Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ДОПУСТИТИ ДО ЗАХИСТУ:
завідувачка кафедри кібербезпеки
та захисту інформації
_____Наталія ЛУКОВА-ЧУЙКО
«14» червня 2022р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

дипломної роботи
бакалавра
(назва освітнього ступеня)

галузь знань 12 Інформаційні технології
(шифр і назва галузі знань)

спеціальність 125 Кібербезпека
(код і назва спеціальності)

освітня програма Кібербезпека
(назва освітньої програми)

на тему: «Дослідження потокових шифрів з регістром зсуву з лінійним зворотнім зв'язком на прикладі ДСТУ 8845:2019 (СТРУМОК)»

Виконавець: студент IV курсу, групи КБ-41

_____ **Максим ШОВКУН**
(підпис) (прізвище ім'я по-батькові)

	Прізвище, ініціали	**Підпис**
Керівник	Андрій ФЕСЕНКО	

Нормоконтроль	Олександр ТОРОШАНКО	

Київ 2022

Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ЗАТВЕРДЖЕНО:
завідувачка кафедри кібербезпеки
та захисту інформації
_____Наталія ЛУКОВА-ЧУЙКО
«01» листопада 2021 р.

ЗАВДАННЯ
на виконання дипломної роботи

спеціальності 125 Кібербезпека
(код і назва спеціальності)

освітньої програми Кібербезпека
(назва освітньої програми)

Студентові КБ-41 **Шовкуну Максиму Ігоровичу**
(група) (прізвище ім'я по-батькові)

Тема дипломної роботи Дослідження потокових шифрів з регістром зсуву з лінійним зворотнім зв'язком на прикладі ДСТУ 8845:2019 (СТРУМОК)

1. ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Тема дипломної роботи затверджена на засіданні кафедри кібербезпеки та захисту інформації протокол №5 від 29.10.2021 р.

2. ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Visual Studio Code, ОС Kali Linux, Ubuntu, алгоритм шифрування ДСТУ 8845:2019

3. ЗМІСТ РОЗРАХУНКОВО-ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ

Алгоритми шифрування. Потокові шифри, що базуються на РЗЛЗЗ. SNOW 2.0. Шифр ДСТУ 8845:2019 (СТРУМОК). Вибір інструментальних засобів. Програмна реалізація. Швидкість шифрування.

4. ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Практична цінність Створення системи аналізу ефективності та властивостей алгоритму шифрування

5. ДАТА ВИДАЧІ ЗАВДАННЯ

Дата видачі завдання: 01 листопада 2021 року

Завдання видав _____ (підпис) Андрій ФЕСЕНКО (ініціали, прізвище)

Завдання прийняв до виконання _____ (підпис) Максим ШОВКУН (ініціали, прізвище)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів робіт	Строки виконання робіт (початок-кінець)	Відмітка про виконання
1	Уточнення постановки задачі	01.11.2021 – 27.01.2021	*виконано*
2	Аналіз літератури	28.01.2021 – 11.02.2021	*виконано*
3	Розгляд потокових шифрів	12.02.2021 – 24.02.2021	*виконано*
4	Розгляд РЗЛЗЗ	25.02.2021 – 24.03.2021	*виконано*
5	Розгляд SNOW 2.0	25.03.2021 – 07.04.2021	*виконано*
6	Вибір інструментальних засобів	08.04.2021 – 20.04.2021	*виконано*
7	Програмна реалізація алгоритму шифрування та дослідження швидкості шифрування	21.04.2021 – 20.05.2021	*виконано*
8	Програмна реалізація ір-телефонії з методом шифрування ДСТУ 8845:2019	21.05.2021 – 04.06.2021	*виконано*
9	Оформлення пояснювальної записки	05.06.2021 – 06.06.2021	*виконано*
10	Підготовка до захисту	07.06.2021 – 13.06.2022	*виконано*

Завдання видав _____ (підпис) Андрій ФЕСЕНКО (ініціали, прізвище)

Завдання прийняв до виконання _____ (підпис) Максим ШОВКУН (ініціали, прізвище)

Термін подання дипломної роботи до ЕК 06 червня 2022 року

РЕФЕРАТ

Пояснювальна записка дипломної роботи складається зі вступу, трьох розділів, загальних висновків, списку використаних джерел та додатків. Основний текст займає 64 сторінки, включає в себе зміст, вступ, чотири розділи дипломної роботи, висновки та список джерел. Крім того, робота містить 8 додатків із загальною кількістю сторінок 45. У пояснювальній записці дипломної роботи міститься 35 рисунків і 1 таблиця.

Метою роботи є розробити захищену ір-телефонію з використанням потокового шифру з регістром зсуву з лінійним зворотнім зв'язком ДСТУ 8845:2019 (СТРУМОК)

Для досягнення зазначеної мети поставлено наступні завдання:

- Провести аналіз та класифікацію алгоритмів шифрування
- Проаналізувати потоковий шифр Snow 2.0
- Проаналізувати потоковий шифр ДСТУ 8845:2019 (СТРУМОК)
- Програмно реалізувати шифр ДСТУ 8845:2019 (СТРУМОК)
- Реалізувати програмний аналізатор шифру ДСТУ 8845:2019 (СТРУМОК)
- Програмно розробити та реалізувати ір-телефонію з відповідним методом шифрування голосових даних

Об’єктом дослідження є потокові шифри з регістром зсуву з лінійним зворотнім зв'язком

Предметом дослідження є мережевий додаток реалізації ір-телефонії

Практичною цінністю отриманих результатів є створення системи аналізу ефективності та властивостей певного алгоритму шифрування, що дає виявити вразливості і можливі атаки.

Ключові слова: поле, РЗЛЗЗ, алгоритм шифрування, скінчений автомат, вказівник, вектор ініціалізації

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Позначення:

$\oplus$ — інфіксний бінарний оператор побітового додавання за модулем 2 (XOR);

& — інфіксний бінарний оператор побітової кон’юнкції;

$\neg$ — префіксний унітарний оператор заперечення;

$+_{64}$ — інфіксний бінарний оператор додавання цілих чисел за модулем 2^{64};

$\otimes$ — інфіксний бінарний оператор множення в арифметиці поля $GF(2^{64})$.

Скорочення:

РЗЗЗ — регістр зсуву з зворотним зв’язком;

РЗЛЗЗ — регістр зсуву з лінійним зворотним зв’язком;

РЗ — регістр зсуву;

ПЗЗ — пристрій зворотного зв'язку;

ПРЗ — період регістра зсуву;

СПШ — Синхронні потокові шифри

АПШ — Самосинхронізуючі потокові шифри

ГПК – генератор потоку ключів

СТРУМОК — познака генератора КП, визначеного цим стандартом;

СА — скінченний автомат;

КП – КП

Init — функція ініціалізації внутрішнього стану;

Next — функція наступного стану;

Strm — функція КП;

T — функція нелінійної підстановки;

ЕЦП – електронний цифровий підпис;

ПЗ – програмне забезпечення;

ІС – інформаційна система;

IDE – інтегроване середовище розробки.

ВСТУП

В сучасному світі зі швидким розвитком різного роду інформаційних технологій росте попит на засоби захисту від запобігання несанкціонованого доступу до інформації. Як метод захисту інформації я обрав напрямок криптографічного методу захисту і тут мова йде про дослідження шифру, на якому базується тематика даної роботи.

Для досягнення мети поставлено такі задачі:

- Провести аналіз та класифікацію алгоритмів шифрування
- Проаналізувати потоковий шифр Snow 2.0
- Проаналізувати потоковий шифр ДСТУ 8845:2019 (СТРУМОК)
- Вибрати інструментальні засоби розробки
- Програмно реалізувати шифр ДСТУ 8845:2019 (СТРУМОК)
- Реалізувати програмний аналізатор шифру ДСТУ 8845:2019 (СТРУМОК)
- Програмно розробити та реалізувати ip-телефонію з відповідним методом шифрування голосових даних

РОЗДІЛ 1
ПОТОКОВІ ШИФРИ З РЕГІСТРОМ ЗСУВУ З ЛІНІЙНИМ ЗВОРОТНІМ ЗВ’ЯЗКОМ

1.1 Алгоритми шифрування. Основні терміни

Паралельно з останніми досягненнями людської думки в сфері комп'ютерних технологій виникли не лише персональні комп'ютери, мережі передачі даних та електронних грошей, але й інші пов'язані поняття: хакерство, інформаційна зброя, комп'ютерні віруси та інші. Загрози ІТС на практиці можуть бути реалізовані різним чином — безпосереднім впливом на інформацію, яка становить інтерес для кінцевих користувачів даних систем, діями, направленими на інформаційні ресурси та телекомунікаційні служби, які забезпечуються в межах даної ІТС

Інформаційна безпеки в ІТС базується на криптографічних методах й засобах захисту інформації. Варто брати до уваги, що забезпечити більш надійний захист можливо лише при застосуванні комплексного підходу, що означає вирішення задачі при допомозі цілої сукупності організаційно-технічних та криптографічних заходів.

Криптографічні методи мають за основу криптографічне перетворення інформації, яке встановлено за визначеними математичними законами, що не допускає несанкціонований доступ та можливість безконтрольного його змінювання не авторизованими користувачами. Криптографічні методи захисту є одним із методів вирішення типових проблем інформаційної безпеки. Інформаційна безпека може бути досягнута цілей захисту завдяки реалізації впровадження криптографічних методів захисту інформаційних ресурсів, конфіденційної інформації. Досягнення забезпечення захисту можливе даними методами:

- шифрування загального інформаційного потоку, передача яких здійснюється відкритими мережами та повідомленнями;

- криптографічна автентифікація різнорівневих об'єктів, які встановлюють зв'язок (тут мова йде щодо рівні моделі взаємодії між відкритими системами);

- шифрування даних, які надаються у формі файлів чи зберігаються у базі даних (БД);

- захист інформаційного потоку, який несе інформацію засобами імітозахисту (це захист від нав'язування хибних повідомлень) та ЕЦП задля забезпечення цілісності та достовірності інформації, яка пересилається;

- використання затемнюючого ЕЦП для приховування анонімності дій клієнтів платіжних систем;

- контроль цілісності ПЗ завдяки застосуванню криптографічних стійких контрольних сум.

Під час застосування більшої частини перелічених вище засобів криптографічного захисту з’являється потреба у обміні певною інформацією. Наведемо приклад: при автентифікації об'єктів ІТС відбувається обмін ідентифікуючих, автентифікуючих даних.

Під час взаємодії в загальному випадку об'єктів (суб'єктів) таких систем постійно спостерігається дотримання деяких домовленостей — протоколу. Формально поняття протоколу означає послідовні дії об'єктів або суб'єктів задля досягнення конкретної мети, що в цій ситуації визначає структуру й специфіку застосування протоколу.

Наприклад, криптографічні протоколи — це ті, де учасники з метою досягнення мети користуються криптографічними перетвореннями інформації. Головні завдання гарантування безпеки в сфері інформації, що вирішуються із застосуванням криптографічних протоколів:

- обмін головними даними із послідуючим встановленням обміну інформацією, що є захищеним; під час цього немає жодних припущень щодо того, чи спілкувалися між собою раніше сторони, які здійснюють обмін ключами

(зокрема, без застосування криптографічних протоколів було б неможливо створити систему, що розподіляє ключову інформацію в розподілених мережах передачі інформації);

- автентифікація об'єктів, між якими встановлюється зв'язок;
- авторизація користувачів системи при прагненні отримати доступ до телекомунікаційних й інформаційних служб.

В умовах сьогодення, за допомогою широкого застосування відкритих мереж передачі даних (мережа Internet і побудовані на її базі мережі intranet, extranet), криптографічні протоколи знайшли широке розповсюдження та застосування з метою вирішення багатогранної сфери завдань та забезпечення послуг, які безперервно розширюються та пропонуються користувачам вказаних мереж.

До того ж, крім означених вище класичних сфер застосування протоколів існує чимало специфічних завдань, що також можливо вирішувати завдяки окремим криптографічним протоколам. Передусім, до них відноситься встановлення частини відомостей без повного чи часткового розкриття власне секрету у його дійсному об'ємі. Наведемо приклад: 20 учасників мають змогу задля досягнення конкретної спільної мети один одному повідомити долю персональної інформації про себе чи об'єднати зусилля та розкрити таємницю, що невідома кожному окремому із них.

Швидке вдосконалення криптографічних протоколів значною мірою підкріплюється розвитком е-банкінгу (систем електронних платежів), інтелектуальних карток, виникненням різноманітних електронних грошей та не лише.

Через те, що на даний момент головним засобом у сфері криптографічного захисту у мережі Internet є протоколи, можемо констатувати: вдосконалення таких засобів захисту даних із обмеженим доступом триватиме у якісному та кількісному плані.

На даним момент широко застосовуються різні криптографічні протоколи задля вирішення задач щодо забезпечення ІБ в локальних та розподілених ІС

вимагає проведення детального розгляду основних їх типів, застосування даних протоколів на практиці та їх побудови на базі спеціальних ІС.

Криптографія, як класична, так і одноключова, спирається на застосування симетричних шифрувальних алгоритмів, де різниця між зашифруванням та розшифруванням полягає лише в порядку виконання і напрямку певних кроків. Зазначені алгоритми застосовують саме той секретний елемент (ключ), а розшифрування — лише просте обернення зашифрування. Через це частіше за все кожний учасник обміну має змогу і зашифровувати, розшифровувати свої повідомлення. Схематично структуру даної системи проілюстровано на рис. 1.1.

По причині великої надмірності природних мов саме у зашифроване повідомлення край складно включити осмислену зміну, через це класична криптографія також захищає і від нав'язування помилкової інформації. При умові, якщо природної надмірності буде виявлено недостатньо задля забезпечення надійного захищення повідомлення від модифікації, дану надмірність є змога штучно збільшити завдяки додаванню в повідомлення певної спеціальної контрольної комбінації, що зветься імітовставкою. Існують різноманітні методи шифрування (рис. 1.4). Частіше за все на практиці використовуються алгоритми підстановки, перестановки та комбіновані методи.

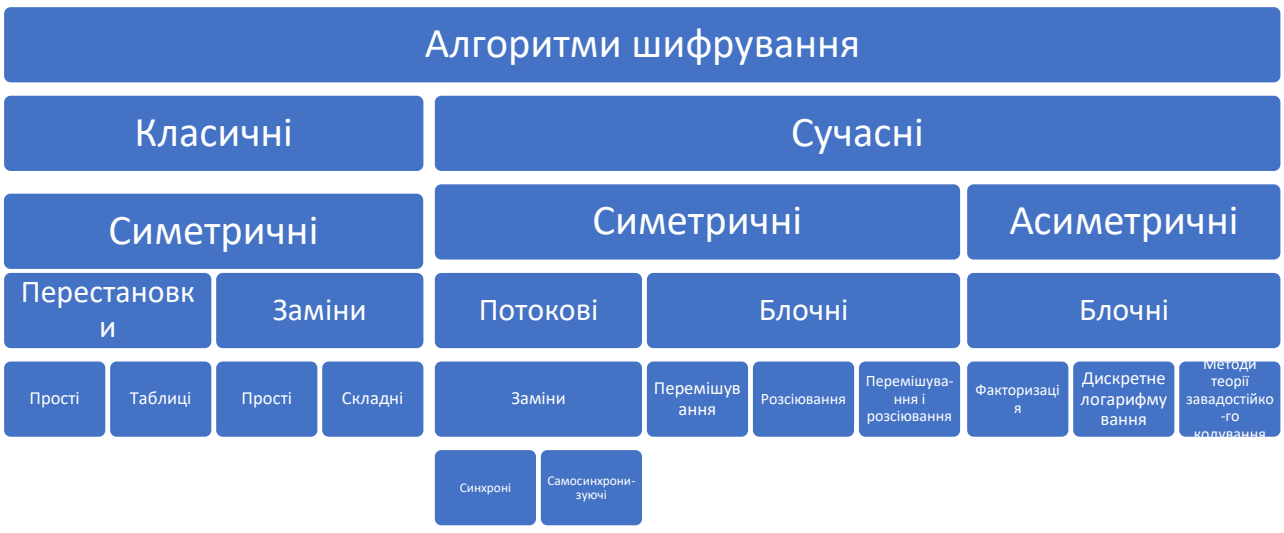

Рисунок 1.1 – Класифікація алгоритмів шифрування

Криптосистеми можна класифікувати за певним типом я на асиметричні , симетричні (рис. 1.4). до того ж, алгоритми розподіляються за обробкою інформації та за перетворенням.

В системах, що відносяться до симетричних, зашифрування та розшифрування здійснюються завдяки одного ключа. Через це його варто зберігати в секреті (з цього й походить друга назва таких систем — криптосистеми із прихованим ключем).

Ключів у асиметричних криптосистемах є два: перший застосовується задля зашифровування інформації (відкритий ключ), другий потрібен задля розшифрування (закритий ключ). Головною відмінністю асиметричних криптосистем є те, що особа, яка із застосуванням відкритого ключа зашифрувала повідомлення, самостійно розшифрувати його при відсутності секретного ключа не зможе. Через це такі криптосистеми є асиметричними, також друга їх назва — відкритоключові.

До гібридних відносяться криптографічні системи, у яких поєднуються два вказані типи криптосистем. В гібридних системах зазвичай повідомлення зашифровується з застосуванням симетричної криптосистеми, після чого використаний секретний ключ зашифровується при допомозі асиметричної криптосистеми.

За типом оброблення вхідної інформаційної послідовності криптосистеми диференціюють на потокові, де перетворенню одразу підлягають усі повідомлення, та блокові, де повідомлення проходять обробку у формі блоків конкретної довжини.

Основною відмінністю сучасної криптографії від тієї, що мала місце до появи комп'ютерних технологій, полягає наступному: криптографічні алгоритми раніше оперували символами, містили природні мови (символи кирилиці чи латиниці). Дані літери замінювалися іншими чи перемішувалися, дотримуючи певного правила. Сучасні ж криптографічні алгоритми застосовують операції із нулями й одиницями (двійковими знаками), а не буквами.

1.2 Потокові шифри

Необхідності виконувати шифрування інформації по символах, наприклад, а не блоками, відповідає потоковий шифр. Він перетворює вхідне повідомлення по одному біту чи байту за окрему операцію. Завдяки потоковому алгоритму шифрування усувається необхідність розбивання повідомлення на ціле число блоків великої довжини й через це є можливість працювати в режимі реального часі. Тому при передачі потоку символів, кожний із них шифрується та передається одразу.

На рис. 1.2. наведено принцип роботи типового потокового шифру

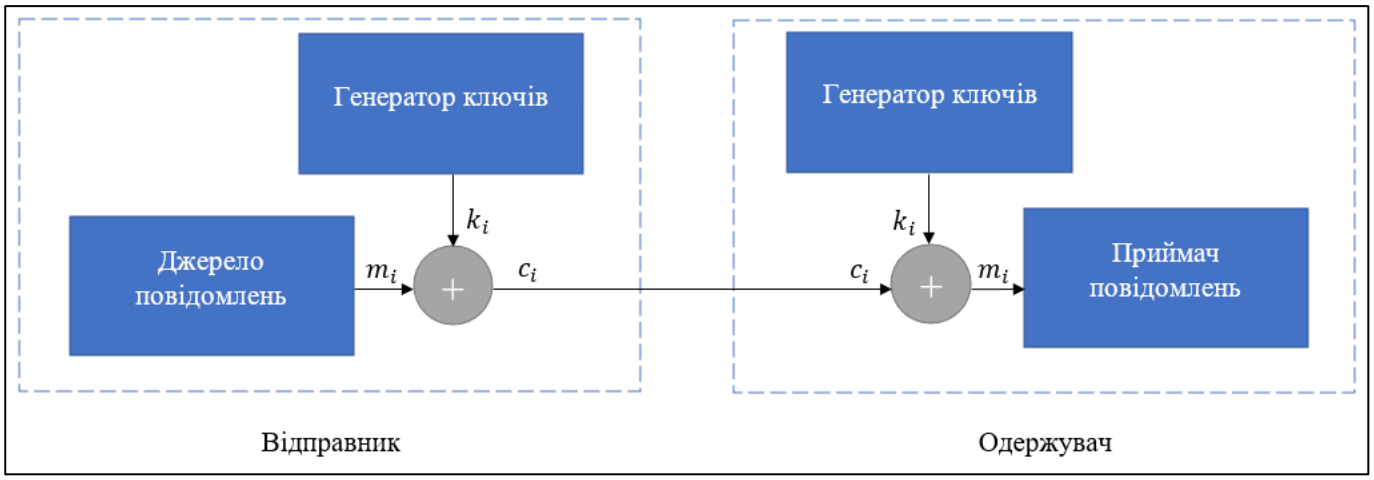

Рисунок 1.2 – Схема потокових шифрів

Генератором ключів видається потік бітів k_i, що використовуватимуться в якості гама. Джерелом повідомлень генеруються біти відкритих даних m_i , що складаються за модулем 2 з гамою, в результаті чого виходять біти зашифрованих даних c_i:

$$c_i = m_i \oplus k_i, i = \overline{1;n}$$

Задля того, щоби з зашифрованого повідомлення $c_1, c_2, \dots, c_n$ відновити початкове $m_1, m_2, \dots, m_n$, потребується згенерувати ідентичну ключову

послідовність $k_1, k_2, \dots, k_n$, як і в ситуації з зашифровуванням, та застосовувати з метою розшифрування наступну формулу:

$$m_i = c_i \oplus k_i, i = \overline{1;n}$$

Це пояснюється тим, що операції порозрядного додавання за модулем 2 має властива оберненість.

Вхідне повідомлення та ключова послідовність — це зазвичай незалежні потоки бітів. Через те, що зашифроване і розшифроване перетворення для усіх потокових шифрів ідентичне, вони мають відрізнятися тільки способом побудови генераторів ключів. Отже, безпека системи залежить від характеристик генератора потоку ключів повністю. Якщо генератор видає послідовність, які складається тільки з одних нулів (чи лише з одиниць), зашифроване повідомлення буде ідентичним до вхідного потоку бітів (зашифроване повідомлення буде інверсією вхідного у разі поодиноких ключів). Наприклад, якщо в якості гама застосовується один символ, що представлений вісьмома бітами, то хоч зашифроване повідомлення і буде ззовні відрізнятися від вхідного, безпека системи буде вкрай низькою. В даному випадку при багаторазовому повторенні коду ключа по всій довжині повідомлення є ризик його розкриття при використанні статистичного методу.

В реальних випадках, якщо до складу вхідного повідомлення входять не тільки одні цифри, а й інші символи, використовується статаналізу, який швидко відновлює ключ та вхідне повідомлення, що має коротку довжину ключа.

1.3 Потокові шифри, що базуються на РЗЗЗ

РЗЗЗ широко використовуються у теорії кодування та криптографії так звані. Вони використовувалися в апаратурі шифрування ще раніше — перед початком масового застосування ЕОМ та сучасних криптографічних шифраторів.

Варіантом РЗЗЗ є отримання псевдовипадкових бітів. РЗЗЗ складається з n бітів РС і самого ПЗЗ, що є зворотнім зв'язком (рис. 1.3).

З РЗ по черзі витягується кожний біт . При необхідності отримання наступний біт, відбувається зсування усіх бітів на 1 розряд в праву сторону. На початок регістра з лівої сторони переходить новий біт. Його видає ПЗЗ і його значення сформоване від решти бітів РЗ. Самі біти РЗ змінюються за певним правилом. Після певного періоду тактів роботи РЗ послідовність бітів починає повторюватися, що очевидно. Довжина отримуваної послідовності перед початком її повторення зоветься ПРЗ.

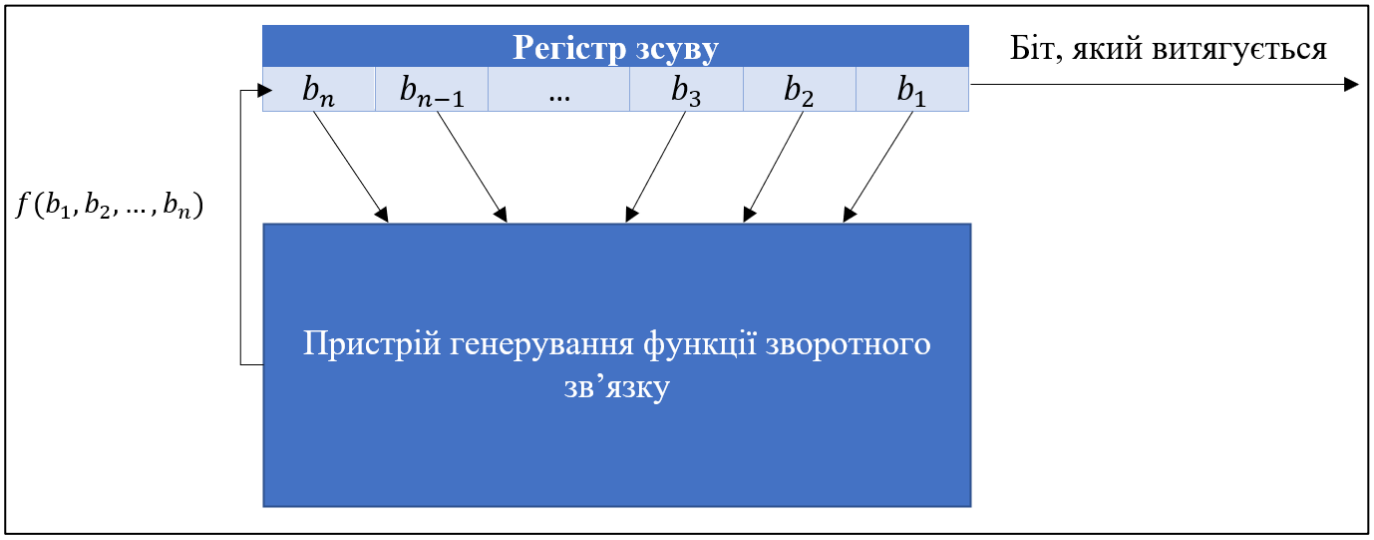

Рис. 1.3. Схема РЗЗЗ

Сучасні цифрові можливості показали змогу потоковим шифрам з РЗ знайти місце у вдалому їх застосуванні.

Наприклад, припустимо, що для потокових шифрів в режимі гамування при передачі по каналу зв'язку здійснилося спотворення певного знаку з зашифрованого повідомлення. Тому всі знаки, які були прийняті без спотворення, розшифруються правильно. Втрачено буде тільки один знак тексту. А зараз давайте уявімо наступне: було втрачено один зі знаків зашифрованого повідомлення при передачі по каналу зв'язку. Наслідком цього стане неправильне розшифрування цілого повідомлення, що йшов після втраченого знаку. Практично у всіх каналах передачі даних існують перешкоди для потокових систем шифрування. Задля запобігання втрати інформації необхідно вирішити питання синхронізації зашифрування та розшифрування

повідомлення. Роблячи акцент на цій темі криптографічні системи можна поділити на самосинхронізуючі та синхронні системи(див. рис. 1.1).

СПШ — шифри, у яких потік ключів генерується не залежить від повідомлення та генерується окремо.

При шифруванні ГПК видаються значення ключів(або їх потік бітів), які є однаковими з ключами(або їх потік бітів) при розшифровці. Через втрату знаку зашифрованого повідомлення виникне порушення синхронізації проміж зазначеними генераторами. Це приведе до неможливості розшифрування шматка повідомлення. Тому за таких умов відправнику й одержувачу задля продовження роботи потрібно синхронізуватися повторно.

Синхронізація здійснюється зазвичай вставкою спеціальних вставок у передане повідомлення. В результаті чого пропущений при передачі знак викликає неправильне розшифрування повідомлення тільки до того моменту, допоки не застосується один із вставок.

Синхронізація має виконуватися таким чином, щоб не було збігу частина потоку ключів. За цієї проблеми ГЗ не відбувається перенесення в ранній стан.

До головних переваг синхронних потокових шифрів відносяться:

- не допускається зміна, доповнення, видалення шифротексту. Ця змінна завдає втрати не виявлення та синхронізації.
- не має поширення помилок (тільки видозмінений, спотворений біт може бути розшифрований невірно);

Уразливість зміни потоку або бітів шифротексту є головним недоліком СПШ. Ця уразливість може бути використана зловмисником для зміни цих бітів певним способом, що після розшифрування відбувається потрібним йому способом.

АПШ — це самосинхронізуючі шифри, в якому є функція ключа і деяке кількість символів шифротексту, які створюють потік ключів.

Через це, для ГПК вхідними даними є попередні n біти шифротексту. Сам ГПК, автоматично охоплює шифрувальний генератор, при кожному прийому бітів.

Здійснення цього режиму проходить наступним чином: повідомлення при кожному такті має певний випадковий n-бітовий початок. Ця частина повинна зашифруватися, відправитись та розшифровуватись. Відповідно спочатку шифротекст матиме не правильне значення. Вже після цього такту відбудеться синхронізація генераторів.

До основних переваги асинхронних потокових шифрів відноситься Статистичне змішування повідомлення. Спостерігається закономірне поширення властивостей статистики на зашифровані повідомлення. Все із-за впливу символу із відкритого повідомлення. Отже, до атак асинхронні потокові шифри можуть бути більш стійкими на основі крайності повідомлення, якщо порівнювати їх із синхронними потоковими шифрами.

Існують наступні недоліки асинхронних потокових шифрів :

- розповсюдження помилки (це пояснюється тим, що кожному окремому неправильного біту повідомлення, яке було зашифровано, є відповіді n-помилки в повідомленні);
- є вразливість повторної передачі.

1.4 SNOW 2.0

Потоковий шифр SNOW 2.0 був запропонований у 2002 році як альтернатива попередньої (слабшої) версії SNOW. На даний момент цей шифр стандартизований і є одним з найшвидших програмно-орієнтованих потокових шифрів. Схема генерації гами шифру наведена на рисунку 1.4

Найпотужнішими з відомих атак на SNOW 2.0 є кореляційні атаки, суть яких полягає у складанні та розв’язуванні систем зашумлених лінійних рівнянь, зокрема, систем рівнянь над полями порядку більше 2. Незважаючи на певний прогрес у

цьому напрямку, є деякі невирішені проблеми, пов’язані з розробкою методів оцінки безпеки та підтвердження безпеки потокових шифрів типу SNOW 2.0 від кореляційних атак. На даний момент відсутні методи, які б дозволили довести захищеність зазначених шифрів від відомих кореляційних атак безпосередньо за допомогою параметрів їх компонентів.

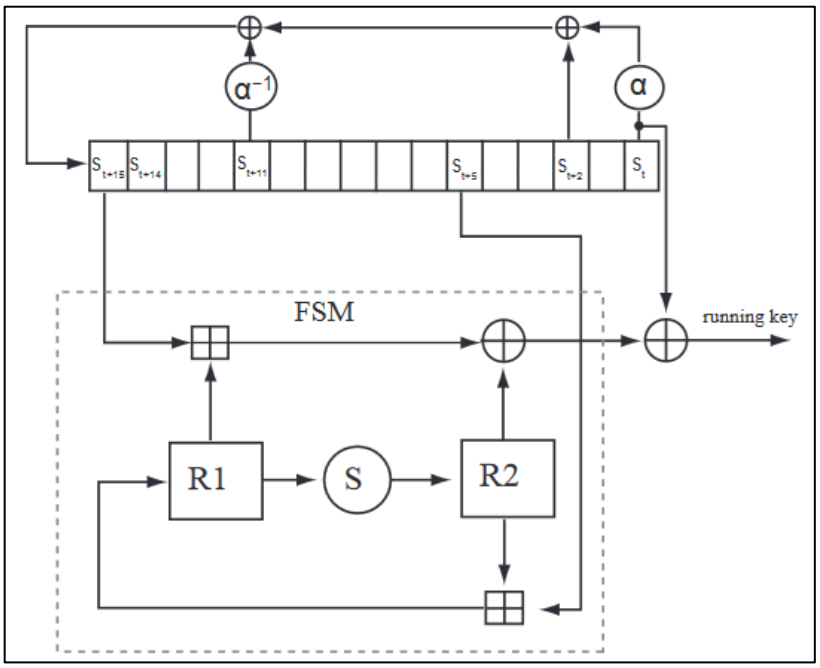

Рис. 1.4. Шифр SNOW 2.0 у режимі генерації гами

Крім того, спроба розширити відомі методи оцінки безпеки SNOW 2.0 від кореляційних атак для ДСТУ 8845:2019 (СТРУМОК) зіткнулася з труднощами, пов’язаними з масштабом проблеми, які необхідно вирішити для отримання меж. На відміну від SNOW 2.0, який будується над полем порядку 2^{32}, шифр ДСТУ 8845:2019 (СТРУМОК) побудований над полем порядку 2^{64}, результатом чого є неможливість практичного застосування окремих алгоритмів, складність яких зростає від $2^{32} \div 2^{37}$ до 2^{64} бітових операцій.

1.5 Шифрування даних в ір-телефонії

1.5.1 Ір-телефонія

IP-телефонія — це загальний термін для технологій, які за допомогою комутації пакетів Інтернет-протоколу з’єднуються з для підтримки голосових даних.

Зазвичай цей зв'язок передавався до того через надані комутаційні з'єднання комутаційної телефонної мережі загального користування. Дзвінки передаються пакетами даних у спільних лініях, при цьому використовуючи інтернет та уникаючи плати за ці з’єднання .

IP-телефонія перетворює голосові дзвінки в цифрові сигнали. Дані цифрові сигнали проходять IP-мережею, наприклад Інтернет, у вигляді пакетів даних, використовуючи IP-з'єднання з комутацією пакетів.

IP-телефонія була заснована в 1991 році при створенні першої подібної програми Speak Freely. Наступного року компанія InSoft створила Communique для проведення настільної відеоконференції. Стандартизування почалось у 1995 році компаніями Radvision, Microsoft і Intel.

Термін IP-телефонія зазвичай використовують ототожнюючи з VoIP, схематичне представлення якої зображено на рисунку 1.5. Ці терміни не є синонімами. Хоч вони обидва використовують локальну мережу щоб підключитись через маршрутизатор до мережі Інтернет, але є певне визначення в їх форматах порівняння.

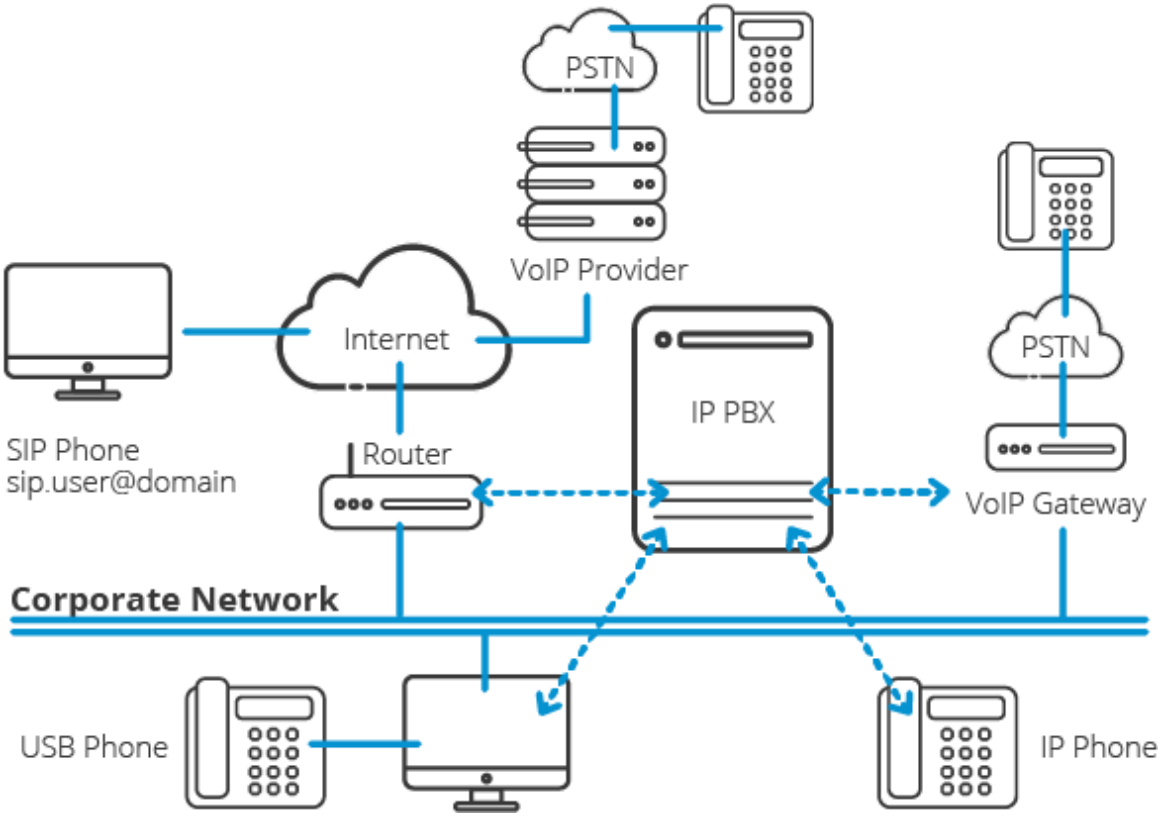

Рис. 1.5. Схема VoIP

VoIP є більш конкретним і описує стандартизовані технології, які надають функції на основі голосу, такі як дзвінки та голосова пошта, через мережі на основі IP;

IP-телефонія, з іншого боку, є більш загальним терміном, який охоплює повний набір технологій, функцій і продуктів - це включає функції на основі голосу, а також функції неголосового зв'язку, такі як обмін миттєвих повідомлень.

Відомими надавачами послуг IP-телефонії є:

- Microsoft
- Zoom
- Skype

1.5.2 Канали передачі даних в ір-телефонії

Лінія зв'язку утворює кілька каналів зв'язку. Це може бути віртуальні або, логічні.

Канал зв'язку – це засіб передачі даних в одну сторону. При умові монопольного використання каналу зв'язку лінії зв'язку, лінія зв'язку називається канал

Цифрові дані передаються за допомогою зміни напруги. Передавання по фізичному середовищі здійснюють за допомогою цифрового і аналогового способами.

Канали можуть бути вузькосмуговими або широкосмуговими. Перший пропускає одну частоту, при цьому абоненти обмінюються даними на одній частоті, а другий багато частот, при цьому абоненти обмінюються даними на різних частотах.

При цифровому способі дані передаються в натуральному вигляді при одній спільній частоті. Цей спосіб забезпечується вузькосмуговою передачею даних, при якій дані передаються між двома користувачами за обмеженою відстанню не більше 1 км з 10 Мбіт/с.

Канал передачі даних - це засоби двостороннього обміну даними, які включають лінії зв'язку та апаратуру передачі (прийому) даних. Канали передачі даних пов'язують між собою джерела інформації та приймачі інформації.

1.5.3 Шифрування даних та каналів передачі даних в ір-телефонії

Для забезпечення захисту даних через канали передачі даних типово використовувати канальне шифрування даних, при якому буде відбуватись шифрування на кожному каналі мережі. При габаритних мережах таке шифрування може бути не рентабельним. До того ж таке шифрування призводить до

закономірностей в зашифровані, накладаючи обмеження на впровадження криптоалгоритмів.

Тому є доцільне на мою думку використання надійного методу шифрування на кінцевих точках мережі. В дипломній роботі побудована ір-телефонія, в якій дані в каналах зв’язку передаються вже зашифровані досліджуваним шифром.

Висновки за розділом 1

У цьому розділі було розглянуто класифікацію алгоритмів шифрування. Виділено з поміж усіх потокові шифри а також розглянуто генератор ключів РЗЛЗЗ, яким є шифр із тематики дипломної роботи. За основу ДСТУ 8845:2019 (СТРУМОК) взято шифр SNOW 2.0, тому йог обуло розглянуто в 1 розділі, та визначено проблеми кореляційних атак. Також розглянуто ір-телефонію, різницю між VoIP, та поставлено теоретичні дані необхідні для предмета дослідження.

РОЗДІЛ 2
ШИФР ДСТУ 8845:2019 (СТРУМОК)

2.1 Основні положення

В потоковому симетричному шифруванні основною операцією є побітове додавання за модулем XOR, що відбувається з ключовим потоком та вихідним текстом. Нормативний документ ДСТУ ISO/IEC 18033-4 має опис різні потокові шифри та генератори випадкових чисел. Всі вони захищають інформацію з обмеженим доступом. В основному під час обробки інформації, повинна бути впроваджена конфіденційність даних, що здійснюється цими засобами. В даному розділі описується вихідна функція синхронного потокового шифру і генератор КП, що також позначений як СТРУМОК.

Генератор СТРУМОК було створено за SNOW-2.0-подібною схемою під сумовувального генератора. Сам генератор SNOW-2.0 розписаний в ДСТУ ISO/IEC 18033-4:2015. У ньому присутній вектор ініціалізації з розміром 256 біт прихований ключ розміром 256 або 512 біт. Сам генератором має рівень стійкості над- та надвисокий; він призначається задля забезпечення конфіденційності інформації при її обробленні, враховуючи можливий квантовий криптографічний аналіз. Установлені рівні захисту та відомості щодо швидкості генерації ключових потоків довідково наведено в додатку А.

Залежно від довжини секретного ключа застосовують два режими генерації ключових потоків:

- режим із 256-бітним ключем, що позначають як СТРУМОК-256;
- режим із 512-бітним ключем, що позначають як СТРУМОК-512.

Моделі потокового шифру визначено в ДСТУ ISO/IEC 18033-4. Потоковий шифр складається з комбінації генератора ключових потоків і вихідної функції.

Для синхронного потокового шифру в ДСТУ ISO/IEC 18033-4 визначено бінарно-адитивну вихідну функцію (задля забезпечення конфіденційності) та вихідну функцію MULTI-S01 (задля забезпечення конфіденційності та цілісності).

Задля забезпечення конфіденційності та цілісності вихідна функція MULTI-S01 застосовує надмірність, для формування якої можна застосовувати ДСТУ 7564.

2.2 Загальні параметри генератора ключових потоків

РЗЛЗЗ та СА є основними структурними компонентами генератора ключових потоків СТРУМОК (в останньому компоненті проводиться нелінійне перетворення). Криптоалгоритм орієнтований на 64-розрядні ЕОМ, тому виправдовується розмір слова в 64 біти. З метою запису байтів застосовують подання від старшого до молодшого.

Спочатку застосовується початкове задання змінної стану S_i (i $\geq$ 0), що складається з шістнадцяти блоків по 64 біти, Вони складаються з таких компонент :

- шістнадцяти $s^{(i)}$-комірок РЗЛЗЗ:

$$s^{(i)} = \left(s_{15}^{(i)}, s_{14}^{(i)} \dots, s_{0}^{(i)}\right);$$

- два регістри СА $r^{(i)}$:

$$r^{(i)} = \left(r_{2}^{(i)}, r_{1}^{(i)}\right);$$

На виході отримують КП або гаму, що формується з 64-бітових слів Z_i. ГПК СТРУМОК у режимі генерації КП шифру зображено на рисунку 2.1.

На рисунку 2.1 зображено роботу генератора в довільний момент часу i. Змінна часової залежності i не зображена.

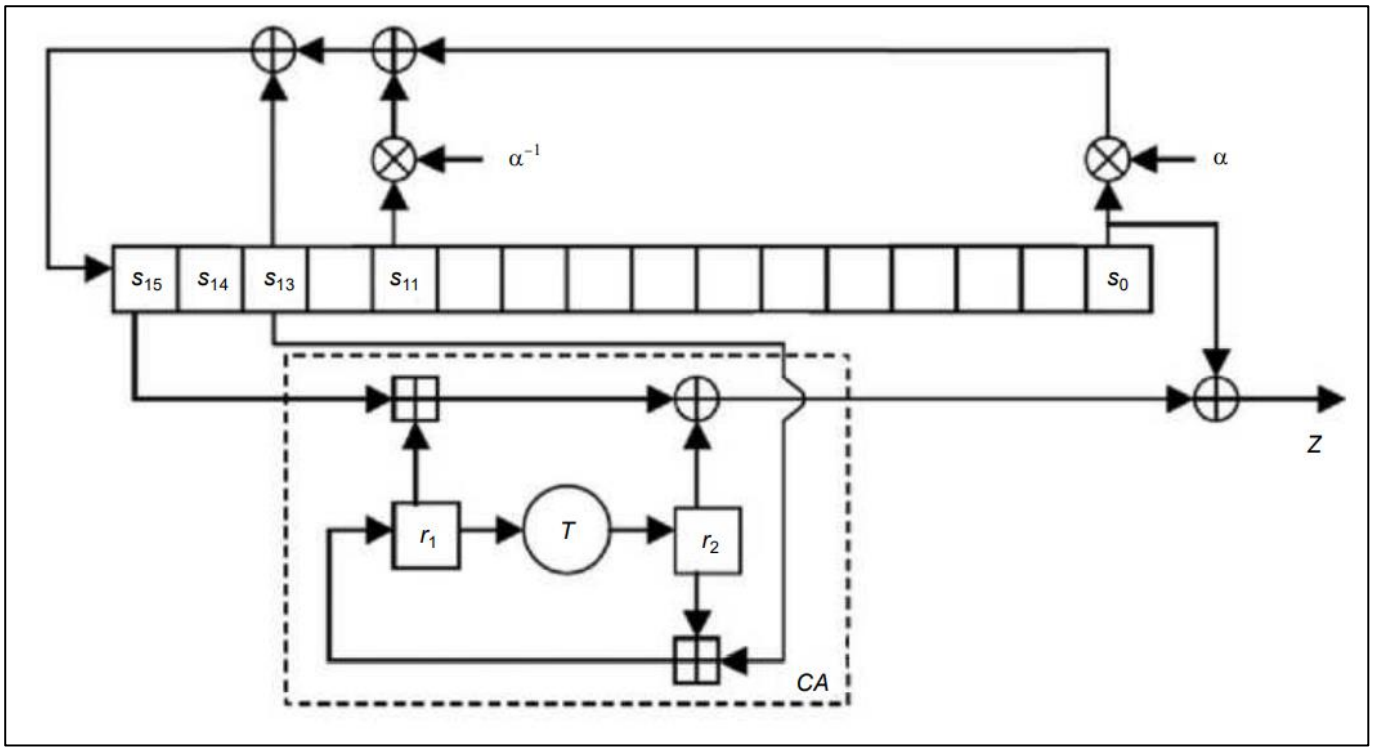

Рисунок 2.1 - ГПК в режимі генерації КП

У РЗЛЗЗ відводи зворотного зв'язку будуються за примітивним над полем GF(264) поліномом:

$$f(x) = x^{16} + x^{13} + \alpha^{-1}x^{11} + \alpha,$$

де α — корінь примітивного над полем $GF(2^8)$ полінома:

$$g(z) = z^8 + \beta^{170}z^7 + \beta^{166}z^6 + \beta^2z^5 + \beta^{224}z^4 + \beta^{70}z^3 + \beta^2.$$

У свою чергу поле $GF(2^8)$ будується за примітивним над полем $GF(2)$ поліномом:

$$p(y) = y^8 + y^4 + y^3 + y^2 + y^1,$$

а коефіцієнти полінома $g(z)$ подано через ступінь примітивного елемента β поля $GF(2^8)$, тобто β — корінь полінома p(y). Відтак отримуємо вежу полів:

$$GF(2) \subset GF(2^8) \subset GF(2^{64}) \subset GF(2^{1024}),$$

де поле $GF(2^{1024})$ задано відводами зворотного зв'язку РЗЛЗЗ як фактор-кільце $GF(2^{64})[x]/(f(x))$;

поле $GF(2^{64})$ задано як фактор-кільце $GF(2^8)[z]/(g(z))$;

поле $GF(2^8)$ задано як фактор-кільце $GF(2)[y]/(p(y)$.

Тому період вихідної послідовності РЗЛЗЗ є максимальним і дорівнює $2^{1024} - 1$.

В алгоритмі СТРУМОК структурно можна виділити наступні три головні функції:

- функцію ініціалізації Init. Вона приймає такі вхідні дані: ключ K по 256, 512 біт і вектор ініціалізації по 256 біт. Ця функція задає значення змінної стану $S_{(0)} = (s^{(0)}, r^{(0)})$; на початку.

- функцію настання наступного стану Next. Вона приймає такі вхідні: змінну стану $S_{(i)} = (s^{(i)}, r^{(i)})$ і виробляє наступне значення змінної стану $S_{(i)} = (s^{(i+1)}, r^{(i+1)})$. Функцію Next можна виконувати у двох режимах (це залежить від способу ітерації) — звичайний чи режим ініціалізації;

- функцію КП Strm. Вона приймає такі вхідні: змінна стану $S_{(i)} = (s^{(i)}, r^{(i)})$ й виробляє в кінці КП Z_i розміром 64 біти.

2.3 Функція Init. Ініціалізації внутрішнього стану

Функція Init (ініціалізації внутрішнього стану) задає значення змінної стану

$$S_{(0)} = (s^{(0)}, r^{(0)}).$$

Ключ для режиму СТРУМОК-256 можемо навести у вигляді чотирьох слів по 64 біта:

$$K = (K_3, K_2, K_1, K_0),$$

а для режиму СТРУМОК-512 — у формі восьми слів по 64 біта:

$$K = (K_7, K_6, K_5, K_4, K_3, K_2, K_1, K_0),$$

де K_3 та K_7 слова для 256-бітових та 512-бітових ключів — максимально значущі слова,

K_0— мінімально вартісні.

Вектор ініціалізації наводиться у вигляді чотирьох слів по 64 біта:

$$IV = (IV_3, IV_2, IV_1, IV_0),$$

де IV_3 — найбільш вартісне слово, а IV_0 — найменш вартісне.

Алгоритм функції:

1. До 16 комірок РЗЛЗЗ заносять значення ключа.

Для версії з 256-бітним ключем виконують операції:

$$s_{15}^{(-33)} = \neg K_0,$$

$$s_{14}^{(-33)} = K_1,$$

$$s_{13}^{(-33)} = \neg K_2,$$

$$s_{12}^{(-33)} = K_3,$$

$$s_{11}^{(-33)} = K_0,$$

$$s_{10}^{(-33)} = \neg K_1,$$

$$s_{9}^{(-33)} = K_2,$$

$$s_{8}^{(-33)} = K_3,$$

$$s_{7}^{(-33)} = \neg K_0,$$

$$s_{6}^{(-33)} = \neg K_1,$$

$$s_{5}^{(-33)} = K_2 \oplus IV_3,$$

$$s_4^{(-33)} = K_3,$$

$$s_3^{(-33)} = K_0 \oplus IV_2,$$

$$s_2^{(-33)} = K_1 \oplus IV_1,$$

$$s_1^{(-33)} = K_2,$$

$$s_0^{(-33)} = K_3 \oplus IV_0,$$

Для версії з 512-бітним ключем K виконують операції:

$$s_{15}^{(-33)} = K_0,$$

$$s_{14}^{(-33)} = \neg K_1,$$

$$s_{13}^{(-33)} = K_2,$$

$$s_{12}^{(-33)} = K_3,$$

$$s_{11}^{(-33)} = \neg K_7,$$

$$s_{10}^{(-33)} = K_5,$$

$$s_9^{(-33)} = \neg K_6,$$

$$s_8^{(-33)} = K_4 \oplus IV_3,$$

$$s_7^{(-33)} = \neg K_0,$$

$$s_6^{(-33)} = K_1,$$

$$s_5^{(-33)} = K_2 \oplus IV_2,$$

$$s_4^{(-33)} = K_3,$$

$$s_3^{(-33)} = K_4 \oplus IV_1,$$

$$s_2^{(-33)} = K_5,$$

$$s_1^{(-33)} = K_6,$$

$$s_0^{(-33)} = K_7 \oplus IV_0,$$

2. Виконують 32 ініціюювальні такти без генерації КП, тобто чотири повні цикли. Формально це наведено так:

$$S_{-1} = Next^{32}(S_{-33}, INIT),$$

що значить, що проводиться 32 ітерації функцією Next при режимі ініціалізації функції INIT, $S_{-33} = (s^{(-33)}, r^{(-33)})$ — змінна стану: обраховані в попередньому шазі шістнадцять комірок РЗ $s^{(-33)}$ та значення двох регістрів $r^{(-33)} = (r_2^{(-33)}, r_1^{(-33)})$ СА, із значенням: $r^{(-33)}$ = (0000000000000000, 0000000000000000).

3. Розраховується початкове значення $S_0 = (s^{(0)}, r^{(0)})$:

$$S_0 = Next(S_{-1}),$$

тобто виконання у звичайного режиму Next.

4. Виводиться обчислене значення $S_0 = (s^{(0)}, r^{(0)})$.

2.4 Функція Next. Наступний стан

Функція має наступний алгоритм:

1. Виконують нелінійну підстановку з метою обновлення значення регістру $r_2^{(i+1)}$ скінченного автомата. Для цього розраховують значення функції T:

$$r_2^{(i+1)} = T(r_1^{(i)}). \quad (2.1)$$

2. Оновлюють значення регістру $r_1^{(i+1)}$ скінченного автомата. Для цього розраховують значення:

$$r_1^{(i+1)} = r_2^{(i)} +_{64} s_{13}^{(i)}, \tag{2.2}$$

де $+_{64}$ відповідає операції додавання цілочисельних даних за модулем 2^{64} (у схемі на рисунках 2.1 та 2.2 цю операцію позначено як $\boxplus$).

3. Оновлюють РЗЛЗЗ для 15-ти комірок:

$$s_j^{(i+1)} = (s_j^{(i)}).$$

для всіх $j = \overline{0; 14}$

4. Оновлюють РЗЛЗЗ, а саме її 16-ї комірки. При звичайному режим функції Next, комірка обчислюється за формулою:

$$s_{15}^{(i+1)} = (s_0^{(i)} \otimes \alpha) \oplus (s_{11}^{(i)} \otimes \alpha^{-1}) \oplus s_{13}^{(i)} \tag{2.3}$$

При обчислені ініціалізації функції Next, обчислення відбувається за формулою:

$$s_{15}^{(i+1)} = FSM(s_{15}^{(i)}, r_1^{(i)}, r_2^{(i)}) \oplus (s_0^{(i)} \otimes \alpha) \oplus (s_{11}^{(i)} \otimes \alpha^{-1}) \oplus s_{13}^{(i)} \tag{2.4}$$

Операції множення $\otimes$ на α та на α^{-1}, та алгоритм СА розглянуто нище.

5. Обчислюють та виводять значення змінної стану $S_i = (s^{(i)}, r^{(i)})$

ГПК СТРУМОК під час виконання функції Next в режимі ініціалізації схематично зображено на рисунку 2.2.

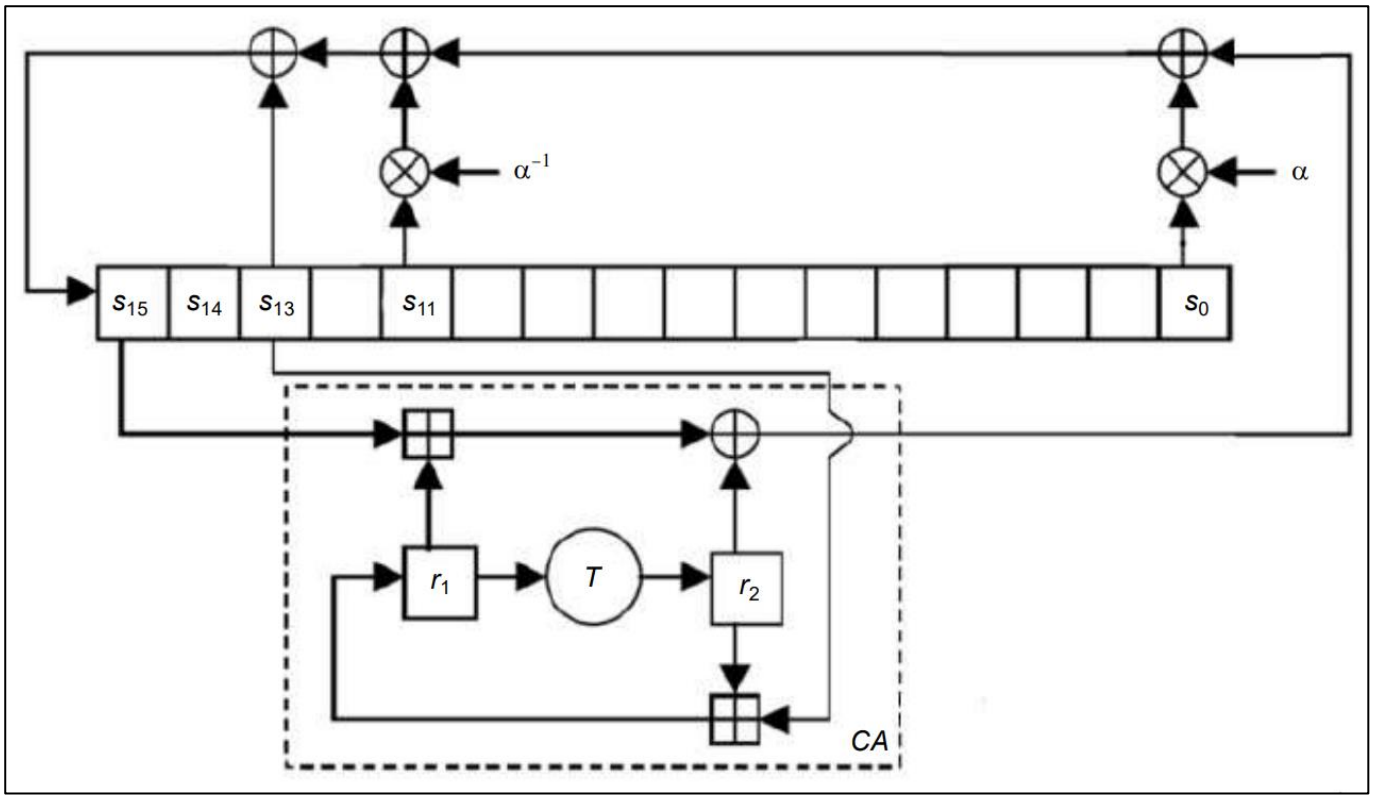

Рисунок 2.2 - ГПК під час виконання функції Next режимом ініціалізації

2.5 Функція Strm. КП

Функцію КП Strm описано так.

Вхід: змінна стану $S_i = \left(s^{(i)}, r^{(i)}\right)$. Вихід: КП Z_i, з розміром 64 бітів.

1. Обчислюють значення:

$$Z_i = FSM(s_{15}^{(i)}, r_1^{(i)}, r_2^{(i)}) \oplus s_0^{(i)}. \tag{2.5}$$

2. Виводять вихідне значення Z_i .

2.6 Функція CA. Скінченний автомат

Функцію скінченного автомата позначено як

$$FSM(s_{15}^{(i)}, r_1^{(i)}, r_2^{(i)}) \tag{2.6}$$

та описано так.

Вхід: три 64-бітові рядки x, y, z.

Вихід: рядок q розміру 64 біти.

1. Обчислення $q = (x +_{64} y) \oplus z$

2. Вивід q.

2.7 Функція T. Нелінійна підстановка

У функція T відбувається перестановка елементів скінченного поля $GF(2^{64})$ згідно ДСТУ 7624 (нацстандарту блокового симетричного криптоперетворення).

Вхід: рядок w слова 64 біт.

Вихід: рядок $T = T(w)$ слова 64 біт.

Алгоритм функції:

1. Рядок w розбивається на підблоки w_j на слова по 8 біт:

$$w = (w_7, w_6, w_5, w_4, w_3, w_2, w_1, w_0).$$

2. Згідно ДСТУ 7624 для кожного отриманого підблока w_j виконують підстановку з алгоритму чотирма табличними перетвореннями $\pi_0, \pi_1, \pi_2, \pi_3$ (див. додаток Е).

У результаті формують вихідний вектор $r = (r_7, r_6, r_5, r_4, r_3, r_2, r_1, r_0)$:

$$r_j = \pi_{j \bmod 4}[w_j],$$

де. $j = \overline{0; 7}$

3. Обчислюють вектор:

$$q = (q_7, q_6, q_5, q_4, q_3, q_2, q_1, q_0).$$

за правилом:

$$\begin{pmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ q_6 \\ q_7 \end{pmatrix} = \begin{pmatrix} 01\ 01\ 05\ 01\ 08\ 06\ 07\ 04 \\ 04\ 01\ 01\ 05\ 01\ 08\ 06\ 07 \\ 07\ 04\ 01\ 01\ 05\ 01\ 08\ 06 \\ 06\ 07\ 04\ 01\ 01\ 05\ 01\ 08 \\ 08\ 06\ 07\ 04\ 01\ 01\ 05\ 01 \\ 01\ 08\ 06\ 07\ 04\ 01\ 01\ 05 \\ 05\ 01\ 08\ 06\ 07\ 04\ 01\ 01 \\ 01\ 05\ 01\ 08\ 06\ 07\ 04\ 01 \end{pmatrix} \cdot \begin{pmatrix} r_0 \\ r_1 \\ r_2 \\ r_3 \\ r_4 \\ r_5 \\ r_6 \\ r_7 \end{pmatrix}$$

де шістнадцяткові елементи матриці й векторів r та q інтерпретують як елементи скінченного поля $GF(2^8)$, заданого як фактор-кільце GF(2)[y]/(p(y)).

Згідно ДСТУ 7624 дану операцію можна також записати скорочено:

$$q_i = (u \gg i) \cdot \begin{pmatrix} r_0 \\ r_1 \\ r_2 \\ r_3 \\ r_4 \\ r_5 \\ r_6 \\ r_7 \end{pmatrix},$$

де u = (01, 01, 05, 01, 08, 06, 07, 04), >>> i — циклічний зсув на і розрядів в право, $i = \overline{0;7}$.

4. Виводять вихідне значення q, інтерпретоване 64-бітовим рядоком. Реалізувати швидке обчислення вектора q можна за правилом:

$$\begin{pmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ q_6 \\ q_7 \end{pmatrix} = \bigoplus_{i=0}^{7} T_i[w_i],$$

де

$$T_0[a] = \begin{pmatrix} 01 \\ 04 \\ 07 \\ 06 \\ 08 \\ 01 \\ 05 \\ 01 \end{pmatrix} \cdot \pi_0[a],$$

$$T_1[a] = \begin{pmatrix} 01 \\ 01 \\ 04 \\ 07 \\ 06 \\ 08 \\ 01 \\ 05 \end{pmatrix} \cdot \pi_1[a],$$

$$T_2[a] = \begin{pmatrix} 05 \\ 01 \\ 01 \\ 04 \\ 07 \\ 06 \\ 08 \\ 01 \end{pmatrix} \cdot \pi_2[a],$$

$$T_3[a] = \begin{pmatrix} 01 \\ 05 \\ 01 \\ 01 \\ 04 \\ 07 \\ 06 \\ 08 \end{pmatrix} \cdot \pi_3[a],$$

$$T_4[a] = \begin{pmatrix} 08 \\ 01 \\ 05 \\ 01 \\ 01 \\ 04 \\ 07 \\ 06 \end{pmatrix} \cdot \pi_0[a],$$

$$T_5[a] = \begin{pmatrix} 06 \\ 08 \\ 01 \\ 05 \\ 01 \\ 01 \\ 04 \\ 07 \end{pmatrix} \cdot \pi_1[a],$$

$$T_6[a] = \begin{pmatrix} 07 \\ 06 \\ 08 \\ 01 \\ 05 \\ 01 \\ 01 \\ 04 \end{pmatrix} \cdot \pi_2[a],$$

$$T_7[a] = \begin{pmatrix} 04 \\ 07 \\ 06 \\ 08 \\ 01 \\ 05 \\ 01 \\ 01 \end{pmatrix} \cdot \pi_3[a],$$

Застосування таблиць-констант $T_i[a]$, $i = \overline{0;7}$ дає змогу суттєво скоротити число операцій. Функцію нелінійної підстановки обчислюють за сім операцій XOR над 64-бітовими рядками. Значення таблиць-констант $T_i[a]$, $i = \overline{0;7}$ для 8-бітових рядків a (які індексують таблиці) наведено в додатку Д.

2.8 Множення на α

Множення на α за правилами поля $GF(2^{64})$ здійснюється з 256 рядків по 64 бітів у кожному за використанням таблиці передобчислень Mul_a.

Вхід: рядок 64-бітового значення w, що становить елемент поля $GF(2^{64})$.

Вихід: 64-бітовий рядок $w' = w \otimes a$, що становить елемент поля $GF(2^{64})$.

Алгоритм множення:

1. Обчислюють значення:

$$w' = (w \ll 8) \oplus Mul_a[w \gg 56],$$

де $w \ll 8$ — результат зсуву ліворуч (у напрямку до вищих розрядів) рядка розміру 64 біти w на 8 розрядів із заповненням нищих розрядів нульовими значеннями;

$w \gg 56$ — результат зсуву праворуч (у напрямку до нищих розрядів) рядка розміром 64 біти w на 56 розрядів разом із заповненням вищих розрядів нулями. Вісім нищих розрядів вектора $w \gg 56$ розуміють як елемент поля $GF(2^8)$ для індексації таблиці Mul_a;

Mul_a — таблиця-константа. Містить 256 рядків по 64 бітів у кожному;

Mul_a [c] —значення таблиці розміром 64 біти передобчислень у рядку по індексу c, де c є елементом поля $GF(2^8)$, Mul_a [c] — елемент поля $GF(2^{64})$.

2. Виводиться вихідне значення w'.

2.9 Множення на α^{-1}

За допомогою таблиці передобчислень з 256 рядків по 64 бітів у кожному реалізують множення на α^{-1} в арифметиці поля $GF(2^{64})$.

Вхід: рядок 64-бітового значення w, що становить елемент поля $GF(2^{64})$.

Вихід: 64-бітовий рядок $w' = w \otimes a^{-1}$, що становить елемент поля $GF(2^{64})$.

Алгоритм множення:

1. Обчислюють значення:

$$w' = (w \gg 8) \oplus Mul_{a^{-1}}[w\&\gamma],$$

де w >> 8 — результат зсуву праворуч (у сторону нищих розрядів) рядка довжини 64 біти w на 8 розрядів із заповненням вищих розрядів нульовими значеннями;

w & γ — значення побітової кон’юнкції рядків довжини 64 біти w γ, що у шістнадцятковому поданні : γ = 00000000000000FF. Вісім молодших розрядів вектора w & γ інтерпретують як елемент поля $GF(2^8)$ задля створення індексів таблиці $Mul_{a^{-1}}$

2. Виводиться вихідне значення w'.

2.10 Значення таблиць-констант Mul_{α} та $Mul_{\alpha^{-1}}$

З метою швидкого шифрування застосовують таблиці передобчислень Mul_{α} та $Mul_{\alpha^{-1}}$ Значення таблиць констант $Mul_{\alpha}[a]$ та $Mul_{\alpha^{-1}}[a]$ для 8-бітових рядків a (які індексують таблиці) наведено в додатку Г.

Висновки за розділом 2

В даному розділі розглянуто вихідну функцію синхронного потокового шифру та генератор псевдовипадкових чисел (КП), позначений як СТРУМОК.

Шифр має однаковий кінцевий автомат із шифром SNOW 2.0.

РОЗДІЛ 3
ПРОГРАМНА ЗДІЙСНЕННЯ ДСТУ 8845:2019 (СТРУМОК)

3.1 Вибір інструментальних засобів

Під час дослідження теми та реалізації алгоритму шифрування постало питання вибору інструментальних засобів, а саме: вибір мови програмування, вибір середовища розробки.

Для даного прикладу я обрав – C. C є універсальною, процедурною, імперативною мовою програмування загального призначення. Було вибрано саме цю мову програмування через можливість працювати з пам’яттю, що дає можливість виграти у швидкості шифрування.

Microsoft Visual Studio — IDE компанії Microsoft. Він застосовується задля розроблення комп’ютерних програм, web-сайтів, web-додатків, мобільних додатків, web-сервісів, ігрових додатків. Visual Studio використовує платформи розробки ПЗ Microsoft (Windows API, Windows Presentation Foundation, Windows Forms, Windows Store, Microsoft Silverlight) та надає змогу створювати і рідний, і керований код. Це середовище дає інструменти для розробки на 36 різних мов програмування та дає редактору коду й налагоджувачу підтримувати (у різній мірі) практично усі мови програмування (при умові, якщо існує спеціальна мова).

C використовується як проміжна мова деякими високорівневими мовами програмування. Тому вихідну програму можна використовувати в проектах для забезпечення шифрування даних.

3.2 Структура додатку

Програмна частина містить 3 файли:

- main.c (див. додаток В)
- strumok.c (див. додаток Б)

- strumok.h (див. додаток А)

Файл main.c

На 7 рядку файлу було створено покажчик на структуру Dstu8845Ctx. Сама структура оголошена у файлі strumok.h, реалізована у файлі strumok.c(див рис. 3.1).

```
679  struct Dstu8845Ctx_st {
680      uint64_t  S[16];
681      uint64_t  r[2];
682      uint64_t  key[8];
683      uint64_t  iv[4];
684      uint8_t   key_size;
685  };
```

Рис. 3.1 – Структура Dstu8845Ctx

Для зберігання значень ключів, було створено масив key на 8 елементів. Для вектора ініціалізації – iv на 4 елемента. Обидва масива зберігають 64-бітові числа. Елементам вектора ініціалізації задано значення від 1 до 4. Елементам ключів задано значення нуля, крім останнього, яке дорівнює 0x8000000000000000.

Також було створено 8-бітову зміну key_size, що розміром ключа. Задано значення 64.

Далі було викликано функцію dstu8845_init, що є функцією ініціалізації внутрішнього стану Init. Ця функція приймає аргументами:

- Вказівник на структуру Dstu8845Ctx
- Ключ шифрування
- Розмір ключа
- Вектор ініціалізації

Далі створюється масив вхідного повідомлення на 4096 елементів.

Після ініціалізації викликається функція dstu8845_crypt, що шифрує повідомлення. Ця функція приймає аргументами:

- Вказівник на структуру Dstu8845Ctx

- Вхідне повідомлення
- Кількість бітів, які шифруються
- Зашифроване повідомлення

Дана функція шифрує частинами по 128 бітів за допомогою функції next_stream_full_crypt. Якщо залишаються біти, вони окремо шифруються в цій функції.

```
1058  int dstu8845_crypt(Dstu8845Ctx *ctx, const uint8_t *in, size_t inl, uint8_t *out)
1059  {
1060      uint8_t i;
1061      uint64_t keystream[16];
1062
1063      for (; inl >= 128; inl -= 128, in += 128, out += 128) {
1064          next_stream_full_crypt(ctx, (uint64_t *)in, (uint64_t *)out);
1065      }
1066
1067      if (inl > 0) {
1068          next_stream(ctx, keystream);
1069
1070          for (i = 0; i < inl; i++) {
1071              out[i] = in[i] ^ ((uint8_t *)keystream)[i];
1072          }
1073      }
1074
1075      return 1;
1076  }
```

Рис. 3.2 – Функція шифрування

Файл strumok.h

На рисунку 3.2 вміст цього файлу. Файл містить прототипи функцій, оголошення типу структури.

```
 1 #include <stdint.h>
 2 #include <string.h>
 3
 4 typedef struct Dstu8845Ctx_st Dstu8845Ctx;
 5
 6 Dstu8845Ctx *dstu8845_alloc();
 7
 8 int dstu8845_init(Dstu8845Ctx *ctx, const uint64_t *key, uint8_t key_size, const uint64_t *iv);
 9 int dstu8845_set_iv(Dstu8845Ctx *ctx, const uint64_t *iv);
10 int dstu8845_crypt(Dstu8845Ctx *ctx, const uint8_t *in, size_t inl, uint8_t *out);
11 void dstu8845_free(Dstu8845Ctx *ctx);
```

Рис. 3.3 – Вміст файлу strumok.h

Файл strumok.c

У файлі оголошено директиви. Множення на a, множення на обернене a, Функція нелінійної підстановки.

```
4   #define byte(n,w)    (((w)>>(n*8)) & 0xff)
5   #define ainv_mul(w) (((w)>>8)^(strumok_alphainv_mul[w&0xff]))
6   #define a_mul(w)     (((w)<<8)^(strumok_alpha_mul[w>>56]))
7   #define T(w)         ((T0[byte(0,(w))])^(T1[byte(1,(w))])^(T2[byte
```

Рис. 3.4 – Директиви strumok.c

На рисунку 3.4 нам143-677 рядках оголошені масиви зі значеннями таблиць-констант T_i.

```
143 > static uint64_t T0[256] = {…
208   };
209
210 > static uint64_t T1[256] = {…
275   };
276
277 > static uint64_t T2[256] = {…
342   };
343
344 > static uint64_t T3[256] = {…
409   };
410
411 > static uint64_t T4[256] = {…
476   };
477
478 > static uint64_t T5[256] = {…
543   };
544
545 > static uint64_t T6[256] = {…
610   };
611
612 > static uint64_t T7[256] = {…
677   };
```

Рис. 3.5 – Директиви strumok.c

На рисунку 3.5 представлено створення вказівника на структуру Dstu8845Ctx, виділення для неї пам’яті. Функція повертає вказівник на цю структуру.

```
889  Dstu8845Ctx *dstu8845_alloc()
890  {
891      Dstu8845Ctx *ctx = NULL;
892
893      ctx = calloc(1, sizeof(Dstu8845Ctx));
894      if (ctx == NULL) {
895          return NULL;
896      }
897      memset(ctx, 0, sizeof(Dstu8845Ctx));
898
899      return ctx;
900  }
```

Рис. 3.6 – Функція виділення пам’яті для структури

Функція dstu8845_free звільняє пам'ять для вказівника структури.

```
687  static inline void next_stream(Dstu8845Ctx *ctx, uint64_t *out_stream)
688  {…
786  }
787
788  static inline void next_stream_full_crypt(Dstu8845Ctx *ctx, uint64_t *in, uint64_t *out)
789  {…
887  }
888
889  Dstu8845Ctx *dstu8845_alloc()
890  {…
900  }
901
902  int dstu8845_init(Dstu8845Ctx *ctx, const uint64_t *key, uint8_t key_size, const uint64_t *iv)
903  {…
1051 }
1052
1053 int dstu8845_set_iv(Dstu8845Ctx *ctx, const uint64_t *iv)
1054 {…
1056 }
1057
1058 int dstu8845_crypt(Dstu8845Ctx *ctx, const uint8_t *in, size_t inl, uint8_t *out)
1059 {…
1076 }
1077
1078 void dstu8845_free(Dstu8845Ctx *ctx)
1079 {…
1081 }
```

Рис. 3.7 – Функції файлу strumok.c

3.2.1 Програмна здійснення функції ініціалізації внутрішнього стану Init

Алгоритмічна функція Init реалізована в середині програмної функції dstu8845_init, яка спочатку заповнює значення змінної розміру ключа key_size у

відповідне поле структури. Потім в структуру ctx копіюється значення масивів вектору ініціалізації та ключів. Далі заповнюються комірки РЗЛЗЗ. СА задається початкові значення нуль.

```
902  int dstu8845_init(Dstu8845Ctx *ctx, const uint64_t *key, uint8_t key_size, const uint64_t *iv)
903  {
904  >   if (key_size == 32) {…
924  >   } else if (key_size == 64) {…
944      } else {
945          return 0;
946      }
947
948      ctx->r[0] = 0;
949      ctx->r[1] = 0;
950  >   for (int i = 0; i < 2; i++) {…
1048     }
1049
1050     return 1;
1051 }
```

Рис. 3.8 – Синтаксис функції dstu8845_init

Функція далі виконує 32 ініціювальні такти без генерації КП. Так як генератор після 16 тактів повертається у вихідне положення, то виконується цикл з 2 ітераціями, де дії відбуваються зі зміщеними індексами по порядку.

Зміна outfrom_fsm є вихідним значенням СА. На першому такті його значення дорівнює формулі 6. Далі обчислюється формула 4, і відразу переміщається в 0 регістр. За формулою 2 обчислюється значення першого регістру СА і поміщається в змінну fsmtmp. За формулою 1 задається значення другому регістру СА. Змінна fsmtmp поміщається в перший регістр.

```
950     for (int i = 0; i < 2; i++) {
951         static uint64_t outfrom_fsm, fsmtmp;
952 
953         outfrom_fsm = (ctx->r[0] + ctx->S[15]) ^ ctx->r[1];
954         ctx->S[0] = a_mul(ctx->S[0]) ^ ctx->S[13] ^ ainv_mul(ctx->S[11]) ^ outfrom_fsm;
955         fsmtmp = ctx->r[1] + ctx->S[13];
956         ctx->r[1] = T(ctx->r[0]);
957         ctx->r[0] = fsmtmp;
958 
959         outfrom_fsm = (ctx->r[0] + ctx->S[0]) ^ ctx->r[1];
960         ctx->S[1] = a_mul(ctx->S[1]) ^ ctx->S[14] ^ ainv_mul(ctx->S[12]) ^ outfrom_fsm;
961         fsmtmp = ctx->r[1] + ctx->S[14];
962         ctx->r[1] = T(ctx->r[0]);
963         ctx->r[0] = fsmtmp;
964 
```

Рис. 3.9 – Цикл з ініціювальними тактами

Так як комірки РЗЛЗЗ зсуваються, то у формулах регістр замінюються на збільшені на 1 по модулю 16.

3.2.2 Програмна здійснення функції наступного стану Next

Алгоритмічна функція Next реалізована в середині програмної функції next_stream_full_crypt, яка 16 раз виконує її і зміщує кожний раз РЗ. Два перші такти в цієї функції представлено на рисунку 3.10. За формулою 3 задається значення і відразу зсувається в 0 регістр. Далі за тим ж принципом як і у функції dstu8845_init, задаються значення регістрам СА. В кінці такту вхідне 64-бітове число додається побітово по модулю 2 за формулою 5 із гамою шифру.

Так як комірки РЗЛЗЗ зсуваються, то у формулах регістр замінюються на збільшені на 1 по модулю 16.

```
788 static inline void next_stream_full_crypt(Dstu8845Ctx *ctx, uint64_t *in, uint64_t *out)
789 {
790     uint64_t fsmtmp;
791 
792     ctx->S[0] = a_mul(ctx->S[0]) ^ ctx->S[13] ^ ainv_mul(ctx->S[11]);
793     fsmtmp = ctx->r[1] + ctx->S[13];
794     ctx->r[1] = T(ctx->r[0]);
795     ctx->r[0] = fsmtmp;
796     out[0] = in[0] ^ (ctx->r[0] + ctx->S[0]) ^ ctx->r[1] ^ ctx->S[1];
797 
798     ctx->S[1] = a_mul(ctx->S[1]) ^ ctx->S[14] ^ ainv_mul(ctx->S[12]);
799     fsmtmp = ctx->r[1] + ctx->S[14];
800     ctx->r[1] = T(ctx->r[0]);
801     ctx->r[0] = fsmtmp;
802     out[1] = in[1] ^ (ctx->r[0] + ctx->S[1]) ^ ctx->r[1] ^ ctx->S[2];
803 
```

Рис. 3.10 – Перші два такти функції next_stream_full_crypt

3.2.3 Програмна здійснення функції КП Strm

КП Strm обчислюється у функції next_stream_full_crypt при кожному такті(рис. 3.10). Цей потік відразу додається побітово по модулю 2 за формулою 5 із вихідним потоком бітів.

3.2.4 Програмна здійснення функції скінченного автомата CA

CA обчислюється у функції next_stream_full_crypt(рис. 3.10), dstu8845_init(рис. 3.8) при кожному такті. CA зберігається як масив 2 регістрів у структурі Dstu8845Ctx.

3.3 Приклад роботи додатку

На рисунку 3.11 представлено шматок коду файлу main.c. В ньому було зашифровано нульову 4096-байтову послідовність(рис. 3.11), час витрачений на шифрування, виведено на екран перші 128 байтів(рис. 3.12).

```
for (int i = 0; i < 1; i++) {
    dstu8845_crypt(ctx, out, 4096, out);
}

clock_t end = clock();
double time_spent = (double)(end - begin) / CLOCKS_PER_SEC;

printf("Encrypted 4096 byte in %f seconds\n", time_spent);

for(int i=0; i < 128; i++){
    printf("%d ", out[i]);
}
```

Рис. 3.11 – Частина коду файлу main.c

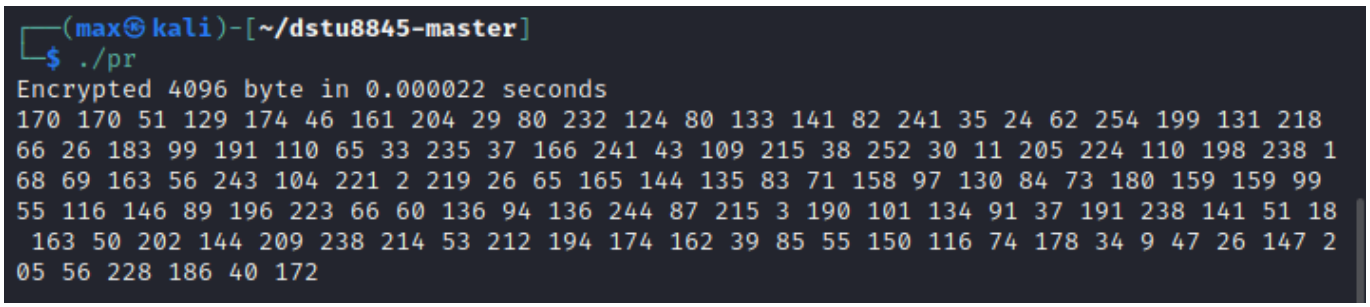

Рис. 3.12 – Вивід в консолі

3.4 Початки криптоаналізу шифра ДСТУ 8845:2019 (СТРУМОК)

Криптографія — це дисципліна, техніка та мистецтво створення закодованих повідомлень, секретних кодів чи ключів, а криптографічний аналіз є наукою й мистецтвом розшифрування повідомлень, зламу цих кодів. Окрім методів криптографії варто також вивчати й методи криптографічного аналізу.

І не через що це може допомогти з зламувати коди користувачів, а задля кращого вивчення вразливих місць у своїх криптографічних системах. Поглиблене вивчення криптоаналізу допомагає створювати кращі, більш надійні секретні коди. Існує чотири загальні типи атак криптографічного аналізу — їх проілюстровано на рис. 4.1

Здійснюючи атаку на зашифрований текст, у криптографічного аналітика є доступ тільки до певного зашифрованого тексту, та він пробує відшукати відповідний ключ та встановити вхідний текст. В цей час, за припущенням, аналітику відомий алгоритм, тому він може перехопити зашифроване повідомлення. Атака на зашифрований текст є найбільш імовірною, адже аналітику для її реалізації потрібен лише вихідний текст. При цьому типу атаки шифр має серйозно перешкоджати та не дозволяти дешифрування тексту противником.

Різні методи можуть використовуватись лише в атаці на зашифрований текст.

Атака «грубої сили»

Із застосуванням методу «грубої сили» криптографічний аналітик намагається використовувати усі варіанти можливих ключів. Припустимо, що йому відомий алгоритм та множина ключів (увесь список можливих варіантів). При застосуванні

даного методу зашифрований текст перехоплюється та використовуються усі можливі ключі до тих пір, поки вхідний текст не буде отримано. Раніше створення такої атаки було важкою справою; із застосуванням комп'ютерів це завдання значно спростилося. Не допустити такий тип атаки можна, якщо кількість можливих ключів буде дуже великим.

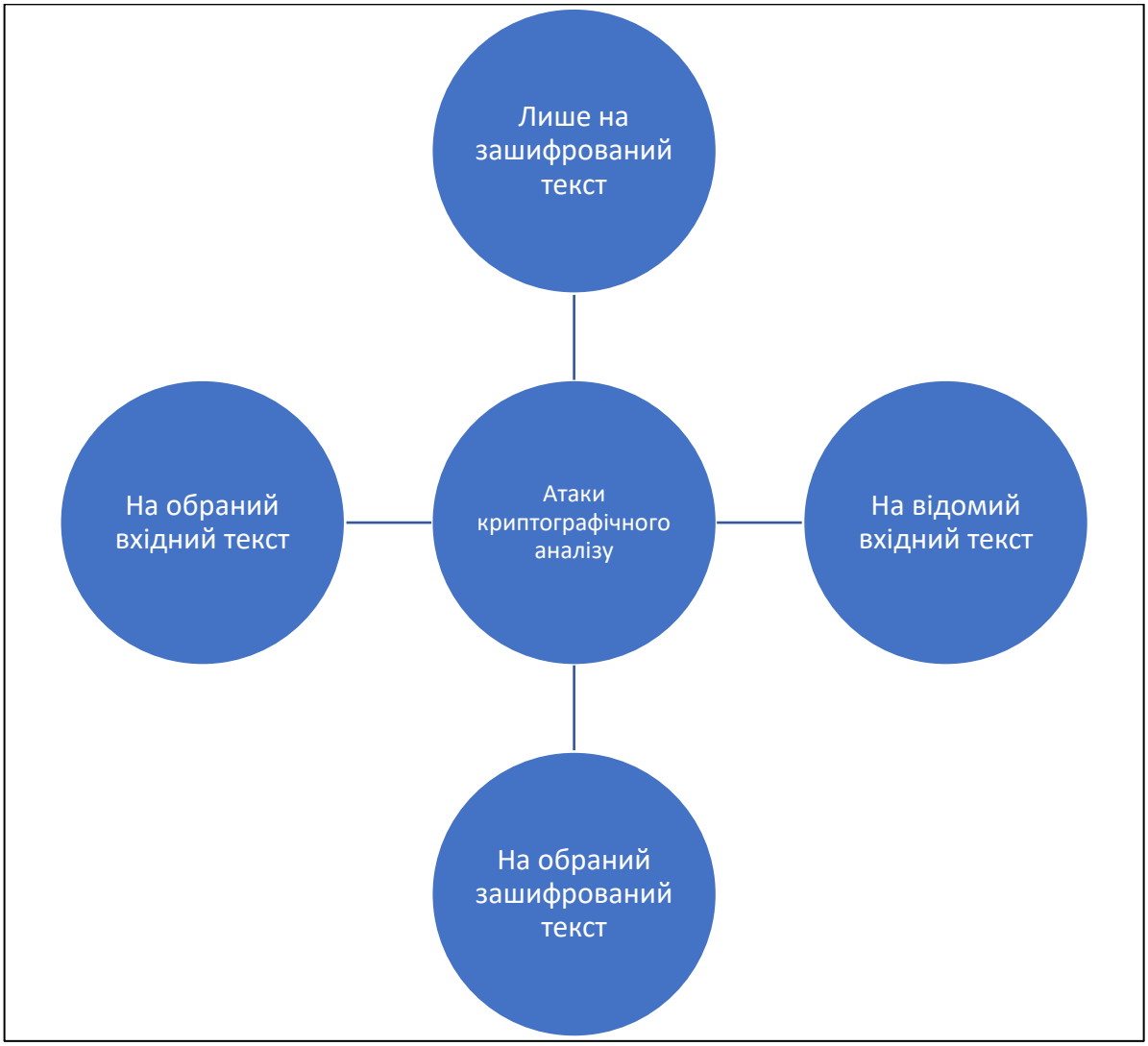

Рис. 3.13 – Класифікація алгоритмів шифрування

Статистична атака

Для криптографічного аналітика є можливість отримати вигоду з певних характеристик, що властиві мові вхідного тексту, задля початку статистичної атаки. До прикладу, нам відомо той факт, що літера Е є в англійському тексті найбільш часто використовуваною. Криптографічний аналітик шукає найчастіше використовуваний символ у зашифрованому тексті й допускає, що це й є літера Е.

Після того, як аналітик визначить декілька пар, він зможе відшукати ключ та розшифрувати повідомлення. Задля запобіганню цього типу атаки, шифр має маскувати ознаки мови.

Атака за зразком

Деякими шифрами характеристики мови приховуються, проте створюються певні зразки у зашифрованому тексті. Криптографічним аналітиком з метою зламання шифру можуть використовуватися атаки за зразком. Через це важливо застосовувати шифри, що здатні зробити зашифрований текст, який в даний час проглядається, невизначеним (абстрактним) максимально.

Під час здійснення атаки на знання вхідного тексту у криптоаналітика є доступ до певних пар «вхідний / зашифрований текст», а не лише перехоплених зашифрований текст.

Пари вхідного / зашифрованого тексту було зібрано раніше. Приклад: відправником було передано таємне повідомлення певному одержувачу, проте ним пізніше було відкрито зміст повідомлення також стороннім. Криптографічним аналітиком було збережено зашифровано та вхідний тексти задля використання їх тоді, коли йому знадобиться зламати послідуюче таємне повідомлення від відправника до одержувача. Під час цього він припускатиме ймовірність того, що відправник свій ключ не змінив, та застосовує відношення між попередньою парою задля аналізу поточного зашифрованого тексту.

Тут можуть застосовуватися такі ж методи, що й при атаці лише зашифрованого тексту. Проте дану атаку зробити простіше, адже у криптографічного аналітика вже є для аналізу більше інформації. Хоча, може статися така ситуація, що відправником вже було змінено свій ключ чи зміст будь-яких попередніх повідомлень ним не розкривалося, — в даному випадку таку атаку здійснити буде неможливою.

Атака з вибіркою вхідного тексту схожа на атаку знання вхідного тексту, проте пари «вхідний / зашифрований текст» обиралися та виготовлялися самим нападником.

Таке може статися у випадку, коли у криптографічного аналітика є доступ до комп'ютера відправника. Тоді їм вибирається певний вхідний текст та створюється зашифрований текст із використанням комп'ютера. В нього немає ключа, адже він зазвичай включений у ПЗ, яке використовується відправником.

Даний тип атаки здійснити помітно простіше, проте він є найменш ймовірним, адже містить забагато невідомих.

Атакування обраного зашифрованого тексту подібне до атаки на обраний вхідний текст, з одним лише винятком: криптографічним аналітиком вибирається і розшифровується певний зашифрований текст задля сформування пари «зашифрований / вхідний текст» (це відбувається у ситуаціях, коли у аналітика є доступ до комп'ютера відправника).

3.4.1 Кореляційний аналіз

Основні кореляційні атаки на заданий в темі дипломного проекту ГЗ по середній часовій складності мають більше 2^{24940} знаків гами та потребує більше 2^{24938} знаків гами. З цього можна зробити висновок, що заданий шифр має практичну стійкість за умови, відрізок гами має довжину, виробленому при будь-якому константному ключі, менше 2^{80}.

3.4.2 Швидкість шифрування

У файл main.h було додано програмну реалізацію обчислення часових витрат при зашифрування кількості даних за таблицею 4.1. На рисунку 4.2 представлено лістинг доданого коду. При запуску програми на консолі, зявляється відповідні значення часу для виділеної памяті, що зображено на рисунках 4.3 та 4.4. Ці дані були записані до таблиці 4.1. На основі цієї вибірки було побудовано гістограми на рисунках 4.5 та 4.6. З графіків випливає, що залежність гіперболічною.

```
32
33      int a=1;
34      for(int item=1; item < 2; item++){
35          long int countOfBit = arr[item];
36          double avr = 0;
37
38          for(int countUmn = 1; countUmn < 1024; countUmn +=32)
39          {
40                for(int g=0;g<1;g++)
41                {
42                    uint8_t out[countOfBit];
43                    memset( out, 0, countOfBit * sizeof(uint8_t) );
44
45                    clock_t begin = clock();
46                    for (int i = 0; i < countUmn; i++)
47                    {
48                        dstu8845_crypt(ctx, out, countOfBit, out);
49                    }
50                    clock_t end = clock();
51
52                    double time_spent = (double)(end - begin) / CLOCKS_PER_SEC;
53                    avr += time_spent;
54                 }
55
56              printf("%.20f s. %d%s\n", avr/1, countUmn, arrName[item]);
57          }
58      }
59      dstu8845_free(ctx);
```

Рис. 3.14 – Лістинг програми

```
┌──(max㉿kali)-[~/dstu8845-master]
└─$ ./pr
0.00002100000000000000 s. 1kB
0.00109599999999999997 s. 65kB
0.00327500000000000013 s. 129kB
0.00619199999999999959 s. 193kB
0.00948599999999999964 s. 257kB
0.01362999999999999975 s. 321kB
0.01865400000000000044 s. 385kB
0.02455099999999999991 s. 449kB
0.03132499999999999868 s. 513kB
0.03971700000000000230 s. 577kB
0.04801600000000000312 s. 641kB
0.05732500000000000095 s. 705kB
0.06751200000000000256 s. 769kB
0.07855900000000000383 s. 833kB
0.09102899999999999880 s. 897kB
0.10442200000000000093 s. 961kB
```

Рис. 3.15 – Лістинг консолі при виміру в kB

```
┌──(max㉿kali)-[~/dstu8845-master]
└─$ ./pr
0.01477900000000000047 s. 1MB
0.91106799999999998896 s. 65MB
2.88468799999999969685 s. 129MB
5.70130399999999948335 s. 193MB
9.33327899999999921477 s. 257MB
13.81088999999999877843 s. 321MB
19.10414699999999754709 s. 385MB
25.21580699999999808369 s. 449MB
32.13728199999999901593 s. 513MB
40.00648699999999990951 s. 577MB
48.61986300000000227328 s. 641MB
58.15075699999999869760 s. 705MB
68.50684200000000600994 s. 769MB
79.80263899999999921420 s. 833MB
95.17086700000000121236 s. 897MB
117.42795499999999719876 s. 961MB
```

Рис. 3.16 – Лістинг консолі при виміру в MB

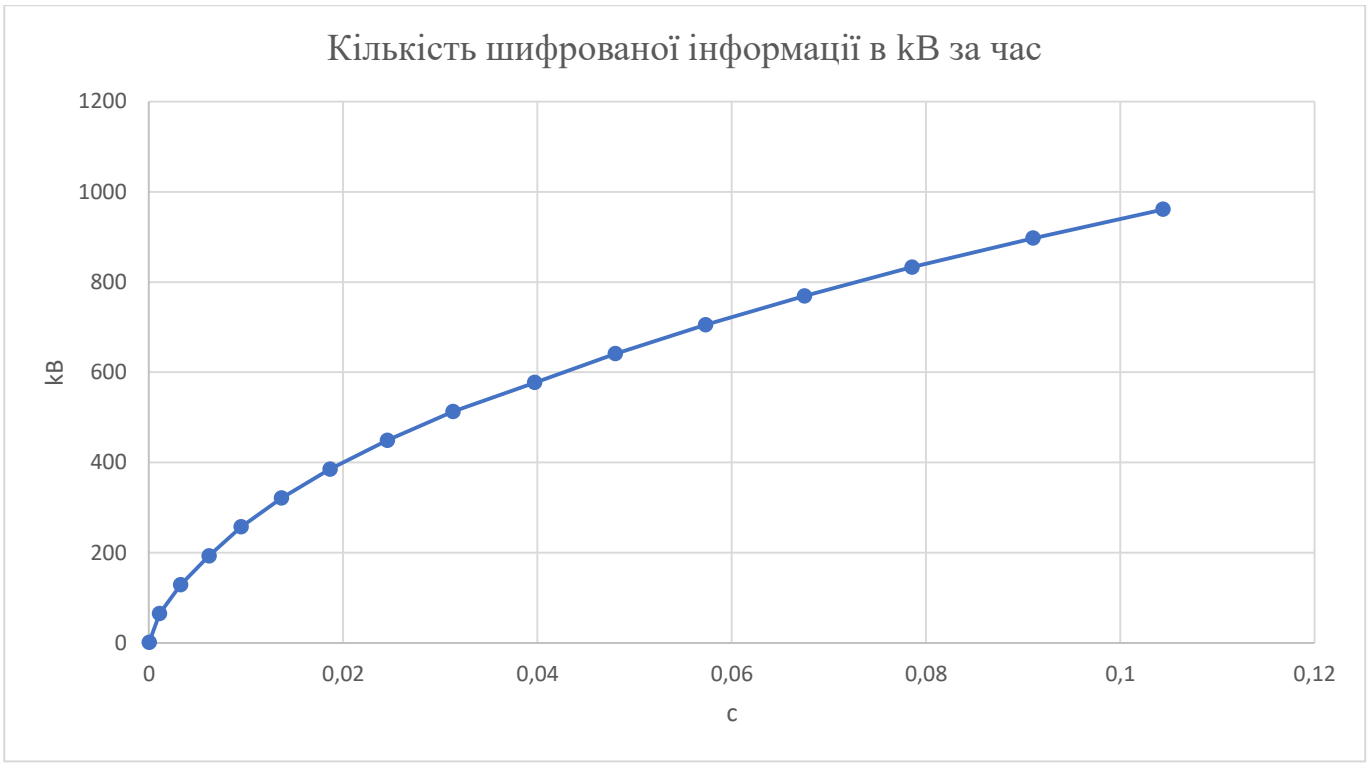

Рис. 3.17 – Кількість шифрованої інформації в kB за час

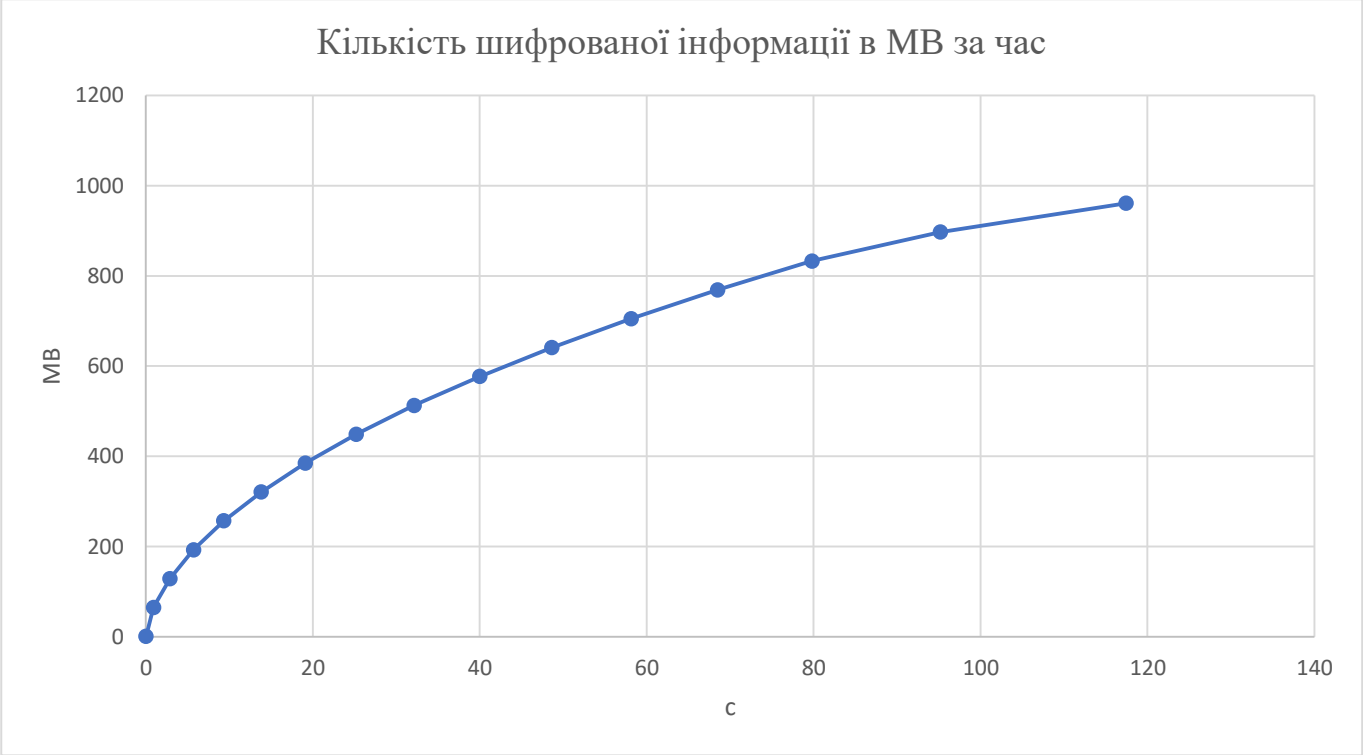

Рис. 3.18 – Кількість шифрованої інформації в МВ за час

Таблиця 3.1

Часові витрати

с	kB
0,000021	1
0,001096	65
0,003275	129
0,006192	193
0,009486	257
0,01363	321
0,018654	385
0,024551	449
0,031325	513
0,039717	577
0,048016	641
0,057325	705
0,067512	769

продовження таблиці 3.1

0,078559	833
0,091029	897
0,104422	961
0,014779	1
0,911068	65
2,884688	129
5,701304	193
9,333279	257
13,81089	321
19,104147	385
25,215807	449
32,137282	513
40,006487	577
48,619863	641
58,150757	705
68,506842	769
79,802639	833
95,170867	897
117,427955	961

Висновки за розділом 3

У з розділі програмно реалізовано шифр ДСТУ 8845:2019 (СТРУТОК). Програмний код розміщений на трьох файлах. Програма написана функціональним способом. Також було розглянуто сучасні види криптоатак, а також обґрунтовано кореляційний аналіз. При доповненні програмної реалізації було проведено вимір обробки шифрованих даних та побудовано гістограму по ним. Що дало гіперболічну залежність між часом і об’ємом пам’яті.

РОЗДІЛ 4
ПОЧАТКИ КРИПТОАНАЛІЗУ ШИФРА ДСТУ 8845:2019 (СТРУТОК)

4.1 Розробка ip-телефонії

Для реалізації ip-телефонії вибрано мережу і розроблено клієнт-серверний додаток. Мережа побудована за топологією зірка, схема якої зображена на рисунку 4.1.

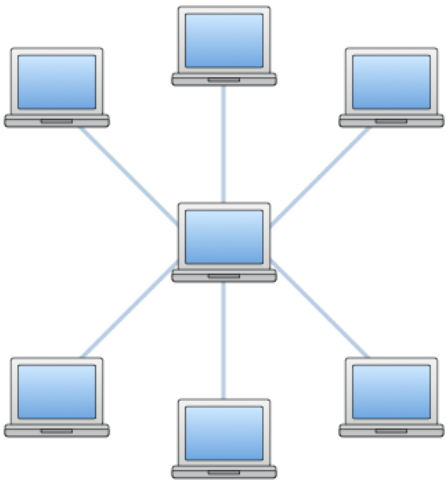

Рис. 4.1 – Схема мережі

Лістинг програми серверу складається з 5 функцій, прототипи яких зображено на рисунку 4.2:

- encryption – шифрування бітів;
- my_signal_handler – вихід із програми;
- grim_reap – очікування підключення
- loop_write - виклик команди UNIX write() у циклі;
- main – створення, отримання передавання голосових даних.

```
45 > void encryption(uint8_t*data){…
48   }
49 > void my_signal_handler(int sig_num) {…
60   }
61 > void grim_reap(int sig) {…
65   }
66 > static ssize_t loop_write(int fd, void*data, size_t size) {…
80   }
81 > int main(int argc, char *argv[]) {…
243  }
```

Рис. 4.2 – Функції серверу

Лістинг програми клієнту аналогічний клієнту за винятком функції grim_reap(див. рис. 4.3).

```
25 > void encryption(uint8_t*data){…
47   }
48 > void my_signal_handler(int sig_num) {…
59   }
60 > static ssize_t loop_write(int fd, const void*data, size_t size) {…
73   }
74 > int main(int argc, char *argv[]) {…
194  }
```

Рис. 4.3 – Функції клієнту

4.2 Серверна частина

Для запуску серверної частини додатку потрібно запустити виконувальний файл server.exe, приклад якого наведено на рисунку 4.1.

```
max@max:~/voip$ ./server
```

Рис. 4.4 – Запуск серверу

4.3 Клієнтська частина

Для запуску клієнта потрібно як аргументи передати ip-адресу серверу та відкритий порт, на якому він працює. Відповідно ip-адресу, було знайдено командою ifconfig. Приклад запуску команди зображено на рисунку 4.2.

```
max@max:~/voip$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.80.144  netmask 255.255.255.0  broadcast 192.168.80.255
```

Рис. 4.5 – Адреса серверу

Для знаходження відкритого порту сервера, було просканово усі порти за ip-адресою за допомогою команди, зображеної на рисунку 4.3, та встановлено відповідно для цієї системи порт 5000.

```
max@max:~/voip$ nmap 192.168.80.144
Starting Nmap 7.80 ( https://nmap.org ) at 2022-06-25 01:59 EEST
Nmap scan report for max (192.168.80.144)
Host is up (0.000080s latency).
Not shown: 999 closed ports
PORT     STATE SERVICE
5000/tcp open  upnp
```

Рис. 4.6 – Порт серверу

Для запуску клієнта потрібно прописати команду, зображену на рисунку 4.4.

```
max@max:~/voip$ ./client 192.168.80.144 5000
```

Рис. 4.7 – Запуск клієнту

4.4 Впровадження шифрування методом ДСТУ 8845:2019 (СТРУМОК) в розроблену ip-телефонію

Реалізований програмним чином в Розділі 3 бібліотеку шифру підключено до лістингу і клієнтської частини і серверної, за рисунком 4.8.

```
17    #include <strumok/strumoks.h>
```

Рис. 4.8 – Підключення бібліотеки шифрування

В обидвох частинах як при читані голосових даних так і при відправлені поточний буфер шифрується(див. рис. 4.9) і розшифровується за допомогою функції encryption, лістинг якої зображено на рисунку 4.10.

```
encryption((uint8_t*)buf);
```

Рис. 4.9 – Шифрування буферу

```
void encryption(uint8_t*data){
 Dstu8845Ctx *ctx = dstu8845_alloc();
    uint64_t key[8], iv[4];

    iv[0] = 1;
    iv[1] = 2;
    iv[2] = 3;
    iv[3] = 4;

    uint8_t key_size = 64;

    key[7] = 0x8000000000000000;
    key[6] = 0x0000000000000000;
    key[5] = 0x0000000000000000;
    key[4] = 0x0000000000000000;
    key[3] = 0x0000000000000000;
    key[2] = 0x0000000000000000;
    key[1] = 0x0000000000000000;
    key[0] = 0x0000000000000000;

    dstu8845_init(ctx, key, key_size, iv);
    dstu8845_crypt(ctx, data, countOfBit, data);
}
```

Рис. 4.10 – Лістинг функції шифрування

Висновки за розділом 4

У даному розділі було програмно розроблено та реалізовано ip-телефонію та впроваджено методи шифрування в неї.

ВИСНОВКИ

При розробці програмного додатку було досліджено основні принципи, за якими працюють потокові шифри з регістром зсуву з лінійним зворотнім зв’язком, проаналізовано часову залежність від кількості шифрованих даних, а також роботу ір-телефонії з використанням механізмів шифрування оцифрованих повідомлень перед передаванням по каналу передачі даних. У ході аналізу існуючих алгоритмів шифрування досліджено основні відмінності між ними. Досліджено також основу, за яку було взято в досліджуваному шифрі. Доповнено програмну реалізацію шифру ДСТУ 8845:2019, що дало зможу на практиці переглянути його застосування при різній обробці кількості вхідних даних, а також його використання в інформаційно-телекомунікаційній системі.

Додаток написаний на мові програмування С, що дозволяє запустити на будь якій ОС. Програмування було покладене функціональне, відповідно шуфрування відбувалось відразу по 8 байтів у різних регістрах, що теж в свою чергу дає виграш у часі. Дослідження показало гіперболічну залежність кількості обробки даних від затраченого часу. Впровадження шифру в ір-телефонію дало змогу впровадження криптографічного захисту передачі даних в каналах зв’язку.

В подальшому доповнений код в програмну реалізацію шифру ДСТУ 8845:2019 можна винести окремою бібліотекою для аналізу будь-якого алгоритму шифрування. За вихідними даними можна сформувати функцію залежності інформації від часу. Також написати додаток, що зрівнює цю функцію з практичним значенням в комплексних системах захисту та при відхилені даних повідомляти про це. Розроблену реалізацію ір-телефонії можна в майбутньому доповнити шлюзами для під’єднання апаратних телефонів та розробити інтерфейс користувача для простого використання системою. Після чого систему можна використовувати для конфіденційних комунікацій як держорганів так і в на ринку праці.

Матеріали роботи є продовженням дослідницької роботи по темі «шифр SNOW 2.0» на конференції «Інтеграція інформаційних систем і інтелектуальних

технологій в умовах трансформації інформаційного суспільства» (IV Міжнародна конференція "Інтеграція інформаційних систем і інтелектуальних технологій в умовах трансформації інформаційного суспільства", присвячена 50-річчю кафедри інформаційних систем та технологій ПДАУ, Полтава, 21-22.10.2021 року).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Інформаційні технології. Криптографічний захист інформації. Алгоритм симетричного потокового перетворення. // ДСТУ. – 2019. – №8845.

2. Інформаційні технології. Методи захисту. Алгоритми шифрування. Частина 4. Потокові шифри (ISO/IEC 18033-4:2011, IDT). // ДСТУ ISO/IEC. – 2015. – №18033.

3. Інформаційні технології. Криптографічний захист інформації. Функція гешування. З поправкою. // ДСТУ. – 2014. – №7564.

4. Інформаційні технології. Криптографічний захист інформації. Функція гешування. З поправкою. // ДСТУ. – 2014. – №7564.

5. Криптографічні атаки: [Електронний ресурс] // Хабр. – 2019. – Режим доступу до ресурсу: https://habr.com/ru/post/462437/.

6. Chepyzhov, Vladimor V., Thomas Johansson, and Ben Smeets. A simple algorithm for fast correlation attacks on stream ciphers. International Workshop on Fast Software Encryption. Springer, Berlin, Heidelberg, 2000.

7. Zhang, Bin, and Dengguo Feng. Multi-pass fast correlation attack on stream ciphers. International Workshop on Selected Areas in Cryptography.

8. Chose, Philippe, Antoine Joux, and Michel Mitton. Fast correlation attacks: An algorithmic point of view. International Conference on the Theory and Applications of Cryptographic Techniques. Springer, Berlin, Heidelberg, 2002.

9. Meier, Willi, and Othmar Staffelbach. Fast correlation attacks on certain stream ciphers. Journal of cryptology 1.3 (1989)

10. Todo, Yosuke, Willi Meier, and Kazumaro Aoki. On the Data Limitation of Small-State Stream Ciphers: Correlation Attacks on Fruit-80 and Plantlet. International Conference on Selected Areas in Cryptography. Springer, Cham, 2019.

11. В.В. Багрій. Застосування криптоаналізу для дослідження стійкості систем захисту інформації / В.В. Багрій, О.П. До́ренський. // Всеукраїнський студентський науково-практичний семінар. – 2012. – С. 58–60.

12. Шрамченко Б.Л. АНАЛІЗ ПОСЛІДОВНОСТЕЙ НА ВИХОДІ ЗСУВНОГО РЕГІСТРУ З ЛІНІЙНИМ ЗВОРОТНИМ ЗВ’ЯЗКОМ [Електронний ресурс] / Шрамченко Б.Л. – Режим доступу до ресурсу: https://knutd.edu.ua/publications/pdf/Ukrainian_editions/st_Shramchenko.pdf.

13. Дегтяров А. К. КРИПТОГРАФІЯ: ЗАГАЛЬНІ ВИЗНАЧЕННЯ, КЛАСИФІКАЦІЯ.АСИМЕТРИЧНІ ТА СИМЕТРИЧНІ КРИПТОАЛГОРИТМИ, ЇХ ПОРІВНЯННЯ. [Електронний ресурс] / Дегтяров А. К.. – 2014. – Режим доступу до ресурсу: https://dehtyarov09.wordpress.com/2014/03/16/%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D1%96%D1%8F-%D0%B7%D0%B0%D0%B3%D0%B0%D0%BB%D1%8C%D0%BD%D1%96-%D0%B2%D0%B8%D0%B7%D0%BD%D0%B0%D1%87%D0%B5%D0%BD%D0%BD%D1%8F-%D0%BA%D0%BB-2/.

14. Chapter 4. Finite Fields [Електронний ресурс] // Imperial College London – Режим доступу до ресурсу: https://www.doc.ic.ac.uk/~mrh/330tutor/ch04s04.html.

15. А.Е.Лагун. ГЕНЕРАТОРИ ПСЕВДОВИПАДКОВИХ ПОСЛІДОВНОСТЕЙ НА ОСНОВІ РЕГІСТРІВ ЗСУВУ З ЛІНІЙНИМ ЗВОРОТНИМ ЗВЯЗКОМ / А.Е.Лагун, Ю.М.Костів. // НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”. – 2012.

16. R.E. Blahut. Theory and Practice of Error Control Codes. Addison-Wesley, 1983

17. A. Klapper and M. Goresky. Feedback Shift Registers, 2-Adic Span, and Combiners with Memory, Journal of Cryptology (1997) 10: 111-147.

18. R. Lidl and H. Niederreiter, "Finite Fields," in Encyclopedia of Mathematics and Its Applications, Vol. 20, Reading, MA: Addison-Wesley, 1983..

19. R. Lidl and H. Niederreiter, Introduction to Finite Fields and Their Applications, London, Cambridge University Press, 1986.

20. Б. Шнайер. Прикладная криптография. Протоколы, алгоритмы и исходные тексты на языке С, 2-ое издание.

21. Дискретная математика и криптология. Курс лекций/ Под общ. Ред. Н.Д. Подуфалова – М.:ДИАЛОГ-МИФИ, 2003 – 400 с.

22. Rock A. Pseudorandom Number Generators for Criptographic Applications / A. Rock. – Salzbuburg, 2005. – p. 57–65.

23. . Zenner E. On Cryptographic Properties of LFSR-based Pseudorandom Generators / E. Zenner. – Mannheim, 2004. – p. 102

24. Schneier B. Applied Cryptography Second Edition: protocols, algorithms and source code in C. John Wiley & Sons Inc., 1996

25. А. М. Олексійчук. ОБҐРУНТУВАННЯ СТІЙКОСТІ ПОТОКОВОГО ШИФРУ«СТРУМОК»ВІДНОСНО КОРЕЛЯЦІЙНИХ АТАК НАД СКІНЧЕННИМИ ПОЛЯМИ ХАРАКТЕРИСТИКИ 2 / А. М. Олексійчук, С. М. Конюшок, М. В. Поремський. // Математичнетакомп’ютернемоделювання. – 2019. – №621. – С. 114–119.

26. Ekdahl P., Johansson T. A new version of the stream cipher SNOW.Selected Areas in Cryptography—SAC.2002. LNCS 2295. Springer-Verlag. P. 47–61.

27. GorbenkoI., KuznetsovA., GorbenkoYu., AlekseychukA., TimchenkoV. StrumokKeystreamGenerator.The9thIEEEInternationalConference on De-pendable Systems, Services and Technologies, DESSERT’2018, 24–27 May, 2018.Kyiv, Ukraine. P. 292–299

28. Lee J.-K., Lee D.H., Park S. Cryptanalysis of SOSEMANUC and SNOW 2.0 using linear masks.Advanced in Cryptology.ASIACRYPT 2008. LNCS 5350. Springer-Verlag. P. 524–538.

29. Олексійчук А.М., Поремський М.В. Загальна схема побудови кореля-ційних атак на SNOW 2.0-подібні потокові шифри.Правове, нормативне та метрологічне забезпечення системи захисту інформації в Україні. Вип. 1(35). 2018. С. 70–79

ДОДАТОК А

Файл strumok.h

```
#include <stdint.h>
#include <string.h>

typedef struct Dstu8845Ctx_st Dstu8845Ctx;

Dstu8845Ctx *dstu8845_alloc();

int dstu8845_init(Dstu8845Ctx *ctx, const uint64_t *key, uint8_t key_size, const uint64_t *iv);
int dstu8845_set_iv(Dstu8845Ctx *ctx, const uint64_t *iv);
int dstu8845_crypt(Dstu8845Ctx *ctx, const uint8_t *in, size_t inl, uint8_t *out);
void dstu8845_free(Dstu8845Ctx *ctx);
```

ДОДАТОК Б

Файл strumok.c

```
#include "strumok.h"
#include <stdlib.h>

#define byte(n,w)   (((w)>>(n*8)) & 0xff)
#define ainv_mul(w) (((w)>>8)^(strumok_alphainv_mul[w&0xff]))
#define a_mul(w)    (((w)<<8)^(strumok_alpha_mul[w>>56]))
#define T(w) ((T0[byte(0,(w))])^(T1[byte(1,(w))])^(T2[byte(2,(w))])^(T3[byte(3,(w))])^(T4[byte(4,(w))])^(T5[byte(5,(w))])^(T6[byte(6,(w))])^(T7[byte(7,(w))]))

static uint64_t strumok_alpha_mul[256] = {
  0x0000000000000000ULL, 0xD73F04125E000004ULL, 0xB37E0824BC000008ULL, 0x64410C36E200000CULL,
  0x7BFC104865000010ULL, 0xACC3145A3B000014ULL, 0xC882186CD9000018ULL, 0x1FBD1C7E8700001CULL,
  0xF6E52090CA000020ULL, 0x21DA248294000024ULL, 0x459B28B476000028ULL, 0x92A42CA62800002CULL,
  0x8D1930D8AF000030ULL, 0x5A2634CAF1000034ULL, 0x3E6738FC13000038ULL, 0xE9583CEE4D00003CULL,
  0xF1D7403D89000040ULL, 0x26E8442FD7000044ULL, 0x42A9481935000048ULL, 0x95964C0B6B00004CULL,
  0x8A2B5075EC000050ULL, 0x5D145467B2000054ULL, 0x3955585150000058ULL, 0xEE6A5C430E00005CULL,
  0x073260AD43000060ULL, 0xD00D64BF1D000064ULL, 0xB44C6889FF000068ULL, 0x63736C9BA100006CULL,
  0x7CCE70E526000070ULL, 0xABF174F778000074ULL, 0xCFB078C19A000078ULL, 0x188F7CD3C400007CULL,
  0xFFB3807A0F000080ULL, 0x288C846851000084ULL, 0x4CCD885EB3000088ULL, 0x9BF28C4CED00008CULL,
  0x844F90326A000090ULL, 0x5370942034000094ULL, 0x37319816D6000098ULL, 0xE00E9C048800009CULL,
  0x0956A0EAC50000A0ULL, 0xDE69A4F89B0000A4ULL, 0xBA28A8CE790000A8ULL, 0x6D17ACDC270000ACULL,
  0x72AAB0A2A00000B0ULL, 0xA595B4B0FE0000B4ULL, 0xC1D4B8861C0000B8ULL, 0x16EBBC94420000BCULL,
  0x0E64C047860000C0ULL, 0xD95BC455D80000C4ULL, 0xBD1AC8633A0000C8ULL, 0x6A25CC71640000CCULL,
  0x7598D00FE30000D0ULL, 0xA2A7D41DBD0000D4ULL, 0xC6E6D82B5F0000D8ULL, 0x11D9DC39010000DCULL,
  0xF881E0D74C0000E0ULL, 0x2FBEE4C5120000E4ULL, 0x4BFFE8F3F00000E8ULL, 0x9CC0ECE1AE0000ECULL,
  0x837DF09F290000F0ULL, 0x5442F48D770000F4ULL, 0x3003F8BB950000F8ULL, 0xE73CFCA9CB0000FCULL,
  0xE37B1DF41E00001DULL, 0x344419E640000019ULL, 0x500515D0A2000015ULL, 0x873A11C2FC000011ULL,
  0x98870DBC7B00000DULL, 0x4FB809AE25000009ULL, 0x2BF90598C7000005ULL, 0xFCC6018A99000001ULL,
  0x159E3D64D400003DULL, 0xC2A139768A000039ULL, 0xA6E0354068000035ULL, 0x71DF315236000031ULL,
  0x6E622D2CB100002DULL, 0xB95D293EEF000029ULL, 0xDD1C25080D000025ULL, 0x0A23211A53000021ULL,
  0x12AC5DC99700005DULL, 0xC59359DBC9000059ULL, 0xA1D255ED2B000055ULL, 0x76ED51FF75000051ULL,
  0x69504D81F200004DULL, 0xBE6F4993AC000049ULL, 0xDA2E45A54E000045ULL, 0x0D1141B710000041ULL,
  0xE4497D595D00007DULL, 0x3376794B03000079ULL, 0x5737757DE1000075ULL, 0x8008716FBF000071ULL,
  0x9FB56D113800006DULL, 0x488A690366000069ULL, 0x2CCB653584000065ULL, 0xFBF46127DA000061ULL,
  0x1CC89D8E1100009DULL, 0xCBF7999C4F000099ULL, 0xAFB695AAAD000095ULL, 0x788991B8F3000091ULL,
  0x67348DC67400008DULL, 0xB00B89D42A000089ULL, 0xD44A85E2C8000085ULL, 0x037581F096000081ULL,
  0xEA2DBD1EDB0000BDULL, 0x3D12B90C850000B9ULL, 0x5953B53A670000B5ULL, 0x8E6CB128390000B1ULL,
  0x91D1AD56BE0000ADULL, 0x46EEA944E00000A9ULL, 0x22AFA572020000A5ULL, 0xF590A1605C0000A1ULL,
  0xED1FDDB3980000DDULL, 0x3A20D9A1C60000D9ULL, 0x5E61D597240000D5ULL, 0x895ED1857A0000D1ULL,
  0x96E3CDFBFD0000CDULL, 0x41DCC9E9A30000C9ULL, 0x259DC5DF410000C5ULL, 0xF2A2C1CD1F0000C1ULL,
```

```
    0x1BFAFD23520000FDULL, 0xCCC5F9310C0000F9ULL, 0xA884F507EE0000F5ULL, 0x7FBBF115B00000F1ULL,
    0x6006ED6B370000EDULL, 0xB739E979690000E9ULL, 0xD378E54F8B0000E5ULL, 0x0447E15DD50000E1ULL,
    0xDBF63AF53C00003AULL, 0x0CC93EE76200003EULL, 0x688832D180000032ULL, 0xBFB736C3DE000036ULL,
    0xA00A2ABD5900002AULL, 0x77352EAF0700002EULL, 0x13742299E5000022ULL, 0xC44B268BBB000026ULL,
    0x2D131A65F600001AULL, 0xFA2C1E77A800001EULL, 0x9E6D12414A000012ULL, 0x4952165314000016ULL,
    0x56EF0A2D9300000AULL, 0x81D00E3FCD00000EULL, 0xE59102092F000002ULL, 0x32AE061B71000006ULL,
    0x2A217AC8B500007AULL, 0xFD1E7EDAEB00007EULL, 0x995F72EC09000072ULL, 0x4E6076FE57000076ULL,
    0x51DD6A80D000006AULL, 0x86E26E928E00006EULL, 0xE2A362A46C000062ULL, 0x359C66B632000066ULL,
    0xDCC45A587F00005AULL, 0x0BFB5E4A2100005EULL, 0x6FBA527CC3000052ULL, 0xB885566E9D000056ULL,
    0xA7384A101A00004AULL, 0x70074E024400004EULL, 0x14464234A6000042ULL, 0xC3794626F8000046ULL,
    0x2445BA8F330000BAULL, 0xF37ABE9D6D0000BEULL, 0x973BB2AB8F0000B2ULL, 0x4004B6B9D10000B6ULL,
    0x5FB9AAC7560000AAULL, 0x8886AED5080000AEULL, 0xECC7A2E3EA0000A2ULL, 0x3BF8A6F1B40000A6ULL,
    0xD2A09A1FF900009AULL, 0x059F9E0DA700009EULL, 0x61DE923B45000092ULL, 0xB6E196291B000096ULL,
    0xA95C8A579C00008AULL, 0x7E638E45C200008EULL, 0x1A22827320000082ULL, 0xCD1D86617E000086ULL,
    0xD592FAB2BA0000FAULL, 0x02ADFEA0E40000FEULL, 0x66ECF296060000F2ULL, 0xB1D3F684580000F6ULL,
    0xAE6EEAFADF0000EAULL, 0x7951EEE8810000EEULL, 0x1D10E2DE630000E2ULL, 0xCA2FE6CC3D0000E6ULL,
    0x2377DA22700000DAULL, 0xF448DE302E0000DEULL, 0x9009D206CC0000D2ULL, 0x4736D614920000D6ULL,
    0x588BCA6A150000CAULL, 0x8FB4CE784B0000CEULL, 0xEBF5C24EA90000C2ULL, 0x3CCAC65CF70000C6ULL,
    0x388D270122000027ULL, 0xEFB223137C000023ULL, 0x8BF32F259E00002FULL, 0x5CCC2B37C000002BULL,
    0x4371374947000037ULL, 0x944E335B19000033ULL, 0xF00F3F6DFB00003FULL, 0x27303B7FA500003BULL,
    0xCE680791E8000007ULL, 0x19570383B6000003ULL, 0x7D160FB55400000FULL, 0xAA290BA70A00000BULL,
    0xB59417D98D000017ULL, 0x62AB13CBD3000013ULL, 0x06EA1FFD3100001FULL, 0xD1D51BEF6F00001BULL,
    0xC95A673CAB000067ULL, 0x1E65632EF5000063ULL, 0x7A246F181700006FULL, 0xAD1B6B0A4900006BULL,
    0xB2A67774CE000077ULL, 0x6599736690000073ULL, 0x01D87F507200007FULL, 0xD6E77B422C00007BULL,
    0x3FBF47AC61000047ULL, 0xE88043BE3F000043ULL, 0x8CC14F88DD00004FULL, 0x5BFE4B9A8300004BULL,
    0x444357E404000057ULL, 0x937C53F65A000053ULL, 0xF73D5FC0B800005FULL, 0x20025BD2E600005BULL,
    0xC73EA77B2D0000A7ULL, 0x1001A369730000A3ULL, 0x7440AF5F910000AFULL, 0xA37FAB4DCF0000ABULL,
    0xBCC2B733480000B7ULL, 0x6BFDB321160000B3ULL, 0x0FBCBF17F40000BFULL, 0xD883BB05AA0000BBULL,
    0x31DB87EBE7000087ULL, 0xE6E483F9B9000083ULL, 0x82A58FCF5B00008FULL, 0x559A8BDD0500008BULL,
    0x4A2797A382000097ULL, 0x9D1893B1DC000093ULL, 0xF9599F873E00009FULL, 0x2E669B956000009BULL,
    0x36E9E746A40000E7ULL, 0xE1D6E354FA0000E3ULL, 0x8597EF62180000EFULL, 0x52A8EB70460000EBULL,
    0x4D15F70EC10000F7ULL, 0x9A2AF31C9F0000F3ULL, 0xFE6BFF2A7D0000FFULL, 0x2954FB38230000FBULL,
    0xC00CC7D66E0000C7ULL, 0x1733C3C4300000C3ULL, 0x7372CFF2D20000CFULL, 0xA44DCBE08C0000CBULL,
    0xBBF0D79E0B0000D7ULL, 0x6CCFD38C550000D3ULL, 0x088EDFBAB70000DFULL, 0xDFB1DBA8E90000DBULL
};

static uint64_t strumok_alphainv_mul[256] = {
    0x0000000000000000ULL, 0x47FCC6018A990000ULL, 0x8EE59102092F0000ULL, 0xC919570383B60000ULL,
    0x01D73F04125E0000ULL, 0x462BF90598C70000ULL, 0x8F32AE061B710000ULL, 0xC8CE680791E80000ULL,
    0x02B37E0824BC0000ULL, 0x454FB809AE250000ULL, 0x8C56EF0A2D930000ULL, 0xCBAA290BA70A0000ULL,
    0x0364410C36E20000ULL, 0x4498870DBC7B0000ULL, 0x8D81D00E3FCD0000ULL, 0xCA7D160FB5540000ULL,
    0x047BFC1048650000ULL, 0x43873A11C2FC0000ULL, 0x8A9E6D12414A0000ULL, 0xCD62AB13CBD30000ULL,
    0x05ACC3145A3B0000ULL, 0x42500515D0A20000ULL, 0x8B49521653140000ULL, 0xCCB59417D98D0000ULL,
    0x06C882186CD90000ULL, 0x41344419E6400000ULL, 0x882D131A65F60000ULL, 0xCFD1D51BEF6F0000ULL,
    0x071FBD1C7E870000ULL, 0x40E37B1DF41E0000ULL, 0x89FA2C1E77A80000ULL, 0xCE06EA1FFD310000ULL,
    0x08F6E52090CA0000ULL, 0x4F0A23211A530000ULL, 0x8613742299E50000ULL, 0xC1EFB223137C0000ULL,
    0x0921DA2482940000ULL, 0x4EDD1C25080D0000ULL, 0x87C44B268BBB0000ULL, 0xC0388D2701220000ULL,
```

0x0A459B28B4760000ULL, 0x4DB95D293EEF0000ULL, 0x84A00A2ABD590000ULL, 0xC35CCC2B37C00000ULL,
0x0B92A42CA6280000ULL, 0x4C6E622D2CB10000ULL, 0x8577352EAF070000ULL, 0xC28BF32F259E0000ULL,
0x0C8D1930D8AF0000ULL, 0x4B71DF3152360000ULL, 0x82688832D1800000ULL, 0xC5944E335B190000ULL,
0x0D5A2634CAF10000ULL, 0x4AA6E03540680000ULL, 0x83BFB736C3DE0000ULL, 0xC443713749470000ULL,
0x0E3E6738FC130000ULL, 0x49C2A139768A0000ULL, 0x80DBF63AF53C0000ULL, 0xC727303B7FA50000ULL,
0x0FE9583CEE4D0000ULL, 0x48159E3D64D40000ULL, 0x810CC93EE7620000ULL, 0xC6F00F3F6DFB0000ULL,
0x10F1D7403D890000ULL, 0x570D1141B7100000ULL, 0x9E14464234A60000ULL, 0xD9E88043BE3F0000ULL,
0x1126E8442FD70000ULL, 0x56DA2E45A54E0000ULL, 0x9FC3794626F80000ULL, 0xD83FBF47AC610000ULL,
0x1242A94819350000ULL, 0x55BE6F4993AC0000ULL, 0x9CA7384A101A0000ULL, 0xDB5BFE4B9A830000ULL,
0x1395964C0B6B0000ULL, 0x5469504D81F20000ULL, 0x9D70074E02440000ULL, 0xDA8CC14F88DD0000ULL,
0x148A2B5075EC0000ULL, 0x5376ED51FF750000ULL, 0x9A6FBA527CC30000ULL, 0xDD937C53F65A0000ULL,
0x155D145467B20000ULL, 0x52A1D255ED2B0000ULL, 0x9BB885566E9D0000ULL, 0xDC444357E4040000ULL,
0x1639555851500000ULL, 0x51C59359DBC90000ULL, 0x98DCC45A587F0000ULL, 0xDF20025BD2E60000ULL,
0x17EE6A5C430E0000ULL, 0x5012AC5DC9970000ULL, 0x990BFB5E4A210000ULL, 0xDEF73D5FC0B80000ULL,
0x18073260AD430000ULL, 0x5FFBF46127DA0000ULL, 0x96E2A362A46C0000ULL, 0xD11E65632EF50000ULL,
0x19D00D64BF1D0000ULL, 0x5E2CCB6535840000ULL, 0x97359C66B6320000ULL, 0xD0C95A673CAB0000ULL,
0x1AB44C6889FF0000ULL, 0x5D488A6903660000ULL, 0x9451DD6A80D00000ULL, 0xD3AD1B6B0A490000ULL,
0x1B63736C9BA10000ULL, 0x5C9FB56D11380000ULL, 0x9586E26E928E0000ULL, 0xD27A246F18170000ULL,
0x1C7CCE70E5260000ULL, 0x5B8008716FBF0000ULL, 0x92995F72EC090000ULL, 0xD565997366900000ULL,
0x1DABF174F7780000ULL, 0x5A5737757DE10000ULL, 0x934E6076FE570000ULL, 0xD4B2A67774CE0000ULL,
0x1ECFB078C19A0000ULL, 0x593376794B030000ULL, 0x902A217AC8B50000ULL, 0xD7D6E77B422C0000ULL,
0x1F188F7CD3C40000ULL, 0x58E4497D595D0000ULL, 0x91FD1E7EDAEB0000ULL, 0xD601D87F50720000ULL,
0x20FFB3807A0F0000ULL, 0x67037581F0960000ULL, 0xAE1A228273200000ULL, 0xE9E6E483F9B90000ULL,
0x21288C8468510000ULL, 0x66D44A85E2C80000ULL, 0xAFCD1D86617E0000ULL, 0xE831DB87EBE70000ULL,
0x224CCD885EB30000ULL, 0x65B00B89D42A0000ULL, 0xACA95C8A579C0000ULL, 0xEB559A8BDD050000ULL,
0x239BF28C4CED0000ULL, 0x6467348DC6740000ULL, 0xAD7E638E45C20000ULL, 0xEA82A58FCF5B0000ULL,
0x24844F90326A0000ULL, 0x63788991B8F30000ULL, 0xAA61DE923B450000ULL, 0xED9D1893B1DC0000ULL,
0x2553709420340000ULL, 0x62AFB695AAAD0000ULL, 0xABB6E196291B0000ULL, 0xEC4A2797A3820000ULL,
0x2637319816D60000ULL, 0x61CBF7999C4F0000ULL, 0xA8D2A09A1FF90000ULL, 0xEF2E669B95600000ULL,
0x27E00E9C04880000ULL, 0x601CC89D8E110000ULL, 0xA9059F9E0DA70000ULL, 0xEEF9599F873E0000ULL,
0x280956A0EAC50000ULL, 0x6FF590A1605C0000ULL, 0xA6ECC7A2E3EA0000ULL, 0xE11001A369730000ULL,
0x29DE69A4F89B0000ULL, 0x6E22AFA572020000ULL, 0xA73BF8A6F1B40000ULL, 0xE0C73EA77B2D0000ULL,
0x2ABA28A8CE790000ULL, 0x6D46EEA944E00000ULL, 0xA45FB9AAC7560000ULL, 0xE3A37FAB4DCF0000ULL,
0x2B6D17ACDC270000ULL, 0x6C91D1AD56BE0000ULL, 0xA58886AED5080000ULL, 0xE27440AF5F910000ULL,
0x2C72AAB0A2A00000ULL, 0x6B8E6CB128390000ULL, 0xA2973BB2AB8F0000ULL, 0xE56BFDB321160000ULL,
0x2DA595B4B0FE0000ULL, 0x6A5953B53A670000ULL, 0xA34004B6B9D10000ULL, 0xE4BCC2B733480000ULL,
0x2EC1D4B8861C0000ULL, 0x693D12B90C850000ULL, 0xA02445BA8F330000ULL, 0xE7D883BB05AA0000ULL,
0x2F16EBBC94420000ULL, 0x68EA2DBD1EDB0000ULL, 0xA1F37ABE9D6D0000ULL, 0xE60FBCBF17F40000ULL,
0x300E64C047860000ULL, 0x77F2A2C1CD1F0000ULL, 0xBEEBF5C24EA90000ULL, 0xF91733C3C4300000ULL,
0x31D95BC455D80000ULL, 0x76259DC5DF410000ULL, 0xBF3CCAC65CF70000ULL, 0xF8C00CC7D66E0000ULL,
0x32BD1AC8633A0000ULL, 0x7541DCC9E9A30000ULL, 0xBC588BCA6A150000ULL, 0xFBA44DCBE08C0000ULL,
0x336A25CC71640000ULL, 0x7496E3CDFBFD0000ULL, 0xBD8FB4CE784B0000ULL, 0xFA7372CFF2D20000ULL,
0x347598D00FE30000ULL, 0x73895ED1857A0000ULL, 0xBA9009D206CC0000ULL, 0xFD6CCFD38C550000ULL,
0x35A2A7D41DBD0000ULL, 0x725E61D597240000ULL, 0xBB4736D614920000ULL, 0xFCBBF0D79E0B0000ULL,
0x36C6E6D82B5F0000ULL, 0x713A20D9A1C60000ULL, 0xB82377DA22700000ULL, 0xFFDFB1DBA8E90000ULL,
0x3711D9DC39010000ULL, 0x70ED1FDDB3980000ULL, 0xB9F448DE302E0000ULL, 0xFE088EDFBAB70000ULL,
0x38F881E0D74C0000ULL, 0x7F0447E15DD50000ULL, 0xB61D10E2DE630000ULL, 0xF1E1D6E354FA0000ULL,

```
    0x392FBEE4C5120000ULL, 0x7ED378E54F8B0000ULL, 0xB7CA2FE6CC3D0000ULL, 0xF036E9E746A40000ULL,
    0x3A4BFFE8F3F00000ULL, 0x7DB739E979690000ULL, 0xB4AE6EEAFADF0000ULL, 0xF352A8EB70460000ULL,
    0x3B9CC0ECE1AE0000ULL, 0x7C6006ED6B370000ULL, 0xB57951EEE8810000ULL, 0xF28597EF62180000ULL,
    0x3C837DF09F290000ULL, 0x7B7FBBF115B00000ULL, 0xB266ECF296060000ULL, 0xF59A2AF31C9F0000ULL,
    0x3D5442F48D770000ULL, 0x7AA884F507EE0000ULL, 0xB3B1D3F684580000ULL, 0xF44D15F70EC10000ULL,
    0x3E3003F8BB950000ULL, 0x79CCC5F9310C0000ULL, 0xB0D592FAB2BA0000ULL, 0xF72954FB38230000ULL,
    0x3FE73CFCA9CB0000ULL, 0x781BFAFD23520000ULL, 0xB102ADFEA0E40000ULL, 0xF6FE6BFF2A7D0000ULL
};

static uint64_t T0[256] = {
    0xA832A829D77F9AA8ULL, 0x4352432297D41143ULL, 0x5F3E5FC2DF80615FULL, 0x061E063014121806ULL,
    0x6BDA6B7F670CB16BULL, 0x75BC758F2356C975ULL, 0x6CC16C477519AD6CULL, 0x592059F2CB927959ULL,
    0x71A871AF3B4AD971ULL, 0xDF84DFB6F8275BDFULL, 0x87A1874C35B22687ULL, 0x95FB95DC59CC6E95ULL,
    0x174B17B872655C17ULL, 0xF017F0D31AEAE7F0ULL, 0xD89FD88EEA3247D8ULL, 0x092D0948363F2409ULL,
    0x6DC46D4F731EA96DULL, 0xF318F3CB10E3EBF3ULL, 0x1D691DE84E53741DULL, 0xCBC0CB16804B0BCBULL,
    0xC9CAC9068C4503C9ULL, 0x4D644D52B3FE294DULL, 0x2C9C2C7DE8C4B02CULL, 0xAF29AF11C56A86AFULL,
    0x798079EF0B72F979ULL, 0xE047E0537A9AA7E0ULL, 0x97F197CC55C26697ULL, 0xFD2EFDBB34C9D3FDULL,
    0x6FCE6F5F7F10A16FULL, 0x4B7A4B62A7EC314BULL, 0x454C451283C60945ULL, 0x39DD39D596AFE439ULL,
    0x3EC63EED84BAF83EULL, 0xDD8EDDA6F42953DDULL, 0xA315A371ED4EB6A3ULL, 0x4F6E4F42BFF0214FULL,
    0xB45EB4C99F2BEAB4ULL, 0xB654B6D99325E2B6ULL, 0x9AC89AA47BE1529AULL, 0x0E360E70242A380EULL,
    0x1F631FF8425D7C1FULL, 0xBF79BF91A51AC6BFULL, 0x154115A87E6B5415ULL, 0xE142E15B7C9DA3E1ULL,
    0x49704972ABE23949ULL, 0xD2BDD2DED6046FD2ULL, 0x93E593EC4DDE7693ULL, 0xC6F9C67EAE683FC6ULL,
    0x92E092E44BD97292ULL, 0x72A772B73143D572ULL, 0x9EDC9E8463FD429EULL, 0x61F8612F5B3A9961ULL,
    0xD1B2D1C6DC0D63D1ULL, 0x63F2633F57349163ULL, 0xFA35FA8326DCCFFAULL, 0xEE71EE235EB09FEEULL,
    0xF403F4F302F6F7F4ULL, 0x197D19C8564F6419ULL, 0xD5A6D5E6C41173D5ULL, 0xAD23AD01C9648EADULL,
    0x582558FACD957D58ULL, 0xA40EA449FF5BAAA4ULL, 0xBB6DBBB1BD06D6BBULL, 0xA11FA161E140BEA1ULL,
    0xDC8BDCAEF22E57DCULL, 0xF21DF2C316E4EFF2ULL, 0x83B5836C2DAE3683ULL, 0x37EB37A5B285DC37ULL,
    0x4257422A91D31542ULL, 0xE453E4736286B7E4ULL, 0x7A8F7AF7017BF57AULL, 0x32FA328DAC9EC832ULL,
    0x9CD69C946FF34A9CULL, 0xCCDBCC2E925E17CCULL, 0xAB3DAB31DD7696ABULL, 0x4A7F4A6AA1EB354AULL,
    0x8F898F0C058A068FULL, 0x6ECB6E577917A56EULL, 0x04140420181C1004ULL, 0x27BB2725D2F59C27ULL,
    0x2E962E6DE4CAB82EULL, 0xE75CE76B688FBBE7ULL, 0xE24DE2437694AFE2ULL, 0x5A2F5AEAC19B755AULL,
    0x96F496C453C56296ULL, 0x164E16B074625816ULL, 0x23AF2305CAE98C23ULL, 0x2B872B45FAD1AC2BULL,
    0xC2EDC25EB6742FC2ULL, 0x65EC650F43268965ULL, 0x66E36617492F8566ULL, 0x0F330F78222D3C0FULL,
    0xBC76BC89AF13CABCULL, 0xA937A921D1789EA9ULL, 0x474647028FC80147ULL, 0x415841329BDA1941ULL,
    0x34E434BDB88CD034ULL, 0x4875487AADE53D48ULL, 0xFC2BFCB332CED7FCULL, 0xB751B7D19522E6B7ULL,
    0x6ADF6A77610BB56AULL, 0x88928834179F1A88ULL, 0xA50BA541F95CAEA5ULL, 0x530253A2F7A45153ULL,
    0x86A4864433B52286ULL, 0xF93AF99B2CD5C3F9ULL, 0x5B2A5BE2C79C715BULL, 0xDB90DB96E03B4BDBULL,
    0x38D838DD90A8E038ULL, 0x7B8A7BFF077CF17BULL, 0xC3E8C356B0732BC3ULL, 0x1E661EF0445A781EULL,
    0x22AA220DCCEE8822ULL, 0x33FF3385AA99CC33ULL, 0x24B4243DD8FC9024ULL, 0x2888285DF0D8A028ULL,
    0x36EE36ADB482D836ULL, 0xC7FCC776A86F3BC7ULL, 0xB240B2F98B39F2B2ULL, 0x3BD73BC59AA1EC3BULL,
    0x8E8C8E04038D028EULL, 0x77B6779F2F58C177ULL, 0xBA68BAB9BB01D2BAULL, 0xF506F5FB04F1F3F5ULL,
    0x144414A0786C5014ULL, 0x9FD99F8C65FA469FULL, 0x0828084030382008ULL, 0x551C5592E3B64955ULL,
    0x9BCD9BAC7DE6569BULL, 0x4C614C5AB5F92D4CULL, 0xFE21FEA33EC0DFFEULL, 0x60FD60275D3D9D60ULL,
    0x5C315CDAD5896D5CULL, 0xDA95DA9EE63C4FDAULL, 0x187818C050486018ULL, 0x4643460A89CF0546ULL,
    0xCDDECD26945913CDULL, 0x7D947DCF136EE97DULL, 0x21A52115C6E78421ULL, 0xB04AB0E98737FAB0ULL,
    0x3FC33FE582BDFC3FULL, 0x1B771BD85A416C1BULL, 0x8997893C11981E89ULL, 0xFF24FFAB38C7DBFFULL,
    0xEB60EB0B40AB8BEBULL, 0x84AE84543FBB2A84ULL, 0x69D0696F6B02B969ULL, 0x3AD23ACD9CA6E83AULL,
```

```
    0x9DD39D9C69F44E9DULL, 0xD7ACD7F6C81F7BD7ULL, 0xD3B8D3D6D0036BD3ULL, 0x70AD70A73D4DDD70ULL,
    0x67E6671F4F288167ULL, 0x405D403A9DDD1D40ULL, 0xB55BB5C1992CEEB5ULL, 0xDE81DEBEFE205FDEULL,
    0x5D345DD2D38E695DULL, 0x30F0309DA090C030ULL, 0x91EF91FC41D07E91ULL, 0xB14FB1E18130FEB1ULL,
    0x788578E70D75FD78ULL, 0x1155118866774411ULL, 0x0105010806070401ULL, 0xE556E57B6481B3E5ULL,
    0x0000000000000000ULL, 0x68D568676D05BD68ULL, 0x98C298B477EF5A98ULL, 0xA01AA069E747BAA0ULL,
    0xC5F6C566A46133C5ULL, 0x020A02100C0E0802ULL, 0xA604A659F355A2A6ULL, 0x74B974872551CD74ULL,
    0x2D992D75EEC3B42DULL, 0x0B270B583A312C0BULL, 0xA210A279EB49B2A2ULL, 0x76B37697295FC576ULL,
    0xB345B3F18D3EF6B3ULL, 0xBE7CBE99A31DC2BEULL, 0xCED1CE3E9E501FCEULL, 0xBD73BD81A914CEBDULL,
    0xAE2CAE19C36D82AEULL, 0xE96AE91B4CA583E9ULL, 0x8A988A241B91128AULL, 0x31F53195A697C431ULL,
    0x1C6C1CE04854701CULL, 0xEC7BEC3352BE97ECULL, 0xF112F1DB1CEDE3F1ULL, 0x99C799BC71E85E99ULL,
    0x94FE94D45FCB6A94ULL, 0xAA38AA39DB7192AAULL, 0xF609F6E30EF8FFF6ULL, 0x26BE262DD4F29826ULL,
    0x2F932F65E2CDBC2FULL, 0xEF74EF2B58B79BEFULL, 0xE86FE8134AA287E8ULL, 0x8C868C140F830A8CULL,
    0x35E135B5BE8BD435ULL, 0x030F03180A090C03ULL, 0xD4A3D4EEC21677D4ULL, 0x7F9E7FDF1F60E17FULL,
    0xFB30FB8B20DBCBFBULL, 0x051105281E1B1405ULL, 0xC1E2C146BC7D23C1ULL, 0x5E3B5ECAD987655EULL,
    0x90EA90F447D77A90ULL, 0x20A0201DC0E08020ULL, 0x3DC93DF58EB3F43DULL, 0x82B082642BA93282ULL,
    0xF70CF7EB08FFFBF7ULL, 0xEA65EA0346AC8FEAULL, 0x0A220A503C36280AULL, 0x0D390D682E23340DULL,
    0x7E9B7ED71967E57EULL, 0xF83FF8932AD2C7F8ULL, 0x500D50BAFDAD5D50ULL, 0x1A721AD05C46681AULL,
    0xC4F3C46EA26637C4ULL, 0x071B073812151C07ULL, 0x57165782EFB84157ULL, 0xB862B8A9B70FDAB8ULL,
    0x3CCC3CFD88B4F03CULL, 0x62F7623751339562ULL, 0xE348E34B7093ABE3ULL, 0xC8CFC80E8A4207C8ULL,
    0xAC26AC09CF638AACULL, 0x520752AAF1A35552ULL, 0x64E9640745218D64ULL, 0x1050108060704010ULL,
    0xD0B7D0CEDA0A67D0ULL, 0xD99AD986EC3543D9ULL, 0x135F13986A794C13ULL, 0x0C3C0C602824300CULL,
    0x125A12906C7E4812ULL, 0x298D2955F6DFA429ULL, 0x510851B2FBAA5951ULL, 0xB967B9A1B108DEB9ULL,
    0xCFD4CF3698571BCFULL, 0xD6A9D6FECE187FD6ULL, 0x73A273BF3744D173ULL, 0x8D838D1C09840E8DULL,
    0x81BF817C21A03E81ULL, 0x5419549AE5B14D54ULL, 0xC0E7C04EBA7A27C0ULL, 0xED7EED3B54B993EDULL,
    0x4E6B4E4AB9F7254EULL, 0x4449441A85C10D44ULL, 0xA701A751F552A6A7ULL, 0x2A822A4DFCD6A82AULL,
    0x85AB855C39BC2E85ULL, 0x25B12535DEFB9425ULL, 0xE659E6636E88BFE6ULL, 0xCAC5CA1E864C0FCAULL,
    0x7C917CC71569ED7CULL, 0x8B9D8B2C1D96168BULL, 0x5613568AE9BF4556ULL, 0x80BA807427A73A80ULL
};

static uint64_t T1[256] = {
    0xD1CE3E9E501FCECEULL, 0x6DBBB1BD06D6BBBBULL, 0x60EB0B40AB8BEBEBULL, 0xE092E44BD9729292ULL,
    0x65EA0346AC8FEAEAULL, 0xC0CB16804B0BCBCBULL, 0x5F13986A794C1313ULL, 0xE2C146BC7D23C1C1ULL,
    0x6AE91B4CA583E9E9ULL, 0xD23ACD9CA6E83A3AULL, 0xA9D6FECE187FD6D6ULL, 0x40B2F98B39F2B2B2ULL,
    0xBDD2DED6046FD2D2ULL, 0xEA90F447D77A9090ULL, 0x4B17B872655C1717ULL, 0x3FF8932AD2C7F8F8ULL,
    0x57422A91D3154242ULL, 0x4115A87E6B541515ULL, 0x13568AE9BF455656ULL, 0x5EB4C99F2BEAB4B4ULL,
    0xEC650F4326896565ULL, 0x6C1CE04854701C1CULL, 0x928834179F1A8888ULL, 0x52432297D4114343ULL,
    0xF6C566A46133C5C5ULL, 0x315CDAD5896D5C5CULL, 0xEE36ADB482D83636ULL, 0x68BAB9BB01D2BABAULL,
    0x06F5FB04F1F3F5F5ULL, 0x165782EFB8415757ULL, 0xE6671F4F28816767ULL, 0x838D1C09840E8D8DULL,
    0xF53195A697C43131ULL, 0x09F6E30EF8FFF6F6ULL, 0xE9640745218D6464ULL, 0x2558FACD957D5858ULL,
    0xDC9E8463FD429E9EULL, 0x03F4F302F6F7F4F4ULL, 0xAA220DCCEE882222ULL, 0x38AA39DB7192AAAAULL,
    0xBC758F2356C97575ULL, 0x330F78222D3C0F0FULL, 0x0A02100C0E080202ULL, 0x4FB1E18130FEB1B1ULL,
    0x84DFB6F8275BDFDFULL, 0xC46D4F731EA96D6DULL, 0xA273BF3744D17373ULL, 0x644D52B3FE294D4DULL,
    0x917CC71569ED7C7CULL, 0xBE262DD4F2982626ULL, 0x962E6DE4CAB82E2EULL, 0x0CF7EB08FFFBF7F7ULL,
    0x2808403038200808ULL, 0x345DD2D38E695D5DULL, 0x49441A85C10D4444ULL, 0xC63EED84BAF83E3EULL,
    0xD99F8C65FA469F9FULL, 0x4414A0786C501414ULL, 0xCFC80E8A4207C8C8ULL, 0x2CAE19C36D82AEAEULL,
    0x19549AE5B14D5454ULL, 0x5010806070401010ULL, 0x9FD88EEA3247D8D8ULL, 0x76BC89AF13CABCBCULL,
    0x721AD05C46681A1AULL, 0xDA6B7F670CB16B6BULL, 0xD0696F6B02B96969ULL, 0x18F3CB10E3EBF3F3ULL,
```

0x73BD81A914CEBDBDULL, 0xFF3385AA99CC3333ULL, 0x3DAB31DD7696ABABULL, 0x35FA8326DCCFFAFAULL,
0xB2D1C6DC0D63D1D1ULL, 0xCD9BAC7DE6569B9BULL, 0xD568676D05BD6868ULL, 0x6B4E4AB9F7254E4EULL,
0x4E16B07462581616ULL, 0xFB95DC59CC6E9595ULL, 0xEF91FC41D07E9191ULL, 0x71EE235EB09FEEEEULL,
0x614C5AB5F92D4C4CULL, 0xF2633F5734916363ULL, 0x8C8E04038D028E8EULL, 0x2A5BE2C79C715B5BULL,
0xDBCC2E925E17CCCCULL, 0xCC3CFD88B4F03C3CULL, 0x7D19C8564F641919ULL, 0x1FA161E140BEA1A1ULL,
0xBF817C21A03E8181ULL, 0x704972ABE2394949ULL, 0x8A7BFF077CF17B7BULL, 0x9AD986EC3543D9D9ULL,
0xCE6F5F7F10A16F6FULL, 0xEB37A5B285DC3737ULL, 0xFD60275D3D9D6060ULL, 0xC5CA1E864C0FCACAULL,
0x5CE76B688FBBE7E7ULL, 0x872B45FAD1AC2B2BULL, 0x75487AADE53D4848ULL, 0x2EFDBB34C9D3FDFDULL,
0xF496C453C5629696ULL, 0x4C451283C6094545ULL, 0x2BFCB332CED7FCFCULL, 0x5841329BDA194141ULL,
0x5A12906C7E481212ULL, 0x390D682E23340D0DULL, 0x8079EF0B72F97979ULL, 0x56E57B6481B3E5E5ULL,
0x97893C11981E8989ULL, 0x868C140F830A8C8CULL, 0x48E34B7093ABE3E3ULL, 0xA0201DC0E0802020ULL,
0xF0309DA090C03030ULL, 0x8BDCAEF22E57DCDCULL, 0x51B7D19522E6B7B7ULL, 0xC16C477519AD6C6CULL,
0x7F4A6AA1EB354A4AULL, 0x5BB5C1992CEEB5B5ULL, 0xC33FE582BDFC3F3FULL, 0xF197CC55C2669797ULL,
0xA3D4EEC21677D4D4ULL, 0xF762375133956262ULL, 0x992D75EEC3B42D2DULL, 0x1E06301412180606ULL,
0x0EA449FF5BAAA4A4ULL, 0x0BA541F95CAEA5A5ULL, 0xB5836C2DAE368383ULL, 0x3E5FC2DF80615F5FULL,
0x822A4DFCD6A82A2AULL, 0x95DA9EE63C4FDADAULL, 0xCAC9068C4503C9C9ULL, 0x0000000000000000ULL,
0x9B7ED71967E57E7EULL, 0x10A279EB49B2A2A2ULL, 0x1C5592E3B6495555ULL, 0x79BF91A51AC6BFBFULL,
0x5511886677441111ULL, 0xA6D5E6C41173D5D5ULL, 0xD69C946FF34A9C9CULL, 0xD4CF3698571BCFCFULL,
0x360E70242A380E0EULL, 0x220A503C36280A0AULL, 0xC93DF58EB3F43D3DULL, 0x0851B2FBAA595151ULL,
0x947DCF136EE97D7DULL, 0xE593EC4DDE769393ULL, 0x771BD85A416C1B1BULL, 0x21FEA33EC0DFFEFEULL,
0xF3C46EA26637C4C4ULL, 0x4647028FC8014747ULL, 0x2D0948363F240909ULL, 0xA4864433B5228686ULL,
0x270B583A312C0B0BULL, 0x898F0C058A068F8FULL, 0xD39D9C69F44E9D9DULL, 0xDF6A77610BB56A6AULL,
0x1B073812151C0707ULL, 0x67B9A1B108DEB9B9ULL, 0x4AB0E98737FAB0B0ULL, 0xC298B477EF5A9898ULL,
0x7818C05048601818ULL, 0xFA328DAC9EC83232ULL, 0xA871AF3B4AD97171ULL, 0x7A4B62A7EC314B4BULL,
0x74EF2B58B79BEFEFULL, 0xD73BC59AA1EC3B3BULL, 0xAD70A73D4DDD7070ULL, 0x1AA069E747BAA0A0ULL,
0x53E4736286B7E4E4ULL, 0x5D403A9DDD1D4040ULL, 0x24FFAB38C7DBFFFFULL, 0xE8C356B0732BC3C3ULL,
0x37A921D1789EA9A9ULL, 0x59E6636E88BFE6E6ULL, 0x8578E70D75FD7878ULL, 0x3AF99B2CD5C3F9F9ULL,
0x9D8B2C1D96168B8BULL, 0x43460A89CF054646ULL, 0xBA807427A73A8080ULL, 0x661EF0445A781E1EULL,
0xD838DD90A8E03838ULL, 0x42E15B7C9DA3E1E1ULL, 0x62B8A9B70FDAB8B8ULL, 0x32A829D77F9AA8A8ULL,
0x47E0537A9AA7E0E0ULL, 0x3C0C602824300C0CULL, 0xAF2305CAE98C2323ULL, 0xB37697295FC57676ULL,
0x691DE84E53741D1DULL, 0xB12535DEFB942525ULL, 0xB4243DD8FC902424ULL, 0x1105281E1B140505ULL,
0x12F1DB1CEDE3F1F1ULL, 0xCB6E577917A56E6EULL, 0xFE94D45FCB6A9494ULL, 0x88285DF0D8A02828ULL,
0xC89AA47BE1529A9AULL, 0xAE84543FBB2A8484ULL, 0x6FE8134AA287E8E8ULL, 0x15A371ED4EB6A3A3ULL,
0x6E4F42BFF0214F4FULL, 0xB6779F2F58C17777ULL, 0xB8D3D6D0036BD3D3ULL, 0xAB855C39BC2E8585ULL,
0x4DE2437694AFE2E2ULL, 0x0752AAF1A3555252ULL, 0x1DF2C316E4EFF2F2ULL, 0xB082642BA9328282ULL,
0x0D50BAFDAD5D5050ULL, 0x8F7AF7017BF57A7AULL, 0x932F65E2CDBC2F2FULL, 0xB974872551CD7474ULL,
0x0253A2F7A4515353ULL, 0x45B3F18D3EF6B3B3ULL, 0xF8612F5B3A996161ULL, 0x29AF11C56A86AFAFULL,
0xDD39D596AFE43939ULL, 0xE135B5BE8BD43535ULL, 0x81DEBEFE205FDEDEULL, 0xDECD26945913CDCDULL,
0x631FF8425D7C1F1FULL, 0xC799BC71E85E9999ULL, 0x26AC09CF638AACACULL, 0x23AD01C9648EADADULL,
0xA772B73143D57272ULL, 0x9C2C7DE8C4B02C2CULL, 0x8EDDA6F42953DDDDULL, 0xB7D0CEDA0A67D0D0ULL,
0xA1874C35B2268787ULL, 0x7CBE99A31DC2BEBEULL, 0x3B5ECAD987655E5EULL, 0x04A659F355A2A6A6ULL,
0x7BEC3352BE97ECECULL, 0x140420181C100404ULL, 0xF9C67EAE683FC6C6ULL, 0x0F03180A090C0303ULL,
0xE434BDB88CD03434ULL, 0x30FB8B20DBCBFBFBULL, 0x90DB96E03B4BDBDBULL, 0x2059F2CB92795959ULL,
0x54B6D99325E2B6B6ULL, 0xEDC25EB6742FC2C2ULL, 0x0501080607040101ULL, 0x17F0D31AEAE7F0F0ULL,
0x2F5AEAC19B755A5AULL, 0x7EED3B54B993EDEDULL, 0x01A751F552A6A7A7ULL, 0xE36617492F856666ULL,
0xA52115C6E7842121ULL, 0x9E7FDF1F60E17F7FULL, 0x988A241B91128A8AULL, 0xBB2725D2F59C2727ULL,
0xFCC776A86F3BC7C7ULL, 0xE7C04EBA7A27C0C0ULL, 0x8D2955F6DFA42929ULL, 0xACD7F6C81F7BD7D7ULL

```
};

static uint64_t T2[256] = {
    0x93EC4DDE769393E5ULL, 0xD986EC3543D9D99AULL, 0x9AA47BE1529A9AC8ULL, 0xB5C1992CEEB5B55BULL,
    0x98B477EF5A9898C2ULL, 0x220DCCEE882222AAULL, 0x451283C60945454CULL, 0xFCB332CED7FCFC2BULL,
    0xBAB9BB01D2BABA68ULL, 0x6A77610BB56A6ADFULL, 0xDFB6F8275BDFDF84ULL, 0x02100C0E0802020AULL,
    0x9F8C65FA469F9FD9ULL, 0xDCAEF22E57DCDC8BULL, 0x51B2FBAA59515108ULL, 0x59F2CB9279595920ULL,
    0x4A6AA1EB354A4A7FULL, 0x17B872655C17174BULL, 0x2B45FAD1AC2B2B87ULL, 0xC25EB6742FC2C2EDULL,
    0x94D45FCB6A9494FEULL, 0xF4F302F6F7F4F403ULL, 0xBBB1BD06D6BBBB6DULL, 0xA371ED4EB6A3A315ULL,
    0x62375133956262F7ULL, 0xE4736286B7E4E453ULL, 0x71AF3B4AD97171A8ULL, 0xD4EEC21677D4D4A3ULL,
    0xCD26945913CDCDDEULL, 0x70A73D4DDD7070ADULL, 0x16B074625816164EULL, 0xE15B7C9DA3E1E142ULL,
    0x4972ABE239494970ULL, 0x3CFD88B4F03C3CCCULL, 0xC04EBA7A27C0C0E7ULL, 0xD88EEA3247D8D89FULL,
    0x5CDAD5896D5C5C31ULL, 0x9BAC7DE6569B9BCDULL, 0xAD01C9648EADAD23ULL, 0x855C39BC2E8585ABULL,
    0x53A2F7A451535302ULL, 0xA161E140BEA1A11FULL, 0x7AF7017BF57A7A8FULL, 0xC80E8A4207C8C8CFULL,
    0x2D75EEC3B42D2D99ULL, 0xE0537A9AA7E0E047ULL, 0xD1C6DC0D63D1D1B2ULL, 0x72B73143D57272A7ULL,
    0xA659F355A2A6A604ULL, 0x2C7DE8C4B02C2C9CULL, 0xC46EA26637C4C4F3ULL, 0xE34B7093ABE3E348ULL,
    0x7697295FC57676B3ULL, 0x78E70D75FD787885ULL, 0xB7D19522E6B7B751ULL, 0xB4C99F2BEAB4B45EULL,
    0x0948363F2409092DULL, 0x3BC59AA1EC3B3BD7ULL, 0x0E70242A380E0E36ULL, 0x41329BDA19414158ULL,
    0x4C5AB5F92D4C4C61ULL, 0xDEBEFE205FDEDE81ULL, 0xB2F98B39F2B2B240ULL, 0x90F447D77A9090EAULL,
    0x2535DEFB942525B1ULL, 0xA541F95CAEA5A50BULL, 0xD7F6C81F7BD7D7ACULL, 0x03180A090C03030FULL,
    0x1188667744111155ULL, 0x0000000000000000ULL, 0xC356B0732BC3C3E8ULL, 0x2E6DE4CAB82E2E96ULL,
    0x92E44BD9729292E0ULL, 0xEF2B58B79BEFEF74ULL, 0x4E4AB9F7254E4E6BULL, 0x12906C7E4812125AULL,
    0x9D9C69F44E9D9DD3ULL, 0x7DCF136EE97D7D94ULL, 0xCB16804B0BCBCBC0ULL, 0x35B5BE8BD43535E1ULL,
    0x1080607040101050ULL, 0xD5E6C41173D5D5A6ULL, 0x4F42BFF0214F4F6EULL, 0x9E8463FD429E9EDCULL,
    0x4D52B3FE294D4D64ULL, 0xA921D1789EA9A937ULL, 0x5592E3B64955551CULL, 0xC67EAE683FC6C6F9ULL,
    0xD0CEDA0A67D0D0B7ULL, 0x7BFF077CF17B7B8AULL, 0x18C0504860181878ULL, 0x97CC55C2669797F1ULL,
    0xD3D6D0036BD3D3B8ULL, 0x36ADB482D83636EEULL, 0xE6636E88BFE6E659ULL, 0x487AADE53D484875ULL,
    0x568AE9BF45565613ULL, 0x817C21A03E8181BFULL, 0x8F0C058A068F8F89ULL, 0x779F2F58C17777B6ULL,
    0xCC2E925E17CCCCDBULL, 0x9C946FF34A9C9CD6ULL, 0xB9A1B108DEB9B967ULL, 0xE2437694AFE2E24DULL,
    0xAC09CF638AACAC26ULL, 0xB8A9B70FDAB8B862ULL, 0x2F65E2CDBC2F2F93ULL, 0x15A87E6B54151541ULL,
    0xA449FF5BAAA4A40EULL, 0x7CC71569ED7C7C91ULL, 0xDA9EE63C4FDADA95ULL, 0x38DD90A8E03838D8ULL,
    0x1EF0445A781E1E66ULL, 0x0B583A312C0B0B27ULL, 0x05281E1B14050511ULL, 0xD6FECE187FD6D6A9ULL,
    0x14A0786C50141444ULL, 0x6E577917A56E6ECBULL, 0x6C477519AD6C6CC1ULL, 0x7ED71967E57E7E9BULL,
    0x6617492F856666E3ULL, 0xFDBB34C9D3FDFD2EULL, 0xB1E18130FEB1B14FULL, 0xE57B6481B3E5E556ULL,
    0x60275D3D9D6060FDULL, 0xAF11C56A86AFAF29ULL, 0x5ECAD987655E5E3BULL, 0x3385AA99CC3333FFULL,
    0x874C35B2268787A1ULL, 0xC9068C4503C9C9CAULL, 0xF0D31AEAE7F0F017ULL, 0x5DD2D38E695D5D34ULL,
    0x6D4F731EA96D6DC4ULL, 0x3FE582BDFC3F3FC3ULL, 0x8834179F1A888892ULL, 0x8D1C09840E8D8D83ULL,
    0xC776A86F3BC7C7FCULL, 0xF7EB08FFFBF7F70CULL, 0x1DE84E53741D1D69ULL, 0xE91B4CA583E9E96AULL,
    0xEC3352BE97ECEC7BULL, 0xED3B54B993EDED7EULL, 0x807427A73A8080BAULL, 0x2955F6DFA429298DULL,
    0x2725D2F59C2727BBULL, 0xCF3698571BCFCFD4ULL, 0x99BC71E85E9999C7ULL, 0xA829D77F9AA8A832ULL,
    0x50BAFDAD5D50500DULL, 0x0F78222D3C0F0F33ULL, 0x37A5B285DC3737EBULL, 0x243DD8FC902424B4ULL,
    0x285DF0D8A0282888ULL, 0x309DA090C03030F0ULL, 0x95DC59CC6E9595FBULL, 0xD2DED6046FD2D2BDULL,
    0x3EED84BAF83E3EC6ULL, 0x5BE2C79C715B5B2AULL, 0x403A9DDD1D40405DULL, 0x836C2DAE368383B5ULL,
    0xB3F18D3EF6B3B345ULL, 0x696F6B02B96969D0ULL, 0x5782EFB841575716ULL, 0x1FF8425D7C1F1F63ULL,
    0x073812151C07071BULL, 0x1CE04854701C1C6CULL, 0x8A241B91128A8A98ULL, 0xBC89AF13CABCBC76ULL,
    0x201DC0E0802020A0ULL, 0xEB0B40AB8BEBEB60ULL, 0xCE3E9E501FCECED1ULL, 0x8E04038D028E8E8CULL,
    0xAB31DD7696ABAB3DULL, 0xEE235EB09FEEEE71ULL, 0x3195A697C43131F5ULL, 0xA279EB49B2A2A210ULL,
```

```
    0x73BF3744D17373A2ULL, 0xF99B2CD5C3F9F93AULL, 0xCA1E864C0FCACAC5ULL, 0x3ACD9CA6E83A3AD2ULL,
    0x1AD05C46681A1A72ULL, 0xFB8B20DBCBFBFB30ULL, 0x0D682E23340D0D39ULL, 0xC146BC7D23C1C1E2ULL,
    0xFEA33EC0DFFEFE21ULL, 0xFA8326DCCFFAFA35ULL, 0xF2C316E4EFF2F21DULL, 0x6F5F7F10A16F6FCEULL,
    0xBD81A914CEBDBD73ULL, 0x96C453C5629696F4ULL, 0xDDA6F42953DDDD8EULL, 0x432297D411434352ULL,
    0x52AAF1A355525207ULL, 0xB6D99325E2B6B654ULL, 0x0840303820080828ULL, 0xF3CB10E3EBF3F318ULL,
    0xAE19C36D82AEAE2CULL, 0xBE99A31DC2BEBE7CULL, 0x19C8564F6419197DULL, 0x893C11981E898997ULL,
    0x328DAC9EC83232FAULL, 0x262DD4F2982626BEULL, 0xB0E98737FAB0B04AULL, 0xEA0346AC8FEAEA65ULL,
    0x4B62A7EC314B4B7AULL, 0x640745218D6464E9ULL, 0x84543FBB2A8484AEULL, 0x82642BA9328282B0ULL,
    0x6B7F670CB16B6BDAULL, 0xF5FB04F1F3F5F506ULL, 0x79EF0B72F9797980ULL, 0xBF91A51AC6BFBF79ULL,
    0x0108060704010105ULL, 0x5FC2DF80615F5F3EULL, 0x758F2356C97575BCULL, 0x633F5734916363F2ULL,
    0x1BD85A416C1B1B77ULL, 0x2305CAE98C2323AFULL, 0x3DF58EB3F43D3DC9ULL, 0x68676D05BD6868D5ULL,
    0x2A4DFCD6A82A2A82ULL, 0x650F4326896565ECULL, 0xE8134AA287E8E86FULL, 0x91FC41D07E9191EFULL,
    0xF6E30EF8FFF6F609ULL, 0xFFAB38C7DBFFFF24ULL, 0x13986A794C13135FULL, 0x58FACD957D585825ULL,
    0xF1DB1CEDE3F1F112ULL, 0x47028FC801474746ULL, 0x0A503C36280A0A22ULL, 0x7FDF1F60E17F7F9EULL,
    0xC566A46133C5C5F6ULL, 0xA751F552A6A7A701ULL, 0xE76B688FBBE7E75CULL, 0x612F5B3A996161F8ULL,
    0x5AEAC19B755A5A2FULL, 0x063014121806061EULL, 0x460A89CF05464643ULL, 0x441A85C10D444449ULL,
    0x422A91D315424257ULL, 0x0420181C10040414ULL, 0xA069E747BAA0A01AULL, 0xDB96E03B4BDBDB90ULL,
    0x39D596AFE43939DDULL, 0x864433B5228686A4ULL, 0x549AE5B14D545419ULL, 0xAA39DB7192AAAA38ULL,
    0x8C140F830A8C8C86ULL, 0x34BDB88CD03434E4ULL, 0x2115C6E7842121A5ULL, 0x8B2C1D96168B8B9DULL,
    0xF8932AD2C7F8F83FULL, 0x0C602824300C0C3CULL, 0x74872551CD7474B9ULL, 0x671F4F28816767E6ULL
};

static uint64_t T3[256] = {
    0x676D05BD6868D568ULL, 0x1C09840E8D8D838DULL, 0x1E864C0FCACAC5CAULL, 0x52B3FE294D4D644DULL,
    0xBF3744D17373A273ULL, 0x62A7EC314B4B7A4BULL, 0x4AB9F7254E4E6B4EULL, 0x4DFCD6A82A2A822AULL,
    0xEEC21677D4D4A3D4ULL, 0xAAF1A35552520752ULL, 0x2DD4F2982626BE26ULL, 0xF18D3EF6B3B345B3ULL,
    0x9AE5B14D54541954ULL, 0xF0445A781E1E661EULL, 0xC8564F6419197D19ULL, 0xF8425D7C1F1F631FULL,
    0x0DCCEE882222AA22ULL, 0x180A090C03030F03ULL, 0x0A89CF0546464346ULL, 0xF58EB3F43D3DC93DULL,
    0x75EEC3B42D2D992DULL, 0x6AA1EB354A4A7F4AULL, 0xA2F7A45153530253ULL, 0x6C2DAE368383B583ULL,
    0x986A794C13135F13ULL, 0x241B91128A8A988AULL, 0xD19522E6B7B751B7ULL, 0xE6C41173D5D5A6D5ULL,
    0x35DEFB942525B125ULL, 0xEF0B72F979798079ULL, 0xFB04F1F3F5F506F5ULL, 0x81A914CEBDBD73BDULL,
    0xFACD957D58582558ULL, 0x65E2CDBC2F2F932FULL, 0x682E23340D0D390DULL, 0x100C0E0802020A02ULL,
    0x3B54B993EDED7EEDULL, 0xB2FBAA5951510851ULL, 0x8463FD429E9EDC9EULL, 0x8866774411115511ULL,
    0xC316E4EFF2F21DF2ULL, 0xED84BAF83E3EC63EULL, 0x92E3B64955551C55ULL, 0xCAD987655E5E3B5EULL,
    0xC6DC0D63D1D1B2D1ULL, 0xB074625816164E16ULL, 0xFD88B4F03C3CCC3CULL, 0x17492F856666E366ULL,
    0xA73D4DDD7070AD70ULL, 0xD2D38E695D5D345DULL, 0xCB10E3EBF3F318F3ULL, 0x1283C60945454C45ULL,
    0x3A9DDD1D40405D40ULL, 0x2E925E17CCCCDBCCULL, 0x134AA287E8E86FE8ULL, 0xD45FCB6A9494FE94ULL,
    0x8AE9BF4556561356ULL, 0x4030382008082808ULL, 0x3E9E501FCECED1CEULL, 0xD05C46681A1A721AULL,
    0xCD9CA6E83A3AD23AULL, 0xDED6046FD2D2BDD2ULL, 0x5B7C9DA3E1E142E1ULL, 0xB6F8275BDFDF84DFULL,
    0xC1992CEEB5B55BB5ULL, 0xDD90A8E03838D838ULL, 0x577917A56E6ECB6EULL, 0x70242A380E0E360EULL,
    0x7B6481B3E5E556E5ULL, 0xF302F6F7F4F403F4ULL, 0x9B2CD5C3F9F93AF9ULL, 0x4433B5228686A486ULL,
    0x1B4CA583E9E96AE9ULL, 0x42BFF0214F4F6E4FULL, 0xFECE187FD6D6A9D6ULL, 0x5C39BC2E8585AB85ULL,
    0x05CAE98C2323AF23ULL, 0x3698571BCFCFD4CFULL, 0x8DAC9EC83232FA32ULL, 0xBC71E85E9999C799ULL,
    0x95A697C43131F531ULL, 0xA0786C5014144414ULL, 0x19C36D82AEAE2CAEULL, 0x235EB09FEEEE71EEULL,
    0x0E8A4207C8C8CFC8ULL, 0x7AADE53D48487548ULL, 0xD6D0036BD3D3B8D3ULL, 0x9DA090C03030F030ULL,
    0x61E140BEA1A11FA1ULL, 0xE44BD9729292E092ULL, 0x329BDA1941415841ULL, 0xE18130FEB1B14FB1ULL,
    0xC050486018187818ULL, 0x6EA26637C4C4F3C4ULL, 0x7DE8C4B02C2C9C2CULL, 0xAF3B4AD97171A871ULL,
```

```
    0xB73143D57272A772ULL, 0x1A85C10D44444944ULL, 0xA87E6B5415415411 5ULL, 0xBB34C9D3FDFD2EFDULL,
    0xA5B285DC3737EB37ULL, 0x99A31DC2BEBE7CBEULL, 0xC2DF80615F5F3E5FULL, 0x39DB7192AAAA38AAULL,
    0xAC7DE6569B9BCD9BULL, 0x34179F1A88889288ULL, 0x8EEA3247D8D89FD8ULL, 0x31DD7696ABAB3DABULL,
    0x3C11981E89899789ULL, 0x946FF34A9C9CD69CULL, 0x8326DCCFFAFA35FAULL, 0x275D3D9D6060FD60ULL,
    0x0346AC8FEAEA65EAULL, 0x89AF13CABCBC76BCULL, 0x375133956262F762ULL, 0x602824300C0C3C0CULL,
    0x3DD8FC902424B424ULL, 0x59F355A2A6A604A6ULL, 0x29D77F9AA8A832A8ULL, 0x3352BE97ECEC7BECULL,
    0x1F4F28816767E667ULL, 0x1DC0E0802020A020ULL, 0x96E03B4BDBDB90DBULL, 0xC71569ED7C7C917CULL,
    0x5DF0D8A028288828ULL, 0xA6F42953DDDD8EDDULL, 0x09CF638AACAC26ACULL, 0xE2C79C715B5B2A5BULL,
    0xBDB88CD03434E434ULL, 0xD71967E57E7E9B7EULL, 0x8060704010105010ULL, 0xDB1CEDE3F1F112F1ULL,
    0xFF077CF17B7B8A7BULL, 0x0C058A068F8F898FULL, 0x3F5734916363F263ULL, 0x69E747BAA0A01AA0ULL,
    0x281E1B1405051105ULL, 0xA47BE1529A9AC89AULL, 0x2297D41143435243ULL, 0x9F2F58C17777B677ULL,
    0x15C6E7842121A521ULL, 0x91A51AC6BFBF79BFULL, 0x25D2F59C2727BB27ULL, 0x48363F2409092D09ULL,
    0x56B0732BC3C3E8C3ULL, 0x8C65FA469F9FD99FULL, 0xD99325E2B6B654B6ULL, 0xF6C81F7BD7D7ACD7ULL,
    0x55F6DFA429298D29ULL, 0x5EB6742FC2C2EDC2ULL, 0x0B40AB8BEBEB60EBULL, 0x4EBA7A27C0C0E7C0ULL,
    0x49FF5BAAA4A40EA4ULL, 0x2C1D96168B8B9D8BULL, 0x140F830A8C8C868CULL, 0xE84E53741D1D691DULL,
    0x8B20DBCBFBFB30FBULL, 0xAB38C7DBFFFF24FFULL, 0x46BC7D23C1C1E2C1ULL, 0xF98B39F2B2B240B2ULL,
    0xCC55C2669797F197ULL, 0x6DE4CAB82E2E962EULL, 0x932AD2C7F8F83FF8ULL, 0x0F4326896565EC65ULL,
    0xE30EF8FFF6F609F6ULL, 0x8F2356C97575BC75ULL, 0x3812151C07071B07ULL, 0x20181C1004041404ULL,
    0x72ABE23949497049ULL, 0x85AA99CC3333FF33ULL, 0x736286B7E4E453E4ULL, 0x86EC3543D9D99AD9ULL,
    0xA1B108DEB9B967B9ULL, 0xCEDA0A67D0D0B7D0ULL, 0x2A91D31542425742ULL, 0x76A86F3BC7C7FCC7ULL,
    0x477519AD6C6CC16CULL, 0xF447D77A9090EA90ULL, 0x0000000000000000ULL, 0x04038D028E8E8C8EULL,
    0x5F7F10A16F6FCE6FULL, 0xBAFDAD5D50500D50ULL, 0x0806070401010501ULL, 0x66A46133C5C5F6C5ULL,
    0x9EE63C4FDADA95DAULL, 0x028FC80147474647ULL, 0xE582BDFC3F3FC33FULL, 0x26945913CDCDDECDULL,
    0x6F6B02B96969D069ULL, 0x79EB49B2A2A210A2ULL, 0x437694AFE2E24DE2ULL, 0xF7017BF57A7A8F7AULL,
    0x51F552A6A7A701A7ULL, 0x7EAE683FC6C6F9C6ULL, 0xEC4DDE769393E593ULL, 0x78222D3C0F0F330FULL,
    0x503C36280A0A220AULL, 0x3014121806061E06ULL, 0x636E88BFE6E659E6ULL, 0x45FAD1AC2B2B872BULL,
    0xC453C5629696F496ULL, 0x71ED4EB6A3A315A3ULL, 0xE04854701C1C6C1CULL, 0x11C56A86AFAF29AFULL,
    0x77610BB56A6ADF6AULL, 0x906C7E4812125A12ULL, 0x543FBB2A8484AE84ULL, 0xD596AFE43939DD39ULL,
    0x6B688FBBE7E75CE7ULL, 0xE98737FAB0B04AB0ULL, 0x642BA9328282B082ULL, 0xEB08FFFBF7F70CF7ULL,
    0xA33EC0DFFEFE21FEULL, 0x9C69F44E9D9DD39DULL, 0x4C35B2268787A187ULL, 0xDAD5896D5C5C315CULL,
    0x7C21A03E8181BF81ULL, 0xB5BE8BD43535E135ULL, 0xBEFE205FDEDE81DEULL, 0xC99F2BEAB4B45EB4ULL,
    0x41F95CAEA5A50BA5ULL, 0xB332CED7FCFC2BFCULL, 0x7427A73A8080BA80ULL, 0x2B58B79BEFEF74EFULL,
    0x16804B0BCBCBC0CBULL, 0xB1BD06D6BBBB6DBBULL, 0x7F670CB16B6BDA6BULL, 0x97295FC57676B376ULL,
    0xB9BB01D2BABA68BAULL, 0xEAC19B755A5A2F5AULL, 0xCF136EE97D7D947DULL, 0xE70D75FD78788578ULL,
    0x583A312C0B0B270BULL, 0xDC59CC6E9595FB95ULL, 0x4B7093ABE3E348E3ULL, 0x01C9648EADAD23ADULL,
    0x872551CD7474B974ULL, 0xB477EF5A9898C298ULL, 0xC59AA1EC3B3BD73BULL, 0xADB482D83636EE36ULL,
    0x0745218D6464E964ULL, 0x4F731EA96D6DC46DULL, 0xAEF22E57DCDC8BDCULL, 0xD31AEAE7F0F017F0ULL,
    0xF2CB927959592059ULL, 0x21D1789EA9A937A9ULL, 0x5AB5F92D4C4C614CULL, 0xB872655C17174B17ULL,
    0xDF1F60E17F7F9E7FULL, 0xFC41D07E9191EF91ULL, 0xA9B70FDAB8B862B8ULL, 0x068C4503C9C9CAC9ULL,
    0x82EFB84157571657ULL, 0xD85A416C1B1B771BULL, 0x537A9AA7E0E047E0ULL, 0x2F5B3A996161F861ULL
};

static uint64_t T4[256] = {
    0xD77F9AA8A832A829ULL, 0x97D4114343524322ULL, 0xDF80615F5F3E5FC2ULL, 0x14121806061E0630ULL,
    0x670CB16B6BDA6B7FULL, 0x2356C97575BC758FULL, 0x7519AD6C6CC16C47ULL, 0xCB927959592059F2ULL,
    0x3B4AD97171A871AFULL, 0xF8275BDFDF84DFB6ULL, 0x35B2268787A1874CULL, 0x59CC6E9595FB95DCULL,
    0x72655C17174B17B8ULL, 0x1AEAE7F0F017F0D3ULL, 0xEA3247D8D89FD88EULL, 0x363F2409092D0948ULL,
```

0x731EA96D6DC46D4FULL, 0x10E3EBF3F318F3CBULL, 0x4E53741D1D691DE8ULL, 0x804B0BCBCBC0CB16ULL,
0x8C4503C9C9CAC906ULL, 0xB3FE294D4D644D52ULL, 0xE8C4B02C2C9C2C7DULL, 0xC56A86AFAF29AF11ULL,
0x0B72F979798079EFULL, 0x7A9AA7E0E047E053ULL, 0x55C2669797F197CCULL, 0x34C9D3FDFD2EFDBBULL,
0x7F10A16F6FCE6F5FULL, 0xA7EC314B4B7A4B62ULL, 0x83C60945454C4512ULL, 0x96AFE43939DD39D5ULL,
0x84BAF83E3EC63EEDULL, 0xF42953DDDD8EDDA6ULL, 0xED4EB6A3A315A371ULL, 0xBFF0214F4F6E4F42ULL,
0x9F2BEAB4B45EB4C9ULL, 0x9325E2B6B654B6D9ULL, 0x7BE1529A9AC89AA4ULL, 0x242A380E0E360E70ULL,
0x425D7C1F1F631FF8ULL, 0xA51AC6BFBF79BF91ULL, 0x7E6B5415154115A8ULL, 0x7C9DA3E1E142E15BULL,
0xABE2394949704972ULL, 0xD6046FD2D2BDD2DEULL, 0x4DDE769393E593ECULL, 0xAE683FC6C6F9C67EULL,
0x4BD9729292E092E4ULL, 0x3143D57272A772B7ULL, 0x63FD429E9EDC9E84ULL, 0x5B3A996161F8612FULL,
0xDC0D63D1D1B2D1C6ULL, 0x5734916363F2633FULL, 0x26DCCFFAFA35FA83ULL, 0x5EB09FEEEE71EE23ULL,
0x02F6F7F4F403F4F3ULL, 0x564F6419197D19C8ULL, 0xC41173D5D5A6D5E6ULL, 0xC9648EADAD23AD01ULL,
0xCD957D58582558FAULL, 0xFF5BAAA4A40EA449ULL, 0xBD06D6BBBB6DBBB1ULL, 0xE140BEA1A11FA161ULL,
0xF22E57DCDC8BDCAEULL, 0x16E4EFF2F21DF2C3ULL, 0x2DAE368383B5836CULL, 0xB285DC3737EB37A5ULL,
0x91D315424257422AULL, 0x6286B7E4E453E473ULL, 0x017BF57A7A8F7AF7ULL, 0xAC9EC83232FA328DULL,
0x6FF34A9C9CD69C94ULL, 0x925E17CCCCDBCC2EULL, 0xDD7696ABAB3DAB31ULL, 0xA1EB354A4A7F4A6AULL,
0x058A068F8F898F0CULL, 0x7917A56E6ECB6E57ULL, 0x181C100404140420ULL, 0xD2F59C2727BB2725ULL,
0xE4CAB82E2E962E6DULL, 0x688FBBE7E75CE76BULL, 0x7694AFE2E24DE243ULL, 0xC19B755A5A2F5AEAULL,
0x53C5629696F496C4ULL, 0x74625816164E16B0ULL, 0xCAE98C2323AF2305ULL, 0xFAD1AC2B2B872B45ULL,
0xB6742FC2C2EDC25EULL, 0x4326896565EC650FULL, 0x492F856666E36617ULL, 0x222D3C0F0F330F78ULL,
0xAF13CABCBC76BC89ULL, 0xD1789EA9A937A921ULL, 0x8FC8014747464702ULL, 0x9BDA194141584132ULL,
0xB88CD03434E434BDULL, 0xADE53D484875487AULL, 0x32CED7FCFC2BFCB3ULL, 0x9522E6B7B751B7D1ULL,
0x610BB56A6ADF6A77ULL, 0x179F1A8888928834ULL, 0xF95CAEA5A50BA541ULL, 0xF7A45153530253A2ULL,
0x33B5228686A48644ULL, 0x2CD5C3F9F93AF99BULL, 0xC79C715B5B2A5BE2ULL, 0xE03B4BDBDB90DB96ULL,
0x90A8E03838D838DDULL, 0x077CF17B7B8A7BFFULL, 0xB0732BC3C3E8C356ULL, 0x445A781E1E661EF0ULL,
0xCCEE882222AA220DULL, 0xAA99CC3333FF3385ULL, 0xD8FC902424B4243DULL, 0xF0D8A0282888285DULL,
0xB482D83636EE36ADULL, 0xA86F3BC7C7FCC776ULL, 0x8B39F2B2B240B2F9ULL, 0x9AA1EC3B3BD73BC5ULL,
0x038D028E8E8C8E04ULL, 0x2F58C17777B6779FULL, 0xBB01D2BABA68BAB9ULL, 0x04F1F3F5F506F5FBULL,
0x786C5014144414A0ULL, 0x65FA469F9FD99F8CULL, 0x3038200808280840ULL, 0xE3B64955551C5592ULL,
0x7DE6569B9BCD9BACULL, 0xB5F92D4C4C614C5AULL, 0x3EC0DFFEFE21FEA3ULL, 0x5D3D9D6060FD6027ULL,
0xD5896D5C5C315CDAULL, 0xE63C4FDADA95DA9EULL, 0x50486018187818C0ULL, 0x89CF05464643460AULL,
0x945913CDCDDECD26ULL, 0x136EE97D7D947DCFULL, 0xC6E7842121A52115ULL, 0x8737FAB0B04AB0E9ULL,
0x82BDFC3F3FC33FE5ULL, 0x5A416C1B1B771BD8ULL, 0x11981E898997893CULL, 0x38C7DBFFFF24FFABULL,
0x40AB8BEBEB60EB0BULL, 0x3FBB2A8484AE8454ULL, 0x6B02B96969D0696FULL, 0x9CA6E83A3AD23ACDULL,
0x69F44E9D9DD39D9CULL, 0xC81F7BD7D7ACD7F6ULL, 0xD0036BD3D3B8D3D6ULL, 0x3D4DDD7070AD70A7ULL,
0x4F28816767E6671FULL, 0x9DDD1D40405D403AULL, 0x992CEEB5B55BB5C1ULL, 0xFE205FDEDE81DEBEULL,
0xD38E695D5D345DD2ULL, 0xA090C03030F0309DULL, 0x41D07E9191EF91FCULL, 0x8130FEB1B14FB1E1ULL,
0x0D75FD78788578E7ULL, 0x6677441111551188ULL, 0x0607040101050108ULL, 0x6481B3E5E556E57BULL,
0x0000000000000000ULL, 0x6D05BD6868D56867ULL, 0x77EF5A9898C298B4ULL, 0xE747BAA0A01AA069ULL,
0xA46133C5C5F6C566ULL, 0x0C0E0802020A0210ULL, 0xF355A2A6A604A659ULL, 0x2551CD7474B97487ULL,
0xEEC3B42D2D992D75ULL, 0x3A312C0B0B270B58ULL, 0xEB49B2A2A210A279ULL, 0x295FC57676B37697ULL,
0x8D3EF6B3B345B3F1ULL, 0xA31DC2BEBE7CBE99ULL, 0x9E501FCECED1CE3EULL, 0xA914CEBDBD73BD81ULL,
0xC36D82AEAE2CAE19ULL, 0x4CA583E9E96AE91BULL, 0x1B91128A8A988A24ULL, 0xA697C43131F53195ULL,
0x4854701C1C6C1CE0ULL, 0x52BE97ECEC7BEC33ULL, 0x1CEDE3F1F112F1DBULL, 0x71E85E9999C799BCULL,
0x5FCB6A9494FE94D4ULL, 0xDB7192AAAA38AA39ULL, 0x0EF8FFF6F609F6E3ULL, 0xD4F2982626BE262DULL,
0xE2CDBC2F2F932F65ULL, 0x58B79BEFEF74EF2BULL, 0x4AA287E8E86FE813ULL, 0x0F830A8C8C868C14ULL,
0xBE8BD43535E135B5ULL, 0x0A090C03030F0318ULL, 0xC21677D4D4A3D4EEULL, 0x1F60E17F7F9E7FDFULL,
0x20DBCBFBFB30FB8BULL, 0x1E1B140505110528ULL, 0xBC7D23C1C1E2C146ULL, 0xD987655E5E3B5ECAULL,

```
    0x47D77A9090EA90F4ULL, 0xC0E0802020A0201DULL, 0x8EB3F43D3DC93DF5ULL, 0x2BA9328282B08264ULL,
    0x08FFFBF7F70CF7EBULL, 0x46AC8FEAEA65EA03ULL, 0x3C36280A0A220A50ULL, 0x2E23340D0D390D68ULL,
    0x1967E57E7E9B7ED7ULL, 0x2AD2C7F8F83FF893ULL, 0xFDAD5D50500D50BAULL, 0x5C46681A1A721AD0ULL,
    0xA26637C4C4F3C46EULL, 0x12151C07071B0738ULL, 0xEFB8415757165782ULL, 0xB70FDAB8B862B8A9ULL,
    0x88B4F03C3CCC3CFDULL, 0x5133956262F76237ULL, 0x7093ABE3E348E34BULL, 0x8A4207C8C8CFC80EULL,
    0xCF638AACAC26AC09ULL, 0xF1A35552520752AAULL, 0x45218D6464E96407ULL, 0x6070401010501080ULL,
    0xDA0A67D0D0B7D0CEULL, 0xEC3543D9D99AD986ULL, 0x6A794C13135F1398ULL, 0x2824300C0C3C0C60ULL,
    0x6C7E4812125A1290ULL, 0xF6DFA429298D2955ULL, 0xFBAA5951510851B2ULL, 0xB108DEB9B967B9A1ULL,
    0x98571BCFCFD4CF36ULL, 0xCE187FD6D6A9D6FEULL, 0x3744D17373A273BFULL, 0x09840E8D8D838D1CULL,
    0x21A03E8181BF817CULL, 0xE5B14D545419549AULL, 0xBA7A27C0C0E7C04EULL, 0x54B993EDED7EED3BULL,
    0xB9F7254E4E6B4E4AULL, 0x85C10D444449441AULL, 0xF552A6A7A701A751ULL, 0xFCD6A82A2A822A4DULL,
    0x39BC2E8585AB855CULL, 0xDEFB942525B12535ULL, 0x6E88BFE6E659E663ULL, 0x864C0FCACAC5CA1EULL,
    0x1569ED7C7C917CC7ULL, 0x1D96168B8B9D8B2CULL, 0xE9BF45565613568AULL, 0x27A73A8080BA8074ULL
};

static uint64_t T5[256] = {
    0x501FCECED1CE3E9EULL, 0x06D6BBBB6DBBB1BDULL, 0xAB8BEBEB60EB0B40ULL, 0xD9729292E092E44BULL,
    0xAC8FEAEA65EA0346ULL, 0x4B0BCBCBC0CB1680ULL, 0x794C13135F13986AULL, 0x7D23C1C1E2C146BCULL,
    0xA583E9E96AE91B4CULL, 0xA6E83A3AD23ACD9CULL, 0x187FD6D6A9D6FECEULL, 0x39F2B2B240B2F98BULL,
    0x046FD2D2BDD2DED6ULL, 0xD77A9090EA90F447ULL, 0x655C17174B17B872ULL, 0xD2C7F8F83FF8932AULL,
    0xD315424257422A91ULL, 0x6B5415154115A87EULL, 0xBF45565613568AE9ULL, 0x2BEAB4B45EB4C99FULL,
    0x26896565EC650F43ULL, 0x54701C1C6C1CE048ULL, 0x9F1A888892883417ULL, 0xD411434352432297ULL,
    0x6133C5C5F6C566A4ULL, 0x896D5C5C315CDAD5ULL, 0x82D83636EE36ADB4ULL, 0x01D2BABA68BAB9BBULL,
    0xF1F3F5F506F5FB04ULL, 0xB8415757165782EFULL, 0x28816767E6671F4FULL, 0x840E8D8D838D1C09ULL,
    0x97C43131F53195A6ULL, 0xF8FFF6F609F6E30EULL, 0x218D6464E9640745ULL, 0x957D58582558FACDULL,
    0xFD429E9EDC9E8463ULL, 0xF6F7F4F403F4F302ULL, 0xEE882222AA220DCCULL, 0x7192AAAA38AA39DBULL,
    0x56C97575BC758F23ULL, 0x2D3C0F0F330F7822ULL, 0x0E0802020A02100CULL, 0x30FEB1B14FB1E181ULL,
    0x275BDFDF84DFB6F8ULL, 0x1EA96D6DC46D4F73ULL, 0x44D17373A273BF37ULL, 0xFE294D4D644D52B3ULL,
    0x69ED7C7C917CC715ULL, 0xF2982626BE262DD4ULL, 0xCAB82E2E962E6DE4ULL, 0xFFFBF7F70CF7EB08ULL,
    0x3820080828084030ULL, 0x8E695D5D345DD2D3ULL, 0xC10D444449441A85ULL, 0xBAF83E3EC63EED84ULL,
    0xFA469F9FD99F8C65ULL, 0x6C5014144414A078ULL, 0x4207C8C8CFC80E8AULL, 0x6D82AEAE2CAE19C3ULL,
    0xB14D545419549AE5ULL, 0x7040101050108060ULL, 0x3247D8D89FD88EEAULL, 0x13CABCBC76BC89AFULL,
    0x46681A1A721AD05CULL, 0x0CB16B6BDA6B7F67ULL, 0x02B96969D0696F6BULL, 0xE3EBF3F318F3CB10ULL,
    0x14CEBDBD73BD81A9ULL, 0x99CC3333FF3385AAULL, 0x7696ABAB3DAB31DDULL, 0xDCCFFAFA35FA8326ULL,
    0x0D63D1D1B2D1C6DCULL, 0xE6569B9BCD9BAC7DULL, 0x05BD6868D568676DULL, 0xF7254E4E6B4E4AB9ULL,
    0x625816164E16B074ULL, 0xCC6E9595FB95DC59ULL, 0xD07E9191EF91FC41ULL, 0xB09FEEEE71EE235EULL,
    0xF92D4C4C614C5AB5ULL, 0x34916363F2633F57ULL, 0x8D028E8E8C8E0403ULL, 0x9C715B5B2A5BE2C7ULL,
    0x5E17CCCCDBCC2E92ULL, 0xB4F03C3CCC3CFD88ULL, 0x4F6419197D19C856ULL, 0x40BEA1A11FA161E1ULL,
    0xA03E8181BF817C21ULL, 0xE2394949704972ABULL, 0x7CF17B7B8A7BFF07ULL, 0x3543D9D99AD986ECULL,
    0x10A16F6FCE6F5F7FULL, 0x85DC3737EB37A5B2ULL, 0x3D9D6060FD60275DULL, 0x4C0FCACAC5CA1E86ULL,
    0x8FBBE7E75CE76B68ULL, 0xD1AC2B2B872B45FAULL, 0xE53D484875487AADULL, 0xC9D3FDFD2EFDBB34ULL,
    0xC5629696F496C453ULL, 0xC60945454C451283ULL, 0xCED7FCFC2BFCB332ULL, 0xDA1941415841329BULL,
    0x7E4812125A12906CULL, 0x23340D0D390D682EULL, 0x72F979798079EF0BULL, 0x81B3E5E556E57B64ULL,
    0x981E898997893C11ULL, 0x830A8C8C868C140FULL, 0x93ABE3E348E34B70ULL, 0xE0802020A0201DC0ULL,
    0x90C03030F0309DA0ULL, 0x2E57DCDC8BDCAEF2ULL, 0x22E6B7B751B7D195ULL, 0x19AD6C6CC16C4775ULL,
    0xEB354A4A7F4A6AA1ULL, 0x2CEEB5B55BB5C199ULL, 0xBDFC3F3FC33FE582ULL, 0xC2669797F197CC55ULL,
    0x1677D4D4A3D4EEC2ULL, 0x33956262F7623751ULL, 0xC3B42D2D992D75EEULL, 0x121806061E063014ULL,
```

```
    0x5BAAA4A40EA449FFULL, 0x5CAEA5A50BA541F9ULL, 0xAE368383B5836C2DULL, 0x80615F5F3E5FC2DFULL,
    0xD6A82A2A822A4DFCULL, 0x3C4FDADA95DA9EE6ULL, 0x4503C9C9CAC9068CULL, 0x0000000000000000ULL,
    0x67E57E7E9B7ED719ULL, 0x49B2A2A210A279EBULL, 0xB64955551C5592E3ULL, 0x1AC6BFBF79BF91A5ULL,
    0x7744111155118866ULL, 0x1173D5D5A6D5E6C4ULL, 0xF34A9C9CD69C946FULL, 0x571BCFCFD4CF3698ULL,
    0x2A380E0E360E7024ULL, 0x36280A0A220A503CULL, 0xB3F43D3DC93DF58EULL, 0xAA5951510851B2FBULL,
    0x6EE97D7D947DCF13ULL, 0xDE769393E593EC4DULL, 0x416C1B1B771BD85AULL, 0xC0DFFEFE21FEA33EULL,
    0x6637C4C4F3C46EA2ULL, 0xC80147474647028FULL, 0x3F2409092D094836ULL, 0xB5228686A4864433ULL,
    0x312C0B0B270B583AULL, 0x8A068F8F898F0C05ULL, 0xF44E9D9DD39D9C69ULL, 0x0BB56A6ADF6A7761ULL,
    0x151C07071B073812ULL, 0x08DEB9B967B9A1B1ULL, 0x37FAB0B04AB0E987ULL, 0xEF5A9898C298B477ULL,
    0x486018187818C050ULL, 0x9EC83232FA328DACULL, 0x4AD97171A871AF3BULL, 0xEC314B4B7A4B62A7ULL,
    0xB79BEFEF74EF2B58ULL, 0xA1EC3B3BD73BC59AULL, 0x4DDD7070AD70A73DULL, 0x47BAA0A01AA069E7ULL,
    0x86B7E4E453E47362ULL, 0xDD1D40405D403A9DULL, 0xC7DBFFFF24FFAB38ULL, 0x732BC3C3E8C356B0ULL,
    0x789EA9A937A921D1ULL, 0x88BFE6E659E6636EULL, 0x75FD78788578E70DULL, 0xD5C3F9F93AF99B2CULL,
    0x96168B8B9D8B2C1DULL, 0xCF05464643460A89ULL, 0xA73A8080BA807427ULL, 0x5A781E1E661EF044ULL,
    0xA8E03838D838DD90ULL, 0x9DA3E1E142E15B7CULL, 0x0FDAB8B862B8A9B7ULL, 0x7F9AA8A832A829D7ULL,
    0x9AA7E0E047E0537AULL, 0x24300C0C3C0C6028ULL, 0xE98C2323AF2305CAULL, 0x5FC57676B3769729ULL,
    0x53741D1D691DE84EULL, 0xFB942525B12535DEULL, 0xFC902424B4243DD8ULL, 0x1B1405051105281EULL,
    0xEDE3F1F112F1DB1CULL, 0x17A56E6ECB6E5779ULL, 0xCB6A9494FE94D45FULL, 0xD8A0282888285DF0ULL,
    0xE1529A9AC89AA47BULL, 0xBB2A8484AE84543FULL, 0xA287E8E86FE8134AULL, 0x4EB6A3A315A371EDULL,
    0xF0214F4F6E4F42BFULL, 0x58C17777B6779F2FULL, 0x036BD3D3B8D3D6D0ULL, 0xBC2E8585AB855C39ULL,
    0x94AFE2E24DE24376ULL, 0xA35552520752AAF1ULL, 0xE4EFF2F21DF2C316ULL, 0xA9328282B082642BULL,
    0xAD5D50500D50BAFDULL, 0x7BF57A7A8F7AF701ULL, 0xCDBC2F2F932F65E2ULL, 0x51CD7474B9748725ULL,
    0xA45153530253A2F7ULL, 0x3EF6B3B345B3F18DULL, 0x3A996161F8612F5BULL, 0x6A86AFAF29AF11C5ULL,
    0xAFE43939DD39D596ULL, 0x8BD43535E135B5BEULL, 0x205FDEDE81DEBEFEULL, 0x5913CDCDDECD2694ULL,
    0x5D7C1F1F631FF842ULL, 0xE85E9999C799BC71ULL, 0x638AACAC26AC09CFULL, 0x648EADAD23AD01C9ULL,
    0x43D57272A772B731ULL, 0xC4B02C2C9C2C7DE8ULL, 0x2953DDDD8EDDA6F4ULL, 0x0A67D0D0B7D0CEDAULL,
    0xB2268787A1874C35ULL, 0x1DC2BEBE7CBE99A3ULL, 0x87655E5E3B5ECAD9ULL, 0x55A2A6A604A659F3ULL,
    0xBE97ECEC7BEC3352ULL, 0x1C10040414042018ULL, 0x683FC6C6F9C67EAEULL, 0x090C03030F03180AULL,
    0x8CD03434E434BDB8ULL, 0xDBCBFBFB30FB8B20ULL, 0x3B4BDBDB90DB96E0ULL, 0x927959592059F2CBULL,
    0x25E2B6B654B6D993ULL, 0x742FC2C2EDC25EB6ULL, 0x0704010105010806ULL, 0xEAE7F0F017F0D31AULL,
    0x9B755A5A2F5AEAC1ULL, 0xB993EDED7EED3B54ULL, 0x52A6A7A701A751F5ULL, 0x2F856666E3661749ULL,
    0xE7842121A52115C6ULL, 0x60E17F7F9E7FDF1FULL, 0x91128A8A988A241BULL, 0xF59C2727BB2725D2ULL,
    0x6F3BC7C7FCC776A8ULL, 0x7A27C0C0E7C04EBAULL, 0xDFA429298D2955F6ULL, 0x1F7BD7D7ACD7F6C8ULL
};

static uint64_t T6[256] = {
    0x769393E593EC4DDEULL, 0x43D9D99AD986EC35ULL, 0x529A9AC89AA47BE1ULL, 0xEEB5B55BB5C1992CULL,
    0x5A9898C298B477EFULL, 0x882222AA220DCCEEULL, 0x0945454C451283C6ULL, 0xD7FCFC2BFCB332CEULL,
    0xD2BABA68BAB9BB01ULL, 0xB56A6ADF6A77610BULL, 0x5BDFDF84DFB6F827ULL, 0x0802020A02100C0EULL,
    0x469F9FD99F8C65FAULL, 0x57DCDC8BDCAEF22EULL, 0x5951510851B2FBAAULL, 0x7959592059F2CB92ULL,
    0x354A4A7F4A6AA1EBULL, 0x5C17174B17B87265ULL, 0xAC2B2B872B45FAD1ULL, 0x2FC2C2EDC25EB674ULL,
    0x6A9494FE94D45FCBULL, 0xF7F4F403F4F302F6ULL, 0xD6BBBB6DBBB1BD06ULL, 0xB6A3A315A371ED4EULL,
    0x956262F762375133ULL, 0xB7E4E453E4736286ULL, 0xD97171A871AF3B4AULL, 0x77D4D4A3D4EEC216ULL,
    0x13CDCDDECD269459ULL, 0xDD7070AD70A73D4DULL, 0x5816164E16B07462ULL, 0xA3E1E142E15B7C9DULL,
    0x394949704972ABE2ULL, 0xF03C3CCC3CFD88B4ULL, 0x27C0C0E7C04EBA7AULL, 0x47D8D89FD88EEA32ULL,
    0x6D5C5C315CDAD589ULL, 0x569B9BCD9BAC7DE6ULL, 0x8EADAD23AD01C964ULL, 0x2E8585AB855C39BCULL,
    0x5153530253A2F7A4ULL, 0xBEA1A11FA161E140ULL, 0xF57A7A8F7AF7017BULL, 0x07C8C8CFC80E8A42ULL,
```

0xB42D2D992D75EEC3ULL, 0xA7E0E047E0537A9AULL, 0x63D1D1B2D1C6DC0DULL, 0xD57272A772B73143ULL,
0xA2A6A604A659F355ULL, 0xB02C2C9C2C7DE8C4ULL, 0x37C4C4F3C46EA266ULL, 0xABE3E348E34B7093ULL,
0xC57676B37697295FULL, 0xFD78788578E70D75ULL, 0xE6B7B751B7D19522ULL, 0xEAB4B45EB4C99F2BULL,
0x2409092D0948363FULL, 0xEC3B3BD73BC59AA1ULL, 0x380E0E360E70242AULL, 0x1941415841329BDAULL,
0x2D4C4C614C5AB5F9ULL, 0x5FDEDE81DEBEFE20ULL, 0xF2B2B240B2F98B39ULL, 0x7A9090EA90F447D7ULL,
0x942525B12535DEFBULL, 0xAEA5A50BA541F95CULL, 0x7BD7D7ACD7F6C81FULL, 0x0C03030F03180A09ULL,
0x4411115511886677ULL, 0x0000000000000000ULL, 0x2BC3C3E8C356B073ULL, 0xB82E2E962E6DE4CAULL,
0x729292E092E44BD9ULL, 0x9BEFEF74EF2B58B7ULL, 0x254E4E6B4E4AB9F7ULL, 0x4812125A12906C7EULL,
0x4E9D9DD39D9C69F4ULL, 0xE97D7D947DCF136EULL, 0x0BCBCBC0CB16804BULL, 0xD43535E135B5BE8BULL,
0x4010105010806070ULL, 0x73D5D5A6D5E6C411ULL, 0x214F4F6E4F42BFF0ULL, 0x429E9EDC9E8463FDULL,
0x294D4D644D52B3FEULL, 0x9EA9A937A921D178ULL, 0x4955551C5592E3B6ULL, 0x3FC6C6F9C67EAE68ULL,
0x67D0D0B7D0CEDA0AULL, 0xF17B7B8A7BFF077CULL, 0x6018187818C05048ULL, 0x669797F197CC55C2ULL,
0x6BD3D3B8D3D6D003ULL, 0xD83636EE36ADB482ULL, 0xBFE6E659E6636E88ULL, 0x3D484875487AADE5ULL,
0x45565613568AE9BFULL, 0x3E8181BF817C21A0ULL, 0x068F8F898F0C058AULL, 0xC17777B6779F2F58ULL,
0x17CCCCDBCC2E925EULL, 0x4A9C9CD69C946FF3ULL, 0xDEB9B967B9A1B108ULL, 0xAFE2E24DE2437694ULL,
0x8AACAC26AC09CF63ULL, 0xDAB8B862B8A9B70FULL, 0xBC2F2F932F65E2CDULL, 0x5415154115A87E6BULL,
0xAAA4A40EA449FF5BULL, 0xED7C7C917CC71569ULL, 0x4FDADA95DA9EE63CULL, 0xE03838D838DD90A8ULL,
0x781E1E661EF0445AULL, 0x2C0B0B270B583A31ULL, 0x1405051105281E1BULL, 0x7FD6D6A9D6FECE18ULL,
0x5014144414A0786CULL, 0xA56E6ECB6E577917ULL, 0xAD6C6CC16C477519ULL, 0xE57E7E9B7ED71967ULL,
0x856666E36617492FULL, 0xD3FDFD2EFDBB34C9ULL, 0xFEB1B14FB1E18130ULL, 0xB3E5E556E57B6481ULL,
0x9D6060FD60275D3DULL, 0x86AFAF29AF11C56AULL, 0x655E5E3B5ECAD987ULL, 0xCC3333FF3385AA99ULL,
0x268787A1874C35B2ULL, 0x03C9C9CAC9068C45ULL, 0xE7F0F017F0D31AEAULL, 0x695D5D345DD2D38EULL,
0xA96D6DC46D4F731EULL, 0xFC3F3FC33FE582BDULL, 0x1A8888928834179FULL, 0x0E8D8D838D1C0984ULL,
0x3BC7C7FCC776A86FULL, 0xFBF7F70CF7EB08FFULL, 0x741D1D691DE84E53ULL, 0x83E9E96AE91B4CA5ULL,
0x97ECEC7BEC3352BEULL, 0x93EDED7EED3B54B9ULL, 0x3A8080BA807427A7ULL, 0xA429298D2955F6DFULL,
0x9C2727BB2725D2F5ULL, 0x1BCFCFD4CF369857ULL, 0x5E9999C799BC71E8ULL, 0x9AA8A832A829D77FULL,
0x5D50500D50BAFDADULL, 0x3C0F0F330F78222DULL, 0xDC3737EB37A5B285ULL, 0x902424B4243DD8FCULL,
0xA0282888285DF0D8ULL, 0xC03030F0309DA090ULL, 0x6E9595FB95DC59CCULL, 0x6FD2D2BDD2DED604ULL,
0xF83E3EC63EED84BAULL, 0x715B5B2A5BE2C79CULL, 0x1D40405D403A9DDDULL, 0x368383B5836C2DAEULL,
0xF6B3B345B3F18D3EULL, 0xB96969D0696F6B02ULL, 0x415757165782EFB8ULL, 0x7C1F1F631FF8425DULL,
0x1C07071B07381215ULL, 0x701C1C6C1CE04854ULL, 0x128A8A988A241B91ULL, 0xCABCBC76BC89AF13ULL,
0x802020A0201DC0E0ULL, 0x8BEBEB60EB0B40ABULL, 0x1FCECED1CE3E9E50ULL, 0x028E8E8C8E04038DULL,
0x96ABAB3DAB31DD76ULL, 0x9FEEEE71EE235EB0ULL, 0xC43131F53195A697ULL, 0xB2A2A210A279EB49ULL,
0xD17373A273BF3744ULL, 0xC3F9F93AF99B2CD5ULL, 0x0FCACAC5CA1E864CULL, 0xE83A3AD23ACD9CA6ULL,
0x681A1A721AD05C46ULL, 0xCBFBFB30FB8B20DBULL, 0x340D0D390D682E23ULL, 0x23C1C1E2C146BC7DULL,
0xDFFEFE21FEA33EC0ULL, 0xCFFAFA35FA8326DCULL, 0xEFF2F21DF2C316E4ULL, 0xA16F6FCE6F5F7F10ULL,
0xCEBDBD73BD81A914ULL, 0x629696F496C453C5ULL, 0x53DDDD8EDDA6F429ULL, 0x11434352432297D4ULL,
0x5552520752AAF1A3ULL, 0xE2B6B654B6D99325ULL, 0x200808208403038ULL, 0xEBF3F318F3CB10E3ULL,
0x82AEAE2CAE19C36DULL, 0xC2BEBE7CBE99A31DULL, 0x6419197D19C8564FULL, 0x1E898997893C1198ULL,
0xC83232FA328DAC9EULL, 0x982626BE262DD4F2ULL, 0xFAB0B04AB0E98737ULL, 0x8FEAEA65EA0346ACULL,
0x314B4B7A4B62A7ECULL, 0x8D6464E964074521ULL, 0x2A8484AE84543FBBULL, 0x328282B082642BA9ULL,
0xB16B6BDA6B7F670CULL, 0xF3F5F506F5FB04F1ULL, 0xF979798079EF0B72ULL, 0xC6BFBF79BF91A51AULL,
0x0401010501080607ULL, 0x615F5F3E5FC2DF80ULL, 0xC97575BC758F2356ULL, 0x916363F2633F5734ULL,
0x6C1B1B771BD85A41ULL, 0x8C2323AF2305CAE9ULL, 0xF43D3DC93DF58EB3ULL, 0xBD6868D568676D05ULL,
0xA82A2A822A4DFCD6ULL, 0x896565EC650F4326ULL, 0x87E8E86FE8134AA2ULL, 0x7E9191EF91FC41D0ULL,
0xFFF6F609F6E30EF8ULL, 0xDBFFFF24FFAB38C7ULL, 0x4C13135F13986A79ULL, 0x7D58582558FACD95ULL,
0xE3F1F112F1DB1CEDULL, 0x0147474647028FC8ULL, 0x280A0A220A503C36ULL, 0xE17F7F9E7FDF1F60ULL,

```
    0x33C5C5F6C566A461ULL, 0xA6A7A701A751F552ULL, 0xBBE7E75CE76B688FULL, 0x996161F8612F5B3AULL,
    0x755A5A2F5AEAC19BULL, 0x1806061E06301412ULL, 0x05464643460A89CFULL, 0x0D444449441A85C1ULL,
    0x15424257422A91D3ULL, 0x100404140420181CULL, 0xBAA0A01AA069E747ULL, 0x4BDBDB90DB96E03BULL,
    0xE43939DD39D596AFULL, 0x228686A4864433B5ULL, 0x4D545419549AE5B1ULL, 0x92AAAA38AA39DB71ULL,
    0x0A8C8C868C140F83ULL, 0xD03434E434BDB88CULL, 0x842121A52115C6E7ULL, 0x168B8B9D8B2C1D96ULL,
    0xC7F8F83FF8932AD2ULL, 0x300C0C3C0C602824ULL, 0xCD7474B974872551ULL, 0x816767E6671F4F28ULL
};

static uint64_t T7[256] = {
    0x6868D568676D05BDULL, 0x8D8D838D1C09840EULL, 0xCACAC5CA1E864C0FULL, 0x4D4D644D52B3FE29ULL,
    0x7373A273BF3744D1ULL, 0x4B4B7A4B62A7EC31ULL, 0x4E4E6B4E4AB9F725ULL, 0x2A2A822A4DFCD6A8ULL,
    0xD4D4A3D4EEC21677ULL, 0x52520752AAF1A355ULL, 0x2626BE262DD4F298ULL, 0xB3B345B3F18D3EF6ULL,
    0x545419549AE5B14DULL, 0x1E1E661EF0445A78ULL, 0x19197D19C8564F64ULL, 0x1F1F631FF8425D7CULL,
    0x2222AA220DCCEE88ULL, 0x03030F03180A090CULL, 0x464643460A89CF05ULL, 0x3D3DC93DF58EB3F4ULL,
    0x2D2D992D75EEC3B4ULL, 0x4A4A7F4A6AA1EB35ULL, 0x53530253A2F7A451ULL, 0x8383B5836C2DAE36ULL,
    0x13135F13986A794CULL, 0x8A8A988A241B9112ULL, 0xB7B751B7D19522E6ULL, 0xD5D5A6D5E6C41173ULL,
    0x2525B12535DEFB94ULL, 0x79798079EF0B72F9ULL, 0xF5F506F5FB04F1F3ULL, 0xBDBD73BD81A914CEULL,
    0x58582558FACD957DULL, 0x2F2F932F65E2CDBCULL, 0x0D0D390D682E2334ULL, 0x02020A02100C0E08ULL,
    0xEDED7EED3B54B993ULL, 0x51510851B2FBAA59ULL, 0x9E9EDC9E8463FD42ULL, 0x1111551188667744ULL,
    0xF2F21DF2C316E4EFULL, 0x3E3EC63EED84BAF8ULL, 0x55551C5592E3B649ULL, 0x5E5E3B5ECAD98765ULL,
    0xD1D1B2D1C6DC0D63ULL, 0x16164E16B0746258ULL, 0x3C3CCC3CFD88B4F0ULL, 0x6666E36617492F85ULL,
    0x7070AD70A73D4DDDULL, 0x5D5D345DD2D38E69ULL, 0xF3F318F3CB10E3EBULL, 0x45454C451283C609ULL,
    0x40405D403A9DDD1DULL, 0xCCCCDBCC2E925E17ULL, 0xE8E86FE8134AA287ULL, 0x9494FE94D45FCB6AULL,
    0x565613568AE9BF45ULL, 0x0808280840303820ULL, 0xCECED1CE3E9E501FULL, 0x1A1A721AD05C4668ULL,
    0x3A3AD23ACD9CA6E8ULL, 0xD2D2BDD2DED6046FULL, 0xE1E142E15B7C9DA3ULL, 0xDFDF84DFB6F8275BULL,
    0xB5B55BB5C1992CEEULL, 0x3838D838DD90A8E0ULL, 0x6E6ECB6E577917A5ULL, 0x0E0E360E70242A38ULL,
    0xE5E556E57B6481B3ULL, 0xF4F403F4F302F6F7ULL, 0xF9F93AF99B2CD5C3ULL, 0x8686A4864433B522ULL,
    0xE9E96AE91B4CA583ULL, 0x4F4F6E4F42BFF021ULL, 0xD6D6A9D6FECE187FULL, 0x8585AB855C39BC2EULL,
    0x2323AF2305CAE98CULL, 0xCFCFD4CF3698571BULL, 0x3232FA328DAC9EC8ULL, 0x9999C799BC71E85EULL,
    0x3131F53195A697C4ULL, 0x14144414A0786C50ULL, 0xAEAE2CAE19C36D82ULL, 0xEEEE71EE235EB09FULL,
    0xC8C8CFC80E8A4207ULL, 0x484875487AADE53DULL, 0xD3D3B8D3D6D0036BULL, 0x3030F0309DA090C0ULL,
    0xA1A11FA161E140BEULL, 0x9292E092E44BD972ULL, 0x41415841329BDA19ULL, 0xB1B14FB1E18130FEULL,
    0x18187818C0504860ULL, 0xC4C4F3C46EA26637ULL, 0x2C2C9C2C7DE8C4B0ULL, 0x7171A871AF3B4AD9ULL,
    0x7272A772B73143D5ULL, 0x444449441A85C10DULL, 0x15154115A87E6B54ULL, 0xFDFD2EFDBB34C9D3ULL,
    0x3737EB37A5B285DCULL, 0xBEBE7CBE99A31DC2ULL, 0x5F5F3E5FC2DF8061ULL, 0xAAAA38AA39DB7192ULL,
    0x9B9BCD9BAC7DE656ULL, 0x8888928834179F1AULL, 0xD8D89FD88EEA3247ULL, 0xABAB3DAB31DD7696ULL,
    0x898997893C11981EULL, 0x9C9CD69C946FF34AULL, 0xFAFA35FA8326DCCFULL, 0x6060FD60275D3D9DULL,
    0xEAEA65EA0346AC8FULL, 0xBCBC76BC89AF13CAULL, 0x6262F76237513395ULL, 0x0C0C3C0C60282430ULL,
    0x2424B4243DD8FC90ULL, 0xA6A604A659F355A2ULL, 0xA8A832A829D77F9AULL, 0xECEC7BEC3352BE97ULL,
    0x6767E6671F4F2881ULL, 0x2020A0201DC0E080ULL, 0xDBDB90DB96E03B4BULL, 0x7C7C917CC71569EDULL,
    0x282888285DF0D8A0ULL, 0xDDDD8EDDA6F42953ULL, 0xACAC26AC09CF638AULL, 0x5B5B2A5BE2C79C71ULL,
    0x3434E434BDB88CD0ULL, 0x7E7E9B7ED71967E5ULL, 0x101050108060704 0ULL, 0xF1F112F1DB1CEDE3ULL,
    0x7B7B8A7BFF077CF1ULL, 0x8F8F898F0C058A06ULL, 0x6363F2633F573491ULL, 0xA0A01AA069E747BAULL,
    0x05051105281E1B14ULL, 0x9A9AC89AA47BE152ULL, 0x434352432297D411ULL, 0x7777B6779F2F58C1ULL,
    0x2121A52115C6E784ULL, 0xBFBF79BF91A51AC6ULL, 0x2727BB2725D2F59CULL, 0x09092D0948363F24ULL,
    0xC3C3E8C356B0732BULL, 0x9F9FD99F8C65FA46ULL, 0xB6B654B6D99325E2ULL, 0xD7D7ACD7F6C81F7BULL,
    0x29298D2955F6DFA4ULL, 0xC2C2EDC25EB6742FULL, 0xEBEB60EB0B40AB8BULL, 0xC0C0E7C04EBA7A27ULL,
```

```
    0xA4A40EA449FF5BAAULL, 0x8B8B9D8B2C1D9616ULL, 0x8C8C868C140F830AULL, 0x1D1D691DE84E5374ULL,
    0xFBFB30FB8B20DBCBULL, 0xFFFF24FFAB38C7DBULL, 0xC1C1E2C146BC7D23ULL, 0xB2B240B2F98B39F2ULL,
    0x9797F197CC55C266ULL, 0x2E2E962E6DE4CAB8ULL, 0xF8F83FF8932AD2C7ULL, 0x6565EC650F432689ULL,
    0xF6F609F6E30EF8FFULL, 0x7575BC758F2356C9ULL, 0x07071B073812151CULL, 0x0404140420181C10ULL,
    0x4949704972ABE239ULL, 0x3333FF3385AA99CCULL, 0xE4E453E4736286B7ULL, 0xD9D99AD986EC3543ULL,
    0xB9B967B9A1B108DEULL, 0xD0D0B7D0CEDA0A67ULL, 0x424257422A91D315ULL, 0xC7C7FCC776A86F3BULL,
    0x6C6CC16C477519ADULL, 0x9090EA90F447D77AULL, 0x0000000000000000ULL, 0x8E8E8C8E04038D02ULL,
    0x6F6FCE6F5F7F10A1ULL, 0x50500D50BAFDAD5DULL, 0x0101050108060704ULL, 0xC5C5F6C566A46133ULL,
    0xDADA95DA9EE63C4FULL, 0x47474647028FC801ULL, 0x3F3FC33FE582BDFCULL, 0xCDCDDECD26945913ULL,
    0x6969D0696F6B02B9ULL, 0xA2A210A279EB49B2ULL, 0xE2E24DE2437694AFULL, 0x7A7A8F7AF7017BF5ULL,
    0xA7A701A751F552A6ULL, 0xC6C6F9C67EAE683FULL, 0x9393E593EC4DDE76ULL, 0x0F0F330F78222D3CULL,
    0x0A0A220A503C3628ULL, 0x06061E0630141218ULL, 0xE6E659E6636E88BFULL, 0x2B2B872B45FAD1ACULL,
    0x9696F496C453C562ULL, 0xA3A315A371ED4EB6ULL, 0x1C1C6C1CE0485470ULL, 0xAFAF29AF11C56A86ULL,
    0x6A6ADF6A77610BB5ULL, 0x12125A12906C7E48ULL, 0x8484AE84543FBB2AULL, 0x3939DD39D596AFE4ULL,
    0xE7E75CE76B688FBBULL, 0xB0B04AB0E98737FAULL, 0x8282B082642BA932ULL, 0xF7F70CF7EB08FFFBULL,
    0xFEFE21FEA33EC0DFULL, 0x9D9DD39D9C69F44EULL, 0x8787A1874C35B226ULL, 0x5C5C315CDAD5896DULL,
    0x8181BF817C21A03EULL, 0x3535E135B5BE8BD4ULL, 0xDEDE81DEBEFE205FULL, 0xB4B45EB4C99F2BEAULL,
    0xA5A50BA541F95CAEULL, 0xFCFC2BFCB332CED7ULL, 0x8080BA807427A73AULL, 0xEFEF74EF2B58B79BULL,
    0xCBCBC0CB16804B0BULL, 0xBBBB6DBBB1BD06D6ULL, 0x6B6BDA6B7F670CB1ULL, 0x7676B37697295FC5ULL,
    0xBABA68BAB9BB01D2ULL, 0x5A5A2F5AEAC19B75ULL, 0x7D7D947DCF136EE9ULL, 0x78788578E70D75FDULL,
    0x0B0B270B583A312CULL, 0x9595FB95DC59CC6EULL, 0xE3E348E34B7093ABULL, 0xADAD23AD01C9648EULL,
    0x7474B974872551CDULL, 0x9898C298B477EF5AULL, 0x3B3BD73BC59AA1ECULL, 0x3636EE36ADB482D8ULL,
    0x6464E9640745218DULL, 0x6D6DC46D4F731EA9ULL, 0xDCDC8BDCAEF22E57ULL, 0xF0F017F0D31AEAE7ULL,
    0x59592059F2CB9279ULL, 0xA9A937A921D1789EULL, 0x4C4C614C5AB5F92DULL, 0x17174B17B872655CULL,
    0x7F7F9E7FDF1F60E1ULL, 0x9191EF91FC41D07EULL, 0xB8B862B8A9B70FDAULL, 0xC9C9CAC9068C4503ULL,
    0x5757165782EFB841ULL, 0x1B1B771BD85A416CULL, 0xE0E047E0537A9AA7ULL, 0x6161F8612F5B3A99ULL
};

struct Dstu8845Ctx_st {
    uint64_t  S[16];
    uint64_t  r[2];
    uint64_t  key[8];
    uint64_t  iv[4];
    uint8_t   key_size;
};

static inline void next_stream(Dstu8845Ctx *ctx, uint64_t *out_stream)
{
    uint64_t fsmtmp;

    ctx->S[0] = a_mul(ctx->S[0]) ^ ctx->S[13] ^ ainv_mul(ctx->S[11]);
    fsmtmp = ctx->r[1] + ctx->S[13];
    ctx->r[1] = T(ctx->r[0]);
    ctx->r[0] = fsmtmp;
    out_stream[0] = (ctx->r[0] + ctx->S[0]) ^ ctx->r[1] ^ ctx->S[1];

    ctx->S[1] = a_mul(ctx->S[1]) ^ ctx->S[14] ^ ainv_mul(ctx->S[12]);
```

```
fsmtmp = ctx->r[1] + ctx->S[14];
ctx->r[1] = T(ctx->r[0]);
ctx->r[0] = fsmtmp;
out_stream[1] = (ctx->r[0] + ctx->S[1]) ^ ctx->r[1] ^ ctx->S[2];

ctx->S[2] = a_mul(ctx->S[2]) ^ ctx->S[15] ^ ainv_mul(ctx->S[13]);
fsmtmp = ctx->r[1] + ctx->S[15];
ctx->r[1] = T(ctx->r[0]);
ctx->r[0] = fsmtmp;
out_stream[2] = (ctx->r[0] + ctx->S[2]) ^ ctx->r[1] ^ ctx->S[3];

ctx->S[3] = a_mul(ctx->S[3]) ^ ctx->S[0] ^ ainv_mul(ctx->S[14]);
fsmtmp = ctx->r[1] + ctx->S[0];
ctx->r[1] = T(ctx->r[0]);
ctx->r[0] = fsmtmp;
out_stream[3] = (ctx->r[0] + ctx->S[3]) ^ ctx->r[1] ^ ctx->S[4];

ctx->S[4] = a_mul(ctx->S[4]) ^ ctx->S[1] ^ ainv_mul(ctx->S[15]);
fsmtmp = ctx->r[1] + ctx->S[1];
ctx->r[1] = T(ctx->r[0]);
ctx->r[0] = fsmtmp;
out_stream[4] = (ctx->r[0] + ctx->S[4]) ^ ctx->r[1] ^ ctx->S[5];

ctx->S[5] = a_mul(ctx->S[5]) ^ ctx->S[2] ^ ainv_mul(ctx->S[0]);
fsmtmp = ctx->r[1] + ctx->S[2];
ctx->r[1] = T(ctx->r[0]);
ctx->r[0] = fsmtmp;
out_stream[5] = (ctx->r[0] + ctx->S[5]) ^ ctx->r[1] ^ ctx->S[6];

ctx->S[6] = a_mul(ctx->S[6]) ^ ctx->S[3] ^ ainv_mul(ctx->S[1]);
fsmtmp = ctx->r[1] + ctx->S[3];
ctx->r[1] = T(ctx->r[0]);
ctx->r[0] = fsmtmp;
out_stream[6] = (ctx->r[0] + ctx->S[6]) ^ ctx->r[1] ^ ctx->S[7];

ctx->S[7] = a_mul(ctx->S[7]) ^ ctx->S[4] ^ ainv_mul(ctx->S[2]);
fsmtmp = ctx->r[1] + ctx->S[4];
ctx->r[1] = T(ctx->r[0]);
ctx->r[0] = fsmtmp;
out_stream[7] = (ctx->r[0] + ctx->S[7]) ^ ctx->r[1] ^ ctx->S[8];

ctx->S[8] = a_mul(ctx->S[8]) ^ ctx->S[5] ^ ainv_mul(ctx->S[3]);
fsmtmp = ctx->r[1] + ctx->S[5];
ctx->r[1] = T(ctx->r[0]);
ctx->r[0] = fsmtmp;
out_stream[8] = (ctx->r[0] + ctx->S[8]) ^ ctx->r[1] ^ ctx->S[9];
```

```
    ctx->S[9] = a_mul(ctx->S[9]) ^ ctx->S[6] ^ ainv_mul(ctx->S[4]);
    fsmtmp = ctx->r[1] + ctx->S[6];
    ctx->r[1] = T(ctx->r[0]);
    ctx->r[0] = fsmtmp;
    out_stream[9] = (ctx->r[0] + ctx->S[9]) ^ ctx->r[1] ^ ctx->S[10];

    ctx->S[10] = a_mul(ctx->S[10]) ^ ctx->S[7] ^ ainv_mul(ctx->S[5]);
    fsmtmp = ctx->r[1] + ctx->S[7];
    ctx->r[1] = T(ctx->r[0]);
    ctx->r[0] = fsmtmp;
    out_stream[10] = (ctx->r[0] + ctx->S[10]) ^ ctx->r[1] ^ ctx->S[11];

    ctx->S[11] = a_mul(ctx->S[11]) ^ ctx->S[8] ^ ainv_mul(ctx->S[6]);
    fsmtmp = ctx->r[1] + ctx->S[8];
    ctx->r[1] = T(ctx->r[0]);
    ctx->r[0] = fsmtmp;
    out_stream[11] = (ctx->r[0] + ctx->S[11]) ^ ctx->r[1] ^ ctx->S[12];

    ctx->S[12] = a_mul(ctx->S[12]) ^ ctx->S[9] ^ ainv_mul(ctx->S[7]);
    fsmtmp = ctx->r[1] + ctx->S[9];
    ctx->r[1] = T(ctx->r[0]);
    ctx->r[0] = fsmtmp;
    out_stream[12] = (ctx->r[0] + ctx->S[12]) ^ ctx->r[1] ^ ctx->S[13];

    ctx->S[13] = a_mul(ctx->S[13]) ^ ctx->S[10] ^ ainv_mul(ctx->S[8]);
    fsmtmp = ctx->r[1] + ctx->S[10];
    ctx->r[1] = T(ctx->r[0]);
    ctx->r[0] = fsmtmp;
    out_stream[13] = (ctx->r[0] + ctx->S[13]) ^ ctx->r[1] ^ ctx->S[14];

    ctx->S[14] = a_mul(ctx->S[14]) ^ ctx->S[11] ^ ainv_mul(ctx->S[9]);
    fsmtmp = ctx->r[1] + ctx->S[11];
    ctx->r[1] = T(ctx->r[0]);
    ctx->r[0] = fsmtmp;
    out_stream[14] = (ctx->r[0] + ctx->S[14]) ^ ctx->r[1] ^ ctx->S[15];

    ctx->S[15] = a_mul(ctx->S[15]) ^ ctx->S[12] ^ ainv_mul(ctx->S[10]);
    fsmtmp = ctx->r[1] + ctx->S[12];
    ctx->r[1] = T(ctx->r[0]);
    ctx->r[0] = fsmtmp;
    out_stream[15] = (ctx->r[0] + ctx->S[15]) ^ ctx->r[1] ^ ctx->S[0];
}

static inline void next_stream_full_crypt(Dstu8845Ctx *ctx, uint64_t *in, uint64_t *out)
{
    uint64_t fsmtmp;
```

```
ctx->S[0] = a_mul(ctx->S[0]) ^ ctx->S[13] ^ ainv_mul(ctx->S[11]);
fsmtmp = ctx->r[1] + ctx->S[13];
ctx->r[1] = T(ctx->r[0]);
ctx->r[0] = fsmtmp;
out[0] = in[0] ^ (ctx->r[0] + ctx->S[0]) ^ ctx->r[1] ^ ctx->S[1];

ctx->S[1] = a_mul(ctx->S[1]) ^ ctx->S[14] ^ ainv_mul(ctx->S[12]);
fsmtmp = ctx->r[1] + ctx->S[14];
ctx->r[1] = T(ctx->r[0]);
ctx->r[0] = fsmtmp;
out[1] = in[1] ^ (ctx->r[0] + ctx->S[1]) ^ ctx->r[1] ^ ctx->S[2];

ctx->S[2] = a_mul(ctx->S[2]) ^ ctx->S[15] ^ ainv_mul(ctx->S[13]);
fsmtmp = ctx->r[1] + ctx->S[15];
ctx->r[1] = T(ctx->r[0]);
ctx->r[0] = fsmtmp;
out[2] = in[2] ^ (ctx->r[0] + ctx->S[2]) ^ ctx->r[1] ^ ctx->S[3];

ctx->S[3] = a_mul(ctx->S[3]) ^ ctx->S[0] ^ ainv_mul(ctx->S[14]);
fsmtmp = ctx->r[1] + ctx->S[0];
ctx->r[1] = T(ctx->r[0]);
ctx->r[0] = fsmtmp;
out[3] = in[3] ^ (ctx->r[0] + ctx->S[3]) ^ ctx->r[1] ^ ctx->S[4];

ctx->S[4] = a_mul(ctx->S[4]) ^ ctx->S[1] ^ ainv_mul(ctx->S[15]);
fsmtmp = ctx->r[1] + ctx->S[1];
ctx->r[1] = T(ctx->r[0]);
ctx->r[0] = fsmtmp;
out[4] = in[4] ^ (ctx->r[0] + ctx->S[4]) ^ ctx->r[1] ^ ctx->S[5];

ctx->S[5] = a_mul(ctx->S[5]) ^ ctx->S[2] ^ ainv_mul(ctx->S[0]);
fsmtmp = ctx->r[1] + ctx->S[2];
ctx->r[1] = T(ctx->r[0]);
ctx->r[0] = fsmtmp;
out[5] = in[5] ^(ctx->r[0] + ctx->S[5]) ^ ctx->r[1] ^ ctx->S[6];

ctx->S[6] = a_mul(ctx->S[6]) ^ ctx->S[3] ^ ainv_mul(ctx->S[1]);
fsmtmp = ctx->r[1] + ctx->S[3];
ctx->r[1] = T(ctx->r[0]);
ctx->r[0] = fsmtmp;
out[6] = in[6] ^(ctx->r[0] + ctx->S[6]) ^ ctx->r[1] ^ ctx->S[7];

ctx->S[7] = a_mul(ctx->S[7]) ^ ctx->S[4] ^ ainv_mul(ctx->S[2]);
fsmtmp = ctx->r[1] + ctx->S[4];
ctx->r[1] = T(ctx->r[0]);
ctx->r[0] = fsmtmp;
out[7] = in[7] ^(ctx->r[0] + ctx->S[7]) ^ ctx->r[1] ^ ctx->S[8];
```

```
ctx->S[8] = a_mul(ctx->S[8]) ^ ctx->S[5] ^ ainv_mul(ctx->S[3]);
fsmtmp = ctx->r[1] + ctx->S[5];
ctx->r[1] = T(ctx->r[0]);
ctx->r[0] = fsmtmp;
out[8] = in[8] ^(ctx->r[0] + ctx->S[8]) ^ ctx->r[1] ^ ctx->S[9];

ctx->S[9] = a_mul(ctx->S[9]) ^ ctx->S[6] ^ ainv_mul(ctx->S[4]);
fsmtmp = ctx->r[1] + ctx->S[6];
ctx->r[1] = T(ctx->r[0]);
ctx->r[0] = fsmtmp;
out[9] = in[9] ^(ctx->r[0] + ctx->S[9]) ^ ctx->r[1] ^ ctx->S[10];

ctx->S[10] = a_mul(ctx->S[10]) ^ ctx->S[7] ^ ainv_mul(ctx->S[5]);
fsmtmp = ctx->r[1] + ctx->S[7];
ctx->r[1] = T(ctx->r[0]);
ctx->r[0] = fsmtmp;
out[10] = in[10] ^(ctx->r[0] + ctx->S[10]) ^ ctx->r[1] ^ ctx->S[11];

ctx->S[11] = a_mul(ctx->S[11]) ^ ctx->S[8] ^ ainv_mul(ctx->S[6]);
fsmtmp = ctx->r[1] + ctx->S[8];
ctx->r[1] = T(ctx->r[0]);
ctx->r[0] = fsmtmp;
out[11] = in[11] ^ (ctx->r[0] + ctx->S[11]) ^ ctx->r[1] ^ ctx->S[12];

ctx->S[12] = a_mul(ctx->S[12]) ^ ctx->S[9] ^ ainv_mul(ctx->S[7]);
fsmtmp = ctx->r[1] + ctx->S[9];
ctx->r[1] = T(ctx->r[0]);
ctx->r[0] = fsmtmp;
out[12] = in[12] ^ (ctx->r[0] + ctx->S[12]) ^ ctx->r[1] ^ ctx->S[13];

ctx->S[13] = a_mul(ctx->S[13]) ^ ctx->S[10] ^ ainv_mul(ctx->S[8]);
fsmtmp = ctx->r[1] + ctx->S[10];
ctx->r[1] = T(ctx->r[0]);
ctx->r[0] = fsmtmp;
out[13] = in[13] ^ (ctx->r[0] + ctx->S[13]) ^ ctx->r[1] ^ ctx->S[14];

ctx->S[14] = a_mul(ctx->S[14]) ^ ctx->S[11] ^ ainv_mul(ctx->S[9]);
fsmtmp = ctx->r[1] + ctx->S[11];
ctx->r[1] = T(ctx->r[0]);
ctx->r[0] = fsmtmp;
out[14] = in[14] ^ (ctx->r[0] + ctx->S[14]) ^ ctx->r[1] ^ ctx->S[15];

ctx->S[15] = a_mul(ctx->S[15]) ^ ctx->S[12] ^ ainv_mul(ctx->S[10]);
fsmtmp = ctx->r[1] + ctx->S[12];
ctx->r[1] = T(ctx->r[0]);
ctx->r[0] = fsmtmp;
```

```
    out[15] = in[15] ^ (ctx->r[0] + ctx->S[15]) ^ ctx->r[1] ^ ctx->S[0];
}

Dstu8845Ctx *dstu8845_alloc()
{
    Dstu8845Ctx *ctx = NULL;

    ctx = calloc(1, sizeof(Dstu8845Ctx));
    if (ctx == NULL) {
        return NULL;
    }
    memset(ctx, 0, sizeof(Dstu8845Ctx));

    return ctx;
}

int dstu8845_init(Dstu8845Ctx *ctx, const uint64_t *key, uint8_t key_size, const uint64_t *iv)
{
    if (key_size == 32) {
        ctx->key_size = key_size;
        memcpy(ctx->iv, iv, 32);
        memcpy(ctx->key, key, 32);
        ctx->S[0] = ctx->key[3] ^ ctx->iv[0];
        ctx->S[1] = ctx->key[2];
        ctx->S[2] = ctx->key[1] ^ ctx->iv[1];
        ctx->S[3] = ctx->key[0] ^ ctx->iv[2];
        ctx->S[4] = ctx->key[3];
        ctx->S[5] = ctx->key[2] ^ ctx->iv[3];
        ctx->S[6] = ~ctx->key[1];
        ctx->S[7] = ~ctx->key[0];
        ctx->S[8] = ctx->key[3];
        ctx->S[9] = ctx->key[2];
        ctx->S[10] = ~ctx->key[1];
        ctx->S[11] = ctx->key[0];
        ctx->S[12] = ctx->key[3];
        ctx->S[13] = ~ctx->key[2];
        ctx->S[14] = ctx->key[1];
        ctx->S[15] = ~ctx->key[0];
    } else if (key_size == 64) {
        ctx->key_size = key_size;
        memcpy(ctx->iv, iv, 32);
        memcpy(ctx->key, key, 64);
        ctx->S[0] = ctx->key[7] ^ ctx->iv[0];
        ctx->S[1] = ctx->key[6];
        ctx->S[2] = ctx->key[5];
        ctx->S[3] = ctx->key[4] ^ ctx->iv[1];
        ctx->S[4] = ctx->key[3];
```

```
    ctx->S[5] = ctx->key[2] ^ ctx->iv[2];
    ctx->S[6] = ctx->key[1];
    ctx->S[7] = ~ctx->key[0];
    ctx->S[8] = ctx->key[4] ^ ctx->iv[3];
    ctx->S[9] = ~ctx->key[6];
    ctx->S[10] = ctx->key[5];
    ctx->S[11] = ~ctx->key[7];
    ctx->S[12] = ctx->key[3];
    ctx->S[13] = ctx->key[2];
    ctx->S[14] = ~ctx->key[1];
    ctx->S[15] = ctx->key[0];
} else {
    return 0;
}

ctx->r[0] = 0;
ctx->r[1] = 0;
for (int i = 0; i < 2; i++) {
    static uint64_t outfrom_fsm, fsmtmp;

    outfrom_fsm = (ctx->r[0] + ctx->S[15]) ^ ctx->r[1];
    ctx->S[0] = a_mul(ctx->S[0]) ^ ctx->S[13] ^ ainv_mul(ctx->S[11]) ^ outfrom_fsm;
    fsmtmp = ctx->r[1] + ctx->S[13];
    ctx->r[1] = T(ctx->r[0]);
    ctx->r[0] = fsmtmp;

    outfrom_fsm = (ctx->r[0] + ctx->S[0]) ^ ctx->r[1];
    ctx->S[1] = a_mul(ctx->S[1]) ^ ctx->S[14] ^ ainv_mul(ctx->S[12]) ^ outfrom_fsm;
    fsmtmp = ctx->r[1] + ctx->S[14];
    ctx->r[1] = T(ctx->r[0]);
    ctx->r[0] = fsmtmp;

    outfrom_fsm = (ctx->r[0] + ctx->S[1]) ^ ctx->r[1];
    ctx->S[2] = a_mul(ctx->S[2]) ^ ctx->S[15] ^ ainv_mul(ctx->S[13]) ^ outfrom_fsm;
    fsmtmp = ctx->r[1] + ctx->S[15];
    ctx->r[1] = T(ctx->r[0]);
    ctx->r[0] = fsmtmp;

    outfrom_fsm = (ctx->r[0] + ctx->S[2]) ^ ctx->r[1];
    ctx->S[3] = a_mul(ctx->S[3]) ^ ctx->S[0] ^ ainv_mul(ctx->S[14]) ^ outfrom_fsm;
    fsmtmp = ctx->r[1] + ctx->S[0];
    ctx->r[1] = T(ctx->r[0]);
    ctx->r[0] = fsmtmp;

    outfrom_fsm = (ctx->r[0] + ctx->S[3]) ^ ctx->r[1];
    ctx->S[4] = a_mul(ctx->S[4]) ^ ctx->S[1] ^ ainv_mul(ctx->S[15]) ^ outfrom_fsm;
    fsmtmp = ctx->r[1] + ctx->S[1];
```

```
ctx->r[1] = T(ctx->r[0]);
ctx->r[0] = fsmtmp;

outfrom_fsm = (ctx->r[0] + ctx->S[4]) ^ ctx->r[1];
ctx->S[5] = a_mul(ctx->S[5]) ^ ctx->S[2] ^ ainv_mul(ctx->S[0]) ^ outfrom_fsm;
fsmtmp = ctx->r[1] + ctx->S[2];
ctx->r[1] = T(ctx->r[0]);
ctx->r[0] = fsmtmp;

outfrom_fsm = (ctx->r[0] + ctx->S[5]) ^ ctx->r[1];
ctx->S[6] = a_mul(ctx->S[6]) ^ ctx->S[3] ^ ainv_mul(ctx->S[1]) ^ outfrom_fsm;
fsmtmp = ctx->r[1] + ctx->S[3];
ctx->r[1] = T(ctx->r[0]);
ctx->r[0] = fsmtmp;

outfrom_fsm = (ctx->r[0] + ctx->S[6]) ^ ctx->r[1];
ctx->S[7] = a_mul(ctx->S[7]) ^ ctx->S[4] ^ ainv_mul(ctx->S[2]) ^ outfrom_fsm;
fsmtmp = ctx->r[1] + ctx->S[4];
ctx->r[1] = T(ctx->r[0]);
ctx->r[0] = fsmtmp;

outfrom_fsm = (ctx->r[0] + ctx->S[7]) ^ ctx->r[1];
ctx->S[8] = a_mul(ctx->S[8]) ^ ctx->S[5] ^ ainv_mul(ctx->S[3]) ^ outfrom_fsm;
fsmtmp = ctx->r[1] + ctx->S[5];
ctx->r[1] = T(ctx->r[0]);
ctx->r[0] = fsmtmp;

outfrom_fsm = (ctx->r[0] + ctx->S[8]) ^ ctx->r[1];
ctx->S[9] = a_mul(ctx->S[9]) ^ ctx->S[6] ^ ainv_mul(ctx->S[4]) ^ outfrom_fsm;
fsmtmp = ctx->r[1] + ctx->S[6];
ctx->r[1] = T(ctx->r[0]);
ctx->r[0] = fsmtmp;

outfrom_fsm = (ctx->r[0] + ctx->S[9]) ^ ctx->r[1];
ctx->S[10] = a_mul(ctx->S[10]) ^ ctx->S[7] ^ ainv_mul(ctx->S[5]) ^ outfrom_fsm;
fsmtmp = ctx->r[1] + ctx->S[7];
ctx->r[1] = T(ctx->r[0]);
ctx->r[0] = fsmtmp;

outfrom_fsm = (ctx->r[0] + ctx->S[10]) ^ ctx->r[1];
ctx->S[11] = a_mul(ctx->S[11]) ^ ctx->S[8] ^ ainv_mul(ctx->S[6]) ^ outfrom_fsm;
fsmtmp = ctx->r[1] + ctx->S[8];
ctx->r[1] = T(ctx->r[0]);
ctx->r[0] = fsmtmp;

outfrom_fsm = (ctx->r[0] + ctx->S[11]) ^ ctx->r[1];
ctx->S[12] = a_mul(ctx->S[12]) ^ ctx->S[9] ^ ainv_mul(ctx->S[7]) ^ outfrom_fsm;
```

```
        fsmtmp = ctx->r[1] + ctx->S[9];
        ctx->r[1] = T(ctx->r[0]);
        ctx->r[0] = fsmtmp;

        outfrom_fsm = (ctx->r[0] + ctx->S[12]) ^ ctx->r[1];
        ctx->S[13] = a_mul(ctx->S[13]) ^ ctx->S[10] ^ ainv_mul(ctx->S[8]) ^ outfrom_fsm;
        fsmtmp = ctx->r[1] + ctx->S[10];
        ctx->r[1] = T(ctx->r[0]);
        ctx->r[0] = fsmtmp;

        outfrom_fsm = (ctx->r[0] + ctx->S[13]) ^ ctx->r[1];
        ctx->S[14] = a_mul(ctx->S[14]) ^ ctx->S[11] ^ ainv_mul(ctx->S[9]) ^ outfrom_fsm;
        fsmtmp = ctx->r[1] + ctx->S[11];
        ctx->r[1] = T(ctx->r[0]);
        ctx->r[0] = fsmtmp;

        outfrom_fsm = (ctx->r[0] + ctx->S[14]) ^ ctx->r[1];
        ctx->S[15] = a_mul(ctx->S[15]) ^ ctx->S[12] ^ ainv_mul(ctx->S[10]) ^ outfrom_fsm;
        fsmtmp = ctx->r[1] + ctx->S[12];
        ctx->r[1] = T(ctx->r[0]);
        ctx->r[0] = fsmtmp;
    }

    return 1;
}

int dstu8845_set_iv(Dstu8845Ctx *ctx, const uint64_t *iv)
{
    return dstu8845_init(ctx, ctx->key, ctx->key_size, iv);
}

int dstu8845_crypt(Dstu8845Ctx *ctx, const uint8_t *in, size_t inl, uint8_t *out)
{
    uint8_t i;
    uint64_t keystream[16];

    for (; inl >= 128; inl -= 128, in += 128, out += 128) {
        next_stream_full_crypt(ctx, (uint64_t *)in, (uint64_t *)out);
    }

    if (inl > 0) {
        next_stream(ctx, keystream);

        for (i = 0; i < inl; i++) {
            out[i] = in[i] ^ ((uint8_t *)keystream)[i];
        }
    }
```

```
    return 1;
}

void dstu8845_free(Dstu8845Ctx *ctx)
{
    free(ctx);
}
```

ДОДАТОК В

Файл main.c

```
#include "strumok.h"
#include <stdio.h>
#include <time.h>
double arr[] =    {8,   8192, 8192*1024, 8192*1024.0*1024};
char* arrName[] = {"B", "kB",  "MB",      "GB"};
int main()
{
   Dstu8845Ctx *ctx = dstu8845_alloc();
   uint64_t key[8], iv[4];

   iv[0] = 1;
   iv[1] = 2;
   iv[2] = 3;
   iv[3] = 4;

   uint8_t key_size = 64;

   key[7] = 0x8000000000000000;
   key[6] = 0x0000000000000000;
   key[5] = 0x0000000000000000;
   key[4] = 0x0000000000000000;
   key[3] = 0x0000000000000000;
   key[2] = 0x0000000000000000;
   key[1] = 0x0000000000000000;
   key[0] = 0x0000000000000000;

   dstu8845_init(ctx, key, key_size, iv);

   int a=1;
   for(int item=1; item < 2; item++){
      long int countOfBit = arr[item];
      double avr = 0;

      for(int countUmn = 1; countUmn < 1024; countUmn +=32)
      {
          for(int g=0;g<1;g++)
          {
              uint8_t out[countOfBit];
              memset( out, 0, countOfBit * sizeof(uint8_t) );
              clock_t begin = clock();
              for (int i = 0; i < countUmn; i++)
```

```
                {
                    dstu8845_crypt(ctx, out, countOfBit, out);
                }
                clock_t end = clock();
                double time_spent = (double)(end - begin) / CLOCKS_PER_SEC;
                avr += time_spent;
            }

        printf("%.20f s. %d%s\n", avr/1, countUmn, arrName[item]);
      }
   }
   dstu8845_free(ctx);
   return 0;
}
```

ДОДАТОК Д

Таблиці-константи констант $Mul_{\alpha}[a]$ та $Mul_{\alpha^{-1}}[a]$

Таблиця Mul_{α}: 0000000000000000, D73F04125E000004, B37E0824BC000008, 64410C36E200000C, 7BFC104865000010, ACC3145A3B000014, C882186CD9000018, 1FBD1C7E8700001C, F6E52090CA000020, 21DA248294000024, 459B28B476000028, 92A42CA62800002C, 8D1930D8AF000030, 5A2634CAF1000034, 3E6738FC13000038, E9583CEE4D00003C, F1D7403D89000040, 26E8442FD7000044, 42A9481935000048, 95964C0B6B00004C, 8A2B5075EC000050, 5D145467B2000054, 3955585150000058, EE6A5C430E00005C, 073260AD43000060, D00D64BF1D000064, B44C6889FF000068, 63736C9BA100006C, 7CCE70E526000070, ABF174F778000074, CFB078C19A000078, 188F7CD3C400007C, FFB3807A0F000080, 288C846851000084, 4CCD885EB3000088, 9BF28C4CED00008C, 844F90326A000090, 5370942034000094, 37319816D6000098, E00E9C048800009C, 0956A0EAC50000A0, DE69A4F89B0000A4, BA28A8CE790000A8, 6D17ACDC270000AC, 72AAB0A2A00000B0, A595B4B0FE0000B4, C1D4B8861C0000B8, 16EBBC94420000BC, 0E64C047860000C0, D95BC455D80000C4, BD1AC8633A0000C8, 6A25CC71640000CC, 7598D00FE30000D0, A2A7D41DBD0000D4, C6E6D82B5F0000D8, 11D9DC39010000DC, F881E0D74C0000E0, 2FBEE4C5120000E4, 4BFFE8F3F00000E8, 9CC0ECE1AE0000EC, 837DF09F290000F0, 5442F48D770000F4, 3003F8BB950000F8, E73CFCA9CB0000FC, E37B1DF41E00001D, 344419E640000019, 500515D0A2000015, 873A11C2FC000011, 98870DBC7B00000D, 4FB809AE25000009, 2BF90598C7000005, FCC6018A99000001, 159E3D64D400003D, C2A139768A000039, A6E0354068000035, 71DF315236000031, 6E622D2CB100002D, B95D293EEF000029, DD1C25080D000025, 0A23211A53000021, 12AC5DC99700005D, C59359DBC9000059, A1D255ED2B000055, 76ED51FF75000051, 69504D81F200004D, BE6F4993AC000049, DA2E45A54E000045, 0D1141B710000041, E4497D595D00007D, 3376794B03000079, 5737757DE1000075, 8008716FBF000071, 9FB56D113800006D, 488A690366000069, 2CCB653584000065, FBF46127DA000061, 1CC89D8E1100009D, CBF7999C4F000099, AFB695AAAD000095, 788991B8F3000091, 67348DC67400008D, B00B89D42A000089, D44A85E2C8000085, 037581F096000081, EA2DBD1EDB0000BD, 3D12B90C850000B9, 5953B53A670000B5, 8E6CB128390000B1, 91D1AD56BE0000AD, 46EEA944E00000A9, 22AFA572020000A5, F590A1605C0000A1, ED1FDDB3980000DD, 3A20D9A1C60000D9, 5E61D597240000D5, 895ED1857A0000D1, 96E3CDFBFD0000CD, 41DCC9E9A30000C9, 259DC5DF410000C5, F2A2C1CD1F0000C1, 1BFAFD23520000FD, CCC5F9310C0000F9, A884F507EE0000F5, 7FBBF115B00000F1, 6006ED6B370000ED, B739E979690000E9, D378E54F8B0000E5, 0447E15DD50000E1, DBF63AF53C00003A, 0CC93EE76200003E, 688832D180000032, BFB736C3DE000036, A00A2ABD5900002A, 77352EAF0700002E, 13742299E5000022, C44B268BBB000026, 2D131A65F600001A, FA2C1E77A800001E, 9E6D12414A000012, 4952165314000016, 56EF0A2D9300000A, 81D00E3FCD00000E, E59102092F000002, 32AE061B71000006, 2A217AC8B500007A, FD1E7EDAEB00007E, 995F72EC09000072, 4E6076FE57000076, 51DD6A80D000006A, 86E26E928E00006E, E2A362A46C000062, 359C66B632000066, DCC45A587F00005A, 0BFB5E4A2100005E, 6FBA527CC3000052, B885566E9D000056, A7384A101A00004A, 70074E024400004E, 14464234A6000042, C3794626F8000046, 2445BA8F330000BA, F37ABE9D6D0000BE, 973BB2AB8F0000B2, 4004B6B9D10000B6, 5FB9AAC7560000AA, 8886AED5080000AE, ECC7A2E3EA0000A2, 3BF8A6F1B40000A6, D2A09A1FF900009A, 059F9E0DA700009E, 61DE923B45000092, B6E196291B000096, A95C8A579C00008A, 7E638E45C200008E, 1A22827320000082, CD1D86617E000086, D592FAB2BA0000FA, 02ADFEA0E40000FE, 66ECF296060000F2, B1D3F684580000F6, AE6EEAFADF0000EA, 7951EEE8810000EE, 1D10E2DE630000E2, CA2FE6CC3D0000E6, 2377DA22700000DA, F448DE302E0000DE, 9009D206CC0000D2, 4736D614920000D6, 588BCA6A150000CA, 8FB4CE784B0000CE, EBF5C24EA90000C2, 3CCAC65CF70000C6, 388D270122000027, EFB223137C000023, 8BF32F259E00002F, 5CCC2B37C000002B, 4371374947000037, 944E335B19000033, F00F3F6DFB00003F, 27303B7FA500003B, CE680791E8000007, 19570383B6000003, 7D160FB55400000F, AA290BA70A00000B, B59417D98D000017, 62AB13CBD3000013, 06EA1FFD3100001F, D1D51BEF6F00001B, C95A673CAB000067, 1E65632EF5000063, 7A246F181700006F, AD1B6B0A4900006B, B2A67774CE000077, 6599736690000073, 01D87F507200007F, D6E77B422C00007B, 3FBF47AC61000047, E88043BE3F000043, 8CC14F88DD00004F, 5BFE4B9A8300004B, 444357E404000057, 937C53F65A000053, F73D5FC0B800005F, 20025BD2E600005B, C73EA77B2D0000A7, 1001A369730000A3, 7440AF5F910000AF, A37FAB4DCF0000AB, BCC2B733480000B7, 6BFDB321160000B3, 0FBCBF17F40000BF, D883BB05AA0000BB, 31DB87EBE7000087, E6E483F9B9000083, 82A58FCF5B00008F, 559A8BDD0500008B, 4A2797A382000097,

9D1893B1DC000093, F9599F873E00009F, 2E669B956000009B, 36E9E746A40000E7, E1D6E354FA0000E3, 8597EF62180000EF, 52A8EB70460000EB, 4D15F70EC10000F7, 9A2AF31C9F0000F3, FE6BFF2A7D0000FF, 2954FB38230000FB, C00CC7D66E0000C7, 1733C3C4300000C3, 7372CFF2D20000CF, A44DCBE08C0000CB, BBF0D79E0B0000D7, 6CCFD38C550000D3, 088EDFBAB70000DF, DFB1DBA8E90000DB.

Таблиця $\boldsymbol{Mul}_{\alpha^{-1}}$:0000000000000000, 47FCC6018A990000, 8EE59102092F0000, C919570383B60000, 01D73F04125E0000, 462BF90598C70000, 8F32AE061B710000, C8CE680791E80000, 02B37E0824BC0000, 454FB809AE250000, 8C56EF0A2D930000, CBAA290BA70A0000, 0364410C36E20000, 4498870DBC7B0000, 8D81D00E3FCD0000, CA7D160FB5540000, 047BFC1048650000, 43873A11C2FC0000, 8A9E6D12414A0000, CD62AB13CBD30000, 05ACC3145A3B0000, 42500515D0A20000, 8B49521653140000, CCB59417D98D0000, 06C882186CD90000, 41344419E6400000, 882D131A65F60000, CFD1D51BEF6F0000, 071FBD1C7E870000, 40E37B1DF41E0000, 89FA2C1E77A80000, CE06EA1FFD310000, 08F6E52090CA0000, 4F0A23211A530000, 8613742299E50000, C1EFB223137C0000, 0921DA2482940000, 4EDD1C25080D0000, 87C44B268BBB0000, C0388D2701220000, 0A459B28B4760000, 4DB95D293EEF0000, 84A00A2ABD590000, C35CCC2B37C00000, 0B92A42CA6280000, 4C6E622D2CB10000, 8577352EAF070000, C28BF32F259E0000, 0C8D1930D8AF0000, 4B71DF3152360000, 82688832D1800000, C5944E335B190000, 0D5A2634CAF10000, 4AA6E03540680000, 83BFB736C3DE0000, C443713749470000, 0E3E6738FC130000, 49C2A139768A0000, 80DBF63AF53C0000, C727303B7FA50000, 0FE9583CEE4D0000, 48159E3D64D40000, 810CC93EE7620000, C6F00F3F6DFB0000, 10F1D7403D890000, 570D1141B7100000, 9E14464234A60000, D9E88043BE3F0000, 1126E8442FD70000, 56DA2E45A54E0000, 9FC3794626F80000, D83FBF47AC610000, 1242A94819350000, 55BE6F4993AC0000, 9CA7384A101A0000, DB5BFE4B9A830000, 1395964C0B6B0000, 5469504D81F20000, 9D70074E02440000, DA8CC14F88DD0000, 148A2B5075EC0000, 5376ED51FF750000, 9A6FBA527CC30000, DD937C53F65A0000, 155D145467B20000, 52A1D255ED2B0000, 9BB885566E9D0000, DC444357E4040000, 1639555851500000, 51C59359DBC90000, 98DCC45A587F0000, DF20025BD2E60000, 17EE6A5C430E0000, 5012AC5DC9970000, 990BFB5E4A210000, DEF73D5FC0B80000, 18073260AD430000, 5FFBF46127DA0000, 96E2A362A46C0000, D11E65632EF50000, 19D00D64BF1D0000, 5E2CCB6535840000, 97359C66B6320000, D0C95A673CAB0000, 1AB44C6889FF0000, 5D488A6903660000, 9451DD6A80D00000, D3AD1B6B0A490000, 1B63736C9BA10000, 5C9FB56D11380000, 9586E26E928E0000, D27A246F18170000, 1C7CCE70E5260000, 5B8008716FBF0000, 92995F72EC090000, D565997366900000, 1DABF174F7780000, 5A5737757DE10000, 934E6076FE570000, D4B2A67774CE0000, 1ECFB078C19A0000, 593376794B030000, 902A217AC8B50000, D7D6E77B422C0000, 1F188F7CD3C40000, 58E4497D595D0000, 91FD1E7EDAEB0000, D601D87F50720000, 20FFB3807A0F0000, 67037581F0960000, AE1A228273200000, E9E6E483F9B90000, 21288C8468510000, 66D44A85E2C80000, AFCD1D86617E0000, E831DB87EBE70000, 224CCD885EB30000, 65B00B89D42A0000, ACA95C8A579C0000, EB559A8BDD050000, 239BF28C4CED0000, 6467348DC6740000, AD7E638E45C20000, EA82A58FCF5B0000, 24844F90326A0000, 63788991B8F30000, AA61DE923B450000, ED9D1893B1DC0000, 2553709420340000, 62AFB695AAAD0000, ABB6E196291B0000, EC4A2797A3820000, 2637319816D60000, 61CBF7999C4F0000, A8D2A09A1FF90000, EF2E669B95600000, 27E00E9C04880000, 601CC89D8E110000, A9059F9E0DA70000, EEF9599F873E0000, 280956A0EAC50000, 6FF590A1605C0000, A6ECC7A2E3EA0000, E11001A369730000, 29DE69A4F89B0000, 6E22AFA572020000, A73BF8A6F1B40000, E0C73EA77B2D0000, 2ABA28A8CE790000, 6D46EEA944E00000, A45FB9AAC7560000, E3A37FAB4DCF0000, 2B6D17ACDC270000, 6C91D1AD56BE0000, A58886AED5080000, E27440AF5F910000, 2C72AAB0A2A00000, 6B8E6CB128390000, A2973BB2AB8F0000, E56BFDB321160000, 2DA595B4B0FE0000, 6A5953B53A670000, A34004B6B9D10000, E4BCC2B733480000, 2EC1D4B8861C0000, 693D12B90C850000, A02445BA8F330000, E7D883BB05AA0000, 2F16EBBC94420000, 68EA2DBD1EDB0000, A1F37ABE9D6D0000, E60FBCBF17F40000, 300E64C047860000, 77F2A2C1CD1F0000, BEEBF5C24EA90000, F91733C3C4300000, 31D95BC455D80000, 76259DC5DF410000, BF3CCAC65CF70000, F8C00CC7D66E0000, 32BD1AC8633A0000, 7541DCC9E9A30000, BC588BCA6A150000, FBA44DCBE08C0000, 336A25CC71640000, 7496E3CDFBFD0000, BD8FB4CE784B0000, FA7372CFF2D20000, 347598D00FE30000, 73895ED1857A0000, BA9009D206CC0000, FD6CCFD38C550000, 35A2A7D41DBD0000, 725E61D597240000, BB4736D614920000, FCBBF0D79E0B0000, 36C6E6D82B5F0000, 713A20D9A1C60000, B82377DA22700000, FFDFB1DBA8E90000, 3711D9DC39010000, 70ED1FDDB3980000, B9F448DE302E0000, FE088EDFBAB70000, 38F881E0D74C0000, 7F0447E15DD50000, B61D10E2DE630000, F1E1D6E354FA0000, 392FBEE4C5120000, 7ED378E54F8B0000, B7CA2FE6CC3D0000, F036E9E746A40000, 3A4BFFE8F3F00000, 7DB739E979690000, B4AE6EEAFADF0000, F352A8EB70460000, 3B9CC0ECE1AE0000, 7C6006ED6B370000, B57951EEE8810000, F28597EF62180000, 3C837DF09F290000, 7B7FBBF115B00000, B266ECF296060000, F59A2AF31C9F0000, 3D5442F48D770000, 7AA884F507EE0000, B3B1D3F684580000, F44D15F70EC10000, 3E3003F8BB950000, 79CCC5F9310C0000, B0D592FAB2BA0000, F72954FB38230000, 3FE73CFCA9CB0000, 781BFAFD23520000, B102ADFEA0E40000, F6FE6BFF2A7D0000.

ДОДАТОК Е

Таблиці-константи констант $Mul_{\alpha}[a]$ та $Mul_{\alpha^{-1}}[a]$

Таблиця Mul_{α}: 0000000000000000, D73F04125E000004, B37E0824BC000008, 64410C36E200000C, 7BFC104865000010, ACC3145A3B000014, C882186CD9000018, 1FBD1C7E8700001C, F6E52090CA000020, 21DA248294000024, 459B28B476000028, 92A42CA62800002C, 8D1930D8AF000030, 5A2634CAF1000034, 3E6738FC13000038, E9583CEE4D00003C, F1D7403D89000040, 26E8442FD7000044, 42A9481935000048, 95964C0B6B00004C, 8A2B5075EC000050, 5D145467B2000054, 3955585150000058, EE6A5C430E00005C, 073260AD43000060, D00D64BF1D000064, B44C6889FF000068, 63736C9BA100006C, 7CCE70E526000070, ABF174F778000074, CFB078C19A000078, 188F7CD3C400007C, FFB3807A0F000080, 288C846851000084, 4CCD885EB3000088, 9BF28C4CED00008C, 844F90326A000090, 5370942034000094, 37319816D6000098, E00E9C048800009C, 0956A0EAC50000A0, DE69A4F89B0000A4, BA28A8CE790000A8, 6D17ACDC270000AC, 72AAB0A2A00000B0, A595B4B0FE0000B4, C1D4B8861C0000B8, 16EBBC94420000BC, 0E64C047860000C0, D95BC455D80000C4, BD1AC8633A0000C8, 6A25CC71640000CC, 7598D00FE30000D0, A2A7D41DBD0000D4, C6E6D82B5F0000D8, 11D9DC39010000DC, F881E0D74C0000E0, 2FBEE4C5120000E4, 4BFFE8F3F00000E8, 9CC0ECE1AE0000EC, 837DF09F290000F0, 5442F48D770000F4, 3003F8BB950000F8, E73CFCA9CB0000FC, E37B1DF41E00001D, 344419E640000019, 500515D0A2000015, 873A11C2FC000011, 98870DBC7B00000D, 4FB809AE25000009, 2BF90598C7000005, FCC6018A99000001, 159E3D64D400003D, C2A139768A000039, A6E0354068000035, 71DF315236000031, 6E622D2CB100002D, B95D293EEF000029, DD1C25080D000025, 0A23211A53000021, 12AC5DC99700005D, C59359DBC9000059, A1D255ED2B000055, 76ED51FF75000051, 69504D81F200004D, BE6F4993AC000049, DA2E45A54E000045, 0D1141B710000041, E4497D595D00007D, 3376794B03000079, 5737757DE1000075, 8008716FBF000071, 9FB56D113800006D, 488A690366000069, 2CCB653584000065, FBF46127DA000061, 1CC89D8E1100009D, CBF7999C4F000099, AFB695AAAD000095, 788991B8F3000091, 67348DC67400008D, B00B89D42A000089, D44A85E2C8000085, 037581F096000081, EA2DBD1EDB0000BD, 3D12B90C850000B9, 5953B53A670000B5, 8E6CB128390000B1, 91D1AD56BE0000AD, 46EEA944E00000A9, 22AFA572020000A5, F590A1605C0000A1, ED1FDDB3980000DD, 3A20D9A1C60000D9, 5E61D597240000D5, 895ED1857A0000D1, 96E3CDFBFD0000CD, 41DCC9E9A30000C9, 259DC5DF410000C5, F2A2C1CD1F0000C1, 1BFAFD23520000FD, CCC5F9310C0000F9, A884F507EE0000F5, 7FBBF115B00000F1, 6006ED6B370000ED, B739E979690000E9, D378E54F8B0000E5, 0447E15DD50000E1, DBF63AF53C00003A, 0CC93EE76200003E, 688832D180000032, BFB736C3DE000036, A00A2ABD5900002A, 77352EAF0700002E, 13742299E5000022, C44B268BBB000026, 2D131A65F600001A, FA2C1E77A800001E, 9E6D12414A000012, 4952165314000016, 56EF0A2D9300000A, 81D00E3FCD00000E, E59102092F000002, 32AE061B71000006, 2A217AC8B500007A, FD1E7EDAEB00007E, 995F72EC09000072, 4E6076FE57000076, 51DD6A80D000006A, 86E26E928E00006E, E2A362A46C000062, 359C66B632000066, DCC45A587F00005A, 0BFB5E4A2100005E, 6FBA527CC3000052, B885566E9D000056, A7384A101A00004A, 70074E024400004E, 14464234A6000042, C3794626F8000046, 2445BA8F330000BA, F37ABE9D6D0000BE, 973BB2AB8F0000B2, 4004B6B9D10000B6, 5FB9AAC7560000AA, 8886AED5080000AE, ECC7A2E3EA0000A2, 3BF8A6F1B40000A6, D2A09A1FF900009A, 059F9E0DA700009E, 61DE923B45000092, B6E196291B000096, A95C8A579C00008A, 7E638E45C200008E, 1A22827320000082, CD1D86617E000086, D592FAB2BA0000FA, 02ADFEA0E40000FE, 66ECF296060000F2, B1D3F684580000F6, AE6EEAFADF0000EA, 7951EEE8810000EE, 1D10E2DE630000E2, CA2FE6CC3D0000E6, 2377DA22700000DA, F448DE302E0000DE, 9009D206CC0000D2, 4736D614920000D6, 588BCA6A150000CA, 8FB4CE784B0000CE, EBF5C24EA90000C2, 3CCAC65CF70000C6, 388D270122000027, EFB223137C000023, 8BF32F259E00002F, 5CCC2B37C000002B, 4371374947000037, 944E335B19000033, F00F3F6DFB00003F, 27303B7FA500003B, CE680791E8000007, 19570383B6000003, 7D160FB55400000F, AA290BA70A00000B, B59417D98D000017, 62AB13CBD3000013, 06EA1FFD3100001F, D1D51BEF6F00001B, C95A673CAB000067, 1E65632EF5000063, 7A246F181700006F, AD1B6B0A4900006B, B2A67774CE000077, 6599736690000073, 01D87F507200007F, D6E77B422C00007B, 3FBF47AC61000047, E88043BE3F000043, 8CC14F88DD00004F, 5BFE4B9A8300004B, 444357E404000057, 937C53F65A000053, F73D5FC0B800005F, 20025BD2E600005B, C73EA77B2D0000A7, 1001A369730000A3, 7440AF5F910000AF, A37FAB4DCF0000AB, BCC2B733480000B7, 6BFDB321160000B3, 0FBCBF17F40000BF, D883BB05AA0000BB, 31DB87EBE7000087, E6E483F9B9000083, 82A58FCF5B00008F, 559A8BDD0500008B, 4A2797A382000097,

9D1893B1DC000093, F9599F873E00009F, 2E669B956000009B, 36E9E746A40000E7, E1D6E354FA0000E3, 8597EF62180000EF, 52A8EB70460000EB, 4D15F70EC10000F7, 9A2AF31C9F0000F3, FE6BFF2A7D0000FF, 2954FB38230000FB, C00CC7D66E0000C7, 1733C3C4300000C3, 7372CFF2D20000CF, A44DCBE08C0000CB, BBF0D79E0B0000D7, 6CCFD38C550000D3, 088EDFBAB70000DF, DFB1DBA8E90000DB.

Таблиця $\boldsymbol{Mul}_{\alpha^{-1}}$:0000000000000000, 47FCC6018A990000, 8EE59102092F0000, C919570383B60000, 01D73F04125E0000, 462BF90598C70000, 8F32AE061B710000, C8CE680791E80000, 02B37E0824BC0000, 454FB809AE250000, 8C56EF0A2D930000, CBAA290BA70A0000, 0364410C36E20000, 4498870DBC7B0000, 8D81D00E3FCD0000, CA7D160FB5540000, 047BFC1048650000, 43873A11C2FC0000, 8A9E6D12414A0000, CD62AB13CBD30000, 05ACC3145A3B0000, 42500515D0A20000, 8B49521653140000, CCB59417D98D0000, 06C882186CD90000, 41344419E6400000, 882D131A65F60000, CFD1D51BEF6F0000, 071FBD1C7E870000, 40E37B1DF41E0000, 89FA2C1E77A80000, CE06EA1FFD310000, 08F6E52090CA0000, 4F0A23211A530000, 8613742299E50000, C1EFB223137C0000, 0921DA2482940000, 4EDD1C25080D0000, 87C44B268BBB0000, C0388D2701220000, 0A459B28B4760000, 4DB95D293EEF0000, 84A00A2ABD590000, C35CCC2B37C00000, 0B92A42CA6280000, 4C6E622D2CB10000, 8577352EAF070000, C28BF32F259E0000, 0C8D1930D8AF0000, 4B71DF3152360000, 82688832D1800000, C5944E335B190000, 0D5A2634CAF10000, 4AA6E03540680000, 83BFB736C3DE0000, C443713749470000, 0E3E6738FC130000, 49C2A139768A0000, 80DBF63AF53C0000, C727303B7FA50000, 0FE9583CEE4D0000, 48159E3D64D40000, 810CC93EE7620000, C6F00F3F6DFB0000, 10F1D7403D890000, 570D1141B7100000, 9E14464234A60000, D9E88043BE3F0000, 1126E8442FD70000, 56DA2E45A54E0000, 9FC3794626F80000, D83FBF47AC610000, 1242A94819350000, 55BE6F4993AC0000, 9CA7384A101A0000, DB5BFE4B9A830000, 1395964C0B6B0000, 5469504D81F20000, 9D70074E02440000, DA8CC14F88DD0000, 148A2B5075EC0000, 5376ED51FF750000, 9A6FBA527CC30000, DD937C53F65A0000, 155D145467B20000, 52A1D255ED2B0000, 9BB885566E9D0000, DC444357E4040000, 1639555851500000, 51C59359DBC90000, 98DCC45A587F0000, DF20025BD2E60000, 17EE6A5C430E0000, 5012AC5DC9970000, 990BFB5E4A210000, DEF73D5FC0B80000, 18073260AD430000, 5FFBF46127DA0000, 96E2A362A46C0000, D11E65632EF50000, 19D00D64BF1D0000, 5E2CCB6535840000, 97359C66B6320000, D0C95A673CAB0000, 1AB44C6889FF0000, 5D488A6903660000, 9451DD6A80D00000, D3AD1B6B0A490000, 1B63736C9BA10000, 5C9FB56D11380000, 9586E26E928E0000, D27A246F18170000, 1C7CCE70E5260000, 5B8008716FBF0000, 92995F72EC090000, D565997366900000, 1DABF174F7780000, 5A5737757DE10000, 934E6076FE570000, D4B2A67774CE0000, 1ECFB078C19A0000, 593376794B030000, 902A217AC8B50000, D7D6E77B422C0000, 1F188F7CD3C40000, 58E4497D595D0000, 91FD1E7EDAEB0000, D601D87F50720000, 20FFB3807A0F0000, 67037581F0960000, AE1A228273200000, E9E6E483F9B90000, 21288C8468510000, 66D44A85E2C80000, AFCD1D86617E0000, E831DB87EBE70000, 224CCD885EB30000, 65B00B89D42A0000, ACA95C8A579C0000, EB559A8BDD050000, 239BF28C4CED0000, 6467348DC6740000, AD7E638E45C20000, EA82A58FCF5B0000, 24844F90326A0000, 63788991B8F30000, AA61DE923B450000, ED9D1893B1DC0000, 2553709420340000, 62AFB695AAAD0000, ABB6E196291B0000, EC4A2797A3820000, 2637319816D60000, 61CBF7999C4F0000, A8D2A09A1FF90000, EF2E669B95600000, 27E00E9C04880000, 601CC89D8E110000, A9059F9E0DA70000, EEF9599F873E0000, 280956A0EAC50000, 6FF590A1605C0000, A6ECC7A2E3EA0000, E11001A369730000, 29DE69A4F89B0000, 6E22AFA572020000, A73BF8A6F1B40000, E0C73EA77B2D0000, 2ABA28A8CE790000, 6D46EEA944E00000, A45FB9AAC7560000, E3A37FAB4DCF0000, 2B6D17ACDC270000, 6C91D1AD56BE0000, A58886AED5080000, E27440AF5F910000, 2C72AAB0A2A00000, 6B8E6CB128390000, A2973BB2AB8F0000, E56BFDB321160000, 2DA595B4B0FE0000, 6A5953B53A670000, A34004B6B9D10000, E4BCC2B733480000, 2EC1D4B8861C0000, 693D12B90C850000, A02445BA8F330000, E7D883BB05AA0000, 2F16EBBC94420000, 68EA2DBD1EDB0000, A1F37ABE9D6D0000, E60FBCBF17F40000, 300E64C047860000, 77F2A2C1CD1F0000, BEEBF5C24EA90000, F91733C3C4300000, 31D95BC455D80000, 76259DC5DF410000, BF3CCAC65CF70000, F8C00CC7D66E0000, 32BD1AC8633A0000, 7541DCC9E9A30000, BC588BCA6A150000, FBA44DCBE08C0000, 336A25CC71640000, 7496E3CDFBFD0000, BD8FB4CE784B0000, FA7372CFF2D20000, 347598D00FE30000, 73895ED1857A0000, BA9009D206CC0000, FD6CCFD38C550000, 35A2A7D41DBD0000, 725E61D597240000, BB4736D614920000, FCBBF0D79E0B0000, 36C6E6D82B5F0000, 713A20D9A1C60000, B82377DA22700000, FFDFB1DBA8E90000, 3711D9DC39010000, 70ED1FDDB3980000, B9F448DE302E0000, FE088EDFBAB70000, 38F881E0D74C0000, 7F0447E15DD50000, B61D10E2DE630000, F1E1D6E354FA0000, 392FBEE4C5120000, 7ED378E54F8B0000, B7CA2FE6CC3D0000, F036E9E746A40000, 3A4BFFE8F3F00000, 7DB739E979690000, B4AE6EEAFADF0000, F352A8EB70460000, 3B9CC0ECE1AE0000, 7C6006ED6B370000, B57951EEE8810000, F28597EF62180000, 3C837DF09F290000, 7B7FBBF115B00000, B266ECF296060000, F59A2AF31C9F0000, 3D5442F48D770000, 7AA884F507EE0000, B3B1D3F684580000, F44D15F70EC10000, 3E3003F8BB950000, 79CCC5F9310C0000, B0D592FAB2BA0000, F72954FB38230000, 3FE73CFCA9CB0000, 781BFAFD23520000, B102ADFEA0E40000, F6FE6BFF2A7D0000.

ДОДАТОК Ж

Нелінійні таблиці підстановок π_0, π_1, π_2 та π_3

Підстановка π_0:

A8 43 5F 06 6B 75 6C 59 71 DF 87 95 17 F0 D8 09

6D F3 1D CB C9 4D 2C AF 79 E0 97 FD 6F 4B 45 39

3E DD A3 4F B4 B6 9A 0E 1F BF 15 E1 49 D2 93 C6

92 72 9E 61 D1 63 FA EE F4 19 D5 AD 58 A4 BB A1

DC F2 83 37 42 E4 7A 32 9C CC AB 4A 8F 6E 04 27

2E E7 E2 5A 96 16 23 2B C2 65 66 0F BC A9 47 41

34 48 FC B7 6A 88 A5 53 86 F9 5B DB 38 7B C3 1E

22 33 24 28 36 C7 B2 3B 8E 77 BA F5 14 9F 08 55

9B 4C FE 60 5C DA 18 46 CD 7D 21 B0 3F 1B 89 FF

EB 84 69 3A 9D D7 D3 70 67 40 B5 DE 5D 30 91 B1

78 11 01 E5 00 68 98 A0 C5 02 A6 74 2D 0B A2 76

B3 BE CE BD AE E9 8A 31 1C EC F1 99 94 AA F6 26

2F EF E8 8C 35 03 D4 7F FB 05 C1 5E 90 20 3D 82

F7 EA 0A 0D 7E F8 50 1A C4 07 57 B8 3C 62 E3 C8

AC 52 64 10 D0 D9 13 0C 12 29 51 B9 CF D6 73 8D

81 54 C0 ED 4E 44 A7 2A 85 25 E6 CA 7C 8B 56 80.

Підстановка π_1:

CE BB EB 92 EA CB 13 C1 E9 3A D6 B2 D2 90 17 F8

42 15 56 B4 65 1C 88 43 C5 5C 36 BA F5 57 67 8D

31 F6 64 58 9E F4 22 AA 75 0F 02 B1 DF 6D 73 4D

7C 26 2E F7 08 5D 44 3E 9F 14 C8 AE 54 10 D8 BC

1A 6B 69 F3 BD 33 AB FA D1 9B 68 4E 16 95 91 EE

4C 63 8E 5B CC 3C 19 A1 81 49 7B D9 6F 37 60 CA

E7 2B 48 FD 96 45 FC 41 12 0D 79 E5 89 8C E3 20

30 DC B7 6C 4A B5 3F 97 D4 62 2D 06 A4 A5 83 5F

2A DA C9 00 7E A2 55 BF 11 D5 9C CF 0E 0A 3D 51

7D 93 1B FE C4 47 09 86 0B 8F 9D 6A 07 B9 B0 98

18 32 71 4B EF 3B 70 A0 E4 40 FF C3 A9 E6 78 F9

8B 46 80 1E 38 E1 B8 A8 E0 0C 23 76 1D 25 24 05

F1 6E 94 28 9A 84 E8 A3 4F 77 D3 85 E2 52 F2 82

50 7A 2F 74 53 B3 61 AF 39 35 DE CD 1F 99 AC AD

72 2C DD D0 87 BE 5E A6 EC 04 C6 03 34 FB DB 59

B6 C2 01 F0 5A ED A7 66 21 7F 8A 27 C7 C0 29 D7.

Підстановка $\boldsymbol{\pi_2}$:

93 D9 9A B5 98 22 45 FC BA 6A DF 02 9F DC 51 59

4A 17 2B C2 94 F4 BB A3 62 E4 71 D4 CD 70 16 E1

49 3C C0 D8 5C 9B AD 85 53 A1 7A C8 2D E0 D1 72

A6 2C C4 E3 76 78 B7 B4 09 3B 0E 41 4C DE B2 90

25 A5 D7 03 11 00 C3 2E 92 EF 4E 12 9D 7D CB 35

10 D5 4F 9E 4D A9 55 C6 D0 7B 18 97 D3 36 E6 48

56 81 8F 77 CC 9C B9 E2 AC B8 2F 15 A4 7C DA 38

1E 0B 05 D6 14 6E 6C 7E 66 FD B1 E5 60 AF 5E 33

87 C9 F0 5D 6D 3F 88 8D C7 F7 1D E9 EC ED 80 29

27 CF 99 A8 50 0F 37 24 28 30 95 D2 3E 5B 40 83

B3 69 57 1F 07 1C 8A BC 20 EB CE 8E AB EE 31 A2

73 F9 CA 3A 1A FB 0D C1 FE FA F2 6F BD 96 DD 43

52 B6 08 F3 AE BE 19 89 32 26 B0 EA 4B 64 84 82

6B F5 79 BF 01 5F 75 63 1B 23 3D 68 2A 65 E8 91

F6 FF 13 58 F1 47 0A 7F C5 A7 E7 61 5A 06 46 44

42 04 A0 DB 39 86 54 AA 8C 34 21 8B F8 0C 74 67.

Підстановка $\boldsymbol{\pi_3}$:

68 8D CA 4D 73 4B 4E 2A D4 52 26 B3 54 1E 19 1F

22 03 46 3D 2D 4A 53 83 13 8A B7 D5 25 79 F5 BD

58 2F 0D 02 ED 51 9E 11 F2 3E 55 5E D1 16 3C 66

70 5D F3 45 40 CC E8 94 56 08 CE 1A 3A D2 E1 DF

B5 38 6E 0E E5 F4 F9 86 E9 4F D6 85 23 CF 32 99

31 14 AE EE C8 48 D3 30 A1 92 41 B1 18 C4 2C 71

72 44 15 FD 37 BE 5F AA 9B 88 D8 AB 89 9C FA 60

EA BC 62 0C 24 A6 A8 EC 67 20 DB 7C 28 DD AC 5B

34 7E 10 F1 7B 8F 63 A0 05 9A 43 77 21 BF 27 09

C3 9F B6 D7 29 C2 EB C0 A4 8B 8C 1D FB FF C1 B2

97 2E F8 65 F6 75 07 04 49 33 E4 D9 B9 D0 42 C7

6C 90 00 8E 6F 50 01 C5 DA 47 3F CD 69 A2 E2 7A

A7 C6 93 0F 0A 06 E6 2B 96 A3 1C AF 6A 12 84 39

E7 B0 82 F7 FE 9D 87 5C 81 35 DE B4 A5 FC 80 EF

CB BB 6B 76 BA 5A 7D 78 0B 95 E3 AD 74 98 3B 36

64 6D DC F0 59 A9 4C 17 7F 91 B8 C9 57 1B E0 61.

ДОДАТОК З

Файл server.c

```
#ifdef HAVE_CONFIG_H
#include <config.h>
#endif
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <stdlib.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/wait.h>
#include <netdb.h>
#include <errno.h>
#include <pulse/simple.h>
#include <pulse/error.h>
#include <strumok/strumoks.h>

#define BUFSIZE 1024
#define PORT_NUM "5000"
#define BACKLOG 50 // maximum length of queue of pending connections may grow
#define ADDR_STR_LEN (NI_MAXHOST+NI_MAXSERV+10)
#define MAX_DATA_SIZE 50



int cfd;
void encryption(uint8_t*data){
    Dstu8845Ctx *ctx = dstu8845_alloc();
   uint64_t key[8], iv[4];

   iv[0] = 1;
   iv[1] = 2;
   iv[2] = 3;
```

```
    iv[3] = 4;

    uint8_t key_size = 64;

    key[7] = 0x8000000000000000;
    key[6] = 0x0000000000000000;
    key[5] = 0x0000000000000000;
    key[4] = 0x0000000000000000;
    key[3] = 0x0000000000000000;
    key[2] = 0x0000000000000000;
    key[1] = 0x0000000000000000;
    key[0] = 0x0000000000000000;
    dstu8845_init(ctx, key, key_size, iv);
    dstu8845_crypt(ctx, data, countOfBit, data);
}
void my_signal_handler(int sig_num) {
    char answer[2];
    if(sig_num==SIGINT) {
        printf("\nReceived SIGINT i.e ctrl+c\n");
        printf("Do you want to terminate the process (Y/N): ");
        scanf("%s",answer);
        if(strncmp(answer,"Y",1)==0) {
            close(cfd);
            exit(0);
        }
    }
}
void grim_reap(int sig) {
    while(waitpid(-1,NULL,WNOHANG)>0) {
        continue;
    }
}
/* A simple routine calling UNIX write() in a loop */
static ssize_t loop_write(int fd, void*data, size_t size) {
    ssize_t ret = 0;
    while (size > 0) {
        ssize_t r;
```

```
        if ((r = write(fd, data, size)) < 0)
            return r;
        if (r == 0)
            break;
        ret += r;
        data = (const uint8_t*) data + r;
        size -= (size_t) r;
    }
    return ret;
}
int main(int argc, char *argv[]) {
    struct addrinfo hints;
    struct addrinfo *result,*rp;
    struct sockaddr_storage claddr;
    socklen_t addrlen;
    int lfd,optval;
    char host[NI_MAXHOST],service[NI_MAXSERV];
    char addr_str[ADDR_STR_LEN];
    ssize_t num_bytes;
    char message[MAX_DATA_SIZE];
    char buffer[MAX_DATA_SIZE];
    /* The Sample format to use */
    static const pa_sample_spec ss = {
        .format = PA_SAMPLE_S16LE,
        .rate = 44100,
        .channels = 2
    };
    pa_simple *s = NULL;
    int ret = 1;
    int error;
    struct sigaction sa;
    sigemptyset(&sa.sa_mask);
    sa.sa_flags=SA_RESTART;
    sa.sa_handler=grim_reap;
    if(sigaction(SIGCHLD,&sa,NULL)==-1) {
        printf("Error in sigaction when trying to change action done on SIGCHLD\n");
        exit(0);
    }
```

```
// installing signal handler to ignore SIGPIPE signal
if(signal(SIGPIPE,SIG_IGN)==SIG_ERR) {
   printf("Error in ignoring the SIGPIPE signal\n");
   return 0;
}
// installing signal handler to handle SIGINT
if(signal(SIGINT,my_signal_handler)==SIG_ERR) {
   printf("Error in handling the SIGINT signal\n");
   return 0;
}
// to fill the sizeof(struct addrinfo) bytes of memory area pointed by &hints by zero
memset(&hints,0,sizeof(struct addrinfo));
// intialize hints structure of type addrinfo
hints.ai_canonname=NULL;
hints.ai_next=NULL;
hints.ai_addr=NULL;
hints.ai_family=AF_UNSPEC;
hints.ai_socktype=SOCK_STREAM;
hints.ai_flags=AI_PASSIVE;
if(getaddrinfo(NULL,PORT_NUM,&hints,&result)!=0) {
   printf("Error in getaddrinfo\n");
   return 0;
}
optval=1;
for(rp=result;rp!=NULL;rp=rp->ai_next) {
   lfd=socket(rp->ai_family,rp->ai_socktype,rp->ai_protocol);
   if(lfd==-1) {
      printf("Error in getting file descriptor to a socket");
      continue; // try next address
   }
   if(setsockopt(lfd,SOL_SOCKET,SO_REUSEADDR,&optval,sizeof(optval))==-1) {
      printf("Error in setsockopt\n");
      return 0;
   }
   if(bind(lfd,rp->ai_addr,rp->ai_addrlen)==0) {
      break; // on success
   }
   close(lfd); // try next address
```

```
}
if(rp==NULL) {
   printf("Could not bind socket to any address\n");
   return 0;
}
if(listen(lfd,BACKLOG)==-1) {
   printf("Error in listen\n");
   return 0;
}
freeaddrinfo(result);
// to handle client requests
for(;;) {
   addrlen=sizeof(claddr);
   cfd=accept(lfd,(struct sockaddr *)&claddr,&addrlen);
   if(cfd==-1) {
      printf("Error in acccepting client's request");
      continue;
   }
   if(getnameinfo((struct sockaddr *)&claddr,addrlen,host,NI_MAXHOST,service,NI_MAXSERV,0)==0) {
      snprintf(addr_str,ADDR_STR_LEN,"(%s, %s)",host,service);
   }
   printf("Connection from %s\n",addr_str);
   // forking a child process to handle each request
   if(!fork()) {
      if (!(s = pa_simple_new(NULL, argv[0], PA_STREAM_RECORD, NULL, "record", &ss, NULL, NULL, &error))) {
      fprintf(stderr, __FILE__": pa_simple_new() failed: %s\n", pa_strerror(error));
      goto finish;
      }
      for (;;) {
         uint8_t buf[BUFSIZE];
         /* Record some data ... */
         if (pa_simple_read(s, buf, sizeof(buf), &error) < 0) {
            fprintf(stderr, __FILE__": pa_simple_read() failed: %s\n", pa_strerror(error));
            goto finish;
         }

         encryption((uint8_t*)buf);
```

```
        /* And write it to STDOUT */
        if (loop_write(cfd, buf, sizeof(buf)) != sizeof(buf)) {
            fprintf(stderr, __FILE__": write() failed: %s\n", strerror(errno));
            goto finish;
        }
    }
    ret = 0;
    finish:
    if (s)
        pa_simple_free(s);
    return ret;
    exit(1);
}
else {
    if (!(s = pa_simple_new(NULL, argv[0], PA_STREAM_PLAYBACK, NULL, "playback", &ss, NULL, NULL, &error))) {
        fprintf(stderr, __FILE__": pa_simple_new() failed: %s\n", pa_strerror(error));
        goto finish_play;
    }
for (;;) {
    uint8_t buf[BUFSIZE];
    ssize_t r;
    #if 0
        pa_usec_t latency;
        if ((latency = pa_simple_get_latency(s, &error)) == (pa_usec_t) -1) {
            fprintf(stderr, __FILE__": pa_simple_get_latency() failed: %s\n", pa_strerror(error));
            goto finish;
        }
        fprintf(stderr, "%0.0f usec    \r", (float)latency);
    #endif
    /* Read some data ... */
    if ((r = read(cfd, buf, sizeof(buf))) <= 0) {
        if (r == 0) /* EOF */
            break;
        fprintf(stderr, __FILE__": read() failed: %s\n", strerror(errno));
        goto finish_play;
    }

    encryption((uint8_t*)buf);
```

```
            /* ... and play it */
            if (pa_simple_write(s, buf, (size_t) r, &error) < 0) {
                fprintf(stderr, __FILE__": pa_simple_write() failed: %s\n", pa_strerror(error));
                goto finish_play;
            }
        }
        /* Make sure that every single sample was played */
        if (pa_simple_drain(s, &error) < 0) {
            fprintf(stderr, __FILE__": pa_simple_drain() failed: %s\n", pa_strerror(error));
            goto finish_play;
        }
        ret = 0;
        finish_play:
            if (s)
                pa_simple_free(s);
        return ret;
        }
    }
    return 0;
}
```

ДОДАТОК И

Файл client.c

```
#ifdef HAVE_CONFIG_H
#include <config.h>
#endif
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <stdlib.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>
#include <errno.h>
#include <pulse/simple.h>
#include <pulse/error.h>
#include <strumok/strumoks.h>

#define BUFSIZE 1024

#define MAX_DATA_SIZE 50


int sfd;

void encryption(uint8_t*data){
 Dstu8845Ctx *ctx = dstu8845_alloc();
    uint64_t key[8], iv[4];

    iv[0] = 1;
    iv[1] = 2;
    iv[2] = 3;
    iv[3] = 4;

    uint8_t key_size = 64;

    key[7] = 0x8000000000000000;
    key[6] = 0x0000000000000000;
    key[5] = 0x0000000000000000;
    key[4] = 0x0000000000000000;
    key[3] = 0x0000000000000000;
    key[2] = 0x0000000000000000;
    key[1] = 0x0000000000000000;
```

```
    key[0] = 0x0000000000000000;

    dstu8845_init(ctx, key, key_size, iv);
    dstu8845_crypt(ctx, data, countOfBit, data);
}
void my_signal_handler(int sig_num) {
    char answer[2];
    if(sig_num==SIGINT) {
        printf("\nReceived SIGINT i.e ctrl+c\n");
        printf("Do you want to terminate the process (Y/N): ");
        scanf("%s",answer);
        if(strncmp(answer,"Y",1)==0) {
            close(sfd);
            exit(0);
        }
    }
}

/* A simple routine calling UNIX write() in a loop */
static ssize_t loop_write(int fd, const void*data, size_t size) {
    ssize_t ret = 0;
    while (size > 0) {
        ssize_t r;
        if ((r = write(fd, data, size)) < 0)
            return r;
        if (r == 0)
            break;
        ret += r;
        data = (const uint8_t*) data + r;
        size -= (size_t) r;
    }
    return ret;
}

int main(int argc, char *argv[]) {
    struct addrinfo hints;
    struct addrinfo *result,*rp;
    char message[MAX_DATA_SIZE];
    char buffer[MAX_DATA_SIZE];
    ssize_t num_bytes;
    static const pa_sample_spec ss = {
        .format = PA_SAMPLE_S16LE,
        .rate = 44100,
        .channels = 2
    };
    pa_simple *s = NULL;
    int ret = 1;
```

```
int error;
// to control number of arguments passed to program
if(argc!=3) {
   printf("Usage: ./client <server_ip> <listening_port_of_server>\n");
   return 0;
}
// installing signal handler to handle SIGINT
if(signal(SIGINT,my_signal_handler)==SIG_ERR) {
   printf("Error in handling the SIGINT signal\n");
   return 0;
}
// to fill the sizeof(struct addrinfo) bytes of memory area pointed by &hints by zero
memset(&hints,0,sizeof(struct addrinfo));
hints.ai_family=AF_UNSPEC;
hints.ai_socktype=SOCK_STREAM;
if(getaddrinfo(argv[1],argv[2],&hints,&result)!=0) {
   printf("Error in getaddrinfo\n");
   return 0;
}
for(rp=result;rp!=NULL;rp=rp->ai_next) {
   sfd=socket(rp->ai_family,rp->ai_socktype,rp->ai_protocol);
   if(sfd==-1) {
      printf("Error in getting file descriptor to a socket");
      continue; // try next address
   }
   if(connect(sfd,rp->ai_addr,rp->ai_addrlen)==0) {
      break;
   }
   close(sfd); // try next address
}
if(rp==NULL) {
   printf("Could not connect to server\n");
   return 0;
}
freeaddrinfo(result);
if(!fork()) {
   if (!(s = pa_simple_new(NULL, argv[0], PA_STREAM_RECORD, NULL, "record", &ss, NULL, NULL, &error))) {
      fprintf(stderr, __FILE__": pa_simple_new() failed: %s\n", pa_strerror(error));
      goto finish;
   }
   for (;;) {
      uint8_t buf[BUFSIZE];
      /* Record some data ... */
      if (pa_simple_read(s, buf, sizeof(buf), &error) < 0) {
         fprintf(stderr, __FILE__": pa_simple_read() failed: %s\n", pa_strerror(error));
         goto finish;
      }
```

```
            encryption((uint8_t*)buf);

            /* And write it to STDOUT */
            if (loop_write(sfd, buf, sizeof(buf)) != sizeof(buf)) {
                fprintf(stderr, __FILE__": write() failed: %s\n", strerror(errno));
                goto finish;
            }
        }
        ret = 0;
        finish:
        if (s)
            pa_simple_free(s);
        return ret;
        exit(1);
    }
    else {
        if (!(s = pa_simple_new(NULL, argv[0], PA_STREAM_PLAYBACK, NULL, "playback", &ss, NULL, NULL, &error))) {
            fprintf(stderr, __FILE__": pa_simple_new() failed: %s\n", pa_strerror(error));
            goto finish_play;
        }
        for (;;) {
            uint8_t buf[BUFSIZE];
            ssize_t r;
            #if 0
                pa_usec_t latency;
                if ((latency = pa_simple_get_latency(s, &error)) == (pa_usec_t) -1) {
                    fprintf(stderr, __FILE__": pa_simple_get_latency() failed: %s\n", pa_strerror(error));
                    goto finish;
                }
                fprintf(stderr, "%0.0f usec    \r", (float)latency);
            #endif
            /* Read some data ... */
            if ((r = read(sfd, buf, sizeof(buf))) <= 0) {
                if (r == 0) /* EOF */
                    break;
                fprintf(stderr, __FILE__": read() failed: %s\n", strerror(errno));
                goto finish_play;
            }

            encryption((uint8_t*)buf);

            /* ... and play it */
            if (pa_simple_write(s, buf, (size_t) r, &error) < 0) {
                fprintf(stderr, __FILE__": pa_simple_write() failed: %s\n", pa_strerror(error));
                goto finish_play;
            }
```

```
        }
        /* Make sure that every single sample was played */
        if (pa_simple_drain(s, &error) < 0) {
            fprintf(stderr, __FILE__": pa_simple_drain() failed: %s\n", pa_strerror(error));
            goto finish_play;
        }
        ret = 0;
        finish_play:
            if (s)
                pa_simple_free(s);
        return ret;
    }
    return 0;
}
```