Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
(повне найменування вищого навчального закладу)
Факультет комп'ютерно-інформаційних систем і програмної інженерії
(назва факультету)
Кафедра кібербезпеки
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(освітній рівень)

на тему: *«Дослідження сучасних методів блочного симетричного шифрування інформації в автоматизованих банківських системах»*

Виконав: студент (ка)
Спеціальності:
125 «Кібербезпека»
(шифр і назва напряму підготовки, спеціальності)
Коробка Н.П.
підпис (прізвище та ініціали)

Керівник *Стадник М.А.*
підпис (прізвище та ініціали)

Нормоконтроль
підпис (прізвище та ініціали)

Завідувач кафедри *Загородна Н.В.*
підпис (прізвище та ініціали)

Рецензент
підпис (прізвище та ініціали)

м. Тернопіль – 2021

АНОТАЦІЯ

Дослідження сучасних методів блочного симетричного шифрування інформації в автоматизованих банківських системах// Дипломна робота ОР «Бакалавр» //Коробка Назар Петрович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра кібербезпеки, група СБсз-41 // Тернопіль, 2021 // С. , рис. – , табл. – , кресл. – , додат. – .

Ключові слова: КОНФІДЕНЦІЙНІСТЬ, ВНУТРИШНЬОПЛАТІЖНА БАНКІВСЬКА СИСТЕМА, ЗАГРОЗИ, АТАКИ, КРИПТОПРИМІТИВИ, МЕТОДИ БЛОЧНОГО СИМЕТРИЧНОГО ШИФРУВАННЯ, МАТЕМАТИЧНА МОДЕЛЬ

Кваліфікаційна робота присвячена аналізу методів блочного симетричного шифрування в автоматизованих банківських системах і підвищення криптографічної стійкості алгоритмів блочного симетричного перетворення інформації на підґрунті динамічно керованих криптографічних примітивів

Розроблено динамічно керованих блоків нелінійних замін, дослідження їх криптографічних властивостей виконані з використанням математичного апарату булевих функцій, методів кореляційного і спектрального аналізу. Розроблено алгоритму блочного симетричного криптографічного перетворення інформації з динамічно керованими примітивами і дослідження криптографічної стійкості проведені з використанням методів теорії захисту інформації. Досліджено криптостійкість алгоритму проведено з використанням теорії імовірності та математичної статистики, теорії захисту інформації.

ANNOTATION

Study of modern methods of information block symmetrical encryption in automated bank systems // Thesis of educational level "Bachelor" // Korobka Nazar Petrovych // Ternopil National Technical University named after Ivan Pulyuy, Faculty of Computer Information Systems and software engineering, Department of Cybersecurity, СБсз-41 group // Ternopil, 2021 // P. , fig. -, table. - , chair. - , added. -.

Keywords: CONFIDENTIALITY, THE INNER-PAYMENT BANKING SYSTEM, THREATS, ATTACKS, KRIPTOPRIMITIVI, METHODS of SECTIONAL SYMMETRIC ENCIPHERING, THE MATHEMATICAL MODEL.

The qualification thesis is devoted to analysis of methods of the block symmetric enciphering in computer-aided bank systems and an increase of cryptographic firmness of algorithms of sectional symmetric transformation of information on the basis of the dynamically guided cryptographic primitives.

Development of the dynamically guided blocks of nonlinear replacements, research of their cryptographic properties executed with the use of mathematical vehicle of boolean functions, methods of cross-correlation and spectral analysis is conducted. Development of algorithm of block symmetric cryptographic transformation of information with the dynamically guided primitives and research of cryptographic firmness is conducted with the use of the methods of the theory of information security. Criptofirmness of algorithm is researched with the use of probability theory and mathematical statistics, the theory of information security.

ЗМІСТ

ВСТУП

В розвитку ринкових відносин чільну роль відіграють комерційні банки, що акумулюють величезні фінансові потоки. Інформаційні банківські системи стають однієї з найбільш уразливих сторін сучасного банку, що притягають до себе зловмисників, як із числа персоналу банку, так і з боку [9 –12].

Порушення роботи банківських систем приводить до втрати не тільки конфіденційної інформації банку, але та до економічних збитків, як банку, так і його клієнтів, що створює загальнонаціональну проблему.

Відповідно до основних положень концепції Національної безпеки України витік конфіденційної інформації вважається однією з найнебезпечніших загроз безпеки в інформаційній сфері [14]. Питання захисту інформації в банківських системах є складовою питання Національної безпеки України.

Основним і найбільш ефективним механізмом криптографічного захисту банківських транзакцій є методи блочного симетричного криптографічного перетворення.

В якості найбільш ефективного симетричного блочного криптографічного алгоритму за показниками швидкодії та стійкості, а також за відкритістю та прозорістю всіх механізмів-перетворень, визнаний переможець відкритого конкурсу Advanced Encryption Standard (AES) і відзначений на європейському конкурсі криптоалгоритмів New European Schemes for Signatures, Integrity, and Encryption (NESSIE) алгоритм Rijndael. Шифр реалізує абсолютно нетрадиційну криптографічну парадигму, повністю відмовившись від мережі Фейстеля.

В ході досліджень [53, 65, 66] встановлено, що для шифру AES доцільно застосовувати методи атак, які б використовували алгебраїчну простоту його SPN-структури, – алгебраїчні атаки. Дані атаки при можливому комбінуванні з іншими відомими підходами можна розглядати як реальну загрозу надійності даного шифру.

Перспективним напрямком в розвитку симетричних криптоалгоритмів є вживання динамічно керованих раундовими ключами криптопримітивів.

Розробка і вивчення методів блочного симетричного криптографічного перетворення інформації з динамічно керованими примітивами є перспективним напрямом досліджень, тобто тема є актуальною.

Метою роботи є аналіз методів блочного симетричного шифрування в автоматизованих банківських системах і підвищення криптографічної стійкості алгоритмів блочного симетричного перетворення інформації на підґрунті динамічно керованих криптографічних примітивів.

Відповідно до мети роботи необхідно вирішити наукове завдання, що полягає в розробці методів формування динамічно керованих криптографічних примітивів для підвищення стійкості блочних симетричних шифрів. Для досягнення поставленої мети необхідно вирішити наступні окремі завдання:

– проаналізувати захист інформації в сучасних умовах інформаційного суспільства;

– проаналізувати і дослідити алгоритм блочного симетричного криптографічного перетворення інформації з динамічно керованими примітивами;

– розробити практичні рекомендації з реалізації криптоалгоритма ADE і експериментальні дослідження його продуктивності.

Об’єктом дослідження є процес підвищення криптографічної стійкості алгоритмів блочного симетричного перетворення інформації на основі динамічно керованих примітивів.

Предметом дослідження є сучасні методи блочного симетричного шифрування інформації в автоматизованих банківських системах.

РОЗДІЛ 1 ЗАХИСТ ІНФОРМАЦІЇ В СУЧАСНИХ УМОВАХ ІНФОРМАЦІЙНОГО СУЛЬСПІЛЬСТВА

1.1. Концептуальні питання створення, функціонування, розвитку та використання Національної системи конфіденційного зв'язку

Сучасне суспільство характеризується високою мірою інформатизації, і це робить його залежним від безпеки інформаційних технологій. Інформаційно-телекомунікаційні системи (ІТС) забезпечують надійність функціонування величезної кількості різного типу інформаційних систем. Більшість таких систем несуть в собі інформацію, що має конфіденційний характер. Таким чином, розв'язання задачі автоматизації процесів обробки даних та підвищення ефективності роботи підприємств оголило питання інформаційної безпеки [22–35, 37].

Рисунок 1.1 - Основні цілі забезпечення ІБ

Довгострокова стратегічна ціль державної інформаційної політики України – це формування відкритого інформаційного суспільства на основі розвитку єдиного інформаційного простору цілісної держави, його інтеграція у світовий інформаційний простір з урахуванням національних особливостей і інтересів при забезпеченні інформаційної безпеки на внутрішньодержавному та міжнародному рівнях.

Основні цілі забезпечення ІБ визначаються пріоритетами національної безпеки, що відповідають інтересам суспільного розвитку, та наведені на рис.1.1.

До основних системо-утворюючих складових інформаційної безпеки належать (рис. 1.2):

– захист інформаційного простору;
– захист інформаційних ресурсів;
– захист інформації з обмеженим доступом.

Рисунок 1.2 - Складові інформаційної безпеки України

Основними функціями системи забезпечення інформаційної діяльності є: створення та забезпечення діяльності державних органів – елементів системи управління діяльністю системи, міжнародне співробітництво в сфері інформаційної безпеки.

В Україні створена та функціонує досить непогано розроблена система забезпечення інформаційної безпеки. Функції та права та обов'язки різних державних органів описані в нормативно-правових актах різного рівня – Конституції України, законах України, указах Президента України, постановах Кабінету Міністрів України, інших, у тому числі відомчих, нормативних актах. Водночас розподіл функцій між окремими суб’єктами системи та схема їх взаємодії потребують вдосконалення.

Як свідчить досвід, ефективно вирішувати завдання щодо захисту інформації, що циркулює в інформаційних та комунікаційних системах, а також гарантувати достатній захист ІТС державних органів від злочинних посягань можна лише , створюючи в їх складі комплексні системи захисту інформації (КСЗІ), що об'єднують різні аспекти: правові, організаційні, криптографічні, інженерні заходи, а також технічні і програмні засоби захисту.

Нині законодавством України визначені основні загрози національній безпеці України[1–4] (рис. 1.3).

Проведений аналіз показав, що ефективна діяльність у сфері забезпечення інформаційної безпеки повинна проходити шляхом нарощування потужностей тих структур, на які покладено реалізацію державної політики за основними технологічними її складовими.

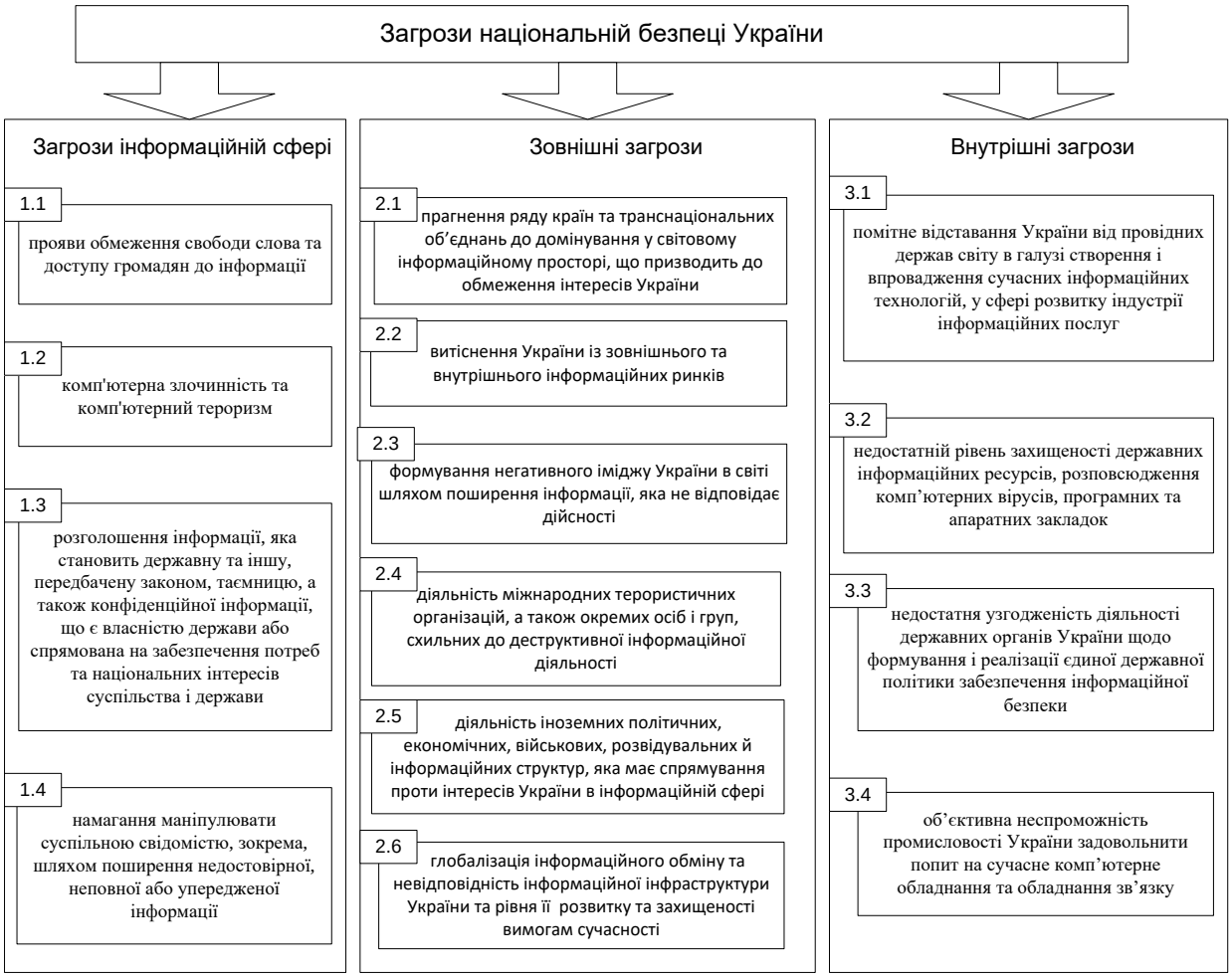

Рисунок 1.3 - Загрози національній безпеці, визначені законодавством України

Як бачимо, кількість загроз є достатньо великою, і потребує створення комплексних систем захисту інформації.

1.2 Побудова моделей атак на внутрішньо-платіжну банківську систему

1.2.1 Аналіз сучасних загроз інформації банківської системи

У розвитку ринкових відносин важливу роль відіграють комерційні банки, що акумулюють величезні фінансові потоки, які залучають величезний інтерес кримінальних структур, спецслужб і конкурентів. Інформаційні системи стають однією з найбільш уразливих сторін сучасного банку, притягаючи до себе зловмисників як із числа персоналу банку, так і з боку.

Необхідно враховувати, що втрата інформації в особистому комп'ютері в результаті проникнення вірусу відчутна для його власника, але порушення роботи банківських систем зачіпає інтереси суспільства і створює загальнонаціональну проблему.

Внутрішньоплатіжна банківська система (ВПБС) є сукупністю правил, організаційних заходів, програмно-технічних засобів, засобів захисту, що використовуються банком для виконання внутрішнього банківського переведення грошей, а також для взаємодії з іншими банківськими платіжними системами, для забезпечення виконання міжбанківського переказу коштів філіями банку [39] (рис. 1.4).

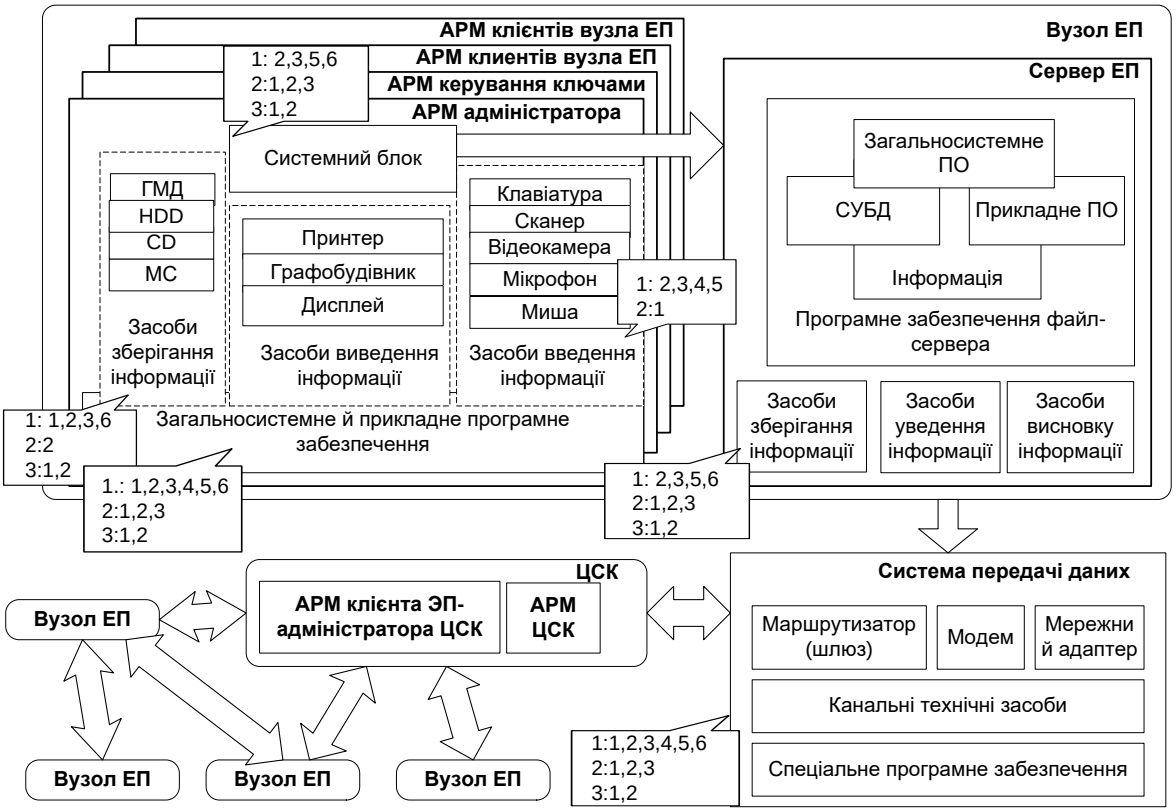

Рисунок 1.4 - Загальна структура підсистеми захисту інформації у ВПБС

Для забезпечення захисту банківської інформації у ВПБС на різних рівнях використовуються криптографічні механізми. Однак, бурхливе зростання обчислювальної техніки, створення систем і технологій кібертероризму приводить до появи нових загроз (активних і пасивних атак) і злому підсистеми захисту ВПБС.

Всі джерела загроз безпеки інформації можна розділити на три основні групи: навмисні загрози безпеки у ВПБС, стихійні лиха та збої [21, 47, 68, 72–74].

На рис. 1.5 наведена загальна класифікація загроз інформаційних ресурсів у ВПБС.

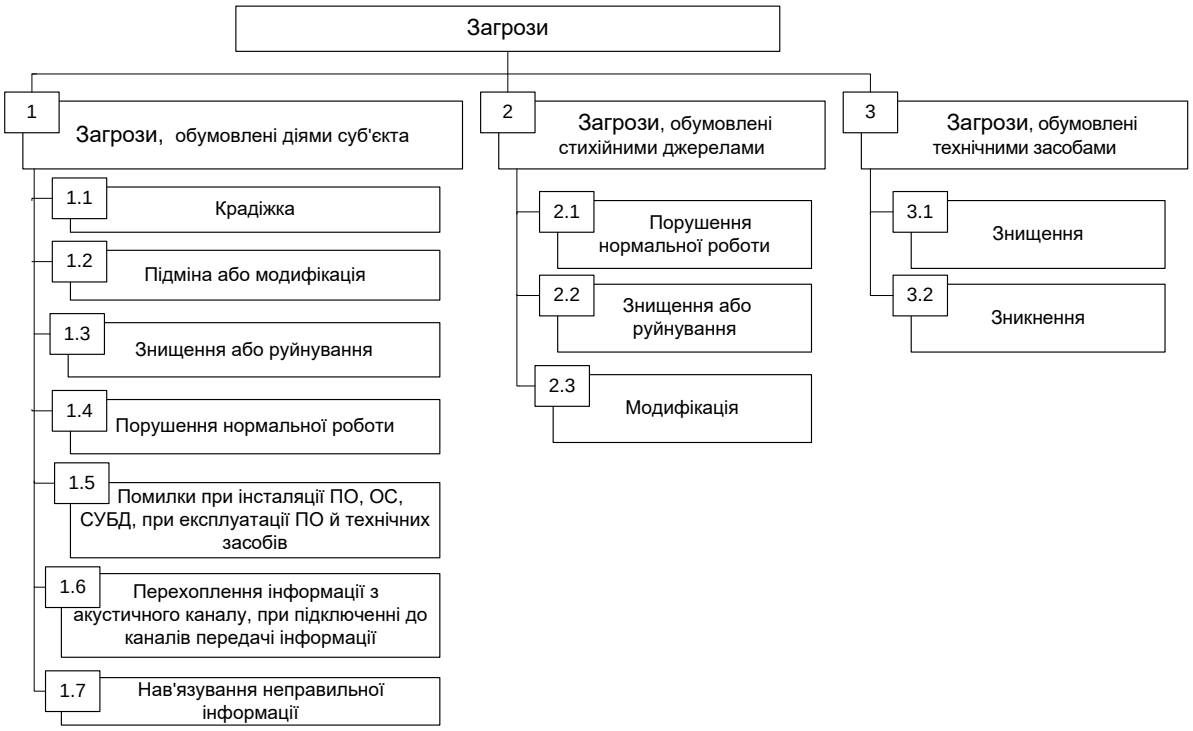

Рисунок 1.5 - Загальна класифікація загроз інформаційних ресурсів у ВПБС

Одним з найбільш уразливих місць у системі електронних платежів є пересилання платіжних і інших повідомлень між банками, між банком і банкоматом, між банком і клієнтом.

Для захисту платіжних повідомлень використовується система захищеної електронної пошти (СЗЕП), що призначена для обміну електронними повідомленнями у форматі SMF-70 через мережу передачі даних довільного типу відповідно до критеріїв НД ТЗІ 2.5-004-99 [37–41]. Загальна структура підсистеми захисту інформації у ВПБС [34] і можливих загроз на окремі її складові представлена на рис. 1.4.

Таким чином, аналіз загроз у конкретних умовах становить основу для планування та здійснення заходів, спрямованих на забезпечення безпеки ВПБС.

1.2.2 Побудова моделі реалізації загроз безпеки ВПБС

Моделювання процесу атак ВПБС доцільно здійснювати на основі побудови логічного ланцюжка: “загрози – джерело загрози – метод реалізації – вразливість – наслідки”. На рис. 1.6 представлена структурна схема моделі реалізації загроз інформаційних ресурсів у ВПБС.

Рисунок 1.6 - Структурна схема моделі реалізації загроз інформаційних ресурсів у ВПБС

Для опису моделі реалізації загроз інформаційних ресурсів у ВПБС (математичних моделей активної та пасивних атак) зафіксуємо кінцеву безліч суб’єктів, взаємодіючих з інформаційною системою (S) [69].

Нехай параметр N – кількість уразливих до атаки комп’ютерів (ПК), а параметр D містить початкове значення середньої кількості атакованих комп’ютерів за обрану одиницю часу. Вважаємо, що D є константою протягом усіх подальших обчислень, незважаючи на відмінності в потужностях і типах обчислювального устаткування, що атакується, і пропускної здатності каналів зв’язку. Крім того, обчислення робляться враховуючи, що той самий комп’ютер не може бути атакований двічі. Нехай $a(t)$ – пропорція вразливих ПК, які були успішно атаковані під час t, тоді $N \cdot a(t)$ – загальна кількість успішно атакованих комп’ютерів. Оскільки частина комп’ютерів вже була успішно атакована (їх частка становить $a(t)$), кожним новим захопленим комп’ютером

буде зроблено не більш $D(1-a(t))$ нових успішних атак. Таким чином, кількість захоплених комп'ютерів за період часу $d(t)$ дорівнює (зафіксувавши $a(t)$):

$$n = aN \cdot D(1-a)dt$$

Враховуючи, що N – константа, тоді $n = d(Na) = Nda$. Тоді вірно наступне рівняння:

$$Nda = aN \cdot D(1-a)dt,$$

веде до диференціального рівняння виду:

$$\frac{da}{dt} = Da(1-a),$$

і має наступне рішення:

$$a = \frac{e^{D(t-T)}}{1+e^{D(t-T)}},$$

де T є часовим параметром, що характеризує найбільший ріст атак [66].

На підставі проведених обчислень, розробимо загальну структуру підсистеми захисту інформаційних ресурсів у ВПБС.

1.2.3 Побудова загальної структури підсистеми безпеки інформаційної безпеки ВПБС

Для побудови загальної структури підсистеми безпеки інформаційної безпеки ВПБС і моделей атак обраний функціональний тип математичних моделей, називаний моделями "чорної скриньки ". Дані моделі побудовані відповідно до методологій IDEF0 і DFD з використанням CASE-засобу BPWin.

Для забезпечення захисту інформації в системі електронного документообігу використовується криптографічний метод електронного

цифрового підпису згідно зі стандартами, ратифікованими в Україні: ДСТУ-4145, ГОСТ 28147-89, ГОСТ 34310-95, ГОСТ 34311-95 [6–9]. Необхідно відзначити, що цифровий підпис дозволяє не тільки автентифікувати автора електронного документа, але та підтвердити цілісність останнього.

На рис.1.7 – 1.8 наведена загальна структура підсистеми захисту інформаційних ресурсів (ІР) ВПБС.

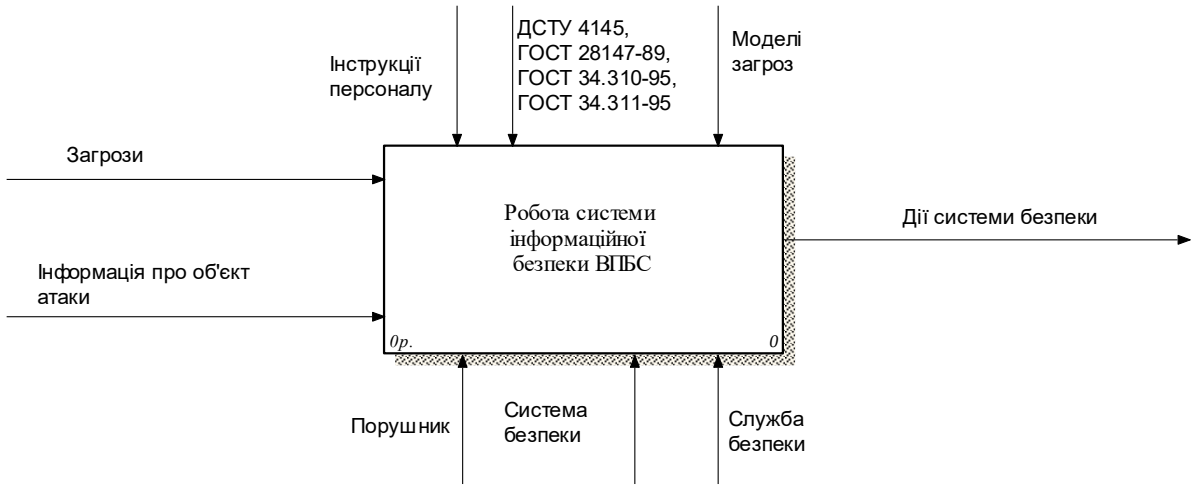

Рисунок 1.7 - Загальна структура підсистеми захисту ІР у ВПБС (контекстна діаграма)

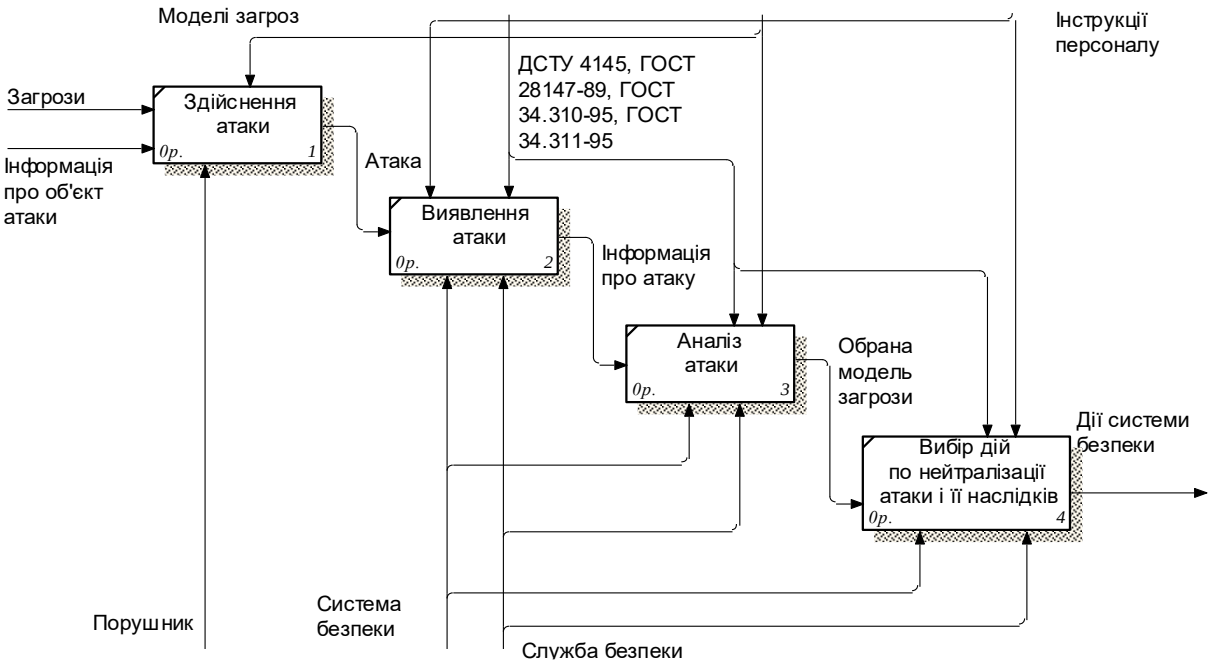

Рисунок 1.8 - Декомпозиція загальної структури підсистеми захисту ІР

На підставі проведеного аналізу загроз і побудованої загальної моделі підсистеми захисту інформаційних ресурсів ВПБС, розглянемо моделі активних і пасивних атак, які можуть бути реалізовані в банківській системі. Загальним для опису даних математичних моделей є процес формування криптограми.

Для цього зафіксуємо кінцеву безліч $I = \{I_1, I_2, ..., I_m\}$ пакетів, переданих у банківській транзакції, причому кожному пакету відповідає ймовірність $P^*(I_j)$. Розподіл ймовірностей випадкового процесу задається сукупним розподілом ймовірностей випадкових величин, тобто безліччю ймовірностей $P^*_{\hat{i}} = \{P^*(I_1), P^*(I_2), ..., P^*(I_m)\}$ [41]. Джерело ключів породжує потік ключів з безлічі K і/або K^*. Кожному ключу Ki ∈ K = {K_1, K_2, …, Kk} відповідає деяка ймовірність $P^*(K_i)$, а кожному $Ki^* \in K^* = \{K_1^*, K_2^*, \ldots, Kk^*\}$ відповідає ймовірність $P^*(K^*_i)$. Випадковий процес вироблення ключів задається безліччю ймовірностей:

$$P^*_K = \{P^*(K_1), P^*(K_2), ..., P^*(K_k)\}$$
$$P^*_{K^*} = \{P^*(K_1^*), P^*(K_2^*), ..., P^*(K_k^*)\}.$$

Вибір ключа Ki визначає конкретне відображення φ_i з безлічі відображень φ. За допомогою відображення φ_i, відповідного до обраного ключа Ki, за пакетом, що надходить i_j , формується криптограма:

$$E_l = \varphi_i(Ki, Ij).$$

Криптограма E_l передається в вузол приймання по деякому каналу. Наступні дії порушника визначаються метою проведення атаки та відповідно її типом.

Оцінка ступеня ефективності атаки може бути здійснена за рахунок проведення аналізу даних, якими володіє зловмисник, аналізу його можливостей

та інших параметрів пасивних атак. Основним методом оцінки можливостей зловмисника при атаці є створення моделей атак. Розглянемо основні моделі атак на ВПБС.

1.2.4 Побудова математичної моделі пасивних атак на ВПБС

Пасивні загрози випливають із прослуховування (несанкціонованого зчитування інформації) і не пов'язані з якою-небудь зміною інформації [60]. Суть атаки полягає в тому, що порушник, визначивши факт виконання криптографічного протоколу, перехоплює всі дані, які були передані по каналу зв'язку. Тобто при передачі криптограми E_l у вузлі приймання по деякому каналу порушник виконує моніторинг мережі. Криптоаналіз протоколу залежить від типу протоколу, кількості та типу ключів, математичного апарату, який використовуються в протоколі, і інших характеристик протоколу.

На приймальній стороні за допомогою зворотного відображення φ_{i-1} (заданого ключем K^*_i) із криптограми E_l відновлюється первісна інформація

$$I_j = \varphi_i^{-1}(K^*_i, E_l).$$

Узагальнена модель пасивних атак представлена на рис. 1.9 – 1.10 [37].

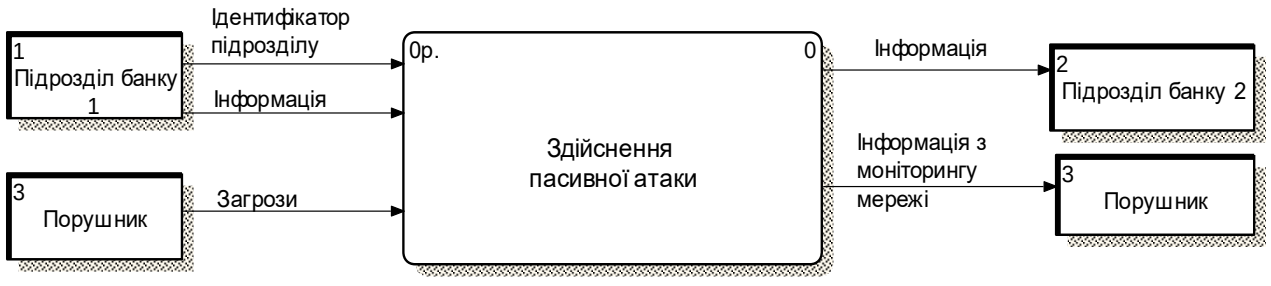

Рисунок 1.9 - Модель пасивних атак на інформаційні ресурси у ВПБС

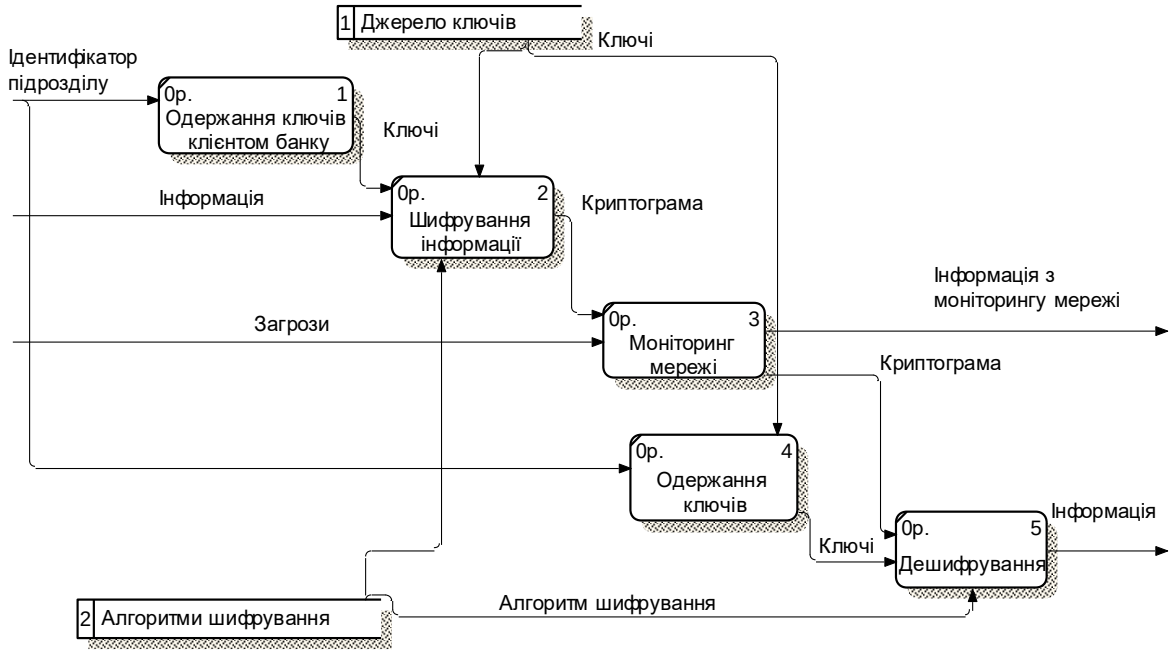

Рисунок 1.10 - Декомпозиція моделі пасивних атак

Таким чином, криптоаналіз є рішенням математичного завдання з метою визначення самого повідомлення або деяких особистих ключів суб'єктів-учасників протоколу. Більш небезпечними з погляду економічного збитку для ВПБС є активні атаки. Розглянемо основні типи активних атак.

1.2.5 Побудова моделі активних атак на ВПБС із блокуванням передачі інформації

Суть атаки із блокуванням передачі інформації полягає в тому, що порушник, визначивши факт виконання криптографічного протоколу, блокує передачу інформації, у результаті чого криптограма не досягає прийомної сторони.

Узагальнена модель активних атак із блокуванням передачі інформації представлена на рис. 1.11 – 1.12. Одержувачем інформації в даній моделі є порушник.

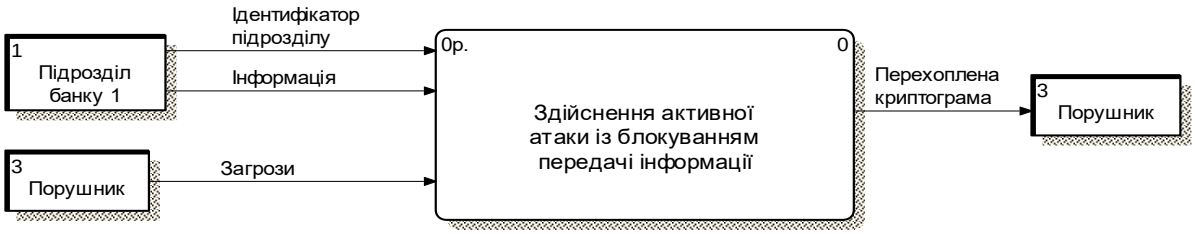

Рисунок 1.11 - Модель активних атак із блокуванням передачі інформації

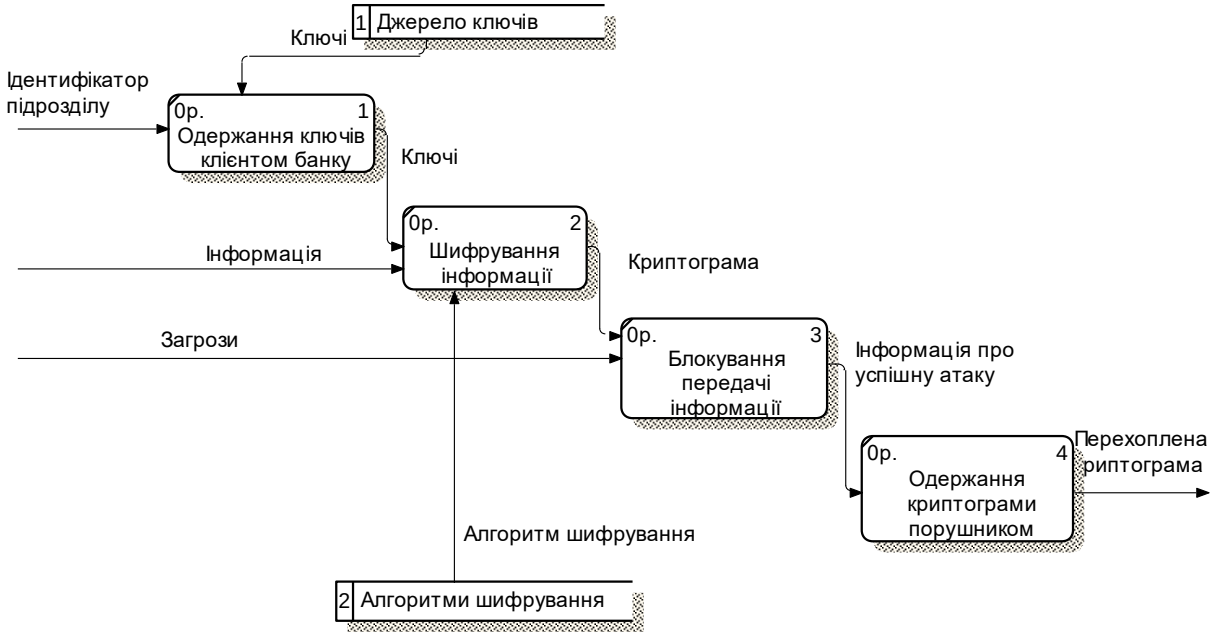

Рисунок 1.12 - Декомпозиція моделі активних атак із блокуванням передачі інформації

Таким чином, при реалізації даної атаки необхідні дані не досягають пункту призначення, або досягають занадто пізно, що приводить до втрати конфіденційної банківської інформації.

1.2.6 Побудова моделі активних атак на ВПБС із внесенням перешкод

Суть атаки із внесенням перешкод полягає в тому, що порушник, визначивши факт виконання криптографічного протоколу, вносить деяку помилку e та передає в вузол приймання криптограму (E_l+e). На прийомному кінці за допомогою зворотного відображення φ_{i-1} (заданого ключем K^*) із криптограми (E_l+e) відновлюється недостовірна інформація

$$I_j^e = \varphi_i^{-1}(K^*_i, E_l+e),$$

тобто підрозділ банку одержує повідомлення відмінне від вихідного $I_j^{\,e} \neq I_j$.

Узагальнена модель активних атак із внесенням перешкод представлена на рис. 1.13 – 1.14.

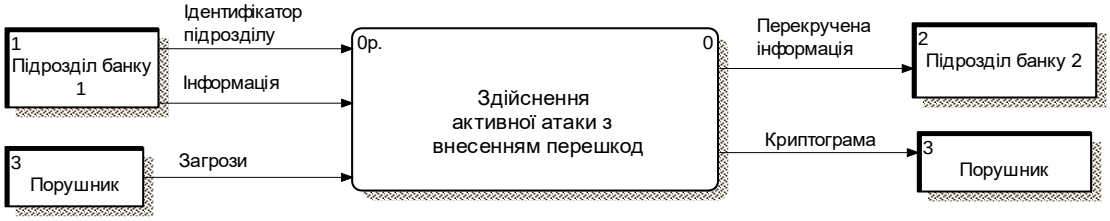

Рисунок 1.13 - Модель активних атак із внесення перешкод

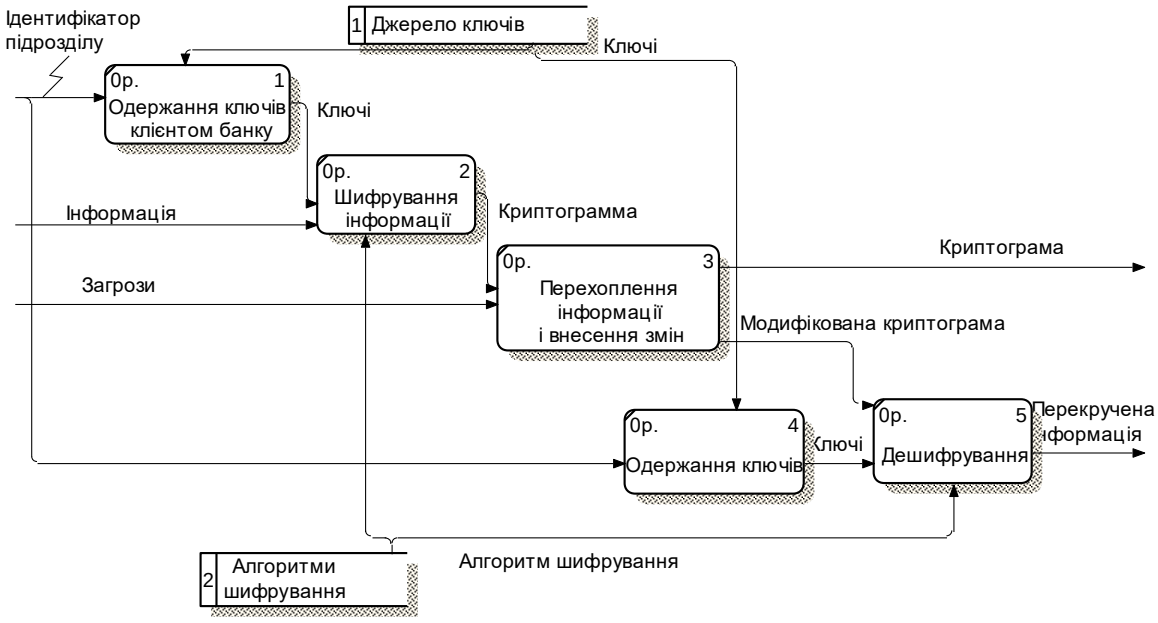

Рисунок 1.14 - Декомпозиція моделі активних атак із внесення перешкод

1.2.7 Побудова моделі активних атак “маскарад” на ВПБС

Суть атаки “маскарад” полягає в тому, що користувач (або процес, підсистема і т.д.) передає інформацію від імені іншого користувача. Способи заміни ідентифікатора можуть бути різні, зазвичай вони визначаються

помилками та особливостями мережних протоколів. Проте, на прийомному вузлі таке повідомлення буде сприйнято як коректне, що може привести до серйозних порушень роботи ВПБС.

Розглянемо процес виконання атаки даного типу. Порушник, визначивши факт виконання криптографічного протоколу, перехоплює криптограму E_l. З її допомогою він може спробувати обчислити апостеріорні ймовірності різних можливих повідомлень:

$$P^*_{\hat{i}|E_l} = \{P^*(I_1|E_l), P^*(I_2|E_l), ..., P^*(I_m|E_l)\};$$

і різних можливих ключів:

$$P^*_{K|E_l} = \{P^*(K_1|E_l), P^*(K_2|E_l), ..., P^*(K_k|E_l)\},$$

які могли бути використані при формуванні криптограми E_l.

Множини апостеріорних ймовірностей утворюють апостеріорні знання порушника про ключі $K = \{K_1, K_2, \ldots, Kk\}$ і про інформацію $I = \{I_1, I_2, \ldots, Im\}$ після перехоплення криптограми E_l. Фактично, безлічі $P^*_{K|E_l}$ і $P^*_{M|E_l}$ являє собою безліч припущень, яким приписані відповідні ймовірності.

Потім, одержавши необхідну інформацію, порушник формує криптограму з недостовірною інформацією

$$E_l{}^e = \varphi_i(K_i, I_j^e)$$

і передає її у вузлі приймання.

На прийомному кінці за допомогою зворотного відображення φ_{i-1} (заданого ключем K^*_i) із криптограми E_l^e відновлюється недостовірна інформація, передана порушником.

$$I_j{}^e = \varphi_i^{-1}(K_i^*, E_l^e),\ I_j{}^e \neq I_j$$

Атака такого типу, як правило, пов’язана зі спробами проникнення усередину периметра безпеки ВПБС і часто реалізується хакерами.

Узагальнена модель активних атак “маскарад” представлена на рис. 1.15 – 1.16.

Найнебезпечнішим в банківських системах електронних платежів є “маскарад”, де невірна ідентифікація клієнта може привести до втрати його конфіденційної інформації та активів.

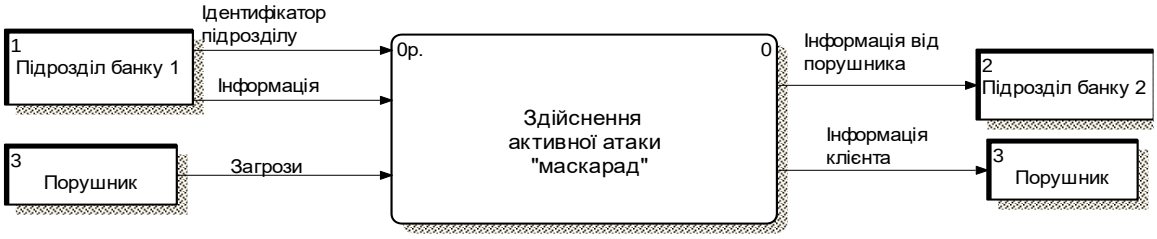

Рисунок 1.15 - Модель активних атак “маскарад”

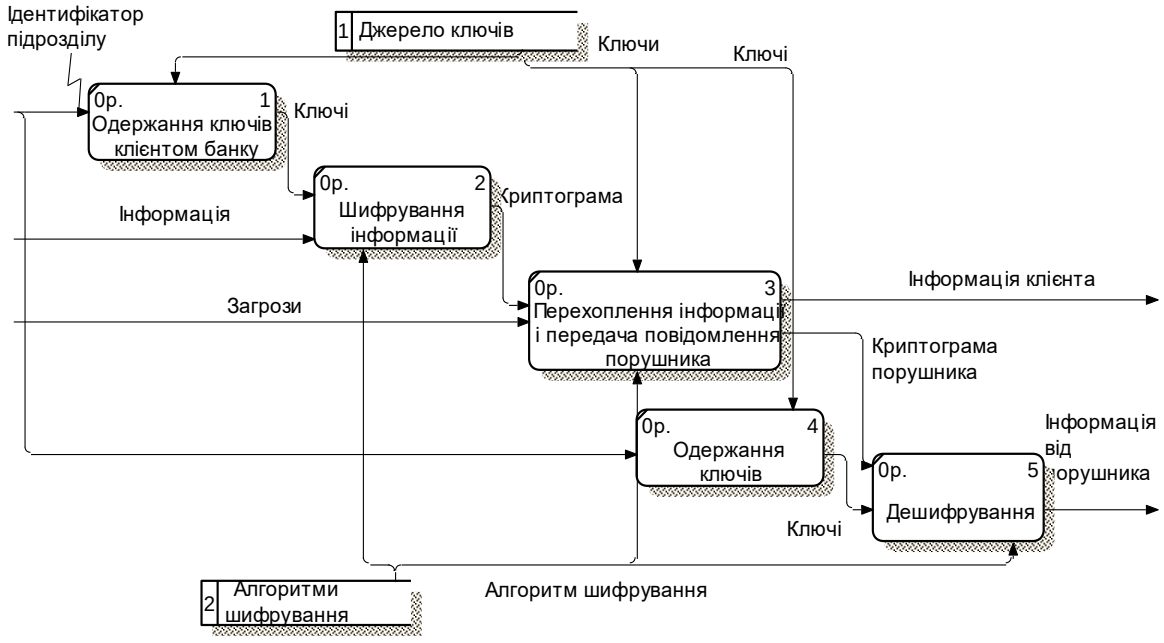

Рисунок 1.16 - Декомпозиція моделі активної атаки “маскарад”

Проведений аналіз показав, що при виконанні пасивної атаки порушник не впливає на протокол. Він намагається отримати інформацію про учасників

протоколів, збираючи повідомлення, які передані різними сторонами, і намагається криптоаналізувати їх.

Таким чином, аналіз розглянутих атак показує, що будь-яка реалізована активна атака приводить до втрати конфіденційної інформації (банку або його клієнтів) і завдає економічної шкоди як активам банку, так і активам його клієнтів. Наслідком втрати конфіденційної інформації є крах банківської системи, що є загальнонаціональною проблемою.

1.3 Аналіз сучасних послуг та механізмів захисту інформації та визначення основних напрямків захисту ВПБС

Для запобігання загроз на інформаційні ресурси ВПБС розглянемо основні напрямки захисту банківської інформації, представлені на рис. 1.17.

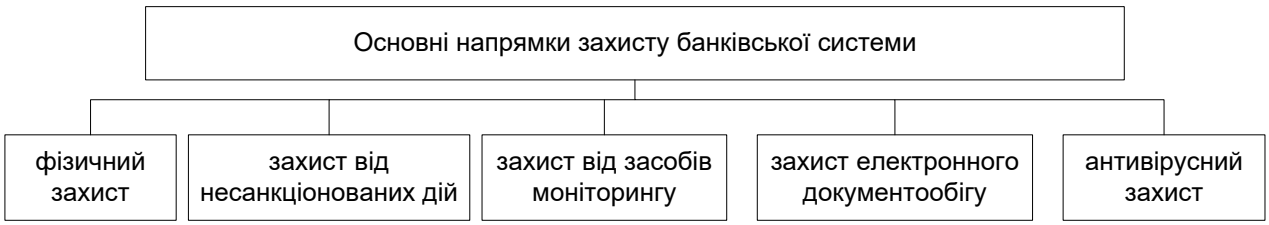

Рисунок 1.17 - Основні напрямки захисту інформації

Аналіз засобів захисту показав, що більшість із них реалізовані за допомогою апаратних, програмно-апаратних, програмних систем і засобів, на основі відповідних криптографічних алгоритмів [45, 46, 56, 62]. Перевагою апаратних засобів є простота їх реалізації, недоліком – неможливість удосконалювання та модернізації, можливість “обходу” зловмисником алгоритму захисту, висока вартість реалізації криптоалгоритма.

На рис. 1.18 наведений взаємозв’язок основних напрямків і засобів захисту інформаційних ресурсів у ВПБС.

Відповідно до міжнародних стандартів ISO 7498, ISO/IEC 10181 для забезпечення необхідних показників безпеки визначені послуги та відповідні механізми безпеки. Механізми безпеки є засобами, за допомогою яких

реалізується та застосовується відповідна послуга. Загальна класифікація механізмів безпеки представлена на рис. 1.19.

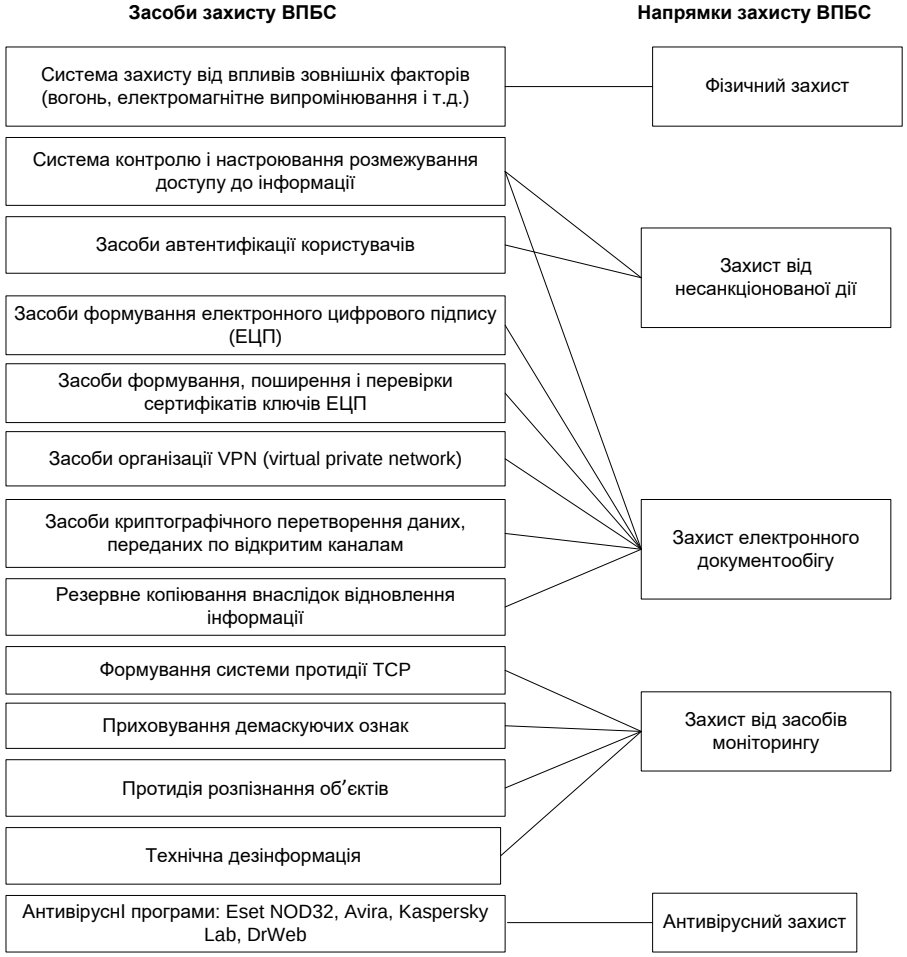

Рисунок 1.18 - Взаємозв'язок напрямків і засобів захисту ВПБС

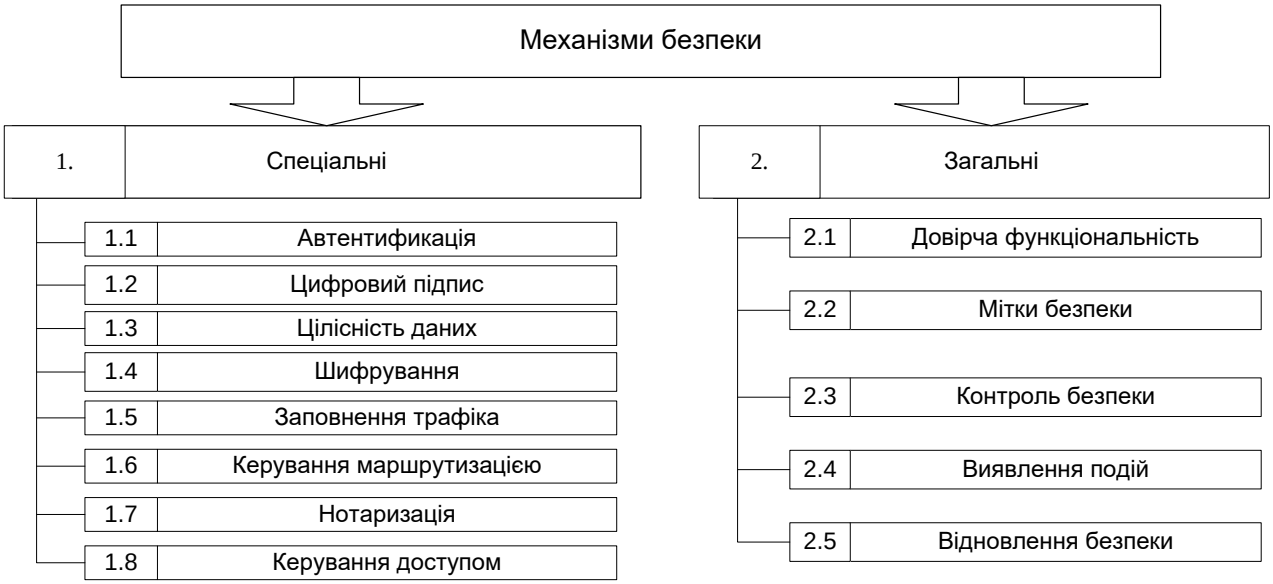

Рисунок 1.19 - Загальна класифікація механізмів безпеки

Взаємозв’язок послуг і механізмів безпеки представлений на рис. 1.20.

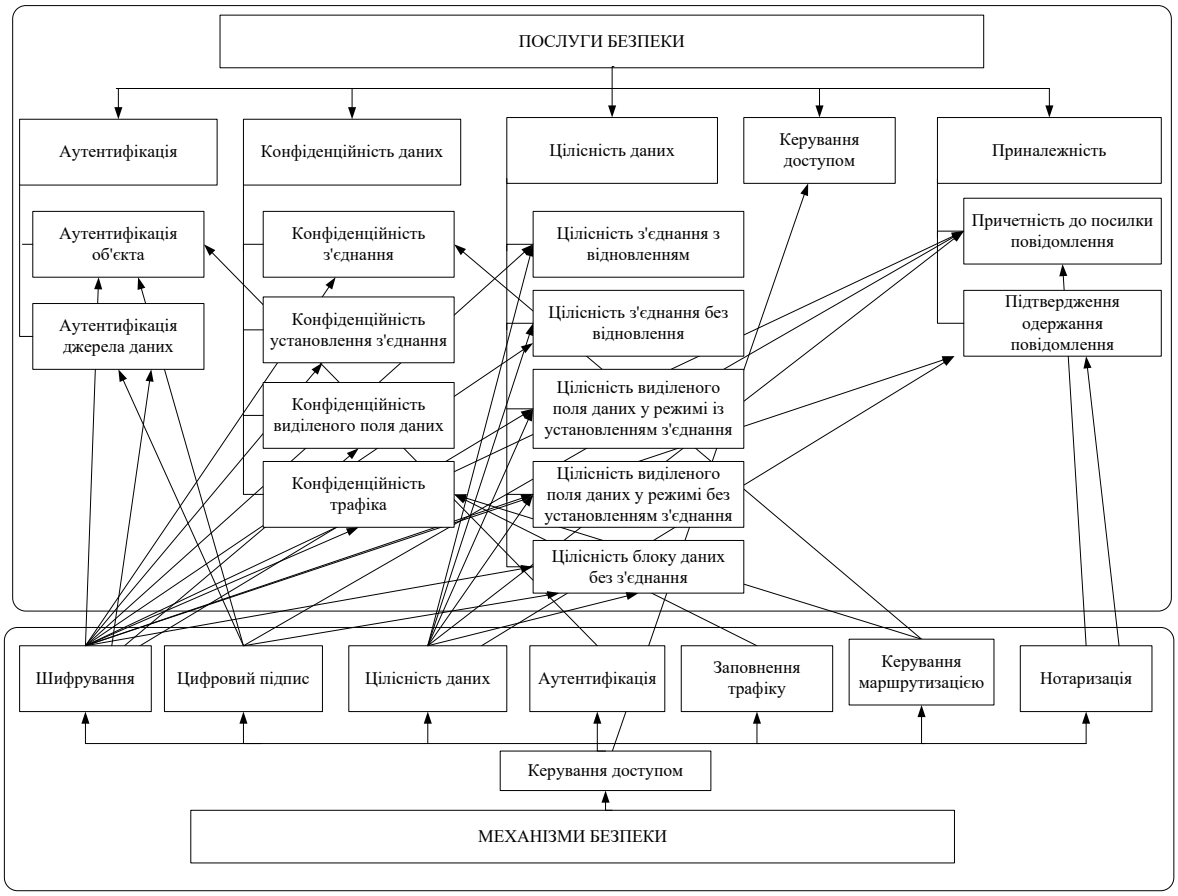

Рисунок 1.20- Взаємозв’язок послуг і механізмів безпеки

Основні механізми забезпечення цілісності та автентичності інформації у ВПБС на різних рівнях засновані на використанні стандартів блочно-симетричних шифрів [50, 71, 72]. Особливе місце серед механізмів забезпечення цілісності і автентичності займає цифровий підпис.

Важливою складовою частиною ВПБС є підсистема криптографічного захисту інформації, який реалізується відповідними протоколами та механізмами [39].

Таким чином, аналіз механізмів захисту показує, що передача інформації вимагає контролю безпеки на всіх рівнях. Основним і найбільш ефективним механізмом криптографічного захисту інформації є методи блочного симетричного шифрування. Блочно-симетричні криптографічні перетворення

дозволяють забезпечити високу стійкість до різних методів криптографічного аналізу.

В табл. 1.1 наведені характеристики деяких сучасних ВПБС, їх функції та рівні захисту.

Таблиця 1.1-Характеристики сучасних ВПБС

Системи електронних платежів банку	Функції системи	Рівні застосування механізмів захисту
1	2	3
ВПБС “ГРАНТ” – призначена для виконання платежів у національній валюті України між головним банком і його філіями, а також платежів у СЗЕП НБУ головним банком і його філіями.	Обробка платіжних документів. Захист інформації. Керування філіями-учасниками ВПС. Взаємодія з інформаційно-пошуковою системою (ІПС) НБУ, а також робота внутрішньої ІПС	Рівень захищеного операційного середовища; рівень СУБД; рівень прикладного програмного забезпечення; рівень засобів криптографічного захисту інформації
Enigma – забезпечує автоматизацію внутрішньобанківських і міжбанківських платежів у багато-філіальних банках України.	Обмін пакетами документів і технологічними файлами. між головним банком, філіями та системою електронних платежів НБУ. Накладення логічних і бухгалтерських обмежень на різні платіжні операції в СЗЕП НБУ, ВПБС банку.	Рівень засобів криптографічного захисту інформації
Profix/TELEBANK – призначена для керування ресурсами багато філіального банку.	Обробка вхідних і вихідних повідомлень. Проведення ручних тверджень повідомлень, що надходять на виконання від служб і філій. Надання послуг іншим банкам у якості клірингового банку.	Система має вбудовані механізми забезпечення безпеки електронних платежів, які побудовані за принципами і форматам СЗЕП. Використовується система шифрування на базі алгоритму DES і електронному підпису на базі алгоритму RSA.

Продовження таблиці 1.1

Внутрішньобанківська Платіжна Система — новий програмний продукт, розроблений фахівцями компанії R-Style Ukraine, призначений для керування фінансовими потоками в багато філіальному банку.	Прийняття від філії відправника внутрішньобанківського платежу і доставка його через розрахунковий центр у філіал-одержувач. Виконання всіх операцій по облікові руху коштів, передбачених обраною бухгалтерською моделлю, забезпечення коректності консолідованого балансу банку та гарантування цілісності даних і захисту інформації.	Рівень захищеного операційного середовища; рівень СУБД; рівень прикладного програмного забезпечення; рівень засобів криптографічного захисту інформації.

Останнім часом у світі розвиток платіжних систем характеризується зростанням кількості банкоматів (рис. 1.21). Таким чином, для забезпечення безпеки банківської системи необхідно застосовувати засоби захисту інформації на кожному рівні системи.

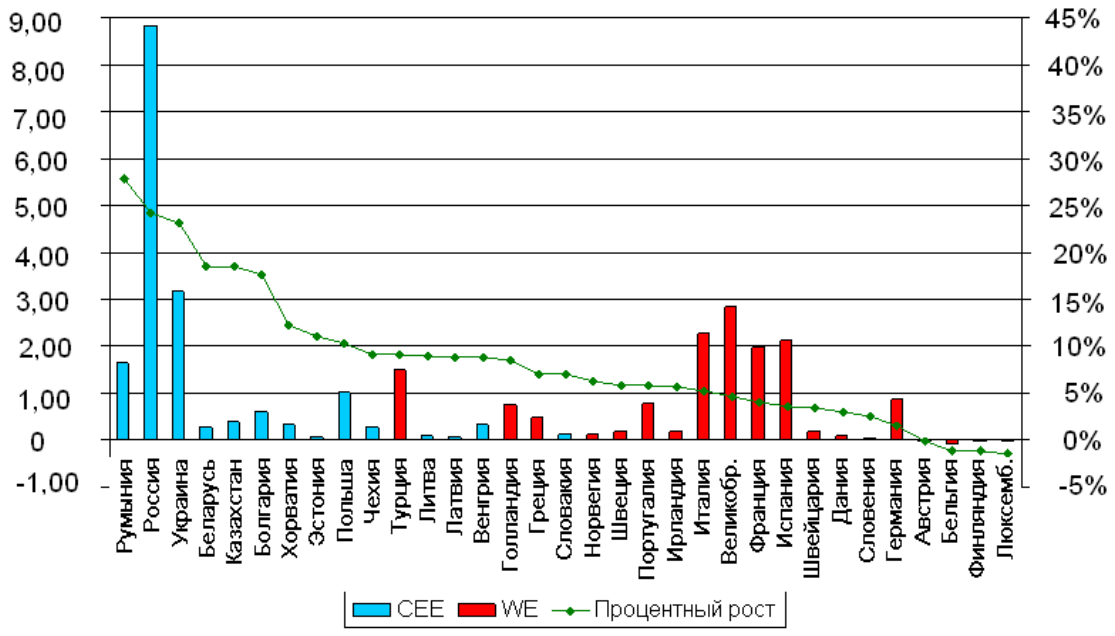

Рис. 1.21. Абсолютний і відносний ріст кількості банкоматів

1.4. Висновки за першим розділом

Головним національним інтересом є розвиток економіки та добробут громадян України. Забезпечення національних інтересів та економічної безпеки – найважливіші функції держави, реалізація яких покликана посилювати позиції в міжнародному співтоваристві.

Сьогодні як ніколи загострюється надзвичайно важливе питання забезпечення інформаційної безпеки України, що є одним з найважливіших національних пріоритетів [36, 37].

Розвиток високорентабельної економіки неможливий без впровадження сучасної системи грошового обігу та використання ефективних платіжних механізмів. Швидкий ріст обсягів оброблюваних даних у сучасних банківських системах, поява нових електронних послуг, стрімкий розвиток обчислювальної техніки висувають нові вимоги до надійності та забезпечення безпеки даних у ВПБС. Незважаючи на широке їх застосування, ВПБС піддані різним атакам і загрозам.

Для забезпечення надійного захисту необхідний комплексний підхід, що включає в себе аналіз: загальної структури ВПБС, можливих загроз і реалізованих атак; вибір ратифікованих стандартів для забезпечення автентичності, цілісності та конфіденційності банківських транзакцій, програмна реалізація обраних криптографічних алгоритмів.

РОЗДІЛ 2 РОЗРОБКА АЛГОРИТМУ БЛОЧНОГО СИМЕТРИЧНОГО КРИПТОГРАФІЧНОГО ПЕРЕТВОРЕННЯ ІНФОРМАЦІЇ З ДИНАМІЧНО КЕРОВАНИМИ ПРИМІТИВАМИ (ALGORITHM OF DYNAMIC ENCRYPTION – ADE)

2.1 Обґрунтування конструкції та структурної схеми алгоритму блочного симетричного криптографічного перетворення інформації з динамічно керованими примітивами

Відповідно до основних положень концепції Національної безпеки України витік конфіденційної інформації вважається однією з найнебезпечніших загроз безпеки в інформаційній сфері [4]. Основним комплексним заходом щодо захисту національного інформаційного простору є побудова Національної системи конфіденційного зв'язку, в якій за допомогою криптографічних і технічних засобів захисту інформації реалізуються послуги інформаційної безпеки [39].

Основним і найбільш ефективним механізмом криптографічного захисту інформації є методи блочного симетричного криптографічної перетворення [11, 37, 39]. Поряд з високою швидкістю перетворень і простотою практичної реалізації симетричні криптоалгоритми дозволяють забезпечити високу стійкість до різних методів криптографічного аналізу [35].

Основу захисту інформації у внутрішньоплатіжних системах становить алгоритм ГОСТ 28147-89. Але на сьогодні він застарів через низьку швидкодію і через те, що використовує S-box, які є власністю Російської Федерації. Тому альтернативним варіантом є використання найбільш ефективного симетричного блочного криптографічного алгоритму за показниками швидкодії і стійкості, а також за відкритістю і прозорістю всіх механізмів перетворень, визнаного переможцем відкритого конкурсу Advanced Encryption Standard (AES) – алгоритму Rijndael. Алгоритм Rijndael був затверджений згодом в якості стандарту симетричного шифрування США (FIPS-197) [64–67]. Тривалі

дослідження (1999 – 2004) підтвердили надійність та ефективність AES, його стійкість проти відомих на той час криптоаналітичних атак [53, 70]. Даний стандарт шифрування одержав останнім часом найбільше практичне впровадження і є де-факто еталоном для знову розроблювальних симетричних блочних криптографічних алгоритмів.

В ході проведених численних досліджень установлене, що для шифру AES доцільно застосовувати методи атак, які б використовували алгебраїчну простоту його SPN-структури [70]. Подібні методи одержали назву алгебраїчних атак і в останні роки переживають бурхливий розвиток [53, 59]. Простота та елегантність структури шифру AES не має на увазі таку ж простоту в реалізації алгебраїчних методів криптоаналізу. Хоча на сьогоднішній день невідомі успішні атаки на повний AES, алгебраїчні атаки при можливому комбінуванні з іншими відомими підходами можна розглядати як реальну загрозу надійності даного шифру. Численні публікації [12, 53, 70] про можливе існування швидких алгебраїчних атак на AES підтверджують необхідність проведення досліджень у цій області.

В основі алгебраїчної атаки лежить, як правило, складання системи рівнянь, що зв’язують значення відкритого тексту, шифр тексту та ключа, і, можливо, деякі величини проміжних обчислень і значень раундових ключів. Використовуючи деякі вхідні дані, такі як пари “відкритий текст – шифротекст”, криптоаналітик підставляє ці значення у відповідні змінні в сформованій системі рівнянь і намагається її розв’язати, розкриваючи, таким чином, секретний ключ. Є множина підходів до складання алгебраїчних рівнянь, що аналітично зв’язують стан відкритого тексту, шифротексту та ключа. У той же час практично всі дослідники сходяться в думці, що алгебраїчна структура AES приводить до деякої “одноманітності” породжуваних рівнянь.

Таким чином, алгебраїчні атаки на блочні шифри є галуззю активних досліджень, саме їх реалізація найбільш очікувана для алгоритмів типу AES. Розвиток нових алгебраїчних методів рішення систем нелінійних рівнянь, у комбінації зі зростаючою ефективністю програмного та апаратного

забезпечення, спонукає до створення шифрів, які при тій же простоті та математичній прозорості (типу AES) будуть описуватися більш складними системами рівнянь. На практиці це означає:

– підвищення числа невідомих при заданій кількості рівнянь;

– підвищення числа одночленів (термів), що входять у кожне рівняння;

образно кажучи бажано, щоб у кожне з рівнянь входило близько половини можливих термів;

– збільшення “різноманітності” термів, що входять у кожне рівняння.

Практично це означає, що в рівняння повинні входити терми з різним степенем і з різними змінними.

Одним з перспективних напрямків у розвитку симетричних криптоалгоритмів є застосування динамічно керованих раундовими ключами криптопримітивів [36]. Передбачається, що використання динамічно змінюваних блоків нелінійних замін і матриць лінійного розсіювання за рахунок лавиноподібного збільшення різностепених невідомих значно збільшить складність системи рівнянь, що описують шифр [63]. Саме ця ідея є в основі алгоритму ADE (Algorithm Of Dynamic Encryption).

Симетричний криптографічний алгоритм ADE є ітеративним блочним шифром з різною довжиною блоку та різною довжиною ключа. По своїй структурі він належить сімейству підстановочно-перестановочних шифрів (SPN – Substitution-Permutation Network), запропонованих К. Шенноном [22, 63].

Щоб зашифрувати блок відкритого тексту використовується почерговість операцій нелінійного (заміни), лінійного (перестановки) перетворення та деяких функціональних перетворень, наприклад, зрушення, додавання по модулю та ін.

Для одержання гарних статистичних властивостей шифру всі види перетворень, що шифрують, повинні залежати від значення секретного ключа, виробленого, по можливості, випадковим образом (датчики ключів також детерміновані обладнанням та будь-яке твердження про випадковий характер формування ключових даних цими датчиками не коректне). В цьому випадку використання алгебраїчних методів криптоаналізу породжує велику кількість

різноманітних рівнянь (якщо набір функціональних перетворень не можна перетворити до одного еквівалентного перетворенню з тим же числом невідомих параметрів, то число рівнянь буде рости експоненційно). В той же час, практична реалізація такого підходу складна та для її спрощення К. Шеннон запропонував використовувати фіксовані блоки підстановок і перестановок, а введення випадкового секретного параметра (ключа) реалізувати за допомогою найпростішого функціонального перетворення – додавання по модулю 2.

Ідея ітеративного підстановочно-перестановочного шифрування була запропонована вченим Фейстелем і практично реалізована в шифрі Lucifer, прототипом якого став алгоритм DES – національний стандарт шифрування США. З метою покращення ефективності практичної реалізації на процесорах з малою розрядністю класична SPN-структура ще більше спрощена. Дані для ітераційної обробки розбивалися на 2 блоки, обробка криптопримітивами проводилася над підблоками. У літератруних джерелах така схема підстановочно-перестановочного шифру одержала назву ланцюга або комірки Файстеля [52].

Таким чином, класичний шифр, що став, DES є значно спрощеною реалізацією ітеративного підстановочно-перестановочного підходу до симетричного шифрування, заснованого К. Шенноном. Внесені спрощення носять наступний характер:

– SPN – структура замінена простіше реалізованим ланцюгом Файстеля;

– введення випадкового секретного параметра (ключа) реалізоване найпростішим функціональним перетворенням – додавання по модулю 2;

– реалізовані блоки підстановок і перестановок мають невеликий розмір.

Подальший розвиток алгоритмів шифрування (на прикладі AES) був спрямований на посилення стійкості до різних методів криптоаналізу. Із цією метою розмірність застосовуваних фіксованих блоків нелінійного (підстановочного) і лінійного (перестановочного) перетворення була значно

збільшена. Крім того, класична перестановка, як лінійне перетворення, замінена на мікшировання – лінійне перетворення, побудоване з використанням лінійних блочних кодів. Цей перехід дозволив увести та теоретично обґрунтувати основний показник ефективності – мінімальне число відмінностей “вхід-вихід”, що називають числом галузей розсіювання. Зміни торкнулися так само та самої структури шифру – розроблювачі AES повернулися до класичної SPN – структурі [57]. Внесені зміни, безсумнівно, підвищили складність практичної реалізації алгоритму шифрування, але наблизили його до класичного трактування підстановочно-перестановочного шифру.

Відзначимо той факт, що введення випадкового секретного параметра (ключа), як і в алгоритмі DES, реалізоване найпростішим функціональним перетворенням – додаванням по модулю 2. З однієї сторони це дозволило, не перевантажуючи схему шифрування додатковими складними перетвореннями, досягти високих показників швидкодії. З іншого боку – цей підхід забезпечує внесення на кожному раунді шифрування деякої невизначеності, що задається раундовим ключем, що та вироджується при алгебраїчних методах криптоаналізу в систему різноманітних рівнянь. В той же час, особливості побудови нелінійних блоків замін алгоритму AES дозволяють спростити класифікацію породжуваних рівнянь. Забігаючи вперед, скажемо, що блоки підстановок реалізуються інвертуванням елементів над кінцевим полем, що еквівалентно багаторазовому зведенню у квадрат і перемножуванню результатів над тим же полем. Фактично це дозволяє казати про “однотипність” породжуваних рівнянь, дозволяє суттєво спростити їхнє рішення.

Таким чином, алгоритм шифрування AES, що є на сьогоднішній день найбільш ефективним по співвідношенню швидкодія/ стійкість не задовольняє тільки одному класичному визначенню підстановочно-перестановочної схеми – уведення випадкового секретного параметра (ключа) реалізоване тільки в одному найпростішому перетворенні, інші блоки не залежать від значення ключа та реалізують детерміноване перетворення. Цей недолік, в сукупності з

регулярними правилами побудови блоку замін є передумовою для появи ефективних методів криптоаналізу, зокрема, алгебраїчних методів.

Розроблений алгоритм ADE побудований за класичною підстановочно-перестановочною схемою та використовує всі переваги алгоритму AES. Основна увага при розробці ADE приділена можливості динамічно управляти процесом лінійного розсіювання та нелінійної заміни в ході криптографічного перетворення інформації. З цією метою в базову структуру алгоритму AES уведені динамічно змінювані блоки (криптопримітиви) лінійного розсіювання та нелінійної заміни, правила функціонування яких задаються значеннями випадкового секретного параметра (раундового ключа).

Властивості введених примітивів криптоалгоритму ADE не уступають криптопримітивам алгоритму AES, а за рахунок їх динамічної зміни на кожному раунді ітеративної обробки вдається динамічно управляти процесом криптографічного перетворення інформації. Це, спричиняє ускладнення процесу вираження взаємозв'язків між відкритим текстом, шифротекстом і ключем системою рівнянь, що суттєво ускладнює завдання криптоаналізу, зокрема, заснованого на алгебраїчних методах. Структурна схема алгоритму ADE у загальному виді представлено на рис. 2.1 [40].

Надалі будуть використані наступні умовні позначення:

– "Subbyte" – блок нелінійного перетворення (підстановка), що функціонує під керуванням раундового ключа;

– "Shiftrow " – блок функціонального перетворення (на основі циклічного зсуву), що функціонує під керуванням раундового ключа;

– "Mixcolumn" – блок лінійного перетворення (аналог перестановки), що функціонує під керуванням раундового ключа;

– "Addroundkey " – блок функціонального перетворення (додавання по модулю 2 (XOR) випадкового секретного параметра (ключа)).

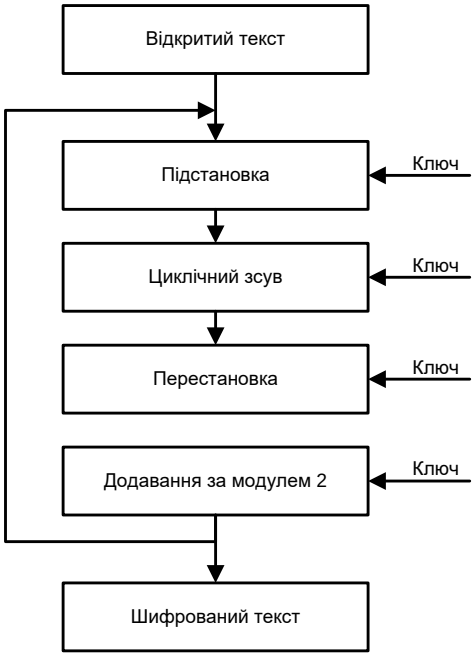

Рисунок 2.1 - Структурна схема алгоритму шифрування ADE

Таким чином, у спрощеному (частці) випадку функціонування криптопримітивів ADE зводиться до відповідних блоків алгоритму AES, тобто при фіксуванні деяких системних параметрів алгоритм ADE легко трансформується в AES.

2.2 Специфікація складених компонентів шифру, стан, ключ шифру та кількість раундів

Згідно зі специфікацією алгоритму ADE, різні перетворення оперують із проміжним результатом, називаним Станом$_i$. На відміну від Стану алгоритму AES, Стан$_i$ алгоритму ADE, залежно від розрядності процесора, представляється у вигляді двомірного масиву байтів з $4i$ рядків, $i = 1,2,...,32$ і Nb стовпців. Кількість стовпців Nb дорівнює довжині блоку, діленої на $32i$.

Ключ шифра$_i$ також представляється у вигляді двомірного масиву з $4i$ рядками. Кількість стовпців Ключа шифра$_i$ позначається через Nk (дорівнює довжині ключа діленої на 32). Ці представлення проілюстровані на рис. 2.2 [40].

Вхідні та вихідні дані, використовувані в ADE, як і в зовнішньому інтерфейсі AES, є одномірними масивами байтів, пронумерованих в порядку зростання від 0 до $4iNb-1$. Ці блоки мають довжини $16i, 24i, 32i, 48i, 56i$ і $64i$ байт, індекси масивів у межах $0..16i-1$, $0..24i-1$, $0..32i-1$, $0..48i-1$, $0..56i-1$, $0..64i-1$.

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$	...	$a_{0,Nb}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	...	$a_{1,Nb}$
...	...	...	...	...	...
$a_{4i-1,0}$	$a_{4i-1,1}$	$a_{4i-1,2}$	$a_{4i-1,3}$	...	$a_{4i-1,Nb}$

$k_{0,0}$	$k_{0,1}$	...	$k_{0,Nk}$
$k_{1,0}$	$k_{1,1}$	...	$k_{1,Nk}$
...	...	...	...
$k_{4i-1,0}$	$k_{4i-1,1}$	...	$k_{4i-1,Nk}$

Рисунок 2.2 а - Представлення Стану$_i$ (з Nb = 6) і Ключа шифра$_i$ (Nk = 4)

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$	$a_{0,4}$	$a_{0,5}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,4}$	$a_{1,5}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	$a_{2,4}$	$a_{2,5}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$	$a_{3,4}$	$a_{3,5}$

$k_{0,0}$	$k_{0,1}$	$k_{0,2}$	$k_{0,3}$
$k_{1,0}$	$k_{1,1}$	$k_{1,2}$	$k_{1,3}$
$k_{2,0}$	$k_{2,1}$	$k_{2,2}$	$k_{2,3}$
$k_{3,0}$	$k_{3,1}$	$k_{3,2}$	$k_{3,3}$

Рисунок 2.2 б - Приклад представлення Стану$_1$ (з Nb = 6) і Ключа шифру$_1$ (Nk = 4)

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$	$a_{0,4}$	$a_{0,5}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,4}$	$a_{1,5}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	$a_{2,4}$	$a_{2,5}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$	$a_{3,4}$	$a_{3,5}$
$a_{4,0}$	$a_{4,1}$	$a_{4,2}$	$a_{4,3}$	$a_{4,4}$	$a_{4,5}$
$a_{5,0}$	$a_{5,1}$	$a_{5,2}$	$a_{5,3}$	$a_{5,4}$	$a_{5,5}$
$a_{6,0}$	$a_{6,1}$	$a_{6,2}$	$a_{6,3}$	$a_{6,4}$	$a_{6,5}$
$a_{7,0}$	$a_{7,1}$	$a_{7,2}$	$a_{7,3}$	$a_{7,4}$	$a_{7,5}$

$k_{0,0}$	$k_{0,1}$	$k_{0,2}$	$k_{0,3}$
$k_{1,0}$	$k_{1,1}$	$k_{1,2}$	$k_{1,3}$
$k_{2,0}$	$k_{2,1}$	$k_{2,2}$	$k_{2,3}$
$k_{3,0}$	$k_{3,1}$	$k_{3,2}$	$k_{3,3}$
$k_{4,0}$	$k_{4,1}$	$k_{4,2}$	$k_{4,3}$
$k_{5,0}$	$k_{5,1}$	$k_{5,2}$	$k_{5,3}$
$k_{6,0}$	$k_{6,1}$	$k_{6,2}$	$k_{6,3}$
$k_{7,0}$	$k_{7,1}$	$k_{7,2}$	$k_{7,3}$

Рисунок 2.2.в - Приклад представлення Стану$_{i\ 2}$ (з Nb = 6) і Ключа шифру$_2$ (Nk = 4)

Ключ шифру розглядається як одномірний масив байтів, пронумерований у порядку зростання від 0 до $4iNk-1$. Ці блоки мають довжини $16i$, $24i$, $32i$, $48i$, $56i$ і $64i$ байт, індекси масивів у межах $0..16i-1$, $0..24i-1$, $0..32i-1$, $0..48i-1$, $0..56i-1$, $0..64i-1$.

Вхідні дані шифру ("відкритий текст", якщо використовується режим шифрування ECB) відображаються в байти $Стану_i$ в порядку $a_{0,0}, a_{1,0}, ..., a_{4i-1,0}, a_{0,1}, a_{1,1}, ..., a_{4i-1,1}, a_{4,1} ...$, і байти ключа шифру також відображаються в масив в порядку $k_{0,0}, k_{1,0}, ..., k_{4i-1,0}, k_{0,1}, k_{1,1}, ..., k_{4i-1,1}, k_{4,1} ...$ Наприкінці роботи шифру вихідні дані витягають із $Стану_i$ за допомогою вибору байтів у такому ж порядку.

Отже, якщо одномірним індексом байта в блоці є n, і двовимірним індексом є (l, j), ми маємо:

$$l = n \bmod 4i; \qquad j = [n / 4i]; \quad n = l + 4i * j.$$

Далі, індекс l є також номером байта в $4i$-байтному векторі або слові і j є індексом вектора (або слова) в окремо взятому блоці.

Кількість раундів позначена через Nr і залежить від значень Nb і Nk (відповідні значення зазначено в таблиці 2.1).

Подальше збільшення довжин блоку та ключа може бути виконане із кроком у $4i$ байта. Кількість раундів $Nr(Nb, Nk)$ як функція довжин блоку та ключа розраховується за виразом:

$$Nr(Nb, Nk) = \max(Nb, Nk) + 6.$$

Таблиця 2.1 -Кількість раундів $Nr(Nb, Nk)$ як функція довжин блоку та ключа

Nr	Nb = 4	Nb = 6	Nb = 8	Nb = 10	Nb = 12	Nb = 16	…	Nb
Nk = 4	10	12	14	16	18	22	…	Nr
Nk = 6	12	12	14	16	18	22	…	Nr
Nk = 8	14	14	14	16	18	22	…	Nr
Nk = 10	16	16	16	16	18	22	…	Nr
Nk = 12	18	18	18	18	18	22	…	Nr
Nk = 16	22	22	22	22	22	22	…	Nr
…	…	…	…	…	…	…	…	Nr
Nk	Nr	Nr	Nr	Nr	Nr	Nr	Nr	Nr

2.3 Дослідження раундових перетворень із динамічно керованими криптопримітивами

2.3.1 Дослідження динамічно керованого нелінійного перетворення Subbyte

Практично всі операції ADE визначаються на рівні байта. Байти можна розглядати як елементи кінцевого поля $GF(2^8)$. Деякі операції визначені в термінах четирьохбайтних слів. Введемо основні математичні поняття, необхідні для обговорення алгоритму:

а) Поле $GF(2^8)$. Елементи кінцевого поля можуть бути представлені кількома різними способами. Для будь-якого степеня простого числа існує єдине кінцеве поле, тому всі уявлення $GF(2^8)$ є ізоморфними [56]. Незважаючи на подібну еквівалентність, подання впливає на складність реалізації. Виберемо класичне поліноміальний подання.

Байт b, що складається з бітів b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0, представляється у вигляді полінома з коефіцієнтами з (0, 1):

$$b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x^1 + b_0$$

Приклад: байт з шістнадцятковим значення “57” (бінарне 01010111) відповідає поліному $x^6 + x^4 + x^2 + x + 1$.

б) Додавання. У поліноміальному поданні сума двох елементів є поліномом з коефіцієнтами, які дорівнюють сумі за модулем 2 (тобто 1 + 1 = 0) коефіцієнтів доданків.

Приклад: “57”+ “83” = “ DA ” або в поліноміальній нотації:

$$(x^6 + x^4 + x^2 + x + 1) + (x^7 + x + 1) = x^7 + x^6 + x^4 + x^2$$

В бінарній нотації маємо: 01010111 + 10000011 = 11011010. Очевидно, що складання відповідає простому XOR (позначається як $\oplus$) на рівні байта.

Виконані всі необхідні умови Абелевих груп: операція додавання (кожній парі елементів зіставляється третій елемент групи, званий їх сумою), асоціативність, нульовий елемент (“00”), зворотний елемент (щодо операції додавання) і комутативність.

в) Множення. В поліноміальному поданні множення в $GF(2^8)$ відповідає множенню поліномів за модулем неприводимого двійкового полінома степеня 8. Поліном є неприводимим, якщо він не має дільників, крім 1 і самого себе. Для ADE такий поліном називається m(x) і визначається таким чином:

$$m(x) = x^8 + x^4 + x^3 + x + 1$$

або “11B ” в шістнадцятковому представленні.

Приклад: перемноження “57” і “83” = “C1”

або

$$(x^6 + x^4 + x^2 + x + 1)(x^7 + x + 1) =$$
$$= x^{13} + x^{11} + x^9 + x^8 + x^7 + x^7 + x^5 + x^3 + x^2 + x + x^6 + x^4 + x^2 + x + 1 =$$
$$= x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1$$
$$(x^8 + x^4 + x^3 + x + 1)(x^5 + x^3) + x^7 + x^6 + 1 = x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1$$

Отже,

$$x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^8 + 1 \bmod (x^8 + x^4 + x^3 + x + 1) = x^7 + x^6 + 1$$

Ясно, що результат є двійковим поліномом не вище 8 степеня. На відміну від складання, простої операції множення на рівні байтів не існує.

Множення є асоціативним, і існує одиничний елемент (“01”). Для будь-якого двійкового полінома b(x) не вище 8-го степеня можна використовувати розширений алгоритм Евкліда для обчислення поліномів a (x) і c (x) таких, що

$$b(x)\, a(x) + m(x)\, c(x) = 1$$

Отже,

$$a(x) \bullet b(x) \bmod m(x) = 1$$

або

$$b^{-1}(x) = a(x) \bmod m(x)$$

Більш того, можна показати, що

$$a(x) \bullet (b(x) + c(x)) = a(x) \bullet b(x) + a(x) \bullet c(x)$$

З усього цього випливає, що багато з 256 можливих значень байта утворює кінцеве поле $GF(2^8)$ с XOR як додавання і множення.

г) Множення на x. Якщо помножити b(x) на поліном x, будемо мати:

$$b_7x^8 + b_6x^7 + b_5x^6 + b_4x^5 + b_3x^4 + b_2x^3 + b_1x^2 + b_0x$$

x • b (x) виходить пониженням попереднього результату за модулем m(x). Якщо $b_7 = 0$, то дане зниження є тотожною операцією. Якщо $b_7 = 1$, m(x) слід відняти (тобто XORed). З цього випливає, що множення на x може бути реалізовано на

рівні байти як лівий зсув і подальший побітового XOR с “1B”. Дана операція позначається як b = xtime (a).

ґ) Поліноми з коефіцієнтами з GF. Поліноми можуть бути визначені з коефіцієнтами з $GF(2^8)$. У цьому випадку четирьохбайтний вектор відповідає поліному степеня 4.

Поліноми можуть бути складені простим додаванням відповідних коефіцієнтів. Як додавання у $GF(2^8)$ є побітовим XOR, так і додавання двох векторів є простим побітовим XOR.

Множення є більш складною дією. Припустимо, що маємо два полінома в $GF(2^8)$.

$$a(x) = a_3x^3 + a_2x^2 + a_1x^1 + a_0$$
$$b(x) = b_3x^3 + b_2x^2 + b_1x^1 + b_0$$

c(x) = a(x) b(x) визначається наступним чином:

$$c(x) = c_6x^6 + c_5x^5 + c_4x^4 + c_3x^3 + c_2x^2 + c_1x + c_0$$

$$c_0 = a_0 \bullet b_0$$
$$c_1 = a_1 \bullet b_0 \oplus a_0 \bullet b_1$$
$$c_2 = a_2 \bullet b_0 \oplus a_1 \bullet b_1 \oplus a_0 \bullet b_2$$
$$c_3 = a_3 \bullet b_0 \oplus a_2 \bullet b_1 \oplus a_1 \bullet b_2 \oplus a_0 \bullet b_3$$
$$c_4 = a_3 \bullet b_1 \oplus a_2 \bullet b_2 \oplus a_1 \bullet b_3$$
$$c_5 = a_3 \bullet b_2 \oplus a_2 \bullet b_3$$
$$c_6 = a_3 \bullet b_3$$

Ясно, що в такому вигляді c(x) не може бути представлений четирьохбайтним вектором. Знижуючи c(x) по модулю полінома 4-го степеня, результат може бути поліномом степеня нижче 4. В ADE це зроблено за допомогою полінома $M(x) = x^4 + 1$ так як $x^j \bmod (x^4 + 1) = x^{j \bmod 4}$

Модуль, що отримується з a (x) і b (x), що позначається d (x) = a (x)⊕b (x), виходить таки м чином:

$$d_0 = a_0 \bullet b_0 \oplus a_3 \bullet b_1 \oplus a_2 \bullet b_2 \oplus a_1 \bullet b_3$$

$$d_1 = a_1 \bullet b_0 \oplus a_0 \bullet b_1 \oplus \bullet b_2 \oplus a_2 \bullet b_3$$

$$d_2 = a_2 \bullet b_0 \oplus a_1 \bullet b_1 \oplus a_0 \bullet b_2 \oplus a_3 \bullet b_3$$

$$d_3 = a_3 \bullet b_0 \oplus a_2 \bullet b_1 \oplus a_1 \bullet b_2 \oplus a_0 \bullet b_3$$

Операція, що складається з множення фіксованого полінома a(x), може бути записана як множення матриці, де матриця є циклічною [56]. Маємо

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 & a_1 & a_0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Перетворення Subbyte є нелінійною заміною байтів, яка незалежно оперує з кожним байтом Стан_i a і з байтом раундового ключа γ.

В якості S-блоку виступає змінювана матриця підстановок, яка будується одержанням мультиплікативно зворотного елемента $(a \cdot \gamma)^{-1}$ над кінцевим полем $GF(2^8)$ і шляхом виконання афінного перетворення

$$b = M \cdot (a \cdot \gamma)^{-1} + \beta$$

над примітивним полем $GF(2)$, M – квадратна невиражена матриця 8x8:

$$\text{Ì} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix},$$

β – вектор з 8 елементів:

$$\beta^{\grave{O}} = (1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0).$$

Підстановка – стохастичне перетворення, яке є нелінійною заміною байт, виконуваною за допомогою S-блоків, незалежно для кожного вхідного байта та реалізує відображення

$$\varphi : (A, \Gamma) \to B,$$

де A - множина вхідних векторів $a = \{a_0, a_1, ..., a_7\}$, B - множина вихідних векторів $b = \{b_0, b_1, ..., b_7\}$, Ã - множина випадкових векторів $\gamma = \{\gamma_0, \gamma_1, ..., \gamma_7\}$, які задаються раундовим ключем, $\tilde{A} \in GF(2^8)$. Вихідний вектор можна представити як

$$y = \varphi(x, \gamma) = M \cdot (x \cdot \gamma)^{-1} + \beta,$$

де множення $x \cdot \gamma$ виконується над кінцевим полем $GF(2^8)$.

Кожний $b = \{b_0, b_1, ..., b_7\}$ залежить як від $a = \{a_0, a_1, ..., a_7\}$, так і від випадкової величини $\gamma = \{\gamma_0, \gamma_1, ..., \gamma_7\}$, яка задається значенням раундового ключа. Таким чином, розглянутий підхід дозволяє виконувати криптографічне перетворення даних, динамічно змінюючи блоки нелінійних замін з фіксованим показником нелінійності та динамічно управляти ітеративною обробкою інформаційних даних.

Так, наприклад, при виборі вектора “4D”

$$\gamma = "4D" = \{1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0\}$$

відповідна матриця підстановки має вигляд таблиці 2.2.

Застосування описаного S-блоку до всіх байтів Стану$_i$ позначається як Subbyte(State, Roundkey). Рис.2.3 ілюструє ефект перетворення Subbyte до Стану$_1$[37, 40].

Таблиця 2.2 – Таблиця замін при зміні випадкової величини

		a_2															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
a_1	0	63	E3	B8	0E	15	AA	D5	41	58	7D	1C	A7	A3	EB	E9	24
	1	65	A1	F7	7F	DC	1D	01	1B	98	79	BC	96	BD	CD	5B	A2
	2	FB	31	99	8D	29	7C	F6	14	27	51	5C	87	C9	CF	C4	A6
	3	9E	4F	6E	30	8C	7A	02	CC	0C	4B	AF	B1	E4	11	18	B6
	4	B4	3D	4A	82	1E	49	8F	B2	46	03	77	84	A9	2D	D8	D0
	5	DA	9A	E1	95	FC	AD	8A	13	36	F9	AE	0D	2B	5A	81	86
	6	06	9B	75	40	7E	C6	CA	4C	0F	4E	EF	71	48	EE	2F	7B
	7	D4	20	EC	8B	05	21	91	F8	A0	67	C1	60	45	3F	89	08
	8	88	57	D7	09	6C	E6	93	A8	DD	5F	ED	72	8E	62	10	C7
	9	F1	33	C8	B0	F2	3B	0B	66	9D	07	DF	3A	BE	F0	BA	5D
	A	BF	56	9F	26	B9	90	83	FE	AC	94	04	FF	97	E8	C0	00
	B	52	6B	B5	DB	85	E5	54	E7	47	39	64	78	12	92	0A	28
	C	D1	9C	1F	44	F3	C3	69	D3	76	2C	2A	61	B7	3C	F4	68
	D	55	E0	F5	80	25	73	6A	59	6D	BB	A5	43	DE	3E	6F	B3
	E	23	D2	C2	D9	A4	53	17	74	50	FD	42	EA	1A	AB	35	22
	F	19	2E	FA	CB	32	CE	E2	38	70	5E	D6	C5	16	37	4D	34

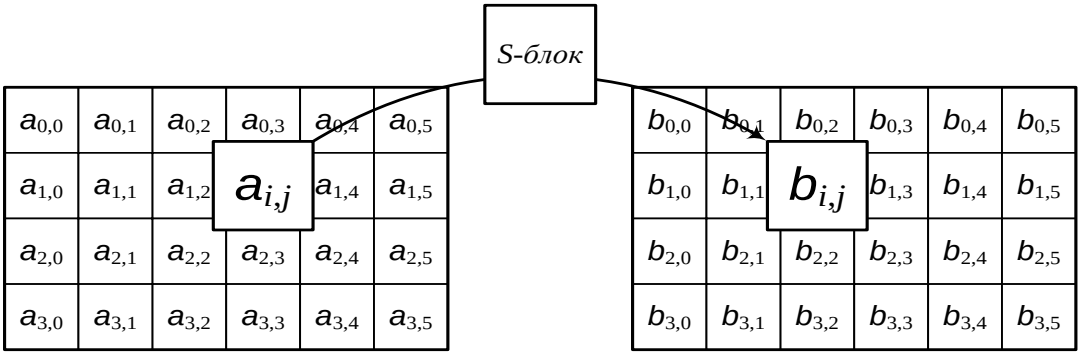

Рисунок 2.3 - Схема роботи Subbyte з окремими байтами Стану

Зворотним до Subbyte є заміна байтів із застосуванням інвертованої таблиці. Це досягається обігом афінного відображення, за яким іде взяття мультиплікативно зворотного в поле $GF(2^8)$.

Слід зазначити, що при виборі восьмирозрядного вектора “01”

$$\gamma = \text{“01”} = \{1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0\}$$

перетворення Bytesub алгоритму ADE повністю відповідає блоку Subbyte алгоритму AES. Відповідна матриця підстановки має вигляд таблиці 2.1, що відповідає таблиці замін алгоритму AES.

Таким чином, у спрощеному випадку (при $\gamma = \text{“01”}$) функціонування криптопримітива Subbyte в ADE зводиться до відповідного блоку алгоритму AES, тобто на цьому етапі зашифровування алгоритм ADE легко трансформується в AES.

2.3.2 Дослідження динамічно керованого лінійного перетворення Shiftrows

У перетворенні Shiftrows (див. рис.2.4) рядки Стану$_i$ зрушуються на різну кількість позицій, що задаються значенням раундового ключа (байта).

Через вектора

$$\begin{aligned}
\lambda^1 &= \{\lambda^1_0 \quad \lambda^1_1 \quad \lambda^1_2 \quad \lambda^1_3 \quad \lambda^1_4 \quad \lambda^1_5 \quad \lambda^1_6 \quad \lambda^1_7\},\\
\lambda^2 &= \{\lambda^2_0 \quad \lambda^2_1 \quad \lambda^2_2 \quad \lambda^2_3 \quad \lambda^2_4 \quad \lambda^2_5 \quad \lambda^2_6 \quad \lambda^2_7\},\\
&\cdots\\
\lambda^i &= \{\lambda^i_0 \quad \lambda^i_1 \quad \lambda^i_2 \quad \lambda^i_3 \quad \lambda^i_4 \quad \lambda^i_5 \quad \lambda^i_6 \quad \lambda^i_7\}
\end{aligned}$$

позначимо i байт раундового ключа, $\lambda^i_1 \in [0,1]$.

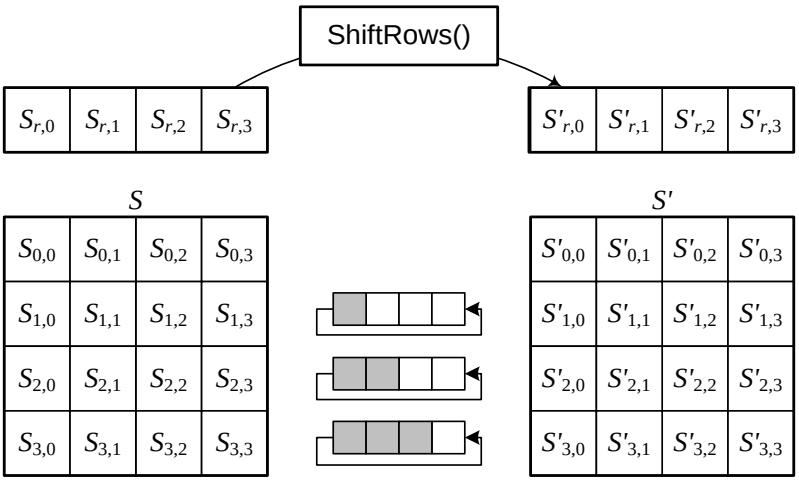

Рисунок 2.4 - Схема виконання перетворення Shiftrows над Станом

Значення $\{\lambda^u_0, \lambda^u_1\}$ байта λ^u задають кількість зсувів $C_{4(u-1)}$ для $4(u-1)$ рядка Стану$_i$, значення $\{\lambda^u_2, \lambda^u_3\}$ байта λ^u задають число зрушень $C_{4(u-1)+1}$ для $4(u-1)+1$ рядка Стану$_i$, значення $\{\lambda^u_4, \lambda^u_5\}$ байта λ^u задають кількість зсувів $C_{4(u-1)+2}$ для $4(u-1)+2$ рядка Стану$_i$, значення $\{\lambda^u_6, \lambda^u_7\}$ байта λ^u задають число зрушень $C_{4(u-1)+3}$ для $4(u-1)+3$ рядка Стану$_i$, $u = 1..i$ [40].

Відзначимо, що при $i = 1$ та відповідних значеннях елемента вектора $\lambda^1 = \{\lambda_0, \lambda_1, ..., \lambda_7\}$ функціонування криптопримітива Shiftrows ADE зводиться до відповідного до блоку алгоритму AES, тобто на цьому етапі алгоритм ADE легко трансформується в AES.

Так, наприклад, при

$$\lambda^1 = \{0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1\}$$

маємо:

$$C_0 = 0,\ C_1 = 1,\ C_2 = 2,\ C_3 = 3,$$

тобто рядок 0 не зміщається, рядок 1 зміщається на 1 байт, рядок 2 – на 2 байта та рядок 3 – на 3 байта, що повністю відповідає функціонуванню блоку Shiftrows алгоритму AES.

Операція зрушення рядків Стану$_1$ на певну величину позначається через Shiftrows(State, Roundkey).

Зворотним до Shiftrows є циклічне зрушення на $Nb - C_{4(u-1)}$, $Nb - C_{4(u-1)+1}$, $Nb - C_{4(u-1)+2}$, $Nb - C_{4(u-1)+3}$, … байт відповідно так, що байт на позиції j в рядку c_l зрушується на позицію $(j + Nb - C_l) \bmod Nb$.

Рис.2.5 ілюструє ефект впливу перетворення Shiftrow на Стан$_1$.

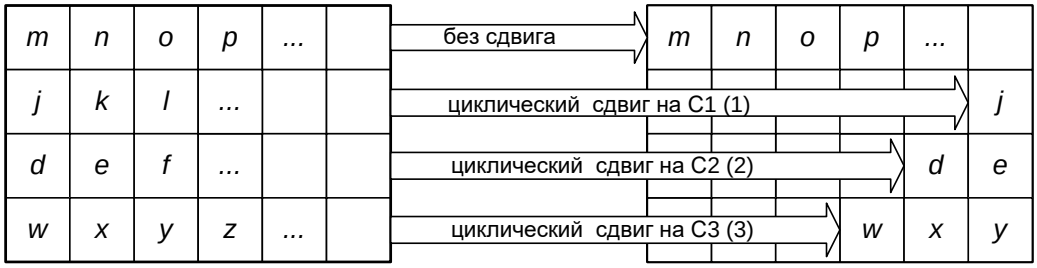

Рисунок 2.5 - Вплив Shiftrows на рядки Стану$_1$

Допускається реалізація перетворення Shiftrows з фіксованими значеннями C_l, наприклад, $C_0 = 0, C_1 = 1, C_2 = 2, C_3 = 3$, …, $C_l = l$.

2.3.3 Дослідження динамічно керованого лінійного перетворення Mixcolumn

У перетворенні Mixcolumn (див. рис.3) алгоритму AES (найближчого прототипу ADE) стовпці Стану розглядаються як многочлени над полем $GF(2^8)$ і множаться по модулю $x^4 + 1$ з фіксованим многочленом $c(x)$, заданим як

$$c(x) = '03'x^3 + '01'x^2 + '01'x + '02'.$$

Цей многочлен є взаємнопростим з $x^4 + 1$ і тому інвертуємим [40]. Ця процедура може бути представлена як матричне множення.

Нехай $b(x) = c(x) \otimes a(x)$, тоді

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

Операція Mixcolumn алгоритму AES виконує лінійне відображення

$$\theta : GF\left(2^8\right)^k \to GF\left(2^8\right)^k, k = 4,$$

причому число галузей лінійного розсіювання (у термінах, введених розроблювачами алгоритму AES), визначене вираженням

$$B_{k,k}(\theta) = \min_{a \neq 0}\left\{w_h(a) + w_h(\theta(a))\right\} = \min_{a \neq 0}\left\{w_h(a) + w_h(b)\right\},$$

де $w_h(x)$ – вага Хемінга вектора x, визначається властивостями квадратної матриці

$$M = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix}.$$

У цьому випадку матриця M є підматрицею

$$G = \left[I_{4\times4}\, M_{4\times4}\right] = \begin{pmatrix} 1 & 0 & 0 & 0 & 2 & 3 & 1 & 1 \\ 0 & 1 & 0 & 0 & 3 & 1 & 1 & 2 \\ 0 & 0 & 1 & 0 & 1 & 1 & 2 & 3 \\ 0 & 0 & 0 & 1 & 1 & 2 & 3 & 1 \end{pmatrix}$$

– породжувальної матриці для (8, 4, 5) лінійного блокового коду з максимальною кодовою відстанню (МКВ-Кодом) над $GF\left(2^8\right)$, що гарантує

$$B_{4,4}(\theta) = 5.$$

Для побудови динамічно змінюваного блоку лінійного розсіювання алгоритму ADE зі значенням, що $B_{k,k}(\theta)$ не уступає алгоритму AES використаний МКВ-Код – розширений $(256, 4i, 256-4i+1)$ код Ріда-Соломона над $GF(2^8)$ [67]. Шар лінійного розсіювання визначений над множиною векторів "$4i$ байт у $4i$ байт", $i \in \{1,2,...,32\}$. Іншими словами Mixcolumni алгоритму ADE реалізує перетворення i стовпців Стану$_i$:

при $i=1$ Mixcolumn$_1$ алгоритму ADE, як і алгоритму AES, обробляє по черзі по одному стовпцю з 4 байт Стану$_1$;

при $i=2$ Mixcolumn$_2$ алгоритму ADE обробляє по черзі по одному стовпцю з 8 байт Стану$_2$ і т.д.

Для реалізації на 32-розрядних і менш процесорах рекомендується використання Mixcolumn$_1$ з обробкою даних Стану$_1$. Для 32 i-х процесорів ($i>1$) рекомендується використання Mixcolumni з відображенням "$4i$ байт у $4i$ байт" Стану$_i$.

Розглянемо розширений $(256, 4i, 256-4i+1)$ код Ріда-Соломона над $GF(2^8)$. По визначенню це МКВ-Код, який забезпечує максимальна кодова відстань при заданих параметрах коду. Породжувальна матриця задається виразом:

$$G=\begin{pmatrix} '00' & '01' & '02' & ... & 'FF' \\ '00' & '01' & ('02')^2 & ... & ('FF')^2 \\ '00' & '01' & ('02')^3 & ... & ('FF')^3 \\ ... & ... & ... & ... & ... \\ '01' & '01' & ('02')^{4i} & ... & ('FF')^{4i} \end{pmatrix},$$

де зведення в ступінь проводиться в поле $GF(2^8)$, $i \in \{1, 2, ..., 32\}$.

Використовуючи вектор з одного байта раундового ключа, позначимо його через s (елемент поля $GF(2^8)$ порядку 255), сформуємо невироджену матрицю

$$M'=\begin{pmatrix} s & s^2 & s^3 & \dots & s^{4i} \\ s^2 & s^4 & s^6 & \dots & s^{8i} \\ s^3 & s^6 & s^9 & \dots & s^{12i} \\ \dots & \dots & \dots & \dots & \dots \\ s^{4i} & s^{8i} & s^{12i} & \dots & s^{16i} \end{pmatrix},$$

яка є підматрицею певної матриці, що вище породжує, $(256, 4i, 256-4i+1)$ розширеного коду Ріда-Соломона над $GF(2^8)$. Властивість невиродженості (оборотності M') забезпечується умовою максимального s порядку елемента в полі $GF(2^8)$. В цьому випадку всі стовпці та строки матриці M' різні та будь-яка підматриця також зворотня.

Розглянемо вектор $a=\{a_0 \quad a_1 \quad a_2 \quad \dots \quad a_{4i-1}\}$ і вектор $b=\{b_0 \quad b_1 \quad b_2 \quad \dots \quad b_{4i-1}\}$ з елементами з $GF(2^8)$. Функціональна відповідність

$$b=\theta'(a,s)=a\cdot M'$$

задає перетворення Mixcolumni за допомогою лінійного відображення $4i$-байтних векторів

$$\theta': \mathbf{GF}(2^8)^k \to \mathbf{GF}(2^8)^k,\ k=4i$$

з кількістю галузей лінійного розсіювання

$$B_{4,4}(\theta')=4i+1.$$

Наприклад, найпростіший випадок з $i=1$ відповідає матриці

$$M'=\begin{pmatrix} s & s^2 & s^3 & s^4 \\ s^2 & s^4 & s^6 & s^8 \\ s^3 & s^6 & s^9 & s^{12} \\ s^4 & s^8 & s^{12} & s^{16} \end{pmatrix},$$

що є підматрицею породжувальної матриці

$$G=\begin{pmatrix} '00' & '01' & '02' & ... & 'FF' \\ '00' & '01' & ('02')^2 & ... & ('FF')^2 \\ '00' & '01' & ('02')^3 & ... & ('FF')^3 \\ '01' & '01' & ('02')^4 & ... & ('FF')^4 \end{pmatrix}$$

розширеного (256, 4, 253) коду Ріда-Соломона над $GF(2^8)$.

Функціональна відповідність

$$b=\theta'(a,s)=a\cdot M'$$

задає перетворення $Mixcolumn_1$ за допомогою лінійного відображення 4-х байтних векторів

$$\theta':GF(2^8)^k\to GF(2^8)^k,\ k=4$$

з кількістю галузей лінійного розсіювання

$$B_{4,4}(\theta')=5.$$

Очевидно, що у випадку $M'=M$, розглянутий спосіб побудови динамічно змінюваного блоку лінійного розсіювання алгоритму ADE вироджується у відповідний блок алгоритму AES.

Випадок з $i=2$ відповідає матриці

$$M'=\begin{pmatrix} s & s^2 & s^3 & s^4 & s^5 & s^6 & s^7 & s^8 \\ s^2 & s^4 & s^6 & s^8 & s^{10} & s^{12} & s^{14} & s^{16} \\ s^3 & s^6 & s^9 & s^{12} & s^{15} & s^{18} & s^{21} & s^{24} \\ s^4 & s^8 & s^{12} & s^{16} & s^{20} & s^{24} & s^{28} & s^{32} \\ s^5 & s^{10} & s^{15} & s^{20} & s^{25} & s^{30} & s^{35} & s^{40} \\ s^6 & s^{12} & s^{18} & s^{24} & s^{30} & s^{36} & s^{42} & s^{48} \\ s^7 & s^{14} & s^{21} & s^{28} & s^{35} & s^{42} & s^{49} & s^{56} \\ s^8 & s^{16} & s^{24} & s^{32} & s^{40} & s^{48} & s^{56} & s^{64} \end{pmatrix},$$

що є підматрицею породжувальної матриці розширеного (256, 8, 249) коду Ріда-Соломона над $GF(2^8)$.

Функціональна відповідність

$$b = \theta'(a, s) = a \cdot M'$$

задає перетворення $Mixcolumn_2$ за допомогою лінійного відображення 8-ми байтних векторів

$$\theta' : GF(2^8)^k \to GF(2^8)^k,\ k = 8$$

с кількістю галузей лінійного розсіювання

$$B_{8,8}(\theta') = 9.$$

Випадок з $i = 3$ відповідає матриці M'_{12x12}, що є підматрицею породжувальної матриці розширеного (256, 12, 245) коду Ріда-Соломона над $GF(2^8)$.

Функціональна відповідність

$$b = \theta'(a, s) = a \cdot M'$$

задає перетворення Mixcolumn_3 за допомогою лінійного відображення 12-ти байтних векторів

$$\theta' : GF\left(2^8\right)^k \to GF\left(2^8\right)^k,\ k = 12$$

с кількістю галузей лінійного розсіювання

$$B_{12,12}(\theta') = 13.$$

Випадок з $i = 4$ відповідає матриці $M'_{16\times16}$, що є підматрицею породжувальної матриці розширеного (256, 16, 241) коду Ріда-Соломона над $GF\left(2^8\right)$.

Функціональна відповідність

$$b = \theta'(a, s) = a \cdot M'$$

задає перетворення Mixcolumn_4 за допомогою лінійного відображення 16-ти байтних векторів

$$\theta' : GF\left(2^8\right)^k \to GF\left(2^8\right)^k,\ k = 16$$

з кількістю галузей лінійного розсіювання

$$B_{16,16}(\theta') = 17,$$

і так далі.

Випадок з $i = 32$ відповідає матриці $M'_{128\times128}$, що є підматрицею породжувальної матриці розширеного (256, 128, 129) коду Ріда-Соломона над $GF\left(2^8\right)$.

Функціональна відповідність

$$b = \theta'(a, s) = a \cdot M'$$

задає перетворення Mixcolumn_{32} за допомогою лінійного відображення 128-ми байтних векторів

$$\theta' : \text{GF}\left(2^8\right)^k \to \text{GF}\left(2^8\right)^k,\ k = 128$$

з кількістю галузей лінійного розсіювання

$$\text{B}_{128,128}\left(\theta'\right) = 129.$$

На рис. 2.6, 2.7 проілюстрований ефект застосування перетворення Mixcolumni до Стану_i. Інверсія перетворення Mixcolumni реалізується множенням стовпця Стану_i на матрицю $\left(\text{M}'\right)^{-1}$ зворотну до матриці M′ з операціями над $\text{GF}\left(2^8\right)$.

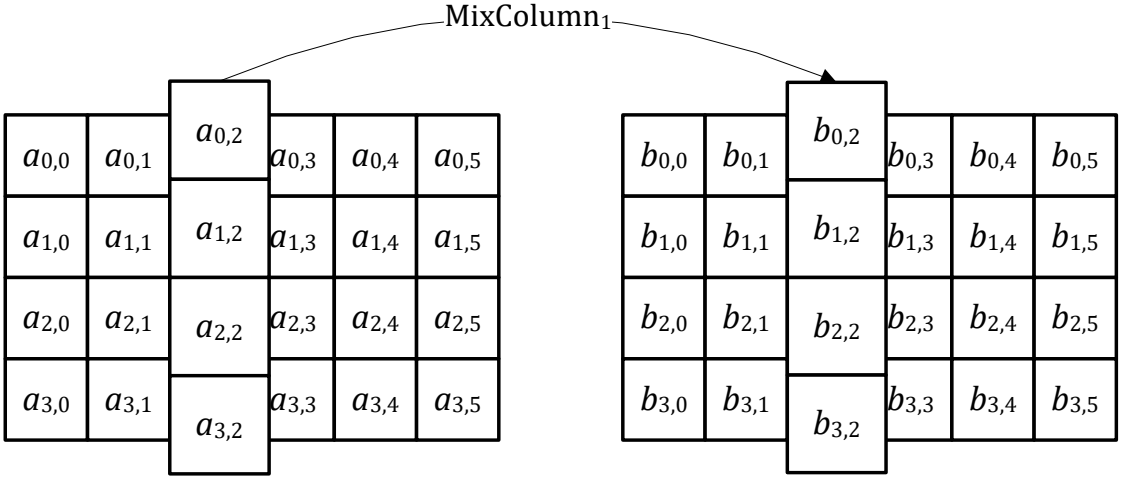

Рисунок 2.6 - Mixcolumn_1 оперує зі стовпцями Стану_1 по 4 байта

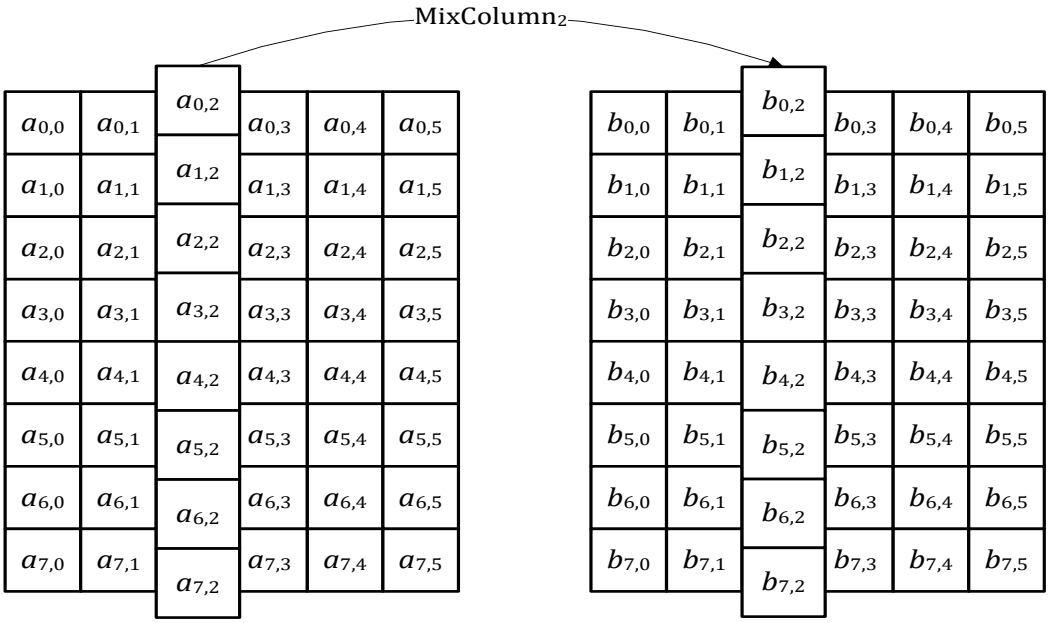

Рисунок 2.7 - Mixcolumn$_2$ оперує зі стовпцями Стану$_2$ по 8 байт

2.3.4 Дослідження динамічно керованого лінійного перетворення Addroundkey

Операція Addroundkey алгоритму ADE виконується аналогічно операції алгоритму AES. В цій операції до Стану$_i$ застосовується перетворення Roundkey за допомогою простого побітного додавання по модулю 2. Раундовий ключ формується із ключа шифру за допомогою процедури формування ключів. Довжина раундового ключа дорівнює довжині блоку Nb.

Перетворення, яке складається з побітового додавання Стану$_i$ і раундового ключа позначається через Addroundkey(State,Roundkey).

Для прикладу, це перетворення над Станом$_1$ проілюстроване на рис.2.8.

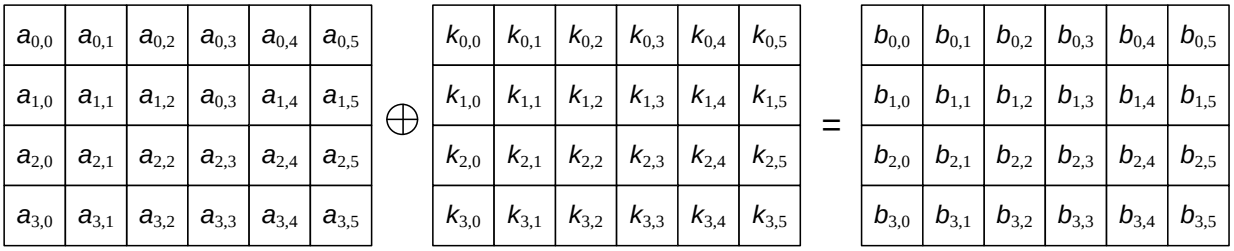

Рисунок 2.8 - Перетворення над станом$_1$

2.4 Обґрунтування процедури формування та розширення ключів

Процедура формування ключів алгоритму ADE визначена аналогічно операції формування ключів алгоритму AES. Раундові ключі формуються із ключа шифру за допомогою процедури формування ключів. Вона містить у собі два компоненти: розширення ключа та вибір раундового ключа. Основний принцип полягає в наступному: загальна кількість біт раундового ключа дорівнює довжині блоку, помноженої на кількість раундів плюс 1 (наприклад, для Стану$_1$ при довжині блоку 128 біт і 10 раундів необхідно 1408 раундових ключів); ключ шифру розширюється в розширений ключ. Раундові ключі беруться з розширеного ключа в такий спосіб: перший раундовий ключ складається з перших Nb слів, другий – з наступних Nb слів, і так далі.

Розширений ключ є одномірним масивом $4i$-байтних слів і позначається як $W(Nb(Nr+1))$ [40]. Перші Nk слів містять ключ шифру. Кожне наступне слово $W[i]$ дорівнює сумі за модулем 2 зі словом $W[i-1]$ і словом, розташованим на Nk позиції раніше $W\lfloor i-Nk\rfloor$. Для слів з позиціями, кратними Nk, перед додаванням за модулем 2 до слова $W\lfloor i-1\rfloor$ застосовується перетворення та додавання за модулем 2 з раундовою константою. Це перетворення складається із циклічного зсуву байт у слові, за яким треба застосовувати табличний пошук до всіх байтів слова.

Раундові константи не залежать від Nk і визначені як:

$$Rcon[i] = (RC[i],'00','00','00'),$$

де $RC[i]$ є елементом поля $GF(2^8)$ зі значенням x^{i-1}, такий, що:

$$RC[1] = 1 \ (\text{т. ч. } '01')$$
$$RC[i] = x \ (\text{т. ч. } '02') \bullet (RC[i-1]) = x^{(i-1)}$$

Раундовий ключ i заданий буфером слів раундового ключа від $W[Nb*i]$ до $W[Nb*(i+1)]$. Це проілюстровано на рис. 2.9.

Примітка: для реалізацій з обмеженням по оперативній пам'яті, раундові ключі можуть бути розраховані швидко з використанням буфера Nk слів майже без обчислювального надлишку.

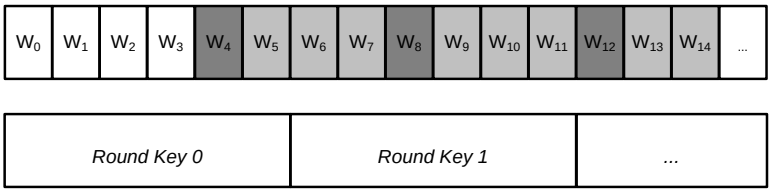

Рисунок 2.9 - Розширення ключа і вибір раундового ключа

Процес виконання шифру ADE, як і AES, складається з початкового додавання раундового ключа, $Nr-1$ раундів та остаточного раунду.

Розширення ключа може бути виконане заздалегідь і ADE може бути визначений у значеннях розширеного ключа.

2.5 Висновки за другим розділом

Таким чином, характерною рисою методів симетричного криптографічного перетворення інформації є той факт, що стійкість перетворень у цілому здійснюється за рахунок використання нелінійних вузлів замін. Тому першорядне значення для підвищення ефективності симетричних криптографічних засобів захисту інформації здобуває розробка математичних моделей і обчислювальних методів формування нелінійних вузлів замін з поліпшеними властивостями.

В рамках роботи розглянутий алгоритм ADE (Algorithm Of Dynamic Encryption). Основна увага при розробці ADE приділена можливості динамічно управляти процесом лінійного розсіювання та нелінійної заміни в ході криптографічного перетворення інформації. Фактично, формовані системи рівнянь мають вигляд випадкових і щільних систем, що суттєво ускладнює завдання криптоаналізу, зокрема, алгебраїчними методами.

РОЗДІЛ 3 РОЗРОБЛЕННЯ ПРАКТИЧНИХ РЕКОМЕНДАЦІЙ З РЕАЛІЗАЦІЇ КРИПТОАЛГОРИТМА ADE І ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ ЙОГО ПРОДУКТИВНОСТІ

3.1 Розроблення практичних рекомендацій з реалізації розробленого криптоалгоритму

3.1.1 Розроблення практичних рекомендацій з реалізації розробленого криптоалгоритма на 32-х бітних процесорах

Шифр ADE, як і AES, пристосований для ефективної реалізації на широкому спектрі процесорів і спеціалізованому апаратному забезпеченні. Різні кроки раундового перетворення алгоритму ADE, як і алгоритму AES, можуть бути скомбіновані для забезпечення ефективної реалізації на процесорах з довжиною слова 32 або вище. При реалізації чотирьох вибіркових таблиць реалізація алгоритму ADE на 32-х бітних процесорах не уступає по швидкодії алгоритму AES.

Виразимо один стовпець Стану$_1$ виходу раунду e у величинах байт входу раунду a. В цьому розділі $a_{i,j}$ позначає байт у рядку i та стовпці j, a_j позначає стовпець j Стану$_1$ a. Розглянемо раундове перетворення над Станом$_1$ [40, 64-66]. Реалізація додатка ключів і перетворення Mixcolumn$_1$ з матрицею лінійного розсіювання

$$M' = \begin{pmatrix} s & s^2 & s^3 & s^4 \\ s^2 & s^4 & s^6 & s^8 \\ s^3 & s^6 & s^9 & s^{12} \\ s^4 & s^8 & s^{12} & s^{16} \end{pmatrix}$$

аналітично запишеться у вигляді:

$$\begin{bmatrix} e_{0,j} \\ e_{1,j} \\ e_{2,j} \\ e_{3,j} \end{bmatrix} = \begin{bmatrix} d_{0,j} \\ d_{1,j} \\ d_{2,j} \\ d_{3,j} \end{bmatrix} \oplus \begin{bmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{bmatrix},$$

де

$$\begin{bmatrix} d_{0,j} \\ d_{1,j} \\ d_{2,j} \\ d_{3,j} \end{bmatrix} = [M] \begin{bmatrix} c_{0,j} \\ c_{1,j} \\ c_{2,j} \\ c_{3,j} \end{bmatrix}.$$

Для перетворень Shiftrows і Bytesub маємо:

$$\begin{bmatrix} c_{0,j} \\ c_{1,j} \\ c_{2,j} \\ c_{3,j} \end{bmatrix} = \begin{bmatrix} b_{0,j-C0} \\ b_{1,j-C1} \\ b_{2,j-C2} \\ b_{3,j-C3} \end{bmatrix},$$

де

$$b_{i,j} = S[a_{i,j}, \gamma],$$

γ – байт раундового ключа, використовуваний для формування таблиці замін.

В цьому виразі індекси стовпців повинні бути взяті по модулю Nb. За допомогою заміни, наведені вище виразу, можуть бути скомбіновані в:

$$\begin{bmatrix} e_{0,j} \\ e_{1,j} \\ e_{2,j} \\ e_{3,j} \end{bmatrix} = [M] \begin{bmatrix} S[a_{0,j-C0}, \gamma] \\ S[a_{1,j-C1}, \gamma] \\ S[a_{2,j-C2}, \gamma] \\ S[a_{3,j-C3}, \gamma] \end{bmatrix} \oplus \begin{bmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{bmatrix}.$$

Матриця

$$M'=\begin{pmatrix} s & s^2 & s^3 & s^4 \\ s^2 & s^4 & s^6 & s^8 \\ s^3 & s^6 & s^9 & s^{12} \\ s^4 & s^8 & s^{12} & s^{16} \end{pmatrix}$$

алгоритму ADE формується за допомогою одного байта раундового ключа *s*, за допомогою формування підматриці породжувальної матриці (256, 4, 253) коду Ріда-Соломона над $GF(2^8)$. Таким чином, матричне множення може бути виражене як лінійна комбінація векторів:

$$\begin{bmatrix} e_{0,j} \\ e_{1,j} \\ e_{2,j} \\ e_{3,j} \end{bmatrix} = S[a_{0,j-C0},\gamma]\begin{bmatrix} s \\ s^2 \\ s^3 \\ s^4 \end{bmatrix} \oplus S[a_{1,j-C1},\gamma]\begin{bmatrix} s^2 \\ s^4 \\ s^6 \\ s^8 \end{bmatrix} \oplus$$

$$\oplus S[a_{2,j-C2},\gamma]\begin{bmatrix} s^3 \\ s^6 \\ s^9 \\ s^{12} \end{bmatrix} \oplus S[a_{3,j-C3},\gamma]\begin{bmatrix} s^4 \\ s^8 \\ s^{12} \\ s^{16} \end{bmatrix} \oplus \begin{bmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{bmatrix}.$$

Множники $S[a_{i,j},\gamma,s]$ чотирьох векторів отримані складанням вибіркової таблиці на входах байт $a_{i,j}$ і байта раундового ключа γ в таблиці S-блоку.

Визначимо таблиці T_0, T_1, T_2, T_3:

$$\text{Ò}_0[a,\gamma,s] = S[a_{0,j-C0},\gamma]\begin{bmatrix} s \\ s^2 \\ s^3 \\ s^4 \end{bmatrix}, \text{Ò}_1[a,\gamma,s] = S[a_{0,j-C0},\gamma]\begin{bmatrix} s^2 \\ s^4 \\ s^6 \\ s^8 \end{bmatrix},$$

$$\text{Ò}_2[a,\gamma,s] = S[a_{0,j-C0},\gamma]\begin{bmatrix} s^3 \\ s^6 \\ s^9 \\ s^{12} \end{bmatrix}, \text{Ò}_3[a,\gamma,s] = S[a_{0,j-C0},\gamma]\begin{bmatrix} s^4 \\ s^8 \\ s^{12} \\ s^{16} \end{bmatrix}.$$

Для кожного раунду алгоритму ADE використовуються свої значення $\gamma, s,$ $C0, C1, C2, C3$. Отже, T_0, T_1, T_2, T_3 – це 4 таблиці (для кожного раунду свої) з 256-ю введеннями 4-байтових слів, що займають до 4 Кбайт пам'яті.

Використовуючи ці таблиці, раундове перетворення може бути виражене як:

$$e_j = T_0[a_{0,j-Ñ0}, \gamma, s] \oplus T_1[a_{1,j-C1}, \gamma, s] \oplus T_2[a_{2,j-C2}, \gamma, s] \oplus T_3[a_{3,j-C3}, \gamma, s] \oplus k_j.$$

Отже, реалізація вибіркових таблиць із 4 Кбайтами пам'яті вимагає тільки 4 вибіркові таблиці та 4 додавання по модулю 2 к стовпцю кожного раунду.

Розмір коду може бути зроблений невеликим за допомогою включення коду для генерації таблиць замість самих таблиць.

3.1.2 Розроблення практичних рекомендацій реалізації інверсного шифру розробленого криптоалгоритму

Аспект реалізації інверсного шифру передбачений його конструкцією. Застосування табличного множення і збереження в пам'яті таблиць замін однаково прискорює як пряме, так і інверсне перетворення.

Інверсія раунду задається так:

```
Function InvRound(State,RoundKey)
{
     AddRoundKey(State,RoundKey);
     InvMixColumn(State,RoundKey);
     InvShiftRows(State,RoundKey);
     InvByteSub(State,RoundKey);
}
```

Інверсія кінцевого раунду задається так:

```
Function InvFinalRound(State,RoundKey)
{
     AddRoundKey(State,RoundKey);
     InvShiftRows(State,RoundKey);
     InvByteSub(State,RoundKey);
}
```

Інверсія двохраундного варіанту ADE складається з інверсії кінцевого раунду, за якою слідує інверсія раунду і накладення раундового ключа:

```
AddRoundKey(State,ExpandedKey+2*Nb);
InvShiftRows(State,RoundKey);
InvByteSub(State,RoundKey);
AddRoundKey(State,ExpandedKey+Nb);
InvMixColumn(State,RoundKey);
InvShiftRows(State,RoundKey);
InvByteSub(State,RoundKey);
```

```
AddRoundKey(State,ExpandedKey);
```

У виробництві еквівалентної структури інверсного шифру використані дві властивості компонентного перетворення.

Перше, порядок ShiftRows і Subbyte не має значення. ShiftRows просто переміщає байти і не впливає на їхні значення. Subbyte оперує з окремими байтами, незалежно від їх розташування.

Друге, послідовність

```
AddRoundKey(State,RoundKey);
InvMixColumn(State,RoundKey);
```

може бути замінена на

```
InvMixColumn(State,RoundKey);
AddRoundKey(State,InvRoundKey);
```

де InvRoundKey отримано застосуванням InvMixColumn на відповідний раундовий ключ.

Це базується на факті, що для лінійного перетворення ми маємо

$$A(x + k) = A(x) + A(k).$$

Використовуючи описані вище властивості, інверсія двохраундного шифру ADE може бути перетворена в:

```
AddRoundKey(State,ExpandedKey+2*Nb);
InvByteSub(State,RoundKey);
InvShiftRow(State,RoundKey);
InvMixColumn(State,RoundKey);
AddRoundKey(State,I_ExpandedKey+Nb);
InvByteSub(State,RoundKey);
InvShiftRow(State,RoundKey);
AddRoundKey(State,ExpandedKey);
```

Можна побачити, що маємо знову початкове накладення раундового ключа, раунду і кінцевого раунду. Раунд і кінцевий раунд мають таку ж структуру, як і в самому шифрі. Це може бути узагальнене для будь-якої кількості раундів.

Визначаємо раунд і кінцевий раунд як наступне:

```
Function I_Round(State,I_RoundKey)
{
        InvByteSub(State,RoundKey);
        InvShiftRow(State,RoundKey);
        InvMixColumn(State,RoundKey);
        AddRoundKey(State,I_RoundKey);
}
Function I_FinalRound(State,I_RoundKey)
{
        InvByteSub(State,RoundKey);
        InvShiftRow(State,RoundKey);
        AddRoundKey(State,RoundKey0);
}
```

Інверсія шифру ADE може бути виражена таким чином:

```
Function I_Rijndael(State,CipherKey)
{
        I_KeyExpansion(CipherKey,I_ExpandedKey) ;
        AddRoundKey(State,I_ExpandedKey+ Nb*Nr);
        For( i=Nr-1 ; i>0 ; i-- )
        Round(State,I_ExpandedKey+ Nb*i) ;
        FinalRound(State,I_ExpandedKey);
}
```

Розширення ключа інверсного шифру визначається наступним чином:

1) застосувати розширення ключа;

2) застосувати InvMixColumn до всіх раундових ключів, за винятком першого і останнього.

На псевдо-Сі це виглядає так:

```
Function I_KeyExpansion(CipherKey,I_ExpandedKey)
{
        KeyExpansion(CipherKey,I_ExpandedKey);
        for( i=1 ; i < Nr ; i++ )
                InvMixColumn(I_ExpandedKey + Nb*i) ;
}
```

Для алгоритму AES вибір многочлена для перетворення MixColumn і розширення ключа частково базувалося на аргументах продуктивності шифру. Спостерігається зниження продуктивності AES на 8-бітних процесорах.

Для алгоритму ADE перетворення MixColumn побудовано без будь-яких міркувань щодо особливостей множення на фіксований многочлен над кільцем по модулю x^4+1. Отже, прямий і зворотній шифр алгоритму ADE еквівалентні по складності реалізації.

У порівнянні з алгоритмом AES пряме шифрування ADE менш ефективне за швидкодією. Для AES перетворення MixColumn реалізується зі спеціально підібраним многочленом (з коефіцієнтами “01”, “02” та “03”). Інверсний шифр AES трохи повільніший за рахунок коефіцієнтів “09”, “0E” і “0D”. Для алгоритму ADE як пряме, так і інверсне шифрування реалізується за допомогою множення на многочлен з коефіцієнтами з $GF(2^8)$.

Операція розширення ключа, що генерує W, визначена таким чином, що можна також почати з останніх Nk слів інформації раундового ключа і розгорнути їх у зворотному порядку щодо оригінального ключа шифру. Таким чином, обчислення раундового ключа “на льоту”, починаючи з “інверсного ключа шифру” є можливим.

Розширення ключа для інверсного шифру трохи повільніше, тому що після розширення ключа тільки два раундових ключа піддаються перетворенню InvMixColumn.

Так як шифр і його інверсія використовують різні перетворення, схема, яка реалізує ADE, не підтримує автоматичне обчислення інверсії ADE. У схемі, що реалізує ADE і його інверсію, частини схеми можуть бути використані для обох функцій.

Це справедливо, наприклад, у випадку нелінійних шарів.

S – блок, сконструйований з двох відображень:

$$S(x) = f(g(x\gamma)),$$

де $g(x\gamma)$ – це відображення:

$$x \Rightarrow (x\gamma)^{-1} \text{ в } GF(2^8).$$

Функція $f(x)$ є афінним відображенням.

Відображення $g(x)$ є своєю інверсією, отже

$$S^{-1}(x\gamma) = g^{-1}(f^{-1}(x\gamma)) = g(f^{-1}(x\gamma)).$$

Тому, якщо необхідно S і S^{-1}, необхідно реалізувати тільки g, f і f^{-1}. Оскільки f і f^{-1} є елементарними функціями на двійковому рівні, їх апаратна реалізація може бути виконана значно простіше у порівнянні з наявністю двох повних S-блоків.

Подібні аргументи застосовуються до повторного використання перетворення xtime в шарі розсіювання.

3.1.3 Розробка інтерфейсу програми ADE

Для реалізації алгоритму був обраний язик програмування Делфі. Лістинг програми наведений в дод.Б. Загальний вигляд програми ADE наведений на рис.3.1.

Робота з програмою є певною послідовністю дій, що необхідно виконати, а саме необхідно:

1) сформувати файл відкритого тексту (рис. 3.2);

2) у вікні “Файл відкритого тексту” задати ім’я файлу, який необхідно зашифрувати (може мати довільну довжину> 0); файл повинен розташовуватися в каталозі ENC;

3) у вікні “Файл ключа” задати ім’я файлу, що містить 16-ти байтний ключ шифрування (32 шістнадцяткові цифри); файл повинен розташовуватися в каталозі KEY;

4) у вікні “Файл закритого тексту” задати ім’я файлу, в який буде записана зашифрована інформація (рис. 3.3); файл буде розміщений в каталозі INT;

5) обрати операцію “Шифрування” і запустити програму, натисканням кнопки з зображенням ключа.

6) у вікні “Файл розшифрованого тексту” задати ім’я файлу, в який буде поміщений результат розшифрування; файл буде розміщений в каталозі DEC;

7) обрати операцію “Розшифрування” і запустити програму натисканням кнопки з зображенням ключа.

8) порівняти файл відкритого тексту (рис. 3.2) з розшифрованим (рис. 3.4) для того, щоб упевнитися в коректній роботі програми.

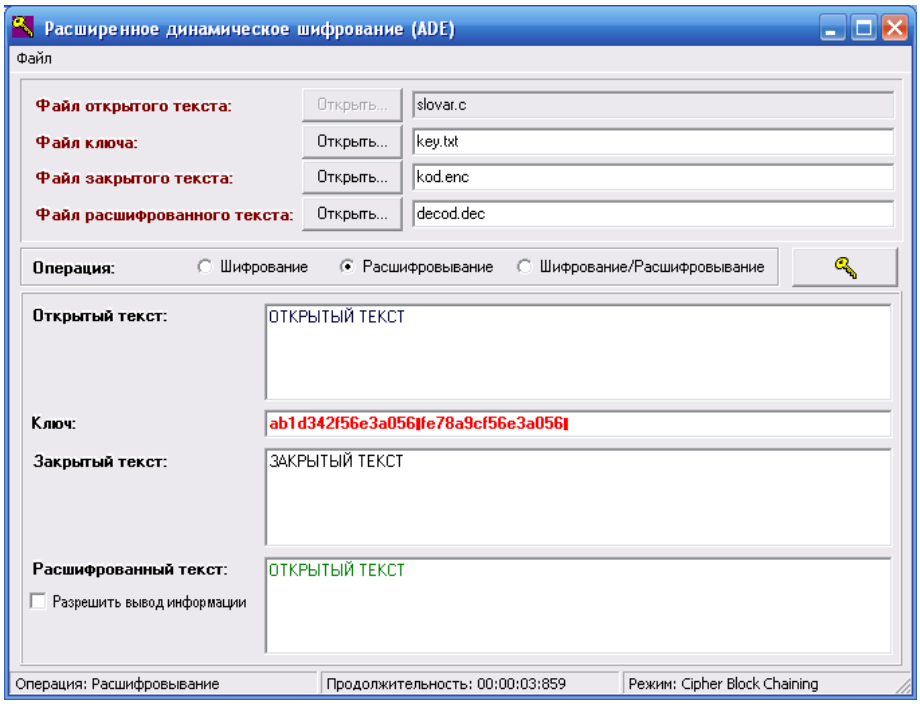

Рисунок 3.1 - Загальний вигляду програми ADE

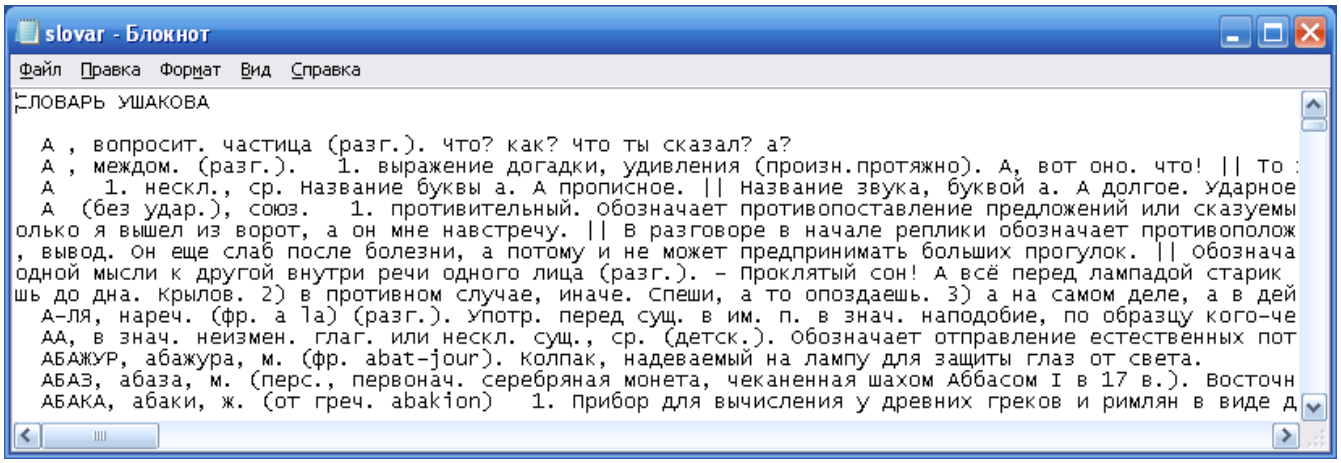

Рисунок 3.2 - Файл відкритого тексту

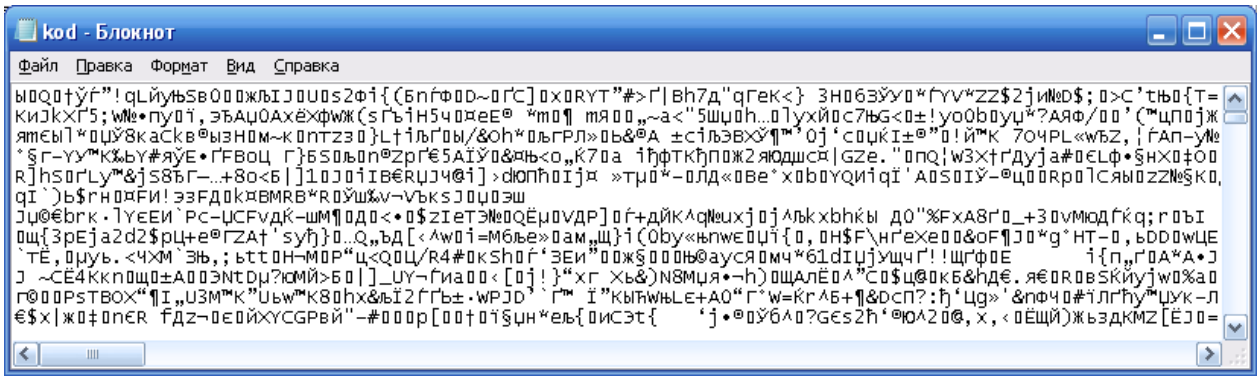

Рисунок 3.3 - Файл закритого тексту

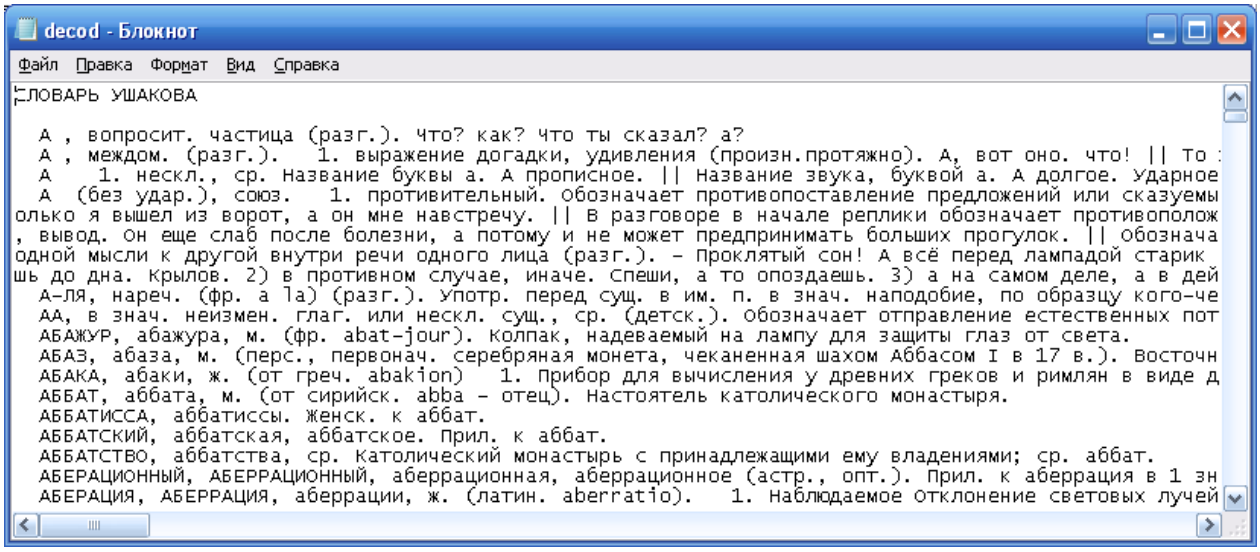

Рисунок 3.4 - Файл розшифрованого тексту

3.2 Дослідження статистичної безпеки алгоритму ADE

3.2.1 Методика дослідження статистичної безпеки алгоритму ADE

Однією з складових оцінки стійкості криптопримітивів є їх оцінка статистичної безпеки. Примітив статистично безпечний, якщо генеруєма послідовність, не поступається випадковій послідовності; такі послідовності називаються “псевдовипадковими” [14, 27, 37, 38, 52, 59]. Запропонований пакет тестів NIST STS для тестування ГПВЧ є одним з підходів реалізації задачі оцінки статистичної безпеки криптографічних примітивів. Використання даного пакету показує, наскільки послідовність, що генерується досліджуваним примітивом, є статистично безпечною.

Під час конкурсу на новий стандарт блочного шифрування був запропонований набір тестів NIST STS, які можуть використовуватись для досліджень статистичних властивостей алгоритмів-кандидатів. На сьогодні запропонована NIST методика тестування є найпоширенішою в розроблювачів криптографічних засобів захисту інформації [48 - 50]. Порядок тестування окремої двійкової послідовності S має такий вигляд [39]:

– висувається нульова гіпотеза H_0 - припущення про те, що дана двійкова послідовність S випадкова;

– за послідовністю S розраховується статистика тесту $c(S)$;

– з використанням спеціальної функції та статистики тесту розраховується значення ймовірності $P = f(c(S)),\ P \in [0,1]$.

Значення ймовірності P порівнюється з рівнем значимості, $\alpha \in [0.001, 0.01]$. Якщо $P \geq \alpha$, то гіпотеза H_0 ухвалюється. А якщо ні, то ухвалюється альтернативна гіпотеза.

Пакет містить у собі 16 статистичних тестів. Але фактично, залежно від вхідних параметрів обчислюється 189 значень імовірності P, які можна розглядати як результат роботи окремих тестів.

В табл. 3.1 наводяться зібрані дані за всіма тестами із вказівкою фізичного змісту статистики тесту та дефекту, на виявлення якого спрямований тест.

Таблиця 3.1

Набір статистичних тестів NIST STS

№ з/п	Статистичний тест	Статистика тесту c(S)	Дефект, що виявляється
1	2	3	4
1	Частотний (монобітний тест)	Нормалізована абсолютна сума значень елементів послідовності	Надто багато нулів або одиниць у послідовності
2	Частотний тест (в середині блоку)	Міра узгодженості кількості одиниць, що спостерігаються із тим, що очікується теоретично.	Локалізовані відхилення частоти появи одиниць в блоці від ідеального значення ½
3	Перевірка накопичених сум	Максимальне відхилення значень накопиченої суми елементів послідовності від початкової точки відліку (точка 0)	Велика кількість одиниць або нулів на початку або наприкінці двійкової послідовності
4	Перевірка серій	Загальна кількість серій на всій довжині послідовності	Надто швидка або надто повільна зміна знаку в ході генерації послідовності
5	Перевірка максимальної довжини серії в блоці.	Міра узгодженості значень максимальної довжини, що спостерігаються, із значенням, що очікується теоретично	Відхилення від теоретичного закону розподілення максимальних довжин серій одиниць.
6	Перевірка рангу двійкової матриці	Міра узгодженості значення рангів різного порядку, що спостерігаються, із значенням, що очікується теоретично	Відхилення емпіричного закону розподілення значень рангів матриць від теоретичного, що вказує на залежність символів у послідовності.
7	Спектральний аналіз на основі дискретного перетворення Фур'є	Нормалізована різниця кількості частотних компонентів, що спостерігаються та очікуються, які перевищують 95% рівень порогу.	Виявлення періодичних складових (трендів) у двійковій послідовності.

Продовження табл.3.1

1	2	3	4
8	Перевірка шаблонів, що перекриваються	Міра узгодженості кількості спостерігаємих шаблонів, що перекриваються, у послідовності із теоретичним значенням.	Велика кількість m-бітних серій з одиниць в послідовності.
9	Універсальний тест Маурера	Сума логарифму відстані між l –бітними шаблонами.	Можливість стиснення послідовності.
10	Ентропійний тест	Міра узгодженості спостерігаємого значення ентропії джерела із тим, що теоретично очікується для випадкового джерела.	Нерівномірність розподілення m-бітних слів в послідовності (регулярність властивостей джерела)
11	Перевірка випадкових відхилень	Міра узгодженості спостерігаємої кількості візитів при випадковому блуканні в заданий стан в середині циклу із тим, що очікується теоретично	Відхилення від теоретичного закону розподілення візитів у конкретний стан при випадковому блуканні
12	Перевірка випадкових відхилень (варіант)	Загальна кількість візитів при випадковому блуканні	Відхилення від теоретично очікуємої загальної кількості візитів при випадковому блуканні в заданий стан.
13	Послідовний тест	Міра узгодженості спостерігаємої кількості усіх варіантів m-бітних шаблонів, що зустрілись із тією, що очікується теоретично.	Нерівномірність розподілення m-бітних слів у послідовності.
14	Перевірка стиснення згідно алгоритму Лемпеля-Зіва	Кількість в послідовності різних слів	Великий ступінь стиснення послідовності, що тестується в зрівнянні з степенем стиснення, що очікується в випадковій послідовності.

Закінчення табл.3.1

1	2	3	4
15	Перевірка шаблонів, що не перекриваються	Міра узгодженості спостерігаємої кількості неперіодичних шаблонів в послідовності із теоретичним значенням.	Велика кількість заданих неперіодичних шаблонів в послідовності.
16	Перевірка лінійної складності	Міра узгодженості спостерігаємої кількості подій, що полягають у появі фіксованої довжини еквівалентного	Відхилення емпіричного розподілу довжин еквівалентних ЛРР для послідовностей фіксованої довжини від теоретичного
17		ЛРР для заданого блоку із теоретичним	закону розподілення для випадкової послідовності, що вказує на недостатню складність послідовності, що тестується.

Отже, в результаті тестування формується вектор значень імовірності $P = \{P_1, P_2 ... P_{189}\}$. Аналіз складових P_i цього вектору вказує на дефекти послідовності, що підлягає тестуванню.

Відповідно до NIST STS рішення на статистичне тестування ухвалюється якщо виконуються правила:

1: пройшло тестування по всім q тестам, ($q = \overline{1,189}$), значення коефіцієнту rj в межах довірчого інтервалу $[0.96,1.00]$;

2: пройшло тестування по всім q тестам, ($q = \overline{1,189}$), для всіх тестів за критерієм χ^2-Пірсона виконується умова $P(\chi^2) > 0{,}0001$.

3.2.2 Результати дослідження статистичної безпеки алгоритмів ADE і AES

Для тестування алгоритмів ADE і AES по методиці NIST STS вибрано параметри:

довжина послідовності n = 10^6 біт;

кількість послідовностей m = 100. Обсяг вибірки тестування склав N = $10^6 \times 100 = 10^8$ біт;

рівень значимості a = 0.01;

кількість тестів q = 189.

Для формування послідовності використовувався шифрофайл з текстом словника великоруської мови (В. Даля). Файл відкритого тексту “slovar.txt”, розмір 17390588 байт. Імена файлів шифротексту:

при використанні алгоритму ADE - “file_ADE.enc”, розмір 17390604 байт;

при використанні алгоритму AES - “file_AES.enc”, розмір 17390604 байт.

Результати тестування алгоритмів AES і ADE наведені на рис.3.5–3.6 відповідно.

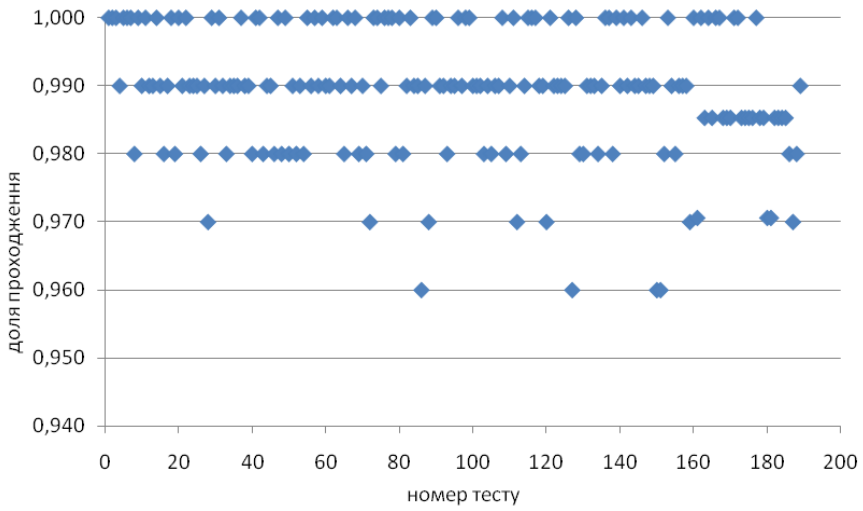

Рисунок 3.5 - Статистиний портрет алгоритму AES

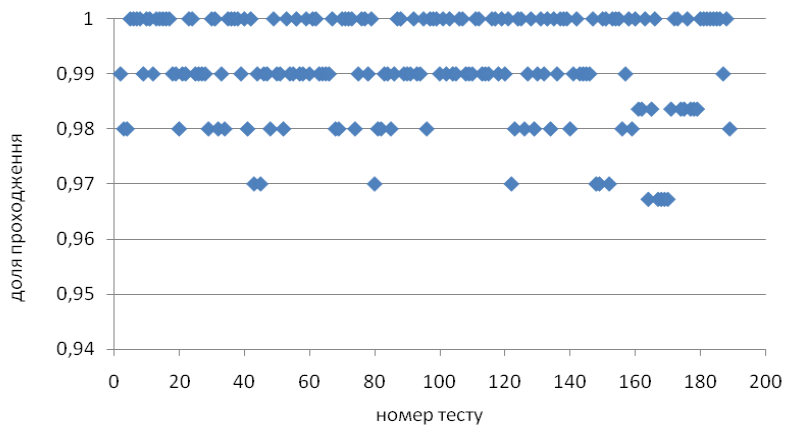

Рисунок 3.6 - Статистичний портрет програмної реалізації алгоритму ADE

В табл. 3.2 наведені підсумкові значення результатів тестування за методикою NIST STS для алгоритмів ADE і Rijndael, повні дані наведені в дод.А.

Таблиця 3.2 - Результати досліджень статистичної безпеки ADE і AES

Генератор	Кількість тестів, у яких позитивне тестування пройшло більше 99% послідовностей	Кількість тестів, у яких позитивне тестування пройшло більше 96% послідовностей	Кількість тестів, у яких позитивне тестування пройшло менше 96% послідовностей
AES (Rijndael)	129 (68,3%)	189 (100%)	0(0%)
ADE	149 (78,8%)	189 (100%)	0 (0%)

Аналіз експериментальних результатів статистичного тестування алгоритмів AES і ADE показує, що 78,8% тестів, з позитивним тестуванням пройшло більше 99% послідовностей, отримано алгоритмом ADE. Це є кращий результат в порівнянні з AES. Доцільно використання алгоритму ADE в банківській та інших системах.

3.3 Розроблення пропозицій застосування ADE в банківських системах

В Україні є фундамент побудови ефективної платіжної банківської системи. Основу складає механізм переведення коштів, який має бути захищеним, забезпечувати низький рівень шахрайства та витрат банків і платіжних системи, та сприяти підвищенню рівня довіри до галузі.

Економічна життєздатність важлива для платіжних систем у зв’язку з їх масштабністю і роллю, яку вони відіграють у сучасній економіці. Ці системи потребують високого ступеня довіри, заснованого на переконанні учасників ринку в захищеності банківських систем, застосування відкритих міжнародних стандартів, що дозволяють знизити витрати, і справедливої фінансово-економічної моделі, що стимулює активність користувачів.

Платіжні системи (ПС), що працюють в Україні, повинні ґрунтуватися на правилах, що дозволяють гарантувати її безперебійне, узгоджене і прозоре функціонування. Крім того, такі системи повинні мати надійну, високошвидкісну, захищену інфраструктуру. Від цих найважливіших чинників залежить ефективність платіжної системи.

ПС виконує п’ять основних функцій, пов’язаних з обробкою платежів (рис. 3.7). Для максимальної економічної віддачі від використання учасниками процесу, платіжна інфраструктура повинна забезпечувати надання додаткових процесінгових послуг, виконання функції керування даними та застосування досконалих механізмів захисту.

Рисунок 3.7 - Функції обробки платежів платіжної системи

Для функціонування ПС необхідним є забезпечення конфіденційності, достовірності та цілісності даних. Це допомагає нарощувати обсяги успішних операцій і в подальшому відіграють важливу роль в підтримці надійної, захищеної від втрати даних, високошвидкісної процесингової системи.

Захищеність - важливий чинник. Без застосування процедур і технологій боротьби з шахрайством, масштаб цього явища буде рости, у результаті знижується рівень довіри до ПС та активність користувачів пластикових карт.

З метою підвищення безпеки світова карткова спільнота використовує мікропроцесорні картки відповідно до стандарту EMV, які повинні витіснити менш захищені карти з магнітною смугою. EMV - міжнародний стандарт для кредитних і дебетних платежів, заснований на технології чіпових карт [22]. EMV

складається з наборів специфікацій і тестових процедур для платіжних карт і устаткування самообслуговування (POS-терміналів і банкоматів). Основними завданнями EMV є забезпечення глобальної взаємодії платіжних мереж, стандартизація вимог до встаткування прийому платежів з використанням чіпових карт, а також підвищення безпеки цих платежів до міжнародних вимог.

Найбільш широко застосовуються ATM-протоколами, які використовуються для передачі даних між банкоматом і ATM-контролером, є протоколи NDC + (NCR Direct Connection) і D912, відомий також як DDC (Diebold Direct Connection), розроблені, відповідно, NCR та Diebold. NCR і Diebold вдалося на базі протоколів-монополістів NDC + і D912 реалізувати можливість прийому EMV-карток [19]. NCR випустив версію протоколу NDC + 7.0, в якій крім функцій прийому EMV-карток забезпечена можливість використання алгоритму шифрування 3DES, віддаленого управління сесійними ключами і т.д. На думку експертів, ці протоколи вже майже вичерпали свої можливості для розширення функціональності. NCR і ACI WorldWide оголосили про співпрацю в області розробки нового протоколу, який отримав назву IFX.

Перехід на використання EMV-карток не гарантує безпеки платежів. EMV-картки важче підробити, ніж магнітні аналоги. Використання чіпових карт дозволяє емітенту говорити про те, що його користувачі захищені від атак шахраїв.

Однак процес EMV-міграції відбувається нерівномірно в різних регіонах. Проблема підготовки термінальної мережі для прийому EMV-карток актуальна на сьогодні. Наразі комбіновані картки обслуговуватимуться в POS-терміналах без використання мікропроцесорної технології, шляхом зчитування магнітної смуги. Отже, проблема підробки карти і забезпечення збереження ПІН-коду буде залишатися гострою навіть для країн з досить високими показниками EMV-міграції. Як відомо, передача ПІН-блоку між терміналом і хостовою системою банку здійснюється в зашифрованому вигляді. На даний момент у більшості випадків проблема захищеності ПІН-коду вирішується застосуванням алгоритму шифрування DES.

DES не в змозі забезпечити рівень захищеності карткових транзакцій. Сьогодні експерти стверджують, що за 1-2 дні зловмисник зламує захист DES на домашньому комп'ютері [14]. Так, у червні 2002р. Американський комітет з стандартизації методів забезпечення інформаційної безпеки ANSI Accredited Standarts Committee X9 опублікував звіт, в якому зазначалося, що алгоритм DES не здатний підтримувати захищеність транзакційної інформації.

Міжнародні платіжні асоціації оперативно відреагували на сигнали експертів і розробили план міграції з DES на застосування безпечного алгоритму 3DES (Triple DES). Спорідненість алгоритмів DES і 3DES зробила вибір на користь останнього. Учасники ринку можуть мігрувати на використання спорідненого алгоритму шифрування з найменшими фінансовими та тимчасовими витратами. Алгоритм 3DES значно перевершує попередній аналог за стійкістю до злому. Процес міграції на 3DES торкнувся банкоматних мереж.

За даними експертів, більшість банкоматів в Європі застосовують алгоритм 3DES і підтримують механізми управління сесійними ключами великої довжини. За вимогами Visa International та MasterCard International пристрої, що встановлюються (ПІН-пади, POS-термінали), що приймають картки, при введенні ПІН-коду повинні підтримувати використання алгоритму 3DES починаючи з січня 2004р. 3DES має слабкі сторони: відсутність можливостей для паралельних обчислень при шифруванні і низьку швидкість. При визначенні рекомендацій використання 3DES в POS-термінальній мережі ПС обійшли стороною кілька важливих питань. Наприклад, 2 основних протоколи, які використовуються для обміну даними між POS-терміналом і хостовою системою банку-еквайєра – SPDH, розроблений компанією ACI WorldWide , і APACS 40, – не здатні підтримувати ключі великої довжини, тому, не підходять для роботи з 3DES. Доцільно перейти на використання алгоритму ADE, швидкодія та криптостійкість його вища, а довжина ключа – менша, ніж 3DES.

Керування сесійними ключами має необхідність заміни ключів, які використовуються для шифрування інформації (у тому числі і ПІН-блоку). В ідеалі процедура заміни ключів повинна здійснюватися на регулярно, щоб

зловмисник не скористався старим ключем для отримання доступу до шифротексту. На даний момент процедура заміни сесійних ключів для алгоритму DES досить чітко пророблена.

На сьогоднішній момент існує два основні методи управління сесійними ключами: Master Key / Session Key і DUKPT (Derived Unique Key Per Transaction). В першому випадку сесійний ключ постійно оновлюється хостовою системою банку-еквайєра і, зашифрований майстер-ключем, передається на термінал. Зашифроване значення нового ключа, термінал розшифровує його за допомогою свого майстер-ключа. Основною перевагою даного методу є те, що при його застосуванні сесійний ключ ніколи не передається з терміналу на хост, а отже, не буває перехоплений шахраєм. Хост визначає ключ для розшифрування отриманого ПІН-блоку на підставі інформації, що передається разом із транзакцією. На відміну від Master Key / Session Key метод управління сесійними ключами отримав схвалення з боку ANSI. DUKPT відповідає новому стандарту відновлення сесійних ключів в 3DES, що отримав назву ANSI ANSI X9.24-2002. Так аналогічний механізм управління ключами застосовується в банкоматах.

В Україні, як і в європейських країнах, DUKPT не отримав широкого розповсюдження. На даний момент використання цього методу реалізовано тільки в процесінговій компанії UCS. При цьому переважна більшість вітчизняних банків продовжують використовувати метод Master Key / Session Key. Таким чином, напередодні початку процесу глобального переходу на 3DES більшість банків в Європі фактично залишилися без стандарту управління сесійними ключами, що відповідає вимогам безпечного застосування алгоритму 3DES.

У результаті перед фахівцями карткового ринку постало питання розробки нового стандарту, який би заповнив порожнечу, що утворилася після того, як Master Key / Session Key був визнаний непридатним для управління ключами 3DES, або використовувати інші алгоритми. Створення нового стандарту взяли на себе фахівці компаній-виробників ACI WorldWide, HP Atalla, Diebold, Thales e-Security і VeriFone під егідою ANSI. Результатом стала розробка нового

стандарту, що отримала назву ANSI X9 TG31 (GISKE). Унікальність GISKE в тому, що механізм управління сесійними ключами не прив'язаний до алгоритму шифрування. GISKE сумісний з поширеними криптографічними алгоритмами, серед яких DES, 3DES і RSA.

У контексті ефективної боротьби з шахрайством важлива роль відводиться використанню механізмів захисту на мережевому протоколі, який представляє собою набір правил, що дозволяє здійснювати з'єднання і обмін даними між банкоматом та хостом.

Найбільшого поширення в банківських мережах отримав набір протоколів IPsec, призначений для забезпечення захисту даних, переданих за міжмережевим протоколом IP, здійснення підтвердження автентичності та / або шифрування IP-пакетів [22]. IPsec також включає в себе протоколи для захищеного обміну ключами в мережі Інтернет.

На даний час визначений тільки один протокол обміну криптографічними ключами - IKE (Internet Key Exchange) - і два протоколи, що забезпечують захист переданого потоку: ESP забезпечує і AH гарантує тільки цілісність потоку [13–15].

Для досягнення конфіденційності необхідно застосовувати протокол ESP. Можливості автентифікації його дещо обмежені в порівнянні з AH, оскільки дані з IP-заголовка не включаються в цей процес. Однак ESP більш ніж достатній, якщо потрібна автентифікація протоколів тільки верхнього рівня. Можливий варіант установки зв'язку тільки з автентифікацією, без шифрування. Наступною особливістю ESP є маскування розміру переданого пакета за допомогою його доповнення не значимою інформацією (padding).

Конфіденційність даних досягається за допомогою шифрування. Алгоритми шифрування в типовому випадку базуються на механізмі ключів. На сьогодні наявні дві основні схеми: симетричне і асиметричне шифрування. Симетричне шифрування приблизно на три порядки швидше, ніж асиметрична, і тому застосовується при необхідності шифрувати великі масиви даних. ESP підтримує різні алгоритми симетричного шифрування. Зокрема, алгоритм DES

використовується вже більше 20 років. Однак з огляду на його недостатню надійності частіше застосовують потрійне шифрування даних з різними ключами (3DES). Сьогодні для шифрування даних на верхніх рівнях протоколу OSI і актуальних даних існує більш надійний алгоритм - ADE, криптостійкість якого вища, ніж алгоритму 3DES. Прототипом ADE є AES. На сьогоднішній день не було виявлено жодної успішно проведеної атаки на AES, що є підтвердженням надійності алгоритму і доцільність його використання.

Загальна структура стандарту з внесеними пропозиціями зображена на рис. 3 .8.

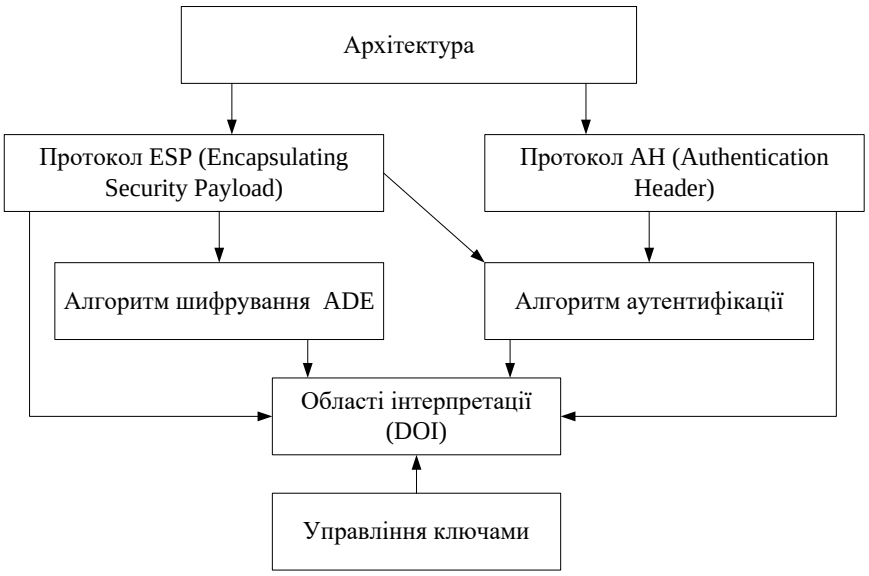

Рисунок 3 .8 - Архітектура IPSec

Щоб обслуговувати різні типи комунікацій, IPsec може оперувати в одному з двох режимів: транспортному або тунельному (рис. 3.9).

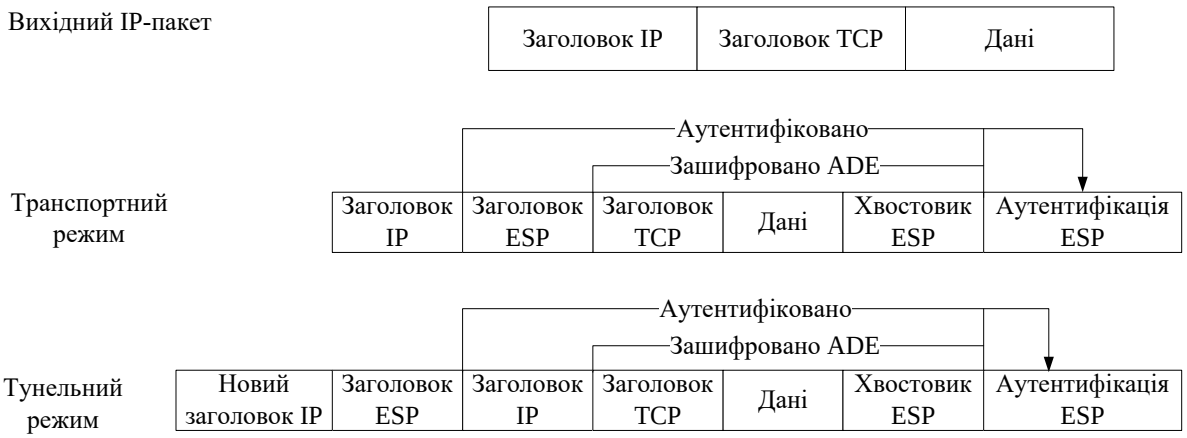

Рисунок 3.9 - Режими роботи IPsec

В транспортному режимі шифрується тільки інформативна частина IP-пакету [13]. Маршрутизація не зачіпається, так як заголовок IP пакету не змінюється (не шифрується). Транспортний режим, як правило, використовується для встановлення з'єднання між хостами. Він може також використовуватися між шлюзами, для захисту тунелів, організованих яким-небудь іншим способом (IP tunnel, GRE та ін.).

Транспортний режим встановлюється, якщо кінцевою точкою є хост або коли комунікації завершуються на кінцевих точках. В разі використання транспортного режиму для типу комунікацій "хост-шлюз" останній повинен діяти як хост-система, на якій безпосередньо функціонують всі протоколи. В іншому випадку для шлюзу потрібен тунельний режим, щоб забезпечити доступ до внутрішньої системи.

В транспортному режимі IP-пакет містить протокол безпеки (AH або ESP), що розташовується після оригінального IP-заголовка і опцій перед будь-яким з протоколів верхнього рівня, що є в пакеті, таким, як TCP або UDP. Таким чином, заголовок IP і опції не беруть участь в процесі автентифікації. Тому, на відміну від інших даних, оригінальна IP-адреса не може бути перевірений на цілісність. AH припускає розширення захисту і на IP-заголовок, чим забезпечується цілісність всього пакету.

Тунельний режим встановлюється для шлюзів і є, по суті, IP-тунелем з автентифікацією і шифруванням. Це найбільш поширений режим

функціонування. Він потрібен для комунікаціях типу “шлюз-шлюз” і “хост-шлюз”. Тунельний режим передбачає два набори IP-заголовків: зовнішній і внутрішній.

Зовнішній IP-заголовок містить IP-адресу VPN-шлюзу, тоді як внутрішній - IP-адресу кінцевої системи позаду шлюзу. Протокол безпеки розміщується після зовнішньої IP-адреси і перед внутрішньою. В транспортному режимі зовнішній IP-заголовок входить до протоколу AH, але не включається до автентифікації при протоколі безпеки ESP, в результаті чого досягається цілісність тільки внутрішнього IP-заголовка та корисного навантаження. Значення параметра TTL (Time To Live), що знаходиться в полі IP-заголовка, зменшується на одиницю при вступі на черговий шлюз системи інкапсуляції, забезпечуючи підрахунок транзитних шлюзів.

3.4. Висновки за третім розділом

Для забезпечення захисту платіжної системи доцільно використовувати алгоритм ADE як на мережному протоколі, так і на протоколі повідомлень. Це пояснюється його високою криптостійкістю, яка забезпечується застосуванням динамічно керованих кріптопримітивів.

4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ХОРОНИ ПРАЦІ

4.1 Естетичне оформлення та ергономічне дослідження робочого місця оператора

Система "людина — дисплей" змушує користувача тривалий час знаходитися в майже фіксованій позі, його рухи обмежені, увага напружена й спрямована на екран руки, фіксовані на клавіатурі. Відносно статичне положення людини при роботі з дисплеєм призводить до напруги м’язово-скелетної системи.

Найкращі умови роботи користувача з клавіатурою забезпечуються в тих випадках, коли його кисті й передпліччя займають положення, близьке до горизонтального.

Ще однією несприятливою дією комп’ютера є акустичний шум, у тому числі ультразвук, який супроводжує роботу комп’ютера.

Сукупність всіх цих факторів негативно впливає на організм людини. Більшість з них можна усунути шляхом організації правильного режиму й умов праці оператора.

Щоб уникнути навантаження на хребетний стовбур оператора, необхідно дуже уважно підійти до питання організації робочого місця. Робочі меблі при роботі з комп’ютером відіграють важливу роль у створенні ергономічно оптимальних умов діяльності людини. Грамотне їх використання дозволяє зменшити ступінь стомленості, збільшити працездатність, продуктивність праці, концентрацію уваги.

Правильний режим роботи має важливе значення у профілактиці професійних захворювань рук оператора. Перерви в роботі, розігрівання м’язів, потягування — усе це допомагає відвернути хворобу.

Для зменшення негативного впливу на зоровий апарат рекомендується періодично змінювати фокусну відстань (дивитися вдалину) та робити

спеціальні вправи для очей (переведення погляду вліво—вправо, вгору—вниз, кругові рухи очима).

Ідеальне робоче місце оператора має відповідати ряду вимог:

1. Оптимальна відстань від очей оператора до екрана монітора й оптимальний уклін лінії погляду (голова не повинна бути підведеною).

2. Достатня освітленість робочих документів і відсутність відблисків на поверхні екрана.

3. Правильна поза сидіння та кут нахилу тулуба.

4. Правильне положення рук на клавіатурі.

5. Регулярне дихання.

Для ідеального робочого місця необхідно:

– монітор, що задовольняє своїми візуальними характеристиками міжнародні вимоги, з регульованою яскравістю та контрастністю екрана, частотою розгорнення не менше 75 Гц (краще 100 Гц);

– за необхідності захисний фільтр;

– регульований стіл для комп’ютера, що дозволяє змінювати висоту положення клавіатури;

– регульоване крісло;

– підставка для ніг;

– тримач для робочих матеріалів.

Раціональна організація режимів праці й відпочинку — важлива умова профілактики передчасного стомлення при роботі з дисплеями.

Помічено, що часті короткі перерви дозволяють значно знизити рівень стомлення і, незважаючи на скорочення загального часу роботи, збільшити продуктивність праці.

4.2 Вимоги до режимів праці і відпочинку при роботі з ВДТ

При організації праці, пов'язаної з використанням ВДТ ЕОМ і ПЕОМ, для збереження здоров'я працюючих, запобігання професійним захворюванням і

підтримки працездатності передбачаються внутрішньозмінні регламентовані перерви для відпочинку.

Внутрішньозмінні режими праці і відпочинку містять додаткові нетривалі перерви в періоди, що передують появі об'єктивних і суб'єктивних ознак стомлення і зниження працездатності.

При виконанні робіт, що належать до різних видів трудової діяльності, за основну роботу з ВДТ слід вважати таку, що займає не менше 50% робочого часу. Впродовж робочої зміни мають передбачатися:

– перерви для відпочинку і вживання їжі (обідні перерви);

– перерви для відпочинку і особистих потреб (згідно з трудовими нормами);

– додаткові перерви, що вводяться для окремих професій з урахуванням особливостей трудової діяльності.

За характером трудової діяльності розрізняють три професійні групи, згідно з діючим класифікатором професій (ДК-003-95 і Зміна N1 до ДК-003-95):

– розробники програм (інженери-програмісти) виконують роботу переважно з відеотерміналом та документацією при необхідності інтенсивного обміну інформацією з ЕОМ і високою частотою прийняття рішень. Робота характеризується інтенсивною розумовою творчою працею з підвищеним напруженням зору, концентрацією уваги на фоні нервово-емоційного напруження, вимушеною робочою позою, загальною гіподинамією, періодичним навантаженням на кисті верхніх кінцівок. Робота виконується в режимі діалогу з ЕОМ у вільному темпі з періодичним пошуком помилок в умовах дефіциту часу;

– оператори електронно-обчислювальних машин виконують роботу, пов'язану з обліком інформації, одержаної з ВДТ за попереднім запитом, або тієї, що надходить з нього, супроводжується перервами різної тривалості, пов'язана з виконанням іншої роботи і характеризується напруженням зору, невеликими фізичними зусиллями, нервовим напруженням середнього ступеня та виконується у вільному темпі;

– оператор комп'ютерного набору виконує одноманітні за характером роботи з документацією та клавіатурою і нечастими нетривалими переключеннями погляду на екран дисплея, з введенням даних з високою швидкістю. Робота характеризується як фізична праця з підвищеним навантаженням на кисті верхніх кінцівок на фоні загальної гіподинамії з напруженням зору (фіксація зору переважно на документи), нервово-емоційним напруженням.

Правилами встановлюються такі внутрішньозмінні режими праці та відпочинку при роботі з ЕОМ при 8-годинній денній робочій зміні в залежності від характеру праці:

– для розробників програм із застосуванням ЕОМ слід призначати регламентовану перерву для відпочинку тривалістю 15 хвилин через кожну годину роботи за ВДТ;

– для операторів із застосуванням ЕОМ слід призначати регламентовані перерви для відпочинку тривалістю 15 хвилин через кожні дві години;

– для операторів комп'ютерного набору слід призначати регламентовані перерви для відпочинку тривалістю 10 хвилин після кожної години роботи за ВДТ.

У всіх випадках, коли виробничі обставини не дозволяють застосувати регламентовані перерви, тривалість безперервної роботи з ВДТ не повинна перевищувати 4 години.

При 12-годинній робочій зміні регламентовані перерви повинні встановлюватися в перші 8 годин роботи аналогічно перервам при 8-годинній робочій зміні, а протягом останніх 4-х годин роботи, незалежно від характеру трудової діяльності, через кожну годину тривалістю 15 хвилин.

Для зниження нервово-емоційного напруження, втомлення зорового аналізатора, поліпшення мозкового кровообігу, подолання несприятливих наслідків гіподинамії, запобігання втомі доцільно деякі перерви використовувати для виконання комплексу вправ, які наведені у Державних

санітарних правилах і нормах роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСанПіН 3.3.2.007-98.

ВИСНОВКИ

Головним національним інтересом є розвиток економіки та добробут громадян України. Забезпечення національних інтересів та економічної безпеки – найважливіші функції держави, реалізація яких покликана посилювати позиції в міжнародному співтоваристві. Національна безпека України як стан захищеності життєво важливих інтересів особистості, суспільства та держави від внутрішніх і зовнішніх загроз - необхідна умова розвитку держави [19, 51].

Сьогодні як ніколи загострюється надзвичайно важливе питання забезпечення інформаційної безпеки України, що є одним з найважливіших національних пріоритетів [37, 39]. Забезпечення інформаційної безпеки є гарантом державної незалежності України, умовою її розвитку.

Розвиток високорентабельної економіки неможливий без впровадження сучасної системи грошового обігу та використання ефективних платіжних механізмів. Швидкий ріст обсягів оброблюваних даних у сучасних інформаційних банківських системах, вихід на ринок нових електронних послуг, стрімкий розвиток комп'ютерної техніки ставлять нові вимоги до надійності та безпеки даних у ВПБС. Така система відноситься до складних багаторівневих систем керування критичного призначення [34]. Все це підвищує ризик проведення атак на ВПБС. Порушення роботи банківських систем приводить до втрати не тільки конфіденційної інформації банку, але та до економічних збитків, як банку, так і його клієнтів, що створює загальнонаціональну проблему. Дані системи потребують комплексного захисту. Важливою складовою частиною ВПБС є підсистема криптографічного захисту інформації, що реалізується відповідними протоколами та механізмами [50, 71, 72]. Незважаючи на широке їх застосування, ВПБС піддані різним атакам і загрозам. Проведений аналіз показав, що навмисні штучні загрози, у першу чергу неправомірні дії співробітників і обслуговуючого персоналу відіграють головну роль у розкраданні конфіденційної інформації. Для оцінки степеня ефективності основних типів атак

доцільно використовувати моделі атак на ВПБС. Це полегшує вибір необхідних засобів і методів захисту інформації в цих системах.

Таким чином, виникає протиріччя між забезпеченням необхідної стійкості та достовірності до переданої конфіденційної інформації та лімітом часу для обробки даних криптографічними засобами захисту (показником оперативності).

Аналіз механізмів захисту довів, що передача інформації вимагає контролю безпеки на всіх рівнях ВПБС. Основним і найбільш ефективним механізмом криптографічного захисту інформації є методи блочного симетричного шифрування. Блочно-симетричні криптографічні перетворення дозволяють забезпечити високу стійкість до різних методів криптографічного аналізу [19, 39, 43-46].

Таким чином, для забезпечення надійного захисту необхідний комплексний підхід, що включає в себе аналіз: загальної структури ВПБС, можливих загроз і реалізованих атак; вибір ратифікованих стандартів для забезпечення автентичності, цілісності та конфіденційності банківських транзакцій, програмна реалізація обраних криптографічних алгоритмів. Характерною рисою методів симетричного криптографічного перетворення інформації є той факт, що стійкість перетворень у цілому здійснюється за рахунок використання нелінійних вузлів замін. Стійкість нелінійних вузлів замін, що здійснюють необоротні/важкооборотні нелінійні перетворення, визначають стійкість використовуваних методів у цілому. Тому першорядне значення для підвищення ефективності симетричних криптографічних засобів захисту інформації здобуває розробка математичних моделей і обчислювальних методів формування нелінійних вузлів замін з поліпшеними властивостями.

В рамках роботи розглянутий алгоритм ADE (Algorithm Of Dynamic Encryption). Основна увага при розробці ADE приділена можливості динамічно управляти процесом лінійного розсіювання та нелінійної заміни під криптографічного шифрування інформації. Із цією метою в базову структуру алгоритму AES введені динамічно змінювані блоки (криптопримітиви) лінійного розсіювання та нелінійних замін, правила функціонування яких задаються

значеннями раундових ключів [18, 40]. За своїх криптографічних властивостях уведені криптопримітиви алгоритму ADE практично не уступають криптопримітивам алгоритму AES, а за рахунок динаміки їх зміни на кожному раунді ітераційного процесу виходить в динаміці управляти криптографічним перетворенням інформації. Це, з одного боку, не погіршує (у порівнянні з AES) стійкість ADE до статистичних методів криптоаналізу, з іншого боку, суттєво ускладнює процес формування алгебраїчних рівнянь, що аналітично зв'язують стан відкритого тексту, шифротексту та ключа. Фактично, формовані системи рівнянь мають вигляд випадкових і щільних систем, що, на нашу думку, суттєво ускладнює завдання криптоаналізу, зокрема, алгебраїчними методами.

Таким чином, застосування блочного симетричного криптографічного перетворення інформації з динамічно керованими примітивами у ВПБС є ефективним.

СПИСОК ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1 Закон України “Про Державну службу спеціального зв’язку та захисту інформації України”: від 23.02.2006 № 3475-IV/ Верховна Рада України. – Офіц. Вид. – К.:Парлам. вид-во, 2006..

2 Закон України “Про електронний цифровий підпис” від 22.05.2003 № 852-IV/ Верховна Рада України. – Офіц. Вид. – К.:Парлам. вид-во, 2003.

3 Закон України “Про захист інформації в інформаційно-телекомунікаційних система”. Відомості Верховної Ради України (ВВР), 1994, N 31, ст.286

4 Закон України “Про інформацію”: від 02.10.1992 № 2657-XII/ Верховна Рада України. – Офіц. Вид. – К.:Парлам. вид-во, 1992.

5 Про перелік відомостей, що не становлять комерційної таємниці: Постанова КМ України від 9.08.1993 р. № 611.

6 ГОСТ 34.310-95. Межгосударственный стандарт. Информационная технология. Криптографическая защита информации. Процедура выработки и проверки электронной цифровой подписи на базе асимметричного криптографического алгоритма. – Киев: Госстандарт Украины, 1998

7 ГОСТ 34.311-95. Межгосударственный стандарт. Информационная технология. Криптографическая защита информации. Функция хеширования. – Киев: Госстандарт Украины, 1998

8 ГОСТ Р 34.10-94. Информационная технология. Криптографическая защита информации. Процедуры выработки и проверки электронной цифровой подписи на базе асимметричного криптографического алгоритма. – Киев: Госстандарт Украины, 1998

9 ГОСТ Р34.11-94. Информационная технология. Криптографическая защита информации. Функция хэширования. Криптография в банковском деле. – М.: МИФИ, 1997. – 274с.

10 Алгоритмы шифрования [електронний ресурс] .– режим доступу: http://www.ixbt.com/soft/alg-encryption-aes-2.shtml

11 Алферов А.П. Основы криптографии./ Алферов А.П., Зубов А.Ю., Кузьмин А.С., Черемушкин А.В. – М.: Гелиос АРВ, 2005 – 250 с.

12 Бабенко Л.К. Особенности дифференциального криптоанализа алгоритма AES [електронний ресурс] / Л.К.Бабенко, Е.А.Ищукова .– режим доступу: http://www.contrterror.tsure.ru/site/magazine8/07-22-Babenko.htm

13 Бакиров Т.В. Анализ защищенности вычислительной сети и методика его проведения [електронний ресурс] // Information Security.– режим доступу: http://www.itsec.ru

14 Бакуренко А.А. Современные методы оценки информационной безопасности автоматизированных систем [електронний ресурс] // Сетевые решения.– режим доступу: http://www.nestor.minsk.by/sr/2006/10/sr61010.html

15 Бардаченко В.Ф. Анализ современных средств аутентификации для систем защиты информации/ В.Ф.Бардаченко, А.В.Кариман, О.К.Колесницкий// Управляющие системы и машины.– К., 2007.– 3.– С.87-92.

16 Біла книга Держспецзв'язку. [електронний ресурс] – режим доступу: http://www.dstszi.gov.ua/dstszi/control/ru/doccatalog/list?currDir=49948

17 Бондаренко В.О. Інформаційна безпека сучасної держави: концептуальні роздуми [електронний ресурс] . – режим доступу: http://www.crime-research.ru/library/strateg.htm

18 Борн Д. Алгоритм шифрования AES теряет стойкость [електронний ресурс] // Daily Digital Digest .– режим доступу: http://www.3dnews.ru/software-news/algoritm_shifrovaniya_aes_teryaet_stoikost/

19 Варналій З.І. Проблеми та шляхи забезпечення економічної безпеки України. [електронний ресурс] // Рада національної безпеки і оборони України . – режим доступу: http://www.rainbow.gov.ua/news/25.html

20 Везоров В.А. Внутренние угрозы в период кризиса. [електронний ресурс] //Компьютерное обозрение.– режим доступу: http://ko.com.ua/node/41711

21 Вихорев С. В. Классификация угроз информационной безопасности [електронний ресурс] – режим доступу: http://www2.cnews.ru/ comments/security/elvis_class.shtml

22 Глоссарий [електронний ресурс] – режим доступу: http://www.glossary.ru

23 Гончарова А.И. Побудова моделей атак на внутришньоплатіжну банківську систему // Актуальні проблеми науки та освіти молоді: теорія, практика, сучасні рішення-2010: (Матеріали міжнародної науково-практичної конференції молодих вчених, аспірантів та студентів) [Електронний ресурс] : ред. В.С. Пономаренко, О.І. Пушкар. – Х.: вид. ХНЕУ, 2010. – 1 електрон. опт. диск (CD-ROM) : кольор. ; 12 см. – Систем. вимоги: Pentium; 32 Mb RAM ; CD-ROM Windows 98/2000/NT/XP; Adobe Acrobat Reader. – Назва з контейнера.

24 Гончарова А.И. Механизмы обеспечения аутентичности во внутриплатежных системах коммерческого банка// Збірник наукових праць студентів специальностей «Інформаційні управляючі системи і технології», «Комп'ютерний еколого-економічний моніторинг». – Х:ХНЕУ, 2009. – С. 88-89.

25 Горбенко И.Д. Сравнительный анализ блочных симметричных шифров, представленных в проекте NESSIE/ И.Д.Горбенко, С.А.Головашич, А.Н.Лепеха// Радиотехника.– 2007.– №134.– С.9-25.

26 Грибунин В.Г. Комплексная система защиты информации на предприятии./ В.Г. Грибунин, В.В. Чудовский. – М:Академия, 2009. – 416 с.

27 Дело [електронний ресурс] – режим доступу: http://delo.ua

28 Евсеев С.П. Построение моделей атак на внутриплатежные банковские системы./ С.П. Евсеев, О.Г. Король, А.И. Гончарова. – Запорожье: ЗНТУ. – 2010. –С. 56-66

29 Евсеев С.П. Построение моделей атак на внутриплатежные банковские системы./ С.П. Евсеев, А.И. Гончарова. // Інформаційні технології та комп'ютерна інженерія (Тези доповідей міжнародної науково-практичної конференції)– Винница: ВНТУ. – 2010. –С. 215-216

30 Ермолаев А.А. Основные угрозы и пути обеспечения национальной безопасности Украины [електронний ресурс]. – режим доступу: http://www.sofia.com.ua/page101.html

31 Золотарев В.В. Фундоментальные основы методик базового экспертного анализа информационных рисков [електронний ресурс].– режим доступу: http://www.lib.tsu.ru/mminfo/000349342/02/image/02-071.pdf

32 Иванов М.А. Стохастические методы и средства защиты информации в компьютерных системах и сетях [електронний ресурс] // Эко-трендз.– режим доступу: http://www.ekot.ru

33 Интернет Университет Информационных Технологий [електронний ресурс] – режим доступу: http://www.intuit.ru

34 Інформаційна безпека [електронний ресурс]. – режим доступу: http://www.ua7.org/mexanizmi-informacijnoї-bezpeki

35 Конеев И. Р. Информационная безопасность предприятия / И. Р.Конеев, А.В. Беляев – Спб.: БХВ-Петербург, 2003. – 752 с.

36 Косякин А.В. Информационная безопасность как объект юрисдикционной охраны [електронний ресурс]/ А.В. Косякин // Административное и муниципальное право.–2009. – №2. – режим доступу до журн.: http://www.recoveryfiles.ru/laws.php?ds=3051

37 Кузнецов А.А. Анализ механизмов обеспечения безопасности банковской информации во внутриплатежных системах коммерческого банка. / А.А. Кузнецов, О.Г. Король, А.М. Ткачов// Матеріали І міжнародної науково-практичної конференції "Безпека та захист інформації в інформаційних і телекомунікаційних системах" 28 – 29 травня 2008 р. Зб. наук. статей "Управління розвитком". ХНЕУ. – 2008. – № 6. – с. 28 – 35.

38 Кузнецов А.А. Исследование статистической безопасности генераторов псевдослучайных чисел [електронний ресурс] .– режим доступу: http://www.lib.ua-ru.net/diss/cont/61642.html

39 Кузнецов А.А.. Механизмы обеспечения аутентичности банковских данных во внутриплатежных системах коммерческого банка. / А.А. Кузнецов, В.Е.Чевардин, С.П. Евсеев, О.Г.Король// Матеріали І міжнародної науково-практичної конференції "Безпека та захист інформації в інформаційних і телекомунікаційних системах" 28 – 29 травня 2008 р. Зб. наук. статей "Управління розвитком". ХНЕУ. – 2008. – № 6 . – с.45-50.

40 Кузнецов О.О. Захист інформації в інформаційних системах. Методи традиційної криптографії. Кузнецов О.О., Євсеєв С.П., Король О.Г. ХНЕУ 2010

41 Кузнецов О.О. Захист інформації та економічна безпека підприємства. Монография/ О.О. Кузнецов, С.П. Євсеєв, С.В. Кавун. – Х.: Вид. ХНЕУ, 2008. – 360 с.

42 Курцвейль Р. Слияние человека с машиной [електронний ресурс] – режим доступу: http://alt-future.narod.ru/Future/kurzweil.htm

43 Лаборатория автоматизации информационных систем [електронний ресурс] – режим доступу: http://www.lais.ru/

44 Логинов А.А. Общие принципы функционирования электронных платежных систем и осуществление мер безопасности при защите от злоупотребления и мошенничества./ Елхимов Н.С. // Конфидент.–1995.–№4.–С.48-54

45 Марущак А.О. Інформаційні ресурси держави: зміст та проблема захисту [електронний ресурс] // Юридичний радник. – режим доступу: http://www.yurradnik.com.ua

46 Мельников В.П. Информационная безопасность и защита информации. – М: Академия,2008. –336 с.

47 Оранжевая книга. [Електронний ресурс] – режим доступу: http://www.zashita-informacii.ru/node/43

48 Панасенко С.В. NESSIE – конкурс криптоалгоритмов [електронний ресурс] – режим доступу: http://www.panasenko.ru/Articles/63/63.html

49 Панасенко С.П. Основы криптографии для экономистов: учебное пособие. Под ред. Л.Г. Гагариной. — М.: Финансы и статистика, 2005 — 176 с.

50 Приходько С.А. Анализ и проектирование распределённых объектно-ориентированных сред выполнения [електронний ресурс] – режим доступу: http://masters.donntu.edu.ua/2008/fvti/prihodko/diss/index.htm

51 Прокоф'єва Д.М. Дослідження змісту категорій інформації з обмеженим доступом відповідно до чинного законодавства України. [електронний ресурс] // Центр информационной безопасности .– режим доступу: http://www.bezpeka.com/ru/lib/spec/art8.html

52 Ровнев В.В. Средства шифрования данных [електронний ресурс] // Aladdin security solution .– режим доступу: http://www.aladdin.ru/catalog/secret_disk/sdsng/tech_details/encoding.php

53 Семенов Ю.А. Алгоритм шифрования AES [електронний ресурс] .– режим доступу: http://book.itep.ru/6/aes.htm

54 Сеть Файстеля [електронний ресурс] – режим доступу: http://www. agran.vp

55 Словник законодавчих термінів [електронний ресурс] – режим доступу: http:// zakon.nau.ua

56 Смарт Н.Криптография: Пер с англ..–М.:Техносфера,2005. – 528с.

57 Столингс В. Криптография и защита сетей: принципы и практика, 2-е изд.: пер. с англ. — М.: издательский дом "Вильям", 2001. — 672 с

58 Украинский ресурс по безопасности. [електронний ресурс] – режим доступу: http://kiev-security.org.ua

59 Фратто М. Сертификация средств безопасности [електронний ресурс] // Сети и системы связи . – режим доступу: http://www.ccc.ru/magazine/depot/03_12/read.html?0502.htm

60 Химка С.С. Разработка моделей и методов для создания системы информационной безопасности корпоративной сети предприятия с учетом различных критериев. [електронний ресурс] – режим доступу: http://masters.donntu.edu.ua/2009/fvti/khimka/diss/index.htm

61 Чевардин В.Е. Метод итерационного хеширования на базе арифметики в группе точек несингулярной эллиптической кривой // Радіоелектронні і комп'ютерні системи. – 2005. – № 3(11). – С. 99-105.

62 Чмола А.Л. Современная прикладная криптография. – М.: Гелиос АРВ, 2002. – 256 с.

63 Шифры Файстеля и SP-сеть [електронний ресурс] – режим доступу: http://www.chhm.net/index.php?articles=79

64 AES Round 1 Information [електронний ресурс].– режим доступу: // http://csrc.nist.gov

65 Bernstein D.J. Cache-timing attacks on AES. [електронний ресурс].– режим доступу: http://cr.yp.to

66 Biham E. Comment on Selecting the Cipher for the AES Second Round. [електронний ресурс].– режим доступу: http://csrc.nist.gov

67 Courtois N.T. Is AES a Secure Cipher. [електронний ресурс].– режим доступу: http://www.cryptosystem.net.

68 Dataforce [електронний ресурс] .– режим доступу: http://www.dataforce.com.ua/Tehnologii/aes.html

69 Digital Security. [електронний ресурс] – режим доступу: http://www.dsec.ru

70 Fuller J. On Linear Redundancy in the AES S-box. [електронний ресурс].– режим доступу: http://eprint.iacr.org

71 Helen Gustafson, et. al. Statistical test suite Crypt-SX. – Available on http://www.isrc.qut.edu.au/cryptx.

72 Leung A.K. Sequence Complexity as Test for Cryptographic Systems. – Advances in Cryptology – CRYPTO’84. Proc. LNCS, Vol. 196 – Springer-Verlag

73 Rabin M.O. Fingerprinting by Random Polynomials // Tech. Rep. TR-15-81, Center: in Compating Technology, Harvard Univ., Cambridge, Mass., 1981

74 Simmons G. I. Authentication theory/coding theory, presented at Crypto’84, Santa Barbara, CA. – 1984. P. 19-22

ДОДАТКИ

Додаток А

Результати експериментального дослідження статистичної безпеки AES та ADE в NIST STS

Таблиця А.1

Результати експериментального дослідження статистичної безпеки AES

Quantity	Proportion	Statistical test
1	2	3
86	0,960	nonperiodic-templates
127	0,960	nonperiodic-templates
150	0,960	nonperiodic-templates
151	0,960	nonperiodic-templates
28	0,970	nonperiodic-templates
72	0,970	nonperiodic-templates
88	0,970	nonperiodic-templates
112	0,970	nonperiodic-templates
120	0,970	nonperiodic-templates
159	0,970	nonperiodic-templates
187	0,970	nonperiodic-templates
161	0,971	nonperiodic-templates
180	0,971	random-excursions-variant
181	0,971	random-excursions-variant
8	0,980	nonperiodic-templates
16	0,980	nonperiodic-templates
19	0,980	nonperiodic-templates
26	0,980	nonperiodic-templates
33	0,980	nonperiodic-templates
40	0,980	nonperiodic-templates
43	0,980	nonperiodic-templates
46	0,980	nonperiodic-templates
48	0,980	nonperiodic-templates
50	0,980	nonperiodic-templates
52	0,980	nonperiodic-templates
54	0,980	nonperiodic-templates
65	0,980	nonperiodic-templates
69	0,980	nonperiodic-templates
71	0,980	nonperiodic-templates
79	0,980	nonperiodic-templates
81	0,980	nonperiodic-templates
93	0,980	nonperiodic-templates
103	0,980	nonperiodic-templates
105	0,980	nonperiodic-templates
109	0,980	nonperiodic-templates
113	0,980	nonperiodic-templates

Продовження табл. А.1

1	2	3
129	0,980	nonperiodic-templates
130	0,980	nonperiodic-templates
134	0,980	nonperiodic-templates
138	0,980	nonperiodic-templates
152	0,980	nonperiodic-templates
155	0,980	nonperiodic-templates
186	0,980	serial
188	0,980	lempel-ziv
163	0,985	random-excursions
165	0,985	random-excursions
168	0,985	random-excursions-variant
169	0,985	random-excursions-variant
170	0,985	random-excursions-variant
173	0,985	random-excursions-variant
174	0,985	random-excursions-variant
175	0,985	random-excursions-variant
176	0,985	random-excursions-variant
178	0,985	random-excursions-variant
179	0,985	random-excursions-variant
182	0,985	random-excursions-variant
183	0,985	random-excursions-variant
184	0,985	random-excursions-variant
185	0,985	random-excursions-variant
4	0,990	cumulative-sums
10	0,990	nonperiodic-templates
12	0,990	nonperiodic-templates
13	0,990	nonperiodic-templates
15	0,990	nonperiodic-templates
17	0,990	nonperiodic-templates
21	0,990	nonperiodic-templates
23	0,990	nonperiodic-templates
24	0,990	nonperiodic-templates
25	0,990	nonperiodic-templates
27	0,990	nonperiodic-templates
30	0,990	nonperiodic-templates
32	0,990	nonperiodic-templates
34	0,990	nonperiodic-templates
35	0,990	nonperiodic-templates
36	0,990	nonperiodic-templates
38	0,990	nonperiodic-templates

Продовження табл. А.1

1	2	3
39	0,990	nonperiodic-templates
44	0,990	nonperiodic-templates
45	0,990	nonperiodic-templates
51	0,990	nonperiodic-templates
53	0,990	nonperiodic-templates
56	0,990	nonperiodic-templates
58	0,990	nonperiodic-templates
60	0,990	nonperiodic-templates
61	0,990	nonperiodic-templates
64	0,990	nonperiodic-templates
67	0,990	nonperiodic-templates
70	0,990	nonperiodic-templates
75	0,990	nonperiodic-templates
82	0,990	nonperiodic-templates
84	0,990	nonperiodic-templates
85	0,990	nonperiodic-templates
87	0,990	nonperiodic-templates
91	0,990	nonperiodic-templates
92	0,990	nonperiodic-templates
94	0,990	nonperiodic-templates
95	0,990	nonperiodic-templates
97	0,990	nonperiodic-templates
100	0,990	nonperiodic-templates
101	0,990	nonperiodic-templates
102	0,990	nonperiodic-templates
104	0,990	nonperiodic-templates
106	0,990	nonperiodic-templates
107	0,990	nonperiodic-templates
110	0,990	nonperiodic-templates
114	0,990	nonperiodic-templates
118	0,990	nonperiodic-templates
119	0,990	nonperiodic-templates
122	0,990	nonperiodic-templates
123	0,990	nonperiodic-templates
124	0,990	nonperiodic-templates
125	0,990	nonperiodic-templates
131	0,990	nonperiodic-templates
132	0,990	nonperiodic-templates
133	0,990	nonperiodic-templates
135	0,990	nonperiodic-templates

Продовження табл. А.1

1	2	3
140	0,990	nonperiodic-templates
142	0,990	nonperiodic-templates
144	0,990	nonperiodic-templates
145	0,990	nonperiodic-templates
147	0,990	nonperiodic-templates
148	0,990	nonperiodic-templates
149	0,990	nonperiodic-templates
154	0,990	nonperiodic-templates
156	0,990	nonperiodic-templates
157	0,990	overlapping-templates
158	0,990	universal
189	0,990	linear-complexity
1	1,000	frequency
2	1,000	block-frequency
3	1,000	cumulative-sums
5	1,000	runs
6	1,000	longest-run
7	1,000	rank
9	1,000	nonperiodic-templates
11	1,000	nonperiodic-templates
14	1,000	nonperiodic-templates
18	1,000	nonperiodic-templates
20	1,000	nonperiodic-templates
22	1,000	nonperiodic-templates
29	1,000	nonperiodic-templates
31	1,000	nonperiodic-templates
37	1,000	nonperiodic-templates
41	1,000	nonperiodic-templates
42	1,000	nonperiodic-templates
47	1,000	nonperiodic-templates
49	1,000	nonperiodic-templates
55	1,000	nonperiodic-templates
57	1,000	nonperiodic-templates
59	1,000	nonperiodic-templates
62	1,000	nonperiodic-templates
63	1,000	nonperiodic-templates
66	1,000	nonperiodic-templates
68	1,000	nonperiodic-templates
73	1,000	nonperiodic-templates
74	1,000	nonperiodic-templates

Закінчення табл. А.1

1	2	3
76	1,000	nonperiodic-templates
77	1,000	nonperiodic-templates
78	1,000	nonperiodic-templates
80	1,000	nonperiodic-templates
83	1,000	nonperiodic-templates
89	1,000	nonperiodic-templates
90	1,000	nonperiodic-templates
96	1,000	nonperiodic-templates
98	1,000	nonperiodic-templates
99	1,000	nonperiodic-templates
108	1,000	nonperiodic-templates
111	1,000	nonperiodic-templates
115	1,000	nonperiodic-templates
116	1,000	nonperiodic-templates
117	1,000	nonperiodic-templates
121	1,000	nonperiodic-templates
126	1,000	nonperiodic-templates
128	1,000	nonperiodic-templates
136	1,000	nonperiodic-templates
137	1,000	nonperiodic-templates
139	1,000	nonperiodic-templates
141	1,000	nonperiodic-templates
143	1,000	nonperiodic-templates
146	1,000	nonperiodic-templates
153	1,000	nonperiodic-templates
160	1,000	random-excursions
162	1,000	random-excursions
164	1,000	random-excursions
166	1,000	random-excursions
167	1,000	random-excursions
171	1,000	random-excursions-variant
172	1,000	random-excursions-variant
177	1,000	random-excursions-variant

Таблиця А.2

Результати експериментального дослідження статистичної безпеки ADE

Quantity	Proportion	Statistical test
1	2	3
2	0,9900	block-frequency
3	0,9800	cumulative-sums
4	0,9800	cumulative-sums
5	1,0000	runs
6	1,0000	longest-run
7	1,0000	rank
8	1,0000	rank
9	0,9900	nonperiodic-templates
10	1,0000	nonperiodic-templates
11	1,0000	nonperiodic-templates
12	0,9900	nonperiodic-templates
13	1,0000	nonperiodic-templates
14	1,0000	nonperiodic-templates
15	1,0000	nonperiodic-templates
16	1,0000	nonperiodic-templates
17	1,0000	nonperiodic-templates
18	0,9900	nonperiodic-templates
19	0,9900	nonperiodic-templates
20	0,9800	nonperiodic-templates
21	0,9900	nonperiodic-templates
22	0,9900	nonperiodic-templates
23	1,0000	nonperiodic-templates
24	1,0000	nonperiodic-templates
25	0,9900	nonperiodic-templates
26	0,9900	nonperiodic-templates
27	0,9900	nonperiodic-templates
28	0,9900	nonperiodic-templates
29	0,9800	nonperiodic-templates
30	1,0000	nonperiodic-templates
31	1,0000	nonperiodic-templates
32	0,9800	nonperiodic-templates
33	0,9900	nonperiodic-templates
34	0,9800	nonperiodic-templates
35	1,0000	nonperiodic-templates
36	1,0000	nonperiodic-templates
37	1,0000	nonperiodic-templates
38	1,0000	nonperiodic-templates
39	0,9900	nonperiodic-templates
40	1,0000	nonperiodic-templates
41	0,9800	nonperiodic-templates

Продовження табл. А.2

1	2	3
42	1,0000	nonperiodic-templates
43	0,9700	nonperiodic-templates
44	0,9900	nonperiodic-templates
45	0,9700	nonperiodic-templates
46	0,9900	nonperiodic-templates
47	0,9900	nonperiodic-templates
48	0,9800	nonperiodic-templates
49	1,0000	nonperiodic-templates
50	0,9900	nonperiodic-templates
51	0,9900	nonperiodic-templates
52	0,9800	nonperiodic-templates
53	1,0000	nonperiodic-templates
54	0,9900	nonperiodic-templates
55	0,9900	nonperiodic-templates
56	1,0000	nonperiodic-templates
57	0,9900	nonperiodic-templates
58	0,9900	nonperiodic-templates
59	1,0000	nonperiodic-templates
60	0,9900	nonperiodic-templates
61	1,0000	nonperiodic-templates
62	1,0000	nonperiodic-templates
63	0,9900	nonperiodic-templates
64	0,9900	nonperiodic-templates
65	0,9900	nonperiodic-templates
66	0,9900	nonperiodic-templates
67	1,0000	nonperiodic-templates
68	0,9800	nonperiodic-templates
69	0,9800	nonperiodic-templates
70	1,0000	nonperiodic-templates
71	1,0000	nonperiodic-templates
72	1,0000	nonperiodic-templates
73	1,0000	nonperiodic-templates
74	0,9800	nonperiodic-templates
75	0,9900	nonperiodic-templates
76	1,0000	nonperiodic-templates
77	1,0000	nonperiodic-templates
78	0,9900	nonperiodic-templates
79	1,0000	nonperiodic-templates
80	0,9700	nonperiodic-templates
81	0,9800	nonperiodic-templates
82	0,9800	nonperiodic-templates
83	0,9900	nonperiodic-templates

Продовження табл. А.2

84	0,9900	nonperiodic-templates
85	0,9800	nonperiodic-templates
86	0,9900	nonperiodic-templates
87	1,0000	nonperiodic-templates
88	1,0000	nonperiodic-templates
89	0,9900	nonperiodic-templates
90	0,9900	nonperiodic-templates
91	0,9900	nonperiodic-templates
92	1,0000	nonperiodic-templates
93	0,9900	nonperiodic-templates
94	0,9900	nonperiodic-templates
95	1,0000	nonperiodic-templates
96	0,9800	nonperiodic-templates
97	1,0000	nonperiodic-templates
98	1,0000	nonperiodic-templates
99	1,0000	nonperiodic-templates
100	0,9900	nonperiodic-templates
101	1,0000	nonperiodic-templates
102	0,9900	nonperiodic-templates
103	1,0000	nonperiodic-templates
104	0,9900	nonperiodic-templates
105	0,9900	nonperiodic-templates
106	1,0000	nonperiodic-templates
107	1,0000	nonperiodic-templates
108	0,9900	nonperiodic-templates
109	0,9900	nonperiodic-templates
110	0,9900	nonperiodic-templates
111	1,0000	nonperiodic-templates
112	1,0000	nonperiodic-templates
113	0,9900	nonperiodic-templates
114	0,9900	nonperiodic-templates
115	0,9900	nonperiodic-templates
116	1,0000	nonperiodic-templates
117	1,0000	nonperiodic-templates
118	0,9900	nonperiodic-templates
119	1,0000	nonperiodic-templates
120	0,9900	nonperiodic-templates
121	1,0000	nonperiodic-templates
122	0,9700	nonperiodic-templates
123	0,9800	nonperiodic-templates
124	1,0000	nonperiodic-templates
125	1,0000	nonperiodic-templates
126	0,9800	nonperiodic-templates

Продовження табл. А.2

1	2	3
127	0,9900	nonperiodic-templates
128	1,0000	nonperiodic-templates
129	0,9800	nonperiodic-templates
130	0,9900	nonperiodic-templates
131	1,0000	nonperiodic-templates
132	0,9900	nonperiodic-templates
133	1,0000	nonperiodic-templates
134	0,9800	nonperiodic-templates
135	1,0000	nonperiodic-templates
136	0,9900	nonperiodic-templates
137	1,0000	nonperiodic-templates
138	1,0000	nonperiodic-templates
139	1,0000	nonperiodic-templates
140	0,9800	nonperiodic-templates
141	0,9900	nonperiodic-templates
142	1,0000	nonperiodic-templates
143	0,9900	nonperiodic-templates
144	0,9900	nonperiodic-templates
145	0,9900	nonperiodic-templates
146	0,9900	nonperiodic-templates
147	1,0000	nonperiodic-templates
148	0,9700	nonperiodic-templates
149	0,9700	nonperiodic-templates
150	1,0000	nonperiodic-templates
151	1,0000	nonperiodic-templates
152	0,9700	nonperiodic-templates
153	1,0000	nonperiodic-templates
154	1,0000	nonperiodic-templates
155	1,0000	nonperiodic-templates
156	0,9800	nonperiodic-templates
157	0,9900	overlapping-templates
158	1,0000	universal
159	0,9800	apen
160	1,0000	random-excursions
161	0,9836	random-excursions
162	0,9836	random-excursions
163	1,0000	random-excursions
164	0,9672	random-excursions
165	0,9836	random-excursions
166	1,0000	random-excursions
167	0,9672	random-excursions
168	0,9672	random-excursions-variant

Закінчення табл. А.2

1	2	3
169	0,9672	random-excursions-variant
170	0,9672	random-excursions-variant
171	0,9836	random-excursions-variant
172	1,0000	random-excursions-variant
173	1,0000	random-excursions-variant
174	0,9836	random-excursions-variant
175	0,9836	random-excursions-variant
176	1,0000	random-excursions-variant
177	0,9836	random-excursions-variant
178	0,9836	random-excursions-variant
179	0,9836	random-excursions-variant
180	1,0000	random-excursions-variant
181	1,0000	random-excursions-variant
182	1,0000	random-excursions-variant
183	1,0000	random-excursions-variant
184	1,0000	random-excursions-variant
185	1,0000	random-excursions-variant
186	1,0000	serial
187	0,9900	serial
188	1,0000	lempel-ziv
189	0,9800	linear-complexity

Додаток Б
Лістинг програмної реалізації алгоритму ADE

//Project1.cpp
```
#include <vcl.h>
#pragma hdrstop
USERES("Project1.res");
USEFORM("Unit1.cpp", Form1);
//---------------------------------------------------------------------------
WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
{
    try
    {
         Application->Initialize();
         Application->CreateForm(__classid(TForm1), &Form1);
         Application->Run();
    }
    catch (Exception &exception)
    {
         Application->ShowException(&exception);
    }
    return 0;
}
```

//Unit1.cpp
```
#include <stdio.h>
#include <sys\stat.h>
#pragma hdrstop
#include "def.h"
#include "data.h"
#include "Unit1.h"
#pragma package(smart_init)
#pragma link "CGAUGES"
#pragma resource "*.dfm"

TForm1      *Form1;
AnsiString  MyFName1,MyFName2,MyFName3,MyFName4;
FILE        *fout,*fout1,*fout2;
FILE        *fin,*fin1,*fin2,*fin3;
fpos_t      flen;                       //  Длина файла
int         kod_dec;                    //  Признак кодирования/декодирования
char        *cp, ch, key[32];

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

void __fastcall TForm1::N2Click(TObject *Sender)
{
    Close();
}

void __fastcall TForm1::Button1Click(TObject *Sender)
{
    char            buf[BLOCK_LEN];
    FILE *fout;
    unsigned long   rlen;
    AnsiString name;

       MyFName1 = "";

    OpenDialog1->Filter = "|*.c";
    OpenDialog1->InitialDir = "D:\\63\\Enc";
    OpenDialog1->FileName = "";
    OpenDialog1->Title = "Выберите файл для шифрования...";
    if(OpenDialog1->Execute())
    {
        MyFName1 = OpenDialog1->FileName;
        struct stat statbuf;
        FILE *handle = fopen(MyFName1.c_str(), "rb");
        stat(MyFName1.c_str(),&statbuf);
        fclose(handle);
        name = ExtractFileName(MyFName1);
        Edit1->Text = name;
        if(CheckBox1->Checked==true)
```

```
{
            if(statbuf.st_size>150000)
            {
                Memo1->Clear();
                Memo1->Lines->Add("ОТКРЫТЫЙ ТЕКСТ");
            }
            else Memo1->Lines->LoadFromFile(MyFName1);
        }
        else
        {
            if(Memo1->Lines->Count>0)
            {
                Memo1->Clear();
                Memo1->Lines->Add("ОТКРЫТЫЙ ТЕКСТ");
            }
            else
                Memo1->Lines->Add("ОТКРЫТЫЙ ТЕКСТ");
        }
    }
    if( (Edit1->Text!="") && (MyFName1!="") )
        Button4->Enabled = true;
    else
    {
        Button4->Enabled = false;
        Edit1->Text = "";
    }
    fin1 = fopen(MyFName1.c_str(), "rb");
    if(fin1)
    {
        fseek(fin1, 0, SEEK_END);
        fgetpos(fin1, &flen);
        rlen = file_len(flen);
        fseek(fin1, 0, SEEK_SET);
        StatusBar1->Panels->Items[1]->Text = " Длина файла в байтах: " + AnsiString(rlen);
    }
    fclose(fin1);
    Memo2->Clear();
    Memo3->Clear();
    buf[0] = ' ';
    fout = fopen(MyFName3.c_str(), "wb");
    if(fout) fwrite(buf, 1, 0, fout);
    fclose(fout);

   fout = fopen(MyFName4.c_str(), "wb");
    if(fout) fwrite(buf, 1, 0, fout);
    fclose(fout);
}

void __fastcall TForm1::Button4Click(TObject *Sender)
{
    char            buf[BLOCK_LEN], dbuf[2 * BLOCK_LEN];
    unsigned long   i, len, rlen;
    AnsiString name;

    MyFName2 = "";
    dbuf[0] = '\x0';

    OpenDialog1->Filter = "|*.txt";
    OpenDialog1->InitialDir = "D:\\63\\Key";
    OpenDialog1->FileName = "";
    OpenDialog1->Title = "Выберите файл ключа...";
    if(OpenDialog1->Execute())
    {
        MyFName2 = OpenDialog1->FileName;
        name = ExtractFileName(MyFName2);
        Edit6->Text = name;
    }

    fin1 = fopen(MyFName2.c_str(), "rb");
    if(fin1)
    {
        fseek(fin1, 0, SEEK_END);
        fgetpos(fin1, &flen);
        rlen = file_len(flen);
        fseek(fin1, 0, SEEK_SET);
```

```
  if(rlen <= BLOCK_LEN)
        {
            len = (unsigned long) fread(dbuf, 1, BLOCK_LEN, fin1);
            if(len != BLOCK_LEN)
                printf("Ошибка чтения\n");
        }
        else
        {
            while(rlen > 0 && !feof(fin1))
            {
                len = (unsigned long)fread(buf, 1, BLOCK_LEN, fin1);
                strcat(dbuf, buf);
                strtok(dbuf," ");

                if(len != BLOCK_LEN)
                    printf("Ошибка чтения\n");
            }
        }
        Edit5->Text = AnsiString(dbuf);
    }
    fclose(fin1);

    if( (Edit6->Text!="") && (MyFName2!="") )
        Button2->Enabled = true;
    else
    {
        Button2->Enabled = false;
                                              Edit6->Text = "";
    }

    if( (Edit6->Text!="") && (Edit1->Text!="") &&
        (Edit2->Text!="") ) BitBtn1->Enabled = true;
    if(RadioButton3->Checked==true)
        Memo2->Clear();
    Memo3->Clear();
}

void __fastcall TForm1::FormCreate(TObject *Sender)
{
    StatusBar1->Panels->Items[0]->Text = " Операция: Шифрование";
    StatusBar1->Panels->Items[2]->Text = " Режим: Cipher Block Chaining";
    Memo1->Clear();
    Memo2->Clear();
    Memo3->Clear();
    if(RadioButton1->Checked==true)
    {
        Edit3->Color = clBtnFace;
        Button3->Enabled = false;
        Button2->Enabled = false;
        Button4->Enabled = false;
    }
}

void __fastcall TForm1::RadioButton2Click(TObject *Sender)
{
    StatusBar1->Panels->Items[0]->Text = " Операция: Расшифровывание";
    StatusBar1->Panels->Items[1]->Text = "";
    Edit3->Color = clWindow;
    Edit1->Color = clBtnFace;
    Memo3->Color = clWindow;
    Button3->Enabled = true;
    Button1->Enabled = false;
    Button2->Enabled = false;
    Button4->Enabled = true;
}

void __fastcall TForm1::RadioButton3Click(TObject *Sender)
{
    StatusBar1->Panels->Items[0]->Text = " Операция: Шифрование/Расшифровывание";
    StatusBar1->Panels->Items[1]->Text = "";
    Button1->Enabled = true;
    Button2->Enabled = true;
    Button3->Enabled = true;
    Button4->Enabled = true;

    Edit3->Color = clWindow;
```

```
Edit1->Color = clWindow;
    Memo3->Color = clWindow;
    if( (Edit1->Text!="") && (Edit6->Text!="") &&
        (Edit2->Text!="") && (Edit3->Text!="") )
    BitBtn1->Enabled = true;
}

void __fastcall TForm1::RadioButton1Click(TObject *Sender)
{
    StatusBar1->Panels->Items[0]->Text = " Операция: Шифрование";

StatusBar1->Panels->Items[1]->Text = "";
    Edit1->Color = clWindow;
    Edit3->Color = clBtnFace;
    Memo3->Color = clBtnFace;
    Edit5->Text = "";
    Memo2->Clear();
    Memo3->Clear();
    Edit2->Text = "";
    Edit3->Text = "";
    Edit6->Text = "";
    Button3->Enabled = false;
    Button2->Enabled = false;
    Button1->Enabled = true;
    Button4->Enabled = true;
}

void __fastcall TForm1::Button2Click(TObject *Sender)
{
    char            buf[BLOCK_LEN], dbuf[2 * BLOCK_LEN];
    AnsiString name;

    MyFName3 = "";
    dbuf[0] = '\x0';

    OpenDialog1->Filter = "|*.enc";
    OpenDialog1->InitialDir = "D:\\63\\Int";
    OpenDialog1->FileName = "";
    OpenDialog1->Title = "Выберите месторасположения для зашифрованного файла...";
    if(OpenDialog1->Execute())
    {
        MyFName3 = OpenDialog1->FileName;
        name = ExtractFileName(MyFName3);
        Edit2->Text = name;
    }
    if( (Edit2->Text!="") && (MyFName3!="") )
    {
        if(RadioButton3->Checked!=true)
            BitBtn1->Enabled = true;
    }
    else
    {
        if(RadioButton3->Checked!=true)
            BitBtn1->Enabled = false;
        Edit2->Text = "";
    }
    if(Edit6->Text!="") Button2->Enabled = true;
}

void __fastcall TForm1::BitBtn1Click(TObject *Sender)
{
    char    buf[2*BLOCK_LEN];
    unsigned long   i, len, rlen;
    WideString buf2;
    TDateTime T1, T2;

    buf[0]   = '\x0';
    buf2 = "";

    if(RadioButton1->Checked==true)
  {
        Label19->Caption = "Шифрование...";
        ProgressBar1->Max = 100;
        ProgressBar1->Position = 0;
        ProgressBar1->Visible = true;
```

```
Application->ProcessMessages();

        BitBtn1->Enabled = false;
        T1 = Time();
        buf2 = "";
        Memo2->Clear();

        adexam();

        if(CheckBox1->Checked==true)
        {
            fin2 = fopen(MyFName3.c_str(), "rb");
            if(fin2)
            {
                fseek(fin2, 0, SEEK_END);
                fgetpos(fin2, &flen);
                rlen = file_len(flen);
                fseek(fin2, 0, SEEK_SET);

                if(rlen>150000)
                    Memo2->Lines->Append("ФАЙЛ СЛИШКОМ ВЕЛИК...");
                else
                {
                    while(rlen > 0 && !feof(fin2))
                    {
                        len = (unsigned long)fread(buf, 1, 2*BLOCK_LEN, fin2);
                        rlen -= len;
                        if( (rlen>1000) && (rlen<1200) ) break;

                        unsigned long i1;
                        for (i1=0;i1<len;i1++)
                        {
                            if(buf[i1]=='\x0')
                                buf[i1]=0x20;
                        }
                        for(i = len;i < 2*BLOCK_LEN;i++)
                            buf[i] = 0;

                        buf2 += buf;
                    }
                }
            }
            fclose(fin2);
            Memo2->Lines->Append(buf2);
        }
        else
            Memo2->Lines->Append("ЗАКРЫТЫЙ ТЕКСТ");

        T2 = Time();
        AnsiString S;
        DateTimeToString(S, "hh:nn:ss:zzz", T2 - T1);
        StatusBar1->Panels->Items[1]->Text = " Продолжительность: " + S;
        BitBtn1->Enabled = true;
        ProgressBar1->Visible = false;
        Label19->Caption = "";
         Application->ProcessMessages();
    }

    if(RadioButton2->Checked==true)
    {
        Label19->Caption = "Расшифровывание...";
        ProgressBar1->Max = 100;
        ProgressBar1->Position = 0;
        ProgressBar1->Visible = true;
        Application->ProcessMessages();

        BitBtn1->Enabled = false;
        T1 = Time();
        Memo3->Clear();

        adexam();

        struct stat statbuf;
        FILE *handle = fopen(MyFName4.c_str(), "rb");
        stat(MyFName4.c_str(),&statbuf);
        fclose(handle);
```

```
if(CheckBox1->Checked==true)
        {
            if( (statbuf.st_size>150000) || (statbuf.st_size<=0) )
            {
                Memo1->Clear();
                Memo1->Lines->Add("ОТКРЫТЫЙ ТЕКСТ");
            }
            else
                Memo3->Lines->LoadFromFile(MyFName4.c_str());
        }
        else
            Memo3->Lines->Add("ОТКРЫТЫЙ ТЕКСТ");

        T2 = Time();
        AnsiString S;
        DateTimeToString(S, "hh:nn:ss:zzz", T2 - T1);
        StatusBar1->Panels->Items[1]->Text = " Продолжительность: " + S;
        Button1->Enabled = false;
        Button3->Enabled = true;
        Button2->Enabled = true;
        Button4->Enabled = true;
        BitBtn1->Enabled = true;

        ProgressBar1->Visible = false;
        Label19->Caption = "";
        Application->ProcessMessages();
    }

    if(RadioButton3->Checked==true)
    {
        Label19->Caption = "Шифрование...";
        ProgressBar1->Max = 100;
        ProgressBar1->Position = 0;
        ProgressBar1->Visible = true;
        Application->ProcessMessages();

        BitBtn1->Enabled = false;
        Memo2->Clear();
        Memo3->Clear();
        buf2 = "";

        kod_dec++;
        T1 = Time();

        adexam();

        if(CheckBox1->Checked==true)
        {
            fin2 = fopen(MyFName3.c_str(), "rb");
            if(fin2)
            {
                fseek(fin2, 0, SEEK_END);
                fgetpos(fin2, &flen);
                rlen = file_len(flen);
                fseek(fin2, 0, SEEK_SET);

                if(rlen>150000)
                    Memo2->Lines->Append("ФАЙЛ СЛИШКОМ ВЕЛИК...");
                else
                {
                    while(rlen > 0 && !feof(fin2))
                    {
                        len = (unsigned long)fread(buf, 1, 2*BLOCK_LEN, fin2);
                        rlen -= len;

                        unsigned long i1;
                        for (i1=0;i1<len;i1++)
                        {
                            if(buf[i1]=='\x0')
                                buf[i1]=0x20;
                        }
                        for(i = len;i < 2*BLOCK_LEN;i++)
                            buf[i] = 0;
                        buf2 += buf;
                    }
                }
```

```
  }
            fclose(fin2);
            Memo2->Lines->Append(buf2);
        }
        else
            Memo2->Lines->Append("ЗАКРЫТЫЙ ТЕКСТ");

        Label19->Caption = "Расшифровывание...";
        Application->ProcessMessages();
        buf2 = "";
        kod_dec++;
        adexam();
        ProgressBar1->Visible = false;
        Label19->Caption = "";
        Application->ProcessMessages();

        struct stat statbuf;
        FILE *handle = fopen(MyFName4.c_str(), "rb");
        stat(MyFName4.c_str(),&statbuf);
        fclose(handle);
        if(CheckBox1->Checked==true)
        {
            if( (statbuf.st_size>150000) || (statbuf.st_size<=0) )
{
                Memo3->Clear();
                Memo3->Lines->Add("ОТКРЫТЫЙ ТЕКСТ");
            }
            else
                Memo3->Lines->LoadFromFile(MyFName4.c_str());
        }
        else
            Memo3->Lines->Add("ОТКРЫТЫЙ ТЕКСТ");
        Memo3->Color = clWindow;
        fclose(fin3);
        fclose(fin2);

        T2 = Time();
        AnsiString S;
        DateTimeToString(S, "hh:nn:ss:zzz", T2 - T1);
        StatusBar1->Panels->Items[1]->Text = " Продолжительность: " + S;
        kod_dec = 0;
        BitBtn1->Enabled = true;
        Button1->Enabled = true;
        Button3->Enabled = true;
        Button2->Enabled = true;
        Button4->Enabled = true;
    }
}

void __fastcall TForm1::Button3Click(TObject *Sender)
{
    AnsiString name;

    MyFName4 = "";
    OpenDialog1->Filter = "|*.dec";
    OpenDialog1->InitialDir = "D:\\63\\Dec";
    OpenDialog1->FileName = "";
    OpenDialog1->Title = "Выберите файл результата расшифровывания...";
    if(OpenDialog1->Execute())
    {
        MyFName4 = OpenDialog1->FileName;
        name = ExtractFileName(MyFName4);
        Edit3->Text = name;
    }
    if( (Edit3->Text!="") && (MyFName4!="") )
        BitBtn1->Enabled = true;
    else
    {
        BitBtn1->Enabled = false;
        Edit3->Text = "";
    }
}

void gen_tabs()
{
    ade_32t  w;
```

```
 int  i;

    for(i = 0, w = 1; i < 10; ++i)
    {
        rcon_tab[i] = bytes2word(w, 0, 0, 0);
        w = f2(w);
    }
for(i = 0; i < 256; ++i)
    {
        ade_08t    b;

        b = fwd_affine(opr_tab[(ade_08t)i]);
        w = bytes2word(b, 0, 0, 0);
        fl_tab[0][i] = w;
        fl_tab[1][i] = upr(w,1);
        fl_tab[2][i] = upr(w,2);
        fl_tab[3][i] = upr(w,3);
    }
}

void __fastcall TForm1::adexam()
{
    char    buf[64];
    FILE    *f1;

    int     i, by, err = 0;

    //  Считывание ключа
    f1 = fopen(MyFName2.c_str(), "rb");
    fscanf(f1,"%s", buf);
    fclose(f1);

    cp = buf;               // Указатель на шестнадцатеричные цифры
    i = 0;
    while(i < 64 && *cp)    // Максимальная длина ключа - 32 байта
    {                       // соответствующая 64 шестнадцатеричным цифрам
        ch = toupper(*cp++);
        if(ch >= '0' && ch <= '9')
            by = (by << 4) + ch - '0';
        else if(ch >= 'A' && ch <= 'F')
            by = (by << 4) + ch - 'A' + 10;
        else
        {
            Application->MessageBox("Ключ должен быть представлен \n"
                                    "шестнадцатеричными цифрами!!!  ",
                                    "Сообщение об ошибке",
                                    MB_OK + MB_ICONINFORMATION);
            err = -2; goto exit;
        }
        // Сохранение байта ключа для каждой пары шестнадцатеричных цифр
        if(i++ & 1)
            key[i / 2 - 1] = by & 0xff;
    }
    //  Формирование массивов для предварительного расширенного ключа
    gen_tabs();
    if(RadioButton1->Checked==true) //  Шифрование
    {
        fin = fopen(MyFName1.c_str(), "rb");
        fout = fopen(MyFName3.c_str(), "wb");

        ade_enc_key(key, k_sch1);
        form_M1(k_sch1);
        ade_enc_key_R(key, k_sch);
        err = encfile(fin, fout, k_sch, MyFName1.c_str(), MyFName3.c_str());
        fclose(fout);
        fclose(fin);
    }
                                  else   // Рашифровывание в режиме CBC
    {
        if(RadioButton2->Checked==true) //  Расшифровывание
        {
            fin = fopen(MyFName3.c_str(), "rb");
            fout = fopen(MyFName4.c_str(), "wb");

            ade_enc_key(key, k_sch1);
            form_M2(k_sch1);
```

```
 ade_enc_key_R(key, k_sch);
            err = decfile(fin, fout, k_sch, MyFName3.c_str(), MyFName4.c_str());
            fclose(fout);
            fclose(fin);
        }
        else
        {
            if(kod_dec==1)
            {
                fin = fopen(MyFName1.c_str(), "rb");
                fout = fopen(MyFName3.c_str(), "wb");

                ade_enc_key(key, k_sch1);
                form_M1(k_sch1);
                ade_enc_key_R(key, k_sch);
                err = encfile(fin, fout, k_sch, MyFName1.c_str(), MyFName3.c_str());
                fclose(fout);
                fclose(fin);
            }
            else
            {
                if(kod_dec==2)
                {
                    fin = fopen(MyFName3.c_str(), "rb");
                    fout = fopen(MyFName4.c_str(), "wb");

                    ade_enc_key(key, k_sch1);
                    form_M2(k_sch1);
                    ade_enc_key_R(key, k_sch);
                    err = decfile(fin, fout, k_sch, MyFName3.c_str(), MyFName4.c_str());
                    fclose(fout);
                    fclose(fin);
                }
            }
        }
    }
exit:
    if(err == READ_ERROR)
        Application->MessageBox("Ошибка чтения \n"
                                "из входного файла!!!",
                                "Сообщение об ошибке",
                                MB_OK + MB_ICONINFORMATION);
    if(err == WRITE_ERROR)
        Application->MessageBox("Ошибка записи \n"
                                "в выходной файл!!!",
                                "Сообщение об ошибке",
                                MB_OK + MB_ICONINFORMATION);
}

unsigned long ade_enc_key(const unsigned char in_key[], ade_32t k_sch[])
{
       ade_32t    ss[4];

       k_sch[0] = ss[0] =  (ade_32t)(in_key[0]  << 24) | (ade_32t)(in_key[1]  << 16) |
                                          (ade_32t)(in_key[2]  <<  8) | in_key[3];
       k_sch[1] = ss[1] =  (ade_32t)(in_key[4]  << 24) | (ade_32t)(in_key[5]  << 16) |
                                          (ade_32t)(in_key[6]  <<  8) | in_key[7];
       k_sch[2] = ss[2] =  (ade_32t)(in_key[8]  << 24) | (ade_32t)(in_key[9]  << 16) |
                                          (ade_32t)(in_key[10] <<  8) | in_key[11];
       k_sch[3] = ss[3] =  (ade_32t)(in_key[12] << 24) | (ade_32t)(in_key[13] << 16) |
                                    (ade_32t)(in_key[14] <<  8) | in_key[15];

    k_sch[4] = ss[0]  ^= ( fl_tab[0][(ade_08t)(ss[3] >> 16)]
                         ^ fl_tab[1][(ade_08t)(ss[3] >>  8)]
                         ^ fl_tab[2][(ade_08t) ss[3]]
                         ^ fl_tab[3][(ade_08t)(ss[3] >> 24)])
                         ^ rcon_tab[0];
    k_sch[5] = ss[1]  ^= ss[0];
    k_sch[6] = ss[2]  ^= ss[1];
    k_sch[7] = ss[3]  ^= ss[2];

    k_sch[8] = ss[0]  ^= ( fl_tab[0][(ade_08t)(ss[3] >> 16)]
                           ^ fl_tab[1][(ade_08t)(ss[3] >>  8)]
                           ^ fl_tab[2][(ade_08t) ss[3]]
                           ^ fl_tab[3][(ade_08t)(ss[3] >> 24)])
                           ^ rcon_tab[1];
```

```
k_sch[9]  = ss[1] ^= ss[0];
  k_sch[10] = ss[2] ^= ss[1];
  k_sch[11] = ss[3] ^= ss[2];

  k_sch[12] = ss[0] ^= ( fl_tab[0][(ade_08t)(ss[3] >> 16)]
                          ^ fl_tab[1][(ade_08t)(ss[3] >>  8)]
                           ^ fl_tab[2][(ade_08t) ss[3]]
                           ^ fl_tab[3][(ade_08t)(ss[3] >> 24)])
                           ^ rcon_tab[2];
  k_sch[13] = ss[1] ^= ss[0];
  k_sch[14] = ss[2] ^= ss[1];
  k_sch[15] = ss[3] ^= ss[2];

  k_sch[16] = ss[0]^= ( fl_tab[0][(ade_08t)(ss[3] >> 16)]
                       ^ fl_tab[1][(ade_08t)(ss[3] >>  8)]
                       ^ fl_tab[2][(ade_08t) ss[3]]
                       ^ fl_tab[3][(ade_08t)(ss[3] >> 24)])
                       ^ rcon_tab[3];
  k_sch[17] = ss[1] ^= ss[0];
  k_sch[18] = ss[2] ^= ss[1];
  k_sch[19] = ss[3] ^= ss[2];

  k_sch[20] = ss[0]^= ( fl_tab[0][(ade_08t)(ss[3] >> 16)]
                       ^ fl_tab[1][(ade_08t)(ss[3] >>  8)]
                       ^ fl_tab[2][(ade_08t) ss[3]]
                       ^ fl_tab[3][(ade_08t)(ss[3] >> 24)])
                       ^ rcon_tab[4];
  k_sch[21] = ss[1] ^= ss[0];
  k_sch[22] = ss[2] ^= ss[1];
                        k_sch[23] = ss[3] ^= ss[2];

  k_sch[24] = ss[0]^= ( fl_tab[0][(ade_08t)(ss[3] >> 16)]
                       ^ fl_tab[1][(ade_08t)(ss[3] >>  8)]
                       ^ fl_tab[2][(ade_08t) ss[3]]
                       ^ fl_tab[3][(ade_08t)(ss[3] >> 24)])
                       ^ rcon_tab[5];
  k_sch[25] = ss[1] ^= ss[0];
  k_sch[26] = ss[2] ^= ss[1];
  k_sch[27] = ss[3] ^= ss[2];

  k_sch[28] = ss[0]^= ( fl_tab[0][(ade_08t)(ss[3] >> 16)]
                       ^ fl_tab[1][(ade_08t)(ss[3] >>  8)]
                       ^ fl_tab[2][(ade_08t) ss[3]]
                       ^ fl_tab[3][(ade_08t)(ss[3] >> 24)])
                       ^ rcon_tab[6];
  k_sch[29] = ss[1] ^= ss[0];
  k_sch[30] = ss[2] ^= ss[1];
  k_sch[31] = ss[3] ^= ss[2];

  k_sch[32] = ss[0]^= ( fl_tab[0][(ade_08t)(ss[3] >> 16)]
                           ^ fl_tab[1][(ade_08t)(ss[3] >>  8)]
                       ^ fl_tab[2][(ade_08t) ss[3]]
                       ^ fl_tab[3][(ade_08t)(ss[3] >> 24)])
                       ^ rcon_tab[7];
  k_sch[33] = ss[1] ^= ss[0];
  k_sch[34] = ss[2] ^= ss[1];
  k_sch[35] = ss[3] ^= ss[2];

  k_sch[36] = ss[0]^= ( fl_tab[0][(ade_08t)(ss[3] >> 16)]
                        ^ fl_tab[1][(ade_08t)(ss[3] >>  8)]
                       ^ fl_tab[2][(ade_08t) ss[3]]
                       ^ fl_tab[3][(ade_08t)(ss[3] >> 24)])
                       ^ rcon_tab[8];
  k_sch[37] = ss[1] ^= ss[0];
  k_sch[38] = ss[2] ^= ss[1];
  k_sch[39] = ss[3] ^= ss[2];

  k_sch[40] = ss[0]^= ( fl_tab[0][(ade_08t)(ss[3] >> 16)]
                      ^ fl_tab[1][(ade_08t)(ss[3] >>  8)]
                       ^ fl_tab[2][(ade_08t) ss[3]]
                       ^ fl_tab[3][(ade_08t)(ss[3] >> 24)])
                       ^ rcon_tab[9];
  k_sch[41] = ss[1] ^= ss[0];
  k_sch[42] = ss[2] ^= ss[1];
  k_sch[43] = ss[3] ^= ss[2];
```

```
return 1;
}

void fillrand(char *buf, const int len)
{
    static unsigned __int64 a[2], mt = 1;
    static long count = 4;
    static char          r[4];
    int                  i;

    if(mt) { mt = 0; cycles((unsigned __int64 *)a); }

    for(i = 0; i < len; ++i)
    {
        if(count == 4)
        {
            *(unsigned long*)r = RAND(a[0], a[1]);
            count = 0;
        }
        buf[i] = r[count++];
    }
}

int encfile(FILE *fin, FILE *fout, ade_32t k_sch[], const char* ifn, const char* ofn)
{
    char            buf[BLOCK_LEN], dbuf[2 * BLOCK_LEN];
    fpos_t          flen;
    unsigned long   i, len, rlen;
    unsigned        j=0;

    // Задание случайного вектора инициализации IV
    fillrand(dbuf, BLOCK_LEN);
    // Определение длины файла
    fseek(fin, 0, SEEK_END);
    fgetpos(fin, &flen);
    rlen = file_len(flen);
    // Установка указателя в начало файла
    fseek(fin, 0, SEEK_SET);

    if(rlen <= BLOCK_LEN)
    {   // Если длина файла меньше или равна 16 байт
        // Чтение байтов файла в буфер и проверка длины
        len = (unsigned long) fread(dbuf + BLOCK_LEN, 1, BLOCK_LEN, fin);
        rlen -= len;
        if(rlen > 0)
            return READ_ERROR;

        // Заполнение байт файла нулями
        for(i = len; i < BLOCK_LEN; ++i)
            dbuf[i + BLOCK_LEN] = 0;

        //  Сложение по модулю 2 байт файла с вектором инициализации
        for(i = 0; i < BLOCK_LEN; ++i)
            dbuf[i + BLOCK_LEN] ^= dbuf[i];

        //  Шифрование старших 16 байт буфера
        ade_enc_blk(dbuf + BLOCK_LEN, dbuf + len, k_sch);

        len += BLOCK_LEN;

        //  Запись вектора инициализации IV и зашифрованых байтов файла
        if(fwrite(dbuf, 1, len, fout) != len)
            return WRITE_ERROR;
    }
    else
    {   //  Если длина файла больше 16 байт
        //  Запись вектора инициализации IV
        if(fwrite(dbuf, 1, BLOCK_LEN, fout) != BLOCK_LEN)
            return WRITE_ERROR;

        while(rlen > 0 && !feof(fin))
        {
 // Чтение блока и проверка длины
            len = (unsigned long)fread(buf, 1, BLOCK_LEN, fin);
            rlen -= len;
```

```
// Проверка длины блока
            if(len != BLOCK_LEN)
                return READ_ERROR;

            //  Формирование CBC цепочки перед шифрованием
            for(i = 0; i < BLOCK_LEN; ++i)
                buf[i] ^= dbuf[i];

            // Шифрование блока
            ade_enc_blk(buf, dbuf, k_sch);

            if(rlen > 0 && rlen < BLOCK_LEN)
            {
                //  Перемещение предыдущего шифрованого текста
                //  во вторую половину двойного буфера
                //  так как rlen байтов выводятся последними
                for(i = 0; i < BLOCK_LEN; ++i)
                    dbuf[i + BLOCK_LEN] = dbuf[i];

                //  Чтение оставшейся части открытого текста
                //  в первую половину буфера
                if(fread(dbuf, 1, rlen, fin) != rlen)
                    return READ_ERROR;

                //  Очистка оставшейся части первой половины буфера
                for(i = 0; i < BLOCK_LEN - rlen; ++i)
                    dbuf[rlen + i] = 0;

                //  Формирование CBC цепочки для предыдущего шифрованого текста
                for(i = 0; i < BLOCK_LEN; ++i)
                    dbuf[i] ^= dbuf[i + BLOCK_LEN];

                //  Шифрование финального блока
                ade_enc_blk(dbuf, dbuf, k_sch);

                //  Определение длины для финального вывода
                len = rlen + BLOCK_LEN; rlen = 0;
            }
            //  Запись зашифрованого блока
            if(fwrite(dbuf, 1, len, fout) != len)
                return WRITE_ERROR;
            j++;
            if(j%5000==0)
            {
                if(Form1->ProgressBar1->Position==100) Form1->ProgressBar1->Position = 0;
                    Form1->ProgressBar1->Position = Form1->ProgressBar1->Position + 1;
            }
        }
    }
    return 0;
}

int decfile(FILE *fin, FILE *fout, ade_32t k_sch[], const char* ifn, const char* ofn)
{
    char            buf1[BLOCK_LEN], buf2[BLOCK_LEN], dbuf[2 * BLOCK_LEN];
    char            *b1, *b2, *bt;
    fpos_t          flen;
    unsigned long   i, len, rlen;
    unsigned long   j;

    //  Нахождение длины файла
    fseek(fin, 0, SEEK_END);
    fgetpos(fin, &flen);
    rlen = file_len(flen);

    //  Установка указателя на начало файла
    fseek(fin, 0, SEEK_SET);

    if(rlen <= 2 * BLOCK_LEN)
    {   // Если длина файла меньше или равна 16 байт
        // читаем байты и проверяем длину
        len = (unsigned long)fread(dbuf, 1, 2 * BLOCK_LEN, fin);
        rlen -= len;
        if(rlen > 0)
            return READ_ERROR;
```

```
// Устанавливаем длину файла
        len -= BLOCK_LEN;

        // Осуществляем расшифровывание с позиции len до len + BLOCK_LEN
        ade_dec_blk(dbuf + len, dbuf + BLOCK_LEN, k_sch);

        // Обратное формирование CBC цепочки
        for(i = 0; i < len; ++i)
            dbuf[i] ^= dbuf[i + BLOCK_LEN];

        // Вывод расшифрованных байт
        if(fwrite(dbuf, 1, len, fout) != len)
            return WRITE_ERROR;
    }
    else
    {
        rlen -= BLOCK_LEN; b1 = buf1; b2 = buf2;

        // ввод вектора инициализации IV
        if(fread(b1, 1, BLOCK_LEN, fin) != BLOCK_LEN)
            return READ_ERROR;

        // Чтение зашифрованного файла
        while(rlen > 0 && !feof(fin))
        {
            // Ввод блока
            len = (unsigned long)fread(b2, 1, BLOCK_LEN, fin);

            rlen -= len;

            // Проверка прочитанной длины
            if(len != BLOCK_LEN)
                return READ_ERROR;

            // Расшифровывание входного буфера
            ade_dec_blk(b2, dbuf, k_sch);

if(rlen > 0 && rlen < BLOCK_LEN)
            {
                // Чтение последнего зашифрованного блока
                if(fread(b2, 1, rlen, fin) != rlen)
                    return READ_ERROR;

                // Добавление последнего расшифрованного блока из второй половины буфера
                for(i = rlen; i < BLOCK_LEN; ++i)
                    b2[i] = dbuf[i];

                // Расшифровывание последнего блока зашифрованного текста
                for(i = 0; i < rlen; ++i)
                    dbuf[i + BLOCK_LEN] = dbuf[i] ^ b2[i];

                // Расшифровывание предпоследнего блока открытого текста
                ade_dec_blk(b2, dbuf, k_sch);

                // Добавление длины последнего блока
                len = rlen + BLOCK_LEN; rlen = 0;
            }

            // Освобождение последней CBC цепочки
            for(i = 0; i < BLOCK_LEN; ++i)
                dbuf[i] ^= b1[i];

            // Запись расшифрованного текста
            if(fwrite(dbuf, 1, len, fout) != len)
                return WRITE_ERROR;

            // Обмен буферными указателями
            bt = b1, b1 = b2, b2 = bt;
            j++;
            if(j%5000==0)
            {
                if(Form1->ProgressBar1->Position==100) Form1->ProgressBar1->Position = 0;
                    Form1->ProgressBar1->Position = Form1->ProgressBar1->Position + 1;
            }
        }
    }
```

```
 return 0;
}

void cycles(volatile unsigned __int64 *rtn)
{
    typedef long time_t;
    time_t      _RTLENTRY _EXPFUNC time(time_t *__timer);

    time_t tt;
    tt     = time(NULL);
    rtn[0] = tt;
    rtn[1] = tt & -369691;
    return;
}

unsigned long ade_enc_blk(const unsigned char in_blk[], unsigned char out_blk[],ade_32t k_sch[])
{
                                    ade_32t        b1[4],b0[4];

    b0[0] = (((ade_32t)(in_blk[0])  << 24) | ((ade_32t)(in_blk[1])  << 16) |
             ((ade_32t)(in_blk[2])  <<  8) | in_blk[3])  ^ k_sch[0];
    b0[1] = (((ade_32t)(in_blk[4])  << 24) | ((ade_32t)(in_blk[5])  << 16) |
             ((ade_32t)(in_blk[6])  <<  8) | in_blk[7])  ^ k_sch[1];
    b0[2] = (((ade_32t)(in_blk[8])  << 24) | ((ade_32t)(in_blk[9])  << 16) |
             ((ade_32t)(in_blk[10]) <<  8) | in_blk[11]) ^ k_sch[2];
    b0[3] = (((ade_32t)(in_blk[12]) << 24) | ((ade_32t)(in_blk[13]) << 16) |
             ((ade_32t)(in_blk[14]) <<  8) | in_blk[15]) ^ k_sch[3];

//================================================================
//     Выполнение 10 циклов
//================================================================
//================================================================
//     1-й цикл
//================================================================
    b1[0] =  k_sch[4] ^ (T1[0][(ade_08t)(b0[delta[0]%4] >> 24)]
                      ^  T2[0][(ade_08t)(b0[delta[1]%4] >> 16)]
                      ^  T3[0][(ade_08t)(b0[delta[2]%4] >>  8)]
                      ^  T4[0][(ade_08t)(b0[delta[3]%4])]);
    b1[1] =  k_sch[5] ^ (T1[0][(ade_08t)(b0[(1 + delta[0])%4] >> 24)]
                      ^  T2[0][(ade_08t)(b0[(1 + delta[1])%4] >> 16)]
                      ^  T3[0][(ade_08t)(b0[(1 + delta[2])%4] >>  8)]
                      ^  T4[0][(ade_08t)(b0[(1 + delta[3])%4])]);
    b1[2] =  k_sch[6] ^ (T1[0][(ade_08t)(b0[(2 + delta[0])%4] >> 24)]
                      ^  T2[0][(ade_08t)(b0[(2 + delta[1])%4] >> 16)]
                      ^  T3[0][(ade_08t)(b0[(2 + delta[2])%4] >>  8)]
                      ^  T4[0][(ade_08t)(b0[(2 + delta[3])%4])]);
    b1[3] =  k_sch[7] ^ (T1[0][(ade_08t)(b0[(3 + delta[0])%4] >> 24)]
                      ^  T2[0][(ade_08t)(b0[(3 + delta[1])%4] >> 16)]
                      ^  T3[0][(ade_08t)(b0[(3 + delta[2])%4] >>  8)]
                      ^  T4[0][(ade_08t)(b0[(3 + delta[3])%4])]);
//================================================================
//     2-й цикл
//================================================================
    b0[0] = k_sch[8]  ^ (T1[1][(ade_08t)(b1[delta[4]%4] >> 24)]
                      ^  T2[1][(ade_08t)(b1[delta[5]%4] >> 16)]
                      ^  T3[1][(ade_08t)(b1[delta[6]%4] >>  8)]
                      ^  T4[1][(ade_08t)(b1[delta[7]%4])]);
    b0[1] =  k_sch[9] ^ (T1[1][(ade_08t)(b1[(1 + delta[4])%4] >> 24)]
                      ^  T2[1][(ade_08t)(b1[(1 + delta[5])%4] >> 16)]
                      ^  T3[1][(ade_08t)(b1[(1 + delta[6])%4] >>  8)]
                      ^  T4[1][(ade_08t)(b1[(1 + delta[7])%4])]);
    b0[2] = k_sch[10] ^ (T1[1][(ade_08t)(b1[(2 + delta[4])%4] >> 24)]
                      ^  T2[1][(ade_08t)(b1[(2 + delta[5])%4] >> 16)]
                      ^  T3[1][(ade_08t)(b1[(2 + delta[6])%4] >>  8)]
                      ^  T4[1][(ade_08t)(b1[(2 + delta[7])%4])]);
    b0[3] = k_sch[11] ^ (T1[1][(ade_08t)(b1[(3 + delta[4])%4] >> 24)]
                      ^  T2[1][(ade_08t)(b1[(3 + delta[5])%4] >> 16)]
                      ^  T3[1][(ade_08t)(b1[(3 + delta[6])%4] >>  8)]
                      ^  T4[1][(ade_08t)(b1[(3 + delta[7])%4])]);
//================================================================
//     3-й цикл
//================================================================
    b1[0] = k_sch[12] ^ (T1[2][(ade_08t)(b0[delta[8]%4]  >> 24)]
                      ^  T2[2][(ade_08t)(b0[delta[9]%4]  >> 16)]
                      ^  T3[2][(ade_08t)(b0[delta[10]%4] >>  8)]
                      ^  T4[2][(ade_08t)(b0[delta[11]%4])]);
```

```
 b1[1] = k_sch[13] ^ (T1[2][(ade_08t)(b0[(1 + delta[8])%4]  >> 24)]
      ^  T2[2][(ade_08t)(b0[(1 + delta[9])%4]  >> 16)]
                      ^  T3[2][(ade_08t)(b0[(1 + delta[10])%4] >>  8)]
                      ^  T4[2][(ade_08t)(b0[(1 + delta[11])%4])]);
    b1[2] = k_sch[14] ^ (T1[2][(ade_08t)(b0[(2 + delta[8])%4]  >> 24)]
                      ^  T2[2][(ade_08t)(b0[(2 + delta[9])%4]  >> 16)]
                      ^  T3[2][(ade_08t)(b0[(2 + delta[10])%4] >>  8)]
                      ^  T4[2][(ade_08t)(b0[(2 + delta[11])%4])]);
    b1[3] = k_sch[15] ^ (T1[2][(ade_08t)(b0[(3 + delta[8])%4]  >> 24)]
                      ^  T2[2][(ade_08t)(b0[(3 + delta[9])%4]  >> 16)]
                      ^  T3[2][(ade_08t)(b0[(3 + delta[10])%4] >>  8)]
                      ^  T4[2][(ade_08t)(b0[(3 + delta[11])%4])]);
//================================================================
//     4-й цикл
//================================================================
    b0[0] = k_sch[16] ^ (T1[3][(ade_08t)(b1[delta[12]%4] >> 24)]
                      ^  T2[3][(ade_08t)(b1[delta[13]%4] >> 16)]
                      ^  T3[3][(ade_08t)(b1[delta[14]%4] >>  8)]
                      ^  T4[3][(ade_08t)(b1[delta[15]%4])]);
    b0[1] = k_sch[17] ^ (T1[3][(ade_08t)(b1[(1 + delta[12])%4] >> 24)]
                      ^  T2[3][(ade_08t)(b1[(1 + delta[13])%4] >> 16)]
                      ^  T3[3][(ade_08t)(b1[(1 + delta[14])%4] >>  8)]
                      ^  T4[3][(ade_08t)(b1[(1 + delta[15])%4])]);
    b0[2] = k_sch[18] ^ (T1[3][(ade_08t)(b1[(2 + delta[12])%4] >> 24)]
                      ^  T2[3][(ade_08t)(b1[(2 + delta[13])%4] >> 16)]
                      ^  T3[3][(ade_08t)(b1[(2 + delta[14])%4] >>  8)]
                      ^  T4[3][(ade_08t)(b1[(2 + delta[15])%4])]);
    b0[3] = k_sch[19] ^ (T1[3][(ade_08t)(b1[(3 + delta[12])%4] >> 24)]
                      ^  T2[3][(ade_08t)(b1[(3 + delta[13])%4] >> 16)]
                      ^  T3[3][(ade_08t)(b1[(3 + delta[14])%4] >>  8)]
                      ^  T4[3][(ade_08t)(b1[(3 + delta[15])%4])]);
//================================================================
//     5-й цикл
//================================================================
    b1[0] = k_sch[20] ^ (T1[4][(ade_08t)(b0[delta[16]%4] >> 24)]
                      ^  T2[4][(ade_08t)(b0[delta[17]%4] >> 16)]
                      ^  T3[4][(ade_08t)(b0[delta[18]%4] >>  8)]
                      ^  T4[4][(ade_08t)(b0[delta[19]%4])]);
    b1[1] = k_sch[21] ^ (T1[4][(ade_08t)(b0[(1 + delta[16])%4] >> 24)]
                      ^  T2[4][(ade_08t)(b0[(1 + delta[17])%4] >> 16)]
                      ^  T3[4][(ade_08t)(b0[(1 + delta[18])%4] >>  8)]
                      ^  T4[4][(ade_08t)(b0[(1 + delta[19])%4])]);
    b1[2] = k_sch[22] ^ (T1[4][(ade_08t)(b0[(2 + delta[16])%4] >> 24)]
                      ^  T2[4][(ade_08t)(b0[(2 + delta[17])%4] >> 16)]
                      ^  T3[4][(ade_08t)(b0[(2 + delta[18])%4] >>  8)]
                      ^  T4[4][(ade_08t)(b0[(2 + delta[19])%4])]);
    b1[3] = k_sch[23] ^ (T1[4][(ade_08t)(b0[(3 + delta[16])%4] >> 24)]
                      ^  T2[4][(ade_08t)(b0[(3 + delta[17])%4] >> 16)]
                      ^  T3[4][(ade_08t)(b0[(3 + delta[18])%4] >>  8)]
                      ^  T4[4][(ade_08t)(b0[(3 + delta[19])%4])]);
//================================================================
//     6-й цикл
//================================================================
    b0[0] = k_sch[24] ^ (T1[5][(ade_08t)(b1[delta[20]%4] >> 24)]
                      ^  T2[5][(ade_08t)(b1[delta[21]%4] >> 16)]
                      ^  T3[5][(ade_08t)(b1[delta[22]%4] >>  8)]
                      ^  T4[5][(ade_08t)(b1[delta[23]%4])]);
    b0[1] = k_sch[25] ^ (T1[5][(ade_08t)(b1[(1 + delta[20])%4] >> 24)]
                      ^  T2[5][(ade_08t)(b1[(1 + delta[21])%4] >> 16)]
                      ^  T3[5][(ade_08t)(b1[(1 + delta[22])%4] >>  8)]
                      ^  T4[5][(ade_08t)(b1[(1 + delta[23])%4])]);
b0[2] = k_sch[26] ^ (T1[5][(ade_08t)(b1[(2 + delta[20])%4] >> 24)]
                      ^  T2[5][(ade_08t)(b1[(2 + delta[21])%4] >> 16)]
                      ^  T3[5][(ade_08t)(b1[(2 + delta[22])%4] >>  8)]
                      ^  T4[5][(ade_08t)(b1[(2 + delta[23])%4])]);
    b0[3] = k_sch[27] ^ (T1[5][(ade_08t)(b1[(3 + delta[20])%4] >> 24)]
                      ^  T2[5][(ade_08t)(b1[(3 + delta[21])%4] >> 16)]
                      ^  T3[5][(ade_08t)(b1[(3 + delta[22])%4] >>  8)]
                      ^  T4[5][(ade_08t)(b1[(3 + delta[23])%4])]);
//================================================================
//     7-й цикл
//================================================================
    b1[0] = k_sch[28] ^ (T1[6][(ade_08t)(b0[delta[24]%4] >> 24)]
                      ^  T2[6][(ade_08t)(b0[delta[25]%4] >> 16)]
                      ^  T3[6][(ade_08t)(b0[delta[26]%4] >>  8)]
                      ^  T4[6][(ade_08t)(b0[delta[27]%4])]);
```

```
b1[1] = k_sch[29] ^ (T1[6][(ade_08t)(b0[(1 + delta[24])%4] >> 24)]
                    ^  T2[6][(ade_08t)(b0[(1 + delta[25])%4] >> 16)]
                    ^  T3[6][(ade_08t)(b0[(1 + delta[26])%4] >>  8)]
                    ^  T4[6][(ade_08t)(b0[(1 + delta[27])%4])]);
    b1[2] = k_sch[30] ^ (T1[6][(ade_08t)(b0[(2 + delta[24])%4] >> 24)]
                    ^  T2[6][(ade_08t)(b0[(2 + delta[25])%4] >> 16)]
                    ^  T3[6][(ade_08t)(b0[(2 + delta[26])%4] >>  8)]
                    ^  T4[6][(ade_08t)(b0[(2 + delta[27])%4])]);
    b1[3] = k_sch[31] ^ (T1[6][(ade_08t)(b0[(3 + delta[24])%4] >> 24)]
                    ^  T2[6][(ade_08t)(b0[(3 + delta[25])%4] >> 16)]
                    ^  T3[6][(ade_08t)(b0[(3 + delta[26])%4] >>  8)]
                    ^  T4[6][(ade_08t)(b0[(3 + delta[27])%4])]);
//================================================================
//     8-й цикл
//================================================================
    b0[0] = k_sch[32] ^ (T1[7][(ade_08t)(b1[delta[28]%4] >> 24)]
                    ^  T2[7][(ade_08t)(b1[delta[29]%4] >> 16)]
                    ^  T3[7][(ade_08t)(b1[delta[30]%4] >>  8)]
                    ^  T4[7][(ade_08t)(b1[delta[31]%4])]);
    b0[1] = k_sch[33] ^ (T1[7][(ade_08t)(b1[(1 + delta[28])%4] >> 24)]
                    ^  T2[7][(ade_08t)(b1[(1 + delta[29])%4] >> 16)]
                    ^  T3[7][(ade_08t)(b1[(1 + delta[30])%4] >>  8)]
                    ^  T4[7][(ade_08t)(b1[(1 + delta[31])%4])]);
    b0[2] = k_sch[34] ^ (T1[7][(ade_08t)(b1[(2 + delta[28])%4] >> 24)]
                    ^  T2[7][(ade_08t)(b1[(2 + delta[29])%4] >> 16)]
                    ^  T3[7][(ade_08t)(b1[(2 + delta[30])%4] >>  8)]
                    ^  T4[7][(ade_08t)(b1[(2 + delta[31])%4])]);
    b0[3] = k_sch[35] ^ (T1[7][(ade_08t)(b1[(3 + delta[28])%4] >> 24)]
                    ^  T2[7][(ade_08t)(b1[(3 + delta[29])%4] >> 16)]
                    ^  T3[7][(ade_08t)(b1[(3 + delta[30])%4] >>  8)]
                    ^  T4[7][(ade_08t)(b1[(3 + delta[31])%4])]);
//================================================================
//     9-й цикл
//================================================================
    b1[0] = k_sch[36] ^ (T1[8][(ade_08t)(b0[delta[32]%4] >> 24)]
                    ^  T2[8][(ade_08t)(b0[delta[33]%4] >> 16)]
                    ^  T3[8][(ade_08t)(b0[delta[34]%4] >>  8)]
                    ^  T4[8][(ade_08t)(b0[delta[35]%4])]);
    b1[1] = k_sch[37] ^ (T1[8][(ade_08t)(b0[(1 + delta[32])%4] >> 24)]
                    ^  T2[8][(ade_08t)(b0[(1 + delta[33])%4] >> 16)]
                    ^  T3[8][(ade_08t)(b0[(1 + delta[34])%4] >>  8)]
                    ^  T4[8][(ade_08t)(b0[(1 + delta[35])%4])]);
    b1[2] = k_sch[38] ^ (T1[8][(ade_08t)(b0[(2 + delta[32])%4] >> 24)]
                    ^  T2[8][(ade_08t)(b0[(2 + delta[33])%4] >> 16)]
                    ^  T3[8][(ade_08t)(b0[(2 + delta[34])%4] >>  8)]
    ^  T4[8][(ade_08t)(b0[(2 + delta[35])%4])]);
    b1[3] = k_sch[39] ^ (T1[8][(ade_08t)(b0[(3 + delta[32])%4] >> 24)]
                    ^  T2[8][(ade_08t)(b0[(3 + delta[33])%4] >> 16)]
                    ^  T3[8][(ade_08t)(b0[(3 + delta[34])%4] >>  8)]
                    ^  T4[8][(ade_08t)(b0[(3 + delta[35])%4])]);
//================================================================
//     10-й цикл
//================================================================
    b0[0] = k_sch[40] ^ (L1[9][(ade_08t)(b1[delta[36]%4] >> 24)]
                    ^  L2[9][(ade_08t)(b1[delta[37]%4] >> 16)]
                    ^  L3[9][(ade_08t)(b1[delta[38]%4] >>  8)]
                    ^  L4[9][(ade_08t)(b1[delta[39]%4])]);
    b0[1] = k_sch[41] ^ (L1[9][(ade_08t)(b1[(1 + delta[36])%4] >> 24)]
                    ^  L2[9][(ade_08t)(b1[(1 + delta[37])%4] >> 16)]
                    ^  L3[9][(ade_08t)(b1[(1 + delta[38])%4] >>  8)]
                    ^  L4[9][(ade_08t)(b1[(1 + delta[39])%4])]);
    b0[2] = k_sch[42] ^ (L1[9][(ade_08t)(b1[(2 + delta[36])%4] >> 24)]
                    ^  L2[9][(ade_08t)(b1[(2 + delta[37])%4] >> 16)]
                    ^  L3[9][(ade_08t)(b1[(2 + delta[38])%4] >>  8)]
                    ^  L4[9][(ade_08t)(b1[(2 + delta[39])%4])]);
    b0[3] = k_sch[43] ^ (L1[9][(ade_08t)(b1[(3 + delta[36])%4] >> 24)]
                    ^  L2[9][(ade_08t)(b1[(3 + delta[37])%4] >> 16)]
                    ^  L3[9][(ade_08t)(b1[(3 + delta[38])%4] >>  8)]
                    ^  L4[9][(ade_08t)(b1[(3 + delta[39])%4])]);

    out_blk[0]  = (ade_08t)(b0[0] >> 24);    out_blk[1]  = (ade_08t)(b0[0] >> 16);
    out_blk[2]  = (ade_08t)(b0[0] >>  8);    out_blk[3]  = (ade_08t) b0[0];
    out_blk[4]  = (ade_08t)(b0[1] >> 24);    out_blk[5]  = (ade_08t)(b0[1] >> 16);
    out_blk[6]  = (ade_08t)(b0[1] >>  8);    out_blk[7]  = (ade_08t) b0[1];
    out_blk[8]  = (ade_08t)(b0[2] >> 24);    out_blk[9]  = (ade_08t)(b0[2] >> 16);
    out_blk[10] = (ade_08t)(b0[2] >>  8);    out_blk[11] = (ade_08t) b0[2];
```

```
out_blk[12] = (ade_08t)(b0[3] >> 24);    out_blk[13] = (ade_08t)(b0[3] >> 16);
    out_blk[14] = (ade_08t)(b0[3] >>  8);    out_blk[15] = (ade_08t) b0[3];

    return 1;
}

unsigned long ade_dec_blk(const unsigned char in_blk[], unsigned char out_blk[],ade_32t k_sch[])
{
        ade_32t                      b1[4],b0[4];

    b0[0] = (((ade_32t)(in_blk[0])  << 24) | ((ade_32t)(in_blk[1])  << 16) |
             ((ade_32t)(in_blk[2])  <<  8) | in_blk[3])  ^ k_sch[40];
    b0[1] = (((ade_32t)(in_blk[4])  << 24) | ((ade_32t)(in_blk[5])  << 16) |
             ((ade_32t)(in_blk[6])  <<  8) | in_blk[7])  ^ k_sch[41];
    b0[2] = (((ade_32t)(in_blk[8])  << 24) | ((ade_32t)(in_blk[9])  << 16) |
             ((ade_32t)(in_blk[10]) <<  8) | in_blk[11]) ^ k_sch[42];
    b0[3] = (((ade_32t)(in_blk[12]) << 24) | ((ade_32t)(in_blk[13]) << 16) |
             ((ade_32t)(in_blk[14]) <<  8) | in_blk[15]) ^ k_sch[43];

//================================================================
//     Выполнение 10 циклов
//================================================================
//================================================================
//     1-й цикл
//================================================================
    b1[0] = (OL1[9][(ade_08t)(b0[(4-delta[36])%4] >> 24)]
          ^  OL2[9][(ade_08t)(b0[(4-delta[37])%4] >> 16)]
          ^  OL3[9][(ade_08t)(b0[(4-delta[38])%4] >>  8)]
          ^  OL4[9][(ade_08t)(b0[(4-delta[39])%4])]) ^ k_sch[36];
    b1[1] = (OL1[9][(ade_08t)(b0[(5-delta[36])%4] >> 24)]
          ^  OL2[9][(ade_08t)(b0[(5-delta[37])%4] >> 16)]
          ^  OL3[9][(ade_08t)(b0[(5-delta[38])%4] >>  8)]
          ^  OL4[9][(ade_08t)(b0[(5-delta[39])%4])]) ^ k_sch[37];
    b1[2] = (OL1[9][(ade_08t)(b0[(6-delta[36])%4] >> 24)]
          ^  OL2[9][(ade_08t)(b0[(6-delta[37])%4] >> 16)]
          ^  OL3[9][(ade_08t)(b0[(6-delta[38])%4] >>  8)]
          ^  OL4[9][(ade_08t)(b0[(6-delta[39])%4])]) ^ k_sch[38];
    b1[3] = (OL1[9][(ade_08t)(b0[(7-delta[36])%4] >> 24)]
          ^  OL2[9][(ade_08t)(b0[(7-delta[37])%4] >> 16)]
          ^  OL3[9][(ade_08t)(b0[(7-delta[38])%4] >>  8)]
          ^  OL4[9][(ade_08t)(b0[(7-delta[39])%4])]) ^ k_sch[39];
//================================================================
//     2-й цикл
//================================================================
    b0[0] = (OT1[8][(ade_08t)(b1[0] >> 24)]
          ^  OT2[8][(ade_08t)(b1[0] >> 16)]
          ^  OT3[8][(ade_08t)(b1[0] >> 8)]
          ^  OT4[8][(ade_08t) b1[0]]);
    b0[1] = (OT1[8][(ade_08t)(b1[1] >> 24)]
          ^  OT2[8][(ade_08t)(b1[1] >> 16)]
          ^  OT3[8][(ade_08t)(b1[1] >> 8)]
          ^  OT4[8][(ade_08t) b1[1]]);
    b0[2] = (OT1[8][(ade_08t)(b1[2] >> 24)]
          ^  OT2[8][(ade_08t)(b1[2] >> 16)]
          ^  OT3[8][(ade_08t)(b1[2] >> 8)]
          ^  OT4[8][(ade_08t) b1[2]]);
    b0[3] = (OT1[8][(ade_08t)(b1[3] >> 24)]
          ^  OT2[8][(ade_08t)(b1[3] >> 16)]
          ^  OT3[8][(ade_08t)(b1[3] >> 8)]
          ^  OT4[8][(ade_08t) b1[3]]);

    b1[0] = (OL1[8][(ade_08t)(b0[(4-delta[32])%4] >> 24)]
          ^  OL2[8][(ade_08t)(b0[(4-delta[33])%4] >> 16)]
          ^  OL3[8][(ade_08t)(b0[(4-delta[34])%4] >>  8)]
          ^  OL4[8][(ade_08t)(b0[(4-delta[35])%4])]) ^ k_sch[32];
    b1[1] = (OL1[8][(ade_08t)(b0[(5-delta[32])%4] >> 24)]
          ^  OL2[8][(ade_08t)(b0[(5-delta[33])%4] >> 16)]
          ^  OL3[8][(ade_08t)(b0[(5-delta[34])%4] >>  8)]
          ^  OL4[8][(ade_08t)(b0[(5-delta[35])%4])]) ^ k_sch[33];
    b1[2] = (OL1[8][(ade_08t)(b0[(6-delta[32])%4] >> 24)]
          ^  OL2[8][(ade_08t)(b0[(6-delta[33])%4] >> 16)]
          ^  OL3[8][(ade_08t)(b0[(6-delta[34])%4] >>  8)]
          ^  OL4[8][(ade_08t)(b0[(6-delta[35])%4])]) ^ k_sch[34];
    b1[3] = (OL1[8][(ade_08t)(b0[(7-delta[32])%4] >> 24)]
          ^  OL2[8][(ade_08t)(b0[(7-delta[33])%4] >> 16)]
          ^  OL3[8][(ade_08t)(b0[(7-delta[34])%4] >>  8)]
```

```
  ^  OL4[8][(ade_08t)(b0[(7-delta[35])%4])]) ^ k_sch[35];
//================================================================
//     3-й цикл
//================================================================
    b0[0] = (OT1[7][(ade_08t)(b1[0] >> 24)]
          ^  OT2[7][(ade_08t)(b1[0] >> 16)]
          ^  OT3[7][(ade_08t)(b1[0] >> 8)]
          ^  OT4[7][(ade_08t) b1[0]]);

b0[1] = (OT1[7][(ade_08t)(b1[1] >> 24)]
          ^  OT2[7][(ade_08t)(b1[1] >> 16)]
          ^  OT3[7][(ade_08t)(b1[1] >> 8)]
          ^  OT4[7][(ade_08t) b1[1]]);
    b0[2] = (OT1[7][(ade_08t)(b1[2] >> 24)]
          ^  OT2[7][(ade_08t)(b1[2] >> 16)]
          ^  OT3[7][(ade_08t)(b1[2] >> 8)]
          ^  OT4[7][(ade_08t) b1[2]]);
    b0[3] = (OT1[7][(ade_08t)(b1[3] >> 24)]
          ^  OT2[7][(ade_08t)(b1[3] >> 16)]
          ^  OT3[7][(ade_08t)(b1[3] >> 8)]
          ^  OT4[7][(ade_08t) b1[3]]);

    b1[0] = (OL1[7][(ade_08t)(b0[(4-delta[28])%4] >> 24)]
          ^  OL2[7][(ade_08t)(b0[(4-delta[29])%4] >> 16)]
          ^  OL3[7][(ade_08t)(b0[(4-delta[30])%4] >>  8)]
          ^  OL4[7][(ade_08t)(b0[(4-delta[31])%4])]) ^ k_sch[28];
    b1[1] = (OL1[7][(ade_08t)(b0[(5-delta[28])%4] >> 24)]
          ^  OL2[7][(ade_08t)(b0[(5-delta[29])%4] >> 16)]
          ^  OL3[7][(ade_08t)(b0[(5-delta[30])%4] >>  8)]
          ^  OL4[7][(ade_08t)(b0[(5-delta[31])%4])]) ^ k_sch[29];
    b1[2] = (OL1[7][(ade_08t)(b0[(6-delta[28])%4] >> 24)]
          ^  OL2[7][(ade_08t)(b0[(6-delta[29])%4] >> 16)]
          ^  OL3[7][(ade_08t)(b0[(6-delta[30])%4] >>  8)]
          ^  OL4[7][(ade_08t)(b0[(6-delta[31])%4])]) ^ k_sch[30];
    b1[3] = (OL1[7][(ade_08t)(b0[(7-delta[28])%4] >> 24)]
          ^  OL2[7][(ade_08t)(b0[(7-delta[29])%4] >> 16)]
          ^  OL3[7][(ade_08t)(b0[(7-delta[30])%4] >>  8)]
          ^  OL4[7][(ade_08t)(b0[(7-delta[31])%4])]) ^ k_sch[31];
//================================================================
//     4-й цикл
//================================================================
    b0[0] = (OT1[6][(ade_08t)(b1[0] >> 24)]
          ^  OT2[6][(ade_08t)(b1[0] >> 16)]
          ^  OT3[6][(ade_08t)(b1[0] >> 8)]
          ^  OT4[6][(ade_08t) b1[0]]);
    b0[1] = (OT1[6][(ade_08t)(b1[1] >> 24)]
          ^  OT2[6][(ade_08t)(b1[1] >> 16)]
          ^  OT3[6][(ade_08t)(b1[1] >> 8)]
          ^  OT4[6][(ade_08t) b1[1]]);
    b0[2] = (OT1[6][(ade_08t)(b1[2] >> 24)]
          ^  OT2[6][(ade_08t)(b1[2] >> 16)]
          ^  OT3[6][(ade_08t)(b1[2] >> 8)]
          ^  OT4[6][(ade_08t) b1[2]]);
    b0[3] = (OT1[6][(ade_08t)(b1[3] >> 24)]
          ^  OT2[6][(ade_08t)(b1[3] >> 16)]
          ^  OT3[6][(ade_08t)(b1[3] >> 8)]
          ^  OT4[6][(ade_08t) b1[3]]);

    b1[0] = (OL1[6][(ade_08t)(b0[(4-delta[24])%4] >> 24)]
          ^  OL2[6][(ade_08t)(b0[(4-delta[25])%4] >> 16)]
          ^  OL3[6][(ade_08t)(b0[(4-delta[26])%4] >>  8)]
          ^  OL4[6][(ade_08t)(b0[(4-delta[27])%4])]) ^ k_sch[24];
    b1[1] = (OL1[6][(ade_08t)(b0[(5-delta[24])%4] >> 24)]
          ^  OL2[6][(ade_08t)(b0[(5-delta[25])%4] >> 16)]
          ^  OL3[6][(ade_08t)(b0[(5-delta[26])%4] >>  8)]
          ^  OL4[6][(ade_08t)(b0[(5-delta[27])%4])]) ^ k_sch[25];
    b1[2] = (OL1[6][(ade_08t)(b0[(6-delta[24])%4] >> 24)]
          ^  OL2[6][(ade_08t)(b0[(6-delta[25])%4] >> 16)]
^  OL3[6][(ade_08t)(b0[(6-delta[26])%4] >>  8)]
          ^  OL4[6][(ade_08t)(b0[(6-delta[27])%4])]) ^ k_sch[26];
    b1[3] = (OL1[6][(ade_08t)(b0[(7-delta[24])%4] >> 24)]
          ^  OL2[6][(ade_08t)(b0[(7-delta[25])%4] >> 16)]
          ^  OL3[6][(ade_08t)(b0[(7-delta[26])%4] >>  8)]
          ^  OL4[6][(ade_08t)(b0[(7-delta[27])%4])]) ^ k_sch[27];
//================================================================
//     5-й цикл
```

```
//=============================================================
    b0[0] = (OT1[5][(ade_08t)(b1[0] >> 24)]
          ^  OT2[5][(ade_08t)(b1[0] >> 16)]
          ^  OT3[5][(ade_08t)(b1[0] >> 8)]
          ^  OT4[5][(ade_08t) b1[0]]);
    b0[1] = (OT1[5][(ade_08t)(b1[1] >> 24)]
          ^  OT2[5][(ade_08t)(b1[1] >> 16)]
          ^  OT3[5][(ade_08t)(b1[1] >> 8)]
          ^  OT4[5][(ade_08t) b1[1]]);
    b0[2] = (OT1[5][(ade_08t)(b1[2] >> 24)]
          ^  OT2[5][(ade_08t)(b1[2] >> 16)]
          ^  OT3[5][(ade_08t)(b1[2] >> 8)]
          ^  OT4[5][(ade_08t) b1[2]]);
    b0[3] = (OT1[5][(ade_08t)(b1[3] >> 24)]
          ^  OT2[5][(ade_08t)(b1[3] >> 16)]
          ^  OT3[5][(ade_08t)(b1[3] >> 8)]
          ^  OT4[5][(ade_08t) b1[3]]);

    b1[0] = (OL1[5][(ade_08t)(b0[(4-delta[20])%4] >> 24)]
          ^  OL2[5][(ade_08t)(b0[(4-delta[21])%4] >> 16)]
          ^  OL3[5][(ade_08t)(b0[(4-delta[22])%4] >>  8)]
          ^  OL4[5][(ade_08t)(b0[(4-delta[23])%4])]) ^ k_sch[20];
    b1[1] = (OL1[5][(ade_08t)(b0[(5-delta[20])%4] >> 24)]
          ^  OL2[5][(ade_08t)(b0[(5-delta[21])%4] >> 16)]
          ^  OL3[5][(ade_08t)(b0[(5-delta[22])%4] >>  8)]
          ^  OL4[5][(ade_08t)(b0[(5-delta[23])%4])]) ^ k_sch[21];
    b1[2] = (OL1[5][(ade_08t)(b0[(6-delta[20])%4] >> 24)]
          ^  OL2[5][(ade_08t)(b0[(6-delta[21])%4] >> 16)]
          ^  OL3[5][(ade_08t)(b0[(6-delta[22])%4] >>  8)]
          ^  OL4[5][(ade_08t)(b0[(6-delta[23])%4])]) ^ k_sch[22];
    b1[3] = (OL1[5][(ade_08t)(b0[(7-delta[20])%4] >> 24)]
          ^  OL2[5][(ade_08t)(b0[(7-delta[21])%4] >> 16)]
          ^  OL3[5][(ade_08t)(b0[(7-delta[22])%4] >>  8)]
          ^  OL4[5][(ade_08t)(b0[(7-delta[23])%4])]) ^ k_sch[23];
//=============================================================
//     6-й цикл
//=============================================================
    b0[0] = (OT1[4][(ade_08t)(b1[0] >> 24)]
          ^  OT2[4][(ade_08t)(b1[0] >> 16)]
          ^  OT3[4][(ade_08t)(b1[0] >> 8)]
          ^  OT4[4][(ade_08t) b1[0]]);
    b0[1] = (OT1[4][(ade_08t)(b1[1] >> 24)]
          ^  OT2[4][(ade_08t)(b1[1] >> 16)]
          ^  OT3[4][(ade_08t)(b1[1] >> 8)]
          ^  OT4[4][(ade_08t) b1[1]]);
    b0[2] = (OT1[4][(ade_08t)(b1[2] >> 24)]
          ^  OT2[4][(ade_08t)(b1[2] >> 16)]
          ^  OT3[4][(ade_08t)(b1[2] >> 8)]
          ^  OT4[4][(ade_08t) b1[2]]);
    b0[3] = (OT1[4][(ade_08t)(b1[3] >> 24)]
          ^  OT2[4][(ade_08t)(b1[3] >> 16)]
          ^  OT3[4][(ade_08t)(b1[3] >> 8)]
^  OT4[4][(ade_08t) b1[3]]);

    b1[0] = (OL1[4][(ade_08t)(b0[(4-delta[16])%4] >> 24)]
          ^  OL2[4][(ade_08t)(b0[(4-delta[17])%4] >> 16)]
          ^  OL3[4][(ade_08t)(b0[(4-delta[18])%4] >>  8)]
          ^  OL4[4][(ade_08t)(b0[(4-delta[19])%4])]) ^ k_sch[16];
    b1[1] = (OL1[4][(ade_08t)(b0[(5-delta[16])%4] >> 24)]
          ^  OL2[4][(ade_08t)(b0[(5-delta[17])%4] >> 16)]
          ^  OL3[4][(ade_08t)(b0[(5-delta[18])%4] >>  8)]
          ^  OL4[4][(ade_08t)(b0[(5-delta[19])%4])]) ^ k_sch[17];
    b1[2] = (OL1[4][(ade_08t)(b0[(6-delta[16])%4] >> 24)]
          ^  OL2[4][(ade_08t)(b0[(6-delta[17])%4] >> 16)]
          ^  OL3[4][(ade_08t)(b0[(6-delta[18])%4] >>  8)]
          ^  OL4[4][(ade_08t)(b0[(6-delta[19])%4])]) ^ k_sch[18];
    b1[3] = (OL1[4][(ade_08t)(b0[(7-delta[16])%4] >> 24)]
          ^  OL2[4][(ade_08t)(b0[(7-delta[17])%4] >> 16)]
          ^  OL3[4][(ade_08t)(b0[(7-delta[18])%4] >>  8)]
          ^  OL4[4][(ade_08t)(b0[(7-delta[19])%4])]) ^ k_sch[19];
//=============================================================
//     7-й цикл
//=============================================================
    b0[0] = (OT1[3][(ade_08t)(b1[0] >> 24)]
          ^  OT2[3][(ade_08t)(b1[0] >> 16)]
          ^  OT3[3][(ade_08t)(b1[0] >> 8)]
```

```
 ^  OT4[3][(ade_08t) b1[0]]);
    b0[1] = (OT1[3][(ade_08t)(b1[1] >> 24)]
          ^  OT2[3][(ade_08t)(b1[1] >> 16)]
          ^  OT3[3][(ade_08t)(b1[1] >> 8)]
          ^  OT4[3][(ade_08t) b1[1]]);
    b0[2] = (OT1[3][(ade_08t)(b1[2] >> 24)]
          ^  OT2[3][(ade_08t)(b1[2] >> 16)]
          ^  OT3[3][(ade_08t)(b1[2] >> 8)]
          ^  OT4[3][(ade_08t) b1[2]]);
b0[3] = (OT1[3][(ade_08t)(b1[3] >> 24)]
          ^  OT2[3][(ade_08t)(b1[3] >> 16)]
          ^  OT3[3][(ade_08t)(b1[3] >> 8)]
          ^  OT4[3][(ade_08t) b1[3]]);

    b1[0] = (OL1[3][(ade_08t)(b0[(4-delta[12])%4] >> 24)]
          ^  OL2[3][(ade_08t)(b0[(4-delta[13])%4] >> 16)]
          ^  OL3[3][(ade_08t)(b0[(4-delta[14])%4] >>  8)]
          ^  OL4[3][(ade_08t)(b0[(4-delta[15])%4])]) ^ k_sch[12];
    b1[1] = (OL1[3][(ade_08t)(b0[(5-delta[12])%4] >> 24)]
          ^  OL2[3][(ade_08t)(b0[(5-delta[13])%4] >> 16)]
          ^  OL3[3][(ade_08t)(b0[(5-delta[14])%4] >>  8)]
          ^  OL4[3][(ade_08t)(b0[(5-delta[15])%4])]) ^ k_sch[13];
    b1[2] = (OL1[3][(ade_08t)(b0[(6-delta[12])%4] >> 24)]
          ^  OL2[3][(ade_08t)(b0[(6-delta[13])%4] >> 16)]
          ^  OL3[3][(ade_08t)(b0[(6-delta[14])%4] >>  8)]
          ^  OL4[3][(ade_08t)(b0[(6-delta[15])%4])]) ^ k_sch[14];
    b1[3] = (OL1[3][(ade_08t)(b0[(7-delta[12])%4] >> 24)]
          ^  OL2[3][(ade_08t)(b0[(7-delta[13])%4] >> 16)]
          ^  OL3[3][(ade_08t)(b0[(7-delta[14])%4] >>  8)]
          ^  OL4[3][(ade_08t)(b0[(7-delta[15])%4])]) ^ k_sch[15];
//================================================================
//     8-й цикл
//================================================================
    b0[0] = (OT1[2][(ade_08t)(b1[0] >> 24)]
          ^  OT2[2][(ade_08t)(b1[0] >> 16)]
          ^  OT3[2][(ade_08t)(b1[0] >> 8)]

  ^  OT4[2][(ade_08t) b1[0]]);
    b0[1] = (OT1[2][(ade_08t)(b1[1] >> 24)]
          ^  OT2[2][(ade_08t)(b1[1] >> 16)]
          ^  OT3[2][(ade_08t)(b1[1] >> 8)]
          ^  OT4[2][(ade_08t) b1[1]]);
    b0[2] = (OT1[2][(ade_08t)(b1[2] >> 24)]
          ^  OT2[2][(ade_08t)(b1[2] >> 16)]
          ^  OT3[2][(ade_08t)(b1[2] >> 8)]
          ^  OT4[2][(ade_08t) b1[2]]);
    b0[3] = (OT1[2][(ade_08t)(b1[3] >> 24)]
          ^  OT2[2][(ade_08t)(b1[3] >> 16)]
          ^  OT3[2][(ade_08t)(b1[3] >> 8)]
          ^  OT4[2][(ade_08t) b1[3]]);

    b1[0] = (OL1[2][(ade_08t)(b0[(4-delta[8])%4]  >> 24)]
          ^  OL2[2][(ade_08t)(b0[(4-delta[9])%4]  >> 16)]
          ^  OL3[2][(ade_08t)(b0[(4-delta[10])%4] >>  8)]
          ^  OL4[2][(ade_08t)(b0[(4-delta[11])%4])]) ^ k_sch[8];
    b1[1] = (OL1[2][(ade_08t)(b0[(5-delta[8])%4]  >> 24)]
          ^  OL2[2][(ade_08t)(b0[(5-delta[9])%4]  >> 16)]
          ^  OL3[2][(ade_08t)(b0[(5-delta[10])%4] >>  8)]
          ^  OL4[2][(ade_08t)(b0[(5-delta[11])%4])]) ^ k_sch[9];
    b1[2] = (OL1[2][(ade_08t)(b0[(6-delta[8])%4]  >> 24)]
          ^  OL2[2][(ade_08t)(b0[(6-delta[9])%4]  >> 16)]
          ^  OL3[2][(ade_08t)(b0[(6-delta[10])%4] >>  8)]
          ^  OL4[2][(ade_08t)(b0[(6-delta[11])%4])]) ^ k_sch[10];
    b1[3] = (OL1[2][(ade_08t)(b0[(7-delta[8])%4]  >> 24)]
          ^  OL2[2][(ade_08t)(b0[(7-delta[9])%4]  >> 16)]
          ^  OL3[2][(ade_08t)(b0[(7-delta[10])%4] >>  8)]
          ^  OL4[2][(ade_08t)(b0[(7-delta[11])%4])]) ^ k_sch[11];
//================================================================
//     9-й цикл
//================================================================
    b0[0] = (OT1[1][(ade_08t)(b1[0] >> 24)]
          ^  OT2[1][(ade_08t)(b1[0] >> 16)]
          ^  OT3[1][(ade_08t)(b1[0] >> 8)]
          ^  OT4[1][(ade_08t) b1[0]]);
    b0[1] = (OT1[1][(ade_08t)(b1[1] >> 24)]
          ^  OT2[1][(ade_08t)(b1[1] >> 16)]
```

```
  ^  OT3[1][(ade_08t)(b1[1] >> 8)]
         ^  OT4[1][(ade_08t) b1[1]]);
    b0[2] = (OT1[1][(ade_08t)(b1[2] >> 24)]
         ^  OT2[1][(ade_08t)(b1[2] >> 16)]
         ^  OT3[1][(ade_08t)(b1[2] >> 8)]
         ^  OT4[1][(ade_08t) b1[2]]);
    b0[3] = (OT1[1][(ade_08t)(b1[3] >> 24)]
         ^  OT2[1][(ade_08t)(b1[3] >> 16)]
         ^  OT3[1][(ade_08t)(b1[3] >> 8)]
         ^  OT4[1][(ade_08t) b1[3]]);



b1[0] = (OL1[1][(ade_08t)(b0[(4-delta[4])%4] >> 24)]
         ^  OL2[1][(ade_08t)(b0[(4-delta[5])%4] >> 16)]
         ^  OL3[1][(ade_08t)(b0[(4-delta[6])%4] >>  8)]
         ^  OL4[1][(ade_08t)(b0[(4-delta[7])%4])]) ^ k_sch[4];
    b1[1] = (OL1[1][(ade_08t)(b0[(5-delta[4])%4] >> 24)]
         ^  OL2[1][(ade_08t)(b0[(5-delta[5])%4] >> 16)]
         ^  OL3[1][(ade_08t)(b0[(5-delta[6])%4] >>  8)]
         ^  OL4[1][(ade_08t)(b0[(5-delta[7])%4])]) ^ k_sch[5];
    b1[2] = (OL1[1][(ade_08t)(b0[(6-delta[4])%4] >> 24)]
         ^  OL2[1][(ade_08t)(b0[(6-delta[5])%4] >> 16)]

^  OL3[1][(ade_08t)(b0[(6-delta[6])%4] >>  8)]
         ^  OL4[1][(ade_08t)(b0[(6-delta[7])%4])]) ^ k_sch[6];
    b1[3] = (OL1[1][(ade_08t)(b0[(7-delta[4])%4] >> 24)]
         ^  OL2[1][(ade_08t)(b0[(7-delta[5])%4] >> 16)]
         ^  OL3[1][(ade_08t)(b0[(7-delta[6])%4] >>  8)]
         ^  OL4[1][(ade_08t)(b0[(7-delta[7])%4])]) ^ k_sch[7];
//================================================================
//     10-й цикл
//================================================================
    b0[0] = (OT1[0][(ade_08t)(b1[0] >> 24)]
         ^  OT2[0][(ade_08t)(b1[0] >> 16)]
         ^  OT3[0][(ade_08t)(b1[0] >> 8)]
         ^  OT4[0][(ade_08t) b1[0]]);
    b0[1] = (OT1[0][(ade_08t)(b1[1] >> 24)]
         ^  OT2[0][(ade_08t)(b1[1] >> 16)]
         ^  OT3[0][(ade_08t)(b1[1] >> 8)]
         ^  OT4[0][(ade_08t) b1[1]]);
    b0[2] = (OT1[0][(ade_08t)(b1[2] >> 24)]
         ^  OT2[0][(ade_08t)(b1[2] >> 16)]
         ^  OT3[0][(ade_08t)(b1[2] >> 8)]
         ^  OT4[0][(ade_08t) b1[2]]);
    b0[3] = (OT1[0][(ade_08t)(b1[3] >> 24)]
         ^  OT2[0][(ade_08t)(b1[3] >> 16)]
         ^  OT3[0][(ade_08t)(b1[3] >> 8)]
         ^  OT4[0][(ade_08t) b1[3]]);

    b1[0] = (OL1[0][(ade_08t)(b0[(4-delta[0])%4] >> 24)]
         ^  OL2[0][(ade_08t)(b0[(4-delta[1])%4] >> 16)]
         ^  OL3[0][(ade_08t)(b0[(4-delta[2])%4] >>  8)]
         ^  OL4[0][(ade_08t)(b0[(4-delta[3])%4])]) ^ k_sch[0];
    b1[1] = (OL1[0][(ade_08t)(b0[(5-delta[0])%4] >> 24)]
         ^  OL2[0][(ade_08t)(b0[(5-delta[1])%4] >> 16)]
         ^  OL3[0][(ade_08t)(b0[(5-delta[2])%4] >>  8)]
         ^  OL4[0][(ade_08t)(b0[(5-delta[3])%4])]) ^ k_sch[1];
    b1[2] = (OL1[0][(ade_08t)(b0[(6-delta[0])%4] >> 24)]
         ^  OL2[0][(ade_08t)(b0[(6-delta[1])%4] >> 16)]
         ^  OL3[0][(ade_08t)(b0[(6-delta[2])%4] >>  8)]
         ^  OL4[0][(ade_08t)(b0[(6-delta[3])%4])]) ^ k_sch[2];
    b1[3] = (OL1[0][(ade_08t)(b0[(7-delta[0])%4] >> 24)]
         ^  OL2[0][(ade_08t)(b0[(7-delta[1])%4] >> 16)]
         ^  OL3[0][(ade_08t)(b0[(7-delta[2])%4] >>  8)]
         ^  OL4[0][(ade_08t)(b0[(7-delta[3])%4])]) ^ k_sch[3];

    out_blk[0]  = (ade_08t)(b1[0] >> 24);    out_blk[1]  = (ade_08t)(b1[0] >> 16);
    out_blk[2]  = (ade_08t)(b1[0] >>  8);    out_blk[3]  = (ade_08t) b1[0];
    out_blk[4]  = (ade_08t)(b1[1] >> 24);    out_blk[5]  = (ade_08t)(b1[1] >> 16);
    out_blk[6]  = (ade_08t)(b1[1] >>  8);    out_blk[7]  = (ade_08t) b1[1];
    out_blk[8]  = (ade_08t)(b1[2] >> 24);    out_blk[9]  = (ade_08t)(b1[2] >> 16);
    out_blk[10] = (ade_08t)(b1[2] >>  8);    out_blk[11] = (ade_08t) b1[2];
    out_blk[12] = (ade_08t)(b1[3] >> 24);    out_blk[13] = (ade_08t)(b1[3] >> 16);
    out_blk[14] = (ade_08t)(b1[3] >>  8);    out_blk[15] = (ade_08t) b1[3];
```

```
 return 1;
}

void __fastcall TForm1::CheckBox1Click(TObject *Sender)
{
    unsigned long   i, len, rlen;
    char    buf[2*BLOCK_LEN];
    struct stat statbuf;
                                            WideString buf2;
    buf2 = "";

    if(CheckBox1->Checked==true)
    {
        Label19->Caption = "Загрузка файла...";
        Application->ProcessMessages();
        if( (Button1->Enabled==true) && (Button4->Enabled==true) )
        {
            FILE *handle = fopen(MyFName1.c_str(), "rb");
            stat(MyFName1.c_str(),&statbuf);
            fclose(handle);
            if( (statbuf.st_size>150000) || (statbuf.st_size<=0) )
            {
                Memo1->Clear();
                Memo1->Lines->Add("ОТКРЫТЫЙ ТЕКСТ");
            }
            else
                Memo1->Lines->LoadFromFile(MyFName1);
        }
        if( (Button2->Enabled==true) && (Edit2->Text!=' ') )
        {
            Memo2->Clear();
            fin2 = fopen(MyFName3.c_str(), "rb");
            if(fin2)
            {
                fseek(fin2, 0, SEEK_END);
                fgetpos(fin2, &flen);
                rlen = file_len(flen);
                fseek(fin2, 0, SEEK_SET);

                if(rlen>150000)
                    Memo2->Lines->Append("ФАЙЛ СЛИШКОМ ВЕЛИК...");
                else
                {
                    while(rlen > 0 && !feof(fin2))
                    {
                        len = (unsigned long)fread(buf, 1, 2*BLOCK_LEN, fin2);
                        rlen -= len;
                        if( (rlen>1000) && (rlen<1200) ) break;

                        unsigned long i1;
                        for (i1=0;i1<len;i1++)
                        {
                            if(buf[i1]=='\x0')
                                buf[i1]=0x20;
                        }
                        buf2 += buf;
                    }
                }
            }
            fclose(fin2);
            Memo2->Lines->Append(buf2);
        }
        if( (Button3->Enabled==true) && (Edit3->Text!="") )
        {

 FILE *handle = fopen(MyFName3.c_str(), "rb");
            stat(MyFName3.c_str(),&statbuf);
            fclose(handle);
            if( (statbuf.st_size>150000) || (statbuf.st_size<=0) )
            {
                Memo3->Clear();
                Memo3->Lines->Add("ОТКРЫТЫЙ ТЕКСТ");
            }
            else
                Memo3->Lines->LoadFromFile(MyFName4);
        }
```

```
 Label19->Caption = "";
        Application->ProcessMessages();
    }
    else
    {
        Label19->Caption = "";
        Application->ProcessMessages();
        if(Memo1->Lines->Count>0)
        {
            Memo1->Clear();
            Memo1->Lines->Add("ОТКРЫТЫЙ ТЕКСТ");
        }
        if( (Memo2->Lines->Count>0) &&
            ( (Edit2->Text==' ') || (BitBtn1->Enabled==true) ) )
        {
            Memo2->Clear();
            Memo2->Lines->Add("ЗАКРЫТЫЙ ТЕКСТ");
        }
        if( (Memo3->Lines->Count>0) && (Edit3->Text!=' ') )
        {
            Memo3->Clear();
            Memo3->Lines->Add("ОТКРЫТЫЙ ТЕКСТ");
        }
    }
}

int form_M1(ade_32t k_sch[])
{
    int i1,i;
    ade_32t  w;

    for(i1=0;i1<10;i1++)
    {
        CKey[i1]        = (ade_08t)(k_sch[(i1+1)*4]);
        if(!CKey[i1]) CKey[i1]++;
        CicleKey[i1]    = (ade_08t)(k_sch[(i1+1)*4+1]);
        if( (CicleKey[i1]==0) || (CicleKey[i1]==1) ) CicleKey[i1] = 2;

        delta[4*i1]     = (ade_08t)((k_sch[(i1+1)*4])>>8)   & 0x03;
        delta[4*i1+1]   = (ade_08t)((k_sch[(i1+1)*4+1])>>8) & 0x03;
        delta[4*i1+2]   = (ade_08t)((k_sch[(i1+1)*4+2])>>8) & 0x03;
        delta[4*i1+3]   = (ade_08t)((k_sch[(i1+1)*4+3])>>8) & 0x03;

//================================================
//    Формирование матрицы M
//================================================
    C00 = CicleKey[i1];                          // x
    C22 = ns_tab[5][CicleKey[i1]];               // x9
    C33 = ns_tab[7][CicleKey[i1]];               // x16
C10 = C01 = ns_tab[0][CicleKey[i1]];         // x2
    C20 = C02 = ns_tab[1][CicleKey[i1]];         // x3
    C11 = C30 = C03 = ns_tab[2][CicleKey[i1]];   // x4
    C21 = C12 = ns_tab[3][CicleKey[i1]];         // x6
    C31 = C13 = ns_tab[4][CicleKey[i1]];         // x8
    C32 = C23 = ns_tab[6][CicleKey[i1]];         // x12

//================================================
//    Вычисление обратной матрицы M
//================================================
    A00 =   nt_tab[C11][nt_tab[C22][C33]^nt_tab[C32][C23]]^
            nt_tab[C12][nt_tab[C21][C33]^nt_tab[C31][C23]]^
            nt_tab[C13][nt_tab[C21][C32]^nt_tab[C31][C22]];

    A10 =   nt_tab[C01][nt_tab[C22][C33]^nt_tab[C32][C23]]^
            nt_tab[C02][nt_tab[C21][C33]^nt_tab[C31][C23]]^
            nt_tab[C03][nt_tab[C21][C32]^nt_tab[C31][C22]];

    A11 =   nt_tab[C00][nt_tab[C22][C33]^nt_tab[C32][C23]]^
            nt_tab[C02][nt_tab[C20][C33]^nt_tab[C30][C23]]^
            nt_tab[C03][nt_tab[C20][C32]^nt_tab[C30][C22]];

    A20 =   nt_tab[C01][nt_tab[C12][C33]^nt_tab[C32][C13]]^
            nt_tab[C02][nt_tab[C11][C33]^nt_tab[C31][C13]]^
            nt_tab[C03][nt_tab[C11][C32]^nt_tab[C31][C12]];

    A21 =   nt_tab[C00][nt_tab[C12][C33]^nt_tab[C32][C13]]^
```

```
nt_tab[C02][nt_tab[C10][C33]^nt_tab[C30][C13]]^
            nt_tab[C03][nt_tab[C10][C32]^nt_tab[C30][C12]];

    A22 =   nt_tab[C00][nt_tab[C11][C33]^nt_tab[C31][C13]]^
            nt_tab[C01][nt_tab[C10][C33]^nt_tab[C30][C13]]^
            nt_tab[C03][nt_tab[C10][C31]^nt_tab[C30][C11]];

    A30 =   nt_tab[C01][nt_tab[C12][C23]^nt_tab[C22][C13]]^
            nt_tab[C02][nt_tab[C11][C23]^nt_tab[C21][C13]]^
            nt_tab[C03][nt_tab[C11][C22]^nt_tab[C21][C12]];

A31 =   nt_tab[C00][nt_tab[C12][C23]^nt_tab[C22][C13]]^
            nt_tab[C02][nt_tab[C10][C23]^nt_tab[C20][C13]]^
            nt_tab[C03][nt_tab[C10][C22]^nt_tab[C20][C12]];

    A32 =   nt_tab[C00][nt_tab[C11][C23]^nt_tab[C21][C13]]^
            nt_tab[C01][nt_tab[C10][C23]^nt_tab[C20][C13]]^
            nt_tab[C03][nt_tab[C10][C21]^nt_tab[C20][C11]];

    A33 =   nt_tab[C00][nt_tab[C11][C22]^nt_tab[C21][C12]]^
            nt_tab[C01][nt_tab[C10][C22]^nt_tab[C20][C12]]^
            nt_tab[C02][nt_tab[C10][C21]^nt_tab[C20][C11]];

//================================================
//  Вычисление определителя
//================================================
    ade_08t det;
    det = opr_tab[nt_tab[C00][A00]^nt_tab[C01][A10]^
                  nt_tab[C02][A20]^nt_tab[C03][A30]];

    B00 = nt_tab[det][A00];
    B11 = B30 = B03 = nt_tab[det][A11];
    B22 = nt_tab[det][A22];



 B33 = nt_tab[det][A33];
    B10 = B01 = nt_tab[det][A10];
    B20 = B02 = nt_tab[det][A20];
    B30 = B03 = nt_tab[det][A30];
    B21 = B12 = nt_tab[det][A21];
    B31 = B13 = nt_tab[det][A31];
    B32 = B23 = nt_tab[det][A32];

        ade_08t b;
        ade_32t w;
        switch(i1)
        {
            case 0: LOAD_MATR1(M_0);
                    for(i=0;i<256;i++)
                    {
                        b = fwd_affine(opr_tab[nt_tab[i][CKey[i1]]]);
                        T1[i1][i]  = bytes2word(nt_tab[M_0[0][0]][b],
                                                nt_tab[M_0[1][0]][b],
                                                nt_tab[M_0[2][0]][b],
                                                nt_tab[M_0[3][0]][b]);
                        T2[i1][i]  = bytes2word(nt_tab[M_0[0][1]][b],
                                                nt_tab[M_0[1][1]][b],
                                                nt_tab[M_0[2][1]][b],
                                                nt_tab[M_0[3][1]][b]);
                        T3[i1][i]  = bytes2word(nt_tab[M_0[0][2]][b],
                                                nt_tab[M_0[1][2]][b],
                                                nt_tab[M_0[2][2]][b],
                                                nt_tab[M_0[3][2]][b]);
                        T4[i1][i]  = bytes2word(nt_tab[M_0[0][3]][b],
                                                nt_tab[M_0[1][3]][b],
                                                nt_tab[M_0[2][3]][b],
                                                nt_tab[M_0[3][3]][b]);
                        w  = bytes2word(b, 0, 0, 0);
                        L1[i1][i]  = w;
                        L2[i1][i]  = upr(w, 1);
                        L3[i1][i]  = upr(w, 2);
                        L4[i1][i]  = upr(w, 3);
                    }
                    break;
            case 1: LOAD_MATR1(M_1);
```

```
 for(i=0;i<256;i++)
                    {
                        b = fwd_affine(opr_tab[nt_tab[i][CKey[i1]]]);
                        T1[i1][i]  = bytes2word(nt_tab[M_1[0][0]][b],
                                                nt_tab[M_1[1][0]][b],
                                                nt_tab[M_1[2][0]][b],
                                                nt_tab[M_1[3][0]][b]);
                        T2[i1][i]  = bytes2word(nt_tab[M_1[0][1]][b],
                                                nt_tab[M_1[1][1]][b],
                                                nt_tab[M_1[2][1]][b],
                                              nt_tab[M_1[3][1]][b]);
                        T3[i1][i]  = bytes2word(nt_tab[M_1[0][2]][b],
                                                nt_tab[M_1[1][2]][b],
                                                nt_tab[M_1[2][2]][b],
                                                nt_tab[M_1[3][2]][b]);
                        T4[i1][i]  = bytes2word(nt_tab[M_1[0][3]][b],
                                                nt_tab[M_1[1][3]][b],
                                                nt_tab[M_1[2][3]][b],
                                                nt_tab[M_1[3][3]][b]);
                        w  = bytes2word(b, 0, 0, 0);

L1[i1][i]  = w;
                        L2[i1][i]  = upr(w, 1);
                        L3[i1][i]  = upr(w, 2);
                        L4[i1][i]  = upr(w, 3);
                    }
                    break;
            case 2: LOAD_MATR1(M_2);
                    for(i=0;i<256;i++)
                    {
                        b = fwd_affine(opr_tab[nt_tab[i][CKey[i1]]]);
                        T1[i1][i]  = bytes2word(nt_tab[M_2[0][0]][b],
                                                nt_tab[M_2[1][0]][b],
                                                nt_tab[M_2[2][0]][b],
                                                nt_tab[M_2[3][0]][b]);
                        T2[i1][i]  = bytes2word(nt_tab[M_2[0][1]][b],
                                                nt_tab[M_2[1][1]][b],
                                                nt_tab[M_2[2][1]][b],
                                                nt_tab[M_2[3][1]][b]);
                        T3[i1][i]  = bytes2word(nt_tab[M_2[0][2]][b],
                                                nt_tab[M_2[1][2]][b],
                                                nt_tab[M_2[2][2]][b],
                                                nt_tab[M_2[3][2]][b]);
                        T4[i1][i]  = bytes2word(nt_tab[M_2[0][3]][b],
                                                nt_tab[M_2[1][3]][b],
                                                nt_tab[M_2[2][3]][b],
                                                nt_tab[M_2[3][3]][b]);
                        w  = bytes2word(b, 0, 0, 0);
                        L1[i1][i]  = w;
                        L2[i1][i]  = upr(w, 1);
                        L3[i1][i]  = upr(w, 2);
                        L4[i1][i]  = upr(w, 3);
                    }
                    break;
            case 3: LOAD_MATR1(M_3);
                    for(i=0;i<256;i++)
                    {
                        b = fwd_affine(opr_tab[nt_tab[i][CKey[i1]]]);
                        T1[i1][i]  = bytes2word(nt_tab[M_3[0][0]][b],
                                                nt_tab[M_3[1][0]][b],
                                                nt_tab[M_3[2][0]][b],
                                                nt_tab[M_3[3][0]][b]);
                        T2[i1][i]  = bytes2word(nt_tab[M_3[0][1]][b],
                                                nt_tab[M_3[1][1]][b],
                                                nt_tab[M_3[2][1]][b],
                                                nt_tab[M_3[3][1]][b]);
                        T3[i1][i]  = bytes2word(nt_tab[M_3[0][2]][b],
                                                nt_tab[M_3[1][2]][b],
                                                nt_tab[M_3[2][2]][b],
                                                nt_tab[M_3[3][2]][b]);
                        T4[i1][i]  = bytes2word(nt_tab[M_3[0][3]][b],
                                                nt_tab[M_3[1][3]][b],
                                                nt_tab[M_3[2][3]][b],
                                                nt_tab[M_3[3][3]][b]);
                        w  = bytes2word(b, 0, 0, 0);
                        L1[i1][i]  = w;
```

```
L2[i1][i]  = upr(w, 1);
                    L3[i1][i]  = upr(w, 2);
                    L4[i1][i]  = upr(w, 3);
                }
                break;

 case 4: LOAD_MATR1(M_4);
                for(i=0;i<256;i++)
                {
                    b = fwd_affine(opr_tab[nt_tab[i][CKey[i1]]]);
                    T1[i1][i]  = bytes2word(nt_tab[M_4[0][0]][b],
                                            nt_tab[M_4[1][0]][b],
                                                nt_tab[M_4[2][0]][b],
                                                nt_tab[M_4[3][0]][b]);
                    T2[i1][i]  = bytes2word(nt_tab[M_4[0][1]][b],
                                                nt_tab[M_4[1][1]][b],
                                                nt_tab[M_4[2][1]][b],
                                                nt_tab[M_4[3][1]][b]);
                    T3[i1][i]  = bytes2word(nt_tab[M_4[0][2]][b],
                                                nt_tab[M_4[1][2]][b],
                                                nt_tab[M_4[2][2]][b],
                                                nt_tab[M_4[3][2]][b]);
                    T4[i1][i]  = bytes2word(nt_tab[M_4[0][3]][b],
                                                nt_tab[M_4[1][3]][b],
                                                nt_tab[M_4[2][3]][b],
                                                nt_tab[M_4[3][3]][b]);
                    w  = bytes2word(b, 0, 0, 0);
                    L1[i1][i]  = w;
                    L2[i1][i]  = upr(w, 1);
                    L3[i1][i]  = upr(w, 2);
                    L4[i1][i]  = upr(w, 3);
                }
                break;
            case 5: LOAD_MATR1(M_5);
                    for(i=0;i<256;i++)
                    {
                        b = fwd_affine(opr_tab[nt_tab[i][CKey[i1]]]);
                        T1[i1][i]  = bytes2word(nt_tab[M_5[0][0]][b],
                                                nt_tab[M_5[1][0]][b],
                                                nt_tab[M_5[2][0]][b],
                                                nt_tab[M_5[3][0]][b]);
                        T2[i1][i]  = bytes2word(nt_tab[M_5[0][1]][b],
                                                nt_tab[M_5[1][1]][b],
                                                nt_tab[M_5[2][1]][b],
                                                nt_tab[M_5[3][1]][b]);
                        T3[i1][i]  = bytes2word(nt_tab[M_5[0][2]][b],
                                                nt_tab[M_5[1][2]][b],
                                                nt_tab[M_5[2][2]][b],
                                                nt_tab[M_5[3][2]][b]);
                        T4[i1][i]  = bytes2word(nt_tab[M_5[0][3]][b],
                                                nt_tab[M_5[1][3]][b],
                                                nt_tab[M_5[2][3]][b],
                                                nt_tab[M_5[3][3]][b]);
                        w  = bytes2word(b, 0, 0, 0);
                        L1[i1][i]  = w;
                        L2[i1][i]  = upr(w, 1);
                        L3[i1][i]  = upr(w, 2);
                        L4[i1][i]  = upr(w, 3);
                    }
                    break;
            case 6: LOAD_MATR1(M_6);
                    for(i=0;i<256;i++)
                    {
                        b = fwd_affine(opr_tab[nt_tab[i][CKey[i1]]]);
                        T1[i1][i]  = bytes2word(nt_tab[M_6[0][0]][b],
                                                nt_tab[M_6[1][0]][b],

                         nt_tab[M_6[2][0]][b],
                                                nt_tab[M_6[3][0]][b]);
                        T2[i1][i]  = bytes2word(nt_tab[M_6[0][1]][b],
                                                nt_tab[M_6[1][1]][b],
                                                nt_tab[M_6[2][1]][b],
                                                nt_tab[M_6[3][1]][b]);
                        T3[i1][i]  = bytes2word(nt_tab[M_6[0][2]][b],
                                                nt_tab[M_6[1][2]][b],
                                                nt_tab[M_6[2][2]][b],
```

```
      nt_tab[M_6[3][2]][b]);
                    T4[i1][i]  = bytes2word(nt_tab[M_6[0][3]][b],
                                            nt_tab[M_6[1][3]][b],
                                            nt_tab[M_6[2][3]][b],
                                            nt_tab[M_6[3][3]][b]);
                    w  = bytes2word(b, 0, 0, 0);
                    L1[i1][i]  = w;
                    L2[i1][i]  = upr(w, 1);
                    L3[i1][i]  = upr(w, 2);
                    L4[i1][i]  = upr(w, 3);
                }
      break;
          case 7: LOAD_MATR1(M_7);
                  for(i=0;i<256;i++)
                  {
                      b = fwd_affine(opr_tab[nt_tab[i][CKey[i1]]]);
                      T1[i1][i]  = bytes2word(nt_tab[M_7[0][0]][b],
                                              nt_tab[M_7[1][0]][b],
                                              nt_tab[M_7[2][0]][b],
                                              nt_tab[M_7[3][0]][b]);
                      T2[i1][i]  = bytes2word(nt_tab[M_7[0][1]][b],
                                              nt_tab[M_7[1][1]][b],
                                              nt_tab[M_7[2][1]][b],
                                              nt_tab[M_7[3][1]][b]);
                      T3[i1][i]  = bytes2word(nt_tab[M_7[0][2]][b],
                                              nt_tab[M_7[1][2]][b],
                                              nt_tab[M_7[2][2]][b],
                                              nt_tab[M_7[3][2]][b]);
                      T4[i1][i]  = bytes2word(nt_tab[M_7[0][3]][b],
                                              nt_tab[M_7[1][3]][b],
                                              nt_tab[M_7[2][3]][b],
                                              nt_tab[M_7[3][3]][b]);
                      w  = bytes2word(b, 0, 0, 0);
                      L1[i1][i]  = w;
                      L2[i1][i]  = upr(w, 1);
                      L3[i1][i]  = upr(w, 2);
                      L4[i1][i]  = upr(w, 3);
                  }
                  break;
          case 8: LOAD_MATR1(M_8);
                  for(i=0;i<256;i++)
                  {
                      b = fwd_affine(opr_tab[nt_tab[i][CKey[i1]]]);
                      T1[i1][i]  = bytes2word(nt_tab[M_8[0][0]][b],
                                              nt_tab[M_8[1][0]][b],
                                              nt_tab[M_8[2][0]][b],
                                              nt_tab[M_8[3][0]][b]);
                      T2[i1][i]  = bytes2word(nt_tab[M_8[0][1]][b],
                                              nt_tab[M_8[1][1]][b],
                                              nt_tab[M_8[2][1]][b],
                                              nt_tab[M_8[3][1]][b]);

T3[i1][i]  = bytes2word(nt_tab[M_8[0][2]][b],
                                              nt_tab[M_8[1][2]][b],
                                              nt_tab[M_8[2][2]][b],
                                              nt_tab[M_8[3][2]][b]);
                      T4[i1][i]  = bytes2word(nt_tab[M_8[0][3]][b],
                                              nt_tab[M_8[1][3]][b],
                                              nt_tab[M_8[2][3]][b],
                                              nt_tab[M_8[3][3]][b]);
                      w  = bytes2word(b, 0, 0, 0);
                      L1[i1][i]  = w;
                      L2[i1][i]  = upr(w, 1);
                      L3[i1][i]  = upr(w, 2);
                      L4[i1][i]  = upr(w, 3);
                  }
                  break;
          case 9:
                  for(i=0;i<256;i++)
                  {
                      b = fwd_affine(opr_tab[nt_tab[i][CKey[i1]]]);
                      w  = bytes2word(b, 0, 0, 0);
                      L1[i1][i]  = w;
                      L2[i1][i]  = upr(w, 1);
                      L3[i1][i]  = upr(w, 2);
                      L4[i1][i]  = upr(w, 3);
```

```
}
                    break;
            default: break;
        }
    }
    return 1;
}

void LOAD_MATR1(ade_08t matrM[][4])
{
    int i,j; for(i=0;i<4;i++) for(j=0;j<4;j++) matrM[i][j]  = c[i][j];
}

void LOAD_MATR2(ade_08t matrOM[][4])
{
    int i,j; for(i=0;i<4;i++) for(j=0;j<4;j++) matrOM[i][j] = b[i][j];
}

int form_M2(ade_32t k_sch[])
{
    int i1,i;
    ade_32t  w;

    for(i1=0;i1<10;i1++)
    {
        CKey[i1]        = (ade_08t)(k_sch[(i1+1)*4]);
        if(!CKey[i1]) CKey[i1]++;
        CicleKey[i1]    = (ade_08t)(k_sch[(i1+1)*4+1]);
        if( (CicleKey[i1]==0) || (CicleKey[i1]==1) ) CicleKey[i1] = 2;

        delta[4*i1]     = (ade_08t)((k_sch[(i1+1)*4])>>8)   & 0x03;
        delta[4*i1+1]   = (ade_08t)((k_sch[(i1+1)*4+1])>>8) & 0x03;
        delta[4*i1+2]   = (ade_08t)((k_sch[(i1+1)*4+2])>>8) & 0x03;
        delta[4*i1+3]   = (ade_08t)((k_sch[(i1+1)*4+3])>>8) & 0x03;

//================================================

//    Формирование матрицы M
//================================================
    C00 = CicleKey[i1];                          // x
    C22 = ns_tab[5][CicleKey[i1]];               // x9
    C33 = ns_tab[7][CicleKey[i1]];               // x16
    C10 = C01 = ns_tab[0][CicleKey[i1]];         // x2
    C20 = C02 = ns_tab[1][CicleKey[i1]];         // x3
    C11 = C30 = C03 = ns_tab[2][CicleKey[i1]];   // x4
    C21 = C12 = ns_tab[3][CicleKey[i1]];         // x6
    C31 = C13 = ns_tab[4][CicleKey[i1]];         // x8
    C32 = C23 = ns_tab[6][CicleKey[i1]];         // x12

//================================================
//    Вычисление обратной матрицы M
//================================================
    A00 =   nt_tab[C11][nt_tab[C22][C33]^nt_tab[C32][C23]]^
            nt_tab[C12][nt_tab[C21][C33]^nt_tab[C31][C23]]^
            nt_tab[C13][nt_tab[C21][C32]^nt_tab[C31][C22]];

    A10 =   nt_tab[C01][nt_tab[C22][C33]^nt_tab[C32][C23]]^
            nt_tab[C02][nt_tab[C21][C33]^nt_tab[C31][C23]]^
            nt_tab[C03][nt_tab[C21][C32]^nt_tab[C31][C22]];

    A11 =   nt_tab[C00][nt_tab[C22][C33]^nt_tab[C32][C23]]^
            nt_tab[C02][nt_tab[C20][C33]^nt_tab[C30][C23]]^
            nt_tab[C03][nt_tab[C20][C32]^nt_tab[C30][C22]];

    A20 =   nt_tab[C01][nt_tab[C12][C33]^nt_tab[C32][C13]]^
            nt_tab[C02][nt_tab[C11][C33]^nt_tab[C31][C13]]^
            nt_tab[C03][nt_tab[C11][C32]^nt_tab[C31][C12]];

    A21 =   nt_tab[C00][nt_tab[C12][C33]^nt_tab[C32][C13]]^
            nt_tab[C02][nt_tab[C10][C33]^nt_tab[C30][C13]]^
            nt_tab[C03][nt_tab[C10][C32]^nt_tab[C30][C12]];

    A22 =   nt_tab[C00][nt_tab[C11][C33]^nt_tab[C31][C13]]^
            nt_tab[C01][nt_tab[C10][C33]^nt_tab[C30][C13]]^
            nt_tab[C03][nt_tab[C10][C31]^nt_tab[C30][C11]];
```

```
  A30 =   nt_tab[C01][nt_tab[C12][C23]^nt_tab[C22][C13]]^
            nt_tab[C02][nt_tab[C11][C23]^nt_tab[C21][C13]]^
            nt_tab[C03][nt_tab[C11][C22]^nt_tab[C21][C12]];

    A31 =   nt_tab[C00][nt_tab[C12][C23]^nt_tab[C22][C13]]^
            nt_tab[C02][nt_tab[C10][C23]^nt_tab[C20][C13]]^
            nt_tab[C03][nt_tab[C10][C22]^nt_tab[C20][C12]];

    A32 =   nt_tab[C00][nt_tab[C11][C23]^nt_tab[C21][C13]]^
            nt_tab[C01][nt_tab[C10][C23]^nt_tab[C20][C13]]^
            nt_tab[C03][nt_tab[C10][C21]^nt_tab[C20][C11]];

A33 =   nt_tab[C00][nt_tab[C11][C22]^nt_tab[C21][C12]]^
            nt_tab[C01][nt_tab[C10][C22]^nt_tab[C20][C12]]^
            nt_tab[C02][nt_tab[C10][C21]^nt_tab[C20][C11]];

//================================================
//  Вычисление определителя
//================================================
    ade_08t det;
    det = opr_tab[nt_tab[C00][A00]^nt_tab[C01][A10]^

 nt_tab[C02][A20]^nt_tab[C03][A30]];

    B00 = nt_tab[det][A00];
    B11 = B30 = B03 = nt_tab[det][A11];
    B22 = nt_tab[det][A22];
    B33 = nt_tab[det][A33];
    B10 = B01 = nt_tab[det][A10];
    B20 = B02 = nt_tab[det][A20];
    B30 = B03 = nt_tab[det][A30];
    B21 = B12 = nt_tab[det][A21];
    B31 = B13 = nt_tab[det][A31];
    B32 = B23 = nt_tab[det][A32];

        ade_08t b;
        ade_32t w;
        switch(i1)
        {
            case 0: LOAD_MATR2(OM_0);
                    for(i=0;i<256;i++)
                    {
                        b = opr_tab[nt_tab[inv_affine((ade_08t)i)][CKey[i1]]];
                        OT1[i1][i] = bytes2word(nt_tab[OM_0[0][0]][i],
                                                nt_tab[OM_0[1][0]][i],
                                                nt_tab[OM_0[2][0]][i],
                                                nt_tab[OM_0[3][0]][i]);
                        OT2[i1][i] = bytes2word(nt_tab[OM_0[0][1]][i],
                                                nt_tab[OM_0[1][1]][i],
                                                nt_tab[OM_0[2][1]][i],
                                                nt_tab[OM_0[3][1]][i]);
                        OT3[i1][i] = bytes2word(nt_tab[OM_0[0][2]][i],
                                                nt_tab[OM_0[1][2]][i],
                                                nt_tab[OM_0[2][2]][i],
                                                nt_tab[OM_0[3][2]][i]);
                        OT4[i1][i] = bytes2word(nt_tab[OM_0[0][3]][i],
                                                nt_tab[OM_0[1][3]][i],
                                                nt_tab[OM_0[2][3]][i],
                                                nt_tab[OM_0[3][3]][i]);
                        w = bytes2word(b, 0, 0, 0);
                        OL1[i1][i] = w;
                        OL2[i1][i] = upr(w, 1);
                        OL3[i1][i] = upr(w, 2);
                        OL4[i1][i] = upr(w, 3);
                    }
                    break;
            case 1: LOAD_MATR2(OM_1);
                    for(i=0;i<256;i++)
                    {
                        b = opr_tab[nt_tab[inv_affine((ade_08t)i)][CKey[i1]]];
                        OT1[i1][i] = bytes2word(nt_tab[OM_1[0][0]][i],
                                                nt_tab[OM_1[1][0]][i],
                                                nt_tab[OM_1[2][0]][i],
                                                nt_tab[OM_1[3][0]][i]);
                        OT2[i1][i] = bytes2word(nt_tab[OM_1[0][1]][i],
                                                nt_tab[OM_1[1][1]][i],
```

```
 nt_tab[OM_1[2][1]][i],
                                          nt_tab[OM_1[3][1]][i]);
                        OT3[i1][i] = bytes2word(nt_tab[OM_1[0][2]][i],
                                          nt_tab[OM_1[1][2]][i],
                                          nt_tab[OM_1[2][2]][i],
                                          nt_tab[OM_1[3][2]][i]);
OT4[i1][i] = bytes2word(nt_tab[OM_1[0][3]][i],
                                          nt_tab[OM_1[1][3]][i],
                                          nt_tab[OM_1[2][3]][i],
                                          nt_tab[OM_1[3][3]][i]);
                        w = bytes2word(b, 0, 0, 0);
OL1[i1][i] = w;
                        OL2[i1][i] = upr(w, 1);
                        OL3[i1][i] = upr(w, 2);
                        OL4[i1][i] = upr(w, 3);
                    }
                    break;
                case 2: LOAD_MATR2(OM_2);
                    for(i=0;i<256;i++)
                    {
                        b = opr_tab[nt_tab[inv_affine((ade_08t)i)][CKey[i1]]];
                        OT1[i1][i] = bytes2word(nt_tab[OM_2[0][0]][i],
                                          nt_tab[OM_2[1][0]][i],
                                          nt_tab[OM_2[2][0]][i],
                                          nt_tab[OM_2[3][0]][i]);
                        OT2[i1][i] = bytes2word(nt_tab[OM_2[0][1]][i],
                                          nt_tab[OM_2[1][1]][i],
                                          nt_tab[OM_2[2][1]][i],
                                          nt_tab[OM_2[3][1]][i]);
                        OT3[i1][i] = bytes2word(nt_tab[OM_2[0][2]][i],
                                          nt_tab[OM_2[1][2]][i],
                                          nt_tab[OM_2[2][2]][i],
                                          nt_tab[OM_2[3][2]][i]);
                        OT4[i1][i] = bytes2word(nt_tab[OM_2[0][3]][i],
                                          nt_tab[OM_2[1][3]][i],
                                          nt_tab[OM_2[2][3]][i],
                                          nt_tab[OM_2[3][3]][i]);
                        w = bytes2word(b, 0, 0, 0);
                        OL1[i1][i] = w;
                        OL2[i1][i] = upr(w, 1);
                        OL3[i1][i] = upr(w, 2);
                        OL4[i1][i] = upr(w, 3);
                    }
                    break;
                case 3: LOAD_MATR2(OM_3);
                    for(i=0;i<256;i++)
                    {
                        b = opr_tab[nt_tab[inv_affine((ade_08t)i)][CKey[i1]]];
                        OT1[i1][i] = bytes2word(nt_tab[OM_3[0][0]][i],
                                          nt_tab[OM_3[1][0]][i],
                                          nt_tab[OM_3[2][0]][i],
                                          nt_tab[OM_3[3][0]][i]);
                        OT2[i1][i] = bytes2word(nt_tab[OM_3[0][1]][i],
                                          nt_tab[OM_3[1][1]][i],
                                          nt_tab[OM_3[2][1]][i],
                                          nt_tab[OM_3[3][1]][i]);
                        OT3[i1][i] = bytes2word(nt_tab[OM_3[0][2]][i],
                                          nt_tab[OM_3[1][2]][i],
                                          nt_tab[OM_3[2][2]][i],
                                          nt_tab[OM_3[3][2]][i]);
                        OT4[i1][i] = bytes2word(nt_tab[OM_3[0][3]][i],
                                          nt_tab[OM_3[1][3]][i],
                                          nt_tab[OM_3[2][3]][i],
                                          nt_tab[OM_3[3][3]][i]);
                        w = bytes2word(b, 0, 0, 0);
                        OL1[i1][i] = w;

 OL2[i1][i] = upr(w, 1);
                        OL3[i1][i] = upr(w, 2);
                        OL4[i1][i] = upr(w, 3);
                    }
                    break;
                case 4: LOAD_MATR2(OM_4);
                    for(i=0;i<256;i++)
                    {
                        b = opr_tab[nt_tab[inv_affine((ade_08t)i)][CKey[i1]]];
```

```
 OT1[i1][i] = bytes2word(nt_tab[OM_4[0][0]][i],
                                            nt_tab[OM_4[1][0]][i],
                                            nt_tab[OM_4[2][0]][i],
                                            nt_tab[OM_4[3][0]][i]);
                        OT2[i1][i] = bytes2word(nt_tab[OM_4[0][1]][i],
                                            nt_tab[OM_4[1][1]][i],
                                            nt_tab[OM_4[2][1]][i],
                                            nt_tab[OM_4[3][1]][i]);
                        OT3[i1][i] = bytes2word(nt_tab[OM_4[0][2]][i],
                                            nt_tab[OM_4[1][2]][i],
                                            nt_tab[OM_4[2][2]][i],

 nt_tab[OM_4[3][2]][i]);
                        OT4[i1][i] = bytes2word(nt_tab[OM_4[0][3]][i],
                                            nt_tab[OM_4[1][3]][i],
                                            nt_tab[OM_4[2][3]][i],
                                            nt_tab[OM_4[3][3]][i]);
                        w = bytes2word(b, 0, 0, 0);
                        OL1[i1][i] = w;
                        OL2[i1][i] = upr(w, 1);
                        OL3[i1][i] = upr(w, 2);
                        OL4[i1][i] = upr(w, 3);
                    }
                    break;
            case 5: LOAD_MATR2(OM_5);
                    for(i=0;i<256;i++)
                    {
                        b = opr_tab[nt_tab[inv_affine((ade_08t)i)][CKey[i1]]];
                        OT1[i1][i] = bytes2word(nt_tab[OM_5[0][0]][i],
                                            nt_tab[OM_5[1][0]][i],
                                            nt_tab[OM_5[2][0]][i],
                                            nt_tab[OM_5[3][0]][i]);
                        OT2[i1][i] = bytes2word(nt_tab[OM_5[0][1]][i],
                                            nt_tab[OM_5[1][1]][i],
                                            nt_tab[OM_5[2][1]][i],
                                            nt_tab[OM_5[3][1]][i]);
                        OT3[i1][i] = bytes2word(nt_tab[OM_5[0][2]][i],
                                            nt_tab[OM_5[1][2]][i],
                                            nt_tab[OM_5[2][2]][i],
                                            nt_tab[OM_5[3][2]][i]);
                        OT4[i1][i] = bytes2word(nt_tab[OM_5[0][3]][i],
                                            nt_tab[OM_5[1][3]][i],
                                            nt_tab[OM_5[2][3]][i],
                                            nt_tab[OM_5[3][3]][i]);
                        w = bytes2word(b, 0, 0, 0);
                        OL1[i1][i] = w;
                        OL2[i1][i] = upr(w, 1);
                        OL3[i1][i] = upr(w, 2);
                        OL4[i1][i] = upr(w, 3);
                    }
                    break;
            case 6: LOAD_MATR2(OM_6);

for(i=0;i<256;i++)
                    {
                        b = opr_tab[nt_tab[inv_affine((ade_08t)i)][CKey[i1]]];
                        OT1[i1][i] = bytes2word(nt_tab[OM_6[0][0]][i],
                                            nt_tab[OM_6[1][0]][i],
                                            nt_tab[OM_6[2][0]][i],
                                            nt_tab[OM_6[3][0]][i]);
                        OT2[i1][i] = bytes2word(nt_tab[OM_6[0][1]][i],
                                            nt_tab[OM_6[1][1]][i],
                                            nt_tab[OM_6[2][1]][i],
                                            nt_tab[OM_6[3][1]][i]);
                        OT3[i1][i] = bytes2word(nt_tab[OM_6[0][2]][i],
                                            nt_tab[OM_6[1][2]][i],
                                            nt_tab[OM_6[2][2]][i],
                                            nt_tab[OM_6[3][2]][i]);
                        OT4[i1][i] = bytes2word(nt_tab[OM_6[0][3]][i],
                                            nt_tab[OM_6[1][3]][i],
                                            nt_tab[OM_6[2][3]][i],
                                            nt_tab[OM_6[3][3]][i]);
                        w = bytes2word(b, 0, 0, 0);
                        OL1[i1][i] = w;
                        OL2[i1][i] = upr(w, 1);
                        OL3[i1][i] = upr(w, 2);
```

```
 OL4[i1][i] = upr(w, 3);
                    }
                    break;
            case 7: LOAD_MATR2(OM_7);
                    for(i=0;i<256;i++)
                    {
                        b = opr_tab[nt_tab[inv_affine((ade_08t)i)][CKey[i1]]];
                        OT1[i1][i] = bytes2word(nt_tab[OM_7[0][0]][i],
                                                nt_tab[OM_7[1][0]][i],
                                                nt_tab[OM_7[2][0]][i],
                                                nt_tab[OM_7[3][0]][i]);
                        OT2[i1][i] = bytes2word(nt_tab[OM_7[0][1]][i],

nt_tab[OM_7[1][1]][i],
                                                nt_tab[OM_7[2][1]][i],
                                                nt_tab[OM_7[3][1]][i]);
                        OT3[i1][i] = bytes2word(nt_tab[OM_7[0][2]][i],
                                                nt_tab[OM_7[1][2]][i],
                                                nt_tab[OM_7[2][2]][i],
                                                nt_tab[OM_7[3][2]][i]);
                        OT4[i1][i] = bytes2word(nt_tab[OM_7[0][3]][i],
                                                nt_tab[OM_7[1][3]][i],
                                                nt_tab[OM_7[2][3]][i],
                                                nt_tab[OM_7[3][3]][i]);
                        w = bytes2word(b, 0, 0, 0);
                        OL1[i1][i] = w;
                        OL2[i1][i] = upr(w, 1);
                        OL3[i1][i] = upr(w, 2);
                        OL4[i1][i] = upr(w, 3);
                    }
                    break;
            case 8: LOAD_MATR2(OM_8);
                    for(i=0;i<256;i++)
                    {
                        b = opr_tab[nt_tab[inv_affine((ade_08t)i)][CKey[i1]]];
                        OT1[i1][i] = bytes2word(nt_tab[OM_8[0][0]][i],
                                                nt_tab[OM_8[1][0]][i],
                                                nt_tab[OM_8[2][0]][i],
                                                nt_tab[OM_8[3][0]][i]);
                        OT2[i1][i] = bytes2word(nt_tab[OM_8[0][1]][i],
                                                nt_tab[OM_8[1][1]][i],
                                                nt_tab[OM_8[2][1]][i],
                                                nt_tab[OM_8[3][1]][i]);
                        OT3[i1][i] = bytes2word(nt_tab[OM_8[0][2]][i],
                                                nt_tab[OM_8[1][2]][i],
                                                nt_tab[OM_8[2][2]][i],
                                                nt_tab[OM_8[3][2]][i]);
                        OT4[i1][i] = bytes2word(nt_tab[OM_8[0][3]][i],
                                                nt_tab[OM_8[1][3]][i],
                                                nt_tab[OM_8[2][3]][i],
                                                nt_tab[OM_8[3][3]][i]);
                        w = bytes2word(b, 0, 0, 0);
                        OL1[i1][i] = w;
                        OL2[i1][i] = upr(w, 1);
                        OL3[i1][i] = upr(w, 2);
                        OL4[i1][i] = upr(w, 3);
                    }
                    break;
            case 9:
                    for(i=0;i<256;i++)
                    {
                        b = opr_tab[nt_tab[inv_affine((ade_08t)i)][CKey[i1]]];
                        w = bytes2word(b, 0, 0, 0);
                        OL1[i1][i] = w;
                        OL2[i1][i] = upr(w, 1);
                        OL3[i1][i] = upr(w, 2);
                        OL4[i1][i] = upr(w, 3);
                    }
                    break;
          default: break;
        }
    }
    return 1;
}

unsigned long ade_enc_key_R(const unsigned char in_key[], ade_32t k_sch[])
```

```
{
        ade_32t    ss[4];

        k_sch[0] = ss[0] =  (ade_32t)(in_key[0]  << 24) | (ade_32t)(in_key[1]  << 16) |
                                        (ade_32t)(in_key[2]  <<  8) | in_key[3];
        k_sch[1] = ss[1] =  (ade_32t)(in_key[4]  << 24) | (ade_32t)(in_key[5]  << 16) |
                                        (ade_32t)(in_key[6]  <<  8) | in_key[7];
        k_sch[2] = ss[2] =  (ade_32t)(in_key[8]  << 24) | (ade_32t)(in_key[9]  << 16) |
                                        (ade_32t)(in_key[10] <<  8) | in_key[11];
        k_sch[3] = ss[3] =  (ade_32t)(in_key[12] << 24) | (ade_32t)(in_key[13] << 16) |
                                       (ade_32t)(in_key[14] <<  8) | in_key[15];

    k_sch[4] = ss[0]  ^= ( L1[0][(ade_08t)(ss[3] >> 16)]



^ L2[0][(ade_08t)(ss[3] >>  8)]
                         ^ L3[0][(ade_08t) ss[3]]
                         ^ L4[0][(ade_08t)(ss[3] >> 24)])
                         ^ rcon_tab[0];
    k_sch[5] = ss[1]  ^= ss[0];
    k_sch[6] = ss[2]  ^= ss[1];
    k_sch[7] = ss[3]  ^= ss[2];


k_sch[8] = ss[0]  ^= ( L1[1][(ade_08t)(ss[3] >> 16)]
                           ^ L2[1][(ade_08t)(ss[3] >>  8)]
                           ^ L3[1][(ade_08t) ss[3]]
                           ^ L4[1][(ade_08t)(ss[3] >> 24)])
                           ^ rcon_tab[1];
    k_sch[9]  = ss[1] ^= ss[0];
    k_sch[10] = ss[2] ^= ss[1];
    k_sch[11] = ss[3] ^= ss[2];

    k_sch[12] = ss[0] ^= ( L1[2][(ade_08t)(ss[3] >> 16)]
                           ^ L2[2][(ade_08t)(ss[3] >>  8)]
                                    ^ L3[2][(ade_08t) ss[3]]
                                    ^ L4[2][(ade_08t)(ss[3] >> 24)])
                                    ^ rcon_tab[2];
    k_sch[13] = ss[1] ^= ss[0];
    k_sch[14] = ss[2] ^= ss[1];
    k_sch[15] = ss[3] ^= ss[2];

    k_sch[16] = ss[0]^= ( L1[3][(ade_08t)(ss[3] >> 16)]
                                    ^ L2[3][(ade_08t)(ss[3] >>  8)]
                                    ^ L3[3][(ade_08t) ss[3]]
                                    ^ L4[3][(ade_08t)(ss[3] >> 24)])
                                    ^ rcon_tab[3];
    k_sch[17] = ss[1] ^= ss[0];
    k_sch[18] = ss[2] ^= ss[1];
    k_sch[19] = ss[3] ^= ss[2];

    k_sch[20] = ss[0]^= ( L1[4][(ade_08t)(ss[3] >> 16)]
                                    ^ L2[4][(ade_08t)(ss[3] >>  8)]
                                    ^ L3[4][(ade_08t) ss[3]]
                                    ^ L4[4][(ade_08t)(ss[3] >> 24)])
                                    ^ rcon_tab[4];
    k_sch[21] = ss[1] ^= ss[0];
    k_sch[22] = ss[2] ^= ss[1];
    k_sch[23] = ss[3] ^= ss[2];

    k_sch[24] = ss[0]^= ( L1[5][(ade_08t)(ss[3] >> 16)]
                                    ^ L2[5][(ade_08t)(ss[3] >>  8)]
                                    ^ L3[5][(ade_08t) ss[3]]
                                    ^ L4[5][(ade_08t)(ss[3] >> 24)])
                                    ^ rcon_tab[5];
    k_sch[25] = ss[1] ^= ss[0];
    k_sch[26] = ss[2] ^= ss[1];
    k_sch[27] = ss[3] ^= ss[2];

    k_sch[28] = ss[0]^= ( L1[6][(ade_08t)(ss[3] >> 16)]
                                    ^ L2[6][(ade_08t)(ss[3] >>  8)]
                                    ^ L3[6][(ade_08t) ss[3]]
                                    ^ L4[6][(ade_08t)(ss[3] >> 24)])
                                    ^ rcon_tab[6];
```

```
  k_sch[29] = ss[1] ^= ss[0];
    k_sch[30] = ss[2] ^= ss[1];
    k_sch[31] = ss[3] ^= ss[2];

    k_sch[32] = ss[0]^= ( L1[7][(ade_08t)(ss[3] >> 16)]
                                ^ L2[7][(ade_08t)(ss[3] >>  8)]
                                  ^ L3[7][(ade_08t) ss[3]]
                                  ^ L4[7][(ade_08t)(ss[3] >> 24)])
                                  ^ rcon_tab[7];
    k_sch[33] = ss[1] ^= ss[0];
k_sch[34] = ss[2] ^= ss[1];
    k_sch[35] = ss[3] ^= ss[2];

    k_sch[36] = ss[0]^= ( L1[8][(ade_08t)(ss[3] >> 16)]
                            ^ L2[8][(ade_08t)(ss[3] >>  8)]
                                  ^ L3[8][(ade_08t) ss[3]]
```

Продовження дод. Б

```
^ L4[8][(ade_08t)(ss[3] >> 24)])
                                  ^ rcon_tab[8];
    k_sch[37] = ss[1] ^= ss[0];
    k_sch[38] = ss[2] ^= ss[1];
    k_sch[39] = ss[3] ^= ss[2];

    k_sch[40] = ss[0]^= ( L1[9][(ade_08t)(ss[3] >> 16)]
                        ^ L2[9][(ade_08t)(ss[3] >>  8)]
                                  ^ L3[9][(ade_08t) ss[3]]
                                  ^ L4[9][(ade_08t)(ss[3] >> 24)])
                                  ^ rcon_tab[9];
    k_sch[41] = ss[1] ^= ss[0];
    k_sch[42] = ss[2] ^= ss[1];
    k_sch[43] = ss[3] ^= ss[2];

       return 1;
}
```

//Def.h

```
#define WPOLY   0x011D
//  Определение арифметических операций в поле
#define f2(x)   ((x<<1) ^ (((x>>7) & 1) * WPOLY))
#define f4(x)   ((x<<2) ^ (((x>>6) & 1) * WPOLY) ^ (((x>>6) & 2) * WPOLY))
#define f8(x)   ((x<<3) ^ (((x>>5) & 1) * WPOLY) ^ (((x>>5) & 2) * WPOLY) \
                        ^ (((x>>5) & 4) * WPOLY))
#define f9(x)   (f8(x) ^ x)
#define fb(x)   (f8(x) ^ f2(x) ^ x)
#define fd(x)   (f8(x) ^ f4(x) ^ x)
#define fe(x)   (f8(x) ^ f4(x) ^ f2(x))
//  Прямые и обратные аффинные преобразования для формирования S-box
#define fwd_affine(x) \
    (w = (ade_32t)x, w ^= (w<<1)^(w<<2)^(w<<3)^(w<<4), 0x63^(ade_08t)(w^(w>>8)))
#define inv_affine(x) \
    (w = (ade_32t)x, w = (w<<1)^(w<<3)^(w<<6), 0x05^(ade_08t)(w^(w>>8)))
//  Обмен байтами в пределах слова
#define upr(x,n)        ((ade_32t)(x) >> 8 * (n) | (ade_32t)(x) << 32 - 8 * (n))
//  Заполнение байт в пределах слова
#define bytes2word(b0, b1, b2, b3)  \
        (((ade_32t)(b0) << 24) | ((ade_32t)(b1) << 16) | ((ade_32t)(b2) << 8) | (b3))
//  Размерность блока данных
#define BLOCK_LEN   16
#define READ_ERROR  -7
#define WRITE_ERROR -8
#define file_len(x) (unsigned long)x
//  Макрорасширение, используемое для формирования вектора инициализации
#define RAND(a,b) (((a = 36969 * (a & 65535) + (a >> 16)) << 16) + \
                   (b = 18000 * (b & 65535) + (b >> 16))  )

typedef     unsigned char ade_08t;  // Тип данных для хранения данных размерностью 1 байт
typedef     unsigned long ade_32t;  // Тип данных для хранения данных размерностью 4 байта
ade_32t     rcon_tab[10];           // Массив констант для формирования расширенных ключей
ade_32t     fl_tab[4][256];         // Массив для формирования предварительного расширенного ключа
ade_32t     k_sch[44];              // Массив для хранения ключей для каждого из 10 циклов
ade_32t     k_sch1[44];             // Массив для хранения предварительного расширенного ключа

#define C00 c[0][0]
#define C01 c[0][1]
#define C02 c[0][2]
```

```
#define C03 c[0][3]
#define C10 c[1][0]
#define C11 c[1][1]
#define C12 c[1][2]
#define C13 c[1][3]
#define C20 c[2][0]
#define C21 c[2][1]
#define C22 c[2][2]
#define C23 c[2][3]
#define C30 c[3][0]
#define C31 c[3][1]
#define C32 c[3][2]
#define C33 c[3][3]

#define A00 d[0][0]
#define A01 d[0][1]
#define A02 d[0][2]
#define A03 d[0][3]
#define A10 d[1][0]

#define A11 d[1][1]
#define A12 d[1][2]
#define A13 d[1][3]
#define A20 d[2][0]
#define A21 d[2][1]
#define A22 d[2][2]
#define A23 d[2][3]
#define A30 d[3][0]
#define A31 d[3][1]
#define A32 d[3][2]
#define A33 d[3][3]

#define B00 b[0][0]
#define B01 b[0][1]
#define B02 b[0][2]
#define B03 b[0][3]
#define B10 b[1][0]
#define B11 b[1][1]
#define B12 b[1][2]
#define B13 b[1][3]
#define B20 b[2][0]
#define B21 b[2][1]
#define B22 b[2][2]
#define B23 b[2][3]
#define B30 b[3][0]
#define B31 b[3][1]
#define B32 b[3][2]
#define B33 b[3][3]
```