МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Механико-математический факультет

Кафедра математической теории интеллектуальных систем

Курсовая работа

ПРОТОТИП КРИПТРОПРОЦЕССОРА ДЛЯ СИММЕТРИЧНОГО ШИФРОВАНИЯ
CRYPTOGRAPHIC PROCESSOR PROTOTYPE FOR SYMMETRICAL ENCRYPTION

Подготовила:
Студентка 405 группы
Елисеева Анастасия Васильевна

Научный руководитель:
Галатенко Алексей Владимирович

Москва, 2025г.

1 Аннотация

Курсовая работа направлена на проектирование и реализацию прототипа криптопроцессора, ориентированного на выполнение операций с использованием симметричных криптоалгоритмов (блочных/потоковых шифров, хэш-функций) и режимов использования.

Основные этапы исследования включают анализ классических режимов шифрования, проектирование архитектуры криптопроцессора с описанием схемы на языке Verilog, а также тестирование схем выбранных режимов.

2 Введение

2.1 Актуальность задач проектирования криптопроцессоров

Криптопроцессоры представляют собой специализированные системы на кристалле (SoC), которые выполняют криптографические операции с максимальной безопасностью [1]. Они используются для различных задач, включая аутентификацию, шифрование и подписание документов [2]. Как отмечают эксперты, такие процессоры могут быть интегрированы в смарт-карты, банкоматы и другие системы, требующие высокого уровня безопасности [3]. Разработка энергоэффективных криптопроцессоров для интернета вещей косвенно стимулирует создание аналогичных решений для сетевого оборудования, включая маршрутизаторы, где требования к скорости и надёжности шифрования особенно высоки.

Исследования показывают, что встроенные криптопроцессоры позволяют экономить ресурсы центрального процессора на 30-40% по сравнению с программными реализациями [4], а также исключают необходимость дополнительных физических средств защиты [5], что делает их привлекательным решением для современных технологий.

По данным отраслевых аналитиков, в последние годы наблюдается растущий интерес к разработке отечественных криптопроцессоров [6]. Эти разработки направлены на создание защищенных чипов для применения в нефтедобывающей и горнодобывающей промышленности [7], а также для производства интеллектуальных сенсоров в автомобильной и авиационной отраслях [8].

2.2 Структура работы

В начале будут разобраны классические режимы шифрования в соответствии со стандартами NIST, затем будет описано проектирование интерфейсов криптопроцессора, реализация модулей и тестовой среды и, наконец, результаты тестирования и сравнительные метрики.

Содержание

3 Основные определения и обозначения

Регистровый сдвиг с линейной обратной связью (РСЛОС, LFSR) – это регистровый сдвиг, входной бит которого линейно зависит от битов предыдущего состояния.
Применяется для генерации псевдослучайных последовательностей битов. При аппаратной реализации может задаваться полиномами.

Блочный шифр – это шифр, оперирующий блоками (дополненного по необходимости) текста равной и фиксированной длины. Один из видов блочных шифров – *SP-сеть* (substitution-permutation network, SPN) – чередование нелинейной подстановки и линейной перестановки на каждом раунде.

S-блок – это нелинейное преобразование, принимающее на вход n бит и возвращающее m бит. Данное преобразование проще всего представлять как таблицу подстановок размером n × m или как булев оператор $(f_1(x_1, ..., x_n), ..., f_m(x_1, ..., x_n)) : E_2^n \to E_2^m$.

Линейная перестановка, P-блок (bit shuffling) – этап работы криптографического алгоритма, на котором входные биты перемешиваются по какому-то правилу в несколько этапов. Этот слой нужен для распространения влияния изменения одного бита входного текста на выходные биты и применяется между нелинейными этапами в течение нескольких *раундов*.

4 Режимы шифрования

DES (Data Encryption Standard) - симметричный блочный алгоритм шифрования, разработанный IBM в 1970-х годах и принятый в качестве федерального стандарта США в 1977 году [9]. Он использует ключ длиной 56 бит (формально 64 бита, включая 8 бит проверки чётности) и обрабатывает данные блоками по 64 бита. Алгоритм включает 16 раундов шифрования, в каждом из которых применяются перестановки, замены (S-блоки) и операция XOR с подключом.

Изначально DES считался криптостойким и широко применялся в финансовой сфере и защите данных. Однако из-за сравнительно короткого ключа он стал уязвим к атаке полным перебором по мере роста вычислительных мощностей. Это привело к появлению усиленных вариантов, таких как *3DES* (тройное шифрование DES), и в конечном итоге — к замене на более современный стандарт *AES (Advanced Encryption Standard)* [10].

Приведенные ниже алгоритмы[11] являются универсальными схемами блочного шифрования и не зависят от стандарта, поэтому разберем их на примере DES.

4.1 Режим ECB

Режим электронной кодовой книги (ECB) определяется следующим образом:

В шифровании ECB блок данных открытого текста $(D_1, D_2, \ldots, D_{64})$ используется напрямую как входной блок $(I_1, I_2, \ldots, I_{64})$ шифрующего устройства. Входной блок обрабатывается устройством в состоянии шифрования. Результирующий выходной блок $(O_1, O_2, \ldots, O_{64})$ используется как шифротекст $(C_1, C_2, \ldots, C_{64})$ или может быть использован в последую-

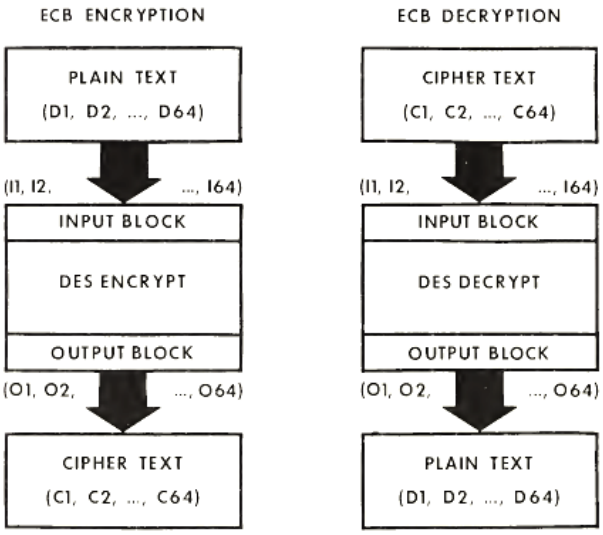

Рис. 1: Схематичное изображение режима ECB (на примере DES)

щих операциях автоматизированной обработки данных.

При расшифровке ECB блок шифротекста $(C_1, C_2, \ldots, C_{64})$ используется напрямую как входной блок $(I_1, I_2, \ldots, I_{64})$ для шифрующего устройства в состоянии расшифрования. Результирующий выходной блок $(O_1, O_2, \ldots, O_{64})$ восстанавливает открытый текст $(D_1, D_2, \ldots, D_{64})$ или может быть использован в последующих операциях. Процесс расшифрования ECB идентичен процессу шифрования, за исключением использования состояния расшифрования в устройстве вместо состояния шифрования.

4.2 Режим CBC

Режим сцепления блоков шифротекста (CBC) определяется следующим образом:

Для шифрования сообщение делится на блоки равной длины и дополняется при необходимости. В шифровании CBC первый входной блок формируется путем побитового сложения по модулю 2 (XOR) первого блока сообщения с 64-битовым вектором инициализации (IV), т.е.:

$$(I_1, I_2, \ldots, I_{64}) = (IV_1 \oplus D_1, IV_2 \oplus D_2, \ldots, IV_{64} \oplus D_{64}).$$

Входной блок обрабатывается устройством в состоянии шифрования, и результирующий выходной блок используется как шифротекст:

$$(C_1, C_2, \ldots, C_{64}) = (O_1, O_2, \ldots, O_{64}).$$

Этот первый блок шифротекста затем подвергается операции XOR со вторым блоком открытого текста для формирования второго входного блока (теперь I' и D' относятся ко второму блоку):

$$(I'_1, I'_2, \ldots, I'_{64}) = (C'_1 \oplus D'_1, C'_2 \oplus D'_2, \ldots, C'_{64} \oplus D'_{64}).$$

Второй входной блок обрабатывается устройством в состоянии шифрования для получения второго блока шифротекста. Этот процесс продолжается, пока не будет зашифрован

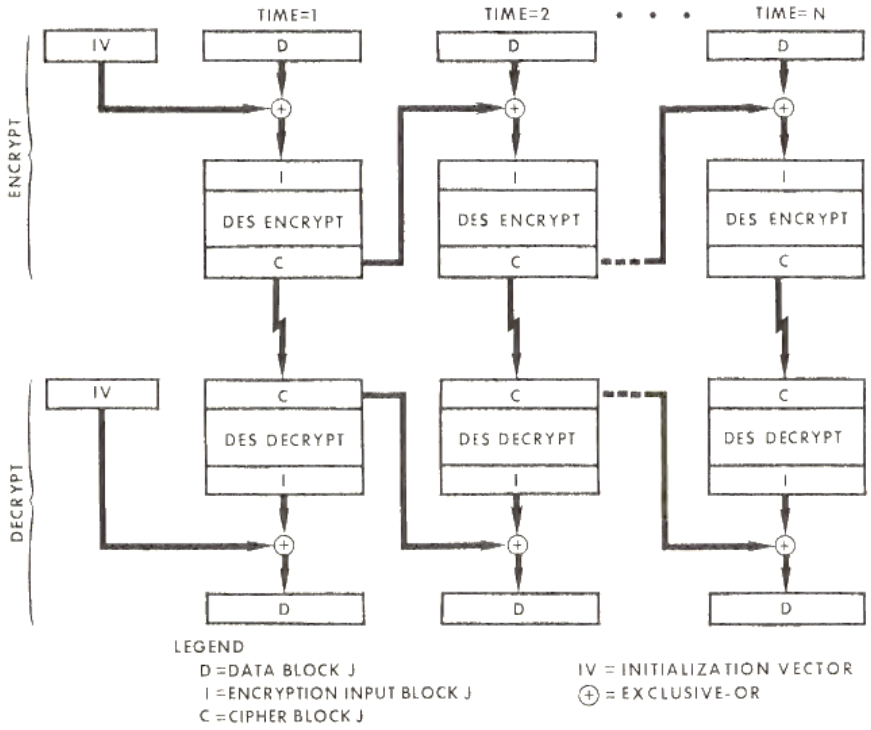

Рис. 2: Схематичное изображение режима CBC (на примере DES)

последний блок открытого текста в сообщении.

При использовании режима CBC каждый блок шифротекста зависит не только от исходного текста, но и от всех предыдущих блоков. Это обеспечивает более высокую безопасность за счёт усложнения процесса шифрования и расшифрования.

В процессе расшифровки CBC первый блок шифротекста используется как входной блок и обрабатывается устройством в состоянии расшифровки, т.е.:

$$(I_1, I_2, \ldots, I_{64}) = (C_1, C_2, \ldots, C_{64}).$$

Результирующий выходной блок, который равен исходному входному блоку при шифровании, подвергается операции исключающего ИЛИ (XOR) с начальным вектором (IV), который должен быть таким же, как использованный при шифровании. Это позволяет получить первый блок открытого текста:

$$(D_1, D_2, \ldots, D_{64}) = (O_1 \oplus IV_1, O_2 \oplus IV_2, \ldots, O_{64} \oplus IV_{64}).$$

Затем второй блок шифротекста используется как входной блок и обрабатывается устройством DES в состоянии расшифрования. Результирующий выходной блок подвергается операции XOR с первым блоком шифротекста для получения второго блока данных открытого текста (D' и O' относятся ко второму блоку):

$$(D'_1, D'_2, \ldots, D'_{64}) = (O'_1 \oplus C_1, O'_2 \oplus C_2, \ldots, O'_{64} \oplus C_{64}),$$

Процесс расшифрования CBC продолжается таким образом до тех пор, пока последний полный блок шифротекста не будет расшифрован.

4.3 Режим CFB

Режим обратной связи по шифру (CFB) определяется следующим образом:

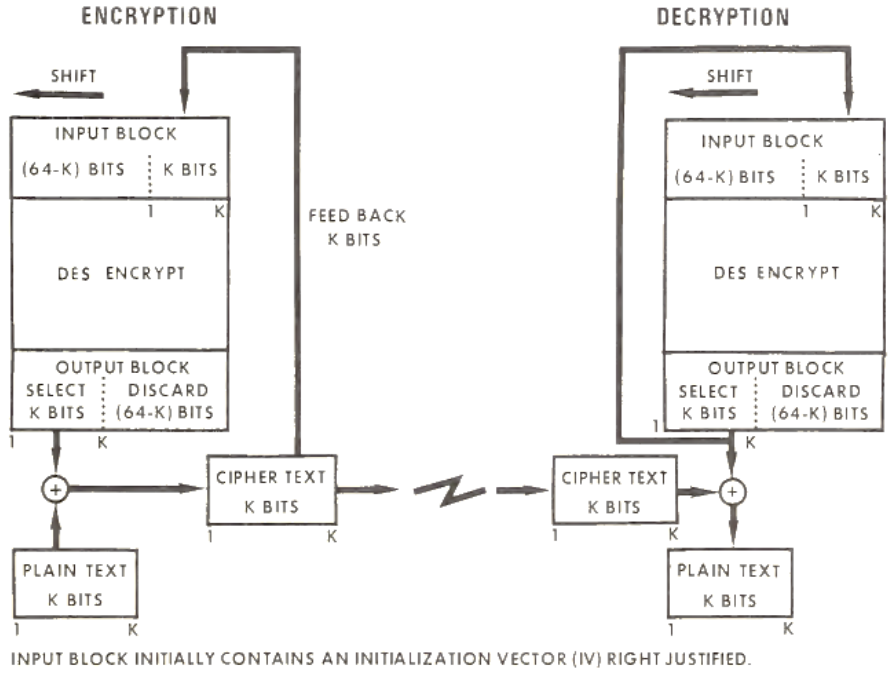

Рис. 3: Схематичное изображение режима CFB (на примере DES)

Сообщение, подлежащее шифрованию, делится на блоки данных, каждый из которых содержит K битов ($K = 1, 2, \ldots, 64$). В операциях как шифрования, так и расшифрования используется вектор инициализации (IV) длиной L. Вектор IV размещается в наименее значимых битах входного блока, а неиспользуемые биты устанавливаются в 0, т.е.:

$$(I_1, I_2, \ldots, I_{64}) = (0, 0, \ldots, 0, IV_1, IV_2, \ldots, IV_L).$$

Этот входной блок обрабатывается устройством в состоянии шифрования для получения выходного блока. При шифровании шифротекст производится путем побитового сложения по модулю 2 (XOR) K-битовой единицы открытого текста с K наиболее значимыми битами выходного блока:

$$(C_1, C_2, \ldots, C_K) = (D_1 \oplus O_1, D_2 \oplus O_2, \ldots, D_K \oplus O_K).$$

Аналогично, при расшифровании открытый текст производится путем XOR K-битовой единицы шифротекста с K наиболее значимыми битами выходного блока:

$$(D_1, D_2, \ldots, D_K) = (C_1 \oplus O_1, C_2 \oplus O_2, \ldots, C_K \oplus O_K).$$

В обоих случаях неиспользуемые биты выходного блока отбрасываются.

Следующий входной блок формируется путем отбрасывания K наиболее значимых битов предыдущего входного блока, сдвига оставшихся битов на K позиций влево и вставки K битов **только что сформированного шифротекста (при шифровании) или использованного (при расшифровании)** в наименее значимые биты:

$$(I_1, I_2, \ldots, I_{64}) = (I_{K+1}, I_{K+2}, \ldots, I_{64}, C_1, C_2, \ldots, C_K).$$

Этот блок затем обрабатывается устройством в состоянии шифрования для получения следующего выходного блока. Этот процесс продолжается до тех пор, пока не будет зашифровано все сообщение открытого текста или не будет расшифровано все сообщение шифротекста.

4.4 Режим OFB

Режим обратной связи по выходу (OFB) определяется аналогично режиму CFB с отличием в том, что во время сдвига блок дополняется битами не шифртекста, а открытого текста:

$$(I_1, I_2, \ldots, I_{64}) = (I_{K+1}, I_{K+2}, \ldots, I_{64}, O_1, O_2, \ldots, O_K).$$

4.5 Режим CTR

Режим счетчика (CTR) определяется следующим образом:

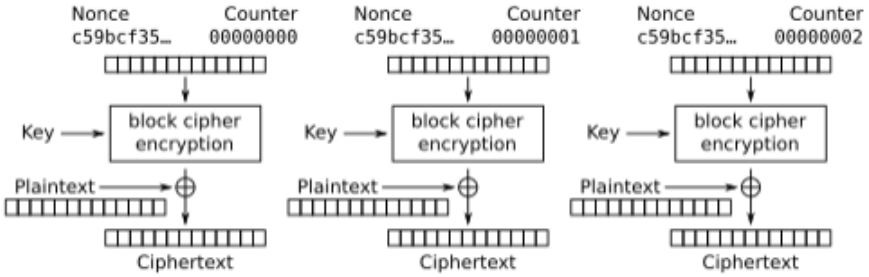

Рис. 4: Схематичное изображение режима CTR [12]

Сообщение, подлежащее шифрованию, делится на единицы данных, каждая из которых содержит K битов ($K = 1, 2, \ldots, 64$). В операциях как шифрования, так и расшифрования используется счетчик длиной L. Счетчик объединяется с уникальным значением блока для формирования входного блока. Наиболее распространенный подход предполагает конкатенацию счетчика и порядкового номера блока:

$$(I_1, I_2, \ldots, I_{64}) = (N_1, N_2, \ldots, N_L, C_1, C_2, \ldots, C_{64-L}).$$

Каждый входной блок обрабатывается устройством в состоянии шифрования для получения выходного блока. При шифровании шифротекст производится путем побитового сложения по модулю 2 (XOR) K-битовой единицы открытого текста с K наиболее значимыми битами выходного блока:

$$(C_1, C_2, \ldots, C_K) = (D_1 \oplus O_1, D_2 \oplus O_2, \ldots, D_K \oplus O_K).$$

Аналогично, при расшифровании открытый текст производится путем XOR K-битовой единицы шифротекста с K наиболее значимыми битами выходного блока:

$$(D_1, D_2, \ldots, D_K) = (C_1 \oplus O_1, C_2 \oplus O_2, \ldots, C_K \oplus O_K).$$

Каждый следующий входной блок формируется путем применения какой-то вектор-функции к битам счетчика для его перехода в новое состояние, причем обычно используется просто

инкремент:

$$(C'_1, C'_2, \ldots, C'_{64-L}) = F(C_1, C_2, \ldots, C_{64-L})$$

Каждый следующий входной блок формируется путем инкремента счетчика:

$$(I_1, I_2, \ldots, I_{64}) = (N_1, N_2, \ldots, N_L, C_1 + 1, C_2 + 1, \ldots, C_{64-L} + 1).$$

Этот блок затем обрабатывается устройством в состоянии шифрования для получения следующего выходного блока. Процесс продолжается до полной обработки сообщения.

4.6 Особенности использования режимов

Режим	**Преимущества**	**Недостатки**	**Применение**
ECB	Простота, быстрота, ошибки локализуются	Уязвимость к анализу частотности	Небольшие данные, шифрование ключей
CBC	Безопаснее ECB	Ошибки распространяются	Блокоориентированная передача и аутентификация
CFB	Обрабатывает поток	Ошибки распространяются	Потокоориентированная передача и аутентификация
OFB	Обрабатывает поток, ошибки локализуются	Менее устойчив к модификации потока, чем CFB	Системы с локализацией ошибок
CTR	Параллелизм, ошибки локализуются	Требует уникального счетчика для каждого ключа	Высокопроизводительные системы, потоковое шифрование

Таблица 1: Сравнение режимов шифрования ECB, CBC, CFB, OFB и CTR

Блочно-ориентированная передача предполагает деление данных на фиксированные блоки, которые передаются целиком [13]. Такой подход обеспечивает эффективное управление данными, особенно в системах с произвольным доступом, например, в дисковых файловых системах [14].

Потоко-ориентированная передача, напротив, представляет собой непрерывный поток данных, который передается без разделения на фиксированные блоки [15]. Такой способ передачи подходит для приложений, где важна непрерывность и последовательность данных, например, в потоковых мультимедиа-приложениях или сетевых протоколах [16].

5 Мультиплексоры и демультиплексоры

Мультиплексор (Multiplexer, MUX) [17] — это схема из функциональных элементов, которая выбирает один из нескольких входных сигналов и передаёт его на единственный выход. Управление осуществляется с помощью *селекторных сигналов* (адресных линий).

Формально, мультиплексор с 2^n входами и 1 выходом реализует функцию:

$$Y = \sum_{i=0}^{2^n-1} D_i \cdot \prod_{j=0}^{n-1} S_j^{b_j(i)}$$

где S_j — биты селектора, $b_j(i)$ — j-й бит числа i.

S_1	S_0	Выход Y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

Таблица 2: Таблица истинности мультиплексора 4:1

Пример реализации мультиплексора на Verilog:

```
module mux_4to1 (
    input [3:0] D,      // 4 input lines
    input [1:0] S,      // 2-bit selector
    output reg Y        // output
);
always @(*) begin
    case (S)
        2'b00: Y = D[0];
        2'b01: Y = D[1];
        2'b10: Y = D[2];
        2'b11: Y = D[3];
    endcase
end
endmodule
```

Демультиплексор (Demultiplexer, DEMUX) выполняет обратную мультиплексору функцию — направляет один входной сигнал на один из нескольких выходов в зависимости от значения селектора.

Формально, демультиплексор с 1 входом и 2^n выходами реализует:

$$Y_i = \begin{cases} D, & \text{если } i = S \\ 0, & \text{иначе} \end{cases}$$

S_1	S_0	D_0	D_1	D_2	D_3
0	0	Y	0	0	0
0	1	0	Y	0	0
1	0	0	0	Y	0
1	1	0	0	0	Y

Таблица 3: Таблица истинности демультиплексора 1:4

Пример реализации демультиплексора на Verilog:

```
module demux_1to4 (
    input Y,            // input
    input [1:0] S,      // selector
    output reg [3:0] D // 4 outputs
);
always @(*) begin
    D = 4'b0;           // by default
    case (S)
        2'b00: D[0] = Y;
        2'b01: D[1] = Y;
        2'b10: D[2] = Y;
        2'b11: D[3] = Y;
    endcase
end
endmodule
```

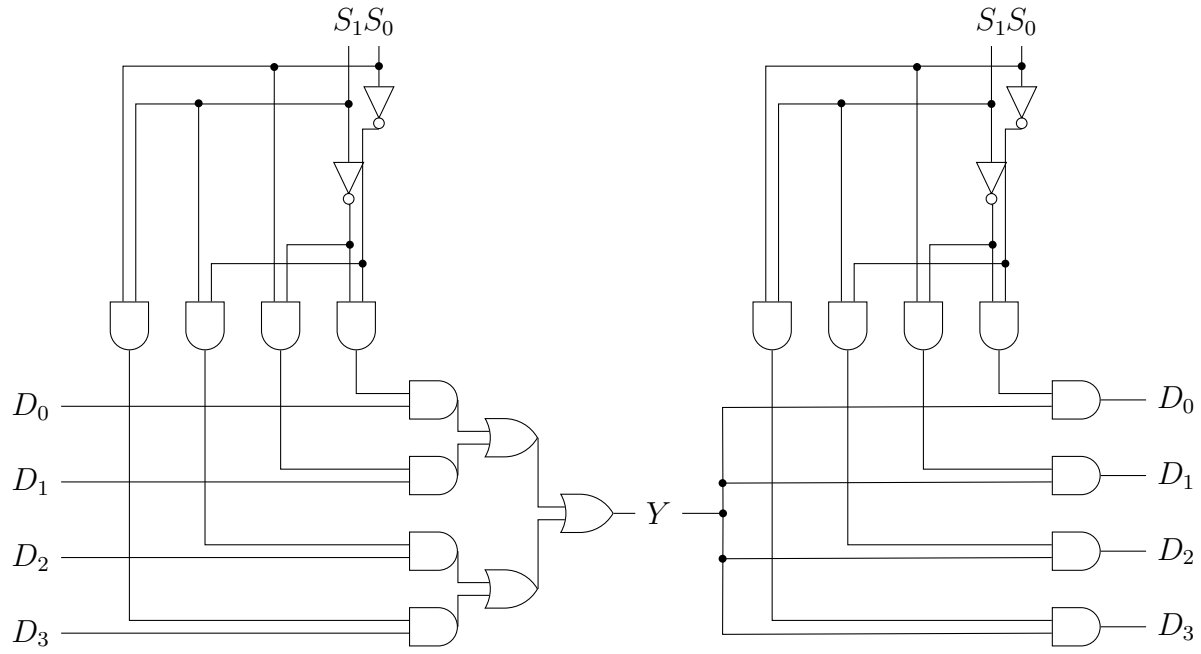

Рис. 5: СФЭ для мультиплексора и демультиплексора

Схема	**Глубина**	**Сложность**
Мультиплексор	4	11
Демультиплексор	2	8

Таблица 4: Характеристики схем мультиплексора и демультиплексора

6 Криптопроцессоры

Криптопроцессор [17] — специализированное вычислительное устройство, предназначенное для эффективного выполнения криптографических операций. Формально можно определить как кортеж:

$$\mathcal{CP} = \langle \mathcal{A}, \mathcal{M}, \mathcal{K}, \mathcal{I}, \mathcal{O} \rangle, \text{ где:} \tag{1}$$

- $\mathcal{A}$ — набор реализованных алгоритмов,
- $\mathcal{M}$ — поддерживаемые режимы работы,
- $\mathcal{K}$ — пространство ключей,
- $\mathcal{I}$ — входные интерфейсы,
- $\mathcal{O}$ — выходные интерфейсы.

Производительность характеризуется тройкой параметров $P = (T_p, L, R)$:

$$\begin{aligned} T_p &= \frac{N_{blocks}}{t_{total}} \text{ — пропускная способность} \\ L &= t_{latency} \text{ — задержка} \\ R &= \frac{T_p}{S} \text{ — эффективность} \end{aligned}$$

где S — площадь кристалла.

Безопасность оценивается через $Sec = \min\{Sec_{alg}, Sec_{impl}\}$:

$$\begin{aligned} Sec_{alg} &\text{— стойкость алгоритма} \\ Sec_{impl} &\text{— стойкость реализации к SCA-атакам} \end{aligned}$$

7 Поля Галуа

7.1 Основные понятия и утверждения из алгебры

С доказательствами можно ознакомиться, например, в [18].

Полем называется множество $\mathbb{F}$ с двумя бинарными операциями

$$+ : \mathbb{F} \times \mathbb{F} \to \mathbb{F} \quad \text{(сложение)}$$
$$\cdot : \mathbb{F} \times \mathbb{F} \to \mathbb{F} \quad \text{(умножение)}$$

удовлетворяющее следующим аксиомам:

1. Абелева группа по сложению:
   - Ассоциативность: $\forall a, b, c \in \mathbb{F} \quad (a + b) + c = a + (b + c)$
   - Коммутативность: $\forall a, b \in \mathbb{F} \quad a + b = b + a$
   - Нулевой элемент: $\exists 0 \in \mathbb{F} \quad \forall a \in \mathbb{F} \quad a + 0 = a$
   - Обратный элемент: $\forall a \in \mathbb{F} \quad \exists(-a) \in \mathbb{F} \quad a + (-a) = 0$
2. Абелева группа по умножению:
   - Ассоциативность: $\forall a, b, c \in \mathbb{F} \quad (a \cdot b) \cdot c = a \cdot (b \cdot c)$
   - Коммутативность: $\forall a, b \in \mathbb{F} \quad a \cdot b = b \cdot a$
   - Единичный элемент: $\exists 1 \in \mathbb{F} \setminus \{0\} \quad \forall a \in \mathbb{F} \quad a \cdot 1 = a$
   - Обратный элемент: $\forall a \in \mathbb{F} \setminus \{0\} \quad \exists a^{-1} \in \mathbb{F} \setminus \{0\} \quad a \cdot a^{-1} = 1$
3. Дистрибутивность умножения относительно сложения:

$$\forall a, b, c \in \mathbb{F} \quad a \cdot (b + c) = a \cdot b + a \cdot c$$

Пусть q — положительное целое число. Факторкольцо, называемое *кольцом вычетов по модулю* q и обозначаемое $\mathbb{Z}/(q)$ или $\mathbb{Z}_q$, представляет собой множество $\{0, \ldots, q-1\}$ с операциями сложения и умножения, определёнными следующим образом:
$a + b = R_q[a + b], \quad a \cdot b = R_q[ab]$, где $R_q[n]$ - остаток n при делении на q.

Утверждение. Факторкольцо $\mathbb{Z}/(q)$ является полем тогда и только тогда, когда q — простое число.

GF(q) обозначают $\mathbb{Z}_q$ при простом q, чтобы подчеркнуть, что это поле.

Полиномом над произвольным полем $\mathbb{F}$ называется формальное выражение вида

$$f(x) = f_{n-1}x^{n-1} + f_{n-2}x^{n-2} + \cdots + f_1 x + f_0,$$

где:

- x — формальная переменная (неизвестное)
- Коэффициенты $f_{n-1}, \ldots, f_0$ принадлежат полю $\mathbb{F}$

- Индекс i коэффициента f_i является целым числом
- Показатели степени при x — целые числа

Нулевой полином определяется как $f(x) = 0$.
Унитарный (приведённый) полином – полином, у которого старший коэффициент f_{n-1} равен единице.
$f(x)$ и $g(x)$ *равны*, если равны их соответствующие коэффициенты ($f_i = g_i$ для всех i).
Степень полинома ($\deg f(x)$) для ненулевого полинома — индекс старшего ненулевого коэффициента. Для нулевого полинома по определению $\deg 0 = -\infty$.

Множество всех полиномов над GF(q) образует *кольцо полиномов* при стандартном определении операций сложения и умножения. Это кольцо обозначается GF(q)[x]. Элементы поля GF(q) в этом контексте называются *скалярами*.

Операции в GF(q)[x]:

- Сумма полиномов:

$$f(x) + g(x) = \sum_{i=0}^{\infty}(f_i + g_i)x^i, \quad deg(f+g) \leq \max(deg(f), deg(g))$$

- Умножение полиномов:

$$f(x) \cdot g(x) = \sum_{i=0}^{\infty}\left(\sum_{j=0}^{i} f_j g_{i-j}\right)x^i, \quad deg(f \cdot g) = deg(f) \cdot deg(g)$$

Полином $s(x)$ *делится* на полином $r(x)$ (или $r(x)$ является *делителем* $s(x)$), если существует полином $a(x)$ такой, что:

$$r(x)a(x) = s(x)$$

Неприводимым полиномом называется полином $p(x)$, который делится только на $\alpha p(x)$ и на α, где α — произвольный ненулевой элемент поля GF(q).

Простым полиномом называется унитарный (приведённый) неприводимый полином степени не менее 1.

Теорема о единственности разложения многочленов
Ненулевой полином $p(x)$ над полем допускает единственное разложение (с точностью до порядка множителей) в виде произведения элемента поля на простые полиномы над этим полем:

$$p(x) = \alpha \cdot p_1(x)^{e_1} p_2(x)^{e_2} \cdots p_k(x)^{e_k}$$

где:

- α — ненулевой элемент поля
- $p_i(x)$ — различные простые (унитарные неприводимые) полиномы
- e_i — натуральные числа (кратности множителей)

Для любого унитарного полинома $p(x)$ ненулевой степени над полем $\mathbb{F}$ *кольцо полиномов по модулю* $p(x)$ состоит из всех полиномов степени меньшей, чем $\deg p(x)$ и операций сложения и умножения полиномов по модулю $p(x)$. Это кольцо обозначается $\mathbb{F}[x]/\langle p(x)\rangle$. Любой полином $r(x) \in \mathbb{F}[x]$ отображается в $\mathbb{F}[x]/\langle p(x)\rangle$ через:

$$r(x) \mapsto R_{p(x)}[r(x)]$$

Два полинома $a(x), b(x) \in \mathbb{F}[x]$ называются *конгруэнтными по модулю* $p(x)$, если они отображаются в один элемент:

$$a(x) \equiv b(x) \pmod{p(x)} \iff \exists Q(x) \in \mathbb{F}[x] \quad b(x) = a(x) + Q(x)p(x)$$

Утверждение $\mathbb{F}[x]/\langle p(x)\rangle$ является полем тогда и только тогда, когда $p(x)$ — простой полином.

Для любого простого полинома степени m над $\mathrm{GF}(q)$ можно построить конечное поле из q^m элементов. При таком построении элементы поля представляются полиномами над $\mathrm{GF}(q)$ степени меньше m.
Если p простое и m — натуральное число, то наименьшее подполе $\mathrm{GF}(p^m)$ есть $\mathrm{GF}(p)$. Элементы $\mathrm{GF}(p)$ называются *целыми* элементами $\mathrm{GF}(p^m)$, а p называется *характеристикой*. $\mathrm{GF}(q^n)$ не является подполем $\mathrm{GF}(q^m)$, если n не делит m.

Полученные конструкции называют *полями Галуа* и обозначают $GF(p^n)$.

Пример построения $\mathrm{GF}(4)$ **на основе** $\mathrm{GF}(2)$**:**

Поле $\mathrm{GF}(2)$ состоит из двух элементов: $\{0, 1\}$ с операциями:

Таблица 5: Операции в $\mathrm{GF}(2)$

$+$	0	1
0	0	1
1	1	0

$\times$	0	1
0	0	0
1	0	1

Используем простой полином $p(x) = x^2 + x + 1$, его неприводимость проверяется перебором всех возможных разложений.
Элементы поля представляются множеством полиномов:

$$\{0,\ 1,\ x,\ x+1\}$$

Операции сложения и умножения задаются таблицей:

Таблица 6: Операции в $\mathrm{GF}(4)$

$+$	0	1	x	$x+1$
0	0	1	x	$x+1$
1	1	0	$x+1$	x
x	x	$x+1$	0	1
$x+1$	$x+1$	x	1	0

$\times$	0	1	x	$x+1$
0	0	0	0	0
1	0	1	x	$x+1$
x	0	x	$x+1$	1
$x+1$	0	$x+1$	1	x

7.2 Поля Галуа в криптографии AES

В AES (Advanced Encryption Standard) [19] используется $GF(2^8)$ (256 элементов). Элементы $GF(2^8)$ представляются:

- Как многочлены степени < 8 с коэффициентами в $\mathbb{Z}_2$:

$$a_7x^7 + a_6x^6 + \cdots + a_0 \quad (a_i \in \{0, 1\})$$

- Как байты (8 бит): `0x00` до `0xFF`

Пример: `0x57` $\equiv x^6 + x^4 + x^2 + x + 1$

Алгоритм AES выполняет шифрование через четыре базовые операции:

1. Добавление раундового ключа (AddRoundKey)

   ```
   State = State + RoundKey (mod 2)
   ```

2. Замена байт (SubBytes)

   ```
   for each byte in State:
       byte = S_box[byte]
   ```

   На этом этапе используется взятие обратного в $G(2^8)$.

3. Сдвиг строк (ShiftRows)

Код	**Режим**
0	0 байт влево
1	1 байт влево
2	2 байта влево
3	3 байта влево

4. Смешивание столбцов (MixColumns)
   Матричное умножение в поле GF(2^8):

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

8 Интерфейс прототипа криптопроцессора

Криптопроцессор реализует три режима блочного шифрования (ECB, CBC, CTR) с использованием алгоритма GAGE в качестве блочного шифра.

8.1 Параметры и сигналы модуля

Ниже приведены возможные значения параметров модуля. В качестве примера взяты длина блока и количество раундов в соответствии в DES стандартом, но эти параметры процессора могут быть изменены по желанию. Важно следить за тем, чтобы значения этих параметров процессора были согласованы с параметрами режимов шифрования.

Таблица 7: Параметры модуля CryptoProcessor

Параметр	**Значение**	**Описание**
BLOCK_SIZE	64	Размер блока данных (бит)
KEY_SIZE	64	Размер ключа шифрования (бит)
CAPACITY	512	Вместимость sponge-функции
INTERNAL_STATE_SIZE	576	Размер внутреннего состояния (бит)
ROUNDS	32	Количество раундов шифрования
MAX_BLOCKS	1024	Максимальное число блоков для обработки

Таблица 8: Порты ввода-вывода CryptoProcessor

Порт	**Напр.**	**Разм.**	**Описание**
clk	input	1	Тактовый сигнал
reset	input	1	Асинхронный сброс (активный 1)
start	input	1	Сигнал запуска обработки
mode	input	2	Режим работы: 00-ECB 01-CBC 10-CTR
key	input	KEY_SIZE	Ключ шифрования
iv_nonce	input	BLOCK_SIZE	Вектор инициализации (CBC) Nonce (CTR)
plaintext	input	BLOCK_SIZE ×MAX_BLOCKS	Входные данные
num_blocks	input	16	Количество блоков для обработки
ciphertext	output	BLOCK_SIZE ×MAX_BLOCKS	Результат шифрования
done	output	1	Сигнал завершения обработки

Таблица 9: Режимы работы криптопроцессора

Код	**Режим**
00	ECB
01	CBC
10	CTR
11	-

Таблица 10: Расчётные временные характеристики

Параметр	**Значение**	**Обоснование**
Тактовая частота	100 МГц	Лимиты целевой ПЛИС
Латентность ECB	32 такта	1 такт/раунд × 32 раунда
Латентность CBC	34 такта	ECB + 2 такта XOR
Латентность CTR	33 такта	ECB + 1 такт XOR
Пропускная способность	1.6 Гбит/с	$\frac{64\ \text{bit} \times 100\ \text{MHz}}{4\ \text{pipeline stages}}$

8.2 Принцип работы

Процессор начинает обработку данных при одновременной активности сигналов `start` и `reset='0'`. Режим работы определяется значением `mode`:

$$\text{Режим} = \begin{cases} \text{ECB}, & \text{если } mode = 00 \\ \text{CBC}, & \text{если } mode = 01 \\ \text{CTR}, & \text{если } mode = 10 \end{cases} \tag{2}$$

После завершения вычислений процессор устанавливает сигнал `done` в '1' на один такт и выдает результат на шину `ciphertext`.

[H] Схема работы

Получить входные данные (M, K, IV)
Определить режим $\mu \in \mathcal{M}$
Выполнить:
if $\mu = ECB$ **then**
    $C \leftarrow Enc_{ECB}(M, K)$
else if $\mu = CBC$ **then**
    $C \leftarrow Enc_{CBC}(M, K, IV)$
else if $\mu = CTR$ **then**
    $C \leftarrow Enc_{CTR}(M, K, IV)$
end if
Вернуть C

8.3 Примеры

В качестве блочного шифра для простоты оценки результатов взят модуль, инвертирующий входные биты:

```
ciphertext <= plaintext ^ {BLOCK_SIZE{1'b1}}
```

Так,

```
64'hA5A5A5A5A5A5A5A5
= 1010 0101 1010 0101 1010 0101 1010 0101 1010 0101 1010 0101 1010 0101 1010 0101

64'h5A5A5A5A5A5A5A5A
= 0101 1010 0101 1010 0101 1010 0101 1010 0101 1010 0101 1010 0101 1010 0101 1010
```

В результате тестирования блоков ожиданется выполнение основных свойств режимов и зависимость результата от изменения входных данных.

```
=== Starting Encryption Test ===
Mode:       ECB
Key: 0123456789abcdef
IV/Nonce: fedcba9876543210
Number of blocks: 2

Plaintext blocks:
Block 0: a5a5a5a5a5a5a5a5
Block 1: 5a5a5a5a5a5a5a5a

Ciphertext blocks:
Block 0: 5a5a5a5a5a5a5a5a
Block 1: a5a5a5a5a5a5a5a5

=== Test Completed ===
```

```
=== Starting Encryption Test ===
Mode:       ECB
Key: 0123456789abcdef
IV/Nonce: fedcba9876543210
Number of blocks: 2

Plaintext blocks:
Block 0: b6a5a5a5a5a5a5a5
Block 1: 5a5a5a5a5a5a5a5a

Ciphertext blocks:
Block 0: 495a5a5a5a5a5a5a
Block 1: a5a5a5a5a5a5a5a5

=== Test Completed ===
```

Рис. 6: Режим ECB

```
=== Starting Encryption Test ===
Mode:       CBC
Key: 0123456789abcdef
IV/Nonce: fedcba9876543210
Number of blocks: 2

Plaintext blocks:
Block 0: a5a5a5a5a5a5a5a5
Block 1: 5a5a5a5a5a5a5a5a

Ciphertext blocks:
Block 0: a486e0c22c0e684a
Block 1: 0123456789abcdef

=== Test Completed ===
```

```
=== Starting Encryption Test ===
Mode:       CBC
Key: 0123456789abcdef
IV/Nonce: fedcba9876543210
Number of blocks: 2

Plaintext blocks:
Block 0: b6a5a5a5a5a5a5a5
Block 1: 5a5a5a5a5a5a5a5a

Ciphertext blocks:
Block 0: b786e0c22c0e684a
Block 1: 1223456789abcdef

=== Test Completed ===
```

Рис. 7: Режим CBC

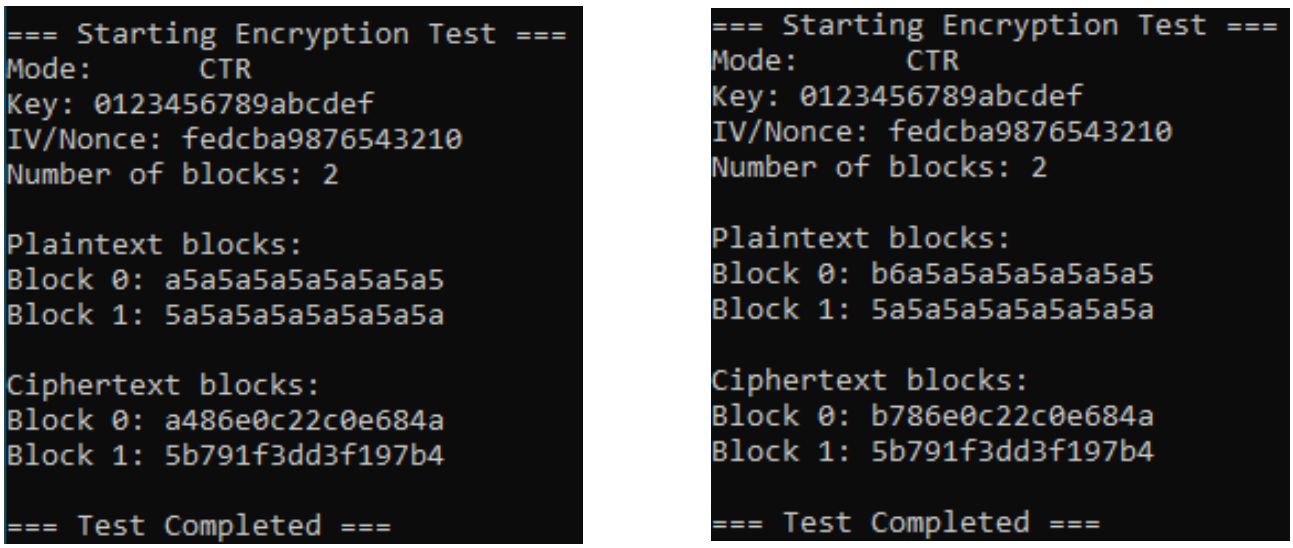

Рис. 8: Режим CTR

Сравнение результатов тестирования:

Таблица 11: Сравнение режимов шифрования

Характеристика	ECB	CBC	CTR
Зависимость блоков	Нет	Да	Нет
Сохранение паттернов	Да	Нет	Нет
Параллельная обработка	Да	Нет	Да
Требования к IV/Nonce	Нет	Да	Да

С реализацией модулей можно ознакомиться в моем github [20].

9 Заключение

В данной курсовой работе были изучены и программно реализованы классические режимы шифрования и стандарты криптографии, проведен их сравнительный анализ, выявлены сильные и слабые стороны, которые были проиллюстрированы с помощью созданного в результате работы прототипа криптопроцессора, использующего эти режимы.

Представляют интерес следующие направления дальнейшего развития:

- поддержка режимов расшифровки;
- поточно-ориентированная реализация с возможностью непрерывной подачи входных данных неограниченной длины;
- реализация дополнения блоков открытого текста до нужной длины (padding);
- реализация конкурентоспособного блочного шифра вместо иллюстративного.

С учебной точки зрения также будет полезным освоение промышленных систем автоматического проектирования (Cadence, Synopsys, Xilinx Vivado).

Список литературы

[1] C. Paar, J. Pelzl, “Understanding Cryptography”, 2010.

[2] A. J. Menezes, P. C. van Oorschot, S. A. Vanstone, “Handbook of Applied Cryptography”, 1996.

[3] P. Kocher, J. Jaffe, B. Jun, “Differential Power Analysis Advances in Cryptology - CRYPTO’99”, 1999.

[4] National Institute of Standards and Technology, “Security Requirements for Cryptographic Modules”, **FIPS PUB 140-3** (2019).

[5] Государственный стандарт РФ, “ГОСТ Р 34.10-2012”, 2012.

[6] Росинформбюро, “Отчет "Рынок криптографических решений РФ 2022"”, 2022.

[7] А. Иванов [и др.], “Криптопроцессор для промышленных систем”, **Патент RU 2685344** (2019).

[8] НИИ "Квант", “Технический отчет по авионике”, 2021.

[9] National Institute of Standards and Technology, “Data Encryption Standard (DES)”, **FIPS PUB 46-3** (1999).

[10] National Institute of Standards and Technology, “Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher”, **NIST Special Publication 800-67 Revision 2** (2012).

[11] U.S. DEPARTMENT OF COMMERCE / National Bureau of Standards, “DES MODES OF OPERATION”, **FIPS PUB 81** (1980).

[12] https://en.wikipedia.org/wiki/Block_cipher_mode_of_operationCounter_(CTR).

[13] A. S. Tanenbaum, D. J. Wetherall, *Computer Networks*, ed. 5th, Pearson Education, 2010.

[14] M. K. McKusick, G. V. Neville-Neil, *The Design and Implementation of the FreeBSD Operating System*, ed. 2nd, Addison-Wesley, 2014.

[15] J. F. Kurose, K. W. Ross, *Computer Networking: A Top-Down Approach*, ed. 7th, Pearson, 2016.

[16] Z. Li, M. Chen, “Adaptive Video Streaming in Wireless Networks”, **15 4** (2013).

[17] Е. П. Угрюмов, *Цифровая схемотехника*, **2**, 2018.

[18] Richard E. Blahut, *Algebraic Codes for Data Transmission*, 2003.

[19] National Institute of Standards and Technology, “Advanced Encryption Standard (AES)”, **FIPS PUB 197** (2023).

[20] https://github.com/MatsuMizu.