декларативного характеру в будівництві. *Новітні комп'ютерні технології.* Випуск XI. ДВНЗ "Криворізький національний університет", 2013. С. 155 -158.

34. Вовк А.І., Гірник Д.А., Підлужня В.А. Концепція автоматизованої системи аудиту та моніторингу енергоефективності будівель. *Інформаційно-комп'ютерні технології:* зб. тез допов. VII Міжнародна науково-технічна конференція. Житомир, МОН України, 2014. С. 13-14.

35. Neumann P.G. Security Criteria for Electronic Voting. Proceedings of National Computer Security Conference, 1993 (16th), Baltimore (USA). DIANE Publishing, 1995, p.p. 478 – 482.

36. Buchmann J., Neumann S., Volkamer M. Tauglichkeit von Common Criteria-Schutzprofilen für Internetwahlen in Deutschland. Datenschutz und Datensicherheit-DuD. Springer Fachmedien Wiesbaden, 2014, Vol. 38, Issue 2, pp. 98 – 102.

37. Cranor L.F., Cytron R.K. Sensus: a security-conscious electronic polling system for the Internet. Proceedings of the Thirtieth Hawaii International Conference on System Sciences. Wailea (USA), 1997, vol.3, pp. 561 – 570.

38. Clarkson M.R., Chong S., Myers A.C. Civitas: Toward a Secure Voting System. IEEE Symposium on Security and Privacy. Oakland (USA), 2008, pp. 354 – 368.

39. EVOTE 2014. Review on the EVOTE 2014 conference at which POLYAS has been represented by Kai Reinhard URL: http://www.micromata.de/en/news/latest-news/news/?tx_ttnews[year]=2014&tx_ttnews[month]=11&tx_ttnews[tt_news]=609&cHash=de8528e29dfb0aca7fe1265134bf0e0a (дата звернення: 05.10.2017).

40. Electronic vote and democracy URL:http://www.electronic-vote.org/INTRO/index.php (дата звернення: 05.10.2017).

41. Вишняков В.М., Пригара М.П., Воронін О.В. Відкрита система таємного голосування. *Управління розвитком складних систем*. 2014. №20. С. 110 – 115.

42. Электронное голосование в Эстонии URL: https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%BE%D0%B5_%D0%B3%D0%BE%D0%BB%D0%BE%D1%81%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%B2_%D0%AD%D1%81%D1%82%D0%BE%D0%BD%D0%B8%D0%B8 (дата звернення: 05.10.2017).

43. Shannon C. Communication Theory of Secrecy Systems. *Bell System Technical Journal*. 1949. 28 (4). Pp. 656–715.

44. E-Voting denied in Switzerland over hacking fears. October 28, 2015 URL: http://e-lected.blogspot.com/search?updated-min=2015-01-01T00:00:00-08:00&updated-max=2016-01-01T00:00:00-08:00&max-results=49 (дата звернення: 05.10.2017).

45. ДСТУ 4145-2002. Інформаційні технології. Криптографічний захист інформації. Цифровий підпис, що ґрунтується на еліптичних кривих. Формування та перевіряння. [Чинний від 2003-07-01] Вид. офіц. Київ, 2003. 22 с.

46. W.Diffie, M.E.Hellman. New Direction in Cryptography. *IEEE Transactions on Information Theory*. 1976. v.IT-22, n.6. Pp. 644-654.

47. НД ТЗІ 1.1-002-99. Загальні положення щодо захисту інформації в комп'ютерних системах від несанкціонованого доступу. [Чинний від 1999-04-28]. Вид. офіц. Київ: ДСТСЗІ СБ України, 1999. 14 с.

48. НД ТЗІ 2.5-004-99. Критерії оцінки захищеності інформації в комп'ютерних системах від несанкціонованого доступу. [Чинний від 1999-04-28]. Вид. офіц. Київ: ДСТСЗІ СБ України, 1999. 53 с.

49. Бройдо В.Л., Ильина О.П. Архитектура ЭВМ и систем: Учебник для вузов. 2-е изд., СПб.: Питер, 2009. 720 с.

50. Гордеев А.В. Операционные системы: Учебник для вузов. 2-е изд., СПб.: Питер, 2004. 416 с.

51. Гмурман В. Е. Теория вероятностей и математическая статистика: Учебное пособие для вузов. 9-е изд., М.: Высшая школа, 2003. 479 с.

52. Вишняков В.М. Захист даних в інформаційних системах**:** Навчальний посібник. К.: КНУБА, 2009. 128 с.

53. Fenollosa C. OpenBSD from a veteran Linux user perspective. June 26, 2015. URL: http://cfenollosa.com/blog/openbsd-from-a-veteran-linux-user-perspective.html (дата звернення: 05.09.2017).

54. Конахович Г.Ф., Климчук В.П., Паук С.М., Потапов В.Г. Защита информации в телекоммуникационных системах: Навч. посіб. К. : МК-Пресс, 2005. 279 с.

55. Сушко С.О., Кузнецов Г.В., Фомичова Л.Я., Корабльов А.В. Математичні основи криптоаналізу: Навч. посіб. Д.: Національний гірничий ун-т, 2010. 465 с.

56. Вентцель Е.С., Овчаров Л.А. Теория вероятностей и ее инженерные приложения. 3-е изд., перераб. и доп. М.: Издательский центр «Академия», 2003. 464 с.

57. Хачатурова С.М. Электронный учебник по дисциплине "Математические модели системного анализа" URL: http://ermak.cs.nstu.ru/mmsa/main/Proba.htm (дата звернення: 05.09.2017).

58. Core Technology Services. North Dakota University System, Grand Forks, ND 58201 URL: https://listserv.nodak.edu (дата звернення: 05.09.2017).

59. Discrete logarithm records URL: https://en.wikipedia.org/wiki/Discrete_logarithm_records (дата звернення: 05.09.2017).

60. Thorsten Kleinjung, 2014 October 17, "Discrete Logarithms in GF(2^1279)". URL: https://listserv.nodak.edu/cgi-bin/wa.exe?A2=NMBRTHRY;256db68e.1410 (дата звернення: 05.09.2017).

61. Antoine Joux, "Discrete logarithms in GF(2^{4080})", Mar 22, 2013, URL: https://listserv.nodak.edu/cgi-bin/wa.exe?A2=ind1303&L=NMBRTHRY&F=&S=&P=13682 (дата звернення: 05.09.2017).

62. Faruk Gologlu et al., On the Function Field Sieve and the Impact of Higher Splitting Probabilities: Application to Discrete Logarithms in GF(2^{1971}), 2013. URL: http://eprint.iacr.org/2013/074 (дата звернення: 05.09.2017).

63. Antoine Joux, "Discrete logarithms in GF(2^{6168}) [=GF((2^{257})^{24})]", May 21, 2013. URL: https://listserv.nodak.edu/cgi-bin/wa.exe?A2=ind1305&L=NMBRTHRY&F=&S=&P=3034 (дата звернення: 05.09.2017).

64. Erich Wenger and Paul Wolfger, “Harder, Better, Faster, Stronger - Elliptic Curve Discrete Logarithm Computations on FPGAs”. URL: http://eprint.iacr.org/2015/143/ (дата звернення: 05.09.2017).

65. Jens Zumbrägel, "Discrete Logarithms in GF(2^9234)", 31 January 2014. URL: https://listserv.nodak.edu/cgi-bin/wa.exe?A2=NMBRTHRY;9aa2b043.1401 (дата звернення: 05.09.2017).

66. Programming languages used on the Internet and the World Wide Web (WWW) URL: http://www.webdevelopersnotes.com/basics/languages_on_the_internet.php3 (дата звернення: 05.09.2017).

67. Шаньгин В.Ф. Защита информации в компьютерных системах и сетях: Учебн. пособ. М.: Изд-во ДМК, 2012. 592 с.

68. НД ТЗІ 2.5-010-03 Вимоги до захисту інформації WEB – сторінки від несанкціонованого доступу. [Чинний від 2003-04-15] Вид. офіц. Київ: ДСТСЗІ СБ України, 2003. 53 с.

69. Колмогоров А. Н., Основные понятия теории вероятностей: 2 изд. – М. : Изд-во «Наука», 1974. 120 с.

70. Каргаполов М. И., Мерзляков Ю. И. Основы теории групп: 3 изд. М. : Изд-во «Наука», 1982. 288 с.

71. Essex A., Clark J., Hengartner U. Cobra: Toward Concurrent Ballot Authorization for Internet Voting. Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE'12). Bellevue (USA), 2012, pp. 1 – 13.

72. НД ТЗІ 2.5-005-99 Класифікація автоматизованих систем і стандартні функціональні профілі захищеності оброблюваної інформації від несанкціонованого доступу. [Чинний від 1999-07-01] Вид. офіц. Київ: ДСТСЗІ СБ України, 1999. 53 с..

73. Корченко О.Г., Козлюк І.О., Муратов О.Є. Модель складної орієнтованої інформаційної мережі ЗВДТ. *Захист інформації*. 2011. №2(52). С. 87-94.

74. Козлюк І.О., Конахович Г.Ф., Антонов В.В. Особливості передавання захищеного мовного трафіка через стандартний радіоканал авіаційних систем зв'язку. *Захист інформації*. 2011. №3(52). С. 72-77.

75. Козлюк І.О., Бахтіяров Д.І. Аналіз ефективності комплексного застосування заходів захищеності для підвищення стійкості функціонування засобів керування БПЛА. *Проблеми розвитку глобальної системи зв'язку, навігації та оргалізації повітряного руху CNS/ATM*: зб. тез. доп. науково-технічна конференція, 17-19 листопада 2014 р.: К., 2014. С. 25-26.

76. Смагин А. Г., Ярославский М. И. Пьезоэлектричество кварца и кварцевые резонаторы. М.: «Энергия», 1970. 488 с.

77. Чуприн В.М., Вишняков В.М., Пригара М.П. Генерування випадкових чисел штатними засобами хостів мережі Інтернет. *Захист інформації*. 2016. Том 18, №4. С. 323-335.

78. Чуприн В.М., Вишняков В.М., Пригара М.П. Метод протидії незаконному впливу на виборців у системі Інтернет голосування. *Безпека інформації*. 2017. Том 23, №1. С. 7-14.

79. Чуприн В.М., Вишняков В.М., Пригара М.П. Захист операційного середовища систем Інтернет голосування. *Захист інформації*. 2017. Том 19, №1. С. 56-66.

80. Вишняков В. М., Пригара М. П., Тарасюк Д. М. Багаторівневий захист даних в системах розв'язання складних задач на суперкомп'ютері. *Новітні комп'ютерні технології*. Матеріали IX Міжнародної науково-технічної конференції NOCOTE'2011 Київ–Севастополь, 13–16 вересня 2011 р. Київ: Мінрегіон України. С. 36-38.

81. Пригара М.П. Методи оптимального вибору рівня захисту комп'ютерних систем. *Наукова конференція молодих вчених, аспірантів і студентів КНУБА*: тези доповідей в 2-х частинах. Ч.1. К.: КНУБА, 2011. – С. 165.

82. Пригара М.П., Чайковська Т.О. Патерн «COMPOSITE». *Наукова конференція молодих вчених, аспірантів і студентів КНУБА*. Збірник тез студентських доповідей : КНУБА, 2012. С. 173.

83. Пригара М.П. Використання метода Форда-Фалкерсона для визначення перевантажених маршрутів та ресурсів комп'ютерних мереж. *Новітні комп'ютерні технології. Випуск XI*.. Кривий Ріг: ДВНЗ «Криворізький національний університет», 2013. С. 185-187.

84. Вишняков В.М., Пригара М.П. Забезпечення свободи волевиявлення в системі Інтернет-голосування (ІГ). *Інформація, комунікація, суспільство 2015*. Матеріали 4-ї Міжнародної наукової конференції ICS-2015. С. 124 – 125.

ДОДАТОК А.
АЛГОРИТМИ ТА ЛІСТИНГИ ПРОГРАМ ДЛЯ РЕАЛІЗАЦІЇ СИСТЕМИ ДИСТАНЦІЙНОГО ВОЛЕВИЯВЛЕННЯ

Д1.1. Алгоритм функціонування системи дистанційного голосування

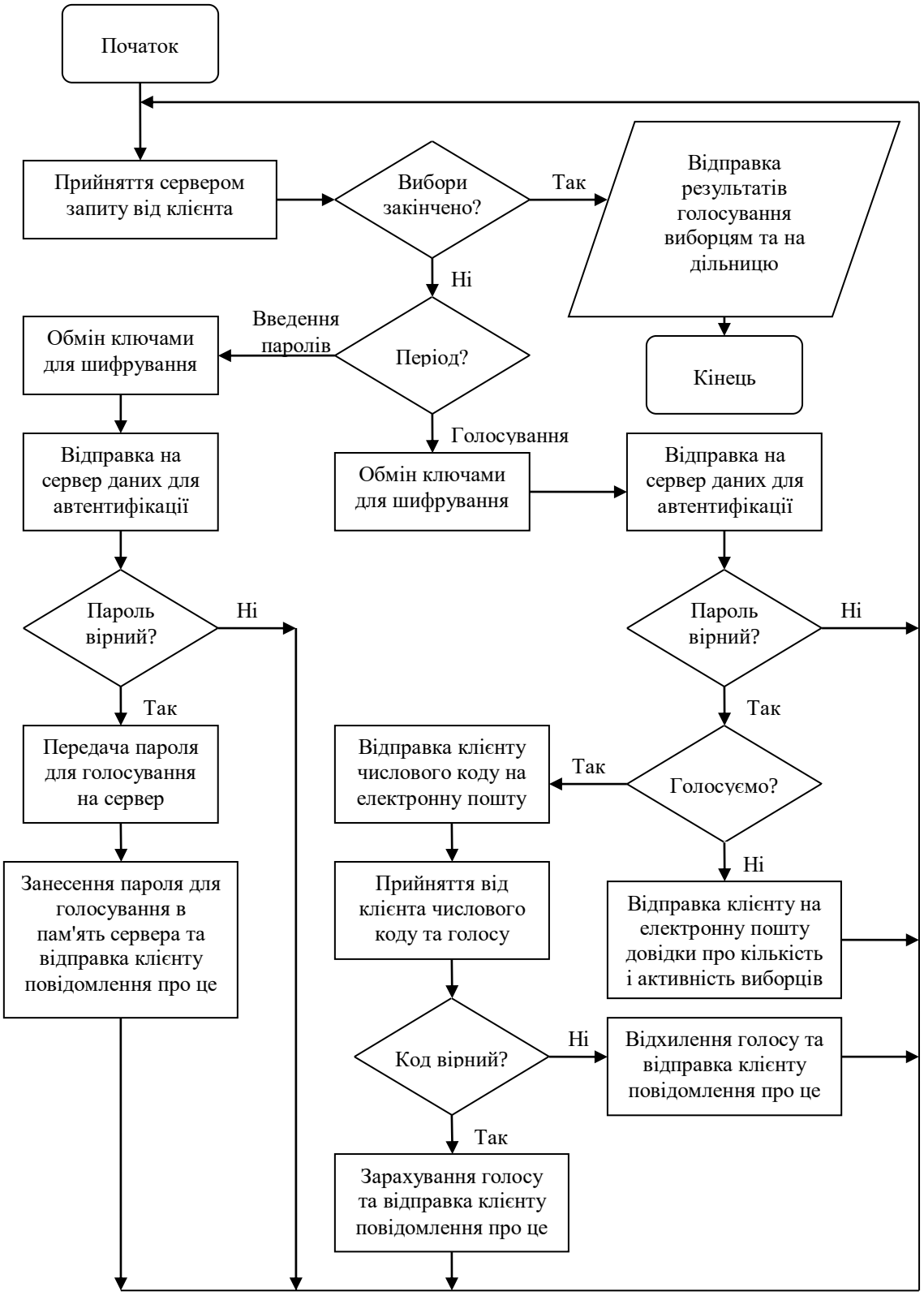

Рис. Д1.1. Схема алгоритму функціонування СДГ

Д1.2. Серверна програма для формування бази даних виборців (файл SBD.js)

```
//  SBD Server Base of Data Ver. 18 March 2015
var http = require('http');
var url = require('url');
var fs = require('fs');
var static = require('node-static');
var querystring = require('querystring');
var file = new static.Server('.');
var M1 = [504];   // Резервируем ячейки для элементов поля Галуа GF(2^503)
var M2 = [504];   // M1[], M2[] - сомножители  R[] - результат умножения
var R  = [504];   // Младший бит храним в [1], старший - в [503]. Нулевые
элементы не используем
// Переменные для работы с базой данных участников голосования. Файл VYBORCI.DBT
var SFILE='';// Строка для чтения и работы с файлом базы данных
var DATA=[216]; // Объявляем массив байт для приема строки данных
var Kstr=0;  // К-во строк данных
var Lstr=0;  // Длина строки данных (включая первый байт-признак)
var Nstr=0;  // Номер текущей строки (начиная с нулевой)
var Astr=0;  // Адрес (номер байта начиная с нулевого) первой строки данных
var EEE=0; // Признак наличия избирателя в базе ( 0-нет, 1-есть)
var OZNAKA='0'; // [0] 0-действующая строка данных, 1-выбыл, 2-данные заменены,
3-ошибочная строка, 4-самоотказ
var SERPASP='AA';     // [1] Серия паспорта
var NUMPASP='000000'; // [3] Номер паспорта
var PAROL='a12345';   // [9] Пароль для доступа клиента на сервер
var EMAIL_D='@';    // [33] Адрес электронной почты для отправки кода
var EMAIL_R='@';    // [69] Адрес электронной почты из ДРВ
var TELEFON='+380';   // [105] Номер мобильного телефона
var KODVUL='000';         // [118] Код улицы избирателя (таблица кодов
составляется на каждом участке)
var NUMBUD='0000';    // [121] Номер дома избирателя
var NUMKV='0000';         // [125] Номер квартиры избирателя
var FAMIL='Ж';            // [129] Фамилия (до 30 символов)
var IMJA='Ч';             // [159] Имя     (до 20 символов)
var OTCH='Ш';             // [179] Отчество (до 20 символов)
var DATE_IN='20141028'; // [199] Дата ввода данных (ГГГГММЧЧ)
var TIME_IN='1939';     // [207] Время ввода данных (ЧЧММ)
var DATE_OUT='20141028';// [211] Дата удаления данных (ГГГГММЧЧ)
var TIME_OUT='1939';    // [219] Время удаления данных (ЧЧММ)
var PRAVO='0';                // [223]  Признак наличия права участия в данных
выборах ("1"-есть, "0"-нет)
var VYBOR='0';                // [224] Признак использования права выбора ("1"-
проголосовал, "0"-нет)
////////////////////////////////////////////////////////////////////////////////
////
var i,j,I,J;
var CYF='0123456789';
var NU2 = [100]; // Массив для преобразование чисел 0-99 в строку из двух
символов
 for(i=0; i<10; i++)
 {
 for(j=0; j<10; j++) NU2[i*10+j]=CYF.substr(i,1)+CYF.substr(j,1);
 }
var M = new Array(504); // Создаем двумерный массив M[][] для степеней
примитивного элемента
  for(i=0; i<504; i++) M[i] = new Array(504);
var MA = new Array(504);      // Создаем двумерный массив MA[][] для степеней
A[] (массива битов, полученных от клиента)
  for(i=0; i<504; i++) MA[i] = new Array(504);
var N = [504];    // Создаем  массив N[] для случайных битов
var C = [504];    // Создаем  массив C[] битов для шифрования
var B = [504];    // Создаем массив B[] для значений X^N[], отправляемых клиенту
```

```
var O = [504];    // Создаем массив O[] для расшифрованых данных
var A = [504];    // Массив для получения от клиента значения X^N[], где X-
примитивный элемент поля Галуа
var NQ=0;       // Номер ожидаемого запроса клиента
var NC=0;       // К-во зашифрованных бит для передачи или приема
var NC1=1;       // Номер бита, с которого следует начинать/продолжать процедуру
шифрования
var T1 = new Date(); // Берем метку времени для шифрования + для проверки
времени выполнения
var TB = T1.getTime(); // TB - момент начала преобразований в милисекундах от
01.01.1970
var TN=TB;                      // Проверяем на четность к-во миллисекунд от
01.01.1970
var TD = TN%2;          // В зависимости от четности заполняем первый случайный
бит в текущей строке
      if (TD>0) N[1]=1; else  N[1]=0;
    for (var i=1;i<=503;i++) M[1][i] = 0; // Обнулили первую строку массива для
примитивного элемента
    M[1][2]=1; // Занесли единицу во второй бит. Это получился примитивный
элемент поля Галуа.
    for (I=2;I<=503;I++) // Начинаем цикл заполнения массива M[][] константами,
а N[] случайными битами
    {    for (J=1;J<=503;J++)
        {
           M1[J]= M2[J]=M[I-1][J]; // Оба сомножителя равны предыдущей строке
        }
    MULT(); // Каждая следующая строка массива M[][] равна квадрату предыдущей
строки
      for (j=1;j<=503;j++) M[I][j]=R[j]; // Занесли результат умножения в
очередную строку
      var T1 = new Date();                 // Начало вычисления случайного бита
      var TN = T1.getTime();              //    Берем к-во миллисекунд от
01.01.1970
      var TD = TN%2; //  В TD вычисляем признак четности к-ва миллисекунд
      if (TD>0) N[I]=1; else N[I]=0;  // Случайный бит зависит от четности к-ва
миллисекунд
      var NN = Math.random();             // и величины стандартного случайного
числа (сумма по модулю 2)
      if (NN>0.5) N[I]=N[I]+1; if (N[I]>1) N[I]=0; // Заполнили очередной
случайный бит
    } // Конец цикла заполнения массива константами
var TBG = T1.getTime(); // TBG - момент начала в миллисекундах от 01.01.1970
var TEND=TBG+3600000*8;   // Установили период (для начала 8 часов)
// Будем отправлять значение B[] клиенту
var TR='N';  // Строка для отправки данных клиенту
var RT='/';  // Строка для приема данных от клиента

http.createServer(function (req, res) // req - запрос от клиента, res - ответ
сервера на запрос клиента
{
 var pathname = url.parse(req.url).pathname;
      var T1 = new Date(); // Берем метку времени для проверки таймаута
      TBG = T1.getTime(); // TBG - момент начала в миллисекундах от 01.01.1970
      if (TEND<T1.getTime())  // если период закончился
      {
      NQ=0; // Устанавливаем номер ожидаемого запроса для нового клиента
      TEND=TBG+3600000; // Установили тайм-аут для ввода данных ( 1 час)
      }
if (pathname=="/BASE.html") file.serve(req, res);
else {  RT=pathname;
      if (RT[1]=='Q') // Все запросы клиента начинаются с буквы Q (после /)
      {
switch (RT[2])          // Следующая цифра означает номер запроса 0, 1 или 2
      {
```

```
      case '0': // Будем отправлять клиенту значение B[] в ответ на A[] (Diffie-
Hellman alhorithm)
            if (NQ==0)
            {
            var T1 = new Date(); // Берем метку времени для отсчета периода  от
стартового момента
            TBG = T1.getTime(); // TBG - момент начала в миллисекундах от
01.01.1970
            TEND=TBG+3600000/30;   // Установили 2 минуты на ожидание намера
паспорта
            NC1=1;
            STEPX(); // Вычисление случайной степени примитивного элемента X
            var T1 = new Date(); // Берем метку времени для шифрования + для
проверки времени выполнения
            var TB = T1.getTime(); // TB - момент начала преобразований в
миллисекундах от 01.01.1970
            TR='';  // Строка для передачи данных на сервер
            for (i=1;i<=503;i++) // Цикл переноса значения B[] в транспортную
строку
            {
            if (B[i]==0) TR=TR+'0'; else TR=TR+'1';
            }
            COUNTN(); //---- Вычисление послеовательностм битов для шифрования
сообщений
            NQ=1; // Установили номер следующего запроса от клиента
            }
            else TR='E8';
            res.end(TR);            // Отправили данные клиенту
      break;
      case '1': // Будем расшифровывать и проверять паспортные данные клиента
            var T1 = new Date(); // Берем метку времени для отсчета периода  от
стартового момента
            TBG = T1.getTime(); // TBG - момент начала в миллисекундах от
01.01.1970
            TEND=TBG+3600000/10;   // Установили 6 минут на ожидание данных о
клиенте
            if (NQ==1)
            {
            OZNAKA='0'; // Устанавливаем признак режима регистрации (R0)
            if (RT[4]!='0') switch (RT[4])
                                    { // Удаление строки по причине ...
                                    case '1': OZNAKA=49; break; //  выбытия
                                    case '2': OZNAKA=50; break; //
корректировка данных
                                    case '3': OZNAKA=51; break; //  ошибочное
занесение
                                    case '4': OZNAKA=52; break; //  отказ от ДГ
                                    }
            NC=RT.length-5; // Вычислили к-во символов шифротекста
            for (i=NC1;i<NC1+NC;i++) //  Цикл расшифрования по алгоритму Вернама
            {j=0; if (RT[i+4]=='1') j=1; O[i]=j+C[i]; if (O[i]>1) O[i]=0;

            } // Результат расшифровки занесли в O[]
            var r=NC1;
            var S4='0000';
            var SP='';
            for (i=0; i<2; i++) // Цикл получения серии паспорта
            {
            j=0;
            if (O[r]==1) j=j+1;
            if (O[r+1]==1)j=j+2;
            if (O[r+2]==1) j=j+4;
            if (O[r+3]==1) j=j+8;
            r=r+4;
```

```
            switch (j)
            {
            case 1: SP=SP+'A'; break; // Латинская кодировка букв
            case 2: SP=SP+'B'; break;
            case 3: SP=SP+'C'; break;
            case 4: SP=SP+'E'; break;
            case 5: SP=SP+'H'; break;
            case 6: SP=SP+'I'; break;
            case 7: SP=SP+'K'; break;
            case 8: SP=SP+'M'; break;
            case 9: SP=SP+'O'; break;
            case 10: SP=SP+'P'; break;
            case 11: SP=SP+'T'; break;
            case 12: SP=SP+'X'; break;
            }
            }
            SERPASP=SP;
            var SP='';
            for (i=0; i<6; i++) // Цикл получения номера паспорта
            {
            j=0;
            if (O[r]==1) j=j+1;
            if (O[r+1]==1)j=j+2;
            if (O[r+2]==1) j=j+4;
            if (O[r+3]==1) j=j+8;
            r=r+4;
            switch (j)
            {
            case 1: SP=SP+'1'; break;
            case 2: SP=SP+'2'; break;
            case 3: SP=SP+'3'; break;
            case 4: SP=SP+'4'; break;
            case 5: SP=SP+'5'; break;
            case 6: SP=SP+'6'; break;
            case 7: SP=SP+'7'; break;
            case 8: SP=SP+'8'; break;
            case 9: SP=SP+'9'; break;
            case 10: SP=SP+'0'; break;
            }
            }
            NUMPASP=SP;
      NC1=NC1+NC; NC=0; // Переустановили номер бита, с которого следует
начинать/продолжать процедуру шифрования
      SFILE=fs.readFileSync('VYBORCI.DBT');      // Чтение файла в строку
      Kstr=SFILE.length/226;                              // К-во строк данных
      Lstr=226;  // Длина строки данных (включая первый байт-признак)
      Nstr=0;  // Номер текущей строки (начиная с нулевой)
      Astr=1;  // Адрес (номер байта начиная с нулевого) первой строки данных
///////////////// Проверяем наличие/отсутствие в базе данных по избирателю и
формируем ответ ('E0' - Ok,  'E1' - Error)
      EEE=0; // Признак наличия избирателя в базе ( 0-нет, 1-есть)
      if (Kstr>0)
      {
      var SERNUM='';
            for (i=0; i<Kstr; i++) //
            {
 SERNUM=String.fromCharCode(SFILE[i*Lstr+1])+
            String.fromCharCode(SFILE[i*Lstr+2])+
            String.fromCharCode(SFILE[i*Lstr+3])+
            String.fromCharCode(SFILE[i*Lstr+4])+
            String.fromCharCode(SFILE[i*Lstr+5])+
            String.fromCharCode(SFILE[i*Lstr+6])+
            String.fromCharCode(SFILE[i*Lstr+7])+
            String.fromCharCode(SFILE[i*Lstr+8])+
```

```
            String.fromCharCode(SFILE[i*Lstr+9]);
            if (SERNUM=='0'+SERPASP+NUMPASP ) {EEE=1; Nstr=i;}
      // Если нашли в базе номер паспорта, то устанавливаем EEE=1; Nstr=i;
            }
      }
      NQ=0; // Установили номер для очередного разрешаемого запроса на случай
удаления строки БД или ошибки
      if (OZNAKA=='0')
      {
      if (EEE==1) TR='E1'; else {TR='E0'; NQ=2;}
      }
      else
      {
            if (EEE==0) TR='E1';
            else
            {
      // Удаление из базы (ставим OZNAKA= 1-4 и дату удаления, все остальное
сохраняем)
      // Здесь будем готовить дату и время удаления данных в DTT
      var T1 = new Date(); // Берем метку времени
      var DTT='20'; // Начинаем формировать строку с первых цифр года
            DTT=DTT+NU2[T1.getFullYear()-2000];
            DTT=DTT+NU2[T1.getMonth()+1];
            DTT=DTT+NU2[T1.getDate()];
            DTT=DTT+NU2[T1.getHours()];
            DTT=DTT+NU2[T1.getMinutes()];
      var STR=[226]; STR[0]=32;
      var WFILE=' ';
      for (i=1;i<226;i++) STR[i]=SFILE[i];
      if (Nstr==0) STR[1]=OZNAKA;
      for (i=1;i<212;i++) WFILE=WFILE+String.fromCharCode(STR[i]);
      if (Nstr==0) for (i=212;i<224;i++) WFILE=WFILE+DTT[i-212];
      else for (i=212;i<224;i++) WFILE=WFILE+String.fromCharCode(STR[i]);
      for (i=224;i<226;i++) WFILE=WFILE+String.fromCharCode(STR[i]);
      if (Kstr>1)
      for (I=1;I<Kstr;I++)
      {
      for (i=0;i<226;i++) STR[i]=SFILE[i+I*Lstr];
      if (Nstr==I) STR[1]=OZNAKA;
      for (i=0;i<212;i++) WFILE=WFILE+String.fromCharCode(STR[i]);
      if (Nstr==I) for (i=212;i<224;i++) WFILE=WFILE+DTT[i-212];
      else for (i=212;i<224;i++) WFILE=WFILE+String.fromCharCode(STR[i]);
      for (i=224;i<226;i++) WFILE=WFILE+String.fromCharCode(STR[i]);
      }
            fs.writeFileSync('VYBORCI.DBT',WFILE); // Запись в файл
            TR='E0'; // Данные удалены (помечены как удаленные)
            }
      }
      }
            else TR='E8';
            res.end(TR);            // Отправили клиенту ответ или код ошибки
      break;
      case '2': // Это запрос Q2 с данными для новой строки. Будем
расшифровывать и заносить в базу строку
            if (NQ==2) // Отправляем код ошибки, если не тот номер запроса
            {
            //var DATA=[216]; // Объявляем массив байт для приема строки данных
            for (i=0;i<216;i++)     DATA[i]=' '; // Для начала опробелили весь
массив
            i=3; // Начальный номер байта L в принятой строке RT зашифрованных
данных
            var LP=[11]; // Объявляем массив длин полей принимаемых данных
            LP[1]=24; LP[2]=36; LP[3]=36; LP[4]=13; LP[5]=3; LP[6]=4;
            LP[7]=4; LP[8]=30; LP[9]=20; LP[10]=20;
```

```
            J=0; // Номер текущего байта в массиве DATA[]
            var k=0; // К-во принимаемых байт текущего элемента данных
            var q, e, w, d; //
            d=0; // Номер байта начала поля в массиве DATA[]
            for (I=1;I<=10;I++) // Цикл по полям (элементам) принимаемых данных
            {
            k=0; while (RT.substr(i+1,2)!=NU2[k]) k++;
            var n;
            n=i+3;      // Нашли начальный номер байта зашифрованного элемента
данных
            i=n+k*7; // Нашли очередной номер байта L в принятой строке RT
зашифрованных данных
            if (k>0)
                  {
                  NC=k*7; // К-во битов шифротекста
                  if (NC1+NC>503) NC1=1;
                  for (j=0;j<k;j++) // Цикл по байтам принимаемых данных
                  {
                        for (q=NC1;q<NC1+NC;q++) //  Цикл расшифрования по
алгоритму Вернама
                        {e=0; if (RT[n+(q-NC1)]=='1') e=1; O[q]=e+C[q]; if
(O[q]>1) O[q]=0;
                        } // Результат расшифровки занесли в O[]
                  e=NC1+j*7;
                  w=0; // Числовое значение байта
                  if (O[e]==1)   w=w+1;
                  if (O[e+1]==1) w=w+2;
                  if (O[e+2]==1) w=w+4;
                  if (O[e+3]==1) w=w+8;
                  if (O[e+4]==1) w=w+16;
                  if (O[e+5]==1) w=w+32;
                  if (O[e+6]==1) w=w+64;
                  DATA[d+j]=String.fromCharCode(w);
                  }
                  }
                  else NC=0;
            NC1=NC1+NC;
            d=d+LP[I]; // Установили номер байта начала следующего поля в
массиве DATA[]
            }
            CONV_MD5(); // Преобразуем пароль по алгоритму MD5 в массиве DATA[]
(начальные 16 байт в 20 байт)
            CONV_MD5(); // Еще раз преобразуем пароль по алгоритму MD5 в массиве
DATA[] (16 байт в 20 байт)
            SFILE=SFILE+' 0'+SERPASP+NUMPASP;          // Записали начало
принятой строки
      // Здесь будем заполнять дату и время занесения данных
            var T1 = new Date(); // Берем метку времени
            var DTT='20'; // Начинаем формировать строку с первых цифр года
            DTT=DTT+NU2[T1.getFullYear()-2000];
            DTT=DTT+NU2[T1.getMonth()+1];
            DTT=DTT+NU2[T1.getDate()];
            DTT=DTT+NU2[T1.getHours()];
            DTT=DTT+NU2[T1.getMinutes()];
            for (j=0;j<12;j++) DATA[190+j]=DTT[j];
            DATA[214]='1'; DATA[215]='0';
            for (j=0;j<216;j++) SFILE=SFILE+DATA[j]; // Дописали все принятые
данные
            fs.writeFileSync('VYBORCI.DBT',SFILE); // Запись в файл
            TR='E0';
            NQ=0; // Установили номер для очередного разрешаемого запроса
            }
            else TR='E8';
            res.end(TR);            // Отправили клиенту ответ или код ошибки
```

```
      break;
      }
      }
      }
}).listen(8080);
console.log('Server listen port 8080');
////////////////////////////////////////////////////////////////////////////
///////////
function MULT()     // Умножение элементов поля Галуа GF(2^503) по правилу
полиномов
{
 var i,j,r,r1,r2,r3;
    for (i=1;i<=503;i++) R[i]=0; // Обнуляем 503 ячейки для битов результата
    for (i=1;i<=503;i++)           // Начинаем двойной цикл умножения элементов
полинома
    if (M1[i]==1) // Умножаем только единичные элементы из M1[], нулевые не
нужны, т.к. они дают нули
    {
        for (j=1;j<=503;j++)
        if (M2[j]==1)
        {
        r=i+j-1;
            if (r>503)
            {
             r=r-503;
             if (r>=501)
                {
                r=r-501;
                r1=1+r; r2=4+r; r3=501+r;
                if (R[r3]==0) R[r3]=1; else R[r3]=0;
                }
            else {r1=r; r2=r+3;}
            if (R[r1]==0) R[r1]=1; else R[r1]=0;
            if (R[r2]==0) R[r2]=1; else R[r2]=0;
            }
            else {if (R[r]==0) R[r]=1; else R[r]=0;}
        }
    }
} // End of function MULT()
function STEPX()     // Вычисление случайной степени примитивного элемента X
{ //---- Вычисление A[]=X[]^N[], где X[]- примитивный элемент поля Галуа, N[]-
503 наших случайных бит
    for (i=1;i<=503;i++) A[i]=0; A[1]=1; // Занесли в A[] единицу
    for (J=1;J<=503;J++) // Начинаем двойной цикл возведения в степень
        if (N[J]==1)
        {
            for (I=1;I<=503;I++)
            {
                M1[I]= M[J][I];
                M2[I]=A[I];
            }
            MULT();
            for (I=1;I<=503;I++)
              A[I]=R[I];
        }  // Конец двойного цикла возведения в степень.
            for (I=1;I<=503;I++) B[I]=R[I]; //Результат перенесли в B[]
} // End of function STEPX()
function COUNTN() //---- Вычисление послеовательностм битов для шифрования
сообщений
 {          for (i=1;i<=503;i++) A[i]=RT[i+2]; // Занесли принятый от клиента
массив битов в A[]
      for (i=1;i<=503;i++) MA[1][i] = A[i]; // В первую строку массива занесли
значение A[] в степени 1
```

```
            for (I=2;I<=503;I++) // Начинаем цикл заполнения массива MA[][]
степенями A[]
            {
            for (J=1;J<=503;J++) M1[J]=M2[J]=MA[I-1][J]; // Оба сомножителя
равны предыдущей строке
            MULT(); // Каждая следующая строка массива MA[][] равна квадрату
предыдущей строки
                  for (j=1;j<=503;j++) MA[I][j]=R[j]; // Занесли результат
умножения в очередную строку
            } // Конец цикла заполнения массива степенями A[]
            for (i=1;i<=503;i++) C[i]=0; C[1]=1; // Занесли в C[] единицу
      for (J=1;J<=503;J++) // Начинаем двойной цикл возведения в степень
        if (N[J]==1)
        {
            for (I=1;I<=503;I++)
            {
                M1[I]= MA[J][I];
                M2[I]=C[I];
            }
            MULT();
            for (I=1;I<=503;I++)
              C[I]=R[I];
        }  // Конец двойного цикла возведения в степень. Результат занесен в C[]

            // --- В C[] получена случайная послеовательность для шифрования
сообщений
} // End of function COUNTN()
function CONV_MD5() // Преобразуем пароль по алгоритму MD5 в массиве DATA[]
(начальные 16 байт в 20 байт)
{      var olda, oldb, oldc, oldd,
            a =  1732584193,
            b = -271733879,
            c = -1732584194,
            d =  271733878;

            olda = a;
            oldb = b;
            oldc = c;
            oldd = d;

            a = md5_ff(a, b, c, d, DATA[0],  7, -680876936);
            d = md5_ff(d, a, b, c, DATA[1], 12, -389564586);
            c = md5_ff(c, d, a, b, DATA[2], 17,  606105819);
            b = md5_ff(b, c, d, a, DATA[3], 22, -1044525330);
            a = md5_ff(a, b, c, d, DATA[4],  7, -176418897);
            d = md5_ff(d, a, b, c, DATA[5], 12,  1200080426);
            c = md5_ff(c, d, a, b, DATA[6], 17, -1473231341);
            b = md5_ff(b, c, d, a, DATA[7], 22, -45705983);
            a = md5_ff(a, b, c, d, DATA[8],  7,  1770035416);
            d = md5_ff(d, a, b, c, DATA[9], 12, -1958414417);
            c = md5_ff(c, d, a, b, DATA[10], 17, -42063);
            b = md5_ff(b, c, d, a, DATA[11], 22, -1990404162);
            a = md5_ff(a, b, c, d, DATA[12],  7,  1804603682);
            d = md5_ff(d, a, b, c, DATA[13], 12, -40341101);
            c = md5_ff(c, d, a, b, DATA[14], 17, -1502002290);
            b = md5_ff(b, c, d, a, DATA[15], 22,  1236535329);

            a = md5_gg(a, b, c, d, DATA[1],  5, -165796510);
            d = md5_gg(d, a, b, c, DATA[6],  9, -1069501632);
            c = md5_gg(c, d, a, b, DATA[11], 14,  643717713);
            b = md5_gg(b, c, d, a, DATA[0],  20, -373897302);
            a = md5_gg(a, b, c, d, DATA[5],  5, -701558691);
            d = md5_gg(d, a, b, c, DATA[10],  9,  38016083);
            c = md5_gg(c, d, a, b, DATA[15], 14, -660478335);
```

```
            b = md5_gg(b, c, d, a, DATA[4], 20, -405537848);
            a = md5_gg(a, b, c, d, DATA[9],  5,  568446438);
            d = md5_gg(d, a, b, c, DATA[14],  9, -1019803690);
            c = md5_gg(c, d, a, b, DATA[3], 14, -187363961);
            b = md5_gg(b, c, d, a, DATA[8], 20,  1163531501);
            a = md5_gg(a, b, c, d, DATA[13],  5, -1444681467);
            d = md5_gg(d, a, b, c, DATA[2],  9, -51403784);
            c = md5_gg(c, d, a, b, DATA[7], 14,  1735328473);
            b = md5_gg(b, c, d, a, DATA[12], 20, -1926607734);

            a = md5_hh(a, b, c, d, DATA[5],  4, -378558);
            d = md5_hh(d, a, b, c, DATA[8], 11, -2022574463);
            c = md5_hh(c, d, a, b, DATA[11], 16,  1839030562);
            b = md5_hh(b, c, d, a, DATA[14], 23, -35309556);
            a = md5_hh(a, b, c, d, DATA[1],  4, -1530992060);
            d = md5_hh(d, a, b, c, DATA[4], 11,  1272893353);
            c = md5_hh(c, d, a, b, DATA[7], 16, -155497632);
            b = md5_hh(b, c, d, a, DATA[10], 23, -1094730640);
            a = md5_hh(a, b, c, d, DATA[13],  4,  681279174);
            d = md5_hh(d, a, b, c, DATA[0],  11, -358537222);
            c = md5_hh(c, d, a, b, DATA[3], 16, -722521979);
            b = md5_hh(b, c, d, a, DATA[6], 23,  76029189);
            a = md5_hh(a, b, c, d, DATA[9],  4, -640364487);
            d = md5_hh(d, a, b, c, DATA[12], 11, -421815835);
            c = md5_hh(c, d, a, b, DATA[15], 16,  530742520);
            b = md5_hh(b, c, d, a, DATA[2], 23, -995338651);

            a = md5_ii(a, b, c, d, DATA[0],  6, -198630844);
            d = md5_ii(d, a, b, c, DATA[7], 10,  1126891415);
            c = md5_ii(c, d, a, b, DATA[14], 15, -1416354905);
            b = md5_ii(b, c, d, a, DATA[5], 21, -57434055);
            a = md5_ii(a, b, c, d, DATA[12],  6,  1700485571);
            d = md5_ii(d, a, b, c, DATA[3], 10, -1894986606);
            c = md5_ii(c, d, a, b, DATA[10], 15, -1051523);
            b = md5_ii(b, c, d, a, DATA[1], 21, -2054922799);
            a = md5_ii(a, b, c, d, DATA[8],  6,  1873313359);
            d = md5_ii(d, a, b, c, DATA[15], 10, -30611744);
            c = md5_ii(c, d, a, b, DATA[6], 15, -1560198380);
            b = md5_ii(b, c, d, a, DATA[13], 21,  1309151649);
            a = md5_ii(a, b, c, d, DATA[4],  6, -145523070);
            d = md5_ii(d, a, b, c, DATA[11], 10, -1120210379);
            c = md5_ii(c, d, a, b, DATA[2], 15,  718787259);
            b = md5_ii(b, c, d, a, DATA[9], 21, -343485551);

var i, W=[4];
            W[0] = safe_add(a, olda); if (W[0]<0) W[0]=4294967296+W[0];
        W[1] = safe_add(b, oldb); if (W[1]<0) W[1]=4294967296+W[1];
        W[2] = safe_add(c, oldc); if (W[2]<0) W[2]=4294967296+W[2];
        W[3] = safe_add(d, oldd); if (W[3]<0) W[3]=4294967296+W[3];
    for (i=0; i<4; i++)
      {
      DATA[0+i]   = W[i]%128;
      DATA[4+i]   = ((W[i]-DATA[0+i])/128)%128;
      DATA[8+i]   = ((W[i]-DATA[0+i]-DATA[4+i]*128)/(128*128))%128;
      DATA[12+i]  = ((W[i]-DATA[0+i]-DATA[4+i]*128-
DATA[8+i]*128*128)/(128*128*128))%128;
      DATA[16+i]  = ((W[i]-DATA[0+i]-DATA[4+i]*128-DATA[8+i]*128*128-
DATA[12+i]*128*128*128)/(128*128*128*128))%128;
    }
      for (i=0; i<20; i++) DATA[i]=String.fromCharCode(DATA[i]);
}
    function safe_add(x, y) // Суммирование целых 32-битных чисел
      {
        var lsw = (x & 0xFFFF) + (y & 0xFFFF),
```

```
            msw = (x >> 16) + (y >> 16) + (lsw >> 16);
        return (msw << 16) | (lsw & 0xFFFF);
    }

    function bit_rol(num, cnt) // Циклический сдвиг влево на cnt бит
      {
        return (num << cnt) | (num >>> (32 - cnt));
    }
// Пять функций к алгоритму MD5
    function md5_cmn(q, a, b, x, s, t) {
        return safe_add(bit_rol(safe_add(safe_add(a, q), safe_add(x, t)), s),
b);
    }
    function md5_ff(a, b, c, d, x, s, t) {
        return md5_cmn((b & c) | ((~b) & d), a, b, x, s, t);
    }
    function md5_gg(a, b, c, d, x, s, t) {
        return md5_cmn((b & d) | (c & (~d)), a, b, x, s, t);
    }
    function md5_hh(a, b, c, d, x, s, t) {
        return md5_cmn(b ^ c ^ d, a, b, x, s, t);
    }
    function md5_ii(a, b, c, d, x, s, t) {
        return md5_cmn(c ^ (b | (~d)), a, b, x, s, t);
    }
```

Д1.3. Клієнтська програма для формування бази даних виборців (файл BASE.html)

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
 <title> БД виборців </title>
</head>
<body>
 <h1>База даних виборців</h1>
 <h2>Виборча дільниця № 1</h2>
<script type="text/javascript">
var FNEW=0;       // Флаг обновления запроса к серверу БД
var M1 = [504];   // Резервируем ячейки для элементов поля Галуа GF(2^503)
var M2 = [504];   // M1[], M2[] - сомножители  R[] - результат умножения
var R  = [504];   // Младший бит храним в [1], старший - в [503]. Нулевые
элементы не используем
function MULT()     // Умножение элементов поля Галуа GF(2^503) по правилу
полиномов
{                   // Для сокращения используем полином  X^503=X^3+1
 var i,j,r,r1,r2,r3;
    for (i=1;i<=503;i++) R[i]=0; // Обнуляем 503 ячейки для битов результата
    for (i=1;i<=503;i++) // Начинаем внешнй (двойной) цикл умножения полиномов
по элементам M1[]
    if (M1[i]==1) // Берем только единичные элементы из M1[], т.к. нулевые дают
нули
    {
        for (j=1;j<=503;j++) // Внутренний цикл по элементам M2[]
        if (M2[j]==1) // Берем только единичные элементы из M2[], т.к. нулевые
дают нули
        {
        r=i+j-1; // Вычисляем номер элемента R[] для занесения результата
умножения
            if (r>503)  // Если номер элемента R[] получился за пределами
массива R[],
            {                // то будем использовать сокращающий полином
X^503=X^3+1
             r=r-503;
             if (r>=501) // Если после использования сокращающего полинома
                {             //    номер элемента R[] все равно получился за
пределами массива R[],
                r=r-501; // то будем вторично использовать сокращающий полином
X^503=X^3+1
                r1=1+r; r2=4+r; r3=501+r; // Вычислили номера трех элементов
R[], к которым нужно добавить 1 по модулю 2
                if (R[r3]==0) R[r3]=1; else R[r3]=0; // Добавление 1 по модулю 2
в R[r3]
                }
            else {r1=r; r2=r+3;} // Вычислили номера двух элементов R[], к
которым нужно добавить 1 по модулю 2
            if (R[r1]==0) R[r1]=1; else R[r1]=0; // Добавление 1 по модулю 2 в
R[r1]
            if (R[r2]==0) R[r2]=1; else R[r2]=0; // Добавление 1 по модулю 2 в
R[r2]
            }
            else {if (R[r]==0) R[r]=1; else R[r]=0;} // Добавление 1 по модулю 2
в R[r]
        }
    }
} // End of function MULT()

function getXmlHttp() // Кроссбраузерная (стандартная) функция создания
XMLHttpRequest
```

```
{ // С помощью этой функции будем осуществлять обмен данными с сервером
  var xmlhttp;
  try {
    xmlhttp = new ActiveXObject("Msxml2.XMLHTTP");
  } catch (e) {
    try {
      xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
    } catch (E) {
      xmlhttp = false;
    }
  }
  if (!xmlhttp && typeof XMLHttpRequest!='undefined') {
    xmlhttp = new XMLHttpRequest();
  }
  return xmlhttp;
} // End of function getXmlHttp()
var i,j,I,J;  // Объявили переменные для циклов
var CYF='0123456789'; // Строка цифр для заполнения массива NU2[]
var NU2 = [100];  // Массив для преобразование чисел 0-99 в строку из двух
символов
 for(i=0; i<10; i++)
 {
 for(j=0; j<10; j++) NU2[i*10+j]=CYF.substr(i,1)+CYF.substr(j,1);
 } // Заполнили массив NU2[] (NU2[0]="00", NU2[1]="01", ... , NU2[99]="99"
var M = new Array(504); // Создаем двумерный массив M[][] для степеней
примитивного элемента
  for(i=0; i<504; i++) M[i] = new Array(504);
// Объявляем массивы для реализации обмена ключами по алгоритму Диффи-Хеллмана
над полем Галуа GF(2^503)
 var N = [504];   // Массив случайных битов N[], сформированный на нашей стороне
 var A = [504];   // Массив битов для отправки на сервер значения X^N[], где X-
примитивный элемент поля Галуа
 var B = [504];   // Массив битов B[], полученных от сервера (аналогично A[])
 var C = [504];   // Массив битов для шифрования, полученный как B[]^N[]
 var O = [504];   // Массив битов для наших открытых данных, которые будем
шифровать.
                        // Результат шифрования будем заносить в C[] на место
случайных битов
 var NC=0;       // К-во зашифрованных бит для передачи
 var NC1=1;      // Номер бита, с которого следует начинать/продолжать процедуру
шифрования
var T1 = new Date(); // Берем метку времени для шифрования + для проверки
времени выполнения
var TB = T1.getTime(); // TB - момент начала преобразований в милисекундах от
01.01.1970
var TIO=TB+300000;      // Установили таймаут для заполнения анкеты (5 минут)
var TN=TB;                      // Проверяем на четность к-во миллисекунд от
01.01.1970
var TD = TN % 2;         // В зависимости от четности будем заполнять первый
случайный бит
      if (TD>0) N[1]=1; else N[1]=0; // Заполнили первый случайный бит в N[]
// Нулевые элементы массивов N[], A[], B[], C[], O[] не используем!!!!!!!!
    for (i=1;i<=503;i++) M[1][i] = 0; // Обнулили первую строку массива для
примитивного элемента
    M[1][2]=1; // Занесли единицу во второй бит. Это получился примитивный
элемент поля Галуа.
    for (I=2;I<=503;I++) // Начинаем цикл заполнения массива M[][] степенями
примитивного элемента, а N[] случайными битами
    {    for (J=1;J<=503;J++)
        {
           M1[J]= M2[J]=M[I-1][J]; // Оба сомножителя равны предыдущей строке
        }
    MULT(); // Каждая следующая строка массива M[][] равна квадрату предыдущей
строки,
```

```
// где M[номер строки][номер элемента в строке]
      for (j=1;j<=503;j++) M[I][j]=R[j]; // Занесли результат умножения в
очередную строку
      var T1 = new Date();                 // Начало вычисления случайного бита
      var TN = T1.getTime();              //    Берем к-во миллисекунд от
01.01.1970
      var TD = TN % 2; //  В TD вычисляем признак четности к-ва миллисекунд
      if (TD>0) N[I]=1; else N[I]=0;  // Случайный бит зависит от четности к-ва
миллисекунд
      var NN = Math.random();             // и величины стандартного случайного
числа (сумма по модулю 2)
      if (NN>0.5) N[I]=N[I]+1; if (N[I]>1) N[I]=0; // Заполнили очередной
случайный бит
    } // Конец цикла заполнения массива степенями (1, 2, 4, 8, 16,...)
примитивного элемента GF(2^503)
//---- Вычисление A[]=X[]^N[], где X[]- примитивный элемент поля Галуа, N[]- 503
наших случайных бит
    for (i=1;i<=503;i++) A[i]=0; A[1]=1; // Занесли в A[] единицу
    for (J=1;J<=503;J++) // Начинаем двойной цикл возведения в степень
        if (N[J]==1) // Берем только единичные элементы N[], т.к. нулевые дают
нули
        {
            for (I=1;I<=503;I++) // Начинаем внутренний цикл подготовки
сомножителей
            {
                M1[I]= M[J][I]; // В M1[] заносим нужную степень примитивного
элемента
                M2[I]=A[I];   // В M2[] заносим результат предыдущего умножения
из A[]
            }
            MULT();
            for (I=1;I<=503;I++)
              A[I]=R[I];
        }  // Конец двойного цикла возведения в степень. Результат занесен в A[]
// Будем отправлять значение A[] на сервер
var TR='/Q0'; // Строка для транспортировки данных на сервер
var RT='/';  // Строка для приема ответных данных от сервера
var Nxx='X'; // Строка для номера строки данных, назначаемая сервером
 for (i=1;i<=503;i++) // Цикл переноса значения A[] в транспортную строку
 {
 if (A[i]==0) TR=TR+'0'; else TR=TR+'1';
 }
 var xmlhttp = getXmlHttp();     // Асинхронно отправляем на сервер значение A[]
через TR,
xmlhttp.open('GET', TR,  true);     // используя стандартную функцию обмена
данными с сервером
xmlhttp.onreadystatechange = function()
{
  if (xmlhttp.readyState == 4)
  {
     if(xmlhttp.status == 200)
       {
        RT=xmlhttp.responseText;  // Ответные данные от сервера заносим в RT
            if (RT[0]=='E') {alert("Лінію зайнято. Зачекайте."); FNEW=1;}
        for (i=1;i<=503;i++) B[i]=RT[i-1]; // Цикл заполнения значения B[]
//      alert("B="+RT);  //-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
/-/-/
     }
       else {alert("Лінію зайнято. Зробіть паузу."); FNEW=1;}
  }
};
xmlhttp.send(null);

if (FNEW==1) NEWIN();
```

```
//-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/
//for (j=1;j<=503;j++)  //-------- Отладочный цикл для визуализации данных
//var T2 = new Date();    // <--- Отладочная проверка времени выполнения программы ---------------
//var TE = T2.getTime();  // TE - момент окончания преобразований в милисекундах от 01.01.1970
//var TT = TE - TB;           // TT - время преобразований в милисекундах
// alert(TT+","+N);     // ---------- Конец проверки времени выполнения программы ---------------
//-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/

// Начало ввода и преобразования идентификационных данных клиента
var SEP; // Серия паспорта (2 большие буквы)
var NUP; // Номер паспорта (6 цифр)
var PAR; // Пароль (не менее 10 и не более 16 латинских букв или цифр с пробелом)
var RPW=''; // Подготовили строку для повторения пароля
var NR=0; // Номер выбранного режима
var FTAB1=0; // Флаг вызова функции TAB1()
var PAROL='a12345';   // [9] Пароль для доступа клиента на сервер
var REPEAT='a12345';  // [21] Повторение пароля
var EMAIL_D='@';    // [33] Адрес электронной почты для отправки кода
var EMAIL_R='@';    // [69] Адрес электронной почты из ДРВ
var TELEFON='+380';   // [105] Номер мобильного телефона
var KODVUL='000';         // [118] Код улицы адреса избирателя (таблица кодов составляется на каждом участке)
var NUMBUD='0000';    // [121] Номер дома избирателя
var NUMKV='0000';         // [125] Номер квартиры избирателя
var FAMIL='Ж';            // [129] Фамилия
var IMJA='Ч';             // [159] Имя
var OTCH='Ш';             // [179] Отчество

var F=''; // Строка для формирования данных по форме Lxx10110...10
var S=''; // Строка для анализа элемента данных
var NE=0; // Номер элемента БД
///////////////////////////////////////////////////////////////////////////////////////////
function TAB1()    // Обработка формы для ввода данных идентификации клиента
{
if (FNEW==1) {NEWIN(); return;}
if (FTAB1!=0) {alert("Ваш запит відправлено. Очікуйте"); return;}
SEP=document.getElementById('SER').value;
NUP=document.getElementById('NUMPASP').value;
NR=document.getElementById('REJIM').options.selectedIndex;
// alert("NR="+NR);  //-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
//var T1 = new Date(); // Берем метку времени для проверки таймаута
//if (T1.getTime()>TIO) {alert("Відведений час вичерпано. Почніть спочатку."); return;}
//if ( Nxx=='X') {alert("Лінію зайнято. Почніть спочатку за пару хвилин."); return;}
if (CONV1()==1) {CODE1(); FTAB1=1; TRAN();}
else alert("Необхідно виправити помилку");
if (FNEW==1) NEWIN();
} //  End of function TAB1()

function CODE1()   // Шифрование данных идентификации клиента шифром Вернама
{
//---- Вычисление C[]=B[]^N[], где N[]- 503 наших случайных бит
 for (i=1;i<=503;i++) M[1][i]=B[i]; // Занесли B[] в M[1][ ]
    for (I=2;I<=503;I++) // Начинаем цикл заполнения массива M[][] степенями B[]
    {    for (J=1;J<=503;J++)
        {
```

```
           M1[J]= M2[J]=M[I-1][J]; // Оба сомножителя равны предыдущей строке
         }
    MULT(); // Каждая следующая строка массива M[][] равна квадрату предыдущей
строки
      for (j=1;j<=503;j++) M[I][j]=R[j]; // Занесли результат умножения в
очередную строку
    } // Конец цикла заполнения массива степенями B[]
      for (i=1;i<=503;i++) C[i]=0; C[1]=1; // Занесли в C[] единицу
    for (J=1;J<=503;J++) // Начинаем двойной цикл возведения в степень
        if (N[J]==1)
        {
            for (I=1;I<=503;I++)
            {
                M1[I]= M[J][I];
                M2[I]=C[I];
            }
            MULT();
            for (I=1;I<=503;I++)
              C[I]=R[I];
        }  // Конец двойного цикла возведения в степень. Результат занесен в C[]
// --- В C[] получена случайная послеовательность для шифрования сообщений
//var TEMP='/'; // Контрольный вывод C[] /-/-/-/-/-/-/-/-/-/-//-/-/-/-/-/-/-/-/-/-
/-/-/-/-/-/-/-/--/-/-//-/-/-/-
//for (i=1;i<=503;i++) TEMP=TEMP+C[i]; ////--/-/-/-/-/-/-/-/-/-//-/-/-/-/-/-/-/-
/-/-/-/-/-/-/-/-/-/-/--/-/-//-/-/-/-
//alert(TEMP);/////////-/-/-/-/-/-/-/-/-/-/--/-/-/-/-/-/-/-/-/-//-/-/-/-/-/-/-/-
/-/-/-/-/-/-/-/-/-/-/--/-/-//-/-/-/-

for (i=1; i<=NC; i++) // Цикл шифрования по алгоритму Вернама
{O[i]=O[i]+C[i]; if (O[i]>1) O[i]=0;}
} // End of function CODE1()

function TRAN()    // Передача зашифрованных данных идентификации клиента на
сервер
{
TR="/Q1R"+CYF.substr(NR,1); // Сформировали заголовок отправляемой строки данных
for (i=NC1; i<NC1+NC; i++) TR=TR+O[i]; // и занесли зашифрованные данные
NC1=NC1+NC; NC=0; // Переустановили номер бита, с которого следует
начинать/продолжать процедуру шифрования

var xmlhttp = getXmlHttp();     // Асинхронно отправляем на сервер данные
клиента через TR
xmlhttp.open('GET', TR,  true); // Открываем соединение для отправки данных на
сервер
xmlhttp.onreadystatechange = function()
{
  if (xmlhttp.readyState == 4)
  {
     if(xmlhttp.status == 200)
       {
        RT=xmlhttp.responseText;  // Ответные данные от сервера заносим в RT
//alert(RT);/////////-/-/-/-/-/-/-/-/-/-/--/-/-/-/-/-/-/-/-/-//-/-/-/-/-/-/-/-
/-/-/-/-/-/-/-/-/-/--/-/-//-/-/-/-
        if (RT[0]=='E')  // Определяем верность данных по первой букве
        {
            switch (RT[1])
            {
            case '0': if (NR==0) alert("Зв'язок встановлено. Нажміть OK.");
                        else {alert("Дані про виборця видалено з
бази.\nЗробіть перезавантаження сторінки"); FNEW=1;}
            break;
            case '1': if (NR==0) {alert("Цього виборця вже введено.\nЗробіть
перезавантаження сторінки"); FNEW=1;}
```

```
                          else {alert("Такого виборця немає в базі.\nЗробіть
перезавантаження сторінки"); FNEW=1;}
            break;
            case '2': alert("Очікувати завершення вводу даних?"); break;
            default: alert("Сервер зайнятий."); FNEW=1; break;
            }
            return;
        }

     }
       else {alert("Лінію зайнято. Зачекайте."); FNEW=1;}
  }
};
xmlhttp.send(null);
} // End of function TRAN{)

function TAB2()    // Обработка формы для ввода данных идентификации клиента
{
if (FNEW==1) {NEWIN(); return;}
var i=0;
PAROL=document.getElementById('PAROL').value;
REPEAT=document.getElementById('REPEAT').value;
if (PAROL!=REPEAT){alert("Заново введіть пароль"); return;}
EMAIL_D=document.getElementById('EMAIL_D').value;
EMAIL_R=document.getElementById('EMAIL_R').value;
TELEFON=document.getElementById('TELEFON').value;
i=document.getElementById('VUL').options.selectedIndex;
KODVUL=NU2[i];
NUMBUD=document.getElementById('NUMBUD').value;
NUMKV=document.getElementById('NUMKV').value;
FAMIL=document.getElementById('FAMIL').value;
IMJA=document.getElementById('IMJA').value;
OTCH=document.getElementById('OTCH').value;
//var T1 = new Date(); // Берем метку времени для проверки таймаута
//if (T1.getTime()>TIO) {alert("Відведений час вичерпано. Почніть спочатку.");
return;}
//if ( Nxx=='X') {alert("Лінію зайнято. Почніть спочатку за пару хвилин.");
return;}
if (CONV2()==1){TRAN2();  NEWIN();}
else alert("Необхідно виправити помилку");
} // End of function TAB2()

function CONV1()   // Конвертирование данных идентификации клиента в битовую
последовательность
{
var i, j, k;
var n = [20];
for (i=0; i<2; i++)
      switch (SEP[i])
      {
      case ' ': n[i]=0; break;
      case 'A': // Украинская кодировка и под ней латинская
      case 'A': n[i]=1; break;
      case 'B':
      case 'B': n[i]=2; break;
      case 'C':
      case 'C': n[i]=3; break;
      case 'E':
      case 'E': n[i]=4; break;
      case 'H':
      case 'H': n[i]=5; break;
      case 'I':
      case 'I': n[i]=6; break;
      case 'K':
```

```
      case 'K': n[i]=7; break;
      case 'M':
      case 'M': n[i]=8; break;
      case 'O':
      case 'O': n[i]=9; break;
      case 'P':
      case 'P': n[i]=10; break;
      case 'T':
      case 'T': n[i]=11; break;
      case 'X':
      case 'X': n[i]=12; break;
      default: return 0;
      }
      if (n[0]==0) if (n[1]!=0) return 0;
      if (n[1]==0) if (n[0]!=0) return 0;
      for (i=0; i<6; i++)
      switch (NUP[i])
      {
      case '1': n[i+2]=1; break;
      case '2': n[i+2]=2; break;
      case '3': n[i+2]=3; break;
      case '4': n[i+2]=4; break;
      case '5': n[i+2]=5; break;
      case '6': n[i+2]=6; break;
      case '7': n[i+2]=7; break;
      case '8': n[i+2]=8; break;
      case '9': n[i+2]=9; break;
      case '0': n[i+2]=10; break;
      default: return 0;
      }
      for (i=1; i<504; i++) O[i]=0;// Обнулили битовый массив O[] для шифрования
данных
      for (i=0; i<8; i++) // Цикл преобразования данных паспорта из первых
восьми элементов n[] в O[]
      {
      if (n[i]>7) {O[i*4+4]=1; n[i]=n[i]-8;}
      if (n[i]>3) {O[i*4+3]=1; n[i]=n[i]-4;}
      if (n[i]>1) {O[i*4+2]=1; n[i]=n[i]-2;}
      O[i*4+1]=n[i];
      }
      NC=32;
      return 1
} // End of function CONV1()
function CONV2() // Конвертирование строки данных клиента в битовую
последовательность
{
F='';
NE=1; S=PAROL; if (CODE2()==0) return 0;
NE=3; S=EMAIL_D; if (CODE2()==0) return 0;
NE=4; S=EMAIL_R; if (CODE2()==0) return 0;
NE=5; S=TELEFON; if (CODE2()==0) return 0;
NE=6; S=KODVUL; if (CODE2()==0) return 0;
NE=7; S=NUMBUD; if (CODE2()==0) return 0;
NE=8; S=NUMKV; if (CODE2()==0) return 0;
NE=9; S=FAMIL; if (CODE2()==0) return 0;
NE=10; S=IMJA; if (CODE2()==0) return 0;
NE=11; S=OTCH; if (CODE2()==0) return 0;
return 1; // Выход без обнаружения ошибок
} // End of function CONV2()


function CODE2() // Формирования элемента данных по форме Lxx10110...10,
{
j=S.length; F=F+'L'+NU2[j]; // где xx - к-во байт
```

```
var i, j, k;
var n = [36]; // Массив чисел, соответствующих символам данных
      for (i=0; i<j; i++)
      switch (S.substr(i,1))
      {
      case ' ': n[i]=32; break;
      case "'": if (NE<9) return 0; n[i]=39; break;
      case '+': if (NE!=5) return 0; n[i]=43; break;
      case ',': n[i]=44; break;
      case '-': n[i]=45; break;
      case '.': n[i]=46; break;
      case '0': n[i]=48; break;
      case '1': n[i]=49; break;
      case '2': n[i]=50; break;
      case '3': n[i]=51; break;
      case '4': n[i]=52; break;
      case '5': n[i]=53; break;
      case '6': n[i]=54; break;
      case '7': n[i]=55; break;
      case '8': n[i]=56; break;
      case '9': n[i]=57; break;
      case '<': n[i]=60; break;
      case '>': n[i]=62; break;
      case '@': if (NE>4 || NE<3) return 0; n[i]=64; break;
      case 'A': if (NE>4) return 0; n[i]=65; break;
      case 'B': if (NE>4) return 0; n[i]=66; break;
      case 'C': if (NE>4) return 0; n[i]=67; break;
      case 'D': if (NE>4) return 0; n[i]=68; break;
      case 'E': if (NE>4) return 0; n[i]=69; break;
      case 'F': if (NE>4) return 0; n[i]=70; break;
      case 'G': if (NE>4) return 0; n[i]=71; break;
      case 'H': if (NE>4) return 0; n[i]=72; break;
      case 'I': if (NE>4) return 0; n[i]=73; break;
      case 'J': if (NE>4) return 0; n[i]=74; break;
      case 'K': if (NE>4) return 0; n[i]=75; break;
      case 'L': if (NE>4) return 0; n[i]=76; break;
      case 'M': if (NE>4) return 0; n[i]=77; break;
      case 'N': if (NE>4) return 0; n[i]=78; break;
      case 'O': if (NE>4) return 0; n[i]=79; break;
      case 'P': if (NE>4) return 0; n[i]=80; break;
      case 'Q': if (NE>4) return 0; n[i]=81; break;
      case 'R': if (NE>4) return 0; n[i]=82; break;
      case 'S': if (NE>4) return 0; n[i]=83; break;
      case 'T': if (NE>4) return 0; n[i]=84; break;
      case 'U': if (NE>4) return 0; n[i]=85; break;
      case 'V': if (NE>4) return 0; n[i]=86; break;
      case 'W': if (NE>4) return 0; n[i]=87; break;
      case 'X': if (NE>4) return 0; n[i]=88; break;
      case 'Y': if (NE>4) return 0; n[i]=89; break;
      case 'Z': if (NE>4) return 0; n[i]=90; break;
      case '_': if (NE>4) return 0; n[i]=95; break;
      case 'a': if (NE>4) return 0; n[i]=97; break;
      case 'b': if (NE>4) return 0; n[i]=98; break;
      case 'c': if (NE>4) return 0; n[i]=99; break;
      case 'd': if (NE>4) return 0; n[i]=100; break;
      case 'e': if (NE>4) return 0; n[i]=101; break;
      case 'f': if (NE>4) return 0; n[i]=102; break;
      case 'g': if (NE>4) return 0; n[i]=103; break;
      case 'h': if (NE>4) return 0; n[i]=104; break;
      case 'i': if (NE>4) return 0; n[i]=105; break;
      case 'j': if (NE>4) return 0; n[i]=106; break;
      case 'k': if (NE>4) return 0; n[i]=107; break;
      case 'l': if (NE>4) return 0; n[i]=108; break;
      case 'm': if (NE>4) return 0; n[i]=109; break;
```

```
case 'n': if (NE>4) return 0; n[i]=110; break;
case 'o': if (NE>4) return 0; n[i]=111; break;
case 'p': if (NE>4) return 0; n[i]=112; break;
case 'q': if (NE>4) return 0; n[i]=113; break;
case 'r': if (NE>4) return 0; n[i]=114; break;
case 's': if (NE>4) return 0; n[i]=115; break;
case 't': if (NE>4) return 0; n[i]=116; break;
case 'u': if (NE>4) return 0; n[i]=117; break;
case 'v': if (NE>4) return 0; n[i]=118; break;
case 'w': if (NE>4) return 0; n[i]=119; break;
case 'x': if (NE>4) return 0; n[i]=120; break;
case 'y': if (NE>4) return 0; n[i]=121; break;
case 'z': if (NE>4) return 0; n[i]=122; break;

case 'А': n[i]=64; break;
case 'Б': n[i]=65; break;
case 'В': n[i]=66; break;
case 'Г': n[i]=67; break;
case 'Д': n[i]=68; break;
case 'Е': n[i]=69; break;
case 'Ж': n[i]=70; break;
case 'З': n[i]=71; break;
case 'И': n[i]=72; break;
case 'Й': n[i]=73; break;
case 'К': n[i]=74; break;
case 'Л': n[i]=75; break;
case 'М': n[i]=76; break;
case 'Н': n[i]=77; break;
case 'О': n[i]=78; break;
case 'П': n[i]=79; break;
case 'Р': n[i]=80; break;
case 'С': n[i]=81; break;
case 'Т': n[i]=82; break;
case 'У': n[i]=83; break;
case 'Ф': n[i]=84; break;
case 'Х': n[i]=85; break;
case 'Ц': n[i]=86; break;
case 'Ч': n[i]=87; break;
case 'Ш': n[i]=88; break;
case 'Щ': n[i]=89; break;
case 'Ъ': n[i]=90; break;
case 'Ы': n[i]=91; break;
case 'Ь': n[i]=92; break;
case 'Э': n[i]=93; break;
case 'Ю': n[i]=94; break;
case 'Я': n[i]=95; break;
case 'а': n[i]=96; break;
case 'б': n[i]=97; break;
case 'в': n[i]=98; break;
case 'г': n[i]=99; break;
case 'д': n[i]=100; break;
case 'е': n[i]=101; break;
case 'ж': n[i]=102; break;
case 'з': n[i]=103; break;
case 'и': n[i]=104; break;
case 'й': n[i]=105; break;
case 'к': n[i]=106; break;
case 'л': n[i]=107; break;
case 'м': n[i]=108; break;
case 'н': n[i]=109; break;
case 'о': n[i]=110; break;
case 'п': n[i]=111; break;
case 'р': n[i]=112; break;
case 'с': n[i]=113; break;
```

```
      case 'т': n[i]=114; break;
      case 'у': n[i]=115; break;
      case 'ф': n[i]=116; break;
      case 'х': n[i]=117; break;
      case 'ц': n[i]=118; break;
      case 'ч': n[i]=119; break;
      case 'ш': n[i]=120; break;
      case 'щ': n[i]=121; break;
      case 'ъ': n[i]=122; break;
      case 'ы': n[i]=123; break;
      case 'ь': n[i]=124; break;
      case 'э': n[i]=125; break;
      case 'ю': n[i]=126; break;
      case 'я': n[i]=127; break;
      case 'Є': n[i]=58; break;
      case 'є': n[i]=59; break;
      case 'Ї': n[i]=60; break;
      case 'ї': n[i]=61; break;
      case 'І': n[i]=62; break;
      case 'і': n[i]=63; break;
      default: return 0;
      }
      var n7=[252]; // Массив битов (чисел 1 или 0), соответствующих байтам
массива n[]
      for (i=0; i<j*7; i++) n7[i]=0;// Обнулили битовый массив n7[] для
шифрования элемента данных
      for (k=0; k<j; k++) // Цикл преобразования элемента БД из n[] в n7[]
      { i=k;
      if (n[i]>63) {n7[k*7+6]=1; n[i]=n[i]-64;}
      if (n[i]>31) {n7[k*7+5]=1; n[i]=n[i]-32;}
      if (n[i]>15) {n7[k*7+4]=1; n[i]=n[i]-16;}
      if (n[i]>7)  {n7[k*7+3]=1; n[i]=n[i]-8;}
      if (n[i]>3)  {n7[k*7+2]=1; n[i]=n[i]-4;}
      if (n[i]>1)  {n7[k*7+1]=1; n[i]=n[i]-2;}
                          n7[k*7]=n[i];
      }
      NC=j*7;
      if (NC1+NC>503) NC1=1;
      for (i=0; i<NC; i++) // Цикл шифрования по алгоритму Вернама
      {
      n7[i]=n7[i]+C[i+NC1];
      if (n7[i]>1) n7[i]=0;
      if (n7[i]==1) F=F+'1'; else F=F+'0';
      }
      NC1=NC1+NC;
      return 1; // Выход без обнаружения ошибок
} // End of function CONV2()

function TRAN2()   // Передача зашифрованных данных об избирателе в БД
{
TR="/Q2"+F; // Сформировали отправляемую строку данных
var xmlhttp = getXmlHttp()     // Асинхронно отправляем на сервер данные клиента
через TR
xmlhttp.open('GET', TR,  true); // Открываем соединение для отправки данных на
сервер
xmlhttp.onreadystatechange = function()
{
  if (xmlhttp.readyState == 4)
  {
     if(xmlhttp.status == 200)
       {
        RT=xmlhttp.responseText;  // Ответные данные от сервера заносим в RT
        if (RT[0]=='E')  // Определяем верность данных по первой букве
        {
```

```
            switch (RT[1])
            {
            case '0': alert("Дані занесено.\nЗробіть перезавантаження
сторінки"); break;
            default: alert("Сервер зайнятий."); break;
            }
            return;
         }
      }
        else {alert("Лінію зайнято. Почніть спочатку пізніше.");}
  }
};
xmlhttp.send(null);
} // End of function TRAN2{)

function NEWIN()   // Новое обращение к БД
{ alert("Зробіть повне перезавантаження сторінки");
} // End of function NEWIN{)
</script>
<div>
<select id="REJIM">
<option value="0">Реєстрація виборця</option>
<option value="1">Анулювання виборця</option>
<option  disabled value="2">Корегування даних виборця</option>
<option value="3">Знищення помилкового запису</option>
<option value="4">Відмова виборця від реєстрації</option>
</select>
</div>
<br>

 <table border="1"   bgcolor="#e0e0e0">
   <caption> </caption>
   <tr>
    <th>Введіть паспортні дані</th>
    </tr>
   <tr><td>Серія: <input id="SER" type="text" size="1" pattern="[A-ZA-Я\s]{2}" >
</td></tr>
   <tr><td>Номер:   <input type="text" size="5" pattern="[0-9]{6}"
id="NUMPASP"></td></tr>
  <tr><td align="center">    <input type="button" value="Дані введено"
onClick=TAB1()> </td></tr>
  </table>
<br>
<table border="1"  bgcolor="#e0e0e0">
   <caption> </caption>
   <tr>
    <th>Заповнюйте після запрошення</th>
    </tr>
   <tr><td>*Пароль<input type="password" pattern="[A-Za-z0-9\s,.<>]{10,16}"
id="PAROL"></td></tr>
   <tr><td>*Ще раз<input type="password" pattern="[A-Za-z0-9\s,.<>]{10,16}"
id="REPEAT"></td></tr>
   <tr><td>E-mail <input type="text"  id="EMAIL_D"></td></tr>
   <tr><td>Вулиця
<div>
   <select id="VUL">
<option value="0">Докучаївська</option>
<option value="1">Волгоградська</option>
</select>
</div>
   </td></tr>
   <tr><td>Номер будинку<input type="text" id="NUMBUD"></td></tr>
   <tr><td>Номер квартири<input type="text" id="NUMKV"></td></tr>
   <tr><td>Прізвище<input type="text" id="FAMIL"></td></tr>
```

```
   <tr><td>Ім'я<input type="text" id="IMJA"></td></tr>
   <tr><td>По-батькові<input type="text" id="OTCH"></td></tr>
   <tr><td>Дата народження<input type="text" id="OTCH"></td></tr>
  <tr><td align="center">    <input type="button" value="Дані введено"
onClick=TAB2()> </td></tr>
  </table>
</body>
</html>
```

Д1.4. Серверна програма виборчої дільниці

(файл SVD.js)

```
//  SVD Server Vyborchoi Dilnici Ver. 3 June 2015
////////////// Параметри, що потребують налаштування //////////////
var EMAILSERV="gmail";  // Ім'я сервісу для електонної пошти виборчої дільниці
var EMAILNAME="vybir800876@gmail.com";  // Електонна пошта виборчої дільниці
var EMAILPASS="xxxxxxx";  // Пароль електонної пошти виборчої дільниці
(замінюється адміністратором)
var TCPW=120; // Тривалість періоду введення паролю для голосування
виборцями(годин)
var TPAUSE=24; // Тривалість паузи (годин) для друку списку голосуючих та
завантаження бюлетенів
var TGOL=12; // Тривалість періоду для голосування (годин)
var DILN='№ 800876'; // Назва виборчої дільниці
var KVUL=3; // Кількість вулиць на виборчій дільниці
var VUL = [3]; // Назви вулиць
    VUL[0]='Волгоградська';
      VUL[1]='Івана Неходи';
      VUL[2]='Волгоградська';
////////////// Кінець параметрів, що потребують налаштування //////////////
var FLAG_END=0; // Ознака завершення виборів (1-вибори завершено)
var NPERIOD=1; //  Встановили номер періоду (1-введення паролів виборцями, 2-
пауза, 3-голосуванння)
var NAMEHTML="/CPW.html";  // Назва клієнтської програми (встановлюємо для 1-го
періоду)
// NAMEHTML="/anketa.html"; Це для 3-го периоду (NPERIOD=3)
// Підключаємо бібліотеки node.js
var http = require('http');
var url = require('url');
var fs = require('fs');
var static = require('node-static');
var querystring = require('querystring');
var nodemailer = require('nodemailer');
var file = new static.Server('.');
var i,j;  // Числові змінні для циклів
var KBUL=0; //Кількість варіантів бюлетенів для голосування
var PASSVYB = [2500]; // Паролі для голосування після заміни (більше ніж 2500
виборців не буває)
 for( i=0; i<2500; i++) PASSVYB[i]=' '; // Для початку заносимо пропуски
var PASSINP = [2500]; // Паролі, що були введені під час голосування
 for( i=0; i<2500; i++) PASSINP[i]=' '; // Для початку заносимо пропуски
var PRAVOG = [2500]; // Ознака заміни паролю (1-має право голосувати, 0-ні)
 for( i=0; i<2500; i++) PRAVOG[i]=0; // Для початку заносимо нулі
var GOLOSF = [2500]; // Ознака голосування з фіктивним або вірним паролем  (1-
голосував, 0-ні)
 for( i=0; i<2500; i++) GOLOSF[i]=0; // Для початку заносимо нулі
var GOLOS = [2500]; // Ознака голосування з  вірним паролем  (1-голосував, 0-ні)
 for( i=0; i<2500; i++) GOLOS[i]=0; // Для початку заносимо нулі
var GOL=new Array(10);  // Двовимірний масив GOL[][] для підрахунку голосів
 for( i=0; i<10; i++) GOL[i] = new Array(100); // GOL[номер бюлетеня][номер
пункту в бюлетені]
 for (i=0;i<10;i++) for (j=0;j<100;j++) GOL[i][j]=0; // Для початку заносимо
нулі GOL[][]
 // Резервуємо змінні для множення елементів поля Галуа GF(2^503)
 // Молодший біт в [1], старший - в [503]. Нульові елементи не використовуємо
var M1 = [504];
var M2 = [504];   // M1[], M2[] - множники  R[] - результат множення
var R  = [504];
var DATA=[20]; // Объявляем массив байт для расшифровки пароля
// Переменные для работы с базой данных участников голосования. Файл VYBORCI.DBT
var SFILE='';// Строка для чтения и работы с файлом базы данных
var Kstr=0;  // К-во строк данных
```

```
var Lstr=0;  // Длина строки данных (включая первый байт-признак)
var Nstr=0;  // Номер текущей строки (начиная с нулевой)
var SNL=0; // Номер строки с данными обслуживаемого клиента
var EEE=0;              // Признак наличия избирателя в базе ( 0-нет, 1-есть)
// Переменные в строке базы данных (в начальном байте строки всегда пробел)
// В скобках указаны (номер начального байта, к-во байт)
var OZNAKA='0';       // (1,1) 0-действующая строка данных, 1-выбыл, 2-данные
заменены, 3-ошибочная строка, 4-самоотказ избирателя
var SERPASP='AA';     // (2,2) Серия паспорта
var NUMPASP='000000'; // (4,6) Номер паспорта
var PAROL='a12345';   // (10,20) Пароль для доступа клиента на сервер
var EMAIL_R='@';    // (34,36) Адрес электронной почты из ДРВ
var EMAIL_D='@';    // (70,36) Адрес электронной почты для отправки кода
var TELEFON='+380';   // (106,13) Номер мобильного телефона
var KODVUL='000';         // (119,3)  Код улицы избирателя (таблица кодов
составляется на каждом участке)
var NUMBUD='0000';    // (122,4)  Номер дома избирателя
var NUMKV='0000';         // (126,4)  Номер квартиры избирателя
var FAMIL='Ф';            // (130,30) Фамилия
var IMJA='И';             // (160,20) Имя
var OTCH='Б';             // (180,20) Отчество
var DATE_IN='20141028'; // (200,8) Дата ввода данных
var TIME_IN='1939';     // (208,4) Время ввода данных
var DATE_OUT='20141028';// (212,8) Дата удаления данных
var TIME_OUT='1939';    // (220,4) Время удаления данных
var PRAVO='1';                // (224,1) Признак наличия права участия в данных
выборах ("1"-есть, "0"-нет)
var VYBOR='0';                // (225,1) Признак использования права выбора
("1"-проголосовал, "0"-нет)
////////////////////////////////////////////////////////////////////////////////
////
SFILE=fs.readFileSync('VYBORCI.DBT');   // Чтение файла данных в строку
            Lstr=226;  // Длина строки данных (включая первый пробел)
            Kstr=SFILE.length/Lstr;             // К-во строк данных
            Nstr=0;  // Номер текущей строки (начиная с нулевой)
////////////////////////////////////////////////////////////////////////////////
/////////////
var T1 = new Date(); // Берем метку времени для отсчета периода голосования от
стартового момента
var TBG = T1.getTime(); // TBG - момент начала периода в миллисекундах от
01.01.1970
var TEND=TBG+3600000*TCPW;   // Установили момент окончания периода для замены
пароля
//var TEND=TBG+3600000*TGOL;   // Установили период для голосования (пока 1 час)
----------------------
////////////////////////////////////////////////////////////////////////////////
//////////////
function MULT()     // Умножение элементов поля Галуа GF(2^503) по правилу
полиномов
{
 var i,j,r,r1,r2,r3;
    for (i=1;i<=503;i++) R[i]=0; // Обнуляем 503 ячейки для битов результата
    for (i=1;i<=503;i++)           // Начинаем двойной цикл умножения элементов
полинома
    if (M1[i]==1) // Умножаем только единичные элементы из M1[], нулевые не
нужны, т.к. они дают нули
    {
        for (j=1;j<=503;j++)
        if (M2[j]==1)
        {
        r=i+j-1;
            if (r>503)
            {
             r=r-503;
```

```
             if (r>=501)
                {
                r=r-501;
                r1=1+r; r2=4+r; r3=501+r;
                if (R[r3]==0) R[r3]=1; else R[r3]=0;
                }
             else {r1=r; r2=r+3;}
             if (R[r1]==0) R[r1]=1; else R[r1]=0;
             if (R[r2]==0) R[r2]=1; else R[r2]=0;
             }
             else {if (R[r]==0) R[r]=1; else R[r]=0;}
         }
     }
} // End of function MULT()
var i,j,I,J;
var CYF='0123456789';
var NU2 = [100]; // Массив для преобразование чисел 0-99 в строку из двух
символов
 for(i=0; i<10; i++)
 {
 for(j=0; j<10; j++) NU2[i*10+j]=CYF[i]+CYF[j];
 }
var A = [504];    // Массив для получения от клиента значения X^N[], где X-
примитивный элемент поля Галуа
var M = new Array(504); // Создаем двумерный массив M[][] для степеней
примитивного элемента
  for(i=0; i<504; i++) M[i] = new Array(504);
var MA = new Array(504);      // Создаем двумерный массив MA[][] для степеней
A[] (массива битов, полученных от клиента)
  for(i=0; i<504; i++) MA[i] = new Array(504);
var NL=0; // Номер строки данных для очередного клиента, в которой подготовлены
случайные биты NLN[NL][] и массив NLB[NL][]
// Всего резервируем 100 строк для одновременного обслуживания 100 клиентов
// В эту строку входят PZ[NL], NQ[NL], NLN[NL][], NLB[NL][], CODE[NL], NC[NL],
NC1[NL], TIO[NL], NstrK[NL]
var NR= [100];          // Номер режима обслуживания клиента (0-голосование, 1,
2, 3 - получение справки)
var PZ= [100];      // Признак занятости строки
for(i=0; i<100; i++) PZ[i]=0;
var NQ= [100];      // Номер ожидаемого запроса клиента
for(i=0; i<100; i++) NQ[i]=0;
var NLN = new Array(100);     // Создаем двумерный массив NLN[][] для случайных
битов, а потом битов для шифрования
  for(var i=0; i<100; i++)
    NLN[i] = new Array(504);
var NLB = new Array(100);     // Создаем двумерный массив NLB[][] для значений
X^NLN[][], а потом для расшифрованых данных
  for(var i=0; i<100; i++)
    NLB[i] = new Array(504);
 var CODE=[100];   // Код для отправки на Email
 var NC= [100];      // К-во зашифрованных бит для передачи или приема
 var NC1=[100];      // Номер бита, с которого следует начинать/продолжать
процедуру шифрования
 for(i=0; i<100; i++) NC1[i]=1;
 var TIO=[100];    // Значения таймаутов
 var NstrK=[100];        // Значения Nstr (номер текущей строки в БД)
облуживаемого клиента
var T1 = new Date(); // Берем метку времени для шифрования + для проверки
времени выполнения
var TB = T1.getTime(); // TB - момент начала преобразований в милисекундах от
01.01.1970
var TN=TB;                    // Проверяем на четность к-во миллисекунд от
01.01.1970
```

```
var TD = TN%2;          // В зависимости от четности заполняем первый случайный
бит в текущей строке
      if (TD>0) NLN[NL][1]=1; else  NLN[NL][1]=0;

    for (var i=1;i<=503;i++) M[1][i] = 0; // Обнулили первую строку массива для
примитивного элемента
    M[1][2]=1; // Занесли единицу во второй бит. Это получился примитивный
элемент поля Галуа.
    for (I=2;I<=503;I++) // Начинаем цикл заполнения массива M[][] константами,
а NLN[NL][] случайными битами
    {    for (J=1;J<=503;J++)
        {
           M1[J]= M2[J]=M[I-1][J]; // Оба сомножителя равны предыдущей строке
        }
    MULT(); // Каждая следующая строка массива M[][] равна квадрату предыдущей
строки
      for (j=1;j<=503;j++) M[I][j]=R[j]; // Занесли результат умножения в
очередную строку
      var T1 = new Date();                 // Начало вычисления случайного бита
      var TN = T1.getTime();              //    Берем к-во миллисекунд от
01.01.1970
      var TD = TN%2; //  В TD вычисляем признак четности к-ва миллисекунд
      if (TD>0) NLN[NL][I]=1; else NLN[NL][I]=0;  // Случайный бит зависит от
четности к-ва миллисекунд
      var NN = Math.random();             // и величины стандартного случайного
числа (сумма по модулю 2)
      if (NN>0.5) NLN[NL][I]=NLN[NL][I]+1; if (NLN[NL][I]>1) NLN[NL][I]=0; //
Заполнили очередной случайный бит
    } // Конец цикла заполнения массива константами
function STEPX()     // Вычисление случайной степени примитивного элемента X
{ //---- Вычисление A[]=X[]^N[], где X[]- примитивный элемент поля Галуа, N[]-
503 наших случайных бит
    for (i=1;i<=503;i++) A[i]=0; A[1]=1; // Занесли в A[] единицу
    for (J=1;J<=503;J++) // Начинаем двойной цикл возведения в степень
        if (NLN[NL][J]==1)
        {
            for (I=1;I<=503;I++)
            {
                M1[I]= M[J][I];
                M2[I]=A[I];
            }
            MULT();
            for (I=1;I<=503;I++)
              A[I]=R[I];
        }  // Конец двойного цикла возведения в степень.
            for (I=1;I<=503;I++) NLB[NL][I]=R[I]; //Результат перенесли в
NLB[NL][]
} // End of function STEPX()
STEPX(); // Вычисление случайной степени примитивного элемента X в строке данных
клиента
// Будем отправлять значение NLB[NL][] клиенту
var TR='N';  // Строка для отправки данных клиенту
var RT='/';  // Строка для приема данных от клиента
////////////////////////////////////////////////////////////////////////////
/
http.createServer(function (req, res) // req - запрос от клиента, res - ответ
сервера на запрос клиента
{
 var pathname = url.parse(req.url).pathname;
    console.log("Request for " + pathname + " received."); //-/-/-/-/-/-/-/-/-/-
/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
var T1 = new Date(); // Берем метку времени для проверки таймаута
if (TEND<T1.getTime()) ENDPER(); // если период закончился
if (pathname==NAMEHTML) file.serve(req, res);
```

```
else {  RT=pathname;
      if (RT[1]=='Q')
      {
if (NPERIOD==1)   //>>>>>>>>>> Початок обробки першого періоду  (введення
паролів виборцями) <<<<<<<<<<<<<<<<
{
switch (RT[2])
      {
      case '0': // Будем отправлять новому клиенту номер его строки и значение
NLB[NL][], если есть пустая строка
            CLSTIO(); // Очистка строк данных для клиентов по таймауту
            PZ[NL]=1; // Установили признак занятости строки и будем заполнять
начальные значения всех параметров
            var T1 = new Date(); // Берем метку времени для шифрования + для
проверки времени выполнения
            var TB = T1.getTime(); // TB - момент начала преобразований в
миллисекундах от 01.01.1970
            TIO[NL]=TB+300000;     // Установили таймаут для заполнения анкеты
(5 минут)
            TR='N'+NU2[NL];  // Строка для передачи данных на сервер
            for (i=1;i<=503;i++) // Цикл переноса значения NLB[NL][] в
транспортную строку
            {
            if (NLB[NL][i]==0) TR=TR+'0'; else TR=TR+'1';
            }
            var NNL=0; // Номер пустой строки для данных следующего клиента
            while ( PZ[NNL]==1 && NNL<100) NNL++; // Поиск пустой строки

            if (NNL==100) TR='/'; // Если пустая строка не найдена, то
отправляем "липовые" данные
            else // Вычисляем биты шифрования для текущего клиента и готовим
строку для данных нового клиента
            {
            NQ[NL]=1; // Установили номер следующего запроса от текущего клиента
            //---- Вычисление послеовательностм битов для шифрования сообщений
            for (i=1;i<=503;i++) A[i]=RT[i+2]; // Занесли принятый от клиента
массив битов в A[]
            var T1 = new Date(); // Берем метку времени для получения случайного
бита
            var TD = T1.getTime()%2; // Проверяем на четность к-во миллисекунд
от 01.01.1970
            if (TD>0) NLN[NNL][1]=1; else  NLN[NNL][1]=0; // В зависимости от
четности заполняем первый случайный бит
      for (i=1;i<=503;i++) MA[1][i] = A[i]; // В первую строку массива занесли
значение A[] в степени 1
            for (I=2;I<=503;I++) // Начинаем цикл заполнения массива MA[][]
степенями A[], а NLN[NNL][] случайными битами
            {
            for (J=1;J<=503;J++) M1[J]=M2[J]=MA[I-1][J]; // Оба сомножителя
равны предыдущей строке
            MULT(); // Каждая следующая строка массива MA[][] равна квадрату
предыдущей строки
                  for (j=1;j<=503;j++) MA[I][j]=R[j]; // Занесли результат
умножения в очередную строку
                  var T1 = new Date(); // Берем метку времени для получения
случайного бита
                  var TD = T1.getTime()%2; // Проверяем на четность к-во
миллисекунд от 01.01.1970
                  if (TD>0) NLN[NNL][I]=1; else NLN[NNL][I]=0;  // Случайный бит
зависит от четности к-ва миллисекунд
                  var NN = Math.random();             // и величины стандартного
случайного числа (сумма по модулю 2)
                  if (NN>0.5) NLN[NNL][I]=NLN[NNL][I]+1; if (NLN[NNL][I]>1)
NLN[NNL][I]=0; // Заполнили очередной случайный бит
```

```
            } // Конец цикла заполнения массива степенями A[]
            for (i=1;i<=503;i++) A[i]=0; A[1]=1; // Занесли в A[] единицу
      for (J=1;J<=503;J++) // Начинаем двойной цикл возведения в степень
        if (NLN[NL][J]==1)
        {
            for (I=1;I<=503;I++)
            {
                M1[I]= MA[J][I];
                M2[I]=A[I];
            }
            MULT();
            for (I=1;I<=503;I++)
              A[I]=R[I];
        }  // Конец двойного цикла возведения в степень. Результат занесен в A[]

            // --- В A[] получена случайная послеовательность для шифрования
сообщений
      //    var STR='/'; ////////////////////////////////////////////////-/-/-/-/
      //    for (i=1;i<=503;i++) STR=STR+A[i];//////////////////////-/-/-/-/-/-
/-/-/-/-/-/--/-/-/
      //    console.log("C="+STR);///////////////////////////////////////-/-/-/
            for (i=1;i<=503;i++) NLN[NL][i]=A[i]; // Перенесли эту
последовательность в строку данных клиента  NLN[NL][]
            NL=NNL;      // Установили номер строки данных для следующего
клиента (эта строка уже подготовлена)
            STEPX(); // Вычисление случайной степени примитивного элемента X в
строке данных следующего клиента
            } // Подготовили в TR биты (массив B) для клиента и подготовили
строку для данных следующего клиента
            res.end(TR);            // Отправили данные клиенту
      break;
      case '1': // Будем расшифровывать и проверять паспортные данные клиента
            var S2='' // Строка для поиска двухбайтного номера строки в массиве
NU2[]
console.log(" RT[[1]="+RT[1]+" RT[2]="+RT[2]+" RT[3]="+RT[3]+" RT[4]="+RT[4]+"
RT[5]="+RT[5] );/////////////////////////////////////-/-/-/
            S2=RT[4]+RT[5]; // Занесли двухбайтный номер строки обслуживаемого
клиента в S2
console.log("S2="+S2);/////////////////////////////////////-/-/-/-/-/-/-/-/-/-/-
/-/-/-/-/-/-/-/-/-/-/-/-/
            var SNL=0; // Номер строки с данными обслуживаемого клиента
            while (NU2[SNL]!=S2) SNL++; // Поиск номера строки обслуживаемого
клиента
console.log("SNL="+SNL+" NU2[SNL]="+NU2[SNL]+" NQ[SNL]="+NQ[SNL]
);//////////////////////////////////-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
/-/-/
            if (PZ[SNL]!=1 || (PZ[SNL]==1 && NQ[SNL]!=1)) TR='E4';
            // Если строка пустая то отправляем код сообщения "Время истекло"
            else
            {
            NC[SNL]=RT.length-6; // Вычислили к-во символов шифротекста

            for (i=NC1[SNL];i<NC1[SNL]+NC[SNL];i++) //  Цикл расшифрования по
алгоритму Вернама
            {j=0; if (RT[i+5]=='1') j=1; NLB[SNL][i]=j+NLN[SNL][i]; if
(NLB[SNL][i]>1) NLB[SNL][i]=0;
            } // Результат расшифровки занесли в NLB[SNL][]
            var r=NC1[SNL];
            var SP='';
            for (i=0; i<2; i++) // Цикл получения серии паспорта
            {
            j=0;
```

```
//console.log("r="+r+" NLB[SNL][1]="+NLB[SNL][1]+" NLB[SNL][2]="+NLB[SNL][2]+"
NLB[SNL][3]="+NLB[SNL][3]+" NLB[SNL][4]="+NLB[SNL][4]
);////////////////////////////////////-/-/-/
            if (NLB[SNL][r]==1) j=j+1;
            if (NLB[SNL][r+1]==1)j=j+2;
            if (NLB[SNL][r+2]==1) j=j+4;
            if (NLB[SNL][r+3]==1) j=j+8;
            r=r+4;
//console.log("j="+j);////////////////////////////////////-/-/-/-/-/-/
            switch (j)
            {
            case 1: SP=SP+'A'; break; // Латинская кодировка букв
            case 2: SP=SP+'B'; break;
            case 3: SP=SP+'C'; break;
            case 4: SP=SP+'E'; break;
            case 5: SP=SP+'H'; break;
            case 6: SP=SP+'I'; break;
            case 7: SP=SP+'K'; break;
            case 8: SP=SP+'M'; break;
            case 9: SP=SP+'O'; break;
            case 10: SP=SP+'P'; break;
            case 11: SP=SP+'T'; break;
            case 12: SP=SP+'X'; break;
            }
            }
            SERPASP=SP;
            var SP='';
            for (i=0; i<6; i++) // Цикл получения номера паспорта
            {
            j=0;
            if (NLB[SNL][r]==1) j=j+1;
            if (NLB[SNL][r+1]==1)j=j+2;
            if (NLB[SNL][r+2]==1) j=j+4;
            if (NLB[SNL][r+3]==1) j=j+8;
            r=r+4;
            switch (j)
            {
            case 1: SP=SP+'1'; break;
            case 2: SP=SP+'2'; break;
            case 3: SP=SP+'3'; break;
            case 4: SP=SP+'4'; break;
            case 5: SP=SP+'5'; break;
            case 6: SP=SP+'6'; break;
            case 7: SP=SP+'7'; break;
            case 8: SP=SP+'8'; break;
            case 9: SP=SP+'9'; break;
            case 10: SP=SP+'0'; break;
            }
            }
            NUMPASP=SP;
// Получение номера режима NR[SNL] облслуживания клиента (0-Введення паролю для
голосування,
            // 1-Введення паролю дільничної пошти)
            j=0;
            if (NLB[SNL][r]==1) j=j+1;
            if (NLB[SNL][r+1]==1)j=j+2;
            if (NLB[SNL][r+2]==1) j=j+4;
            if (NLB[SNL][r+3]==1) j=j+8;
            r=r+4;
            NR[SNL]=j;
console.log("NR="+j);////////////////////////////////////-/-/-/-/-/-/

            var L=(NC[SNL]-36)/7; // Длину зашифрованного пароля занесли в L
console.log("L="+L);////////////////////////////////////-/-/-/-/-/-/
```

```
var PW='';
for (i=0; i<L; i++) // Цикл получения символов пароля
{
j=0;
if (NLB[SNL][r]==1) j=j+1;
if (NLB[SNL][r+1]==1)j=j+2;
if (NLB[SNL][r+2]==1) j=j+4;
if (NLB[SNL][r+3]==1) j=j+8;
if (NLB[SNL][r+4]==1) j=j+16;
if (NLB[SNL][r+5]==1) j=j+32;
if (NLB[SNL][r+6]==1) j=j+64;
r=r+7;
switch (j)
{
case 32: PW=PW+' '; break; // 32
case 44: PW=PW+','; break; // 44
case 46: PW=PW+'.'; break; // 46
case 48: PW=PW+'0'; break; // 48
case 49: PW=PW+'1'; break; // 49
case 50: PW=PW+'2'; break; // 50
case 51: PW=PW+'3'; break; // 51
case 52: PW=PW+'4'; break; // 52
case 53: PW=PW+'5'; break; // 53
case 54: PW=PW+'6'; break; // 54
case 55: PW=PW+'7'; break; // 55
case 56: PW=PW+'8'; break; // 56
case 57: PW=PW+'9'; break; // 57
case 60: PW=PW+'<'; break; // 60
case 62: PW=PW+'>'; break; // 62
case 65: PW=PW+'A'; break; // 65
case 66: PW=PW+'B'; break; // 66
case 67: PW=PW+'C'; break; // 67
case 68: PW=PW+'D'; break; // 68
case 69: PW=PW+'E'; break; // 69
case 70: PW=PW+'F'; break; // 70
case 71: PW=PW+'G'; break; // 71
case 72: PW=PW+'H'; break; // 72
case 73: PW=PW+'I'; break; // 73
case 74: PW=PW+'J'; break; // 74
case 75: PW=PW+'K'; break; // 75
case 76: PW=PW+'L'; break; // 76
case 77: PW=PW+'M'; break; // 77
case 78: PW=PW+'N'; break; // 78
case 79: PW=PW+'O'; break; // 79
case 80: PW=PW+'P'; break; // 80
case 81: PW=PW+'Q'; break; // 81
case 82: PW=PW+'R'; break; // 82
case 83: PW=PW+'S'; break; // 83
case 84: PW=PW+'T'; break; // 84
case 85: PW=PW+'U'; break; // 85
case 86: PW=PW+'V'; break; // 86
case 87: PW=PW+'W'; break; // 87
case 88: PW=PW+'X'; break; // 88
case 89: PW=PW+'Y'; break; // 89
case 90: PW=PW+'Z'; break; // 90
case 97: PW=PW+'a'; break; // 97
case 98: PW=PW+'b'; break; // 98
case 99: PW=PW+'c'; break; // 99
case 100: PW=PW+'d'; break; // 100
case 101: PW=PW+'e'; break; // 101
case 102: PW=PW+'f'; break; // 102
case 103: PW=PW+'g'; break; // 103
case 104: PW=PW+'h'; break; // 104
case 105: PW=PW+'i'; break; // 105
```

```
            case 106: PW=PW+'j'; break; // 106
            case 107: PW=PW+'k'; break; // 107
            case 108: PW=PW+'l'; break; // 108
            case 109: PW=PW+'m'; break; // 109
            case 110: PW=PW+'n'; break; // 110
            case 111: PW=PW+'o'; break; // 111
            case 112: PW=PW+'p'; break; // 112
            case 113: PW=PW+'q'; break; // 113
            case 114: PW=PW+'r'; break; // 114
            case 115: PW=PW+'s'; break; // 115
            case 116: PW=PW+'t'; break; // 116
            case 117: PW=PW+'u'; break; // 117
            case 118: PW=PW+'v'; break; // 118
            case 119: PW=PW+'w'; break; // 119
            case 120: PW=PW+'x'; break; // 120
            case 121: PW=PW+'y'; break; // 121
            case 122: PW=PW+'z'; break; // 122
            }
            }
            NC1[SNL]=NC1[SNL]+NC[SNL]; // Переустановили номер бита, с которого
следует начинать/продолжать процедуру шифрования
            PAROL=PW;
console.log(SERPASP+NUMPASP+' PW: '+PW); /////-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/--
/-/-/-/-/-/-/-/-/
// Проверяем наличие/отсутствие в базе данных по избирателю и формируем ответ
('E0' ... 'E4' - Error или 'S...ответ на пароль... )
            EEE=0; // Признак наличия избирателя в базе ( 0-нет, 1-есть)
            if (Kstr>0)
            {
console.log("Kstr="+Kstr);  /////-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/--/-/-/-/-/-/-
/-/-/
//console.log("SFILE="+SFILE);  /////-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/--/-/-/-/-
/-/-/-/-/
            var SERNUM='';
            for (i=0; i<Kstr; i++) // Цикл формирования строки с данными
паспорта
            {
            SERNUM=String.fromCharCode(SFILE[i*Lstr+1])+
                  String.fromCharCode(SFILE[i*Lstr+2])+
                  String.fromCharCode(SFILE[i*Lstr+3])+
                  String.fromCharCode(SFILE[i*Lstr+4])+
                  String.fromCharCode(SFILE[i*Lstr+5])+
                  String.fromCharCode(SFILE[i*Lstr+6])+
                  String.fromCharCode(SFILE[i*Lstr+7])+
                  String.fromCharCode(SFILE[i*Lstr+8])+
                  String.fromCharCode(SFILE[i*Lstr+9]);
            console.log("SERNUM="+SERNUM);  /////-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
/--/-/-/-/-/-/-/-/-/
            if (SERNUM=='0'+SERPASP+NUMPASP ) {EEE=1; Nstr=i; break;}
            // Если нашли в базе номер паспорта, то устанавливаем EEE=1; Nstr=i;
            }
            }
console.log("Nstr="+Nstr);  /////-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/--/-/-/-/-/-/-
/-/-/

      // Проверяем наличие такого клиента в обработке (попытка многократного
обращения)
            for (i=0; i<100; i++)
            if (PZ[i]==1 && NstrK[i]==Nstr) EEE=0; //
            if (EEE!=1) {CLSTRD(); TR="E0";} //  "Помилкові паспортні дані."
            else
            { // Проверяем на попытку повторного ввода пароля для голосования
            if (PRAVOG[Nstr]!=0) {CLSTRD(); TR="E1";} // "Пароль вже введено."
            else
```

```
            {     // Проверяем первоначальный пароль
            j=PAROL.length;
            EEE=0;
            console.log("PAROL="+PAROL+' j='+j);  /////-/-/-/-/-/-/-/-/-/-/-/-/-/-
/-/-/-/--/-/-/-/-/-/-/-/-/
            for (i=0; i<16; i++) DATA[i]=' ';
            for (i=0; i<j; i++) DATA[i]=PAROL[i];
            CONV_MD5(); // Преобразуем пароль по алгоритму MD5 в массиве DATA[]
(начальные 16 байт в 20 байт)
            CONV_MD5(); // Еще раз преобразуем пароль по алгоритму MD5 в массиве
DATA[] (16 байт в 20 байт)
            for (i=0; i<20; i++) {if
(DATA[i]!=String.fromCharCode(SFILE[Nstr*Lstr+10+i])) EEE=1;}
            if (EEE==1) {CLSTRD(); TR="E3";} // "Помилковий пароль."
            else
            {
//var TEMP='/'; // Контрольный вывод C[] /-/-/-/-/-/-/-/-/-//-/-/-/-/-/-/-/-/-/-
/-/-/-/-/-/-/-/-/--/-/-//-/-/-/-
//for (i=NC1[SNL];i<NC1[SNL]+NC[SNL];i++) TEMP=TEMP+NLB[SNL][i]; ////--/-/-/-/-
/-/-/-/-/-/-//-/-/-/-/-/-/-/-/-/
//console.log(TEMP);/////////-/-/-/-/-/-/-/-/-/-/--/-/-/-/-/-/-/-/-/-//-/-/-/-/-
/-/-/-/-/-/-/-/-/-/-/-/-/-/--/-/-
            TR='S'; // Готовим строку для отправки ответа на пароль
            NstrK[SNL]=Nstr;   // Запоминаем Nstr (номер текущей строки в БД)
облуживаемого клиента
            NQ[SNL]=2; // Установили номер следующего запроса от текущего
клиента
            } // TR="E3"  "Помилковий пароль."
            } // TR='E1'  "Пароль вже введено."
            } // TR='E0'  "Помилкові паспортні дані."
            } // TR='E4'  "Відведений час вичерпано. Почніть спочатку."

console.log("TR="+TR);////////////////////////////////////-/-/-/
            res.end(TR);            // Отправили клиенту ответ на пароль или код
ошибки
      break;
      case '2': // Розшифруємо пароль для голосування
            var S2='' // Строка для поиска двухбайтного номера строки в массиве
NU2[]
            S2=RT[4]+RT[5]; // Занесли двухбайтный номер строки обслуживаемого
клиента в S2
            var SNL=0; // Номер строки с данными обслуживаемого клиента
            while ( NU2[SNL]!=S2) SNL++; // Поиск номера строки обслуживаемого
клиента
            if (PZ[SNL]!=1 || NQ[SNL]!=2) TR='E4'; // Если строка пустая то
отправляем код сообщения "Время истекло"
            else
            {
            NC[SNL]=RT.length-6; // Вычислили к-во символов шифротекста

            for (i=NC1[SNL];i<NC1[SNL]+NC[SNL];i++) //  Цикл расшифровки по
алгоритму Вернама
            {j=0; if (RT[i-NC1[SNL]+6]=='1') j=1; NLB[SNL][i]=j+NLN[SNL][i]; if
(NLB[SNL][i]>1) NLB[SNL][i]=0;
            } // Результат расшифровки занесли в NLB[SNL][]
            var r=NC1[SNL];   // В r установили номер элемента для начала
преобразования из двоичного в десятичный
            var SP='';
            var L=NC[SNL]/7; // Вычислили к-во символов пароля в L
console.log("L="+L);////////////////////////////////////-/-/-/-/-/-/
            var PW='';
            for (i=0; i<L; i++) // Цикл получения символов пароля
            {
            j=0;
```

```
if (NLB[SNL][r]==1) j=j+1;
if (NLB[SNL][r+1]==1)j=j+2;
if (NLB[SNL][r+2]==1) j=j+4;
if (NLB[SNL][r+3]==1) j=j+8;
if (NLB[SNL][r+4]==1) j=j+16;
if (NLB[SNL][r+5]==1) j=j+32;
if (NLB[SNL][r+6]==1) j=j+64;
r=r+7;
switch (j)
{
case 32: PW=PW+' '; break; // 32
case 44: PW=PW+','; break; // 44
case 46: PW=PW+'.'; break; // 46
case 48: PW=PW+'0'; break; // 48
case 49: PW=PW+'1'; break; // 49
case 50: PW=PW+'2'; break; // 50
case 51: PW=PW+'3'; break; // 51
case 52: PW=PW+'4'; break; // 52
case 53: PW=PW+'5'; break; // 53
case 54: PW=PW+'6'; break; // 54
case 55: PW=PW+'7'; break; // 55
case 56: PW=PW+'8'; break; // 56
case 57: PW=PW+'9'; break; // 57
case 60: PW=PW+'<'; break; // 60
case 62: PW=PW+'>'; break; // 62
case 65: PW=PW+'A'; break; // 65
case 66: PW=PW+'B'; break; // 66
case 67: PW=PW+'C'; break; // 67
case 68: PW=PW+'D'; break; // 68
case 69: PW=PW+'E'; break; // 69
case 70: PW=PW+'F'; break; // 70
case 71: PW=PW+'G'; break; // 71
case 72: PW=PW+'H'; break; // 72
case 73: PW=PW+'I'; break; // 73
case 74: PW=PW+'J'; break; // 74
case 75: PW=PW+'K'; break; // 75
case 76: PW=PW+'L'; break; // 76
case 77: PW=PW+'M'; break; // 77
case 78: PW=PW+'N'; break; // 78
case 79: PW=PW+'O'; break; // 79
case 80: PW=PW+'P'; break; // 80
case 81: PW=PW+'Q'; break; // 81
case 82: PW=PW+'R'; break; // 82
case 83: PW=PW+'S'; break; // 83
case 84: PW=PW+'T'; break; // 84
case 85: PW=PW+'U'; break; // 85
case 86: PW=PW+'V'; break; // 86
case 87: PW=PW+'W'; break; // 87
case 88: PW=PW+'X'; break; // 88
case 89: PW=PW+'Y'; break; // 89
case 90: PW=PW+'Z'; break; // 90
case 97: PW=PW+'a'; break; // 97
case 98: PW=PW+'b'; break; // 98
case 99: PW=PW+'c'; break; // 99
case 100: PW=PW+'d'; break; // 100
case 101: PW=PW+'e'; break; // 101
case 102: PW=PW+'f'; break; // 102
case 103: PW=PW+'g'; break; // 103
case 104: PW=PW+'h'; break; // 104
case 105: PW=PW+'i'; break; // 105
case 106: PW=PW+'j'; break; // 106
case 107: PW=PW+'k'; break; // 107
case 108: PW=PW+'l'; break; // 108
case 109: PW=PW+'m'; break; // 109
```

```
            case 110: PW=PW+'n'; break; // 110
            case 111: PW=PW+'o'; break; // 111
            case 112: PW=PW+'p'; break; // 112
            case 113: PW=PW+'q'; break; // 113
            case 114: PW=PW+'r'; break; // 114
            case 115: PW=PW+'s'; break; // 115
            case 116: PW=PW+'t'; break; // 116
            case 117: PW=PW+'u'; break; // 117
            case 118: PW=PW+'v'; break; // 118
            case 119: PW=PW+'w'; break; // 119
            case 120: PW=PW+'x'; break; // 120
            case 121: PW=PW+'y'; break; // 121
            case 122: PW=PW+'z'; break; // 122
            }
            }
            NC1[SNL]=NC1[SNL]+NC[SNL]; // Переустановили номер бита, с которого
следует начинать/продолжать процедуру шифрования
            PASSVYB[NstrK[SNL]]=PW; // Пароль для голосування занесли в пам'ять
            PRAVOG[NstrK[SNL]]=1;   // Встановили ознаку щодо права голосування
            PZ[SNL]=0; // Звільняемо рядок обробки даних клієнта
console.log('PW: '+PW); /////-/-/-/-/-
            CLSTRD();
            TR="E0"; //  Подготовили код сообщения "Пароль для голосування
прийнято"
            }           // TR='E4' Подготовили код сообщения "Время истекло"
      console.log("TR="+TR);//////////////////////////////////////-/-/-/

            res.end(TR);            // Отправляем клиенту код сообщения

      break;
      }
}
/////////////////////// Закінчено обробку для першого періоду
//////////////////////
////////////////////////////// Початок паузи ////////////////////////////////

if (NPERIOD==2)
      {  TR='E4';       // Устанавливаем код сообщения "Время истекло"
      res.end(TR);      // Отправляем клиенту код ошибки
      }
////////////////////////////// Закінчено паузу ////////////////////////////////
//////////////////////// Початок періоду голосування ////////////////////////

if (NPERIOD==3)
{
switch (RT[2])
      {
      case '0': // Будем отправлять новому клиенту номер его строки и значение
NLB[NL][], если есть пустая строка
            CLSTIO(); // Очистка строк данных для клиентов по таймауту
            PZ[NL]=1; // Установили признак занятости строки и будем заполнять
начальные значения всех параметров
            var T1 = new Date(); // Берем метку времени для шифрования + для
проверки времени выполнения
            var TB = T1.getTime(); // TB - момент начала преобразований в
миллисекундах от 01.01.1970
            TIO[NL]=TB+300000;     // Установили таймаут для заполнения анкеты
(5 минут)
            TR='N'+NU2[NL];  // Строка для передачи данных на сервер
            for (i=1;i<=503;i++) // Цикл переноса значения NLB[NL][] в
транспортную строку
            {
            if (NLB[NL][i]==0) TR=TR+'0'; else TR=TR+'1';
            }
```

```
            var NNL=0; // Номер пустой строки для данных следующего клиента
            while ( PZ[NNL]==1 && NNL<100) NNL++; // Поиск пустой строки

            if (NNL==100) TR='/'; // Если пустая строка не найдена, то
отправляем "липовые" данные
            else // Вычисляем биты шифрования для текущего клиента и готовим
строку для данных нового клиента
            {
            NQ[NL]=1; // Установили номер следующего запроса от текущего клиента
            //---- Вычисление послеовательностм битов для шифрования сообщений
            for (i=1;i<=503;i++) A[i]=RT[i+2]; // Занесли принятый от клиента
массив битов в A[]
            var T1 = new Date(); // Берем метку времени для получения случайного
бита
            var TD = T1.getTime()%2; // Проверяем на четность к-во миллисекунд
от 01.01.1970
            if (TD>0) NLN[NNL][1]=1; else  NLN[NNL][1]=0; // В зависимости от
четности заполняем первый случайный бит
      for (i=1;i<=503;i++) MA[1][i] = A[i]; // В первую строку массива занесли
значение A[] в степени 1
            for (I=2;I<=503;I++) // Начинаем цикл заполнения массива MA[][]
степенями A[], а NLN[NNL][] случайными битами
            {
            for (J=1;J<=503;J++) M1[J]=M2[J]=MA[I-1][J]; // Оба сомножителя
равны предыдущей строке
            MULT(); // Каждая следующая строка массива MA[][] равна квадрату
предыдущей строки
                  for (j=1;j<=503;j++) MA[I][j]=R[j]; // Занесли результат
умножения в очередную строку
                  var T1 = new Date(); // Берем метку времени для получения
случайного бита
                  var TD = T1.getTime()%2; // Проверяем на четность к-во
миллисекунд от 01.01.1970
                  if (TD>0) NLN[NNL][I]=1; else NLN[NNL][I]=0;  // Случайный бит
зависит от четности к-ва миллисекунд
                  var NN = Math.random();             // и величины стандартного
случайного числа (сумма по модулю 2)
                  if (NN>0.5) NLN[NNL][I]=NLN[NNL][I]+1; if (NLN[NNL][I]>1)
NLN[NNL][I]=0; // Заполнили очередной случайный бит
            } // Конец цикла заполнения массива степенями A[]
            for (i=1;i<=503;i++) A[i]=0; A[1]=1; // Занесли в A[] единицу
      for (J=1;J<=503;J++) // Начинаем двойной цикл возведения в степень
        if (NLN[NL][J]==1)
        {
            for (I=1;I<=503;I++)
            {
                M1[I]= MA[J][I];
                M2[I]=A[I];
            }
            MULT();
            for (I=1;I<=503;I++)
              A[I]=R[I];
        }  // Конец двойного цикла возведения в степень. Результат занесен в A[]

            // --- В A[] получена случайная послеовательность для шифрования
сообщений
      //    var STR='/'; ////////////////////////////////////////////-/-/-/-/
      //    for (i=1;i<=503;i++) STR=STR+A[i];//////////////////////-/-/-/-/-/-
/-/-/-/-/-/--/-/-/
      //    console.log("C="+STR);/////////////////////////////////////-/-/-/
            for (i=1;i<=503;i++) NLN[NL][i]=A[i]; // Перенесли эту
последовательность в строку данных клиента  NLN[NL][]
            NL=NNL;      // Установили номер строки данных для следующего
клиента (эта строка уже подготовлена)
```

```
            STEPX(); // Вычисление случайной степени примитивного элемента X в
строке данных следующего клиента
            } // Подготовили в TR биты (массив B) для клиента и подготовили
строку для данных следующего клиента
            res.end(TR);            // Отправили данные клиенту
      break;
      case '1': // Будем расшифровывать и проверять паспортные данные клиента
            var S2='' // Строка для поиска двухбайтного номера строки в массиве
NU2[]
console.log(" RT[[1]="+RT[1]+" RT[2]="+RT[2]+" RT[3]="+RT[3]+" RT[4]="+RT[4]+"
RT[5]="+RT[5] );/////////////////////////////////////-/-/-/
            S2=RT[4]+RT[5]; // Занесли двухбайтный номер строки обслуживаемого
клиента в S2
console.log("S2="+S2);////////////////////////////////////-/-/-/-/-/-/-/-/-/-/-
/-/-/-/-/-/-/-/-/-/-/-/-/
            var SNL=0; // Номер строки с данными обслуживаемого клиента
            while (NU2[SNL]!=S2) SNL++; // Поиск номера строки обслуживаемого
клиента
console.log("SNL="+SNL+" NU2[SNL]="+NU2[SNL]+" NQ[SNL]="+NQ[SNL]
);///////////////////////////////////-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
/-/-/
            if (PZ[SNL]!=1 || (PZ[SNL]==1 && NQ[SNL]!=1)) TR='E4';
            // Если строка пустая то отправляем код сообщения "Время истекло"
            else
            {
            NC[SNL]=RT.length-6; // Вычислили к-во символов шифротекста

            for (i=NC1[SNL];i<NC1[SNL]+NC[SNL];i++) //  Цикл расшифрования по
алгоритму Вернама
            {j=0; if (RT[i+5]=='1') j=1; NLB[SNL][i]=j+NLN[SNL][i]; if
(NLB[SNL][i]>1) NLB[SNL][i]=0;
            } // Результат расшифровки занесли в NLB[SNL][]
            var r=NC1[SNL];
            var SP='';
            for (i=0; i<2; i++) // Цикл получения серии паспорта
            {
            j=0;
//console.log("r="+r+" NLB[SNL][1]="+NLB[SNL][1]+" NLB[SNL][2]="+NLB[SNL][2]+"
NLB[SNL][3]="+NLB[SNL][3]+" NLB[SNL][4]="+NLB[SNL][4]
);////////////////////////////////////-/-/-/
            if (NLB[SNL][r]==1) j=j+1;
            if (NLB[SNL][r+1]==1)j=j+2;
            if (NLB[SNL][r+2]==1) j=j+4;
            if (NLB[SNL][r+3]==1) j=j+8;
            r=r+4;
//console.log("j="+j);/////////////////////////////////////-/-/-/-/-/-/
            switch (j)
            {
            case 1: SP=SP+'A'; break; // Латинская кодировка букв
            case 2: SP=SP+'B'; break;
            case 3: SP=SP+'C'; break;
            case 4: SP=SP+'E'; break;
            case 5: SP=SP+'H'; break;
            case 6: SP=SP+'I'; break;
            case 7: SP=SP+'K'; break;
            case 8: SP=SP+'M'; break;
            case 9: SP=SP+'O'; break;
            case 10: SP=SP+'P'; break;
            case 11: SP=SP+'T'; break;
            case 12: SP=SP+'X'; break;
            }
            }
            SERPASP=SP;
            var SP='';
```

```
            for (i=0; i<6; i++) // Цикл получения номера паспорта
            {
            j=0;
            if (NLB[SNL][r]==1) j=j+1;
            if (NLB[SNL][r+1]==1)j=j+2;
            if (NLB[SNL][r+2]==1) j=j+4;
            if (NLB[SNL][r+3]==1) j=j+8;
            r=r+4;
            switch (j)
            {
            case 1: SP=SP+'1'; break;
            case 2: SP=SP+'2'; break;
            case 3: SP=SP+'3'; break;
            case 4: SP=SP+'4'; break;
            case 5: SP=SP+'5'; break;
            case 6: SP=SP+'6'; break;
            case 7: SP=SP+'7'; break;
            case 8: SP=SP+'8'; break;
            case 9: SP=SP+'9'; break;
            case 10: SP=SP+'0'; break;
            }
            }
            NUMPASP=SP;
// Получение номера режима NR[SNL] облслуживания клиента
//  1-Довідка по дільниці, 2-Довідка по вулиці, 3-Довідка по будинку)
            j=0;
            if (NLB[SNL][r]==1) j=j+1;
            if (NLB[SNL][r+1]==1)j=j+2;
            if (NLB[SNL][r+2]==1) j=j+4;
            if (NLB[SNL][r+3]==1) j=j+8;
            r=r+4;
            NR[SNL]=j;
console.log("NR="+j);////////////////////////////////////-/-/-/-/-/-/

            var L=(NC[SNL]-36)/7; // Длину зашифрованного пароля занесли в L
console.log("L="+L);////////////////////////////////////-/-/-/-/-/-/
            var PW='';
            for (i=0; i<L; i++) // Цикл получения символов пароля
            {
            j=0;
            if (NLB[SNL][r]==1) j=j+1;
            if (NLB[SNL][r+1]==1)j=j+2;
            if (NLB[SNL][r+2]==1) j=j+4;
            if (NLB[SNL][r+3]==1) j=j+8;
            if (NLB[SNL][r+4]==1) j=j+16;
            if (NLB[SNL][r+5]==1) j=j+32;
            if (NLB[SNL][r+6]==1) j=j+64;
            r=r+7;
            switch (j)
            {
            case 32: PW=PW+' '; break; // 32
            case 44: PW=PW+','; break; // 44
            case 46: PW=PW+'.'; break; // 46
            case 48: PW=PW+'0'; break; // 48
            case 49: PW=PW+'1'; break; // 49
            case 50: PW=PW+'2'; break; // 50
            case 51: PW=PW+'3'; break; // 51
            case 52: PW=PW+'4'; break; // 52
            case 53: PW=PW+'5'; break; // 53
            case 54: PW=PW+'6'; break; // 54
            case 55: PW=PW+'7'; break; // 55
            case 56: PW=PW+'8'; break; // 56
            case 57: PW=PW+'9'; break; // 57
            case 60: PW=PW+'<'; break; // 60
```

```
            case 62: PW=PW+'>'; break; // 62
            case 65: PW=PW+'A'; break; // 65
            case 66: PW=PW+'B'; break; // 66
            case 67: PW=PW+'C'; break; // 67
            case 68: PW=PW+'D'; break; // 68
            case 69: PW=PW+'E'; break; // 69
            case 70: PW=PW+'F'; break; // 70
            case 71: PW=PW+'G'; break; // 71
            case 72: PW=PW+'H'; break; // 72
            case 73: PW=PW+'I'; break; // 73
            case 74: PW=PW+'J'; break; // 74
            case 75: PW=PW+'K'; break; // 75
            case 76: PW=PW+'L'; break; // 76
            case 77: PW=PW+'M'; break; // 77
            case 78: PW=PW+'N'; break; // 78
            case 79: PW=PW+'O'; break; // 79
            case 80: PW=PW+'P'; break; // 80
            case 81: PW=PW+'Q'; break; // 81
            case 82: PW=PW+'R'; break; // 82
            case 83: PW=PW+'S'; break; // 83
            case 84: PW=PW+'T'; break; // 84
            case 85: PW=PW+'U'; break; // 85
            case 86: PW=PW+'V'; break; // 86
            case 87: PW=PW+'W'; break; // 87
            case 88: PW=PW+'X'; break; // 88
            case 89: PW=PW+'Y'; break; // 89
            case 90: PW=PW+'Z'; break; // 90
            case 97: PW=PW+'a'; break; // 97
            case 98: PW=PW+'b'; break; // 98
            case 99: PW=PW+'c'; break; // 99
            case 100: PW=PW+'d'; break; // 100
            case 101: PW=PW+'e'; break; // 101
            case 102: PW=PW+'f'; break; // 102
            case 103: PW=PW+'g'; break; // 103
            case 104: PW=PW+'h'; break; // 104
            case 105: PW=PW+'i'; break; // 105
            case 106: PW=PW+'j'; break; // 106
            case 107: PW=PW+'k'; break; // 107
            case 108: PW=PW+'l'; break; // 108
            case 109: PW=PW+'m'; break; // 109
            case 110: PW=PW+'n'; break; // 110
            case 111: PW=PW+'o'; break; // 111
            case 112: PW=PW+'p'; break; // 112
            case 113: PW=PW+'q'; break; // 113
            case 114: PW=PW+'r'; break; // 114
            case 115: PW=PW+'s'; break; // 115
            case 116: PW=PW+'t'; break; // 116
            case 117: PW=PW+'u'; break; // 117
            case 118: PW=PW+'v'; break; // 118
            case 119: PW=PW+'w'; break; // 119
            case 120: PW=PW+'x'; break; // 120
            case 121: PW=PW+'y'; break; // 121
            case 122: PW=PW+'z'; break; // 122
            }
            }
            NC1[SNL]=NC1[SNL]+NC[SNL]; // Переустановили номер бита, с которого
следует начинать/продолжать процедуру шифрования
            PAROL=PW;
console.log(SERPASP+NUMPASP+' PW: '+PW); /////-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/--
/-/-/-/-/-/-/-/-/
// Проверяем наличие/отсутствие в базе данных по избирателю и формируем ответ
('E0' ... 'E4' - Error или 'S...ответ на пароль... )
            EEE=0; // Признак наличия избирателя в базе ( 0-нет, 1-есть)
            if (Kstr>0)
```

```
            {
console.log("Kstr="+Kstr);  /////-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/--/-/-/-/-/-/-
/-/-/
//console.log("SFILE="+SFILE);  /////-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/--/-/-/-/-
/-/-/-/-/
            var SERNUM='';
            for (i=0; i<Kstr; i++) // Цикл формирования строки с данными
паспорта
            {
            SERNUM=String.fromCharCode(SFILE[i*Lstr+1])+
                  String.fromCharCode(SFILE[i*Lstr+2])+
                  String.fromCharCode(SFILE[i*Lstr+3])+
                  String.fromCharCode(SFILE[i*Lstr+4])+
                  String.fromCharCode(SFILE[i*Lstr+5])+
                  String.fromCharCode(SFILE[i*Lstr+6])+
                  String.fromCharCode(SFILE[i*Lstr+7])+
                  String.fromCharCode(SFILE[i*Lstr+8])+
                  String.fromCharCode(SFILE[i*Lstr+9]);
            console.log("SERNUM="+SERNUM);  /////-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
/--/-/-/-/-/-/-/-/-/
            if (SERNUM=='0'+SERPASP+NUMPASP ) {EEE=1; Nstr=i; break;}
            // Если нашли в базе номер паспорта, то устанавливаем EEE=1; Nstr=i;
            }
            }
console.log("Nstr="+Nstr);  /////-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/--/-/-/-/-/-/-
/-/-/
            if (NR[SNL]!=0 && EEE==1)
      {           // Вивід довідок
            var SUBJ; // Тема листа
            var TEXT='Кількість виборців, що голосують дистанційно \r\n';
            var KZ,KZV; // Кількість зареєстрованих виборців, ... всього
            var KG,KGV; // Кількість виборців, що проголосували, ... всього
            switch (NR[SNL])
            {
            case 1: SUBJ='Довідка по дільниці';
                        TEXT=TEXT+'        по дільниці '+DILN+'\r\n'+
                        '-------------------------------------------------------
---------------------------- \r\n'+
                        ':   Назва вулиці     : Зареєстровано : Проголосували
\r\n' +
                        '-------------------------------------------------------
---------------------------- \r\n';
                  KZ=KZV=KG=KGV=0;
                  for (j=0; j<KVUL; j++) // Цикл по вулицям
                  {     S2=NU2[j]
                        for (i=0; i<Kstr; i++) // Цикл по всім рядкам БД
                        {
                              if (String.fromCharCode(SFILE[i*Lstr+1])=='0')
                              { // Обираємо тільки дійсних виборців
                                if
(String.fromCharCode(SFILE[i*Lstr+119])==S2[0] &&

String.fromCharCode(SFILE[i*Lstr+120])==S2[1])
                                    { // Обираємо тільки j-ту  вулицю
                                    KZ++;
                                    if (GOLOSF[i]==1) KG++;
                                    }
                              }
                        }
                  TEXT=TEXT+VUL[j]+'\t\t\t'+KZ+'\t\t\t'+KG+'\r\n';
                  KZV=KZV+KZ; KGV=KGV+KG; KZ=KG=0;
                  }
                  TEXT=TEXT+'-------------------------------------------------------
------------------------------- \r\n'+
```

```
                  ' Усього'+'\t\t\t\t'+KZV+'\t\t\t'+KGV+'\r\n';
            break;
            case 2: SUBJ='Довідка по вулиці';
                        TEXT=TEXT+'        по вулиці'+VUL[KODVUL]+'\r\n'+
                        '-------------------------------------------------------
--------------------------- \r\n'+
                        ':   Номер будинку     : Зареєстровано : Проголосували
\r\n' +
                        '-------------------------------------------------------
--------------------------- \r\n';
            break;
            case 3: SUBJ='Довідка по будинку';
                        TEXT=TEXT+'        по будинку № '+NUMBUD+' по вулиці
'+VUL[KODVUL]+
                        '-------------------------------------------------------
--------------------------- \r\n'+
                        ':   Номер будинку     : Зареєстровано : Проголосували
\r\n' +
                        '-------------------------------------------------------
--------------------------- \r\n';
            break;
            }
            EMAIL_D='';
            for (i=0; i<36; i++)
            { // Переносим адрес электронной почты в ячейку EMAIL_D
            if (String.fromCharCode(SFILE[Nstr*Lstr+70+i])!=' ')
            EMAIL_D=EMAIL_D+String.fromCharCode(SFILE[Nstr*Lstr+70+i]);
            }
            // Отправка справки на Email
            var transporter = nodemailer.createTransport({
            service: EMAILSERV,
            auth:       {
                        user: EMAILNAME,
                        pass: EMAILPASS
                        }
            });
            transporter.sendMail({
            from: EMAILNAME,
            to: EMAIL_D,
            subject: SUBJ,
            text: TEXT
            });
      CLSTRD(); TR="E8";
      }
      else
      {
      // Проверяем наличие такого клиента в обработке (попытка продублировать
свой голос)
            for (i=0; i<100; i++)
            if (PZ[i]==1 && NstrK[i]==Nstr) EEE=0; //
            if (EEE!=1) {CLSTRD(); TR="E0";} //  "Помилкові паспортні дані."
            else
            { // Проверяем наличие права голоса
            PRAVO=String.fromCharCode(SFILE[Nstr*Lstr+224]);
            if (PRAVO!='1') {CLSTRD(); TR="E1";} // "Не надано права голосу."
//          else         // Отключается 4 строки для случая возможности
принудительного голосования
//          { // Проверяем использование права голоса
//          VYBOR=String.fromCharCode(SFILE[Nstr*Lstr+225]);
//          if (VYBOR!='0') TR="E2"; // "Право голосу вже використано."
            else
            {     // Проверяем правильность пароля
            j=PAROL.length;
```

```
            if (PASSVYB[Nstr].length!=j) {CLSTRD(); TR="E3";} // "Помилковий
пароль."
            else
            {
            PASSINP[Nstr]=PAROL;
            console.log("PAROL="+PAROL+' j='+j);  /////-/-/-/-/-/-/-/-/-/-/-/-/-
/-/-/-/--/-/-/-/-/-/-/-/-/
//var TEMP=''; // Контрольный вывод C[] /-/-/-/-/-/-/-/-/-/-//-/-/-/-/-/-/-/-/-/-
/-/-/-/-/-/-/-/-/--/-/-//-/-/-/-
//for (i=NC1[SNL];i<NC1[SNL]+NC[SNL];i++) TEMP=TEMP+NLB[SNL][i]; ////--/-/-/-/-
/-/-/-/-/-/-//-/-/-/-/-/-/-/-/-/-/
//console.log(TEMP);/////////-/-/-/-/-/-/-/-/-/-/--/-/-/-/-/-/-/-/-/-//-/-/-/-/-
/-/-/-/-/-/-/-/-/-/-/-/-/-/-/--/-/-
            TR='S'; // Готовим строку для отправки ответа на пароль
            NstrK[SNL]=Nstr;   // Запоминаем Nstr (номер текущей строки в БД)
облуживаемого клиента
            CODE[SNL]='';
            for (i=0;i<3;i++) // Цикл формирования строки из случайных шести
цифр
            {NN = Math.random(); CODE[SNL]=CODE[SNL]+NU2[Math.floor(NN*99.9)];}
            EMAIL_D='';
            for (i=0; i<36; i++)
            { // Переносим адрес электронной почты для отправки кода из БД в
ячейку EMAIL_D
            if (String.fromCharCode(SFILE[Nstr*Lstr+70+i])!=' ')
            EMAIL_D=EMAIL_D+String.fromCharCode(SFILE[Nstr*Lstr+70+i]);
            }
            NQ[SNL]=2; // Установили номер следующего запроса от текущего
клиента
            // Отправка кода на Email
            var transporter = nodemailer.createTransport({
            service: EMAILSERV,
            auth:       {
                        user: EMAILNAME,
                        pass: EMAILPASS
                        }
            });
            transporter.sendMail({
            from: EMAILNAME,
            to: EMAIL_D,
            subject: 'Код для голосування',
            text: 'Код: '+CODE[SNL]
            });
            NQ[SNL]=2; // Установили номер следующего запроса от текущего
клиента
            } // TR="E3"  "Помилковий пароль."
//          } // TR="E2"  "Право голосу вже використано."  // Отключается эта
строка для случая принудительного голосования
            } // TR='E1'  "Не надано права голосу."
            } // TR='E0'  "Помилкові паспортні дані."
            } // TR='E8'  "Довідку Вам відправлено на E-mail."
            } // TR='E4'  "Відведений час вичерпано. Почніть спочатку."

console.log("TR="+TR);////////////////////////////////////-/-/-/
            res.end(TR);            // Отправили клиенту ответ на пароль или код
ошибки
      break;
      case '2': // Будем расшифровывать код и результат голосования
            var S2='' // Строка для поиска двухбайтного номера строки в массиве
NU2[]
            S2=RT[4]+RT[5]; // Занесли двухбайтный номер строки обслуживаемого
клиента в S2
            var SNL=0; // Номер строки с данными обслуживаемого клиента
```

```
            while ( NU2[SNL]!=S2) SNL++; // Поиск номера строки обслуживаемого
клиента
            if (PZ[SNL]!=1 || NQ[SNL]!=2) TR='E4'; // Если строка пустая то
отправляем код сообщения "Время истекло"
            else
            {
            NC[SNL]=RT.length-6; // Вычислили к-во символов шифротекста

            for (i=NC1[SNL];i<NC1[SNL]+NC[SNL];i++) //  Цикл расшифровки по
алгоритму Вернама
            {j=0; if (RT[i-NC1[SNL]+6]=='1') j=1; NLB[SNL][i]=j+NLN[SNL][i]; if
(NLB[SNL][i]>1) NLB[SNL][i]=0;
            } // Результат расшифровки занесли в NLB[SNL][]
            var r=NC1[SNL];   // В r установили номер элемента для начала
преобразования из двоичного в десятичный
            var SP='';
            for (i=0; i<6; i++) // Цикл получения шестизначного номера (кода)
            {
            j=0;
            if (NLB[SNL][r]==1) j=j+1;
            if (NLB[SNL][r+1]==1)j=j+2;
            if (NLB[SNL][r+2]==1) j=j+4;
            if (NLB[SNL][r+3]==1) j=j+8;
            r=r+4;
            switch (j)
            {
            case 1: SP=SP+'1'; break;
            case 2: SP=SP+'2'; break;
            case 3: SP=SP+'3'; break;
            case 4: SP=SP+'4'; break;
            case 5: SP=SP+'5'; break;
            case 6: SP=SP+'6'; break;
            case 7: SP=SP+'7'; break;
            case 8: SP=SP+'8'; break;
            case 9: SP=SP+'9'; break;
            case 10: SP=SP+'0'; break;
            default: SP=SP+'x';
            }
            }
console.log("KOD="+SP);/////////-/-/-/-/-/-/-/-/-/-/--/-/-/-/-/-/-/-/-/-//-/-/-
/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/--/-/-//-/-
            if (CODE[SNL]!=SP) TR='E1'; // Отправляем сообщение об ошибке
"Ошибочный код"
//          else // Отключается 4 строки для случая возможности принудительного
голосования
//          {
//          VYBOR=String.fromCharCode(SFILE[NstrK[SNL]*Lstr+225]);
//          if (VYBOR!='0') TR="E2"; // Отправляем сообщение "Право голосу вже
використано."
            else  // Эти 4 строки вводятся только для случая возможности
принудительного голосования
            {
            if (GOLOS[NstrK[SNL]]==1 ||  PASSVYB[NstrK[SNL]] !=
PASSINP[NstrK[SNL]])
                  {GOLOSF[NstrK[SNL]]=1; CLSTRD(); TR="E0";} //  Отправляем
сообщение "Ваш голос прийнято"
            else
            {
            KBUL=(NC[SNL]-24)/8;    // Вычислили к-во бюллетней
                  for (var b=0; b<KBUL; b++) // Цикл по бюлетням
                  {
                  var VYB=0;
                  for (i=0; i<2; i++) // Цикл определения номера, выбранного
избирателем в бюллетне
```

```
                        {
                        j=0;
                        if (NLB[SNL][r]==1) j=j+1;
                        if (NLB[SNL][r+1]==1)j=j+2;
                        if (NLB[SNL][r+2]==1) j=j+4;
                        if (NLB[SNL][r+3]==1) j=j+8;
                        if (j==10) j=0;
                        if (i==0) VYB=10*j;
                        else VYB=VYB+j;
                        r=r+4;
                        }
                  GOL[b][VYB]=GOL[b][VYB]+1; // Голос засчитали
                  GOLOS[NstrK[SNL]]=1; // Отметили использование права голоса
                  GOLOSF[NstrK[SNL]]=1
                  CLSTRD(); TR="E0"; //  Отправляем сообщение "Ваш голос
прийнято"
                  }
            } // TR="E2"  "Право голосу вже використано."  или  TR="E0" для
случая принудительного голосования
            } // TR='E1' Отправляем сообщение об ошибке "Ошибочный код"
            } // TR='E4' Отправляем код сообщения "Время истекло"
      console.log("TR="+TR);//////////////////////////////////////-/-/-/

            res.end(TR);            // Отправили клиенту ответ на пароль или код
ошибки
      break;
      }
      }
      }
      }
}).listen(8080);

console.log('Server listen port 8080');
function CLSTIO()     // Очистка строк данных для клиентов по тайм-ауту
{
var T1 = new Date(); // Берем метку времени для проверки таймаута
for (i=0;i<100;i++)
            {
            if (PZ[i]==1)                        // Если строка занята и
                  if (TIO[i]<T1.getTime()) // если время истекло, очищаем строку
                  {        // Устанавливем начальные значения переменных.
                  PZ[i]=0; // Признак занятости строки
                  NQ[i]=0; // Номер ожидаемого запроса клиента
                  NC1[i]=1;// Номер бита, с которого следует начинать/продолжать
процедуру шифрования
                  }
            }

} // Enf of function CLSTIO()

function CLSTRD()     // Очистка строки данных клиента
{
      PZ[SNL]=0; // Признак занятости строки
      NQ[SNL]=0; // Номер ожидаемого запроса клиента
      NC1[SNL]=1;//     Номер бита, с которого следует начинать/продолжать
процедуру шифрования
} // End of function CLSTRD()

function ENDPER()     // Завершение периода
{     if (FLAG_END==1) return; // Флаг завершения выборов
switch (NPERIOD)
{
case 1: NPERIOD=2; // Завершено період введення паролів для голосування
      var T1 = new Date(); // Берем метку времени для отсчета очередного периода
```

```
      var TBG = T1.getTime(); // TBG - момент начала периода в миллисекундах от
01.01.1970
      var TEND=TBG+3600000*TPAUSE; // Установили момент окончания паузы
      var REGISTR=''; // Строка для формирования результата регистрации (ввода
пароля для голосования)
      for (Nstr=0; Nstr<Kstr; Nstr++)
      {
      if (PRAVOG[Nstr]==0) REGISTR=REGISTR+'0';
      else REGISTR=REGISTR+'1';
      }
      fs.writeFileSync('REGISTR.TXT', REGISTR); // Запись в файл
break;
case 2: NPERIOD=3;  // Завершено паузу
      var T1 = new Date(); // Берем метку времени для отсчета очередного периода
      var TBG = T1.getTime(); // TBG - момент начала периода в миллисекундах от
01.01.1970
      var TEND=TBG+3600000*TGOL;   // Установили момент окончания периода
голосования
      NAMEHTML="/anketa.html";
break;
case 3:   // Завершено вибори
      var RESULT=''; // Строка для формирования результата голосования
      var GOLOSA=''; // Строка для признака участия в голосовании
      var S1;
for (var b=0; b<KBUL; b++) // Цикл по бюлетням
      {
      S1=CYF[b+1];
      RESULT=RESULT+'Bulletin #'+S1+'\r\n';
      for (i=0;i<100;i++)
            {
            if (GOL[b][i]>0) RESULT=RESULT+'#'+i+': '+GOL[b][i]+'\r\n';
            }
      }
      fs.writeFileSync('RESULT.TXT', RESULT); // Запись в файл
      for (Nstr=0; Nstr<Kstr; Nstr++)
      {
      if (GOLOSF[Nstr]==0) GOLOSA=GOLOSA+'0';
      else GOLOSA=GOLOSA+'1';
      }
      fs.writeFileSync('GOLOSA.TXT', GOLOSA); // Запись в файл
      FLAG_END=1; // Флаг завершения выборов (1-выборы закончены)
// Отправляем результаты голосования всем избирателям участка
      for (Nstr=0; Nstr<Kstr; Nstr++)
      {
      if (GOLOSF[Nstr]!=0)
      { // Отправляем только тем, кто проголосовал
      EMAIL_D='';
            for (i=0; i<36; i++)
            { // Переносим адрес электронной почты в ячейку EMAIL_D
            if (String.fromCharCode(SFILE[Nstr*Lstr+70+i])!=' ')
            EMAIL_D=EMAIL_D+String.fromCharCode(SFILE[Nstr*Lstr+70+i]);
            }
            // Отправка результатов на Email
      var transporter = nodemailer.createTransport({
            service: EMAILSERV,
            auth:       {
                        user: EMAILNAME,
                        pass: EMAILPASS
                        }
      });
      transporter.sendMail({
            from: EMAILNAME,
            to: EMAIL_D,
            subject: 'Результати голосування на дільниці '+DILN,
```

```
            text: RESULT
      });
      }
      }
      break;
}
} // Enf of function ENDPER()
function CONV_MD5() // Преобразуем пароль по алгоритму MD5 в массиве DATA[]
(начальные 16 байт в 20 байт)
{      var olda, oldb, oldc, oldd,
            a =  1732584193,
            b = -271733879,
            c = -1732584194,
            d =  271733878;

            olda = a;
            oldb = b;
            oldc = c;
            oldd = d;

            a = md5_ff(a, b, c, d, DATA[0],  7, -680876936);
            d = md5_ff(d, a, b, c, DATA[1], 12, -389564586);
            c = md5_ff(c, d, a, b, DATA[2], 17,  606105819);
            b = md5_ff(b, c, d, a, DATA[3], 22, -1044525330);
            a = md5_ff(a, b, c, d, DATA[4],  7, -176418897);
            d = md5_ff(d, a, b, c, DATA[5], 12,  1200080426);
            c = md5_ff(c, d, a, b, DATA[6], 17, -1473231341);
            b = md5_ff(b, c, d, a, DATA[7], 22, -45705983);
            a = md5_ff(a, b, c, d, DATA[8],  7,  1770035416);
            d = md5_ff(d, a, b, c, DATA[9], 12, -1958414417);
            c = md5_ff(c, d, a, b, DATA[10], 17, -42063);
            b = md5_ff(b, c, d, a, DATA[11], 22, -1990404162);
            a = md5_ff(a, b, c, d, DATA[12],  7,  1804603682);
            d = md5_ff(d, a, b, c, DATA[13], 12, -40341101);
            c = md5_ff(c, d, a, b, DATA[14], 17, -1502002290);
            b = md5_ff(b, c, d, a, DATA[15], 22,  1236535329);

            a = md5_gg(a, b, c, d, DATA[1],  5, -165796510);
            d = md5_gg(d, a, b, c, DATA[6],  9, -1069501632);
            c = md5_gg(c, d, a, b, DATA[11], 14,  643717713);
            b = md5_gg(b, c, d, a, DATA[0],  20, -373897302);
            a = md5_gg(a, b, c, d, DATA[5],  5, -701558691);
            d = md5_gg(d, a, b, c, DATA[10],  9,  38016083);
            c = md5_gg(c, d, a, b, DATA[15], 14, -660478335);
            b = md5_gg(b, c, d, a, DATA[4], 20, -405537848);
            a = md5_gg(a, b, c, d, DATA[9],  5,  568446438);
            d = md5_gg(d, a, b, c, DATA[14],  9, -1019803690);
            c = md5_gg(c, d, a, b, DATA[3], 14, -187363961);
            b = md5_gg(b, c, d, a, DATA[8], 20,  1163531501);
            a = md5_gg(a, b, c, d, DATA[13],  5, -1444681467);
            d = md5_gg(d, a, b, c, DATA[2],  9, -51403784);
            c = md5_gg(c, d, a, b, DATA[7], 14,  1735328473);
            b = md5_gg(b, c, d, a, DATA[12], 20, -1926607734);

            a = md5_hh(a, b, c, d, DATA[5],  4, -378558);
            d = md5_hh(d, a, b, c, DATA[8], 11, -2022574463);
            c = md5_hh(c, d, a, b, DATA[11], 16,  1839030562);
            b = md5_hh(b, c, d, a, DATA[14], 23, -35309556);
            a = md5_hh(a, b, c, d, DATA[1],  4, -1530992060);
            d = md5_hh(d, a, b, c, DATA[4], 11,  1272893353);
            c = md5_hh(c, d, a, b, DATA[7], 16, -155497632);
            b = md5_hh(b, c, d, a, DATA[10], 23, -1094730640);
            a = md5_hh(a, b, c, d, DATA[13],  4,  681279174);
            d = md5_hh(d, a, b, c, DATA[0],  11, -358537222);
```

```
            c = md5_hh(c, d, a, b, DATA[3], 16, -722521979);
            b = md5_hh(b, c, d, a, DATA[6], 23,  76029189);
            a = md5_hh(a, b, c, d, DATA[9],  4, -640364487);
            d = md5_hh(d, a, b, c, DATA[12], 11, -421815835);
            c = md5_hh(c, d, a, b, DATA[15], 16,  530742520);
            b = md5_hh(b, c, d, a, DATA[2], 23, -995338651);

            a = md5_ii(a, b, c, d, DATA[0],  6, -198630844);
            d = md5_ii(d, a, b, c, DATA[7], 10,  1126891415);
            c = md5_ii(c, d, a, b, DATA[14], 15, -1416354905);
            b = md5_ii(b, c, d, a, DATA[5], 21, -57434055);
            a = md5_ii(a, b, c, d, DATA[12],  6,  1700485571);
            d = md5_ii(d, a, b, c, DATA[3], 10, -1894986606);
            c = md5_ii(c, d, a, b, DATA[10], 15, -1051523);
            b = md5_ii(b, c, d, a, DATA[1], 21, -2054922799);
            a = md5_ii(a, b, c, d, DATA[8],  6,  1873313359);
            d = md5_ii(d, a, b, c, DATA[15], 10, -30611744);
            c = md5_ii(c, d, a, b, DATA[6], 15, -1560198380);
            b = md5_ii(b, c, d, a, DATA[13], 21,  1309151649);
            a = md5_ii(a, b, c, d, DATA[4],  6, -145523070);
            d = md5_ii(d, a, b, c, DATA[11], 10, -1120210379);
            c = md5_ii(c, d, a, b, DATA[2], 15,  718787259);
            b = md5_ii(b, c, d, a, DATA[9], 21, -343485551);


var i, W=[4];
            W[0] = safe_add(a, olda); if (W[0]<0) W[0]=4294967296+W[0];
          W[1] = safe_add(b, oldb); if (W[1]<0) W[1]=4294967296+W[1];
          W[2] = safe_add(c, oldc); if (W[2]<0) W[2]=4294967296+W[2];
          W[3] = safe_add(d, oldd); if (W[3]<0) W[3]=4294967296+W[3];
    for (i=0; i<4; i++)
      {
      DATA[0+i]   = W[i]%128;
      DATA[4+i]   = ((W[i]-DATA[0+i])/128)%128;
      DATA[8+i]   = ((W[i]-DATA[0+i]-DATA[4+i]*128)/(128*128))%128;
      DATA[12+i]  = ((W[i]-DATA[0+i]-DATA[4+i]*128-
DATA[8+i]*128*128)/(128*128*128))%128;
      DATA[16+i]  = ((W[i]-DATA[0+i]-DATA[4+i]*128-DATA[8+i]*128*128-
DATA[12+i]*128*128*128)/(128*128*128*128))%128;
    }
      for (i=0; i<20; i++) DATA[i]=String.fromCharCode(DATA[i]);
}
    function safe_add(x, y) // Суммирование целых 32-битных чисел
      {
        var lsw = (x & 0xFFFF) + (y & 0xFFFF),
            msw = (x >> 16) + (y >> 16) + (lsw >> 16);
        return (msw << 16) | (lsw & 0xFFFF);
    }

    function bit_rol(num, cnt) // Циклический сдвиг влево на cnt бит
      {
        return (num << cnt) | (num >>> (32 - cnt));
    }
// Пять функций к алгоритму MD5
    function md5_cmn(q, a, b, x, s, t) {
        return safe_add(bit_rol(safe_add(safe_add(a, q), safe_add(x, t)), s),
b);
    }
    function md5_ff(a, b, c, d, x, s, t) {
        return md5_cmn((b & c) | ((~b) & d), a, b, x, s, t);
    }
    function md5_gg(a, b, c, d, x, s, t) {
        return md5_cmn((b & d) | (c & (~d)), a, b, x, s, t);
    }
```

```
function md5_hh(a, b, c, d, x, s, t) {
    return md5_cmn(b ^ c ^ d, a, b, x, s, t);
}
function md5_ii(a, b, c, d, x, s, t) {
    return md5_cmn(c ^ (b | (~d)), a, b, x, s, t);
}
```

Д1.5. Клієнтська програма введення паролів

(файл CPW.html)

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
 <title> Виборча дільниця № 1</title>
</head>
<body>
 <h2>Виборча дільниця № 1</h2>
<script type="text/javascript">
var KOD; // код из Email (или СМС) (6 цифр) -------------------
var NBUL=0; // Номер бюлетеня --------------------------------
var PAROL='a12345';   // Пароль для голосування
var REPEAT='a12345';  // Повторення паролю
var VYB=[9]; // Обраний номер пункту по кожному з бюлетенів
var M1 = [504];   // Резервируем ячейки для элементов поля Галуа GF(2^503)
var M2 = [504];   // M1[], M2[] - сомножители  R[] - результат умножения
var R  = [504];   // Младший бит храним в [1], старший - в [503]. Нулевые
элементы не используем
function MULT()     // Умножение элементов поля Галуа GF(2^503) по правилу
полиномов
{
 var i,j,r,r1,r2,r3;
    for (i=1;i<=503;i++) R[i]=0; // Обнуляем 503 ячейки для битов результата
    for (i=1;i<=503;i++)            // Начинаем двойной цикл умножения элементов
полинома
    if (M1[i]==1) // Умножаем только единичные элементы из M1[], нулевые не
нужны, т.к. они дают нули
    {
        for (j=1;j<=503;j++)
        if (M2[j]==1)
        {
        r=i+j-1;
            if (r>503)
            {
             r=r-503;
             if (r>=501)
                {
                r=r-501;
                r1=1+r; r2=4+r; r3=501+r;
                if (R[r3]==0) R[r3]=1; else R[r3]=0;
                }
            else {r1=r; r2=r+3;}
            if (R[r1]==0) R[r1]=1; else R[r1]=0;
            if (R[r2]==0) R[r2]=1; else R[r2]=0;
            }
            else {if (R[r]==0) R[r]=1; else R[r]=0;}
        }
    }
} // End of function MULT()

function getXmlHttp() // Кроссбраузерная функция создания XMLHttpRequest
{ // С помощью этой функции осуществляем обмен данными с сервером
  var xmlhttp;
  try {
    xmlhttp = new ActiveXObject("Msxml2.XMLHTTP");
  } catch (e) {
    try {
      xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
    } catch (E) {
      xmlhttp = false;
    }
```

```
  }
  if (!xmlhttp && typeof XMLHttpRequest!='undefined') {
    xmlhttp = new XMLHttpRequest();
  }
  return xmlhttp;
} // End of function getXmlHttp()
var i,j,I,J;
var CYF='0123456789';
var NU2 = [100];  // Массив для преобразование чисел 0-99 в строку из двух
символов
 for(i=0; i<10; i++)
 {
 for(j=0; j<10; j++) NU2[i*10+j]=CYF[i]+CYF[j];
 }
var M = new Array(504); // Создаем двумерный массив M[][] для степеней
примитивного элемента
  for(i=0; i<504; i++) M[i] = new Array(504);

 var N = [504];   // Массив случайных битов N[], сформированный на нашей стороне
 var A = [504];   // Массив для отправки на сервер значения X^N[], где X-
примитивный элемент поля Галуа
 var B = [504];   // Массив битов B[], полученных от сервера
 var C = [504];   // Массив битов для шифрования, полученный как B[]^N[]
 var O = [504];   // Массив битов для наших открытых данных, которые будем
шифровать.
                        // Результат шифрования будем заносить в C[] на место
случайных битов
 var NC=0;       // К-во зашифрованных бит для передачи
 var NC1=1;      // Номер бита, с которого следует начинать/продолжать процедуру
шифрования
var T1 = new Date(); // Берем метку времени для шифрования + для проверки
времени выполнения
var TB = T1.getTime(); // TB - момент начала преобразований в милисекундах от
01.01.1970
var TIO=TB+120000;     // Установили таймаут для заполнения первой анкеты (2
минуты)
var TN=TB;                      // Проверяем на четность к-во миллисекунд от
01.01.1970
var TD = TN%2;          // В зависимости от четности заполняем первый случайный
бит
      if (TD>0) N[1]=1; else N[1]=0;

    for (i=1;i<=503;i++) M[1][i] = 0; // Обнулили первую строку массива для
примитивного элемента
    M[1][2]=1; // Занесли единицу во второй бит. Это получился примитивный
элемент поля Галуа.
    for (I=2;I<=503;I++) // Начинаем цикл заполнения массива M[][] степенями
примитивного элемента, а N[] случайными битами
    {    for (J=1;J<=503;J++)
        {
           M1[J]= M2[J]=M[I-1][J]; // Оба сомножителя равны предыдущей строке
        }
    MULT(); // Каждая следующая строка массива M[][] равна квадрату предыдущей
строки
      for (j=1;j<=503;j++) M[I][j]=R[j]; // Занесли результат умножения в
очередную строку
      var T1 = new Date();                 // Начало вычисления случайного бита
      var TN = T1.getTime();              //    Берем к-во миллисекунд от
01.01.1970
      var TD = TN%2; //  В TD вычисляем признак четности к-ва миллисекунд
      if (TD>0) N[I]=1; else N[I]=0;  // Случайный бит зависит от четности к-ва
миллисекунд
      var NN = Math.random();             // и величины стандартного случайного
числа (сумма по модулю 2)
```

```
      if (NN>0.5) N[I]=N[I]+1; if (N[I]>1) N[I]=0; // Заполнили очередной
случайный бит
    } // Конец цикла заполнения массива степенями примитивного элемента
//---- Вычисление A[]=X[]^N[], где X[]- примитивный элемент поля Галуа, N[]- 503
наших случайных бит
    for (i=1;i<=503;i++) A[i]=0; A[1]=1; // Занесли в A[] единицу
    for (J=1;J<=503;J++) // Начинаем двойной цикл возведения в степень
        if (N[J]==1)
        {
            for (I=1;I<=503;I++)
            {
                M1[I]= M[J][I];
                M2[I]=A[I];
            }
            MULT();
            for (I=1;I<=503;I++)
              A[I]=R[I];
        }  // Конец двойного цикла возведения в степень. Результат занесен в A[]
// Будем отправлять значение A[] на сервер
var TR='/Q0'; // Строка для транспортировки данных на сервер
var RT='/';  // Строка для приема ответных данных от сервера
var Nxx='X'; // Строка для номера строки данных (от 00 до 99), назначаемого
сервером
 for (i=1;i<=503;i++) // Цикл переноса значения A[] в транспортную строку
 {
 if (A[i]==0) TR=TR+'0'; else TR=TR+'1';
 }
 var xmlhttp = getXmlHttp()     // Асинхронно отправляем на сервер значение A[]
через TR
xmlhttp.open('GET', TR,  true);
xmlhttp.onreadystatechange = function()
{
  if (xmlhttp.readyState == 4)
  {
     if(xmlhttp.status == 200)
       {
        RT=xmlhttp.responseText;  // Ответные данные от сервера заносим в RT
        if (RT[0]=='N')  // Определяем верность данных по первой букве
        {
        Nxx=RT[0]+RT[1]+RT[2];  // Приняли номер строки наших данных от сервера
        for (i=1;i<=503;i++) B[i]=RT[i+2]; // Цикл заполнения значения B[]
        }
        //  alert(RT);
     }
       else {alert("Лінію зайнято. Почніть спочатку за пару хвилин.");}
  }
};
xmlhttp.send(null);
//-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
/-/-/-/-/-/-/-/-/-/
//for (j=1;j<=503;j++)  //-------- Отладочный цикл для визуализации данных
//var T2 = new Date();    // <--- Отладочная проверка времени выполнения
программы ---------------
//var TE = T2.getTime();  // TE - момент окончания преобразований в милисекундах
от 01.01.1970
//var TT = TE - TB;           // TT - время преобразований в милисекундах
// alert(TT+","+N);     // ---------- Конец проверки времени выполнения
программы ---------------
//-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
/-/-/-/-/-/-/-/-/-/
// Начало ввода и преобразования идентификационных данных клиента
var SEP; // Серия паспорта (2 большие буквы)
var NUP; // Номер паспорта (6 цифр)
var PAR; // Пароль (не менее 10 и не более 16 латинских букв или цифр)
```

```
var NR; // Номер режима (одна цифра из выпадающего меню 'REJIM')
var RPW=''; // Подготовили строку для ответа на пароль
var NTAB1=0; // К-во вызовов функции TAB1()
function TAB1()    // Обработка формы для ввода данных идентификации клиента
{
if (NTAB1>0) return;
SEP=document.getElementById('SER').value;
NUP=document.getElementById('NUMPASP').value;
PAR=document.getElementById('PAROL1').value;
NR=document.getElementById('REJIM').options.selectedIndex;
//alert(SEP+", "+NUP+", "+PAR);   /// --- Отладочный вывод -/-/-/-/-/-/-/-/-/-/-
/-/-/-/-/-/-/-/-/-/-/
var T1 = new Date(); // Берем метку времени для проверки таймаута
if (T1.getTime()>TIO) {alert("Відведений час вичерпано. Почніть спочатку.");
return;}
var TIO=T1.getTime+180000;     // Установили таймаут для голосования (3 минуты)
if ( Nxx=='X') {alert("Лінію зайнято. Почніть спочатку за пару хвилин.");
return;}
if (CONV1()==1) {NTAB1=1; CODE1(); TRAN(NC);}
else alert("Необхідно виправити помилку");
} // End of function TAB1()

function CONV1()   // Конвертирование данных идентификации клиента в битовую
последовательность
{
var i, j, k;
var n = [25];
for (i=0; i<2; i++)
      switch (SEP[i]) // Конвертуємо серію паспорту
      {
      case ' ': n[i]=0; break;
      case 'A': // Украинская кодировка и под ней латинская
      case 'A': n[i]=1; break;
      case 'B':
      case 'B': n[i]=2; break;
      case 'C':
      case 'C': n[i]=3; break;
      case 'E':
      case 'E': n[i]=4; break;
      case 'H':
      case 'H': n[i]=5; break;
      case 'I':
      case 'I': n[i]=6; break;
      case 'K':
      case 'K': n[i]=7; break;
      case 'M':
      case 'M': n[i]=8; break;
      case 'O':
      case 'O': n[i]=9; break;
      case 'P':
      case 'P': n[i]=10; break;
      case 'T':
      case 'T': n[i]=11; break;
      case 'X':
      case 'X': n[i]=12; break;
      default: return 0;
      }
      if (n[0]==0) if (n[1]!=0) return 0;
      if (n[1]==0) if (n[0]!=0) return 0;
      for (i=0; i<6; i++)
      switch (NUP[i])   // Конвертуємо номер паспорту
      {
      case '1': n[i+2]=1; break;
      case '2': n[i+2]=2; break;
```

```
      case '3': n[i+2]=3; break;
      case '4': n[i+2]=4; break;
      case '5': n[i+2]=5; break;
      case '6': n[i+2]=6; break;
      case '7': n[i+2]=7; break;
      case '8': n[i+2]=8; break;
      case '9': n[i+2]=9; break;
      case '0': n[i+2]=10; break;
      default: return 0;
      }
//    Заносим номер режима в n[8]
    n[8]=NR;
//alert(NR);/////////-/-/-/-/-/-/-/-/-/--/-/-/-/-/-/-/-/-//-/-/-/-/-/-/-/-/-
/-/-/-/-/-/-/-/-/--/-/-//-/-/-/-
j=PAR.length;
      for (i=1; i<=j; i++)
      switch (PAR[i-1])     // Конвертуємо пароль (первинній)
      {
      case ' ': n[i+8]=32; break;
      case '1': n[i+8]=49; break;
      case '2': n[i+8]=50; break;
      case '3': n[i+8]=51; break;
      case '4': n[i+8]=52; break;
      case '5': n[i+8]=53; break;
      case '6': n[i+8]=54; break;
      case '7': n[i+8]=55; break;
      case '8': n[i+8]=56; break;
      case '9': n[i+8]=57; break;
      case '0': n[i+8]=48; break;
      case 'A': n[i+8]=65; break;
      case 'B': n[i+8]=66; break;
      case 'C': n[i+8]=67; break;
      case 'D': n[i+8]=68; break;
      case 'E': n[i+8]=69; break;
      case 'F': n[i+8]=70; break;
      case 'G': n[i+8]=71; break;
      case 'H': n[i+8]=72; break;
      case 'I': n[i+8]=73; break;
      case 'J': n[i+8]=74; break;
      case 'K': n[i+8]=75; break;
      case 'L': n[i+8]=76; break;
      case 'M': n[i+8]=77; break;
      case 'N': n[i+8]=78; break;
      case 'O': n[i+8]=79; break;
      case 'P': n[i+8]=80; break;
      case 'Q': n[i+8]=81; break;
      case 'R': n[i+8]=82; break;
      case 'S': n[i+8]=83; break;
      case 'T': n[i+8]=84; break;
      case 'U': n[i+8]=85; break;
      case 'V': n[i+8]=86; break;
      case 'W': n[i+8]=87; break;
      case 'X': n[i+8]=88; break;
      case 'Y': n[i+8]=89; break;
      case 'Z': n[i+8]=90; break;
      case 'a': n[i+8]=97; break;
      case 'b': n[i+8]=98; break;
      case 'c': n[i+8]=99; break;
      case 'd': n[i+8]=100; break;
      case 'e': n[i+8]=101; break;
      case 'f': n[i+8]=102; break;
      case 'g': n[i+8]=103; break;
      case 'h': n[i+8]=104; break;
      case 'i': n[i+8]=105; break;
```

```
      case 'j': n[i+8]=106; break;
      case 'k': n[i+8]=107; break;
      case 'l': n[i+8]=108; break;
      case 'm': n[i+8]=109; break;
      case 'n': n[i+8]=110; break;
      case 'o': n[i+8]=111; break;
      case 'p': n[i+8]=112; break;
      case 'q': n[i+8]=113; break;
      case 'r': n[i+8]=114; break;
      case 's': n[i+8]=115; break;
      case 't': n[i+8]=116; break;
      case 'u': n[i+8]=117; break;
      case 'v': n[i+8]=118; break;
      case 'w': n[i+8]=119; break;
      case 'x': n[i+8]=120; break;
      case 'y': n[i+8]=121; break;
      case 'z': n[i+8]=122; break;
      default: return 0;
      }
      for (i=1; i<504; i++) O[i]=0;// Обнулили битовый массив O[] для шифрования
данных
      for (i=0; i<9; i++) // Цикл преобразования данных паспорта и NR из первых
девяти элементов n[] в O[]
      {
      if (n[i]>7) {O[i*4+4]=1; n[i]=n[i]-8;}
      if (n[i]>3) {O[i*4+3]=1; n[i]=n[i]-4;}
      if (n[i]>1) {O[i*4+2]=1; n[i]=n[i]-2;}
      O[i*4+1]=n[i];
      }
      for (k=0; k<j; k++) // Цикл преобразования пароля из элементов n[] в O[]
      { i=k+9;
      if (n[i]>63){O[k*7+43]=1; n[i]=n[i]-64;}
      if (n[i]>31){O[k*7+42]=1; n[i]=n[i]-32;}
      if (n[i]>15){O[k*7+41]=1; n[i]=n[i]-16;}
      if (n[i]>7) {O[k*7+40]=1; n[i]=n[i]-8;}
      if (n[i]>3) {O[k*7+39]=1; n[i]=n[i]-4;}
      if (n[i]>1) {O[k*7+38]=1; n[i]=n[i]-2;}
      O[k*7+37]=n[i];
      }
      NC=j*7+36;
      return 1
} // End of function CONV1()

function CONV2()   // Конвертування паролю для голосування в бітову
послідовність
{
var n = [16]; // Масив чисел для перетворення символів паролю
var i, j, k;
j=PAROL.length;
      for (i=1; i<=j; i++)
      switch (PAROL[i-1])     // Конвертуємо символи в числа
      {
      case ' ': n[i]=32; break;
      case '1': n[i]=49; break;
      case '2': n[i]=50; break;
      case '3': n[i]=51; break;
      case '4': n[i]=52; break;
      case '5': n[i]=53; break;
      case '6': n[i]=54; break;
      case '7': n[i]=55; break;
      case '8': n[i]=56; break;
      case '9': n[i]=57; break;
      case '0': n[i]=48; break;
      case 'A': n[i]=65; break;
```

```
      case 'B': n[i]=66; break;
      case 'C': n[i]=67; break;
      case 'D': n[i]=68; break;
      case 'E': n[i]=69; break;
      case 'F': n[i]=70; break;
      case 'G': n[i]=71; break;
      case 'H': n[i]=72; break;
      case 'I': n[i]=73; break;
      case 'J': n[i]=74; break;
      case 'K': n[i]=75; break;
      case 'L': n[i]=76; break;
      case 'M': n[i]=77; break;
      case 'N': n[i]=78; break;
      case 'O': n[i]=79; break;
      case 'P': n[i]=80; break;
      case 'Q': n[i]=81; break;
      case 'R': n[i]=82; break;
      case 'S': n[i]=83; break;
      case 'T': n[i]=84; break;
      case 'U': n[i]=85; break;
      case 'V': n[i]=86; break;
      case 'W': n[i]=87; break;
      case 'X': n[i]=88; break;
      case 'Y': n[i]=89; break;
      case 'Z': n[i]=90; break;
      case 'a': n[i]=97; break;
      case 'b': n[i]=98; break;
      case 'c': n[i]=99; break;
      case 'd': n[i]=100; break;
      case 'e': n[i]=101; break;
      case 'f': n[i]=102; break;
      case 'g': n[i]=103; break;
      case 'h': n[i]=104; break;
      case 'i': n[i]=105; break;
      case 'j': n[i]=106; break;
      case 'k': n[i]=107; break;
      case 'l': n[i]=108; break;
      case 'm': n[i]=109; break;
      case 'n': n[i]=110; break;
      case 'o': n[i]=111; break;
      case 'p': n[i]=112; break;
      case 'q': n[i]=113; break;
      case 'r': n[i]=114; break;
      case 's': n[i]=115; break;
      case 't': n[i]=116; break;
      case 'u': n[i]=117; break;
      case 'v': n[i]=118; break;
      case 'w': n[i]=119; break;
      case 'x': n[i]=120; break;
      case 'y': n[i]=121; break;
      case 'z': n[i]=122; break;
      default: return 0;
      }
      for (k=0; k<j; k++) // Цикл перетворенняя паролю з чисел n[] в біти O[]
(кожне число у 7 бітів)
      { i=k+1;
      if (n[i]>63){O[k*7+NC1+6]=1; n[i]=n[i]-64;}
      if (n[i]>31){O[k*7+NC1+5]=1; n[i]=n[i]-32;}
      if (n[i]>15){O[k*7+NC1+4]=1; n[i]=n[i]-16;}
      if (n[i]>7) {O[k*7+NC1+3]=1; n[i]=n[i]-8;}
      if (n[i]>3) {O[k*7+NC1+2]=1; n[i]=n[i]-4;}
      if (n[i]>1) {O[k*7+NC1+1]=1; n[i]=n[i]-2;}
      O[k*7+NC1]=n[i];
      }
```

```
      NC=j*7;
      return 1
} // End of function CONV2()

function CODE1()   // Шифрование данных идентификации клиента шифром Вернама
{
//---- Вычисление C[]=B[]^N[], где N[]- 503 наших случайных бит
 for (i=1;i<=503;i++) M[1][i]=B[i]; // Занесли B[] в M[1][ ]
    for (I=2;I<=503;I++) // Начинаем цикл заполнения массива M[][] степенями B[]
    {    for (J=1;J<=503;J++)
        {
           M1[J]= M2[J]=M[I-1][J]; // Оба сомножителя равны предыдущей строке
        }
    MULT(); // Каждая следующая строка массива M[][] равна квадрату предыдущей
строки
      for (j=1;j<=503;j++) M[I][j]=R[j]; // Занесли результат умножения в
очередную строку
    } // Конец цикла заполнения массива степенями B[]
      for (i=1;i<=503;i++) C[i]=0; C[1]=1; // Занесли в C[] единицу
    for (J=1;J<=503;J++) // Начинаем двойной цикл возведения в степень
        if (N[J]==1)
        {
            for (I=1;I<=503;I++)
            {
                M1[I]= M[J][I];
                M2[I]=C[I];
            }
            MULT();
            for (I=1;I<=503;I++)
              C[I]=R[I];
        }  // Конец двойного цикла возведения в степень. Результат занесен в C[]
// --- В C[] получена случайная последовательность для шифрования сообщений
//var TEMP='/'; // Контрольный вывод C[] /-/-/-/-/-/-/-/-/-/-//-/-/-/-/-/-/-/-/-/-
/-/-/-/-/-/-/-/--/-/-//-/-/-/-
//for (i=1;i<=503;i++) TEMP=TEMP+C[i]; ////--/-/-/-/-/-/-/-/-/-//-/-/-/-/-/-/-/-
/-/-/-/-/-/-/-/-/-/--/-/-//-/-/-/-
//alert(TEMP);/////////-/-/-/-/-/-/-/-/-/-/--/-/-/-/-/-/-/-/-/-//-/-/-/-/-/-/-/-
/-/-/-/-/-/-/-/-/-/--/-/-//-/-/-/-

for (i=1; i<=NC; i++) // Цикл шифрования по алгоритму Вернама
{C[i]=O[i]+C[i]; if (C[i]>1) C[i]=0;}
} // End of function CODE1()

function TRAN()    // Передача зашифрованных данных идентификации клиента на
сервер
{
TR="/Q1"+Nxx; // Сформировали заголовок отправляемой строки данных
for (i=NC1; i<NC1+NC; i++) TR=TR+C[i]; // и занесли зашифрованные данные
NC1=NC1+NC; // Переустановили номер бита, с которого следует начинать/продолжать
процедуру шифрования

var xmlhttp = getXmlHttp()     // Асинхронно отправляем на сервер данные клиента
через TR
xmlhttp.open('GET', TR,  true); // Открываем соединение для отправки данных на
сервер
xmlhttp.onreadystatechange = function()
{
  if (xmlhttp.readyState == 4)
  {
     if(xmlhttp.status == 200)
       {
        RT=xmlhttp.responseText;  // Ответные данные от сервера заносим в RT
        if (RT[0]=='E')  // Определяем по первой букве сообщение об ошибке
        {
```

```
        switch (RT[1])
            {
            case '0': alert("Помилкові паспортні дані.\nПочніть запит
спочатку."); break;
            case '1': alert("Пароль вже введено."); break;
            case '2': alert("Ваше право голосу вже використано."); break;
            case '3': alert("Помилковий пароль.\nПочніть запит спочатку.");
break;
            case '4': alert("Відведений час вичерпано."); break;
            case '8': alert("Довідку відправлено Вам на E-mail."); break;
            }
            return;
        }
        if (RT[0]=='S')  // Определяем верность ответа на пароль по букве S
        {
        alert("До початку занесення паролю нажміть OK.");
        }
     }
       else {alert("Лінію зайнято.\nПочніть спочатку за пару хвилин.");}
  }
};
xmlhttp.send(null);
} // End of function TRAN{)

function TRAN2()   // Шифрування і передача паролю для голосування на сервер
{
for (i=NC1; i<=NC1+NC; i++) // Цикл шифрувания
{C[i]=O[i]+C[i]; if (C[i]>1) C[i]=0;}
TR="/Q2"+Nxx; // Сформировали заголовок отправляемой строки данных
for (i=NC1; i<NC1+NC; i++) TR=TR+C[i]; // и занесли зашифрованные данные
NC1=NC1+NC;  // Переустановили номер бита, с которого следует
начинать/продолжать процедуру шифрования

var xmlhttp = getXmlHttp()     // Асинхронно отправляем на сервер данные клиента
через TR
xmlhttp.open('GET', TR,  true); // Открываем соединение для отправки данных на
сервер
xmlhttp.onreadystatechange = function()
{
  if (xmlhttp.readyState == 4)
  {
     if(xmlhttp.status == 200)
       {
        RT=xmlhttp.responseText;  // Ответные данные от сервера заносим в RT
        if (RT[0]=='E')  // Определяем по первой букве принятие сообщения
        {
        switch (RT[1])
            {
            case '0': alert("Пароль для голосування прийнято."); break;
            case '4': alert("Відведений час вичерпано.\nПочніть запит
спочатку."); break;
            }
            return;
        }

     }
       else {alert("Лінію зайнято.\nПочніть спочатку за пару хвилин.");}
  }
};
xmlhttp.send(null);
} // End of function TRAN2{)

function TAB2()    // Обробка форми для ввода паролю для голосування
{
```

```
var i=0;
PAROL=document.getElementById('PAROL').value;
REPEAT=document.getElementById('REPEAT').value;
if (PAROL!=REPEAT){alert("Заново введіть пароль"); return;}
if (CONV2()==1){TRAN2();}
else alert("Необхідно виправити помилку");
} // End of function TAB2()
</script>
 <table border="1"   bgcolor="#e0e0e0">
   <caption> </caption>
   <tr>
    <th>Введіть дані свого паспорту</th>
    </tr>
   <tr><td>Серія: <input id="SER" type="text" size="1" pattern="[A-ZA-Я]{2}" >
</td></tr>
   <tr><td>Номер:   <input type="text" size="5" pattern="[0-9]{6}"
id="NUMPASP"></td></tr>
   <tr><td>Пароль:  <input type="password" pattern="[A-Za-z0-9\s,.<>]{10,16}"
id="PAROL1"></td></tr>
   <tr><td>Оберіть потрібну дію:
<div>
   <select id="REJIM">
<option value="0">Введення паролю для голосування</option>
<option value="1">Введення паролю дільничної пошти</option>
</select>
</div>
   </td></tr>
  <tr><td align="center">    <input type="button" value="Дані введено"
onClick=TAB1()> </td></tr>
  </table>
<br>
<table border="1"  bgcolor="#e0e0e0">
   <caption> </caption>
   <tr>
    <th>Заповнюйте після запрошення</th>
    </tr>
       <tr>
    <th>(тільки латинь та цифри від 10 до 16) </th>
    </tr>
   <tr><td>*Пароль<input type="password" pattern="[A-Za-z0-9\s,.<>]{10,16}"
id="PAROL"></td></tr>
   <tr><td>*Ще раз<input type="password" pattern="[A-Za-z0-9\s,.<>]{10,16}"
id="REPEAT"></td></tr>
  <tr><td align="center">    <input type="button" value="Пароль для голосування
введено" onClick=TAB2()> </td></tr>
  </table>
</body>
</html>
```

Д1.6. Клієнтська програма для голосування та отримання довідок (файл anketa.html)

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
 <title> Виборча дільниця НАУ</title>
</head>
<body>
 <h2>Виборча дільниця НАУ</h2>
<script type="text/javascript">
var KOD; // код из Email (или СМС) (6 цифр)
var NBUL=0; // Номер бюлетеня
var VYB=[9]; // Обраний номер пункту по кожному з бюлетенів
var M1 = [504];   // Резервируем ячейки для элементов поля Галуа GF(2^503)
var M2 = [504];   // M1[], M2[] - сомножители  R[] - результат умножения
var R  = [504];   // Младший бит храним в [1], старший - в [503]. Нулевые
элементы не используем
function MULT()     // Умножение элементов поля Галуа GF(2^503) по правилу
полиномов
{
 var i,j,r,r1,r2,r3;
    for (i=1;i<=503;i++) R[i]=0; // Обнуляем 503 ячейки для битов результата
    for (i=1;i<=503;i++)            // Начинаем двойной цикл умножения элементов
полинома
    if (M1[i]==1) // Умножаем только единичные элементы из M1[], нулевые не
нужны, т.к. они дают нули
    {
        for (j=1;j<=503;j++)
        if (M2[j]==1)
        {
        r=i+j-1;
            if (r>503)
            {
             r=r-503;
             if (r>=501)
                {
                r=r-501;
                r1=1+r; r2=4+r; r3=501+r;
                if (R[r3]==0) R[r3]=1; else R[r3]=0;
                }
            else {r1=r; r2=r+3;}
            if (R[r1]==0) R[r1]=1; else R[r1]=0;
            if (R[r2]==0) R[r2]=1; else R[r2]=0;
            }
            else {if (R[r]==0) R[r]=1; else R[r]=0;}
        }
    }
} // End of function MULT()

function getXmlHttp() // Кроссбраузерная функция создания XMLHttpRequest
{ // С помощью этой функции осуществляем обмен данными с сервером
  var xmlhttp;
  try {
    xmlhttp = new ActiveXObject("Msxml2.XMLHTTP");
  } catch (e) {
    try {
      xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
    } catch (E) {
      xmlhttp = false;
    }
  }
  if (!xmlhttp && typeof XMLHttpRequest!='undefined') {
```

```
    xmlhttp = new XMLHttpRequest();
  }
  return xmlhttp;
} // End of function getXmlHttp()
var i,j,I,J;
var CYF='0123456789';
var NU2 = [100];  // Массив для преобразование чисел 0-99 в строку из двух
символов
 for(i=0; i<10; i++)
 {
 for(j=0; j<10; j++) NU2[i*10+j]=CYF[i]+CYF[j];
 }
var M = new Array(504); // Создаем двумерный массив M[][] для степеней
примитивного элемента
  for(i=0; i<504; i++) M[i] = new Array(504);

 var N = [504];   // Массив случайных битов N[], сформированный на нашей стороне
 var A = [504];   // Массив для отправки на сервер значения X^N[], где X-
примитивный элемент поля Галуа
 var B = [504];   // Массив битов B[], полученных от сервера
 var C = [504];   // Массив битов для шифрования, полученный как B[]^N[]
 var O = [504];   // Массив битов для наших открытых данных, которые будем
шифровать.
                        // Результат шифрования будем заносить в C[] на место
случайных битов
 var NC=0;      // К-во зашифрованных бит для передачи
 var NC1=1;     // Номер бита, с которого следует начинать/продолжать процедуру
шифрования
var T1 = new Date(); // Берем метку времени для шифрования + для проверки
времени выполнения
var TB = T1.getTime(); // TB - момент начала преобразований в милисекундах от
01.01.1970
var TIO=TB+120000;     // Установили таймаут для заполнения первой анкеты (2
минуты)
var TN=TB;                     // Проверяем на четность к-во миллисекунд от
01.01.1970
var TD = TN%2;          // В зависимости от четности заполняем первый случайный
бит
      if (TD>0) N[1]=1; else N[1]=0;

    for (i=1;i<=503;i++) M[1][i] = 0; // Обнулили первую строку массива для
примитивного элемента
    M[1][2]=1; // Занесли единицу во второй бит. Это получился примитивный
элемент поля Галуа.
    for (I=2;I<=503;I++) // Начинаем цикл заполнения массива M[][] степенями
примитивного элемента, а N[] случайными битами
    {    for (J=1;J<=503;J++)
        {
           M1[J]= M2[J]=M[I-1][J]; // Оба сомножителя равны предыдущей строке
        }
    MULT(); // Каждая следующая строка массива M[][] равна квадрату предыдущей
строки
      for (j=1;j<=503;j++) M[I][j]=R[j]; // Занесли результат умножения в
очередную строку
      var T1 = new Date();                 // Начало вычисления случайного бита
      var TN = T1.getTime();              //    Берем к-во миллисекунд от
01.01.1970
      var TD = TN%2; //  В TD вычисляем признак четности к-ва миллисекунд
      if (TD>0) N[I]=1; else N[I]=0;  // Случайный бит зависит от четности к-ва
миллисекунд
      var NN = Math.random();             // и величины стандартного случайного
числа (сумма по модулю 2)
      if (NN>0.5) N[I]=N[I]+1; if (N[I]>1) N[I]=0; // Заполнили очередной
случайный бит
```

```
    } // Конец цикла заполнения массива степенями примитивного элемента
//---- Вычисление A[]=X[]^N[], где X[]- примитивный элемент поля Галуа, N[]- 503
наших случайных бит
    for (i=1;i<=503;i++) A[i]=0; A[1]=1; // Занесли в A[] единицу
    for (J=1;J<=503;J++) // Начинаем двойной цикл возведения в степень
        if (N[J]==1)
        {
            for (I=1;I<=503;I++)
            {
                M1[I]= M[J][I];
                M2[I]=A[I];
            }
            MULT();
            for (I=1;I<=503;I++)
              A[I]=R[I];
        }  // Конец двойного цикла возведения в степень. Результат занесен в A[]
// Будем отправлять значение A[] на сервер
var TR='/Q0'; // Строка для транспортировки данных на сервер
var RT='/';  // Строка для приема ответных данных от сервера
var Nxx='X'; // Строка для номера строки данных (от 00 до 99), назначаемого
сервером
 for (i=1;i<=503;i++) // Цикл переноса значения A[] в транспортную строку
 {
 if (A[i]==0) TR=TR+'0'; else TR=TR+'1';
 }
 var xmlhttp = getXmlHttp()     // Асинхронно отправляем на сервер значение A[]
через TR
xmlhttp.open('GET', TR,  true);
xmlhttp.onreadystatechange = function()
{
  if (xmlhttp.readyState == 4)
  {
     if(xmlhttp.status == 200)
       {
        RT=xmlhttp.responseText;  // Ответные данные от сервера заносим в RT
        if (RT[0]=='N')  // Определяем верность данных по первой букве
        {
        Nxx=RT[0]+RT[1]+RT[2];  // Приняли номер строки наших данных от сервера
        for (i=1;i<=503;i++) B[i]=RT[i+2]; // Цикл заполнения значения B[]
        }
        //  alert(RT);
     }
       else {alert("Лінію зайнято. Почніть спочатку за пару хвилин.");}
  }
};
xmlhttp.send(null);
//-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
/-/-/-/-/-/-/-/-/-/
//for (j=1;j<=503;j++)  //-------- Отладочный цикл для визуализации данных
//var T2 = new Date();    // <--- Отладочная проверка времени выполнения
программы ---------------
//var TE = T2.getTime();  // TE - момент окончания преобразований в милисекундах
от 01.01.1970
//var TT = TE - TB;           // TT - время преобразований в милисекундах
// alert(TT+","+N);     // ---------- Конец проверки времени выполнения
программы ---------------
//-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-/-
/-/-/-/-/-/-/-/-/-/
// Начало ввода и преобразования идентификационных данных клиента
var SEP; // Серия паспорта (2 большие буквы)
var NUP; // Номер паспорта (6 цифр)
var PAR; // Пароль (не менее 10 и не более 16 латинских букв или цифр)
var NR; // Номер режима (одна цифра из выпадающего меню 'REJIM')
var RPW=''; // Подготовили строку для ответа на пароль
```

```
var NTAB1=0; // К-во вызовов функции TAB1()
function TAB1()    // Обработка формы для ввода данных идентификации клиента
{
if (NTAB1>0) return;
SEP=document.getElementById('SER').value;
NUP=document.getElementById('NUMPASP').value;
PAR=document.getElementById('PAROL').value;
NR=document.getElementById('REJIM').options.selectedIndex;
//alert(SEP+", "+NUP+", "+PAR);   /// --- Отладочный вывод -/-/-/-/-/-/-/-/-/-/-
/-/-/-/-/-/-/-/-/-/
var T1 = new Date(); // Берем метку времени для проверки таймаута
if (T1.getTime()>TIO) {alert("Відведений час вичерпано. Почніть спочатку.");
return;}
var TIO=T1.getTime+180000;     // Установили таймаут для голосования (3 минуты)
if ( Nxx=='X') {alert("Лінію зайнято. Почніть спочатку за пару хвилин.");
return;}
if (CONV1()==1) {NTAB1=1; CODE1(); TRAN(NC);}
else alert("Необхідно виправити помилку");
} // End of function TAB1()

function CONV1()   // Конвертирование данных идентификации клиента в битовую
последовательность
{

var i, j, k;
var n = [25];
for (i=0; i<2; i++)
      switch (SEP[i])  // Конвертуємо серію паспорту
      {
      case ' ': n[i]=0; break;
      case 'A': // Украинская кодировка и под ней латинская
      case 'A': n[i]=1; break;
      case 'B':
      case 'B': n[i]=2; break;
      case 'C':
      case 'C': n[i]=3; break;
      case 'E':
      case 'E': n[i]=4; break;
      case 'H':
      case 'H': n[i]=5; break;
      case 'I':
      case 'I': n[i]=6; break;
      case 'K':
      case 'K': n[i]=7; break;
      case 'M':
      case 'M': n[i]=8; break;
      case 'O':
      case 'O': n[i]=9; break;
      case 'P':
      case 'P': n[i]=10; break;
      case 'T':
      case 'T': n[i]=11; break;
      case 'X':
      case 'X': n[i]=12; break;
      default: return 0;
      }
      if (n[0]==0) if (n[1]!=0) return 0;
      if (n[1]==0) if (n[0]!=0) return 0;
      for (i=0; i<6; i++)
      switch (NUP[i])    // Конвертуємо номер паспорту
      {
      case '1': n[i+2]=1; break;
      case '2': n[i+2]=2; break;
      case '3': n[i+2]=3; break;
```

```
      case '4': n[i+2]=4; break;
      case '5': n[i+2]=5; break;
      case '6': n[i+2]=6; break;
      case '7': n[i+2]=7; break;
      case '8': n[i+2]=8; break;
      case '9': n[i+2]=9; break;
      case '0': n[i+2]=10; break;
      default: return 0;
      }
//    Заносим номер режима в n[8]
    n[8]=NR;
//alert(NR);/////////-/-/-/-/-/-/-/-/-/-/--/-/-/-/-/-/-/-/-//-/-/-/-/-/-/-/-/-
/-/-/-/-/-/-/-/-/--/-/-//-/-/-/-
j=PAR.length;
      for (i=1; i<=j; i++)
      switch (PAR[i-1])
      {
      case ' ': n[i+8]=32; break;
      case '1': n[i+8]=49; break;
      case '2': n[i+8]=50; break;
      case '3': n[i+8]=51; break;
      case '4': n[i+8]=52; break;
      case '5': n[i+8]=53; break;
      case '6': n[i+8]=54; break;
      case '7': n[i+8]=55; break;
      case '8': n[i+8]=56; break;
      case '9': n[i+8]=57; break;
      case '0': n[i+8]=48; break;
      case 'A': n[i+8]=65; break;
      case 'B': n[i+8]=66; break;
      case 'C': n[i+8]=67; break;
      case 'D': n[i+8]=68; break;
      case 'E': n[i+8]=69; break;
      case 'F': n[i+8]=70; break;
      case 'G': n[i+8]=71; break;
      case 'H': n[i+8]=72; break;
      case 'I': n[i+8]=73; break;
      case 'J': n[i+8]=74; break;
      case 'K': n[i+8]=75; break;
      case 'L': n[i+8]=76; break;
      case 'M': n[i+8]=77; break;
      case 'N': n[i+8]=78; break;
      case 'O': n[i+8]=79; break;
      case 'P': n[i+8]=80; break;
      case 'Q': n[i+8]=81; break;
      case 'R': n[i+8]=82; break;
      case 'S': n[i+8]=83; break;
      case 'T': n[i+8]=84; break;
      case 'U': n[i+8]=85; break;
      case 'V': n[i+8]=86; break;
      case 'W': n[i+8]=87; break;
      case 'X': n[i+8]=88; break;
      case 'Y': n[i+8]=89; break;
      case 'Z': n[i+8]=90; break;
      case 'a': n[i+8]=97; break;
      case 'b': n[i+8]=98; break;
      case 'c': n[i+8]=99; break;
      case 'd': n[i+8]=100; break;
      case 'e': n[i+8]=101; break;
      case 'f': n[i+8]=102; break;
      case 'g': n[i+8]=103; break;
      case 'h': n[i+8]=104; break;
      case 'i': n[i+8]=105; break;
      case 'j': n[i+8]=106; break;
```

```
      case 'k': n[i+8]=107; break;
      case 'l': n[i+8]=108; break;
      case 'm': n[i+8]=109; break;
      case 'n': n[i+8]=110; break;
      case 'o': n[i+8]=111; break;
      case 'p': n[i+8]=112; break;
      case 'q': n[i+8]=113; break;
      case 'r': n[i+8]=114; break;
      case 's': n[i+8]=115; break;
      case 't': n[i+8]=116; break;
      case 'u': n[i+8]=117; break;
      case 'v': n[i+8]=118; break;
      case 'w': n[i+8]=119; break;
      case 'x': n[i+8]=120; break;
      case 'y': n[i+8]=121; break;
      case 'z': n[i+8]=122; break;
      default: return 0;
      }
      for (i=1; i<504; i++) O[i]=0;// Обнулили битовый массив O[] для шифрования
данных
      for (i=0; i<9; i++) // Цикл преобразования данных паспорта и NR из первых
девяти элементов n[] в O[]
      {
      if (n[i]>7) {O[i*4+4]=1; n[i]=n[i]-8;}
      if (n[i]>3) {O[i*4+3]=1; n[i]=n[i]-4;}
      if (n[i]>1) {O[i*4+2]=1; n[i]=n[i]-2;}
      O[i*4+1]=n[i];
      }
      for (k=0; k<j; k++) // Цикл преобразования пароля из элементов n[] в O[]
      { i=k+9;
      if (n[i]>63){O[k*7+43]=1; n[i]=n[i]-64;}
      if (n[i]>31){O[k*7+42]=1; n[i]=n[i]-32;}
      if (n[i]>15){O[k*7+41]=1; n[i]=n[i]-16;}
      if (n[i]>7) {O[k*7+40]=1; n[i]=n[i]-8;}
      if (n[i]>3) {O[k*7+39]=1; n[i]=n[i]-4;}
      if (n[i]>1) {O[k*7+38]=1; n[i]=n[i]-2;}
      O[k*7+37]=n[i];
      }
      NC=j*7+36;
      return 1
} // End of function CONV1()

function CONV2()   // Конвертирование результатов голосования в битовую
последовательность
{
var i, j, k;
var n = [24]; // Массив чисел, соответствующих цифрам кода и результатов
голоования
      for (i=0; i<6; i++) // Цикл преобразования цифр кода (из Email) в числовую
форму
      switch (KOD[i])
      {
      case '1': n[i]=1; break;
      case '2': n[i]=2; break;
      case '3': n[i]=3; break;
      case '4': n[i]=4; break;
      case '5': n[i]=5; break;
      case '6': n[i]=6; break;
      case '7': n[i]=7; break;
      case '8': n[i]=8; break;
      case '9': n[i]=9; break;
      case '0': n[i]=10; break;
      default: return 0;
      }
```

```
      for (j=0; j<NBUL; j++) // Цикл по бюллетням
      {
      for (i=0; i<2; i++) // Цикл по двум цифрам номера, выбранного в бюллетне
      switch (NU2[VYB[j+1]][i])
      {
      case '1': n[6+j*2+i]=1; break;
      case '2': n[6+j*2+i]=2; break;
      case '3': n[6+j*2+i]=3; break;
      case '4': n[6+j*2+i]=4; break;
      case '5': n[6+j*2+i]=5; break;
      case '6': n[6+j*2+i]=6; break;
      case '7': n[6+j*2+i]=7; break;
      case '8': n[6+j*2+i]=8; break;
      case '9': n[6+j*2+i]=9; break;
      case '0': n[6+j*2+i]=10; break;
      }
      }


      for (i=0; i<8+NBUL*2; i++) // Цикл преобразования данных из n[] в битовую
последовательность O[]
      {
      if (n[i]>7) {O[NC1+i*4+3]=1; n[i]=n[i]-8;}
      if (n[i]>3) {O[NC1+i*4+2]=1; n[i]=n[i]-4;}
      if (n[i]>1) {O[NC1+i*4+1]=1; n[i]=n[i]-2;}
                        O[NC1+i*4+0]=n[i];
      }
      NC=24+NBUL*8; // Вычислили к-во бит для шифрования
      return 1
} // End of function CONV2()

function CODE1()   // Шифрование данных идентификации клиента шифром Вернама
{
//---- Вычисление C[]=B[]^N[], где N[]- 503 наших случайных бит
 for (i=1;i<=503;i++) M[1][i]=B[i]; // Занесли B[] в M[1][ ]
    for (I=2;I<=503;I++) // Начинаем цикл заполнения массива M[][] степенями B[]
    {    for (J=1;J<=503;J++)
        {
           M1[J]= M2[J]=M[I-1][J]; // Оба сомножителя равны предыдущей строке
        }
    MULT(); // Каждая следующая строка массива M[][] равна квадрату предыдущей
строки
      for (j=1;j<=503;j++) M[I][j]=R[j]; // Занесли результат умножения в
очередную строку
    } // Конец цикла заполнения массива степенями B[]
      for (i=1;i<=503;i++) C[i]=0; C[1]=1; // Занесли в C[] единицу
    for (J=1;J<=503;J++) // Начинаем двойной цикл возведения в степень
        if (N[J]==1)
        {
            for (I=1;I<=503;I++)
            {
                M1[I]= M[J][I];
                M2[I]=C[I];
            }
            MULT();
            for (I=1;I<=503;I++)
              C[I]=R[I];
        }  // Конец двойного цикла возведения в степень. Результат занесен в C[]
// --- В C[] получена случайная последовательность для шифрования сообщений
//var TEMP='/'; // Контрольный вывод C[] /-/-/-/-/-/-/-/-/-/-//-/-/-/-/-/-/-/-/-/-
/-/-/-/-/-/-/-/-/--/-/-//-/-/-/-
//for (i=1;i<=503;i++) TEMP=TEMP+C[i]; ////--/-/-/-/-/-/-/-/-/-/-//-/-/-/-/-/-/-/-
/-/-/-/-/-/-/-/-/-/-/--/-/-//-/-/-/-
```

```
//alert(TEMP);/////////-/-/-/-/-/-/-/-/-/-/--/-/-/-/-/-/-/-/-/-//-/-/-/-/-/-/-/-
/-/-/-/-/-/-/-/-/-/-/--/-/-//-/-/-/-

for (i=1; i<=NC; i++) // Цикл шифрования по алгоритму Вернама
{C[i]=O[i]+C[i]; if (C[i]>1) C[i]=0;}
} // End of function CODE1()

function TRAN()    // Передача зашифрованных данных идентификации клиента на
сервер
{
TR="/Q1"+Nxx; // Сформировали заголовок отправляемой строки данных
for (i=NC1; i<NC1+NC; i++) TR=TR+C[i]; // и занесли зашифрованные данные
NC1=NC1+NC; // Переустановили номер бита, с которого следует начинать/продолжать
процедуру шифрования

var xmlhttp = getXmlHttp()     // Асинхронно отправляем на сервер данные клиента
через TR
xmlhttp.open('GET', TR,  true); // Открываем соединение для отправки данных на
сервер
xmlhttp.onreadystatechange = function()
{
  if (xmlhttp.readyState == 4)
  {
     if(xmlhttp.status == 200)
       {
        RT=xmlhttp.responseText;  // Ответные данные от сервера заносим в RT
        if (RT[0]=='E')  // Определяем по первой букве сообщение об ошибке
        {
        switch (RT[1])
            {
            case '0': alert("Помилкові паспортні дані.\nПочніть запит
спочатку."); break;
            case '1': alert("Вам не надано права голосу."); break;
            case '2': alert("Ваше право голосу вже використано."); break;
            case '3': alert("Помилковий пароль.\nПочніть запит спочатку.");
break;
            case '4': alert("Відведений час вичерпано.\nПочніть запит
спочатку."); break;
            case '8': alert("Довідку відправлено Вам на E-mail."); break;
            }
            return;
        }
        if (RT[0]=='S')  // Определяем верность ответа на пароль по букве S
        {
        alert("Для початку голосування нажміть OK.");
        }
     }
       else {alert("Лінію зайнято.\nПочніть спочатку за пару хвилин.");}
  }
};
xmlhttp.send(null);
} // End of function TRAN{)

function TRAN2()   // Шифрование и передача результата голосования на сервер
{
for (i=NC1; i<=NC1+NC; i++) // Цикл шифрования результатов голосования
{C[i]=O[i]+C[i]; if (C[i]>1) C[i]=0;}
TR="/Q2"+Nxx; // Сформировали заголовок отправляемой строки данных
for (i=NC1; i<NC1+NC; i++) TR=TR+C[i]; // и занесли зашифрованные данные
NC1=NC1+NC;  // Переустановили номер бита, с которого следует
начинать/продолжать процедуру шифрования

var xmlhttp = getXmlHttp()     // Асинхронно отправляем на сервер данные клиента
через TR
```

```
xmlhttp.open('GET', TR,  true); // Открываем соединение для отправки данных на
сервер
xmlhttp.onreadystatechange = function()
{
  if (xmlhttp.readyState == 4)
  {
     if(xmlhttp.status == 200)
       {
        RT=xmlhttp.responseText;  // Ответные данные от сервера заносим в RT
        if (RT[0]=='E')  // Определяем по первой букве принятие сообщения
        {
        switch (RT[1])
            {
            case '0': alert("Ваш голос прийнято і враховано."); break;
            case '1': alert("Голос відхилено через помилку коду.\nПочніть запит
спочатку."); break;
            case '2': alert("Ваше право голосу вже використано."); break;
            case '4': alert("Відведений час вичерпано.\nПочніть запит
спочатку."); break;
            }
            return;
        }

     }
       else {alert("Лінію зайнято.\nПочніть спочатку за пару хвилин.");}
  }
};
xmlhttp.send(null);
} // End of function TRAN2{)

function TAB2()    // Обработка формы для ввода кода из Email (или СМС)
{
var T1 = new Date(); // Берем метку времени для проверки таймаута
if (T1.getTime()>TIO) {alert("Відведений час вичерпано. Почніть спочатку.");
return;}
KOD=document.getElementById('KOD').value;
NBUL=1;  ////////////// Блок обробки бюлетня № 1 ////////////////////
var NRB=document.getElementsByName("bul1");
for (i=0; i<NRB.length;i++)
{if(NRB[i].checked)  break;}
VYB[NBUL]=NRB[i].value;
NBUL=2;     ////////////// Блок обробки бюлетня № 2 ////////////////////
var NRB=document.getElementsByName("bul2");
for (i=0; i<NRB.length;i++)
{if(NRB[i].checked)  break;}
VYB[NBUL]=NRB[i].value;
NBUL=3;     ////////////// Блок обробки бюлетня № 3 ////////////////////
var NRB=document.getElementsByName("bul3");
for (i=0; i<NRB.length;i++)
{if(NRB[i].checked)  break;}
VYB[NBUL]=NRB[i].value;
NBUL=4;     ////////////// Блок обробки бюлетня № 4 ////////////////////
var NRB=document.getElementsByName("bul4");
for (i=0; i<NRB.length;i++)
{if(NRB[i].checked)  break;}
VYB[NBUL]=NRB[i].value;
NBUL=5;     ////////////// Блок обробки бюлетня № 5 ////////////////////
var NRB=document.getElementsByName("bul5");
for (i=0; i<NRB.length;i++)
{if(NRB[i].checked)  break;}
VYB[NBUL]=NRB[i].value;
NBUL=6;     ////////////// Блок обробки бюлетня № 6 ////////////////////
var NRB=document.getElementsByName("bul6");
for (i=0; i<NRB.length;i++)
```

```
{if(NRB[i].checked)  break;}
VYB[NBUL]=NRB[i].value;
NBUL=7;     /////////////// Блок обробки бюлетня № 7 ////////////////////
var NRB=document.getElementsByName("bul7");
for (i=0; i<NRB.length;i++)
{if(NRB[i].checked)  break;}
VYB[NBUL]=NRB[i].value;
NBUL=8;     /////////////// Блок обробки бюлетня № 8 ////////////////////
var NRB=document.getElementsByName("bul8");
for (i=0; i<NRB.length;i++)
{if(NRB[i].checked)  break;}
VYB[NBUL]=NRB[i].value;
NBUL=9;     /////////////// Блок обробки бюлетня № 9 ////////////////////
var NRB=document.getElementsByName("bul9");
for (i=0; i<NRB.length;i++)
{if(NRB[i].checked)  break;}
VYB[NBUL]=NRB[i].value;
///////////////  Обробку бюлетнів закінчено  ///////////////////////
if (CONV2()==1) TRAN2();
else alert("Необхідно виправити помилку");
}
</script>
 <table border="1"   bgcolor="#e0e0e0">
   <caption> </caption>
   <tr>
    <th>Введіть дані свого паспорту</th>
    </tr>
   <tr><td>Серія: <input id="SER" type="text" size="1" pattern="[A-ZA-Я]{2}" >
</td></tr>
   <tr><td>Номер:   <input type="text" size="5" pattern="[0-9]{6}"
id="NUMPASP"></td></tr>
   <tr><td>Пароль:  <input type="text" pattern="[A-Za-z0-9\s,.<>]{10,16}"
id="PAROL"></td></tr>
   <tr><td>Оберіть потрібну дію:
<div>
   <select id="REJIM">
<option value="0">Голосування</option>
<option value="1">Довідка по дільниці</option>
<option disabled value="2">Довідка по вулиці</option>
<option disabled value="3">Довідка по будинку</option>
</select>
</div>
   </td></tr>
  <tr><td align="center">    <input type="button" value="Дані введено"
onClick=TAB1()> </td></tr>
  </table>
<br>

<div>

 <p><b>Бюлетень № 1 про ставлення до дисципліни "Математика"</b><Br>
  <label> <input type="radio" name="bul1" value="1"/> Вважаю, що ця дисципліна
дуже потрібна і вимагає збільшення обсягу викладання    </label> <Br>
  <label> <input type="radio" name="bul1" value="2"/> Вважаю, що ця дисципліна
потрібна і обсяг викладання є достатнім                  </label> <Br>
  <label> <input type="radio" name="bul1" value="3"/> Вважаю, що ця дисципліна
потрібна, але обсяг викладання треба зменшити            </label> <Br>
  <label> <input type="radio" name="bul1" value="4"/> Вважаю, що ця дисципліна
для мене не дуже потрібна, але є потреба в її викладанні</label> <Br>
  <label> <input type="radio" name="bul1" value="5"/> Вважаю, що вивчення цієї
дисципліни мені непотрібно                               </label> <Br>
  <label> <input type="radio" name="bul1" value="0" checked/> Відмовляюсь від
вибору, бо не вивчав </label> <Br>
  </p>
```

```
  <p><b>Бюлетень № 2 про ставлення до дисципліни "Основи електроніки"</b><Br>
  <label> <input type="radio" name="bul2" value="1"/> Вважаю, що ця дисципліна
дуже потрібна і вимагає збільшення обсягу викладання    </label> <Br>
  <label> <input type="radio" name="bul2" value="2"/> Вважаю, що ця дисципліна
потрібна і обсяг викладання є достатнім                  </label> <Br>
  <label> <input type="radio" name="bul2" value="3"/> Вважаю, що ця дисципліна
потрібна, але обсяг викладання треба зменшити            </label> <Br>
  <label> <input type="radio" name="bul2" value="4"/> Вважаю, що ця дисципліна
для мене не дуже потрібна, але є потреба в її викладанні</label> <Br>
  <label> <input type="radio" name="bul2" value="5"/> Вважаю, що вивчення цієї
дисципліни мені непотрібно                               </label> <Br>
  <label> <input type="radio" name="bul2" value="0" checked/> Відмовляюсь від
вибору, бо не вивчав</label> <Br>
  </p>

  <p><b>Бюлетень № 3 про ставлення до дисципліни "Основи схемотехніки"</b><Br>
  <label> <input type="radio" name="bul3" value="1"/> Вважаю, що ця дисципліна
дуже потрібна і вимагає збільшення обсягу викладання    </label> <Br>
  <label> <input type="radio" name="bul3" value="2"/> Вважаю, що ця дисципліна
потрібна і обсяг викладання є достатнім                  </label> <Br>
  <label> <input type="radio" name="bul3" value="3"/> Вважаю, що ця дисципліна
потрібна, але обсяг викладання треба зменшити            </label> <Br>
  <label> <input type="radio" name="bul3" value="4"/> Вважаю, що ця дисципліна
для мене не дуже потрібна, але є потреба в її викладанні</label> <Br>
  <label> <input type="radio" name="bul3" value="5"/> Вважаю, що вивчення цієї
дисципліни мені непотрібно                               </label> <Br>
  <label> <input type="radio" name="bul3" value="0" checked/> Відмовляюсь від
вибору, бо не вивчав </label> <Br>
  </p>

  <p><b>Бюлетень № 4 про ставлення до дисципліни "Теорія
электро'звязку"</b><Br>
  <label> <input type="radio" name="bul4" value="1"/> Вважаю, що ця дисципліна
дуже потрібна і вимагає збільшення обсягу викладання    </label> <Br>
  <label> <input type="radio" name="bul4" value="2"/> Вважаю, що ця дисципліна
потрібна і обсяг викладання є достатнім                  </label> <Br>
  <label> <input type="radio" name="bul4" value="3"/> Вважаю, що ця дисципліна
потрібна, але обсяг викладання треба зменшити            </label> <Br>
  <label> <input type="radio" name="bul4" value="4"/> Вважаю, що ця дисципліна
для мене не дуже потрібна, але є потреба в її викладанні</label> <Br>
  <label> <input type="radio" name="bul4" value="5"/> Вважаю, що вивчення цієї
дисципліни мені непотрібно                               </label> <Br>
  <label> <input type="radio" name="bul4" value="0" checked/> Відмовляюсь від
вибору, бо не вивчав </label> <Br>
  </p>

  <p><b>Бюлетень № 5 про ставлення до дисципліни "Іформаційні та
телекомуніікаційні мережі"</b><Br>
  <label> <input type="radio" name="bul5" value="1"/> Вважаю, що ця дисципліна
дуже потрібна і вимагає збільшення обсягу викладання    </label> <Br>
  <label> <input type="radio" name="bul5" value="2"/> Вважаю, що ця дисципліна
потрібна і обсяг викладання є достатнім                  </label> <Br>
  <label> <input type="radio" name="bul5" value="3"/> Вважаю, що ця дисципліна
потрібна, але обсяг викладання треба зменшити            </label> <Br>
  <label> <input type="radio" name="bul5" value="4"/> Вважаю, що ця дисципліна
для мене не дуже потрібна, але є потреба в її викладанні</label> <Br>
  <label> <input type="radio" name="bul5" value="5"/> Вважаю, що вивчення цієї
дисципліни мені непотрібно                               </label> <Br>
  <label> <input type="radio" name="bul5" value="0" checked/> Відмовляюсь від
вибору, бо не вивчав </label> <Br>
  </p>
```

```
  <p><b>Бюлетень № 6 про ставлення до дисципліни "Системи авіаційного
радіо'звязку"</b><Br>
  <label> <input type="radio" name="bul6" value="1"/> Вважаю, що ця дисципліна
дуже потрібна і вимагає збільшення обсягу викладання    </label> <Br>
  <label> <input type="radio" name="bul6" value="2"/> Вважаю, що ця дисципліна
потрібна і обсяг викладання є достатнім                  </label> <Br>
  <label> <input type="radio" name="bul6" value="3"/> Вважаю, що ця дисципліна
потрібна, але обсяг викладання треба зменшити            </label> <Br>
  <label> <input type="radio" name="bul6" value="4"/> Вважаю, що ця дисципліна
для мене не дуже потрібна, але є потреба в її викладанні</label> <Br>
  <label> <input type="radio" name="bul6" value="5"/> Вважаю, що вивчення цієї
дисципліни мені непотрібно                               </label> <Br>
  <label> <input type="radio" name="bul6" value="0" checked/> Відмовляюсь від
вибору, бо не вивчав </label> <Br>
  </p>

  <p><b>Бюлетень № 7 про ставлення до дисципліни "Основи телебачення та
радіомовлення"</b><Br>
  <label> <input type="radio" name="bul7" value="1"/> Вважаю, що ця дисципліна
дуже потрібна і вимагає збільшення обсягу викладання    </label> <Br>
  <label> <input type="radio" name="bul7" value="2"/> Вважаю, що ця дисципліна
потрібна і обсяг викладання є достатнім                  </label> <Br>
  <label> <input type="radio" name="bul7" value="3"/> Вважаю, що ця дисципліна
потрібна, але обсяг викладання треба зменшити            </label> <Br>
  <label> <input type="radio" name="bul7" value="4"/> Вважаю, що ця дисципліна
для мене не дуже потрібна, але є потреба в її викладанні</label> <Br>
  <label> <input type="radio" name="bul7" value="5"/> Вважаю, що вивчення цієї
дисципліни мені непотрібно                               </label> <Br>
  <label> <input type="radio" name="bul7" value="0" checked/> Відмовляюсь від
вибору, бо не вивчав </label> <Br>
  </p>

  <p><b>Бюлетень № 8 про ставлення до дисципліни "Стратегії обслуговування
телекомунікаційних систем"</b><Br>
  <label> <input type="radio" name="bul8" value="1"/> Вважаю, що ця дисципліна
дуже потрібна і вимагає збільшення обсягу викладання    </label> <Br>
  <label> <input type="radio" name="bul8" value="2"/> Вважаю, що ця дисципліна
потрібна і обсяг викладання є достатнім                  </label> <Br>
  <label> <input type="radio" name="bul8" value="3"/> Вважаю, що ця дисципліна
потрібна, але обсяг викладання треба зменшити            </label> <Br>
  <label> <input type="radio" name="bul8" value="4"/> Вважаю, що ця дисципліна
для мене не дуже потрібна, але є потреба в її викладанні</label> <Br>
  <label> <input type="radio" name="bul8" value="5"/> Вважаю, що вивчення цієї
дисципліни мені непотрібно                               </label> <Br>
  <label> <input type="radio" name="bul8" value="0" checked/> Відмовляюсь від
вибору, бо не вивчав </label> <Br>
  </p>

  <p><b>Бюлетень № 9 про ставлення до дисципліни "Експлуатація
телекомунікаційних систем"</b><Br>
  <label> <input type="radio" name="bul9" value="1"/> Вважаю, що ця дисципліна
дуже потрібна і вимагає збільшення обсягу викладання    </label> <Br>
  <label> <input type="radio" name="bul9" value="2"/> Вважаю, що ця дисципліна
потрібна і обсяг викладання є достатнім                  </label> <Br>
  <label> <input type="radio" name="bul9" value="3"/> Вважаю, що ця дисципліна
потрібна, але обсяг викладання треба зменшити            </label> <Br>
  <label> <input type="radio" name="bul9" value="4"/> Вважаю, що ця дисципліна
для мене не дуже потрібна, але є потреба в її викладанні</label> <Br>
  <label> <input type="radio" name="bul9" value="5"/> Вважаю, що вивчення цієї
дисципліни мені непотрібно                               </label> <Br>
  <label> <input type="radio" name="bul9" value="0" checked/> Відмовляюсь від
вибору, бо не вивчав </label> <Br>
  </p>
</div>
```

```
<br>
 <table border="1"   bgcolor="#e0e0e0">
   <caption> </caption>
      <tr> <th>Вам відправлено код на E-mail (або SMS)</th> </tr>
      <tr> <th>Введіть цей код, завершивши свій вибір</th> </tr>
   <tr><td>Код:   <input type="text" size="5" pattern="[0-9]{6}"
id="KOD"></td></tr>
  <tr><td align="center">    <input type="button" value="Код введено"
onClick=TAB2()> </td></tr>
  </table>
</body>
</html>
```

ДОДАТОК Б.
АКТ ВПРОВАДЖЕННЯ РЕЗУЛЬТАТІВ РОБОТИ

З А Т В Е Р Д Ж У Ю

В.о. директора ДП Державний науково-дослідний інститут автоматизованих систем в будівництві (ДНДІАСБ)

_____ С.В. Басько

«19» квітня 2018 р.

АКТ ВПРОВАДЖЕННЯ

результатів дисертаційної роботи здобувача наукового ступеня кандидата технічних наук Пригари Михайла Петровича у комп'ютерній мережі ДНДІАСБ

Я, що нижче підписався, *радник директора ДП ДНДІАСБ Слободян Я.О.,* склав цей акт про те, що результати наукових досліджень *за темою дисертаційної роботи на здобуття наукового ступеня кандидата технічних наук Пригари Михайла Петровича «Захищена система технічної підтримки процесів дистанційного волевиявлення»* використовуються в комп'ютерній мережі ДНДІАСБ.

Найменування впровадженого результату	**Форма впровадження і досягнутий фактичний ефект**
Система дистанційного опитування студентів щодо їх ставлення до потреби набуття знань з навчальних дисциплін та до якості викладання	*Діюче програмне забезпечення системи дистанційного опитування студентів, яке розроблено на основі результатів даної дисертаційної роботи, встановлено на сервері ДНДІАСБ за електронною адресою в мережі Інтернет http://91.198.50.7:9000/VYBIR.html* *Завдяки впровадженню даної системи проведено опитування студентів технічних ВУЗів м. Києва (КНУБА, НАУ, НТУУ КПІ) і досягнута впевненість у збереженні таємниці голосів та неможливості підробки результатів волевиявлення. Виявлено що значення часу обслуговування запитів сервером не більше 6 с.*
Генератор дійсно (а не псевдо) випадкових послідовностей бітів на будь-яких типових засобах доступу до Інтернету без використання додаткових апаратних або програмних засобів	*Діюче програмне забезпечення для генерації дійсно випадкових бітових послідовностей, яке розроблено на основі результатів даної дисертаційної роботи, встановлено на сервері ДНДІАСБ за електронною адресою в мережі Інтернет http://91.198.50.7:11111/expco.htm* *Завдяки впровадженню цього програмного забезпечення досягнута можливість кожному користувачу Інтернету впевнитись у тому, що його засіб доступу до мережі здатен генерувати дійсно випадкові бітові послідовності, а також виміряти статистичні характеристики цих послідовностей.*

Радник директора ДП
ДНДІАСБ д.т.н., професор _____ Я.О. Слободян