ЗМІСТ

ВСТУП …… 6
РОЗДІЛ 1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ …… 8
1.1. Загальна характеристика UMTS ..…… 8
1.2. Взаємна аутентифікація в UMTS ……... 9
1.3. Набір алгоритмів MILENAGE ……... 10
1.4. Алгоритм конфіденційності даних ……..... 13
1.5. Алгоритм цілісності даних ……. 15
1.6. Алгоритм шифрування KASUMI ……... 16
1.7. Алгоритм SNOW-3G ……... 27
1.8. Поняття аудиту безпеки і цілі його проведення ……... 29
1.9. Криптоаналіз як метод проведення аудиту криптосистем …….. 32
1.10. Висновки до розділу 1 …… 34
РОЗДІЛ 2 КРИПТОАНАЛІЗ АЛГОРИТМУ KASUMI …… 35
2.1. Сендвіч атака …… 35
2.2. Основи бумеранг атаки з пов’язаними ключами …….. 35
2.3. Сендвіч атака з пов’язаними ключами …….. 37
2.4. Прямокутна сендвіч атака ……... 41
2.5. Сендвіч порівнювач з пов’язаними ключами для 7-раундового KASUMI …… 42
2.6. Зміна планування ключів …… 47
2.7. Сендвіч атака з пов’язаними ключами на повний KASUMI …… 51
2.8. Прямокутна сендвіч атака з пов’язаними ключами на повний KASUMI …… 55
2.9. Висновки до розділу 2 …… 56

РОЗДІЛ 3 РЕАЛІЗАЦІЯ ПРОГРАМИ ……... 57
3.1. Реалізація алгоритму KASUMI ……... 57
3.2. Реалізація алгоритму криптоаналізу …….. 58
3.3. Оптимізація з використанням технології MPI …….. 63
3.4. Тестування програми …….. 65
3.5. Профілювання …….. 68
3.6. Висновки до розділу 3 ……. 71
РОЗДІЛ 4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ …… 72
4.1. Охорона праці …….. 72
4.2. Надзвичайні ситуації при пожежах ……... 75
4.3. Висновки до розділу 4 ……. 81
РОЗДІЛ 5 ЕКОЛОГІЯ ……. 82
5.1. Відчуження земель під лінії електропередач …… 82
5.2. Методи узагальнення екологічної інформації …….. 84
5.3. Моніторинг довкілля та система спостережень за впливом на довкілля антропогенних факторів …….. 86
5.4. Висновки до розділу 5 ……. 88
РОЗДІЛ 6 ОБҐРУНТУВАННЯ ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ ……. 89
6.1. Визначення стадій технологічного процесу ……. 89
6.2. Визначення накладних витрат та на оплату праці ……... 90
6.3. Розрахунок витрат на електроенергію …….. 93
6.4. Розрахунок матеріальних витрат та амортизаційних відрахувань .. 93
6.5. Складання кошторису витрат та визначення собівартості …….. 94
6.6. Розрахунок ціни програмного продукту ……... 96
6.7. Визначення економічної ефективності і терміну окупності капітальних вкладень …….. 96
6.8. Висновки до розділу 6 ……. 97

ВИСНОВКИ …… 99
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ …….. 101
АНОТАЦІЯ ……. 104
Додаток А Лістинг класу Kasumi …….. 107
Додаток Б Лістинг програми криптоаналізу …… 112

ВСТУП

Актуальність теми. З розвитком та поширенням телекомунікаційних технологій постає задача забезпечення конфіденційності інформації. Одним із завдань, яке необхідно вирішувати є верифікація існуючих систем захисту. На зміну другому поколінню безпровідних технологій зв'язку приходять стандарти третього покоління, зокрема UMTS. Universal Mobile Telecommunications System (UMTS) – технологія стільникового зв’язку, розроблена Європейським інститутом телекомунікаційних стандартів (ETSI) для впровадження 3G в Європі. Аналіз захищеності інформації в UMTS мережах дає змогу вказати на вразливі місця в системі безпеки, а також сформулювати можливі рекомендації для підвищення ступеня захисту.

Тема дипломної роботи магістра: «Дослідження захищеності криптографічних засобів захисту мережі передачі даних UMTS».

Для забезпечення конфіденційності та цілісності даних у UMTS використовуються алгоритми – UEA1 і UIA1, які базується на блоковому шифрі KASUMI [1]. Це зумовлює актуальність дослідження особливостей алгоритму KASUMI та розробки криптоаналітичних засобів для верифікації відповідних систем захисту.

Метою роботи є аналіз системи захисту інформації в UMTS мережах, виявлення вразливостей даної системи.

Для досягнення поставленої мети необхідно:

1. Проаналізувати систему захисту в UMTS мережі, зокрема криптографічні засоби що використовуються.
2. Проаналізувати особливості використання алгоритму шифрування KASUMI в криптографічних засобах захисту мережі UMTS.
3. Проаналізувати можливі методи криптоаналізу KASUMI та обґрунтовано вибір використовуваного методу.
4. Розробити алгоритм криптоаналізу.

5. Обґрунтувати вибір технології програмування та засобів оптимізації.

6. Реалізувати програму криптоаналізу з використанням засобів паралельних та розподілених обчислень.

7. Здійснити тестування програми та профілювання, зробити аналіз отриманих результатів.

Об’єктом дослідження є криптографічні засоби захисту в UMTS мережах.

Предметом дослідження є криптостійкість алгоритму шифрування KASUMI, що використовується в алгоритмах UEA1 і UIA1.

Наукова новизна одержаних результатів. Удосконалено алгоритм криптоаналізу шифру KASUMI методом сендвіч атаки з пов’язаними ключами шляхом його розпаралелення для забезпечення можливості його реалізації з використанням засобів паралельних та розподілених обчислень.

Результати роботи апробовані на міжнародній науково-практичній конференції молодих учених та студентів «Актуальні задачі сучасних технологій» (Тернопіль, 2012 р.), міжнародній науковій конференції «Питання оптимізації обчислень (ПОО-XL)» (Ялта, 2013 р.), міжнародній науковій конференції «Cluster Computing» (Львів, 2013 р.), ІІ науково-технічній конференції Тернопільського національного технічного університету імені Івана Пулюя «Інформаційні моделі, системи та технології» (Тернопіль, 2012).

РОЗДІЛ 1
ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Загальна характеристика UMTS

Universal Mobile Telecommunications System (UMTS) – технологія третього покоління для стільникових мереж, що базуються на технології GSM. UMTS повністю визначає мережеву систему, яка включає в себе мережу радіодоступу (UMTS Terrestrial Radio Access Network, або UTRAN), базову мережу (прикладна частина або MAP) і аутентифікацію користувачів за допомогою SIM (модуль ідентифікації абонента) [2].

UMTS використовує W-CDMA технологію радіо доступу, що забезпечує високу спектральну ефективність та пропускну здатність для операторів мобільного зв’язку.

Основні механізми захисту в UMTS мережах були перенесені з механізмів захисту мереж GSM. Однак у стандарт були введені деякі удосконалення:

– взаємна аутентифікація (тепер не тільки абонент аутентифікований мережею, але і абонент аутентифікує мережу, щоб усунути можливість атаки з використанням хибної базової станції);

– захист цілісності (введені вдосконалені алгоритми і ключі для забезпечення цілісності даних);

– мережева безпека (забезпечує шифрування всередині і між різними мережами);

– безпека сервісів і додатків (вдосконалені механізми забезпечення надійності для сервісів і додатків);

– взаємодія між мережами і роумінг.

UMTS став першим масовим стандартом бездротового зв'язку, що забезпечує серйозний рівень захисту інформації користувача.

Для забезпечення конфіденційності та цілісності даних у UMTS використовуються два набори алгоритмів: перший набір – UEA1 і UIA1 [1], який базується на блоковому шифрі KASUMI; другий – UEA2 і UIA2 — на потоковому шифрі SNOW 3G. Використання тих чи інших систем шифрування обумовлено апаратним забезпеченням оператора зв'язку, абонента, а також аспектами роумінгу абонентських терміналів покоління 2G. Тому досить часто використовуються алгоритми UEA1/UIA1, а це зумовлює актуальність розробки відповідних криптоаналітичних засобів для верифікації відповідних систем захисту.

1.2. Взаємна аутентифікація в UMTS

У механізмі аутентифікації в UMTS системах залучені три елементи:

- домашнє середовище та Authentication Center (AuC);
- обслуговуюча мережа з Visitor Location Register (VLR);
- термінал, конкретніше – Universal Subscriber Identity Module (USIM).

Так як і в GSM, обслуговуюча мережа перевіряє особу користувача методикою відповіді на запит, поки термінал перевіряє, що обслуговуюча мережа авторизована середовищем. Дана функція є новою особливістю UMTS в порівнянні з GSM, тобто термінал може перевірити, що під’єднаний до легітимної мережі [2, 3].

AuC має копію ключа K абонента. За запитом від обслуговуючої мережі AuC одержує вектори аутентифікації з п’ятьох компонент - RAND, XRES, CK, IK та AUTN, чотири з яких обчислюються з RAND, IK та послідовності чисел SQN. В обслуговуючій мережі, один вектор аутентифікації потрібен для кожної операції аутентифікації. Обслуговуюча мережа відправляє запит аутентифікації користувача до мобільного терміналу. Дане повідомлення містить два параметри з вектора аутентифікації – RAND і AUTH. Ці параметри направляють до модуля USIM,

що розташований в середині середовища Universal Integrated Circuit Card (UICC). USIM містить ключ K та використовує його з параметрами RAND і AUTH, для виконання обчислення, використовуючи Authentication and Key Agreement (AKA) алгоритм. Результат обчислень дає можливість USIM перевірити чи параметр AUTH дійсно генерований в AuC, та не надсилався до USIM раніше. Ця перевірка базується на значенні SQN. Для цього два лічильники підтримують синхронізацію в AuC та USIM. Якщо USIM проходить перевірку, тоді обчислена відповідь RES надсилається назад до обслуговуючої мережі в повідомленні-відповіді аутентифікації абонента. Обслуговуюча мережа порівнює RES з очікуваною відповіддю XRES, яка включена у вектор аутентифікації. У випадку співпадіння аутентифікація закінчується позитивно.

1.3. Набір алгоритмів MILENAGE

В загальному, використовуються п’ять функцій для обчислення вектора аутентифікації. Дані функції позначені як f1, f2, f3, f4 та f5. Їх вибір залежить від специфіки оператора тому, що вони використовуються лише в AuC та в USIM, і один домашній оператор контролює їх обох. Проте, для досягнення сумісності різних USIM реалізацій з AuC версіями простіше використати стандартизований алгоритм. Також проектування та впровадження стійкого криптографічного алгоритму ніколи не було простим завданням, і не є доступною можливістю для всіх операторів. 3GPP забезпечила приклад набору AKA алгоритмів, що можуть бути використані операторами [2, 5].

Даний набір AKA алгоритмів назвали MILENAGE. Дана конструкція використовує стійкий 128-бітний алгоритм шифрування як функцію ядра. Рекомендується використання алгоритму AES як функції ядра, але оператор може змінити його на будь-який блоковий шифр, що підходить за вимогами.

Основна криптографічна стійкість MILENAGE залежить від блокового шифру ядра. Проведено багато екстенсивних криптоаналізів обраного

алгоритму AES. Але всі виявлені вразливості не представляють ніякої практичної загрози. Вони лише означають, що теоретична безпека ядра MILENAGE є дещо меншою від передбачуваної стійкості 128-бітного шифру.

Найбільш відмінні особливості MILENAGE можуть бути представлені в наступній спрощеній функціональній моделі:

$$z_i = E_K(E_K(x) \oplus a_i) , \qquad (1.1)$$

де z_i – вихідні дані;

E_K – функція шифрування;

x – вхідні дані;

a_i – параметрична змінна;

i – лічильник, 0, 1, 2, … t.

Параметр t є фіксованим, і визначає число різних вихідних блоків розміром n з блокового шифру E_K. Для MILENAGE конструкції функцій від f2 до f5* – t = 4.

Для підтвердження безпеки таких конструкцій, потрібно показати, що немає способу використання будь-яких комбінацій істотно менших за $2^{n/2}$ вихідних значень z_1, z_2, …, z_t для передбачення будь-яких нових вихідних значень для будь-якого блоку z_i. Тобто, потрібно показати, що якщо E_K виконується як випадкова перестановка набору $\{0, 1\}^n$, тоді функція f, яка відображає x на кортежі z_1, z_2, …, z_t, не може бути порівняна з випадковою функцією f* з $\{0, 1\}^n$ до $\{0, 1\}^{nt}$ ефективним способом.

Порівняння базується на довільних порівнювальних алгоритмах (порівнювачі) A з необмеженими обчислювальними потужностями. Це подається як чорний ящик, який містить функцію. Функція всередині є або функцією f, або функцією f*. Тоді порівнювач дає змогу зробити фіксоване число запитів щодо вихідних значень функції, для різних обраних або адаптивно обраних вхідних значень. Після отримання результатів,

порівнювач повертає 1-бітне значення 0 або 1. Позначимо p – ймовірність, що A поверне 1, коли функція всередині – f*. Тоді, можна сказати, що функція f не може бути порівняна з ідеальною випадковою функцією f*, якщо для будь-якого порівнювача A ймовірності p і p* будуть таким ж. Абсолютне значення |p-p*| називається перевагою A, і позначається Adv A(f, f*).

Функції f0-f5 використовуються для забезпечення взаємної аутентифікації між USIM та AuC, тобто для отримання ключів, щоб захистити дані, які передаються через канал радіодоступу, а також для приховання SQN, щоб захистити конфіденційність особи абонента. Функція f1* лише використовується для забезпечення аутентифікації даних , при виникненні помилки синхронізації. Функція f5* використовується для забезпечення конфіденційності особи абонента при ресинхронізації.

Функція f0 – здійснює псевдо-випадкове генерування чисел, і відображає внутрішній стан генератора у змінну RAND.

На рис. 1.1 представлена схема обчислення MILENAGE функцій.

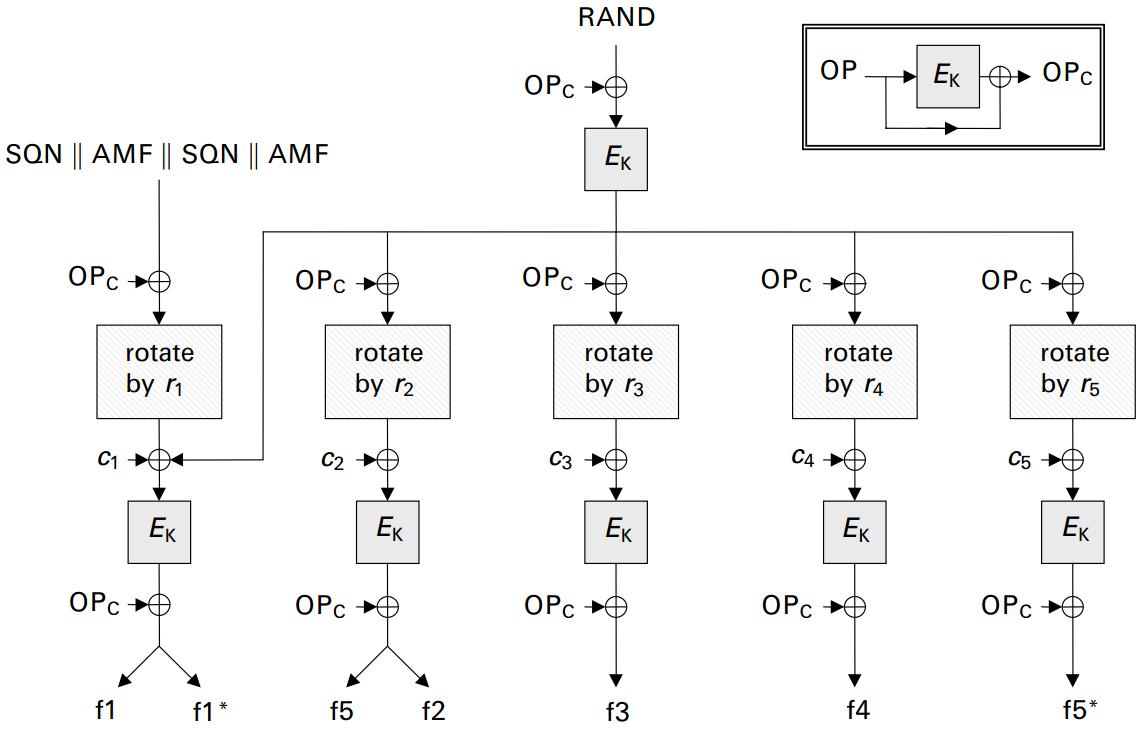

Рис. 1.1. Обчислення MILENAGE

OP – 128-бітне конфігураційне поле алгоритму обраного оператором. OP_C – 128-бітне значення, отримане з OP та K. RAND генерується функцією f0. SQN – послідовність, що генерується в AuC. AMF – поле управлінням аутентифікацією. Константи r_i та c_i представлені в таблиці 1.1.

Таблиця 1.1

Константи r_i та c_i

i	r_i	c_i
1	64	000…00000
2	0	000…00001
3	32	000…00010
4	64	000…00100
5	96	000…01000

Функції f1-f5, f1* і f5* реалізовуються в AuC та USIM. Функція f0 реалізовується лише в AuC.

1.4. Алгоритм конфіденційності даних

Алгоритм конфіденційності f8 – потоковий шифр, який використовується для шифруванні і дешифрування блоків даних з секретним ключем (СК) [1]. Розмір блоку може бути в межах від 1 до 20000 біт. Алгоритм використовує KASUMI в режимі OFB як генератор потокового ключа. OFB – режим зворотного зв’язку, який перетворює блочний шифр в синхронний потік.

Алгоритм f8 використовує два 64-бітні регістри: статичний регістр A і лічильник BLKCNT. Ініціалізація A здійснюється з використанням 64-бітної змінної ініціалізації IV:

$$IV = COUNT \,||\, BEARER \,||\, DIRECTION \,||\, 0 \ldots 0\,, \qquad (1.2)$$

де COUNT – 32-бітна стрічка;

BEARER – 5 бітна стрічка;

DIRECTION – 1-бітна стрічка.

IV отримується шляхом з'єднання 32 біт COUNT, 5 біт BEARER, 1 біта DIRECTION і 26 нульових бітів. Змінна COUNT ініціалізується під час встановлення з'єднання. Змінна BEARER ідентифікує об'єкт, з яким встановлено з'єднання. Змінна DIRECTION вказує напрямок передачі, «0» – для вихідного з'єднання, «1» – для вхідного з'єднання. Початкове значення лічильника BLKCNT встановлюється в 0. Схема генератора потокового ключа показана на рис. 1.2.

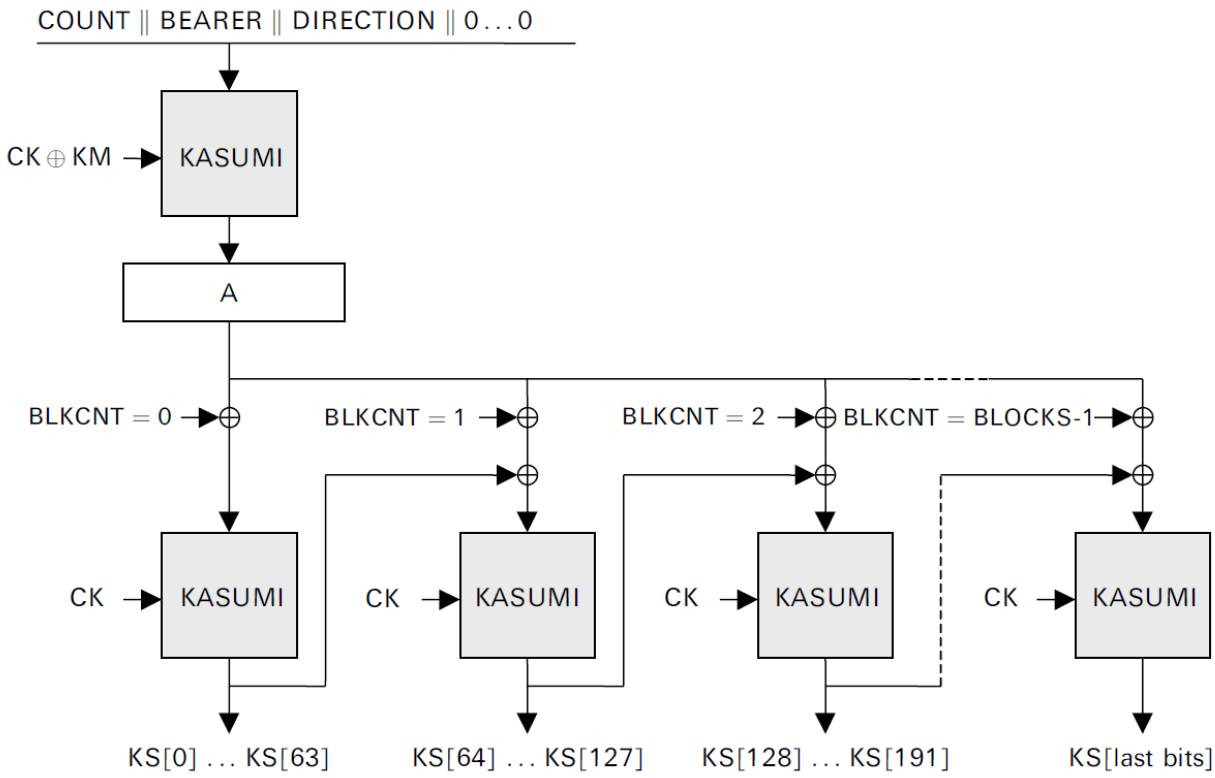

Рис. 1.2. Генератор потокового ключа

f8 використовує константу модифікатора ключа (МК), яка рівна октету 0x55 = 01010101, повтореного 16 разів.

1.5. Алгоритм цілісності даних

3GPP алгоритм цілісності даних f9 обчислює 32-бітне значення Message Authentication Code (MAC) для вхідного повідомлення з ключем IK і не накладає жодних обмежень на довжину вхідного повідомлення [1].

Для легкого застосування, алгоритм базується на блочному шифрі KASUMI, який використовується в алгоритмі конфіденційності даних f8. KASUMI використовується в модифікованій процедурі CBC-MAC; відрізняє її від стандартної процедури – додавання операції, яка об'єднує всі проміжні вихідні значення, використовуючи побітовий XOR, і застосовує додаткове шифрування KASUMI на комбінований блок. 64-бітне вихідне значення з останнього шифрування KASUMI усікається до 32-бітного значення MAC-I.

Стандартна f9 функція використовує два 64-бітні регістри A і B. Початкові значення обох регістрів 0. Функція також використовує 128-бітну константу KM, яка рівна 16 повторам октету 0xAA = 10101010. Всі вхідні значення з'єднуються в єдине, потім в кінець додається один біт «1», а за ним від 0 до 63 біт «0», щоб загальна довжина була кратна 64. Дана стрічка називається Padded String (PS). Отже:

$$PS = COUNT \,||\, FRESH \,||\, MESSAGE \,||\, DIRECTION \,||\, 1 \,||\, 0 \ldots 0 , \qquad (1.3)$$

де COUNT – 32-бітна стрічка;
FRESH – 32-бітна стрічка;
MESSAGE – стрічка довільної довжини;
DIRECTION – 1-бітна стрічка

Змінна COUNT ініціалізується під час встановлення з'єднання. Змінна FRESH забезпечує відсутність повторів старих MAC-I. Змінна DIRECTION вказує напрямок передачі, «0» – для вихідного з'єднання, «1» – для вхідного з'єднання. Змінна MESSAGE – сигнальні дані.

PS ділиться на 64-бітні блоки PS_i. Очевидно, що перший блок буде:

$$PS_0 = COUNT \,||\, FRESH\ . \qquad (1.4)$$

На рис. 1.2 показана схема обчислення коду аутентифікації повідомлення.

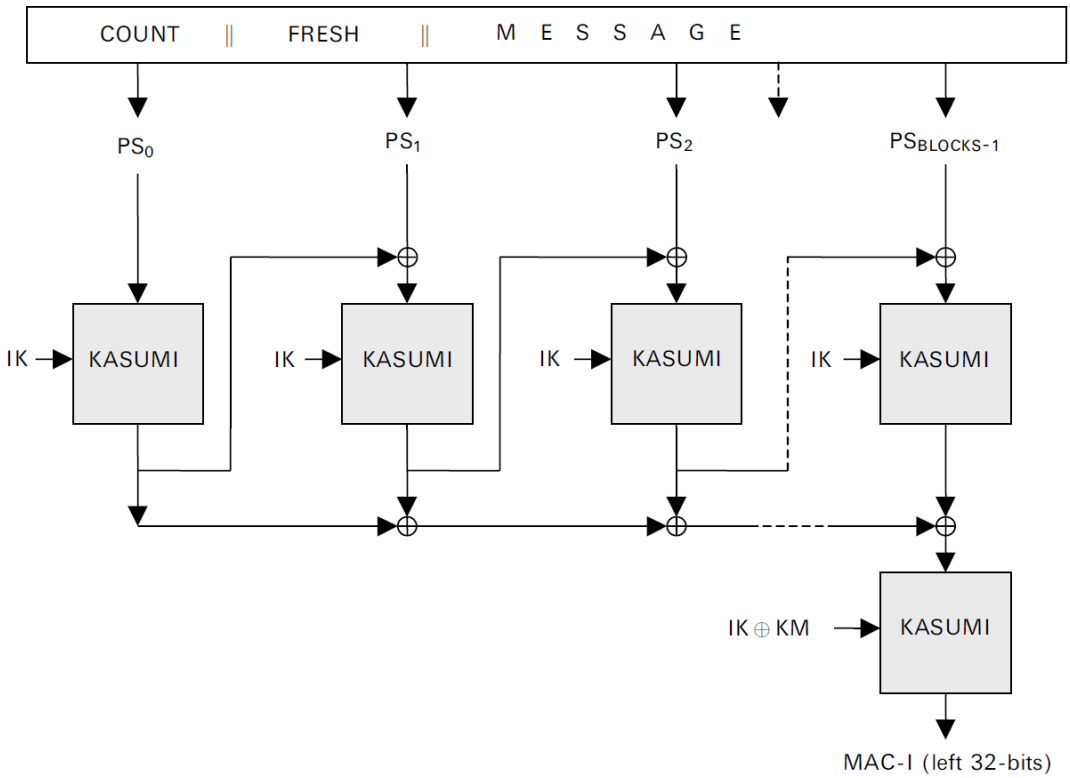

Рис. 1.3 Обчислення коду аутентифікації повідомлення

1.6. Алгоритм шифрування KASUMI

KASUMI – 64 бітний блочний шифр з 128-бітним ключем. Він має рекурсивну мережу Фейстеля, так як і його попередник MISTY. Шифр має 8 раундів мережі Фейстеля [5]. Схема шифрування KASUMI зображена рис. 1.4.

KASUMI оперує 64-бітними блоками вхідних даних (INPUT) з використанням 128-бітного ключа (K) для створення 64-бітного блоку вихідних даних (OUTPUT). Вхідні дані діляться на 2 32-бітні стрічки:

$$INPUT = L_0 \,||\, R_0 \,, \tag{1.5}$$

де L_0 – ліва 32-бітна частина вхідного блоку;

R_0 – права 32-бітна частина вхідного блоку.

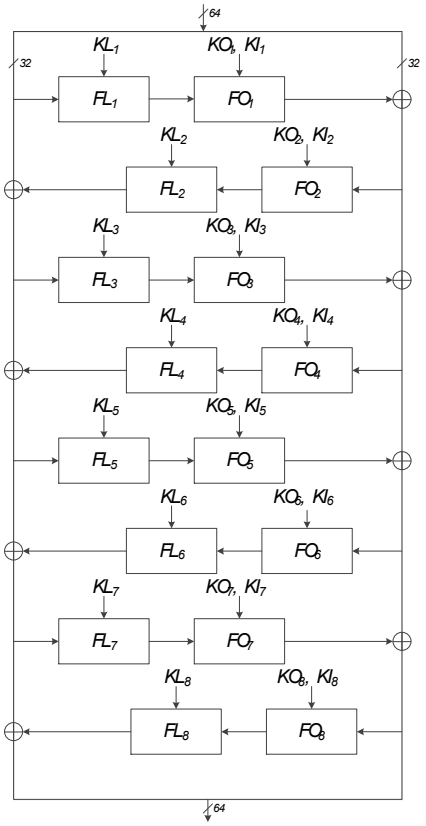

Рис. 1.4. Схема шифрування KASUMI

В кожному з 8 раундів здійснюються операції:

$$R_i = L_{i-1} \,, \tag{1.6}$$

$$L_i = R_{i-1} \oplus f_i(L_{i-1}, RK_i) \,, \tag{1.7}$$

де i – лічильник раунду;

L_{i-1} – ліва частина вхідного блоку;

R_{i-1} – права частина вхідного блоку;

R_i – права частина вихідного блоку;

L_i – ліва частина вихідного блоку;

f_i – операція шифрування одного раунду;

RK_i – раундовий ключ.

Вихідний блок складається з двох 32-бітних стрічок:

$$OUTPUT = L_8 \,||\, R_8 \,, \tag{1.8}$$

де L_8 – ліва частина вихідного блоку 8 раунду;

R_8 – права частина вихідного блоку 8 раунду.

Кожна f_i функція приймає 32-бітне вхідне значення (I) і повертає 32-бітне вихідне значення (O) під управлінням раундового ключа RK_i, який складається з трійки субключів (KL_i, KO_i, KI_i). f_i функція складається з двох функцій: FL_i і FO_i, з субключами KL_i (використовується в FL_i), KO_i і KI_i (використовуються в FO_i).

Функція f_i змінює свою форму в залежності від того, чи є раунд непарним чи парним. Для першого, третього, п’ятого і сьомого раундів f_i визначається як:

$$f_i(I, RK_i) = FO_i(FL_i(I, KL_i), KO_i, KI_i) \,. \tag{1.9}$$

Для другого, четвертого, шостого і восьмого раундів f_i визначається як:

$$f_i(I, RK_i) = FL_i(FO_i(I, KO_i, KI_i), KL_i) \,. \tag{1.10}$$

Функція FL_i приймає 32-бітне вхідне значення і 32-бітний субключ KL_i. Субключ ділиться на два 16-бітних ключа:

$$KL_i = KL_{i,1} \parallel KL_{i,1} , \quad (1.11)$$

де $KL_{i,1}$ – перша 16-бітна частина ключа;

$KL_{i,2}$ – друга 16-бітна частина ключа.

Блок вхідних даних ділиться на дві частини: ліва 16-бітна частина L та права 16-бітна частина R. В FL функції виконуються 3 операції: побітова операція І, побітова операція АБО та зсув вліво на один біт. Вихідні значення FL_i функції визначаються за формулою:

$$L' = L \oplus ROL(R' \cup KL_{i,2}) , \quad (1.12)$$

$$R' = R \oplus ROL(L' \cap KL_{i,1}) , \quad (1.13)$$

де L' – ліва частина вихідного блоку;

R' – права частина вихідного блоку;

ROL() – зсув на один біт.

Загальна схема FL функції представлена на рис. 1.5.

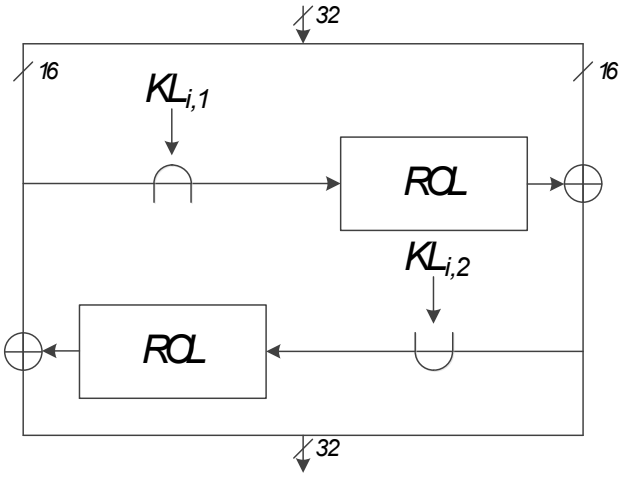

Рис. 1.5. Схема FL функції

FO функція має рекурсивну структуру, тобто є мережею Фейстеля з трьома раундами. Вона приймає 32 бітне вхідне значення і два набори

субключів: 48-бітний KO_i і 48-бітний KI_i. 32-бітний вхідний блок ділиться на дві 16-бітні частина L_0 і R_0. В кожному раунді функції FO_i виконуються операції:

$$R_j = FI_{i,j}(L_{j-1} \oplus KO_{i,j}, KI_{i,j}) \oplus R_{j-1}\ , \tag{1.14}$$

$$L_j = R_{j-1}\ , \tag{1.15}$$

де j – лічильник;

R_j – вихідне 16-бітне значення j-го раунду;

L_j – вихідне 16-бітне значення j-го раунду;

$FI_{i,j}$ – функція FI;

R_{j-1} – значення отримане з попереднього раунду;

L_{j-1} – значення отримане з попереднього раунду.

Загальна схема функції FO показана на рис. 1.6.

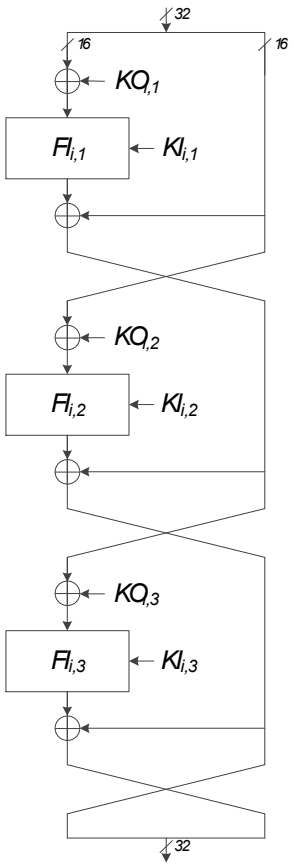

Рис. 1.6. Схема функції FO

Вихідне значення функції FO_i – 32-бітний блок $L_3 \parallel R_3$.

Функція $FI_{i,j}$ є мережею Фейстеля з чотирма раундами, яка приймає 16-бітне значення та 16-бітний субключ $KI_{i,j}$. вхідний блок розділяється на 2 частини: 9-бітна частина L_0 і 7-бітна частина R_0. За таким же ж принципом розділяється субключ $KI_{i,j}$ на компоненти $KI_{i,j,1}$ і $KI_{i,j,2}$.

В кожному раунді функції FI використовуються S-бокси: S7, який з вхідного 7-бітного значення генерує вихідне 7-бітне значення, і S9, який з вхідного 9-бітного значення генерує вихідне 9-бітне значення.

Загальна схема функції FI показана на рис. 1.7.

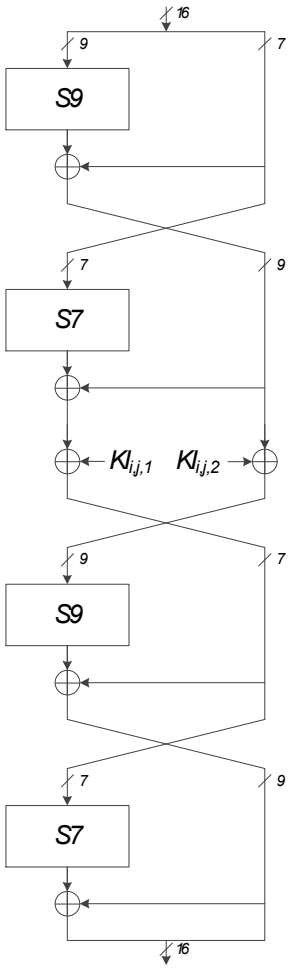

Рис. 1.7 Схема функції FI

В функції FI використовуються дв і додаткові операції ZE і TR. ZE – приймає 7-бітне вхідне значення і конвертує в 9-бітне вихідне значення, додаючи зліва 2 нульових біти. TR – приймає 9-бітне вхідне значення і генерує 7-бітне вихідне значення, шляхом відкидання двох старших біт.

Функція $FI_{i,j}$ визначається серією операцій, що представлені в таблиці 1.2.

Таблиця 1.2

Операції функції $FI_{i,j}$

Раунд функції	L_j	R_j
1	$L_1 = R_0$	$R_1 = S9[L_0] \oplus ZE(R_0)$
2	$L_2 = R_1 \oplus KI_{i,j,2}$	$R_2 = S7[L_1] \oplus TR(R_1) \oplus KI_{i,j,1}$
3	$L_3 = R_2$	$R_3 = S9[L_2] \oplus ZE(R_2)$
4	$L_4 = S7[L_3] \oplus TR(R_3)$	$R_4 = R_3$

Вихідні дані функції $FI_{i,j}$ – 16-бітний блок $L_4 \| R_4$.

Два S-бокси спроектовані так, що вони можуть легко впроваджені як комбінаційна логіка чи оглядова таблиця. Вхідне значення x складаються з семи чи дев’яти біт, з відповідним числом біт вихідного значення:

$$x = x_8 \| x_7 \| x_6 \| x_5 \| x_4 \| x_3 \| x_2 \| x_1 \| x_0 , \quad (1.16)$$

$$y = y_8 \| y_7 \| y_6 \| y_5 \| y_4 \| y_3 \| y_2 \| y_1 \| y_0 , \quad (1.17)$$

де x_8, y_8 і x_7, y_7 – біти, що використовуються лише в S9;

x_0 і y_0 – молодші біти.

Логічні вирази, що визначають S7, зведені в таблицю 1.3.

Таблиця 1.3

Логічні вирази для S7

Біти вихідного значення	Логічні вирази
y_0	$x_1x_3 \oplus x_4 \oplus x_0x_1x_4 \oplus x_5 \oplus x_2x_5 \oplus x_3x_4x_5 \oplus x_6 \oplus x_0x_6 \oplus x_1x_6 \oplus x_3x_6 \oplus x_2x_4x_6 \oplus x_1x_5x_6 \oplus x_4x_5x_6$
y_1	$x_0x_1 \oplus x_0x_4 \oplus x_2x_4 \oplus x_5 \oplus x_1x_2x_5 \oplus x_0x_3x_5 \oplus x_6 \oplus x_0x_2x_6 \oplus x_3x_6 \oplus x_4x_5x_6 \oplus 1$
y_2	$x_0 \oplus x_0x_3 \oplus x_2x_3 \oplus x_1x_2x_4 \oplus x_0x_3x_4 \oplus x_1x_5 \oplus x_0x_2x_5 \oplus x_0x_6 \oplus x_0x_1x_6 \oplus x_2x_6 \oplus x_4x_6 \oplus 1$
y_3	$x_1 \oplus x_0x_1x_2 \oplus x_1x_4 \oplus x_3x_4 \oplus x_0x_5 \oplus x_0x_1x_5 \oplus x_2x_3x_5 \oplus x_1x_4x_5 \oplus x_2x_6 \oplus x_1x_3x_4x_6$
y_4	$x_0x_2 \oplus x_3 \oplus x_1x_3 \oplus x_1x_4 \oplus x_0x_1x_4 \oplus x_2x_3x_4 \oplus x_0x_5 \oplus x_1x_3x_5 \oplus x_0x_4x_5 \oplus x_1x_6 \oplus x_3x_6 \oplus x_0x_3x_6 \oplus x_5x_6 \oplus 1$
y_5	$x_2 \oplus x_0x_2 \oplus x_0x_3 \oplus x_1x_2x_3 \oplus x_0x_2x_4 \oplus x_0x_5 \oplus x_2x_5 \oplus x_4x_5 \oplus x_1x_6 \oplus x_1x_2x_6 \oplus x_0x_3x_6 \oplus x_3x_4x_6 \oplus x_2x_5x_6 \oplus 1$
y_6	$x_1x_2 \oplus x_0x_1x_3 \oplus x_0x_4 \oplus x_1x_5 \oplus x_3x_5 \oplus x_6 \oplus x_0x_1x_6 \oplus x_2x_3x_6 \oplus x_1x_4x_6 \oplus x_0x_5x_6$

Оглядова таблиця S7 представлена у формі масиву значень:

```
int S7[128] = {
   54, 50, 62, 56, 22, 34, 94, 96, 38,  6, 63, 93,  2, 18,123, 33,
   55,113, 39,114, 21, 67, 65, 12, 47, 73, 46, 27, 25,111,124, 81,
   53,  9,121, 79, 52, 60, 58, 48,101,127, 40,120,104, 70, 71, 43,
   20,122, 72, 61, 23,109, 13,100, 77,  1, 16,  7, 82, 10,105, 98,
  117,116, 76, 11, 89,106,  0,125,118, 99, 86, 69, 30, 57,126, 87,
  112, 51, 17,  5, 95, 14, 90, 84, 91,  8, 35,103, 32, 97, 28, 66,
  102, 31, 26, 45, 75,  4, 85, 92, 37, 74, 80, 49, 68, 29,115, 44,
   64,107,108, 24,110, 83, 36, 78, 42, 19, 15, 41, 88,119, 59,  3
};
```

Логічні вирази, що визначають S9, зведені в таблицю 1.4.

Таблиця 1.4

Логічні вирази для S9

Біти вихідного значення	Логічні вирази
y_0	$x_0x_2 \oplus x_3 \oplus x_2x_5 \oplus x_5x_6 \oplus x_0x_7 \oplus x_1x_7 \oplus x_2x_7 \oplus x_4x_8 \oplus x_5x_8 \oplus x_7x_8 \oplus 1$
y_1	$x_1 \oplus x_0x_1 \oplus x_2x_3 \oplus x_0x_4 \oplus x_1x_4 \oplus x_0x_5 \oplus x_3x_5 \oplus x_6 \oplus x_1x_7 \oplus x_2x_7 \oplus x_5x_8 \oplus 1$
y_2	$x_1 \oplus x_0x_3 \oplus x_3x_4 \oplus x_0x_5 \oplus x_2x_6 \oplus x_3x_6 \oplus x_5x_6 \oplus x_4x_7 \oplus x_5x_7 \oplus x_6x_7 \oplus x_8 \oplus x_0x_8 \oplus 1$
y_3	$x_0 \oplus x_1x_2 \oplus x_0x_3 \oplus x_2x_4 \oplus x_5 \oplus x_0x_6 \oplus x_1x_6 \oplus x_4x_7 \oplus x_0x_8 \oplus x_1x_8 \oplus x_7x_8$
y_4	$x_0x_1 \oplus x_1x_3 \oplus x_4 \oplus x_0x_5 \oplus x_3x_6 \oplus x_0x_7 \oplus x_6x_7 \oplus x_1x_8 \oplus x_2x_8 \oplus x_3x_8$
y_5	$x_2 \oplus x_1x_4 \oplus x_4x_5 \oplus x_0x_6 \oplus x_1x_6 \oplus x_3x_7 \oplus x_4x_7 \oplus x_6x_7 \oplus x_5x_8 \oplus x_6x_8 \oplus x_7x_8 \oplus 1$
y_6	$x_0 \oplus x_2x_3 \oplus x_1x_5 \oplus x_2x_5 \oplus x_4x_5 \oplus x_3x_6 \oplus x_4x_6 \oplus x_5x_6 \oplus x_7 \oplus x_1x_8 \oplus x_3x_8 \oplus x_5x_8 \oplus x_7x_8$
y_7	$x_0x_1 \oplus x_0x_2 \oplus x_1x_2 \oplus x_3 \oplus x_0x_3 \oplus x_2x_3 \oplus x_4x_5 \oplus x_2x_6 \oplus x_3x_6 \oplus x_2x_7 \oplus x_5x_7 \oplus x_8 \oplus 1$
y_8	$x_0x_1 \oplus x_2 \oplus x_1x_2 \oplus x_3x_4 \oplus x_1x_5 \oplus x_2x_5 \oplus x_1x_6 \oplus x_4x_6 \oplus x_7 \oplus x_2x_8 \oplus x_3x_8$

Оглядова таблиця S7 представлена у формі масиву значень:

```
int S9[512] = {
  167,239,161,379,391,334,  9,338, 38,226, 48,358,452,385, 90,397,
  183,253,147,331,415,340, 51,362,306,500,262, 82,216,159,356,177,
```

```
  175,241,489, 37,206, 17,  0,333, 44,254,378, 58,143,220, 81,400,
   95,  3,315,245, 54,235,218,405,472,264,172,494,371,290,399, 76,
  165,197,395,121,257,480,423,212,240, 28,462,176,406,507,288,223,
  501,407,249,265, 89,186,221,428,164, 74,440,196,458,421,350,163,
  232,158,134,354, 13,250,491,142,191, 69,193,425,152,227,366,135,
  344,300,276,242,437,320,113,278, 11,243, 87,317, 36, 93,496, 27,
  487,446,482, 41, 68,156,457,131,326,403,339, 20, 39,115,442,124,
  475,384,508, 53,112,170,479,151,126,169, 73,268,279,321,168,364,
  363,292, 46,499,393,327,324, 24,456,267,157,460,488,426,309,229,
  439,506,208,271,349,401,434,236, 16,209,359, 52, 56,120,199,277,
  465,416,252,287,246,  6, 83,305,420,345,153,502, 65, 61,244,282,
  173,222,418, 67,386,368,261,101,476,291,195,430, 49, 79,166,330,
  280,383,373,128,382,408,155,495,367,388,274,107,459,417, 62,454,
  132,225,203,316,234, 14,301, 91,503,286,424,211,347,307,140,374,
   35,103,125,427, 19,214,453,146,498,314,444,230,256,329,198,285,
   50,116, 78,410, 10,205,510,171,231, 45,139,467, 29, 86,505, 32,
   72, 26,342,150,313,490,431,238,411,325,149,473, 40,119,174,355,
  185,233,389, 71,448,273,372, 55,110,178,322, 12,469,392,369,190,
    1,109,375,137,181, 88, 75,308,260,484, 98,272,370,275,412,111,
  336,318,  4,504,492,259,304, 77,337,435, 21,357,303,332,483, 18,
   47, 85, 25,497,474,289,100,269,296,478,270,106, 31,104,433, 84,
  414,486,394, 96, 99,154,511,148,413,361,409,255,162,215,302,201,
  266,351,343,144,441,365,108,298,251, 34,182,509,138,210,335,133,
  311,352,328,141,396,346,123,319,450,281,429,228,443,481, 92,404,
  485,422,248,297, 23,213,130,466, 22,217,283, 70,294,360,419,127,
  312,377,  7,468,194,  2,117,295,463,258,224,447,247,187, 80,398,
  284,353,105,390,299,471,470,184, 57,200,348, 63,204,188, 33,451,
   97, 30,310,219, 94,160,129,493, 64,179,263,102,189,207,114,402,
  438,477,387,122,192, 42,381,  5,145,118,180,449,293,323,136,380,
   43, 66, 60,455,341,445,202,432,  8,237, 15,376,436,464, 59,461
};
```

Кожен раунд KASUMI використовує 128-біт, які отримуються з ключа K. Перед обчисленням раундових субключів, знаходиться масив з 16-бітних значень наступним методом:

$$K_j' = K_j \oplus C_j , \qquad (1.18)$$

де $2C_j$ – константа, яка показана в таблиці 1.5.

Таблиця 1.5

Константа C_j

C_j	Значення
C_1	0x0123
C_2	0x4567
C_3	0x89AB
C_4	0xCDEF
C_5	0xFEDC
C_6	0xBA98
C_7	0x7654
C_8	0x3210

Кожен 128-бітний раундів субключ – лінійно модифікований вихідний ключ. Деталі планування субключів показано в таблиці 1.6.

Таблиця 1.6

Планування ключів

Раунд	$KL_{i,1}$	$KL_{i,2}$	$KO_{i,1}$	$KO_{i,2}$	$KO_{i,3}$	$KI_{i,1}$	$KI_{i,2}$	$KI_{i,3}$
1	K_1 <<< 1	K_3’	K_2 <<< 5	K_6 <<< 8	K_7 <<< 13	K_5’	K_4’	K_8’
2	K_2 <<< 1	K_4’	K_3 <<< 5	K_7 <<< 8	K_8 <<< 13	K_6’	K_5’	K_1’
3	K_3 <<< 1	K_5’	K_4 <<< 5	K_8 <<< 8	K_1 <<< 13	K_7’	K_6’	K_2’
4	K_4 <<< 1	K_6’	K_5 <<< 5	K_1 <<< 8	K_2 <<< 13	K_8’	K_7’	K_3’
5	K_5 <<< 1	K_7’	K_6 <<< 5	K_2 <<< 8	K_3 <<< 13	K_1’	K_8’	K_4’
6	K_6 <<< 1	K_8’	K_7 <<< 5	K_3 <<< 8	K_4 <<< 13	K_2’	K_1’	K_5’
7	K_7 <<< 1	K_1’	K_8 <<< 5	K_4 <<< 8	K_5 <<< 13	K_3’	K_2’	K_6’
8	K_8 <<< 1	K_2’	K_1 <<< 5	K_5 <<< 8	K_6 <<< 13	K_4’	K_3’	K_7’
(X <<< i) – циклічний зсув X вліво на і біт								

1.7. Алгоритм SNOW-3G

Потоковий алгоритм шифрування SNOW-3G використовується для забезпечення захищеності каналів комунікацій, а саме в алгоритмах забезпечення цілісності та конфіденційності інформації UIA2 та UEA2 [6].

SNOW-3G генерує послідовність з 32-бітових слів з вхідного 128-бітного ключа та 128-бітної встановленої змінної. Спочатку виконується встановлення ключа, і лише після того здійснюється генерування 32-бітних слів. На рис. 1.8 представлена схема функціонування алгоритму SNOW-3G.

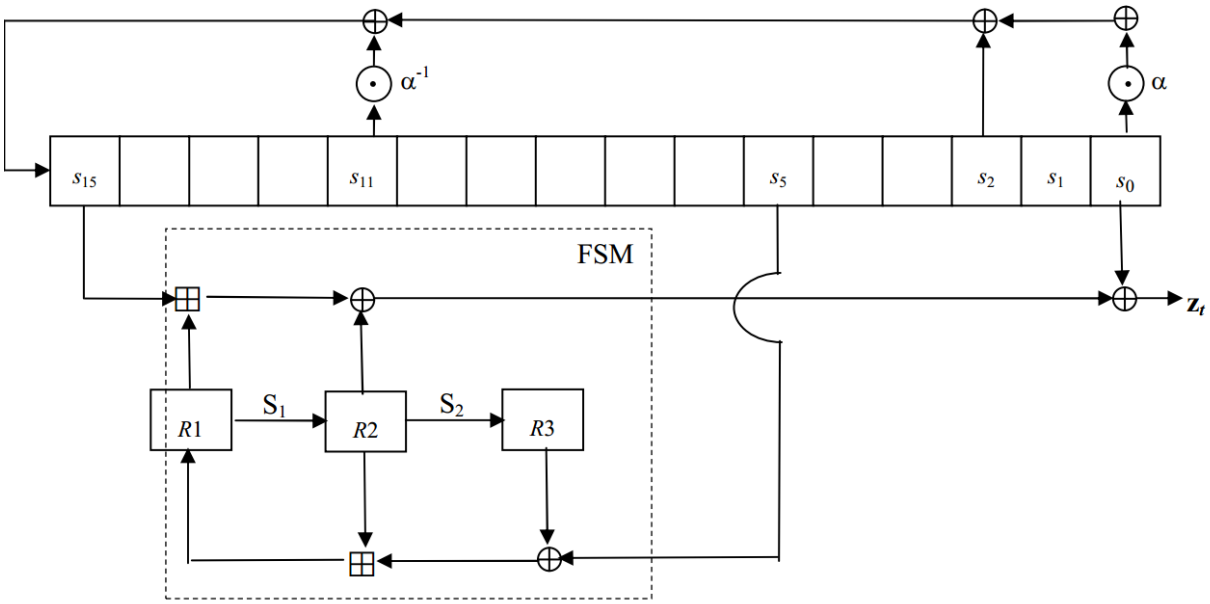

Рис. 1.8. Схема функціонування алгоритму SNOW 3G

В даному алгоритмі використовується функція MULx, яка здійснює перетворення двох 8-бітних значень в одне 8-бітне згідно з формулою:

$$MULx(V, c) = (V <<_8 1) \oplus c\ . \qquad (1.19)$$

Функція MULxPOW виконує перетворення двох 8-бітних значень та цілого числа i в одне 8-бітне згідно з формулою:

$$MULxPOW(V, i, c) = MULx(MULxPOW(V, i - 1, c), c) , \quad (1.20)$$

при i = 0, MULxPOW(V, i, c) = V.

Регістр зсуву з лінійним зворотнім зв'язком (LFSR) складається зі шістнадцяти компонентів $s_0, \ldots, s_{15}$, кожен з яких містить 32 біти.

Скінченний автомат (FSM) має три 32-бітних регістри R1, R2, R3. S-бокси S_1 та S_2 використовуються для оновлення вмісту регістрів R2 та R3.

Хронометрична операція MUL_α здійснює перетворення 8-бітного значення в 32-бітне за формулою:

$$MUL_\alpha(c) = (MULxPOW(c, 23, 0xA9) \,||\, MULxPOW(c, 245, 0xA9) \,||\, || \, MULxPOW(c, 48, 0xA9) \,||\, MULxPOW(c, 239, 0xA9)) . \quad (1.21)$$

Хронометрична операція DIV_α здійснює перетворення 8-бітного значення в 32-бітне за формулою:

$$DIV_\alpha(c) = (MULxPOW(c, 16, 0xA9) \,||\, MULxPOW(c, 39, 0xA9) \,||\, || \, MULxPOW(c, 6, 0xA9) \,||\, MULxPOW(c, 64, 0xA9)) . \quad (1.22)$$

В режимі ініціалізації LFSR отримує вхідне 32-бітне слово F, яке є вихідним з FSM. Тоді змінна V обчислюється за формулою:

$$V = (s_{0,1} \,||\, s_{0,2} \,||\, s_{0,3} \,||\, 0x00) \oplus MUL_\alpha(s_{0,0}) \oplus s2 \oplus (0x00 \,||\, || \, s_{11,0} \,||\, s_{11,1} \,||\, s_{11,2}) \oplus DIV_\alpha(s_{11,3}) \oplus F . \quad (1.23)$$

В режимі генерування потокового ключа LFSR не отримує жодних вхідних значень. Тоді змінна V обчислюється за формулою:

$$V = (s_{0,1} \| s_{0,2} \| s_{0,3} \| 0x00) \oplus MUL_{\alpha}(s_{0,0}) \oplus s2 \oplus (0x00 \| \| s_{11,0} \| s_{11,1} \| s_{11,2}) \oplus DIV_{\alpha}(s_{11,3}) . \quad (1.24)$$

FSM отримує два вхідних слова s_{15} та s_5 від LFSR. Генерує вихідне слово F:

$$F = (s_{15} \boxplus R1) \oplus R2 . \quad (1.25)$$

SNOW-3G алгоритм здійснює шифрування швидше за своїх попередників SNOW 1.0 та SNOW 2.0, а також є простішим в реалізації.

1.8. Поняття аудиту безпеки і цілі його проведення

Аудит безпеки, спрямований на виявлення подій, що впливають на безпеку системи та забезпечення реагування системи на виявлення вторгнень, а також на забезпечення формування необхідних даних для подальшого відновлення системи в безпечний стан [7]. По суті, аудит безпеки виконує функцію контролю безпеки системи, під яким розуміють збір, накопичення інформації про події, що відбуваються в інформаційній системі та аналіз записів безпеки з метою перевірки ефективності управління системою, забезпечення гарантій відповідності функціонування системи політиці безпеки та вироблення рекомендацій про необхідні зміни в управлінні, політиці та процесах безпеки.

Особливістю аудиту є його сильна залежність від інших послуг та механізмів безпеки. Так ідентифікація та аутентифікація є відправною точкою підзвітності користувачів. Для забезпечення конфіденційності та цілісності реєстраційної інформації застосовують механізми управління доступом.

Аудит представляє собою незалежну експертизу окремих областей функціонування системи. Розрізняють зовнішній і внутрішній аудит. Зовнішній аудит - це, як правило, разовий захід, що проводиться за ініціативою керівництва організації або акціонерів. Внутрішній аудит являє собою безперервну діяльність, яка здійснюється на підставі "Положення по внутрішньому аудиті" та відповідно до плану, підготовка якого здійснюється підрозділом внутрішнього аудиту і затверджується керівництвом організації. Аудит безпеки інформаційних систем є однією зі складових IT аудиту. Цілями проведення аудиту безпеки є:

- аналіз ризиків, пов'язаних з можливістю здійснення загроз безпеки щодо ресурсів ІС;
- оцінка поточного рівня захищеності ІС;
- локалізація вузьких місць у системі захисту ІС;
- оцінка відповідності ІС існуючим стандартам в галузі інформаційної безпеки;
- вироблення рекомендацій щодо впровадження нових та підвищення ефективності існуючих механізмів безпеки ІС.

Принаймні участь у впровадженні тієї ж підсистеми аудиту безпеки, яка змогла б надавати аудитору вихідні дані для аналізу поточної ситуації, він взяти може і повинен. Звичайно, в цьому випадку, аудитор вже не зможе об'єктивно оцінити реалізацію цієї підсистеми і вона природним чином випадає з плану проведення аудиту. Точно також, внутрішній аудитор може взяти діяльну участь у розробці політик безпеки, надавши можливість оцінювати якість цих документів зовнішнім аудиторам.

Роботи з аудиту безпеки інформаційної системи (ІС) включають в себе ряд послідовних етапів, які в цілому відповідають етапам проведення комплексного аудиту IT АС, який включає в себе наступне:

- ініціювання процедури аудиту;
- збір інформації аудиту;
- аналіз даних аудиту;

- вироблення рекомендацій;
- підготовка аудиторського звіту.

Використовувані методи аналізу даних визначаються обраними підходами до проведення аудиту, які можуть відрізнятися.

Перший підхід, найскладніший, базується на аналізі ризиків. Спираючись на методи аналізу ризиків, аудитор визначає для обстежуваної ІС індивідуальний набір вимог безпеки, найбільшою мірою враховує особливості даної ІС, середовища її функціонування й існуючі в даному середовищі загрози безпеки. Даний підхід є найбільш трудомістким і вимагає найвищої кваліфікації аудитора. На якість результатів аудиту, в цьому випадку, сильно впливає використовувана методологія аналізу та управління ризиками та її застосування до даного типу ІС.

Другий підхід, самий практичний, спирається на використання стандартів інформаційної безпеки. Стандарти визначають базовий набір вимог безпеки для широкого класу ІС, який формується в результаті узагальнення світової практики. Стандарти можуть визначати різні набори вимог безпеки, в залежності від рівня захищеності ІС. Від аудитора в даному випадку потрібно правильно визначити набір вимог стандарту, відповідність яким потрібно забезпечити для даної ІС. Необхідна також методика, що дозволяє оцінити цю відповідність. Із-за своєї простоти (стандартний набір вимог для проведення аудиту вже заздалегідь визначений стандартом) та надійності (стандарт - є стандарт і його вимоги ніхто не спробує оскаржити), описаний підхід найбільш розповсюджений на практиці (особливо при проведенні зовнішнього аудиту). Він дозволяє при мінімальних витратах ресурсів робити обґрунтовані висновки про стан ІС.

Третій підхід, найбільш ефективний, припускає комбінування перших двох. Базовий набір вимог безпеки, що пред'являються до ІС, визначається стандартом. Додаткові вимоги, максимальною мірою враховують особливості функціонування даної ІС, формуються на основі аналізу ризиків. Цей підхід є набагато простіше першого, тому що більша частина вимог

безпеки вже визначена стандартом, і, в той же час, він позбавлений недоліку другого підходу, що вимоги стандарту можуть не враховувати специфіки обстежуваної ІС.

1.9. Криптоаналіз як метод проведення аудиту криптосистем

Основне завдання криптоаналізу — розробка та ефективне застосування методів, систем, комплексів, алгоритмів та засобів аналізу криптографічних систем. Криптоаналіз здійснюється при відомих вхідних та вихідних даних, алгоритмах чи засобах криптографічних перетворень, а також, можливо, частина ключової інформації.

Основною метою проведення криптоаналізу є визначення криптографічної стійкості криптографічних перетворень, перш за все щодо неможливості визначення спеціальних (ключових) даних тощо. Криптоаналіз має здійснюватися спершу розробником з метою доведення рівня гарантій криптографічної стійкості, що задекларована розробником та очікується замовником. Крім того, на практиці підтверджено, що в процесі різних протиборств на різних рівнях можуть (як правило, здійснюються) криптоаналітичні атаки. Для визначення степеня небезпечності таких дій можна застосувати різні підходи.

Першими кроками при загальному розгляді методів криптоаналізу є визначення моделі порушника та моделі загроз. Такий підхід ґрунтується на факті, що потенційна вразливість інформаційної системи залежить від матеріально-технічних ресурсів, які можуть бути використані в процесі здійснення криптоаналізу.

В умовах дій порушників у відповідній інформаційній чи інформаційно-телекомунікаційній системах з необхідною якістю криптографічною системою мають надаватися базові послуги (виконуватися функції) або розв'язуватися задачі забезпечення з необхідним рівнем гарантій

конфіденційності цілісності, справжності (автентичності), неспростовності (спостережливості), доступності та надійності.

Конфіденційність інформації — властивість захищеності інформації з наперед заданою якістю (імовірністю) від неавторизованого доступу до неї та спроб розкриття (отримання змісту) неавторизованими користувачами та (або) процесами.

Цілісність інформації — властивість захищеності інформації, яка полягає в тому, що інформація практично не може бути змінена випадково чи навмисне неавторизованими суб'єктами (порушниками) чи об'єктами (процесами), причому факт можливості порушення цілісності може бути визначений з наперед заданою ймовірністю.

Справжність (автентичність) — властивість об'єктів/суб'єктів (в тому числі інформації, ресурсів, повідомлень, даних, користувачів тощо) забезпечити встановлення достовірності твердження, що суб'єкт або об'єкт має заявлені (очікувані) властивості.

Доступність — властивість ресурсу системи (інформації, послуги, об'єкта інформаційної системи), яка полягає в тому, що авторизований користувач або процес, наділений відповідними повноваженнями, може використовувати ресурс згідно з правилами та певної якості, у тому числі за рахунок виконання криптографічних перетворень.

Неспростовність — властивість запобігати можливості заперечення реальними суб'єктами (користувачами) та об'єктами (процесами) фактів повного або часткового взяття участі в інформаційному обміні або інформаційній взаємодії. Як правило включає формування, надання та передавання доказів реальної участі в інформаційному обміні чи інформаційній взаємодії і, в основному, ґрунтується на виконанні криптографічних перетворень з використанням особистих ключів.

Спостережливість — властивість ресурсу системи, що дозволяє реєструвати (фіксувати) роботу та дії користувачів і процесів, використання ресурсу системи, однозначно встановлювати ідентифікатори (імена)

причетних до певних подій користувачів та процесів, а також реагувати на ці події з метою мінімізації можливих втрат у системі, що здійснюється в тому числі за рахунок використання криптографічних перетворень.

Надійність — властивість незмінності певної поведінки та результатів.

1.10. Висновки до розділу

Розглянуто особливості функціонування криптографічних засобів захисту мережі UMTS: механізми взаємної аутентифікації; особливості набору АКА алгоритмів MILENAGE, які використовуються для визначення вектора аутентифікації; функціонування алгоритмів f8 та f9, які забезпечуються конфіденційність та цілісність даних відповідно, і базуються на алгоритмі шифрування KASUMI; проаналізовано алгоритми шифрування KASUMI та SNOW-3G; обґрунтовано використання криптоаналізу як метода аудиту криптосистем. Це дало змогу визначити методи для проведення дослідження захищеності даних криптографічних засобів.

РОЗДІЛ 2
КРИПТОАНАЛІЗ АЛГОРИТМУ KASUMI

2.1. Сендвіч атака

В цьому розділі описуються методи, що використовуються в атаці на KASUMI. Розглядаються базова бумеранг атака, опис нової структури (з пов'язаними ключами), яка називається сендвіч атака (з пов'язаними ключами), що використовує залежність між основними диференціалами, щоб отримати більш точну оцінку ймовірності порівнювача [8]. Також описується варіант атаки на основі адаптивно підібраного відкритого тексту, яка називається – прямокутна сендвіч атака (з пов'язаними ключами). Слід відзначити, що ідея використовувати залежність між диференціалами з метою підвищення бумеранг порівнювача неявно запропонована Вагнером в [9], а також використовується в деяких простих сценаріях в [10, 11]. Таким чином, дана структура може розглядатися – як формальний розгляд та узагальнення ідей, запропонованих в [9, 10, 11].

2.2. Основи бумеранг атаки з пов'язаними ключами

Бумеранг атака з пов'язаними ключами представлена Кімом в [12, 13], і незалежно Біхамом в [14], як комбінація бумеранг атаки в [9] і диференціальної атаки з пов'язаними ключами в [15]. У цій атаці, шифр розглядається як каскад з двох субшифрів $E = E_1 \circ E_0$, і диференціали субшифрів E_0 і E_1 об'єднані в порівнювач адаптивного підібраного тексту і шифротексту для E.

Припускається, що існує диференціал пов'язаних ключів $\alpha \rightarrow \beta$ для E_0 з різницею ключів ΔK_{ab} з ймовірністю p. (наприклад $Pr[E_{0(K)}(P) \oplus E_{0(K \oplus Kab)}(P \oplus \alpha) = \beta] = p$, де $E_{0(K)}$ визначає шифрування E_0 з ключем K). Також припускається, що існує диференціал пов'язаних ключів $\gamma \rightarrow \delta$ для

E_1 з різницею ключів ΔK_{ac} з ймовірністю q. Бумеранг порівнювач з пов'язаними ключами потребує шифрування/дешифрування з секретним ключем K_a і під зв'язні ключі $K_b = K_a \oplus \Delta K_{ab}$, $K_c = K_a \oplus \Delta K_{ac}$, і $K_d = K_c \oplus \Delta K_{ab} = K_b \oplus \Delta K_{ac}$.

Алгоритм порівнювача є дуже простий:

1. Вибрати M відкритих текстів навмання, ініціалізувати лічильник C=0. Для кожного відкритого тексту P_a, виконати наступне:

а) Здійснити шифрування $C_a = E_{Ka}(P_a)$ і $C_b = E_{Kb}(P_b)$ де $P_b = P_a \oplus \alpha$.

б) Здійснити дешифрування $P_c = E_{Kc}(C_c)$ і $P_d = E_{Kd}(C_d)$ де $C_c = C_a \oplus \delta$ і $C_d = C_b \oplus \delta$.

в) Якщо $P_c \oplus P_d = \alpha$, збільшити лічильник C на 1.

2. Якщо C > «порогове значення», вивести «E». В іншому випадку, вивести «Випадкова перестановка».

Квартет сконструйований бумеранг атакою зображений в лівій частині рис. 2.1.

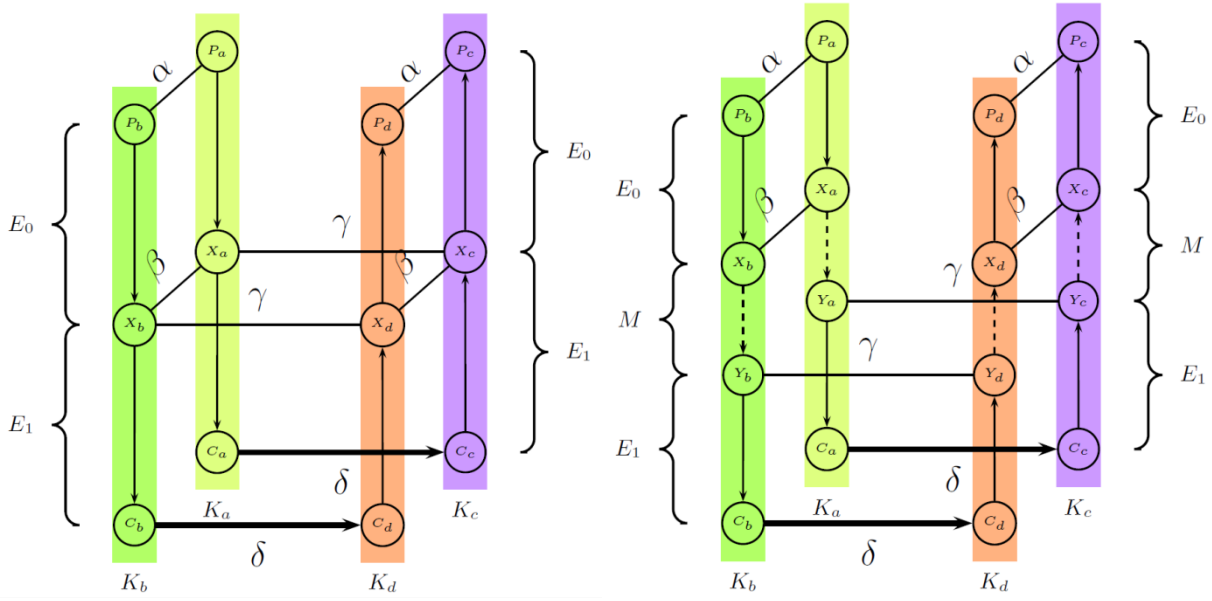

Рис. 2.1. Бумеранг квартет з пов'язаними ключами і сендвіч квартет з пов'язаними ключами

Для випадкової перестановки ймовірність, що остання умова буде задоволена – 2^{-n}, де n – розмір блоку. Для E, ймовірність що пара (P_a, P_b) є правильною по відношенню до першого диференціалу (наприклад, ймовірність, що проміжна різниця після E_0 рівна β, як передбачає диференціал) є p. Незалежно припускаючи, ймовірність, що обидві пари (C_a, C_c) і (C_b, C_d) є правильними парами по відношенню до другого диференціалу є q^2. Якщо ці всі пари правильні, тоді $E_1^{-1}(C_c) \oplus E_1^{-1}(C_d) = \beta = E_0(P_c) \oplus E_0(P_d)$. Таким чином, з ймовірністю p, $P_c \oplus P_d = \alpha$. Отже, загальна ймовірність цього квартету відкритих текстів і шифротекстів забезпечити умову $P_c \oplus P_d = \alpha$ є близька до $(pq)^2$. Проте, якщо $pq \gg 2^{-n/2}$, алгоритм дозволяє характеризувати E з випадковою перестановкою даних $O((pq)^{-2})$ адаптивно підібраних відкритих текстів та шифротекстів.

Порівнювач може бути вдосконалений, шляхом розгляду декількох диференціалів форми $\alpha \rightarrow \beta'$ і $\gamma' \rightarrow \delta$ (для тих же ж α і δ). Дане покращення детально описано в [10], але воно не використовується в кінцевому варіанті атаки.

Звичайний спосіб використання бумеранг порівнювача з пов’язаними ключами у атаці по відновленню ключа – додавання раунду перед порівнювачем, в результаті одержання частини ключа в першому раунді.

В випадку з KASUMI використовується подвійна техніка додавання раунду після порівнювача, застосовуючи атаку з адаптивно підібраним шифротекстом/відкритим текстом, і отримується матеріал ключа в останньому раунді.

2.3. Сендвіч атака з пов’язаними ключами

В цьому розділі здійснено поділ шифру на три субшифри, $E=E_1+M+E_0$. Припущення такі ж як і в базовій атаці: нехай існує диференціал пов’язаних ключів $\alpha \rightarrow \beta$ для E_0 з різницею ключів ΔK_{ab} з ймовірністю p, і диференціал

пов'язаних ключів $\gamma \to \delta$ для E_1 з різницею ΔK_{ac} з ймовірністю q. Алгоритм атаки є таким же ж як і в базовій атаці (ігнорування центрального субшифру M). Проте, аналіз є більш детальніший і потребує великої уваги в аналізі залежності між різними розподілами.

Головна ідея сендвіч атаки – перехід середини. В базовій бумеранг атаці, якщо пара (P_a, P_b) є правильною парою по відношенню до першого диференціалу, і обидві пари ($C_в$, C_c) і (C_b, C_d) є правильними парами по відношенню до другого диференціалу, тоді отримуємо:

$$(X_a \oplus X_b = \beta) \wedge (X_a \oplus X_c = \gamma) \wedge (X_b \oplus X_d = \gamma) , \qquad (2.1)$$

де X_i – проміжне значення шифрування P_i.

Тоді

$$X_c \oplus X_d = (X_c \oplus X_a) \oplus (X_a \oplus X_b) \oplus (X_b \oplus X_d) = \beta \oplus \gamma \oplus \gamma = \beta , \qquad (2.2)$$

в результаті $P_c \oplus P_d = \alpha$ з ймовірністю p.

У випадку сендвіч атаки, замість (1.26) отримуємо:

$$(X_a \oplus X_b = \beta) \wedge (Y_a \oplus Y_c = \gamma) \wedge (Y_b \oplus Y_d = \gamma) . \qquad (2.3)$$

Проте, ймовірність трьохрівневого бумеранг порівнювача з пов'язаними ключами – p^2q^2r, де

$$r = Pr\,[(Xc \oplus Xd = \beta)|(Xa \oplus Xb = \beta) \wedge (Ya \oplus Yc = \gamma) \wedge (Yb \oplus Yd = \gamma)] . \qquad (2.4)$$

Без подальших припущень про M, r буде дуже малим (близько 2^{-n}), і таким чином порівнювач не принесе очікуваного результату. Проте, як сказано в [9, 10, 11], в деяких випадках відмінність між E_0 і E_1 може бути

обрана такою, що ймовірність r переходу через центральний субшифр (хоча б в одному напрямку) є 1, яка є набагато більшою ніж очікувалось.

Приклад цього феномену представлений в [9] і описаний в [11] під ім'ям «Заміна Фейстеля» («Feistel switch»). Нехай E шифр з мережею Фейстеля, представлений як $E = E_1 \circ M \circ E_0$, де M складається з одного раунду мережі Фейстеля, що зображено на рисунку 2.2. Припускається, що диференціали $\alpha \to \beta$ (для E_0) і $\gamma \to \delta$ (для E_1) не мають відмінності ключів (тобто $\Delta K_{ab} = \Delta K_{ac} = 0$), і задовольняють $\beta^R = \gamma^L$ (тобто права частина β еквівалентна лівій частині γ).

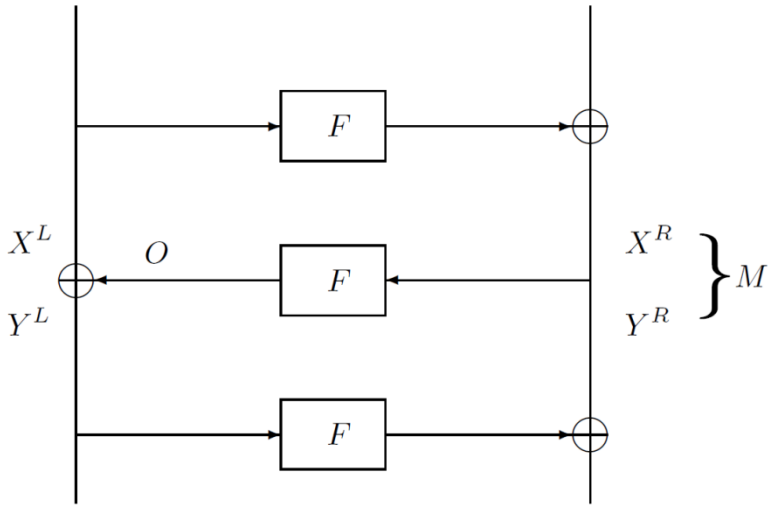

Рис. 2.2. Конструкція Фейстеля

Припустимо, що вираз (2.3) виконується. В цьому випадку, згідно конструкції Фейстеля $X_i^R = Y_i^L$ для всіх i, тобто отримуємо:

$$X_a^R \oplus X_b^R = \beta^R = \gamma^L = X_a^R \oplus X_c^R = X_b^R \oplus X_d^R, \qquad (2.5)$$

тоді

$$(X_a^R = X_d^R) \text{ і } (X_b^R \oplus X_c^R) . \qquad (2.6)$$

Тому, вихідні значення F-функції в раунді мережі Фейстеля представленим M, позначені (O_a, O_b, O_c, O_d), і задовольняють ($O_a = O_b$) і ($O_c = O_d$).

Згідно конструкції Фейстеля, $X_i^L = Y_i^R \oplus O_i$ і виразу (3), $Y_a \oplus Y_b \oplus Y_c \oplus Y_d = 0$ випливає:

$$X_a \oplus X_b \oplus X_c \oplus X_d = 0 , \qquad (2.7)$$

що завдяки виразу (2.3) визначає $X_c \oplus X_d = \beta$. Таким чином, в цьому випадку отримуємо:

$$r = Pr((X_c \oplus X_d = \beta) \mid (X_a \oplus X_b = \beta) \wedge (Y_a \oplus Y_c = \gamma) \wedge (Y_b \oplus Y_d = \gamma)) = 1, \qquad (2.8)$$

незалежно від вибору F-функції.

Інші приклади цього ж феномену описані в [10] (під ім’ям «середньораундовий S-box трюк»), і в [11] (під ім’ям «заміна кроку» і «заміна S-box»).

Дана атака на KASUMI є першим не тривіальним прикладом цього феномену, де детальний аналіз показує, що r є меншим ніж 1, але набагато більшим ніж очікувана величина при стандартних незалежних припущеннях. Шифр E (7 раундовий KASUMI) є конструкцією Фейстеля, M складається з одного раунду, і $\beta = \gamma$. Проте, аргументи представлені вище не можуть бути застосовані безпосередньо тому, що є не нульова різниця ключів в M, отже нульова вхідна різниця F-функції не визначає нульової вихідної різниці. Тому аналізується F-функція повністю, і в цьому випадку $r = 2^{-6}$ (на відміну від 2^{-32}, що є очікуваним значенням для випадкового раунду Фейстеля в 64-бітному шифрі).

2.4. Прямокутна сендвіч атака

Перетворення бумеранг порівнювача (з пов'язаними ключами) в прямокутну атаку на основі адаптивно підібраного відкритого тексту ґрунтується на породжених-парадоксальних аргументах. Поділ на субшифри і припущення такі ж як і в бумеранг порівнювані (з пов'язаних ключами). Ключова ідея перетворення – шифрування великої кількості відкритих текстів з вхідною відмінністю α, і моніторинг квартетів (пар, що складаються з пар), що утворяться у відповідності до вимог бумеранг процесу. Іншими словами, вважаємо квартети відкритих текстів типу ((P_a, $P_b = P_a \oplus \alpha$), (P_c, $P_d = P_c \oplus \alpha$)) зашифровані з пов'язаними ключами K_a,K_b,K_c і K_d відповідно, і ці квартети називаються «правильні квартети», якщо задоволені наступні умови:

1. $E_{0(Ka)}(P_a) \oplus E_{0(Kb)}(P_b) = \beta = E_{0(Kc)}(P_c) \oplus E_{0(Kd)}(P_d)$.

2. $E_{0(Ka)}(P_a) \oplus E_{0(Ka)}(P_c) = \gamma$ (це приводить до $E_{0(Kb)}(P_b) \oplus E_{0(Kd)}(P_d) = \gamma$, якщо ця умова виконана з попередньою).

3. $C_a \oplus C_c = \delta = C_b \oplus C_d$.

Ймовірність квартету бути правильним квартетом є нижньою межею ймовірності події:

$$C_a \oplus C_c = \delta = C_b \oplus C_d\,. \qquad (2.9)$$

Звичайне припущення – кожна з попередніх умов є незалежною від інших, і отже ймовірність того, що даний квартет ((P_1, P_2), (P_3, P_4)) є правильним квартетом – $p^2 \cdot 2^{-n} \cdot q^2$. Звідси для випадкової перестановки, ймовірність умови (1.27) є 2^{-2n}, даний процес може бути використаний для порівняння E та випадкової перестановки, якщо $pq \gg 2^{-n/2}$ (таке ж значення як і в стандартному бумеранг порівнювачеві).

Проте, складність даних порівнювача є $O(2^{n/2}(pq)^{-1})$, яка є набагато

більшою ніж складність бумеранг порівнювача. Більша складність даних випливає з факту, що подія $E_{0(Ka)}(P_a) \oplus E_{0(Kc)}(P_c) = \gamma$ відбувається з випадковою ймовірністю 2^{-n} (насправді, це породжений-парадоксальний аргумент поза конструкцією). Ідентифікація правильних квартетів є більш складною, ніж у випадку бумеранг атаки, так як замість перевірки умови на парах, потрібно перебрати всі можливі квартети. В цей же ж час, природа адаптивно підібраного відкритого тексту дозволяє використати потужніші техніки відновлення ключа.

Перетворення сендвіч каркасу – в сендвіч каркас прямокутного типу здійснюється таким же ж способом. Алгоритм порівнювача залишається сталим. І ймовірність квартету бути правильним квартетом є $p^2 * 2^{-n} * r' * q^2$, де

$$r' = Pr\ ((Y_b \oplus Y_d = \gamma) | (X_a \oplus X_b = \beta) \wedge (X_c \oplus X_d = \beta) \wedge (Y_a \oplus Y_c = \gamma)). \quad (2.10)$$

Випливає з міркувань симетрії, що у випадку, де E – шифр на основі мережі Фейстеля, M складається з одного раунду, і $\beta^R = \gamma^L$, отримуємо $r' = r$. В атаці на KASUMI, можливо використати обчислення r в сендвіч каркасі, щоб знайти ймовірність прямокутного сендвіч порівнювача з пов'язаними ключами у випадку 3-рівневої сендвіч структури.

2.5. Сендвіч порівнювач з пов'язаними ключами для 7-раундового KASUMI

В даному порівнювачі 7 раундів, шифр KASUMI позиціонуються як каскад $E = E_1 \circ M \circ E_0$, де E_0 складається з 1-3 раундів, M – з 4 раунду, E_1 – з 5-7 раундів. Диференціал пов'язаних ключів, який використовується для E_0, є незначною модифікацією диференціальної характеристики представленої в [16], в якій

$$\alpha = (0_x, 0010\ 0000_x) \rightarrow (0_x, 0010\ 0000_x) = \beta. \qquad (2.11)$$

Відповідна різниця ключів K_{ab} = (0, 0, 8000x, 0, 0, 0, 0, 0), тобто лише третє ключове слово має одиничну різницю бітів $K_3 = 8000_x$. Цей диференціал пов'язаних ключів зображено на рис. 2.3. Диференціал пов'язаних ключів, який використовується для E_1, є таким же ж диференціалом, але зміщеним на 4 раунди, в якому різниця даних є такою ж, проте ключова різниця K_{ac} = (0, 0, 0, 0, 0, 0, 8000_x, 0).

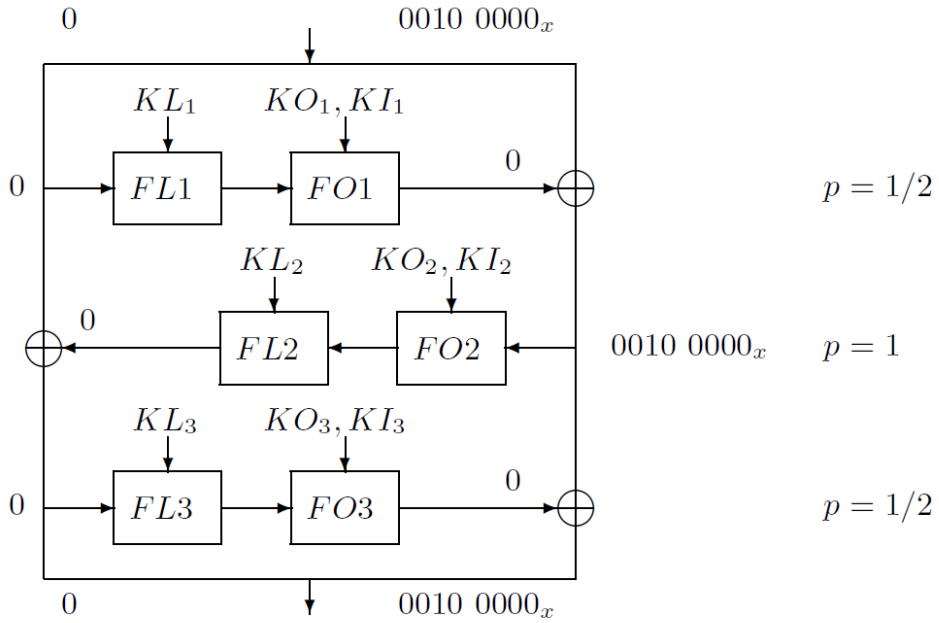

Рис. 2.3. 3-раундова диференціальна характеристика з пов'язаними ключами

Як показано в [16], ймовірність кожної з цих 3-раундових характеристик складає 0,25. Для того, щоб знайти ймовірність сендвіч порівнювача з пов'язаними ключами, потрібно обчислити ймовірність:

$$\Pr\left((X_c \oplus X_d = \beta \mid (X_a \oplus X_b = \beta) \wedge (Y_a \oplus Y_c = \gamma) \wedge (Y_b \oplus Y_d = \gamma)\right), \qquad (2.12)$$

де (X_a,X_b,X_c,X_d) і (Y_a, Y_b, Y_c, Y_d) є проміжними значеннями перед і після середнього рівня сендвіч структури а під час шифрування/дешифрування квартету (P_a, P_b, P_c, P_d).

Розглянемо квартет (P_a, P_b, P_c, P_d), для якого виконується наступна умова:

$$X_a \oplus X_b = \beta) \wedge (Y_a \oplus Y_c = \gamma) \wedge (Y_b \oplus Y_d = \gamma) . \quad (2.13)$$

Оскільки M складається з одного раунду мережі Фейстеля, це означає що

$$(X_a^R = X_d^R) \wedge (X_b^R = X_c^R) , \quad (2.14)$$

де X_i^R визначає праву частину X_i, що подається на вхід функції FO4.

Більше того, якщо права частина різниці $\beta = \gamma$ рівна нулю, тоді

$$X_a^{RR} = X_b^{RR} = X_c^{RR} = X_d^{RR} , \quad (2.15)$$

де X_i^{RR} визначає праву частину (тобто 16 правих біт) X_i^{RR}.

На рис. 2.4 показане утворення різниць в FO_4 і $FI_{4,3}$. Функція FO4 – 3 раундова мережа Фейстеля, чиї 32-бітні значення після і-го раунду визначаються (X_a^j, $X_b^{j,}$, $X_c^{j,}$, $X_d^{j,}$), і функція FI є 4-раундовою мережею Фейстеля, чиї 16-бітні значення після раунду і визначаються (I_a^j, I_b^j, I_c^j, I_d^j).

Значення позначені однаковим символом є рівними. Значення позначені B є балансованими (результат операції XOR всіх чотирьох значень рівний нулю). Значення в $FI_{4,3}$, які є або сірими, або чорними мають одне з двох можливих значень.

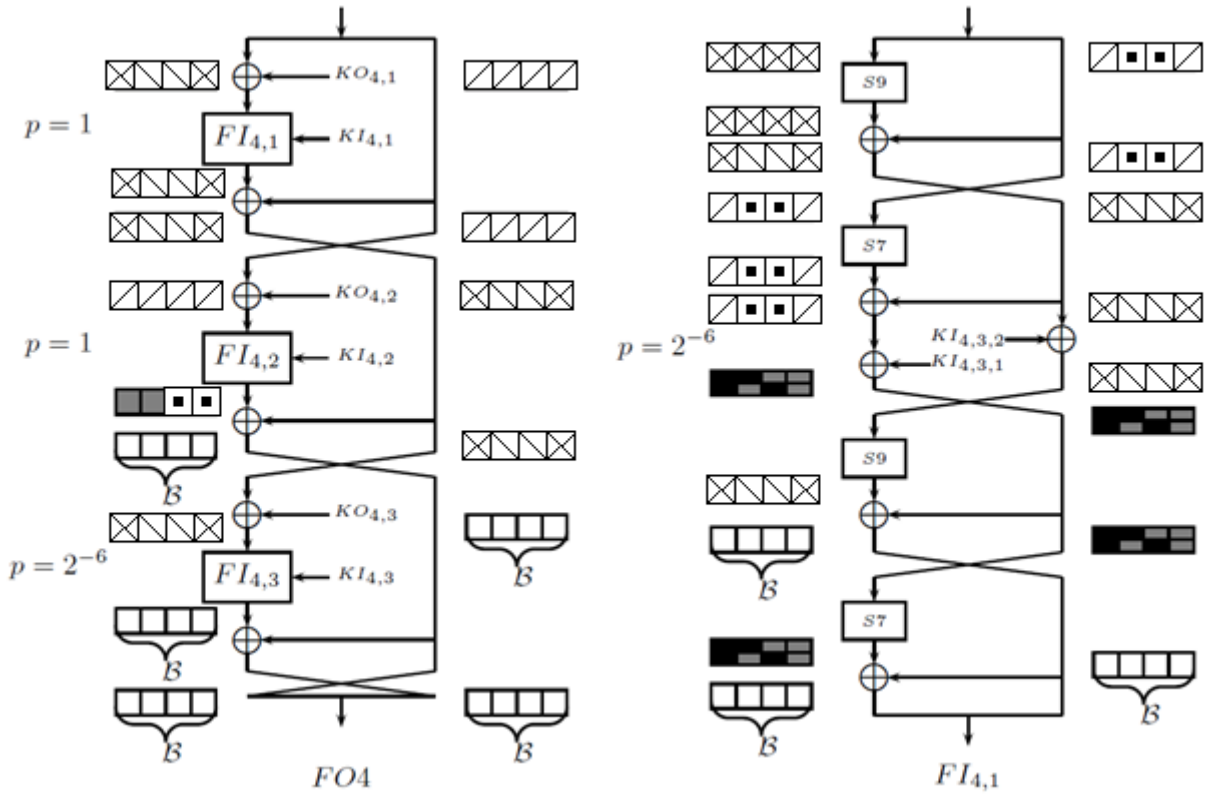

Рис. 2.4. Різниці в FO_4 і $FI_{4,3}$

Зазначимо, що різниці ключів ΔK_{ab} і ΔK_{ac} впливають в раунді 4 на субключі $KI_{4,3}$ і $KI_{4,2}$ відповідно, і зокрема, немає різниці ключів в першому раунді FO4. В результаті, з рівняння (2.14) випливає:

$$(X_a^1 = X_d^1) \wedge (X_a^1 = X_d^1) . \tag{2.16}$$

Крім того, немає ніякої різниці ключів в парах, що відповідають (P_a, P_b) і (P_c, P_d) в другому раунді FO4. З рівняння (2.15) випливає:

$$(I_a^2 = I_b^2) \wedge (I_c^2 = I_d^2) . \tag{2.17}$$

Об’єднуючи рівняння (2.16) і (2.17), отримаємо наступне відношення в правій частині проміжного значення після раунду 3 функції FO4:

$$X_a^{3R} \oplus X_b^{3R} \oplus X_c^{3R} \oplus X_d^{3R} = 0 . \quad (2.18)$$

В F-функції 3-го раунду FO4 розглянемо пари (P_a, P_d) і (P_b, P_c). Оскільки різниця ключів в цих парах (еквівалентна $K_{ab} \oplus K_{ac}$) впливає лише на субключ $KI_{4,3,1}$, з рівності (2.16) випливає:

$$I_a^{3R} \oplus I_b^{3R} \oplus I_c^{3R} \oplus I_d^{3R} = 0 . \quad (2.19)$$

В лівій частині результату, XOR чотирьох значень не є обов'язково рівним 0, згідно різниці субключів, яка впливає на вхідні другого S7 в $FI_{4,3}$. Проте, якщо ці 7 вхідних біт, позначити як (J_a, J_b, J_c, J_d), виконується одна з умов:

$$((J_a = J_b) \wedge (J_c = J_d)) \text{ або } ((J_a = J_c) \wedge (J_b = J_d)) . \quad (2.20)$$

Тоді з рівності (2.19) випливає:

$$I_a^{3L} \oplus I_b^{3L} \oplus I_c^{3L} \oplus I_d^{3L} = 0 . \quad (2.21)$$

Оскільки отримали вираз $J_a \oplus J_d = J_b \oplus J_c$ (обидва еквівалентні різниці субключів в $KI_{4,3,1}$), кожна з умов виразу (2.20) буде виконуватись з ймовірністю 2^{-7}. Проте, об'єднуючи рівності (2.18), (2.19) і (2.21), отримаємо що наступна умова буде виконуватись з ймовірністю 2^{-6}:

$$X_a^3 \oplus X_b^3 \oplus X_c^3 \oplus X_d^3 = 0 . \quad (2.22)$$

Оскільки функція FL є лінійною для даного ключа і немає різниці ключів в FL4, можна зробити висновок, що коли рівність (2.22) справджується, вихідні значення F функції в 4 раунді (позначені як (O_a^4, O_b^4, O_c^4, O_d^4)) задовольняють з ймовірністю 2^{-6} умову:

$$O_a^4 \oplus O_b^4 \oplus O_c^4 \oplus O_d^4 = 0 . \quad (2.23)$$

З виразу (2.23) випливає:

$$Y_a^L \oplus Y_b^L \oplus Y_c^L \oplus Y_d^L = 0 . \quad (2.24)$$

З цього слідує, що

$$X_a^L \oplus X_b^L \oplus X_c^L \oplus X_d^L = 0, \quad (2.25)$$

що виконується з ймовірністю 2^{-6}. Об'єднавши це з рівністю (2.24) отримаємо:

$$\Pr((X_c \oplus X_d = \beta) | (X_a \oplus X_b = \beta) \wedge (Y_a \oplus Y_c = \gamma) \wedge (Y_b \oplus Y_d = \gamma)) = 2^{-6}. \quad (2.26)$$

Отже, загальна ймовірність сендвіч порівнювача з пов'язаними ключами складає $(1/4)^2 * (1/4)^2 * 2^{-6} = 2^{-14}$, яка є набагато більшою за ймовірність $(1/4)^2 * (1/4)^2 * 2^{-32} = 2^{-40}$, при стандартному аналізі сендвіч структури.

2.6. Зміна планування ключів

В цьому розділі представлено приклад, в якому зроблено малу зміну в плануванні ключів KASUMI, яка не має ніякого ефекту на диференціальні ймовірнісні характеристики кожного з трьох субшифрів. Проте, для прикладу, ймовірність порівнювача рівна нулю.

Єдина зміна зроблена в KASUMI – в першому субключі, ролі ключових слів K_5' і K_8' є взаємозамінними. Решта планування ключів є модифікованою

згідно правила оригінального KASUMI, тобто ключові слова замінені циклічно в кожному раунді ($KI_{2,1} = K_1'$, $KI_{3,3} = K_7'$).

Оскільки заміна впливає лише на субключі, що використовуються в $KI_{i,1}$ і $KI_{i,3}$ в кожному раунді, диференціали, що використовуються в порівнювачі KASUMI, залишаються такими ж для нового варіанту (з такими ж вхідною/вихідною різницями, такою ж різницею ключів і такими ж ймовірностями). Проте, покажемо, що в цьому випадку:

$$r = Pr((X_c \oplus X_d = \beta)(X_a \oplus X_b = \beta) \wedge (Y_a \oplus Y_c = \gamma) \wedge (Y_b \oplus Y_d = \gamma)) = 0, \quad (2.27)$$

і ймовірність порівнювача рівна нулю. У всіх обчисленнях нижче, позначення є такими ж як і в оригінальному порівнювачі вище. Неможливий перехід зображений на рис. 2.5. Значення позначені однаковим символом рівні. Значення позначені -B є незбалансованими (результат операції XOR всіх чотирьох значень не є нульовим).

Якщо диференціали є рівними як і в початковому порівнювачі, отримаємо:

$$(X_a^R = X_d^R) \wedge (X_b^R = X_c^R), \quad (2.28)$$

$$X_a^{RR} = X_b^{RR} = X_c^{RR} = X_d^{RR}. \quad (2.29)$$

Також, якщо другий раунд FO4 є незмінним, отримаємо:

$$(I_a^2 = I_b^2) \wedge (I_c^2 = I_d^2). \quad (2.30)$$

Проте:

$$X_a^{3R} \oplus X_b^{3R} \oplus X_c^{3R} \oplus X_d^{3R} = I_a^1 \oplus I_b^1 \oplus I_c^1 \oplus I_d^1. \quad (2.31)$$

В першому раунді FO4 отримали різницю між модифікованим варіантом і вихідним KASUMI, оскільки в модифікованому варіанті є різниця субключів по відношенню до пар (P_a, P_b) і (P_c, P_d) в MSB (старший біт) субключа $KI_{4,1,1}$. Проаналізуємо функцію $FI_{4,1}$.

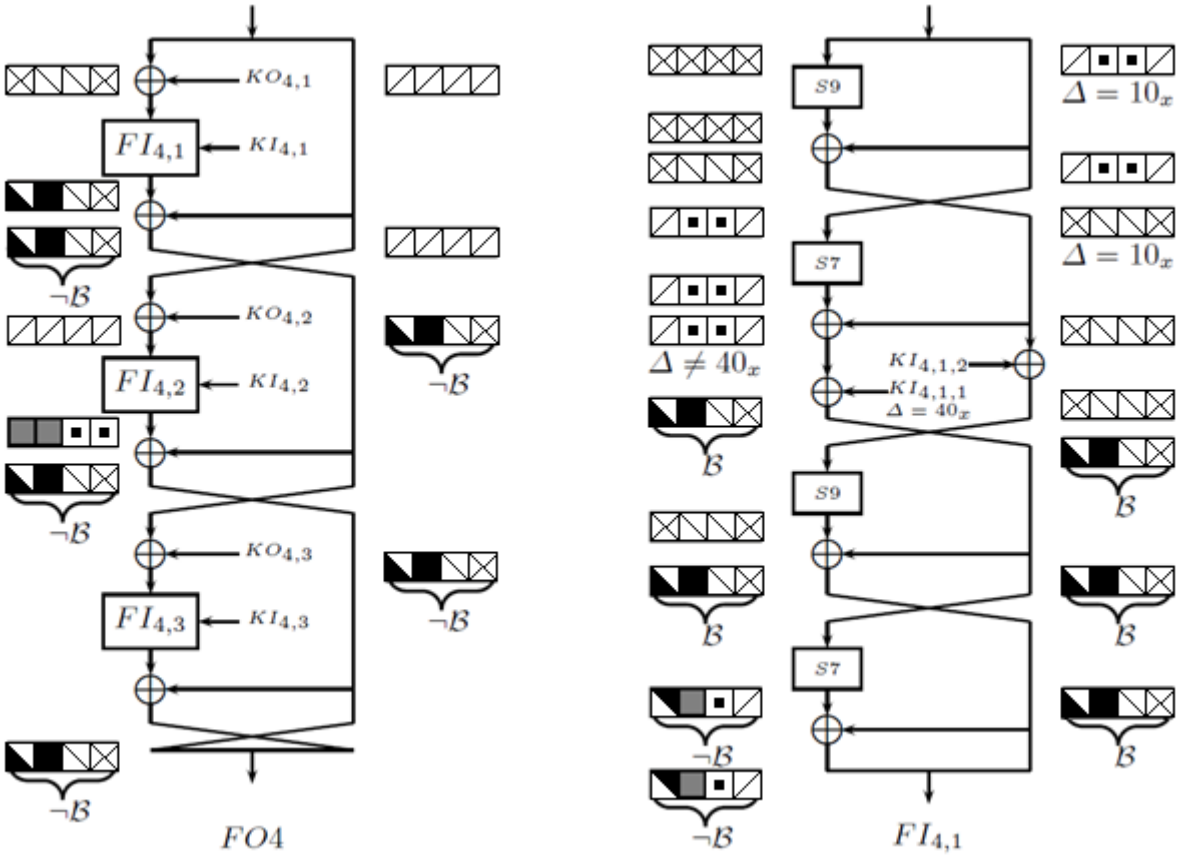

Рис. 2.5. Різниці в FO_4 і $FI_{4,1}$ в модифікованому KASUMI

Згідно структури диференціалу, його вхідні значення мають форму:

$$(X_a^R \oplus KO_{4,1}, X_b^R \oplus KO_{4,1}, X_c^R \oplus KO_{4,1}, X_c^R \oplus KO_{4,1}) = (t, t \oplus 0010_x, t \oplus 0010_x, t)\ . \quad (2.32)$$

Після першого раунду $FI_{4,1}$ значення зберігаються у формі (t’, t’ $\oplus$ 0010_x, t’ $\oplus$ 0010_x, t’). Після другого раунду значення приймають форму u, v, v, u. Стверджуємо, що u $\oplus$ v $\neq$ MSB. Справді, якщо u $\oplus$ v = MSB, тоді вихідних 7 біт S7 мають бути у формі (u’, u’ $\oplus$ 1000000_2 $\oplus$ 0010000_2, u’ $\oplus$

$1000000_2 \oplus 0010000_2$, u'). Проте диференціал ($0010000_2 \oplus 1010000_2$) є неможливим для S7, отже ця подія не може відбутися.

Можна стверджувати, що чотири 7-бітних значення після операції XOR з субключем $KI_{4,1,1}$ є різними. Дійсно, форма цих значень є (u ⊕ k, v ⊕ k ⊕ MSB, v ⊕ k, u ⊕ k ⊕ MSB), ці значення всі різні, оскільки u ≠ v і u ≠ v ⊕ MSB.

Розглянемо S-box S7 в 4 раунді $FI_{4,1}$, його вхідні значення є різними, і можуть бути розділеними на дві пари (u ⊕ k, v ⊕ k ⊕ MSB) і (v ⊕ k, u ⊕ k ⊕ MSB) з однаковою різницею. Оскільки S7 є майже досконалою нелінійною заміною, це значить, що дві відповідні пари вихідних даних мають відмінну різницю, отже, операція XOR цих чотирьох вихідних значень є обов'язково не нульовою. Оскільки результат операції XOR вихідних значень в правій половині є нульовим, отримуємо:

$$I_a^1 \oplus I_b^1 \oplus I_c^1 \oplus I_d^1 \neq 0 . \qquad (2.33)$$

Отже

$$X_a^{3R} \oplus X_b^{3R} \oplus X_c^{3R} \oplus X_d^{3R} \neq 0 . \qquad (2.34)$$

Отже, результат операції XOR чотирьох вихідних значень FO4 є не нульовим з ймовірністю 1, і тому FL є лінійною, це призводить до того, що результат операції XOR чотирьох вихідних значень FL є не нульовим з ймовірністю 1. Це підтверджує наступне:

$$\Pr((X_c \oplus X_d = \beta) \mid (X_a \oplus X_b = \beta) \wedge (Y_a \oplus Y_c = \gamma) \wedge (Y_b \oplus Y_d = \gamma)) = 0 , \qquad (2.35)$$

і тому порівнювач буде невдалим в цьому варіанті KASUMI, як зазначалося.

2.7. Сендвіч атака з пов'язаними ключами на повний KASUMI

Атака на повний KASUMI зображена на рис. 2.6 і застосовує порівнювач представлений в підрозділі 2.5 для 1-7 раунду, і отримує частину ключа в раунді 8.

Нехай $\Delta K_{ab} = (0, 0, 8000_x, 0, 0, 0, 0, 0)$ і $\Delta K_{ac} = (0, 0, 0, 0, 0, 0, 8000_x, 0)$, і нехай K_a, $K_b = K_a \oplus \Delta K_{ab}$, $K_c = K_a \oplus \Delta K_{ac}$, $K_d = K_c \oplus \Delta K_{ab}$ – невідомі пов'язані ключі, які необхідно отримати.

Алгоритм атаки полягає в наступному:

1. Фаза збору даних:

а) Обрати структуру з 2^{24} шифротекстів у формі $C_a = (X_a, A)$, де A є фіксоване і X_a передбачає 2^{24} довільних різних значень. Здійснити дешифрування всіх шифротекстів з ключем K_a і зберегти відкриті тексти у відповідності $C_a - P_a$. Для кожного P_a здійснити шифрування $P_b = P_a \oplus (0_x, 00100000_x)$ з ключем K_b і зберегти результуючі шифротексти C_b. Зберегти пари (C_a, C_b) в хеш-таблиці, індексованій 32-бітним значенням C_b^R (права частина C_b).

б) Обрати структуру з 2^{24} шифротекстів у формі $C_c = (Y_c, A \oplus 00100000_x)$, де A – константа, що використана вище, і Y_c передбачає 2^{24} довільних різних значень. Здійснити дешифрування шифротекстів з ключем K_c і зберегти відкриті тексти у відповідності $C_c - P_c$. Для кожного P_c здійснити шифрування $P_d = P_c \oplus (0_x, 00100000_x)$ з ключем K_d і зберегти результуючі шифротексти C_d. Тоді, здійснити доступ до запису хеш-таблиці відповідно до значення $C_d^R \oplus (0_x, 00100000_x)$ для кожної пари (C_a, C_b), знайденої в цьому записі; застосувати 2 пункт для квартету (C_a, C_b, C_c, C_d).

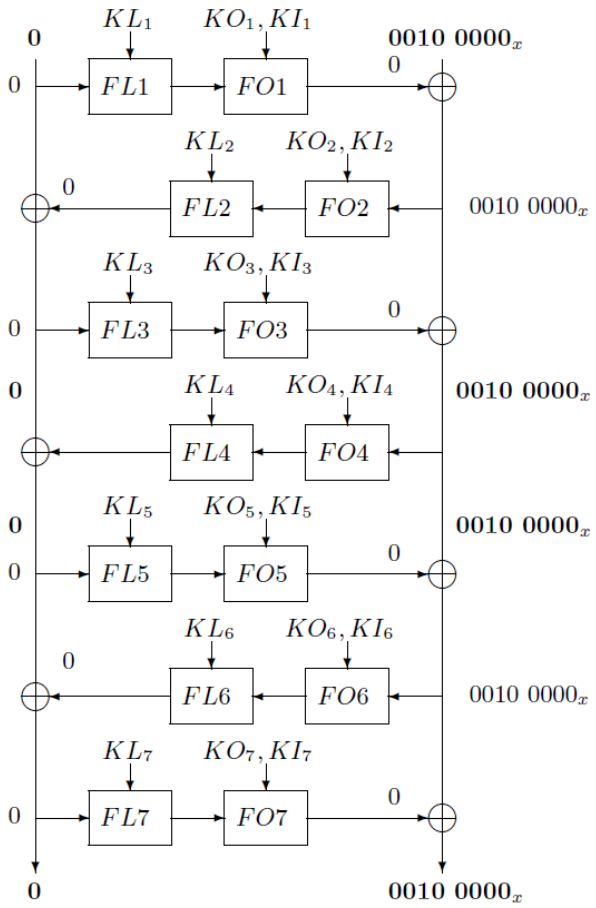

Рис. 2.6. Семираундовий сендвіч порівнювач з пов’язаними ключами

В першому пункті, що описаний вище, для $(2^{24})^2 = 2^{48}$ можливих квартетів здійснено відбір згідно умови, відомих 32 біт різниці (згідно вихідної різниці δ порівнювача), результат – близько 2^{16} квартетів з необхідною різницею.

В пункті 2 можна миттєво ідентифікувати правильні квартети, використовуючи властивість структури KASUMI. Зазначимо, що пара (C_a, C_c) може бути правильним квартетом тоді і тільки тоді, якщо

$$C_a^L \oplus FL8(FO8(C_a^R)) = C_c^L \oplus FL8(FO8(C_c^R)) , \qquad (2.36)$$

згідно структури Фейстеля, це єдиний випадок, коли різниця після раунду 7 є вихідною різницею сендвіч порівнювача (тобто $\delta = (0_x, 00100000_x)$). Проте

значення C_a^R і C_c^R є сталими для всіх шифротекстів, що розглядаються, отже рівність (2.36) дає наступне:

$$C_a^L \oplus C_c^L = FL8(FO8(A)) \oplus FL8(FO8(A \oplus (0_x, 00100000_x))) = \text{const} . \quad (2.37)$$

Тому значення $C_a^L \oplus C_c^L$ є однаковим для всіх правильних квартетів. Це дозволяє виконати наступну вибірку.

2. Ідентифікація правильних квартетів:

а) Записати приблизно 2^{16} запам'ятовуваних квартетів (C_a, C_b, C_c, C_d) в хеш-таблицю індексовану 32-бітним значенням $C_a^L \oplus C_c^L$, і застосувати пункт 3 тільки для контейнерів, що містять хоча б три квартети.

Оскільки ймовірність потрійного збігу в списку з 2^{16} випадкових 32-бітних значень є меншою ніж 2^{-18}, тоді з дуже великою ймовірністю після фільтрування залишаться лише правильні квартети.

В наступному кроці розглядаються всі квартети, що залишилися, як правильні квартети. Завдяки цьому припущенню, знаємо не тільки поточні вхідні дані в 8 раунд, але також різниці у вихідних даних раунду 8.

3. Аналіз правильних квартетів:

а) Для кожного квартету (C_a, C_b, C_c, C_d), припустити 32-бітне значення $KO_{8,1}$ і $KI_{8,1}$. Для двох пар (C_a, C_c) і (C_b, C_d) використати значення припущеного ключа щоб обчислити вхідні і вихідні різниці операції OR в останньому раунді обох пар. Для кожного біта цієї 16-бітної операції OR FL8, можливі значення відповідних бітів $KL_{8,2}$ показані на рисунку 1.14.

З ймовірністю $(8/16)^{16} = 2^{-16}$ значення $KL_{8,2}$ пропонується кожним квартетом і припущеним значенням $KO_{8,1}$ $KI_{8,1}$. Оскільки всі правильні квартети пропонують однаковий ключ, всі неправильні ключі відкидаються з величезною ймовірністю, і отримується коректне значення $KO_{8,1}$, $KI_{8,1}$, $KL_{8,2}$.

б) Припустити 32-бітне значення $KO_{8,3}$ і $KI_{8,3}$, і обчислити вхідні і вихідні різниці операції AND в обох парах кожного квартету. Для кожного біта з 16-бітної операції AND FL8, можливі значення відповідних бітів $KL_{8,1}$

показані на рис. 2.7. З ймовірністю $(8/16)^{16} = 2^{-16}$ значення $KL_{8,1}$ пропонується кожним квартетом і припущеним значенням $KO_{8,3}$ $KI_{8,3}$. Отримується коректне значення $KO_{8,3}$, $KI_{8,3}$, $KL_{8,1}$.

OR — $KL_{8,2}$

(X'_{ac}, Y'_{ac})	(X'_{bd}, Y'_{bd}) (0,0)	(0,1)	(1,0)	(1,1)
(0,0)	{0,1}	—	1	0
(0,1)	—	—	—	—
(1,0)	1	—	1	—
(1,1)	0	—	—	0

AND — $KL_{8,1}$

(X'_{ac}, Y'_{ac})	(X'_{bd}, Y'_{bd}) (0,0)	(0,1)	(1,0)	(1,1)
(0,0)	{0,1}	—	0	1
(0,1)	—	—	—	—
(1,0)	0	—	0	—
(1,1)	1	—	—	1

Рис. 2.7. Можливі значення $KL_{8,2}$ і $KL_{8,1}$

4. Знаходження правильного ключа: для кожного значення 96 біт ($KO_{8,1}$, $KI_{8,1}$, $KO_{8,3}$, $KI_{8,3}$, $KL_{8,1}$, $KL_{8,2}$) запропонованого в пункті 3, припустити 32 біти ключа, що залишилися, і виконати пробне шифрування.

Складність даних атаки складає 2^{25} обраних шифротекстів і 2^{25} адаптивно обраних відкритих текстів шифрованих/дешифрованих під одним з чотирьох ключів. Часова складність – це переважно пробні шифрування виконані в підрозділі 2.5, щоб знайти останні 32 біти ключа, і вона складає близько 2^{32} шифрувань.

Складність пам’яті атаки є також невеликою. Потрібно лише зберегти 2^{26} пар відкритих текстів/шифротекстів, де кожна пара займає 16 байт. Отже, повний об’єм пам’яті, що використовується в атаці – 2^{30} байт, тобто приблизно – 1 Гбайт.

2.8. Прямокутна сендвіч атака з пов'язаними ключами на повний KASUMI

Сендвіч порівнювач з пов'язаними ключами 7-раундового KASUMI представлений в розділі 4 і може бути трансформований в стандартний спосіб в прямокутний сендвіч порівнювач з пов'язаними ключами, в якому ймовірність, що квартет буде правильним квартетом є $(1/4)^2 * (1/4)^2 * 2^{-64} * 2^{-6} = 2^{-78}$. Варто зазначити, що завдяки типу обраних відкритих текстів можна запевнити, що перший раунд диференціальної характеристики для E_0 проходить з ймовірністю швидше 1, а не 1/2, отже кінцева ймовірність – 2^{-76}. Цей порівнювач можна використати для прямокутної сендвіч атаки з пов'язаними ключами на повний KASUMI. Ця атака є дуже подібною до атаки детально представленої в [17], отже опускаємо пояснення, і просто вкажемо зміни.

Замість початкових 2^{51} відкритих текстів в кожній структурі, достатньо 2^{39} відкритих текстів. Ці відкриті тексти містять 2^{78} можливих квартетів, і після пункту фільтрування залишиться лише 2^{14} квартетів. Тоді в пункті 2(a) атаки отримуємо 2^{30} пропозицій для 48 біт (замість 2^{54} як в [17]), і всі хибні пропозиції можна відкинути (оскільки правильні пари пропонують однакове значення). В результаті, часова складність наступних кроків стає незначною, і загальна часова складність складається переважно з часу потрібного для шифрування 2^{41} обраних відкритих текстів. Це гірше, ніж часова складність 2^{32} сендвіч атаки, але досі є практичною, і має вагому перевагу – потребує лише обраних відкритих текстів, замість обраних шифротекстів/адаптивно підібраних відкритих текстів в сендвіч атаці.

Так як і в звичайній сендвіч атаці, пам'ять в прямокутній сендвіч атаці використовується переважно для зберігання відкритих текстів і шифротекстів. Після першого шифрування даних з ключами K_a і K_b, результат зберігається в сортованій таблиці, і потім шифруючи дані з ключами K_c і K_d методом пара за парою, потрібно зберегти лише 2^{40}

відкритих текст/шифротекст пар. Отже, Загальна складність пам’яті близько 16 Тбайт (2^{44} байт). Але доступ до пам’яті є послідовним, і може бути повільним, тому потрібно лише декілька жорстких дисків, щоб помістити ці дані.

2.9. Висновки до розділу

Досліджено особливості сендвіч атаки зі пов’язаними ключами та спроектовано алгоритм криптоаналізу. Дана атака розглядає шифр як каскад з трьох субшифрів. Розглянутий метод криптоаналізу алгоритму шифрування KASUMI має складність даних 2^{24} відкритих текстів та 2^{24} шифротектсів. Часова складність рівна 2^{32}, оскільки найбільш ємнісним за часом є перебір останніх 32 біт ключа. Розроблено алгоритм криптоаналізу, що складається з чотирьох етапів: планування пов’язаних ключів та збору даних, ідентифікація правильних квартетів, аналіз правильних квартетів, знаходження правильного ключа.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ПРОГРАМИ

3.1. Реалізація алгоритму KASUMI

Вихідний алгоритм KASUMI реалізовано на мові C++, взявши за основу [18]. Використання реалізації KASUMI, що подається в технічній специфікації [5], не було використано, оскільки в ній відсутня реалізація функції шифрування. Алгоритм KASUMI поданий в додатку А.

Ініціалізація екземпляру класу Kasumi здійснюється з двома значеннями – keyLeft, keyRight, які є лівою та правою частинами відповідно 128-бітного ключа шифрування.

В даній реалізації шифрування здійснює метод encrypt(), а дешифрування decrypt (). Дані методи приймають 2 значення – вхідний 64-бітний блок даних та кількість раундів, повертають – вихідний 64-бітний блок.

В конструкторі класу Kasumi здійснюється поділ двох 64-бітних частин ключа на вісім 16-бітних частин. Метод keySchedule() відповідає за планування ключів, тобто здійснює їх ініціалізацію. Метод InitializeInstanceFields() здійснює ініціалізацію сталої C.

Методи FO(), FI(), FL() здійснюють операції над вхідними даними відповідно до функцій FO, FI, FL алгоритму KASUMI.

Метод ZE() здійснює умовне розширення вхідного значення, оскільки згідно алгоритму KASUMI, в функції FI відбувається розширення 7-бітного значення до 9-бітного, шляхом додавання двох старших бітів, з нульовими значеннями. Дана операція реалізована шляхом операції логічного «І» вхідного значення з сталою 0x1FF. В методі TR() відбувається зворотна операція – перетворення 9-бітного значення в 7-бітне. Це реалізовано шляхом операції логічного «І» з сталою 0x7F.

3.2. Реалізація алгоритму криптоаналізу

Реалізація алгоритму криптоаналізу представлена у додатку Б. Реалізований клас DataCollect, який містить всі методи, які відповідають кожному етапу криптоаналізу.

Згідно першого етапу алгоритму криптоаналізу, проводиться пошук правильних квартетів. Але спершу відбувається ініціалізація екземпляру класу DataCollect, передаючи два параметри keyLeft та keyRight, дві частини ключа, відносно яких будується залежність пов'язаних ключів: $К_a$, $К_b = К_a \oplus \Delta К_{ab}$, $К_c = К_a \oplus \Delta К_{ac}$, $К_d = К_c \oplus \Delta К_{ab}$, де $\Delta К_{ab} = (0, 0, 8000_x, 0, 0, 0, 0, 0)$ і $\Delta К_{ac} = (0, 0, 0, 0, 0, 0, 8000_x, 0)$. Дані операції реалізовані наступним чином:

```
akeyLeft = keyLeft;
akeyRight = keyRight;
bkeyLeft = akeyLeft ^ 0x0000000080000000L;
bkeyRight = akeyRight;
ckeyLeft = akeyLeft;
ckeyRight = akeyRight ^ 0x0000000080000000L;
dkeyLeft = ckeyLeft ^ 0x0000000080000000L;
dkeyRight = ckeyRight;
```

Вище описана частина коду визначена в конструкторі класу DataCollect. Кожен з пов'язаних ключів складається з двох частин.

Оголошуються чотири екземпляри класу KASUMI з чотирма пов'язаними ключами, для подальшого здійснення шифрування та дешифрування:

```
Kasumi *ka = new Kasumi(dc->akeyLeft, dc->akeyRight);
Kasumi *kb = new Kasumi(dc->bkeyLeft, dc->bkeyRight);
Kasumi *kc = new Kasumi(dc->ckeyLeft, dc->ckeyRight);
Kasumi *kd = new Kasumi(dc->dkeyLeft, dc->dkeyRight);
```

Для знаходження правильних квартетів, необхідно визначити попередні значення valCa, valCb, valCc та valCd. Згідно алгоритму необхідно $2^{24} = 16777216$ попередніх значень квартетів.

64-бітне значення valCa складається з лівої 32-бітної частини, яка генерується псевдовипадковим генератором, та з правої 32-бітної частини, яка є сталою і в прийнята за 0xffffffff.

64-бітне значення valCc також складається з двох 32-бітних части: лівої та правої. Ліва частина генерується генератором псевдовипадкових чисел. Права частина рівна результату операції додавання за модулем 2 сталої, що використовується в ініціалізації valCa та значення 0x00100000.

Вище описані операції реалізовані наступним чином:

```
void InitInValue()
{
	long i;
	srand(time(NULL));
	for(i=0; i<16777216; i++)
	{
		valCa[i] = LongGen() | aConst;
		valCc[i] = LongGen() | (aConst ^ 0x0000000000100000L);
	}
}
```

Генератор псевдовипадкових чисел:

```
long long LongGen()
{

	int q;
	long long bac = 0x0;
	for (q=0; q<2; q++)
	{
		bac = (bac | static_cast<long long>(rand() % 0xFFFF));
		if (q != 1)
		bac = bac << 16;
	}
```

```
      bac = bac << 32;
      return bac;
}
```

Пошук попередніх значень valCc та valCd відбувається у наступний спосіб:

```
for(i=0; i<16777216; i++)
{
      dc->valCb[i] = kb->encrypt((ka->decrypt(dc->valCa[i], 7) ^
0x0000000000100000L), 7);
      dc->valCd[i] = kd->encrypt((kc->decrypt(dc->valCc[i], 7) ^
0x0000000000100000L), 7);
}
```

Тобто спершу відбувається дешифрування значень valCa та valCc, і відповідні результуючі значення додають за модулем 2 до значення 0x0000000000100000, після чого результати шифрують.

Значення valCa та valCb об’єднуються в пару:

```
for(i=0; i<16777216; i++)
{
      dc->PairCaCb(dc->valCa[i], dc->valCb[i]);
}
```

Метод PairCaCb здійснює запис пари valCa-valCb як значення за ключем, що собою представляє праву 32-бітну частину значення valCb, в асоціативну структуру даних multimap:

```
void PairCaCb(long long valOne, long long valTwo)
{
      pairCaCb = make_pair(valOne, valTwo);
      tmpPair = make_pair(static_cast<int>(valTwo), pairCaCb);
      mmpOne.insert(tmpPair);
}
```

Multimap – тип асоціативного контейнера, який зберігає елементи у формі пари ключа та значення. Основна відмінність від контейнера map – можливість зберігати різні елементи з однаковим ключем. Клас multimap визначений в заготовочному файлі map. Multimap являється одним з компонентів Стандартної бібліотеки шаблонів (STL). STL – набір узгоджених узагальнених алгоритмів, контейнерів, засобів доступу до їх вмісту та інших допоміжних функцій.

Метод EqualsCheck() здійснює пошук в multimap mmpOne пар за ключем, що результатом операції додавання за модулем 2 значень valCd та 0x0000000000100000. При знаходженні збігу значення з mmpOne, тобто пара valCa-valCb, разом з відповідною парою valCc-valCd зберігаються в двовимірному масиві aMas, в якому квартети розміщуються за ключем, що є результатом операції додавання за модулем 2 лівих 32-бітних частин valCa та valCc. Вище описані операції реалізовані наступним способом:

```
void EqualsCheck()
{
      long i;
      multimap <int, longpair> :: iterator iter;
      pair <multimap <int,longpair>::iterator, multimap <int,longpair> ::
iterator> ret;
      for(i=0; i<16777216; i++)
      {
            ret = mmpOne.equal_range(static_cast<int>(valCd[i] ^
0x0000000000100000L));
            for (iter=ret.first; iter!=ret.second; ++iter)
            {
                  pairCaCb = iter->second;
                  sMas[c] = static_cast<int>(pairCaCb.first>>32) ^
static_cast<int>(valCc[i]>>32);
                  aMas[c][0] = sMas[c];
                  aMas[c][1] = pairCaCb.first;
                  aMas[c][2] = pairCaCb.second;
                  aMas[c][3] = valCa[i];
                  aMas[c][4] = valCb[i];
                  c=c++;
```

```
            }
      }
      printf("quartets = %d\n", c);
}
```

Змінна c вказує на кількість попередніх значень квартетів, що записані в aMas.

Згідно алгоритму криптоаналізу правильними квартетами є квартети що мають хоча б потрійний збіг ключів. Метод Stest() здійснює моніторинг правильних квартетів:

```
void Stest()
{
      vector <long> tVector;
      vector <long> fRes;
      tVector.assign(sMas, sMas+c);
      stable_sort(tVector.begin(), tVector.end());
      for (vector <long>::iterator it=tVector.begin();it!=(tVector.end()-
3);++it)
      {
            if (*it==*(it++) && *it==*(it+2))
            {
                  fRes.push_back(*it);
            }
      }
      vector <long>::iterator itsec;
      itsec = unique(fRes.begin(), fRes.end());
      fRes.resize(distance(fRes.begin(), itsec));
      for (vector <long>::iterator itth=fRes.begin();itth!=fRes.end();++itth)
      {
            index = *itth;
            printf("index = %8lx\n", index);
      }
      for (s=0; s<c; s++)
      {
            if (aMas[s][0] == index)
            {
                  printf("Ca = %16llx\n", aMas[s][1]);
                  printf("Cb = %16llx\n", aMas[s][2]);
```

```
                printf("Cc = %16llx\n", aMas[s][3]);
                printf("Cd = %16llx\n", aMas[s][4]);
            }
        }
}
```

Для знаходження ключів $KL_{8,1}$ та $KL_{8,2}$ необхідно припустити значення $KO_{8,1}$, $KI_{8,1}$, $KO_{8,3}$ та $KO_{8,3}$. Для пар (C_a, C_c) та (C_b, C_d) потрібно обчислити вхідні та вихідні різниці для операцій І та АБО. Значення $KL_{8,1}$ та $KL_{8,2}$ визначаються з даних рисунку 1.14.

Останні 32 біт ключа, а саме $KI_{8,2}$ та $KO_{8,2}$ шукаються шляхом повного перебору, тобто 2^{32} ітерацій. Значення цих ключів в кожній наступній ітерації інкрементуються на одиницю.

3.3. Оптимізація з використанням технології MPI

Оптимізація програми криптоаналізу алгоритму KASUMI здійснюється з використанням технології MPI, зокрема реалізацій OpenMPI та Intel MPI. Технологія MPI – програмний інтерфейс для передачі інформації, який дозволяє обмінюватися повідомленнями між процесами, що виконують одне завдання[19]. В першу чергу MPI орієнтований на системи з розподіленою пам’яттю.

Метод MPI_Init() ініціалізує MPI середовище виконання програми:

```
MPI_Init(&argc, &argv);
```

Методи MPI_Comm_size() та MPI_Comm_rank() присвоюють змінним значення кількості ініціалізованих процесів та рангів процесів:

```
MPI_Comm_size(MPI_COMM_WORLD,&numtasks);
MPI_Comm_rank(MPI_COMM_WORLD,&rank);
```

Одним із аргументів вищеописаних методів є комунікатор MPI_COMM_WORLD, який є комунікатором по замовчуванню, і встановлюється методом MPI_Init(). Даний комунікатор містить всі процеси, що показано на рис. 4.1.

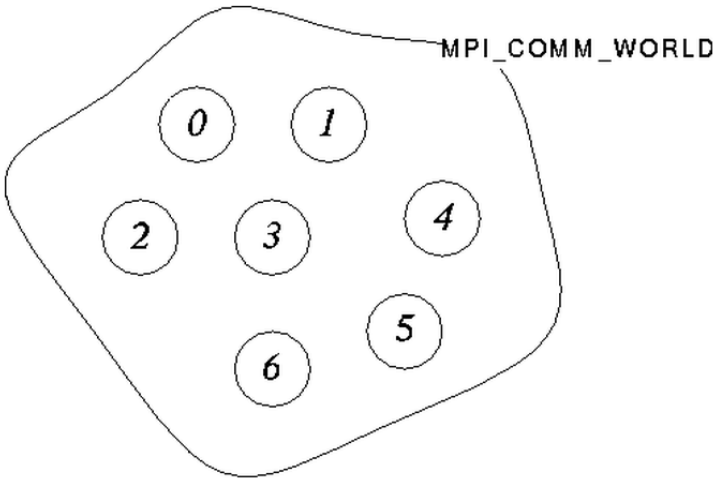

Рис. 3.1. Комунікатор MPI_COMM_WORLD

Ініціалізація екземплярів класів Kasumi та DataCollect здійснюється кожним процесом.

Розподілення даних для обробки виконується методом MPI_Scatter(), аргументами якого є: адреса даних для відправки, кількість елементів для відправки кожному процесу, тип елементів, адреса даних для отримання, кількість отриманих елементів, тип елементів, ранг процесу відправки, комунікатор. В даній програмі здійснюється надсилання значень valCa та valCc:

```
MPI_Scatter(dc->valCa, div, MPI_LONG_LONG, bufCa, div, MPI_LONG_LONG, 0,
MPI_COMM_WORLD);
MPI_Scatter(dc->valCc, div, MPI_LONG_LONG, bufCc, div, MPI_LONG_LONG, 0,
MPI_COMM_WORLD);
```

Збирання даних здійснюється методом MPI_Gather(), аргументи якого – такі ж як і у вищеописаного методу MPI_Scatter():

```
MPI_Gather(bufCb, div, MPI_LONG_LONG, dc->valCb, div, MPI_LONG_LONG, 0,
MPI_COMM_WORLD);
MPI_Gather(bufCd, div, MPI_LONG_LONG, dc->valCd, div, MPI_LONG_LONG, 0,
MPI_COMM_WORLD);
```

Для синхронізації роботи процесів використовується метод MPI_Barrier(), аргументом якого є комунікатор MPI_COMM_WORLD:

```
MPI_Barrier(MPI_COMM_WORLD);
```

Для завершення виконання в середовищі MPI використовується метод MPI_Finalize().

3.4. Тестування програми

Для тестування виконання програми криптоаналізу алгоритму KASUMI використано два вузли А і Б. Вузол А має наступні характеристики: Intel Core i3-2120 CPU 3,3 ГГц, 8 ГБ ОЗП. Характеристики вузла Б: Intel Core i3-3120 CPU 2,5 ГГц, 4 ГБ ОЗП. З'єднання між вузлами – Fast Ethernet (100 Мбіт/с).

Тестування послідовного виконання програми криптоаналізу алгоритму шифрування KASUMI здійснювалось на вузлі А. Середній час виконання програми склав 341 с.

Тестування паралельного виконання програми криптоаналізу алгоритму шифрування KASUMI проводилось двома способами: з використанням вузла А в першому випадку, та вузлів А і Б – в другому випадку.

Результати, які отримали при тестуванні в першому випадку, представлені в таблиці 3.1.

Таблиця 3.1

Результати тестування програми на 1 вузлі

Кількість процесів	Час виконання, с				
	Запуск 1	Запуск 2	Запуск 3	Запуск 4	Середнє значення
2	205	212	202	202	205
3	225	234	233	220	228
4	208	203	204	205	205
5	177	175	177	174	176
6	155	157	154	158	156
7	141	144	144	142	143
8	131	133	131	133	132

Очевидно, що з ростом кількості процесів зменшується середній час виконання програми. Виконання програми з 2 процесами швидше, ніж виконання з 3 процесами, оскільки це обумовлено архітектурою процесора вузла, який фізично має 2 ядра, але підтримує віртуалізацію в 4 потоки. На основі даних з таблиці 3.1 побудована гістограма, яка подана на рис. 3.2.

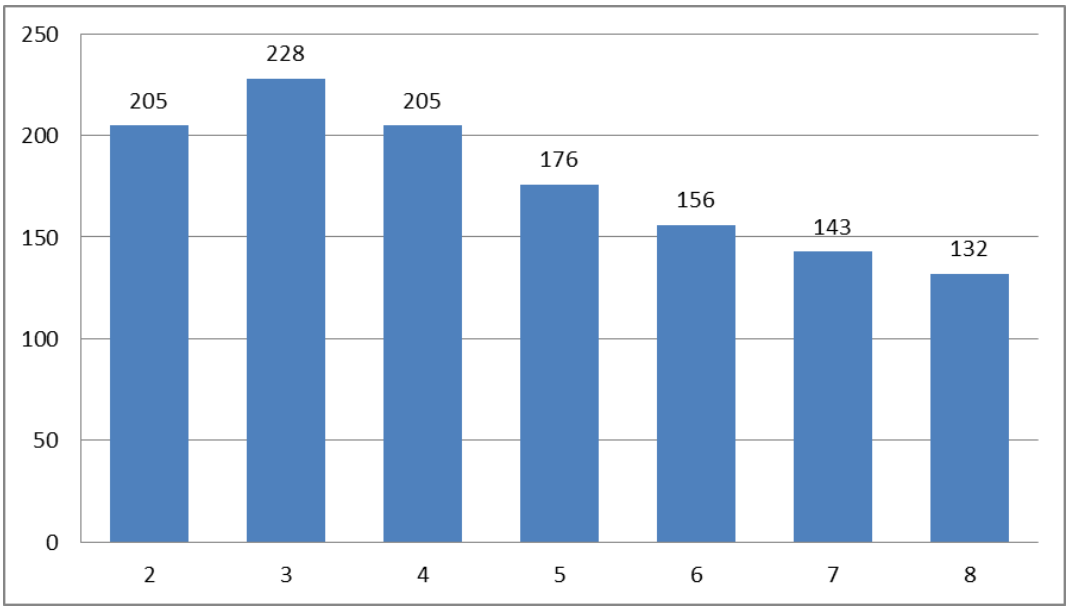

Рис. 3.2. Залежність часу виконання програми від кількості процесів (1 вузол)

Дані тестування другим способом представлені в таблиці 4.2.

Таблиця 3.2

Результати тестування програми на 2 вузлах

Кількість процесів	Час виконання, с				
	Запуск 1	Запуск 2	Запуск 3	Запуск 4	Середнє значення
2	268	261	264	266	265
3	203	195	200	198	199
4	177	172	177	173	175
5	205	185	191	200	195
6	203	200	203	201	202
7	190	191	185	182	187
8	192	177	187	184	185

Дані з таблиці 3.2 не зберігають суворої тенденції зменшення часу виконання від збільшення кількості процесів, на відміну від даних з таблиці 3.1. Відхилення отриманих значень від середнього значення більше ніж у випадку тестування на одному вузлі. Це можна пояснити тим, що при взаємодії вузлів, через мережу передається велика кількість даних, і тому важливого значення набуває латентність мережі та швидкість передачі інформації. На основі даних з таблиці 3.2 побудована гістограма, яка подана на рис. 3.3.

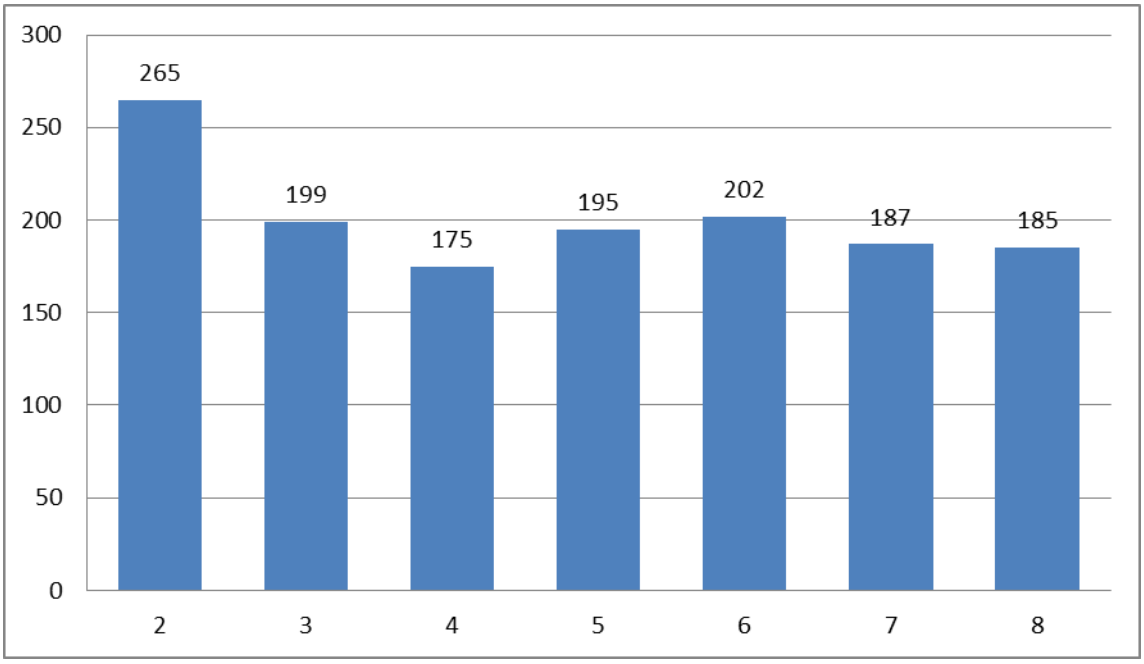

Рис. 3.3. Залежність часу виконання програми від кількості процесів (2 вузла)

Слід також звернути увагу на те, що важливу роль відіграє синхронізація MPI процесів. Оскільки, можливі випадки «гонки процесів», а також боротьби за ресурси системи. Тому в різних випадках час синхронізації може значно відрізнятися.

3.5. Профілювання

Профілювання – збір та аналіз інформації про виконання програми з метою оптимізації її роботи. Профілювання програми криптоаналізу алгоритму шифрування KASUMI здійснювалась з використанням Intel Cluster Studio – високопродуктивний гібридний набір для розробки паралельних програм.

Компіляція програми здійснювалась оптимізуючим компілятором Intel C++ Compiler, а також з використанням пропрієтарної бібліотеки Intel MPI Library. Виконання програми з чотирма процесами, з врахуванням вищеописаних оптимізуючих засобів, склало 156 с.

На рис. 3.4 представлена діаграма подій виконуваної програми. Дана діаграма дає графічне відображення виконання програми процесами, а також їх взаємодію у вигляді колективних MPI операцій.

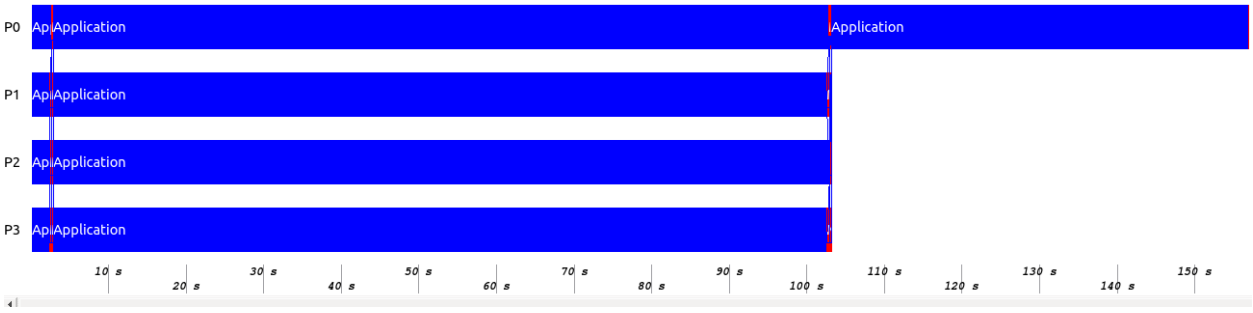

Рис. 3.4. Діаграма подій

На рис. 3.5 та 3.6 детально представлено взаємодію процесів при розсиланні та зборі даних головним процесом.

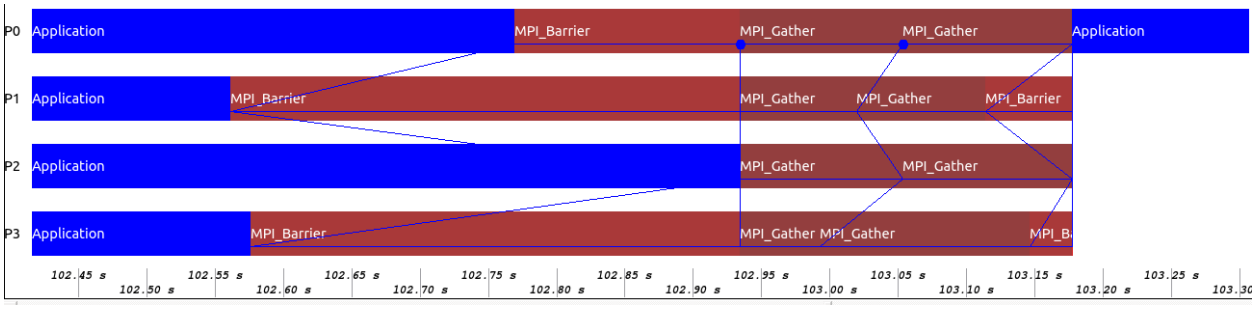

Рис. 3.5. Взаємодія процесів при розсиланні даних

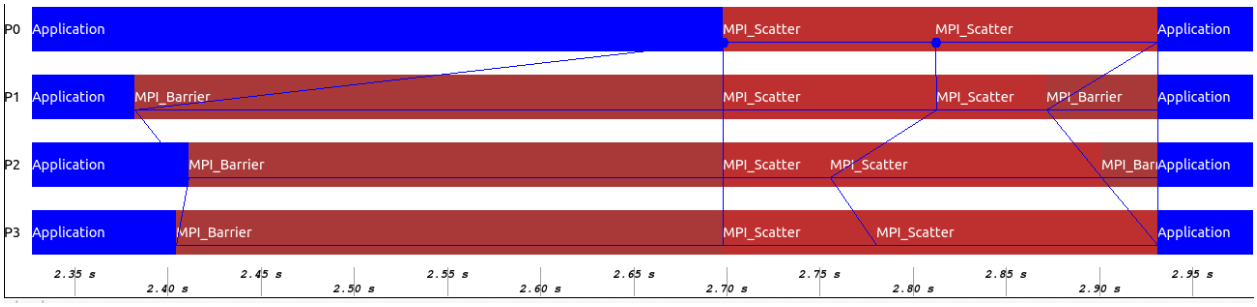

Рис. 3.6. Взаємодія процесів при зборі даних

На рис. 3.7 представлені дані виконання методів кожним процесом. В ній представлений час виконання процесом методу та кількість викликів процесом методу.

Name	TSelf	#Calls	TSelf /Call
Process 0			
Group Application	156.302 s	1	156.302 s
MPI_Comm_size	1e-6 s	1	1e-6 s
MPI_Comm_rank	1e-6 s	1	1e-6 s
MPI_Finalize	42.465e-3 s	1	42.465e-3 s
MPI_Scatter	233.103e-3 s	2	116.551e-3 s
MPI_Barrier	166.059e-3 s	5	33.2118e-3 s
MPI_Gather	243.038e-3 s	2	121.519e-3 s
Process 1			
Group Application	102.012 s	1	102.012 s
MPI_Comm_size	1e-6 s	1	1e-6 s
MPI_Comm_rank	1e-6 s	1	1e-6 s
MPI_Finalize	277e-6 s	1	277e-6 s
MPI_Scatter	173.475e-3 s	2	86.7375e-3 s
MPI_Barrier	813.171e-3 s	5	162.634e-3 s
MPI_Gather	179.401e-3 s	2	89.7005e-3 s
Process 2			
Group Application	102.414 s	1	102.414 s
MPI_Comm_size	1e-6 s	1	1e-6 s
MPI_Comm_rank	1e-6 s	1	1e-6 s
MPI_Finalize	270e-6 s	1	270e-6 s
MPI_Scatter	203.049e-3 s	2	101.524e-3 s
MPI_Barrier	317.235e-3 s	5	63.447e-3 s
MPI_Gather	242.901e-3 s	2	121.45e-3 s
Process 3			
Group Application	102.049 s	1	102.049 s
MPI_Comm_size	1e-6 s	1	1e-6 s
MPI_Comm_rank	1e-6 s	1	1e-6 s
MPI_Finalize	269e-6 s	1	269e-6 s
MPI_Scatter	233.236e-3 s	2	116.618e-3 s
MPI_Barrier	683.512e-3 s	5	136.702e-3 s
MPI_Gather	211.962e-3 s	2	105.981e-3 s

Рис. 3.7. Виконання методів процесами

На рис. 3.8 подані статистичні дані виконання колективних операцій: сума часу виконання, середнє значення та середньоквадратичне відхилення.

	P0	P1	P2	P3	Sum	Mean	StdDev
MPI_Barrier	165.695e-3	812.79e-3	317.23e-3	683.131e-3	1.97885	494.711e-3	262.882e-3
MPI_Gather	243.038e-3	179.401e-3	242.901e-3	211.962e-3	877.302e-3	219.325e-3	26.2977e-3
MPI_Scatter	233.103e-3	173.475e-3	203.049e-3	233.236e-3	842.863e-3	210.716e-3	24.7689e-3
Sum	641.836e-3	1.16567	763.18e-3	1.12833	3.69901		
Mean	213.945e-3	388.555e-3	254.393e-3	376.11e-3		308.251e-3	
StdDev	34.3584e-3	299.989e-3	47.3172e-3	217.271e-3			202.155e-3

Рис. 3.8. Статистичні дані виконання колективних операцій

Згідно з результатами профілювання, можна зробити висновок про те, що більшість часу виконання колективних операцій витрачається на синхронізування потоків методом MPI_Barrier.

3.6. Висновки до розділу 3

Обгрунтовано технологію програмування на мові С та засоби оптимізації – технологію MPI, що дає змогу здійснити розпарарлелення виконання програми на декілька процесів.

Реалізовано програму криптоаналізу алгоритму шифрування KASUMI метод сендвіч атаки з пов’язаними ключами, зокрема реалізовано клас Kasumi, методи якого виконують шифрування та дешифрування даних, та DataCollect, методи якого виконують операції згідно з етапами алгоритму криптоаналізу.

Здійснено тестування: послідовне виконання програми тривало 341 с., паралельне виконання з чотирма процесами – 205с. Проведено профілювання програми, що дало змогу проаналізувати режим комунікацій процесів між собою, а також визначити структуру використання ресурсів системи процесами та методами програми.

РОЗДІЛ 4
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1. Охорона праці

В сучасному світі охорона праці має велике значення у зв’язку з інтенсивним розвитком різноманітних галузей діяльності людини. Дотримання її принципів дозволяє вирішити цілий ряд задач, серед яких: гарантування захисту персоналу від шкідливих та небезпечних факторів, зменшення видатків на забезпечення виробничого процесу, виключення втрат робочого часу, підвищення продуктивності та якості праці.

Тема дипломної роботи магістра – «Дослідження захищеності криптографічних засобів захисту мережі передачі даних UMTS». Під час проведення досліджень використовувалась електронно-обчислювальна техніка для тестування та профілювання програми криптоаналізу. Тому важливими постають питання безпечної експлуатації ЕОМ, організації робочого місця, дотримання заходів електробезпеки.

При роботі з ЕОМ потрібно дотримуватись правил і норм, які описані в НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин». Згідно з вище згаданим документом, робочі місця мають відповідати вимогам ДСанПіН 3.3.2.007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин». А саме площа на одне робоче місце має становити не менше ніж 6,0 кв. м, а об'єм не менше ніж 20,0 куб. м.

Приміщення для проведення дослідження захищеності криптографічних засобів повинні мати природне та штучне освітлення відповідно до ДБН В.2.5-28-2006 «Природне і штучне освітлення». Природне освітлення має здійснюватись через світлові прорізи, орієнтовані переважно на північ чи північний схід і забезпечувати коефіцієнт природної освітленості

(КПО) не нижче ніж 1,5 %. Віконні прорізи приміщень мають бути обладнані регульованими пристроями (жалюзі, завіски, зовнішні козирки).

Також дані приміщення мають бути обладнані системами опалення, кондиціонування повітря, або припливно-витяжною вентиляцією відповідно до ДБН В.2.5-67:2013 «Опалення, вентиляція та кондиціонування», а також оснащенні аптечками першої допомоги. Щоденно має проводитися вологе прибирання.

У приміщеннях, що використовуються при проведені досліджень, мають забезпечуватись допустимі значення параметрів мікроклімату відповідно до ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень». Оптимальні значення:

- температура повітря повинна становити в холодний період року 22-24 ºС, в теплий період – 23-25 ºС;
- відносна вологість повинна бути в межах 40-60%;
- швидкість руху повітря – 0,1 м/с.

Для забезпечення гігієнічних вимог до організації і обладнання робочого місця з ЕОМ, конструкція всіх елементів робочого місця та їх взаємного розташування має відповідати ергономічним вимогам з урахуванням характеру і особливостей трудової діяльності (ГОСТ 12.2.032-78, ГОСТ 22.269-76, ГОСТ 21.889-76). Робоче місце розташоване відносно світових прорізів таким чином, що природнє світло падало зліва. Екран має розташовуватися на оптимальній відстані від очей користувача, що становить 600...700 мм, але не ближче ніж за 600 мм з урахуванням розміру літерно-цифрових знаків і символів,а також має забезпечуватись зручність зорового спостереження у вертикальній площині під кутом + 30 град. до нормальної лінії погляду працюючого.

При проведені досліджень криптостійкості елементів захисту мережі UMTS, необхідно дотримуватися заходів електробезпеки, які визначені в НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин», НПАОП 40.1-1.21-98 «Правила

безпечної експлуатації електроустановок споживачів», ПУЕ:2009 «Правила улаштування електроустановок». Електронно-обчислювальна техніка повинна підключатися до електромережі тільки за допомогою справних штепсельних з'єднань і електророзеток заводського виготовлення. У штепсельних з'єднаннях та електророзетках, крім контактів фазового та нульового робочого провідників, мають бути спеціальні контакти для підключення нульового захисного провідника. Їхня конструкція має бути такою, щоб приєднання нульового захисного провідника відбувалося раніше, ніж приєднання фазового та нульового робочого провідників. Порядок роз'єднання при відключенні має бути зворотним. Не допускається підключення обчислювальної техніки до звичайної двопровідної електромережі, в тому числі – з використанням перехідних пристроїв.

Під час дослідження криптостійкості засобів захисту системи UMTS, для збереження здоров'я, запобігання професійним захворюванням і підтримки працездатності слід передбачити внутрішньозмінні регламентовані перерви для відпочинку, а також необхідно передбачати нетривалі перерви в періоди що передують появі об'єктивних і суб'єктивних ознак втомлення і зниження працездатності.

Відповідно до НПАОП 0.00-1.28-10 необхідно перед початком роботи очищати екран від пилу та інших забруднень, а також перевірити справність підключення до електричної мережі, перевірити кабелі живлення на пошкодження, пересвідчитися у цілісності корпусів і блоків обчислювальної техніки, при виявленні пошкоджень забороняється експлуатація комп’ютерної техніки. Під час роботи, пересвідчившись у справності обладнання, увімкнути електроживлення, розпочати роботу, дотримуючись умов інструкції з її експлуатації даної техніки. Забороняється при поданій напрузі торкатися до проводів живлення, з’єднувальних кабелів та тильної сторони екрану. Після закінчення роботи електронно-обчислювальна техніка повинна бути відключена від електричної мережі. У разі виникнення аварійної ситуації необхідно негайно відключити ЕОМ і периферійні

пристрої від електричної мережі. Суворо забороняється обслуговування, ремонт і налагодження ЕОМ безпосередньо на робочому місці.

Отже, при проведенні дослідження захищеності криптографічних засобів захисту мережі передачі даних UMTS, дотримання правил охорони праці під час експлуатації електронно-обчислювальної техніки та заходів електробезпеки забезпечить безпеку праці.

4.2. Надзвичайні ситуації при пожежах

Цивільний захист – це функція держави, спрямована на захист населення, територій, навколишнього природного середовища та майна від надзвичайних ситуацій шляхом запобігання таким ситуаціям, ліквідації їх наслідків і надання допомоги постраждалим у мирний час та в особливий період.

Правовою основою цивільного захисту є Конституція України, Кодекс цивільного захисту України, інші закони України, а також акти Президента України та Кабінету Міністрів України.

Надзвичайна ситуація – обстановка на окремій території чи суб’єкті господарювання на ній або водному об’єкті, яка характеризується порушенням нормальних умов життєдіяльності населення, спричинена катастрофою, аварією, пожежею, стихійним лихом, епідемією, епізоотією, епіфітотією, застосуванням засобів ураження або іншою небезпечною подією, що призвела (може призвести) до виникнення загрози життю або здоров’ю населення, великої кількості загиблих і постраждалих, завдання значних матеріальних збитків, а також до неможливості проживання населення на такій території чи об’єкті, провадження на ній господарської діяльності.

Надзвичайні ситуації класифікуються за характером походження, ступенем поширення, розміром людських втрат та матеріальних збитків.

Залежно від характеру походження подій, що можуть зумовити виникнення надзвичайних ситуацій на території України, визначаються такі види надзвичайних ситуацій: техногенного характеру, природного характеру, соціальні, воєнні.

Залежно від обсягів заподіяних надзвичайною ситуацією наслідків, обсягів технічних і матеріальних ресурсів, необхідних для їх ліквідації, визначаються такі рівні надзвичайних ситуацій: державний, регіональний, місцевий, об’єктовий.

Забезпечення пожежної безпеки — це один із важливих напрямків щодо охорони життя та здоров'я людей, національного багатства і навколишнього середовища.

Основним нормативним документом, що регламентує вимоги щодо пожежної безпеки, є Кодекс цивільного захисту України, прийнятий 2 жовтня 2012 року, та введений в дію з 1 липня 2013 року. Цей кодекс визначає загальні правові, економічні та соціальні основи забезпечення пожежної безпеки на території України, регулює відносини державних органів, юридичних і фізичних осіб у цій галузі незалежно від виду їх діяльності та форм власності.

Пожежа – це процес горіння, що вийшов з-під контролю та знищує матеріальні цінності і створює загрозу життю і здоров'ю людей.

Найбільш поширеними джерелами виникнення надзвичайних ситуацій техногенного характеру є пожежі та вибухи, що відбуваються:

- на промислових об'єктах;
- на об'єктах видобутку, зберігання і переробки легкозаймистих, горючих та вибухових речовин;
- на транспорті;
- в шахтах, гірських виробках, метрополітенах;
- в будівлях і спорудах житлового, соціально-побутового та культурного призначення.

Основними причинами пожежі є: несправності в електричних мережах, порушення технологічного режиму і заходів пожежної безпеки.

Основними небезпечними факторами пожежі є теплове випромінювання, висока температура, отруйна дія диму (продуктів згоряння: окису вуглецю тощо) і зниження видимості при задимленні. Критичними значеннями параметрів для людини, при тривалому впливі вказаних значень небезпечних факторів пожежі, є:

- температура – 70 º С;
- щільність теплового випромінювання - 1,26 кВт/м2;
- концентрація окису вуглецю - 0,1% об'єму;
- видимість у зоні задимлення - 6-12 м.

Вибух – це горіння, що супроводжується звільненням великої кількості енергії в обмеженому об'ємі за короткий проміжок часу. Вибух призводить до утворення та розповсюдження з надзвуковою швидкістю вибухової ударної хвилі (з надлишковим тиском більше 5 кПа), що надає ударний механічний вплив на навколишні предмети.

Основними уражаючими факторами вибуху є повітряна ударна хвиля і уламки різного роду об'єктів, що летять, технологічного обладнання, вибухових пристроїв.

При пожежі потрібно остерігатися обвалу конструкцій будинків і споруд, вибухів технологічного обладнання і приладів.

При рятуванні потерпілих із будинків, що горять, і при гасінні пожежі, потрібно дотримуватись наступних правил:

- перш ніж увійти в приміщення, що горить, накритись мокрим покривалом, пальтом, плащем, щільною тканиною;
- двері в задимлене приміщення відкривати обережно, щоб уникнути займання від великого притоку свіжого повітря;
- у дуже задимленому приміщенні рухатись повзком або пригнувшись;
- для захисту від чадного газу дихати через зволожену тканину.

Потрібно дотримуватись правил використання засобів гасіння пожеж. Для приведення в дію пінного вогнегасника потрібно підняти рукоятку вгору і перекиньте до упору; потім перевернути вогнегасник догори дном. Струмінь піни, що виникає, направити на поверхню, яка горить (при відсутності струменю піни потрібно струсити вогнегасник або прочистити отвір).

Вуглекислотний вогнегасник слід направити розтрубом на поверхню, що горить, обертаючи маховичок проти ходу годинникової стрілки до упору, і відкрити запірний вентиль. Снігоподібною масою, яка викидається з розтруба, покрити поверхню, яка горить, до повного закінчення горіння. Не слід тримати розтруб голою рукою – можна отримати обмороження.

Для приведення в дію пожежних кранів, які знаходяться в будинку (споруді), необхідно розмотати в напрямку осередку пожежі рукав, який з'єднаний з краном і стволом. Відкрити вентиль поворотом маховика проти ходу годинникової стрілки і направити струмінь води із ствола на осередок горіння. Найдоступнішими засобами гасіння займань і пожеж є вода, пісок, або ґрунт, ручні вогнегасники, азбестові і брезентові покривала або одяг.

При знаходженні в загазованому приміщенні або на загазованій ділянці місцевості з метою запобігання виникненню пожежі (вибуху) газоповітряної суміші потрібно дотримуватися певних правил та норм. В загазованому приміщенні для запобігання виникнення іскріння, яке може призвести до вибуху (загорання газоповітряної суміші), заборонено вмикати і вимикати електричні прилади, користуватися електричними і акумуляторними ліхтарями, які не мають вибухонебезпечного виконання, користуватися електродзвінком, дзвонити по телефону, виконувати дії з металевими предметами, для виключення удару один об одного, користуватися відкритим вогнем.

На загазованій ділянці місцевості для запобігання виникнення іскріння, яке може призвести до вибуху (загорання газоповітряної суміші), заборонено ставити машини, заводити машини, що стоять, і проїжджати біля

загазованого газового колодязя, газорозподільного пункту, шафного газорозподільного пункту, групової резервуарної установки ближче 15 м з підвітряної сторони. Заборонено користуватися електричними і акумуляторними ліхтарями, які не мають вибухонебезпечного виконання, виконувати дії з металевими предметами з метою виключення удару один об одного, а також користуватися відкритим вогнем і розводити вогнища ближче 50 м від загазованої ділянки місцевості.

При виявленні запаху газу в приміщенні, де встановлені газові прилади негайно закрити крани на газових приладах і перед ними (при розміщенні балонів зі скрапленим газом додатково закрити вентилі у балонів), вжити заходи щодо виведення людей із загазованих приміщень (квартири, дому). З незагазованого приміщення за телефоном "104" викликати аварійну газову службу і організувати провітрювання приміщення. По прибутті бригади аварійної газової служби діяти за їх вказівками.

При вибуху (пожежі) в квартирі закрити всі крани на газових приладах і перед ними (при розміщенні балонів всередині кухні додатково закрити вентилі балонів). За телефоном "104" повідомити в аварійну газову службу. Організувати гасіння пожежі, повідомити за телефоном "101" черговому пожежної охорони. Організувати надання першої медичної допомоги потерпілим, при необхідності викликати за телефоном "103" швидку медичну допомогу. Не допускати сторонніх у приміщення, де виник вибух, зберегти місце і обстановку після вибуху (пожежі).

При запаху газу в під'їзді, підвалі житлового будинку або на вулиці за телефоном "104" повідомити місцеву аварійну газову службу. Організувати провітрювання під'їзду (підвалу) відкриттям дверей (по можливості і вікон). Організувати охорону місця загазованості до прибуття бригади аварійної газової служби, не допускати в загазовані приміщення людей з відкритим вогнем. По прибутті бригади аварійної газової служби діяти за їх вказівками.

При виникненні надзвичайної ситуації при пожежі, оповіщування про та управління евакуацією людей здійснюється згідно з НАПБ А.01.003-2009

Правила улаштування та експлуатації систем оповіщення про пожежу та управління евакуацією людей в будинках та спорудах. Оповіщенні здійснюється одним з таких способів або їх комбінацією:

- поданням звукових і (або) світлових сигналів в усі приміщення будинку з постійним або тимчасовим перебуванням людей;
- трансляцією текстів про необхідність евакуації, шляхи евакуації, напрямок руху й інші дії, спрямовані на забезпечення безпеки людей;
- трансляцією спеціально розроблених текстів, спрямованих на запобігання паніці й іншим явищам, що ускладнюють евакуацію;
- розміщенням знаків безпеки на шляхах евакуації згідно з ДСТУ ISO 6309;
- ввімкненням евакуаційних знаків «Вихід»;
- ввімкненням евакуаційного освітлення та світлових покажчиків напрямку евакуації;
- дистанційним відкриванням дверей евакуаційних виходів;
- зв'язком пожежного поста (диспетчерської) із зонами оповіщування.

Для запобігання виникненню надзвичайних ситуацій при пожежах, а також для гарантування безпеки людей, потрібно дотримуватись норм і правил, що в визначені в НАПБ А.01.001-2004 Правила пожежної безпеки в Україні. Зокрема, виконання попереджувальних заходів:

- усунення причин, які можуть викликати пожежу (вибух);
- обмеження (локалізація) розповсюдження пожеж;
- створення умов для евакуації людей і майна при пожежі;
- своєчасне виявлення пожежі та оповіщення;
- підтримання сил ліквідації пожеж у постійній готовності.

4.3. Висновки до розділу

При проведенні дослідження захищеності криптографічних засобів захисту мережі передачі даних UMTS, необхідно дотримуватись правил охорони праці під час експлуатації електронно-обчислювальної техніки та заходів електробезпеки, що в свою чергу забезпечить безпеку праці.

Для забезпечення пожежної безпеки, дослідження криптографічних засобів захисту мережі передачі даних UMTS повинно проводитись в приміщенні оснащеному системою автоматичної пожежної сигналізації та автоматичними установками гасіння пожежі. Необхідно розмістити додаткові вогнегасники. Потрібно дотримуватись вимог, наведених в правилах, а також проводити регулярні інструктажі серед працюючих з пожежної безпеки.

РОЗДІЛ 5
ЕКОЛОГІЯ

5.1. Відчуження земель під лінії електропередач

Відчуження земель при прокладці траси повітряних ліній електропередач (ЛЕП) призводить до порушення верхніх родючих шарів ґрунту, вирубки лісів, викликають перешкоди веденню сільськогосподарських робіт, зміни середовища існування тварин і птахів [19]. Розчищена при будівництві лінії електропередачі траса зазвичай досить швидко заростає, що призводить до збільшення числа відключень ЛЕП через перекриття, і вимагає періодичного розчищення трас або їх хімічної обробки арборицидами. Останнє також значно впливає на живі організми, що мешкають на трасах ліній. З метою захисту населення та персоналу, що проводить роботи поблизу ЛЕП , встановлено межі санітарно-захисних зон, в межах яких напруженість електричного поля перевищує 1 кВ/м.

Відчуження землі при спорудженні ЛЕП виконується у вигляді площадок для встановлення опор. Однак при виборі і погодженні трас ЛЕП апеляції з боку землекористувачів базуються не на об’ємах відчужених земель, а на завадах для використання сільськогосподарських угідь, створюваних ЛЕП. З цієї точки зору рекомендується оперувати поняттям охорони зон електричних мереж, які встановлюються повздовж ЛЕП у вигляді земельної ділянки, обмеженої вертикальними площинами по обидві сторони ліній. Ширина охоронної зони визначається за формулою:

$$Ш=2D+2l \qquad (5.1)$$

де D – відстань між фазами ЛЕП;

l – відстань від проекції на землю крайньої фази до зони в напрямку, перпендикулярному до ЛЕП.

Як правило, полоса відчуження не переходить у власність енергоорганізації. Земельні ділянки, що входять в охоронні зони, не вилучаються у землекористувачів, але зобов’язує дотримуватися певних вимог: заборонено розміщувати житлових і громадських будівель, організовувати стоянки транспорту і заправки його пальним, а також ремонт машин і механізмів. Сільгоспроботи дозволені, але, наприклад в охоронній зоні ЛЕП 750 кВ і більше тривалість всіх робіт не повинна перевищувати 1,5 години на добу. В таблиці 6.1 представлені відстані між крайніми проводами та площинами обмежень, площі і ширини охоронної зони.

Таблиця 5.1

Показники охоронних зон

Напруга ЛЕП, кВ	Відстань між крайнім проводом і площиною обмеження, м	Ширина охоронної зони, м	Площа охоронної зони, га/км
до 20	10	26	2,6
35	15	38	3,8
110	20	50	5,0
220	25	64	6,4
330	30	78	7,8
500	30	84	8,4
750	40	120	12,0

В лісових масивах, висотою більше 4 м, для всіх ЛЕП з напругою 330 кВ та вище, а також для радіальних ЛЕП з напругою 220 кВ, ширина просіки визначається відстанню між крайніми проводами лінії плюс відстані від крайніх проводів до лісного масиву, які рівні висоті дерев основного лісового масиву.

5.2. Методи узагальнення екологічної інформації

Узагальнення екологічної інформації – це стадія роботи з екологічною інформацією, коли оброблений екологічний матеріал потребує узагальнення, наочного подання та відображення складних екологічних ситуацій [20]. Методами наочного подання та викладання фізичних величин, що використовують для більш раціонального та систематизованого викладення цифрової інформації – статистичні таблиці і статистичні графіки, тобто табличний та графічний методи.

Статистичні таблиці – це форма раціонального та систематизованого викладання цифрової інформації. Основною перевагою цифрової інформації, зведеної в таблиці, є компактність, наочність, виразність. Інформація стає легкодоступною і рельєфною, компактною і раціональною.

Мета побудови таблиць багатогранна: систематизація цифрової інформації, полегшення і прискорення ефекту сприйняття, інтенсифікація пізнавального процесу, економія місця при викладені інформації.

Таблиці складають не лише на заключному етапі дослідження. В процесі обробки статистичних даних користуються допоміжними, робочими таблицями. Статистичними таблицями вважають тільки ті, що містять наслідки статистичного аналізу еколого-економічних явищ і процесів.

Таблиця за своїм логічним змістом розглядається як «статистичне речення», що має свій підмет і присудок. Підмет таблиці характеризує об’єкт дослідження, а присудок – це система показників, що відображує підмет як об’єкт.

Вірність побудови таблиць залежить від трьох компонентів: правильності розміщення підмета і присудка, правильності розробки присудка (присудок повинен мати мінімальне число рядків), послідовності розміщення показників присудка (найбільш важливі повинні розміщуватися попереду другорядних).

За призначенням і побудовою підмета таблиці поділяють на декілька видів:

- за призначенням – аналітичні і допоміжні;
- за побудовою підмета – прості, групові, комбінаційні.

Статистичні графіки – це спосіб умовного зображення цифрової інформації у вигляді крапок, ліній, стовпчиків, кругів або фігур. Статистичний графік являє собою рисунок, який описує статистичні сукупності умовною мовою геометричних знаків певної форми.

Мета побудови потрійна: популяризація цифрової інформації, забезпечення доступності сприйняття інформації, узагальнення цифрової інформації.

Призначення графіків багатогранне: порівняння між собою різних величин, характеристика складу, структури і структурних зрушень сукупностей, з'ясування ступеня розповсюдження явищ в просторі, виявлення взаємозв'язку між явищами та їх ознаками, виявлення хронологічних явищ і їх ознак, дослідження темпів, тенденцій, закономірностей і перспектив розвитку явищ.

Елементами графіку є графічний образ та допоміжні елементи. Графічний образ – це сукупність геометричних або графічних знаків, що замінюють числові дані і використовуються для зображення статистичних даних. Допоміжні елементи – складові частини, що роз'яснюють суть графічного образу: загальний заголовок, осі координат, числові дані на шкалах.

Поле графіка – це простір, у якому розміщуються геометричні або інші графічні знаки, що утворюють графік. Просторові орієнтири в статистичних графіках використовують для визначення порядку розміщення геометричних знаків у полі графіка. Масштабні орієнтири визначаються системою масштабних шкал або спеціальними знаками. Експлікація графіка являє собою словесне пояснення основних елементів графіка та його змісту.

5.3. Моніторинг довкілля та система спостережень за впливом на довкілля антропогенних факторів

Моніторинг довкілля – система спостереження і контролю за природними, природно-антропогенними комплексами, процесами, що відбуваються у них, навколишнім середовищем загалом з метою раціонального використання природних ресурсів і охорони довкілля, прогнозування масштабів неминучих змін [21].

Метою моніторингу довкілля є екологічне обґрунтування перспектив та удосконалення системи моніторингу навколишнього середовища,оцінювання фактичного і прогнозованого його стану; попередження про зниження біорізноманітності екосистем, порушення екологічного рівноваги у довкіллі, погіршенні умов життєдіяльності людей.

Предметом моніторингу довкілля як науки є організація і функціонування системи моніторингу, оцінювання і прогнозування стану екологічних систем, їх елементів, біосфери , характеру впливу на них природних і антропогенних факторів.

Об’єктами моніторингу довкілля, залежно від рівня мети досліджень, можуть бути навколишнє середовище, його елементи і джерела впливу на довкілля.

Антропогенні фактори – форми господарської діяльності людини, що впливають на організм чи екосистеми, природне середовище загалом.

Дію антропогенних факторів на біосферу оцінюють, зважаючи на зміни властивостей основних її елементів, геофізичні, геохімічні, біологічні, екологічні наслідки їх впливу (порушення в екосистемах), а також на зміни стану здоров’я людей.

Спостереження у межах системи моніторингу за дією основних антропогенних факторів і процесів, які вони зумовлюють, групують за такими напрямами:

- спостереження за локальними джерелами забруднення і забруднюючими факторами;
- спостереження за станом навколишнього природного середовища;
- спостереження за станом біотичної складової біосфери;
- спостереження за реакцією великих систем (клімату, Світового океану, біосфери);

Спостереження за локальними джерелами забруднення і забруднюючими факторами здійснюються на територіях окремих об'єктів у формі контролювання кількісного і якісного складу забруднюючих речовин, що містяться у викидах і скидах, місцях їх зберігання.

Спостереження за станом навколишнього природного середовища зосередженні на відстежу ванні геофізичних, фізико-географічних, геохімічних, хімічних явищ, процесів і змін з фіксуванням відповідних даних.

У процесі спостереження за станом біотичної складової біосфери відстежують реакції біоти на різні фактори, тобто реакції окремих організмів, популяцій, або угруповувань, а також спостерігають за функціональними і структурними біологічними ознаками.

Моніторинг за реакцією великих систем потребує фізичні, хімічні і біологічні показники. Для встановлення динаміки змін стану біосфери, заміри повторюють через певні проміжки часу, а важливі показники відстежують безперервно.

В організації спостережень активно використовують авіаційні і супутникові засоби. Отримані за їх допомогою результати аналізують з огляду на зміни середовищ, а також на відповідні реакції біоти, що виникли внаслідок антропогенного спливу.

Отже, основною метою моніторингу довкілля є спостереження за змінами в екосистемах, зумовленими антропогенними факторами.

5.4. Висновки до розділу

Розглянуто порядок відчуження замель під лінії електропередач, проаналізовано принципи визначення ширини та площі охоронної зони в залежності від напруги ліній.

Проаналізовано методи узагальнення екологічної інформації – статистичні таблиці і статистичні графіки, тобто табличний та графічний методи. Визначенно особливості даних методів та їх переваги та недоліки.

Розглянуто систему моніторингу довкілля за впливом антропогенних факторів. Визначено мету, предмет та об’єкти моніторингу. Проведено класифікацію систем моніторингу за дією основних антропогенних факторів і процесів.

РОЗДІЛ 6
ОБҐРУНТУВАННЯ ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ

Метою даного розділу дипломної роботи є здійснення економічних розрахунків, спрямованих на визначення економічної ефективності проведення дослідження захищеності криптографічних засобів захисту мережі передачі даних UMTS.

6.1. Визначення стадій технологічного процесу

Для здійснення економічної оцінки потрібно зробити розрахунки трудомісткості кожної операції, що мала місце при проведенні досліджень, що наведено в таблиці 6.1. Виконавцем усіх операцій являється інженер.

Таблиця 6.1

Операції технологічного процесу та час їх виконання

№ п/п	Назва операції	Виконавець	Середній час виконання, год.
1	Аналіз поставленого завдання	інженер	6
2	Збір необхідної інформації	інженер	6
3	Аналіз криптографічних засобів	інженер	20
4	Розробка алгоритму крипто аналізу KASUMI	інженер	48
5	Реалізація алгоритму криптоаналізу KASUMI	інженер	20
6	Проведення тестувань	інженер	6
7	Підготовка технічної документації	інженер	10
Разом			106

6.2. Визначення накладних витрат та на оплату праці

В Законі України «Про оплату праці» сказано, що заробітна плата – це винагорода, обчислена, як правило, у грошовому виразі, яку за трудовим договором власник або уповноважений ним орган виплачує працівникові за виконану ним роботу.

Розмір заробітної плати залежить від складності та умов виконуваної роботи, професійно-ділових якостей працівника, результатів його праці та господарської діяльності підприємства.

Заробітна плата складається з двох частин: основна заробітна плата, додаткова заробітна плата й інші заохочувальні та компенсаційні виплати.

Основна заробітна плата – винагорода за виконану роботу відповідно до встановлених норм праці (норми часу, виробітку, обслуговування, посадові обов'язки). Вона встановлюється у вигляді тарифних ставок (окладів) і відрядних розцінок для робітників та посадових окладів для службовців.

Додаткова заробітна плата – винагорода за працю понад установлені норми, за трудові успіхи та винахідливість і за особливі умови праці. Вона включає доплати, надбавки, гарантійні і компенсаційні виплати, передбачені чинним законодавством; премії, пов'язані з виконанням виробничих завдань і функцій.

Інші заохочувальні та компенсаційні виплати. До них належать виплати у формі винагород за підсумками роботи за рік, премії за спеціальними системами і положеннями, компенсаційні та інші грошові і матеріальні виплати, які не передбачені актами чинного законодавства, або які провадяться понад встановлені зазначеними актами норми.

Для обчислення основної заробітної плати використаємо формулу:

$$З_{осн} = Т_с \cdot К_г \, , \qquad (6.1)$$

де $Т_с$ – тарифна ставка, грн.;

$К_г$ – кількість відпрацьованих годин.

Оскільки всі види робіт виконує інженер, тарифна ставка якого рівна 20,0 грн./год., то $З_{осн} = 20 \cdot 106 = 2120$ грн.

Додаткова заробітна плата становить 10–15 % від суми основної заробітної плати і обчислюється за формулою:

$$З_{дод} = З_{осн} \cdot К_{допл}\ , \qquad (6.2)$$

де $К_{допл}$ – коефіцієнт додаткових виплат працівникам.

Нехай коефіцієнт додаткових виплат рівний 0,15, тоді додаткова заробітна плата $З_{дод} = 2120 \cdot 0{,}15 = 318$ грн.

Загальні витрати на оплату праці визначаються за формулою:

$$В_{о.п.} = З_{осн} + З_{дод}\ . \qquad (6.3)$$

Отже, $В_{о.п.} = 2120 + 318 = 2438$ грн.

Крім того, слід визначити відрахування на соціальні заходи. Діяльність у сфері інформаційних технологій і комп’ютерних систем відноситься до 2 класу професійного ризику виробництва. Згідно з Законом України «Про збір та облік єдиного внеску на загальнообов'язкове державне соціальне страхування» розмір відрахувань становитиме 36,77 %. Отже, сума відрахувань на соціальні заходи буде становити:

$$В_{с.з.} = В_{о.п.} \cdot 0{,}3677\ . \qquad (6.4)$$

Отже, $В_{с.з.} = 2438 \cdot 0{,}3677 = 896{,}45$ грн.

Загальні витрати на оплату праці обчислюються за формулою:

$$В_з = В_{о.п.} + В_{с.з.}\ . \qquad (6.5)$$

Отже, $В_з = 2438 + 896,45 = 3334,45$ грн.

Проведені обчислення витрат на оплату праці зведені в таблицю 6.2.

Таблиця 6.2

Розрахунки витрат на оплату праці

№ п/п	Категорія працівника	Основна заробітна плата, грн.	Додаткова заробітна плата, грн.	Соціальні заходи, грн.	Загальні витрати на оплату праці, грн.
1	Інженер	2120	318	896,45	3334,45

Накладні витрати пов’язані з обслуговуванням виробництва, утриманням апарату управління спілкою та створення необхідних умов праці.

В залежності від організаційно-правової форми діяльності господарюючого суб’єкта, накладні витрати можуть становити 20–60 % від суми основної та додаткової заробітної плати працівників.

Накладні витрати обчислюються за формулою:

$$Н_в = В_{о.п.} \cdot 0,2 \ . \qquad (6.6)$$

Отже, $Н_в = 2438 \cdot 0,2 = 487,6$ грн.

6.3. Розрахунок витрат на електроенергію

Затрати на електроенергію для одиниці обладнання визначаються за формулою:

$$З_в = W \cdot T \cdot S , \quad (6.7)$$

де W – це споживана потужність, кВт;

T – час роботи обладнання, год.;

S – вартість кіловат-години, грн.

Згідно існуючих тарифів вартість однієї кіловат-години юридичних осіб, а саме непромислових споживачів, рівна 0,9733 грн. Потужність комп'ютерної системи згідно технічного завдання становить 900 Вт. Час роботи згідно даних таблиці 3.1 – 106 годин.

Тоді, $З_в = 0,9 \cdot 106 \cdot 0,9733 = 92,85$ грн.

6.4. Розрахунок матеріальних витрат та суми амортизаційних відрахувань

Загальні матеріальні витрати визначаються як добуток кількості витрачених матеріалів та їх ціни:

$$З_{м.в.} = \sum (q_i \cdot p_i) , \quad (6.8)$$

де q_i – кількість витраченого матеріалу і-го виду;

p_i – ціна матеріалу і-го виду.

Обчислення загальних матеріальних витрат показані в таблиці 6.3.

Таблиця 6.3

Матеріальні витрати

№ п/п	Найменування	Кількість	Ціна, грн	Вартість, грн
1	Комп'ютерна система, визначена технічним завданням	1	8000	8000

Отже, загальна вартість рівна 8000 грн.

Амортизація – це процес перенесення вартості основних фондів на вартість новоствореної продукції з метою їх повного відновлення.

Комп'ютери та оргтехніка належать до четвертої групи основних фондів. Для цієї групи річна норма амортизації дорівнює 60 % (квартальна – 15 %). Амортизаційні відрахування визначаються за формулою:

$$А = Б_в \cdot Н_а , \qquad (6.9)$$

де $Б_в$ – балансова вартість групи основних фондів на початок звітного періоду, грн.;

$Н_а$ – норма амортизації.

В даному випадку до групи основних фондів належить комп'ютерна система, вартість якої визначена у технічному завданні і становить 4000 грн. Тоді, амортизаційні відрахування $А = 8000 \cdot 0,6 = 4800$ грн.

6.5. Складання кошторису витрат та визначення собівартості

Собівартість продукції — це виражені в грошовій формі сукупні витрати на підготовку і випуск продукції. В більш узагальненому вигляді собівартість можна визначити як грошовий вираз величини ресурсів, використаних з конкретною метою. Таке визначення собівартості містить у собі три важливих моменти:

- собівартість відображає, скільки і яких ресурсів було використано у виробництві;
- величина використаних ресурсів представлена в грошовому виразі, що дозволяє розраховувати загальну вартість ресурсів;
- конкретна мета використання ресурсів зумовлює необхідність чітко встановити об'єкт собівартості (виробництво чи реалізація).

Собівартість розробки визначається за формулою:

$$С_в = В_{о.п.} + В_{с.з.} + З_в + Н_в + З_{м.в.} + А \ . \qquad (6.10)$$

Отже, собівартість $С_в$ = 2438 + 896,45 + 91,85 + 487,6 + 8000 + 4800 = 16713,9 грн.

Результати проведених вище обчислень показані в таблиці 6.4.

Таблиця 6.4

Кошторис витрат

№ п/п	Зміст витрат	Сума, грн.	Частка від загальної суми, %
1	Витрати на оплату праці	2438	14,58
2	Відрахування на соціальні заходи	896,45	5,36
3	Витрати на електроенергію	91,85	0,55
4	Накладні витрати	487,6	2,92
5	Матеріальні витрати	8000	47,86
6	Амортизаційні витрати	4800	28,71
Собівартість		16713,9	100

Отже, собівартість даної розробки становить 16713,9 грн.

6.6. Розрахунок ціни програмного продукту

Створений програмний продукт використовується в межах проведення кафедральних науково-дослідних робіт «Створення грід-орієнтованого програмного забезпечення для здійснення криптоаналізу» та «Моделювання та розробка алгоритмів криптоаналізу з використанням паралельних та розподілених комп'ютерних систем», тоді доцільно при визначенні ціни приймати те, що програмний продукт виконаний і реалізовується в кількості одного екземпляра. Тоді ціна розробки визначається за формулою:

$$Ц = C_B \cdot (1 + P_{рен}) \cdot (1 + ПДВ), \quad (6.11)$$

де Ц – ціна програмного продукту;

$Р_{рен}$ – рівень рентабельності, 30%;

ПДВ – податок на додану вартість, 20%.

Отже, Ц = 16713,9 · (1 + 0,3) · (1 + 0,2) = 26073,7 грн.

6.7. Визначення економічної ефективності і терміну окупності капітальних вкладень

Ефективність виробництва – це узагальнене і повне відображення кінцевих результатів використання робочої сили, засобів та предметів праці на підприємстві за певний проміжок часу.

Визначення планового прибутку виконується за наступною формулою:

$$П_{пл} = Ц - С_В \quad (6.12)$$

Тоді, $П_{пл}$ = 26073,7 - 16713,9 = 9359,8 грн.

Економічна ефективність ($Е_р$) полягає у відношенні результату виробництва до затрачених ресурсів:

$$E_р = П_{пл} / C_В . \quad (6.13)$$

Отже, $E_р = 9359,8 / 16713,9 = 0,56$.

Термін окупності капітальних вкладень визначається за формулою:

$$T_Р = 1 / E_р . \quad (6.14)$$

Тоді, $T_Р = 1 / 0,56 = 1,8$ років.

В таблиці 7.5 представлені основні техніко-економічні показники.

Таблиця 6.5

Техніко-економічні показники

№ п/п	Показник	Значення
1.	Собівартість, грн.	16713,9
2.	Плановий прибуток, грн..	9359,8
3.	Ціна, грн.	26073,7
4.	Економічна ефективність	0,56
5.	Термін окупності, рік	1,8

Підсумувавши всі обчислення, можна зробити висновок, що розробка вважається доцільною і економічно вигідною за всіма техніко-економічними показниками.

6.8. Висновки до розділу

Проведено обчислення основних техніко-економічних показників економічної ефективності. Сумарний час виконання вміх операцій технологічного процесу склав 106 годин. Загальні витрати на оплату праці

становлять 3334,45 грн., накладні витрати – 487,6 грн., витрати на електроенергію – 92,85 грн., матеріальні витрати – 8000 грн, амортизаційні відрахування – 4800 грн. Ціна розробки програмного продукту рівна 26073,7 грн. Плановий прибуток очікується в розмірі 9359,8 грн. Економічна ефективність становить 0,56, термін окупності – 1,8 років.

Отже, можна зробити висновок, що розробка вважається доцільною і економічно вигідною за всіма техніко-економічними показниками.

ВИСНОВКИ

З розвитком інформаційних технологій постає важлива задача– захист інформаційних систем. Для вирішення цього питання використовуються криптографічні засоби, які забезпечують як конфіденційність, так і цілісність даних. Тому важливим є оцінювання рівня захисту даних систем, тобто застосування криптоаналітичних методів для виявлення вразливостей криптографічних засобів.

В результаті виконання магістерської дипломної роботи розглянуто особливості функціонування криптографічних засобів захисту мережі UMTS, зокрема алгоритмів забезпечення конфіденційності та цілісності даних f8 та f9, які базуються на алгоритмі шифрування KASUMI, що дало змогу визначити методи для проведення дослідження захищеності даних засобів.

Досліджено особливості сендвіч атаки з пов’язаними ключами та спроектовано алгоритм криптоаналізу. Розглянутий метод криптоаналізу алгоритму шифрування KASUMI має складність даних 2^{24} відкритих текстів та 2^{24} шифротектсів. Часова складність рівна 2^{32}, оскільки найбільш ємнісним за часом є перебір останніх 32 біт ключа.

Обгрунтовано технологію програмування на мові С та засоби розпаралелення – технологію MPI, що дає змогу зменшити час виконання програми шляхом її паралельного виконання.

Реалізовано програму криптоаналізу алгоритму шифрування KASUMI. Здійснено тестування: послідовне виконання програми тривало 341 с., паралельне виконання з чотирма процесами – 205с. Проведено профілювання програми, що дало змогу проаналізувати режими міжпроцесної комунікації, а також визначити стан використання системних ресурсів процесами та методами програми.

Дані дослідження підтверджують високий рівень захисту алгоритмів UEA1 та UIA1, що використовують KASUMI як генератор потокового ключа.

Дипломну роботу виконано в рамках таких науково-дослідних робіт: ДФ 196-12 «Створення грід-орієнтованого програмного забезпечення для здійснення криптоаналізу» та ВК 34-11 «Моделювання та розробка алгоритмів криптоаналізу з використанням паралельних та розподілених комп'ютерних систем».

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. 3GPP TS 35.201: «3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Specification of the 3GPP Confidentiality and Integrity Algorithms; Document 1: f8 and f9 Specification» [Електронний ресурс]. – Режим доступу: http://www.3gpp.org/DynaReport/35201.htm – Назва з екрану.

2. V. Niemi. UMTS Security / V. Niemi, K. Nyberg.. – Finland, Helsinki: John Wiley & Sons, 2005. – 273 с.

3. 3GPP TS 33.220: «3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Generic Authentication Architecture (GAA); Generic Bootstrapping Architecture (GBA)» [Електронний ресурс]. – Режим доступу: http://www.3gpp.org/DynaReport/33220.htm – Назва з екрану.

4. 3GPP TS 35.205: «3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Specification of the MILENAGE Algorithm Set: An example algorithm set for the 3GPP authentication and key generation functions f1, f1*, f2, f3, f4, f5 and f5*; Document 1: General» [Електронний ресурс]. – Режим доступу: http://www.3gpp.org/DynaReport/35205.htm – Назва з екрану.

5. 3GPP TS 35.202: «3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G Security; Specification of the 3GPP Confidentiality and Integrity Algorithms; Document 2: KASUMI Specification» [Електронний ресурс]. – Режим доступу: http://www.3gpp.org/DynaReport/35202.htm – Назва з екрану.

6. 3GPP TS 35.216: «3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Specification of the 3GPP Confidentiality and Integrity Algorithms UEA2 & UIA2; Document 2: SNOW 3G specification» [Електронний ресурс]. – Режим доступу: http://www.3gpp.org/DynaReport/35216.htm – Назва з екрану.

7. С. В. Кавун. Інформаційна безпека / С. В. Кавун, В. В. Носов, О. В. Манжай. – Харків: Вид. ХНЕУ, 2008. – 196 с.

8. Orr Dunkelman, Nathan Keller, Adi Shamir (2010-01-10). A Practical-Time Attack on the A5/3 Cryptosystem Used in Third Generation GSM Telephony, Weizmann Institute of Science 10 January 2010 [Електронний ресурс]. – Режим доступу: http://eprint.iacr.org/2010/013.pdf – Назва з екрану.

9. David Wagner. The Boomerang Attack / David Wagner // Lecture Notes in Computer Science. – 1999. – №1636. – P. 156–170.

10. Alex Biryukov. Cryptanalysis of SAFER++ / Alex Biryukov, Christophe De Canni`ere, Gustaf Dellkrantz // Lecture Notes in Computer Science. – 2003. – №2729. – P. 195–211.

11. Alex Biryukov. Related-key Cryptanalysis of the Full AES-192 and AES-256 / Alex Biryukov, Dmitry Khovratovich // Lecture Notes in Computer Science. – 2009. – №5912. – P. 1–18.

12. Jongsung Kim. The Related-Key Rectangle Attack — Application to SHACAL-1 / Jongsung Kim, Guil Kim, Seokhie Hong, Dowon Hong // Lecture Notes in Computer Science. – 2004. – №3108. – P. 123–136.

13. Seokhie Hong. Related-Key Rectangle Attacks on Reduced Versions of SHACAL-1 and AES-192 / Seokhie Hong, Jongsung Kim, Guil Kim, Sangjin Lee, Bart Preneel // Lecture Notes in Computer Science. – 2005. – №3557. – P. 368–383.

14. Eli Biham. Related-Key Boomerang and Rectangle Attacks / Eli Biham, Orr Dunkelman, Nathan Keller // Lecture Notes in Computer Science. – 2005. – №3494. – P. 507–525.

15. John Kelsey. Key Schedule Cryptanalysis of IDEA, G-DES, GOST, SAFER, and Triple-DES / John Kelsey, Bruce Schneier, David Wagner // Lecture Notes in Computer Science. – 1996. – №1109. – P. 237–251.

16. Mark Blunden. Related Key Attacks on Reduced Round KASUMI / Mark Blunden, Adrian Escott // Lecture Notes in Computer Science. – 2002. – №2355. – P. 277–285.

17. Eli Biham. A Related-Key Rectangle Attack on the Full KASUMI / Eli Biham, Orr Dunkelman, Nathan Keller // Lecture Notes in Computer Science. – 2005. – №3788. – P. 443–461.

18. Encrypt the given plain text with the given key using the KASUMI block cipher [Електронний ресурс]. – Режим доступу:URL: http://people.rit.edu/aar3301/Kasumi.java – Назва з екрану.

19. Справочник по проектированию электрических сетей / под ред. Д. Л. Файбисовича. – 4-е изд., перераб. и доп. – М. : ЭНАС, 2012. – 376 с.

20. Тарасова, В.В. Екологічна статистика. / В.В. Тарасова. – К.: «Центр учбової літератури», 2008. – 391 с.

21. Клименко, М.О. Моніторинг довкілля. / М.О. Клименко, А.М. Прищепа, Н.М. Вознюк. – К.: Видавничий центр «Академія», 2006. – 360 с.

АНОТАЦІЯ

Кондрацький Ю.О. Дослідження захищеності криптографічних засобів захисту мережі передачі даних UMTS.

Метою дослідження є аналіз криптографічних засобів захисту інформації в UMTS мережах, а саме дослідження криптостійкості алгоритму KASUMI та реалізація програми криптоаналізу.

Предмет дослідження – криптостійкість алгоритму шифрування KASUMI, що використовується в алгоритмах UEA1 і UIA1.

Об’єкт дослідження – криптографічні засоби захисту в UMTS мережах.

Дипломна робота спрямована на дослідження криптостійкості криптографічних засобів в мережі UMTS. Проаналізовано сендвіч атаку з пов’язаними ключами на алгоритм KASUMI. Спроектовано програмне забезпечення для криптоаналізу. Здійснено оптимізацію, використовуючи технологію MPI. Проведено тестування програми та профілювання.

Рік виконання – 2013.

Рік захисту – 2013.

Ключові слова: KASUMI, A5/3, сендвіч атака, UMTS, блоковий шифр, MPI, криптоаналіз.

Дипломна робота містить 114 сторінки, 14 таблиць, 23 рисунків, список літератури 21 найменувань, 2 додатки.

ABSTRACT

Kondratskyi Y.O. Research protection of cryptographic tools in UMTS data network.

The aim of the research is to analyze the cryptographic protection of information in UMTS networks, namely the research of cryptographic algorithm KASUMI and implementing a program of cryptanalysis .

Purpose of the research – cryptographic protection of encryption algorithm KASUMI, which is used in the algorithms and UEA1 UIA1.

Object of research – cryptographic protection in UMTS networks.

Research paper aims to analyze the reliability of cryptographic tools in network UMTS. Analyzed sandwich attack with related key on the algorithm KASUMI. Designed software for cryptanalysis . Done optimization using MPI technology. A program testing and profiling.

Release year – 2013.

Defend year – 2013.

Keywords : KASUMI, A5 / 3, the sandwich attack , UMTS, block cipher , MPI, cryptanalysis.

Thesis contains 114 pages, 14 tables , 23 figures , 21 references, 2 annexes.

ДОДАТКИ

Додаток А

Лістинг класу Kasumi

```
class Kasumi
{
	// Instance Variables

	// Constants used to XOR with keyJ in order to get keyJPrime {C1,...,C8}
private:
	int KEYCONSTANTS[8];
	//Message to be encrypted
	long long plaintext;
	// Keys variables K and K'
	int keyJ[8];
	int keyJPrime[8];

	// Sub-keys variables KL, KO and KI
public:
	int subkeyL[8][2];
	int subkeyO[8][3];
	int subkeyI[8][3];

	Kasumi(long long keyLeft, long long keyRight)
	{

		// Variable used to select chunks of 16-bits from K1 to K8
		InitializeInstanceFields();
		long long keySlot = 0xffff000000000000L;
		int shift = 48; // Used to move between slots
		for (int i = 0; i < 4; i++)
		{
			keyJ[i] = static_cast<int>((keyLeft & keySlot) >> shift);
			keyJ[i + 4] = static_cast<int>((keyRight & keySlot) >>
shift);
			keySlot = static_cast<long long>(static_cast<unsigned long
long>(keySlot) >> 16); // Move to the next key slot
			shift -= 16;
		} // End for-loop
		//Run KeySchedule to generate the sub-keys.
		keySchedule();
	}

	void keySchedule()
	{
		// Create second array of subkeys by XOR with KEYCONSTANTS
		for (int i = 0; i < 8; i++)
		{
			keyJPrime[i] = keyJ[i] ^ KEYCONSTANTS[i];
		}
		// Generate function round keys
		// See Table 1:Round subkeys on page 14 of KASUMI 3GPP spec
		for (int i = 0; i < 8; i++)
		{
			subkeyL[i][0] = (keyJ[i] << 1 |
static_cast<int>(static_cast<unsigned int>(keyJ[i]) >> 15)) & 0xFFFF;
			subkeyL[i][1] = keyJPrime[(i + 2) % 8];
```

```
                  subkeyO[i][0] = (keyJ[(i + 1) % 8] << 5 |
static_cast<int>(static_cast<unsigned int>(keyJ[(i + 1) % 8]) >> 11)) &
0xFFFF;
                  subkeyO[i][1] = ((keyJ[(i + 5) % 8] << 8 |
static_cast<int>(static_cast<unsigned int>(keyJ[(i + 5) % 8]) >> 8))) &
0xFFFF;
                  subkeyO[i][2] = ((keyJ[(i + 6) % 8] << 13 |
static_cast<int>(static_cast<unsigned int>(keyJ[(i + 6) % 8]) >> 3))) &
0xFFFF;

                  subkeyI[i][0] = keyJPrime[(i + 4) % 8];
                  subkeyI[i][1] = keyJPrime[(i + 3) % 8];
                  subkeyI[i][2] = keyJPrime[(i + 7) % 8];
            } //end of for-loop
      } // end of KeySchedule()

      long long encrypt(long long plaintext, int rounds)
      {
            this->plaintext = plaintext;
            int left = static_cast<int>(static_cast<long
long>(static_cast<unsigned long long>(plaintext) >> 32));
            int right= static_cast<int>(plaintext & 0xFFFFFFFF);
            int prevRight;
            for(int i = 1; i <= rounds; i++)
            {
                  prevRight = right;
                  right = left;
                  if(i % 2 != 0)
                  {
                        left = prevRight ^ FO(FL(left, i - 1), i - 1);
                  }
                  else
                  {
                        left = prevRight ^ FL(FO(left, i - 1), i - 1);
                  }
            }
            return static_cast<long long>(left) << 32 | (static_cast<long
long>(right) & 0x00000000FFFFFFFFL);
      }

      long long decrypt(long long ciphertext, int rounds)
      {
            int left = static_cast<int>(static_cast<long
long>(static_cast<unsigned long long>(ciphertext) >> 32));
            int right= static_cast<int>(ciphertext & 0xFFFFFFFF);
            int prevLeft;
            for(int i = rounds; i >= 1; i--)
            {
                  prevLeft = left;
                  left = right;
                  if(i % 2 != 0)
                  {
                        right = prevLeft ^ FO(FL(right, i - 1), i - 1);
                  }
                  else
                  {
                        right = prevLeft ^ FL(FO(right, i - 1), i - 1);
                  }
            }
            return static_cast<long long>(left) << 32 | (static_cast<long
long>(right) & 0x00000000FFFFFFFFL);
      }
```

```
	int FL(int in32Bits, int round)
	{
		// Local variables
		int L0, L1, R0, R1;
		// Split the 32-Bit input I in 2 half, 16-bits each
		R0 = in32Bits & 0xffff;
		L0 = (static_cast<int>(static_cast<unsigned int>(in32Bits) >> 16))
& 0xffff;
		// Running Function FL as described on page 10 of KASUMI 3GPP spec
		R1 = R0 ^ (((L0 & subkeyL[round][0]) << 1 |
static_cast<int>(static_cast<unsigned int>((L0 & subkeyL[round][0])) >> 15))
& 0xFFFF);
		L1 = L0 ^ (((R1 | subkeyL[round][1]) << 1 |
static_cast<int>(static_cast<unsigned int>((R1 | subkeyL[round][1])) >> 15))
& 0xFFFF);
		// Merge results for output
		int out32Bits = (L1 << 16) | R1;
		// return out16Bits;
		return out32Bits;
	} // end of FL(int, int)

	int FO(int in32Bits, int round)
	{
		// Local variables
		int L0, L1, L2, L3, R0, R1, R2, R3;
		// Split the 32-Bit input I in 2 half, 16-bits each
		R0 = in32Bits & 0xffff;
		L0 = (static_cast<int>(static_cast<unsigned int>(in32Bits) >> 16 &
0xffff));
		// Running Function FO as described on page 11 of KASUMI 3GPP spec
		R1 = FI((L0 ^ subkeyO[round][0]), subkeyI[round][0]) ^ R0;
		L1 = R0;
		R2 = FI((L1 ^ subkeyO[round][1]), subkeyI[round][1]) ^ R1;
		L2 = R1;
		R3 = FI((L2 ^ subkeyO[round][2]), subkeyI[round][2]) ^ R2;
		L3 = R2;
		// Merge results for output
		int out32Bits = (L3 << 16) | R3;
		//return out16Bits;
		return out32Bits;
	} // end of FO(int, long, long)

private:
	int ZE(int in7Bits)
	{
		int out9Bits = in7Bits & 0x1FF;
		return out9Bits;
	} // end of ZE(int)
	int TR(int in9Bits)
	{
		int out7Bits = in9Bits & 0x7F;
		return out7Bits;
	} // end of TR(int)

public:
	int FI(int in16Bits, int subkey)
	{
		// Local variables
		int L0, L1, L2, L3, L4, R0, R1, R2, R3, R4, key9Bits, key7Bits;
		// Split the input I in 2 half, 9-bit left and 7-bit right
		R0 = in16Bits & 0x7f;
		L0 = (static_cast<int>(static_cast<unsigned int>(in16Bits) >> 7 &
0x1FF));
```

```
            // Split the key in 2 half, 7-bit left and 9-bit right
            key9Bits = subkey & 0x1FF;
            key7Bits = (static_cast<int>(static_cast<unsigned int>(subkey) >>
9 & 0x7f));
            // Running Function FI as described on page 11 of KASUMI 3GPP spec
            L1 = R0;
            R1 = S9(L0) ^ ZE(R0);
            L2 = R1 ^ key9Bits;
            R2 = S7(L1) ^ TR(R1) ^ key7Bits;
            L3 = R2;
            R3 = S9(L2) ^ ZE(R2);
            L4 = S7(L3) ^ TR(R3);
            R4 = R3;
            // Merge results for output
            int out16Bits = (L4 << 9) | R4;
            return out16Bits;
      } // end of FI(int, int)

private:
      int S7(int in7Bits)
      {
            // Extract bits from input to be used by S7 Block
            int x0 = (in7Bits & 1), x1 = (in7Bits >> 1) & 1, x2 = (in7Bits >>
2) & 1, x3 = (in7Bits >> 3) & 1, x4 = (in7Bits >> 4) & 1, x5 = (in7Bits >> 5)
& 1, x6 = (in7Bits >> 6) & 1;
            // S7 Block Gate Logic
            // See 4.5.1: Gate Logic on page 12 of KASUMI 3GPP spec
            int y0 = (x1 & x3 ^ x4 ^ x0 & x1 & x4 ^ x5 ^ x2 & x5 ^ x3 & x4 &
x5 ^ x6 ^ x0 & x6 ^ x1 & x6 ^ x3 & x6 ^ x2 & x4 & x6 ^ x1 & x5 & x6 ^ x4 & x5
& x6);
            int y1 = (x0 & x1 ^ x0 & x4 ^ x2 & x4 ^ x5 ^ x1 & x2 & x5 ^ x3 &
x5 ^ x6 ^ x0 & x2 & x6 ^ x3 & x6 ^ x4 & x5 & x6 ^ 1);
            int y2 = (x0 ^ x0 & x3 ^ x2 & x3 ^ x1 & x2 & x4 ^ x0 & x3 & x4 ^
x1 & x5 ^ x0 & x2 & x5 ^ x0 & x6 ^ x0 & x1 & x6 ^ x2 & x6 ^ x4 & x6 ^ 1);
            int y3 = (x1 ^ x0 & x1 & x2 ^ x1 & x4 ^ x3 & x4 ^ x0 & x5 ^ x0 &
x1 & x5 ^ x2 & x3 & x5 ^ x1 & x4 & x5 ^ x2 & x6 ^ x1 & x3 & x6);
            int y4 = (x0 & x2 ^ x3 ^ x1 & x3 ^ x1 & x4 ^ x0 & x1 & x4 ^ x2 &
x3 & x4 ^ x0 & x5 ^ x1 & x3 & x5 ^ x0 & x4 & x5 ^ x1 & x6 ^ x3 & x6 ^ x0 & x3
& x6 ^ x5 & x6 ^ 1);
            int y5 = (x2 ^ x0 & x2 ^ x0 & x3 ^ x1 & x2 & x3 ^ x0 & x2 & x4 ^
x0 & x5 ^ x2 & x5 ^ x4 & x5 ^ x1 & x6 ^ x1 & x2 & x6 ^ x0 & x3 & x6 ^ x3 & x4
& x6 ^ x2 & x5 & x6 ^ 1);
            int y6 = (x1 & x2 ^ x0 & x1 & x3 ^ x0 & x4 ^ x1 & x5 ^ x3 & x5 ^
x6 ^ x0 & x1 & x6 ^ x2 & x3 & x6 ^ x1 & x4 & x6 ^ x0 & x5 & x6);
            // Merge results for output
            int out7Bits = (y6 << 6 | y5 << 5 | y4 << 4 | y3 << 3 | y2 << 2 |
y1 << 1 | y0);
            return out7Bits;
      } // end of S7(int)

      int S9(int in9Bits)
      {
            // Extract bits from input to be used by S9 Block
            int x0 = (in9Bits & 1), x1 = (in9Bits >> 1) & 1, x2 = (in9Bits >>
2) & 1, x3 = (in9Bits >> 3) & 1, x4 = (in9Bits >> 4) & 1, x5 = (in9Bits >> 5)
& 1, x6 = (in9Bits >> 6) & 1, x7 = (in9Bits >> 7) & 1, x8 = (in9Bits >> 8) &
1;
            // S9 Block Gate Logic
            // See 4.5.2: Gate Logic on page 13 of KASUMI 3GPP spec
            int y0 = (x0 & x2 ^ x3 ^ x2 & x5 ^ x5 & x6 ^ x0 & x7 ^ x1 & x7 ^
x2 & x7 ^ x4 & x8 ^ x5 & x8 ^ x7 & x8 ^ 1);
            int y1 = (x1 ^ x0 & x1 ^ x2 & x3 ^ x0 & x4 ^ x1 & x4 ^ x0 & x5 ^
x3 & x5 ^ x6 ^ x1 & x7 ^ x2 & x7 ^ x5 & x8 ^ 1);
```

```
            int y2 = (x1 ^ x0 & x3 ^ x3 & x4 ^ x0 & x5 ^ x2 & x6 ^ x3 & x6 ^
x5 & x6 ^ x4 & x7 ^ x5 & x7 ^ x6 & x7 ^ x8 ^ x0 & x8 ^ 1);
            int y3 = (x0 ^ x1 & x2 ^ x0 & x3 ^ x2 & x4 ^ x5 ^ x0 & x6 ^ x1 &
x6 ^ x4 & x7 ^ x0 & x8 ^ x1 & x8 ^ x7 & x8);
            int y4 = (x0 & x1 ^ x1 & x3 ^ x4 ^ x0 & x5 ^ x3 & x6 ^ x0 & x7 ^
x6 & x7 ^ x1 & x8 ^ x2 & x8 ^ x3 & x8);
            int y5 = (x2 ^ x1 & x4 ^ x4 & x5 ^ x0 & x6 ^ x1 & x6 ^ x3 & x7 ^
x4 & x7 ^ x6 & x7 ^ x5 & x8 ^ x6 & x8 ^ x7 & x8 ^ 1);
            int y6 = (x0 ^ x2 & x3 ^ x1 & x5 ^ x2 & x5 ^ x4 & x5 ^ x3 & x6 ^
x4 & x6 ^ x5 & x6 ^ x7 ^ x1 & x8 ^ x3 & x8 ^ x5 & x8 ^ x7 & x8);
            int y7 = (x0 & x1 ^ x0 & x2 ^ x1 & x2 ^ x3 ^ x0 & x3 ^ x2 & x3 ^
x4 & x5 ^ x2 & x6 ^ x3 & x6 ^ x2 & x7 ^ x5 & x7 ^ x8 ^ 1);
            int y8 = (x0 & x1 ^ x2 ^ x1 & x2 ^ x3 & x4 ^ x1 & x5 ^ x2 & x5 ^
x1 & x6 ^ x4 & x6 ^ x7 ^ x2 & x8 ^ x3 & x8);
            // Merge results for output
            int out9Bits = (y8 << 8 | y7 << 7 | y6 << 6 | y5 << 5 | y4 << 4 |
y3 << 3 | y2 << 2 | y1 << 1 | y0);
            return out9Bits;
      } // end of S9(int)

private:
      void InitializeInstanceFields()
      {
            KEYCONSTANTS[0] = 0x0123;
            KEYCONSTANTS[1] = 0x4567;
            KEYCONSTANTS[2] = 0x89ab;
            KEYCONSTANTS[3] = 0xcdef;
            KEYCONSTANTS[4] = 0xfedc;
            KEYCONSTANTS[5] = 0xba98;
            KEYCONSTANTS[6] = 0x7654;
            KEYCONSTANTS[7] = 0x3210;
      } // end of InitializeInstanceFields()

}; // end of kasumi Class
```

Додаток Б

Лістинг програми криптоаналізу

```
#include <string>
#include <stdio.h>
#include <map>
#include <stdlib.h>
#include <time.h>
#include <algorithm>
#include <vector>
#include "mpi.h"
using namespace std;
class DataCollect
{
public: long long *valCa = new long long[16777216];
      long long *valCb = new long long[16777216];
      long long *valCc = new long long[16777216];
      long long *valCd = new long long[16777216];
      long long aConst;
      typedef pair <long long, long long> longpair;
      typedef pair <int, longpair> doublepair;
      typedef pair <longpair, longpair> triplepair;
      typedef pair <int, triplepair> quadropair;
      longpair pairCaCb;
      longpair pairCcCd;
      doublepair tmpPair;
      triplepair tmp2Pair;
      quadropair tmp3Pair;
      multimap <int, longpair> mmpOne;
      multimap <int, triplepair> mmpTwo;
      long long aMas[70000][5];
      long sMas[70000];
      long rMas[70000];
      int  numtasks, rank, rc, source, dest;
      int tag;
      int c;
      long index;
      int s;
      DataCollect()
      {
            aConst = 0x00000000ffffffffL;
            InitInValue();
      }
      void Dest()
      {
            delete[] valCa;
            delete[] valCb;
            delete[] valCc;
            delete[] valCd;
      }
      void InitInValue()
      {
            long i;
            c=0;
            //srand(time(NULL));
            for(i=0; i<16777216; i++)
            {
                  valCa[i] = LongGen() | aConst;
```

```
                valCc[i] = LongGen() | (aConst ^ 0x0000000000100000L);

        }
    }
    long long LongGen()
    {
        int q;
        long long bac = 0x0;
        for (q=0; q<2; q++)
        {
            bac = (bac | static_cast<long long>(rand() % 0xFFFF));
            if (q != 1)
            bac = bac << 16;
        }
        bac = bac << 32;
        return bac;
    }
    void PairCaCb(long long valOne, long long valTwo)
    {
        pairCaCb = make_pair(valOne, valTwo);
        tmpPair = make_pair(static_cast<int>(valTwo), pairCaCb);
        mmpOne.insert(tmpPair);
    }
    void EqualsCheck()
    {
        long i;
        multimap <int, longpair> :: iterator iter;
        pair <multimap <int,longpair>::iterator, multimap <int,longpair>
:: iterator> ret;
        for(i=0; i<16777216; i++)
        {
            ret = mmpOne.equal_range(static_cast<int>(valCd[i] ^
0x0000000000100000L));
            for (iter=ret.first; iter!=ret.second; ++iter)
            {
                pairCaCb = iter->second;
                sMas[c] = static_cast<int>(pairCaCb.first>>32) ^
static_cast<int>(valCc[i]>>32);
                aMas[c][0] = sMas[c];
                aMas[c][1] = pairCaCb.first;
                aMas[c][2] = pairCaCb.second;
                aMas[c][3] = valCa[i];
                aMas[c][4] = valCb[i];
                c=c++;
            }
        }
        printf("quartets = %d\n", c);
    }
    void Stest()
    {
        vector <long> tVector;
        vector <long> fRes;
        tVector.assign(sMas, sMas+c);
        stable_sort(tVector.begin(), tVector.end());
        for (vector <long>::iterator
it=tVector.begin();it!=(tVector.end()-3);++it)
        {
            if (*it==*(it++) && *it==*(it+2))
            {
                fRes.push_back(*it);
            }
        }
        vector <long>::iterator itsec;
```

```
            itsec = unique(fRes.begin(), fRes.end());
            fRes.resize(distance(fRes.begin(), itsec));
            for (vector <long>::iterator
itth=fRes.begin();itth!=fRes.end();++itth)
            {
                  index = *itth;
                  printf("index = %8lx\n", index);
            }
            for (s=0; s<c; s++)
            {
                  if (aMas[s][0] == index)
                  {
                        printf("Ca = %16llx\n", aMas[s][1]);
                        printf("Cb = %16llx\n", aMas[s][2]);
                        printf("Cc = %16llx\n", aMas[s][3]);
                        printf("Cd = %16llx\n", aMas[s][4]);
                  }
            }
      }

};

int main(int argc,char* argv[])
{
      int numtasks;
      int rank;

      MPI_Init(&argc, &argv);
      MPI_Comm_size(MPI_COMM_WORLD,&numtasks);
      MPI_Comm_rank(MPI_COMM_WORLD,&rank);

      int ost = 16777216 % numtasks;
      int div = 16777216 / numtasks;

      long long *bufCa = new long long [div];
      long long *bufCb = new long long [div];
      long long *bufCc = new long long [div];
      long long *bufCd = new long long [div];


      long long keyLeft = 0x0000000000000000L;
      long long keyRight = 0x0000000000000000L;
      long long akeyLeft = keyLeft;
      long long akeyRight = keyRight;
      long long bkeyLeft = akeyLeft ^ 0x0000000080000000L;
      long long bkeyRight = akeyRight;
      long long ckeyLeft = akeyLeft;
      long long ckeyRight = akeyRight ^ 0x0000000080000000L;
      long long dkeyLeft = ckeyLeft ^ 0x0000000080000000L;
      long long dkeyRight = ckeyRight;

      clock_t ot = clock();
      printf ("Kasumi init in proc # %i [%f]\n", rank,
((float)ot)/CLOCKS_PER_SEC);
      Kasumi *ka = new Kasumi(akeyLeft, akeyRight);
      Kasumi *kb = new Kasumi(bkeyLeft, bkeyRight);
      Kasumi *kc = new Kasumi(ckeyLeft, ckeyRight);
      Kasumi *kd = new Kasumi(dkeyLeft, dkeyRight);
      MPI_Barrier(MPI_COMM_WORLD);

      ot = clock();
      printf ("DC init in proc # %i [%f]\n", rank,
((float)ot)/CLOCKS_PER_SEC);
```

```
	DataCollect *dc = new DataCollect();
	MPI_Barrier(MPI_COMM_WORLD);

	MPI_Scatter(dc->valCa, div, MPI_LONG_LONG, bufCa, div, MPI_LONG_LONG, 0,
MPI_COMM_WORLD);
	MPI_Scatter(dc->valCc, div, MPI_LONG_LONG, bufCc, div, MPI_LONG_LONG, 0,
MPI_COMM_WORLD);
	MPI_Barrier(MPI_COMM_WORLD);

	ot = clock();
	printf ("crypt in proc # %i [%f]\n", rank, ((float)ot)/CLOCKS_PER_SEC);
	long i;
	for(i=0; i<div; i++)
	{
		bufCb[i] = kb->encrypt((ka->decrypt(bufCa[i], 7) ^
0x0000000000100000L), 7);
		bufCd[i] = kd->encrypt((kc->decrypt(bufCc[i], 7) ^
0x0000000000100000L), 7);
	}
	MPI_Barrier(MPI_COMM_WORLD);

	MPI_Gather(bufCb, div, MPI_LONG_LONG, dc->valCb, div, MPI_LONG_LONG, 0,
MPI_COMM_WORLD);
	MPI_Gather(bufCd, div, MPI_LONG_LONG, dc->valCd, div, MPI_LONG_LONG, 0,
MPI_COMM_WORLD);
	MPI_Barrier(MPI_COMM_WORLD);

	if (rank == 0)
	{
		ot = clock();
		printf ("pair making [%f]\n", ((float)ot)/CLOCKS_PER_SEC);
		for(i=0; i<16777216; i++)
		{
			dc->PairCaCb(dc->valCa[i], dc->valCb[i]);
		}

		ot = clock();
		printf ("EqualsCheck() start [%f]\n", ((float)ot)/CLOCKS_PER_SEC);
		dc->EqualsCheck();
		ot = clock();
		printf ("EqualsCheck() end [%f]\n", ((float)ot)/CLOCKS_PER_SEC);

		dc->Dest();
		ot = clock();
		printf ("dest done [%f]\n", ((float)ot)/CLOCKS_PER_SEC);

		dc->Stest();
		ot = clock();
		printf ("true find [%f]\n", ((float)ot)/CLOCKS_PER_SEC);
	}

	MPI_Finalize();
	return 0;
} // end of main program
```