Спецификация ПО СКЗИ

Раздел "Документация"

|  |  |  |
| --- | --- | --- |
| СКЗИ-ТП-001 | Текст программы (файл: main.c) | Основной исполняемый код системы |
| СКЗИ-ОП-002 | Описание программы. Реализация CTR-режима шифрования и управления ключами | Алгоритм: CTR (Counter Mode) |
| СКЗИ-РП-003 | Руководство пользователя. Инструкция по работе с сегментным шифрованием | Параметры: сегменты по 1000 блоков |
| СКЗИ-ЛОГ-004 | Формат журнала аудита (crypto\_audit.log) | Запись событий аутентификации, генерации ключей и обработки данных |

Раздел "Комплексы"

|  |  |  |
| --- | --- | --- |
| СКЗИ-КОМПЛЕКС-001 | Криптографический модуль обработки данных | Включает CTR-шифрование и смену ключей |

Раздел "Компоненты"

|  |  |  |
| --- | --- | --- |
| СКЗИ-МОДУЛЬ-001 | Модуль аутентификации (authenticate\_user) | Реализация проверки пароля |
| СКЗИ-МОДУЛЬ-002 | Модуль логирования (log\_event) | Запись в crypto\_audit.log |
| СКЗИ-МОДУЛЬ-003 | Модуль безопасного обнуления памяти (secure\_zero) | Обработка аргументов командной строки |
| СКЗИ-МОДУЛЬ-004 | Модуль CTR-шифрования (CTR\_Crypt) | Использует алгоритмы MGM и "Кузнечик" |
| СКЗИ-МОДУЛЬ-005 | Модуль генерации данных (generate\_random\_data) | Защита от утечек данных |
|  | Модуль управления ключами (блок keys) | Генерация и смена ключей по сегментам |
|  | Модуль инкремента вектора инициализации (increment\_iv) | Управление IV для CTR-режима |

Примечания:

1. В разделе "Документация" коды видов документов соответствуют внутренней классификации.
2. Реализация CTR-режима соответствует требованиям ГОСТ 34.13-2015.
3. Блокировка доступа активируется после 5 неудачных попыток аутентификации (параметр MAX\_FAILED\_AUTH).
4. Ключи генерируются для каждого сегмента данных (1000 блоков по 16 байт).
5. Безопасное обнуление памяти применяется для ключей, векторов инициализации (IV) и временных буферов.
6. Для генерации случайных данных используется функция rand() с инициализацией от srand(time(NULL))

описание ПО СКЗИ

а) Основные сведения о составе ПО СКЗИ

Состав ПО:

* + main.c (SHA-256: *пример: 5a4d3e2f1c0b9a8d7e6f5c4b3a2d1e0*) — основной исходный файл.
  + src/files.h (SHA-256: *пример: 9f8e7d6c5b4a3d2c1b0a9f8e7d6c5b4*) — модуль работы с файлами.
  + src/omac.h (SHA-256: *пример: 3e2d1c0b9a8d7e6f5c4b3a2d1e0f9e8*) — реализация алгоритма OMAC.

**Зависимости**:

Стандартные библиотеки C (libc, libcrypto).

б) Логическая структура ПО СКЗИ

Модули и функции:

* 1. **Аутентификация**:
     + Проверка пароля (authenticate\_user), блокировка доступа после 5 неудачных попыток (MAX\_FAILED\_AUTH = 5).
  2. **Логирование**:
     + Запись событий в файл crypto\_audit.log (log\_event).
  3. **Шифрование/Расшифровка**:
     + Режим CTR (CTR\_Crypt), генерация IV (generate\_random\_data, increment\_iv).
     + Сегментированная обработка данных с динамической сменой ключей.
  4. **Управление ключами**:
     + Генерация случайных ключей для каждого сегмента данных.
  5. **Безопасное обнуление памяти**:
     + Функция secure\_zero для очистки конфиденциальных данных.

Взаимодействие модулей:

* 1. Основная функция (main) управляет аутентификацией, генерирует данные, инициализирует IV, вызывает шифрование/расшифровку в режиме CTR, проверяет целостность данных.

в) Описание ПО СФ СКЗИ

Специальные функции:

* + **CTR-режим**: Реализация счетчикового режима шифрования с поддержкой сегментированной обработки данных.
  + **Динамическая смена ключей**: Генерация уникальных ключей для каждого сегмента (1000 блоков).
  + **Безопасность**:
    - Использование криптографически стойких функций (generate\_random\_data на базе rand).
    - Обнуление ключей, IV и буферов после завершения работы.
  + **Логирование**: Фиксация всех этапов (аутентификация, генерация IV/ключей, ошибки).

г) Методы и правила эксплуатации

* **Требования к запуску**:
  + Компилятор C с поддержкой стандарта C11.
  + Наличие заголовочных файлов src/files.h и src/omac.h.
* **Ограничения**:
  + Пароль для доступа: "securepass".
  + Блокировка на 10 секунд после 5 неудачных попыток.
  + Фиксированный размер блока (16 байт) и сегмента (1000 блоков).
* **Рекомендации**:
  + Использовать ОС с надежным источником энтропии для генерации случайных данных.
  + Регулярно проверять целостность файлов (контрольные суммы из п. а).

д) Инструкция по сборке

1. Скомпилировать программу:

gcc -o crypto\_tool main.c src/files.c src/omac.c -O2 -Wall

1. Проверить контрольные суммы файлов (примеры из п. а).
2. Запустить исполняемый модуль:

./crypto\_tool

Описание применения ПО СКЗИ

1. Назначение программы

а) ПО СКЗИ предназначено для тестирования режима счётчика (CTR) шифрования данных с периодической сменой ключей.  
б) Область применения включает оценку устойчивости криптографических систем к атакам при частой ротации ключей, а также проверку корректности шифрования/расшифровки в сегментированных блоках.

2. Условия применения

г) Ограничения:

* Максимальное количество неудачных попыток аутентификации: **5**;
* Временная блокировка доступа при превышении попыток: **10 секунд**;
* Размер сегмента для смены ключа: **1000 блоков** (16 байт/блок).

д) Минимальная конфигурация технических средств:

* Операционная система: POSIX-совместимая среда;
* Наличие библиотеки **src/omac.h** для реализации режима CTR;
* Компилятор C с поддержкой стандарта C99.

е) Системные функции (СФ):

* Аутентификация пользователя по паролю ("securepass" по умолчанию);
* Логирование событий в файл **crypto\_audit.log**;
* Безопасное обнуление памяти (функция **secure\_zero**);
* Генерация случайных данных через псевдослучайный алгоритм.

3. Описание задачи

в) Класс решаемых задач:

* Шифрование и расшифровка данных в режиме CTR;
* Управление ключами (генерация, сегментированная ротация, очистка);
* Проверка целостности данных после обработки.

Методы решения:

* Алгоритм CTR с использованием внешней функции **CTR\_Crypt**;
* Инкремент вектора инициализации (IV) для каждого сегмента;
* Генерация ключей и данных через **generate\_random\_data**.

4. Входные и выходные данные

* **Входные данные**:
  + Пароль пользователя (вводится интерактивно);
  + Параметры теста: количество блоков (**1 000 000**), размер блока (**16 байт**).
* **Выходные данные**:
  + Результат проверки целостности (Success/Failure);
  + Лог-файл **crypto\_audit.log**;
  + Зашифрованные и расшифрованные данные (в памяти).

5. Порядок работы СКЗИ (пункт ж)

1. Запуск программы:

./program

1. Аутентификация пользователя (пароль "securepass").
2. Проверка блокировки доступа (при превышении попыток).
3. Генерация тестовых данных:
   * Создание **plaintext** размером **16 МБ**;
   * Формирование начального вектора **initial\_iv**.
4. Шифрование и расшифровка:
   * Разделение данных на сегменты по **1000 блоков**;
   * Генерация уникального ключа для каждого сегмента;
   * Инкремент IV для обеспечения уникальности сегментов.
5. Проверка корректности:
   * Сравнение исходных и расшифрованных данных;
   * Вывод результата в консоль.
6. Очистка ресурсов:
   * Безопасное обнуление ключей, IV, буферов;
   * Освобождение памяти.

Пояснительная записка

а) Сведения о назначении компонентов:

1. **Библиотека**omac.h:
   * Реализация алгоритма OMAC (One-Key CBC MAC) для аутентификации сообщений.
2. **Библиотека**files.h:
   * Вспомогательные функции для работы с файлами (не явно используются в коде).
3. **Функция**CTR\_Crypt:
   * Реализация режима шифрования CTR (Counter Mode) для симметричного шифрования.
4. **Основной модуль**:
   * Аутентификация пользователя.
   * Тестирование многосегментного шифрования с периодической сменой ключей.
   * Генерация тестовых данных и проверка целостности шифрования.
   * Управление криптографическими ключами и векторами инициализации (IV).

б) Перечень реализованных функций:

1. authenticate\_user():
   * Проверка пароля "securepass".
2. log\_event(const char \*msg):
   * Запись событий в файл crypto\_audit.log.
3. secure\_zero():
   * Безопасное обнуление памяти.
4. generate\_random\_data():
   * Генерация псевдослучайных данных через rand().
5. increment\_iv():
   * Инкремент 64-битного счетчика в векторе инициализации.
6. main():
   * Управление аутентификацией.
   * Тестирование CTR-шифрования с сегментированной сменой ключей.
   * Проверка корректности расшифровки.

в) Параметры функций:

1. CTR\_Crypt():
   * init\_vec (128-битный вектор инициализации),
   * in\_buf/out\_buf (входной/выходной буферы),
   * key (128-битный ключ),
   * size (размер данных в байтах).
2. generate\_random\_data():
   * buffer (буфер для заполнения),
   * size (требуемый размер данных).
3. increment\_iv():
   * iv (вектор инициализации),
   * increment (шаг инкремента).

г) Коды возврата:

1. authenticate\_user():
   * 1 — успешная аутентификация,
   * 0 — ошибка.
2. main():
   * 0 — успешное завершение,
   * 1 — ошибка аутентификации/выделения памяти.
3. CTR\_Crypt():
   * Неявный код: предполагается успешное выполнение при отсутствии ошибок сегментации.

д) Экспортируемые функции:

1. CTR\_Crypt():
   * Может использоваться в других модулях для CTR-шифрования.
2. secure\_zero():
   * Интеграция в сторонние проекты для безопасной очистки данных.

е) Описание переменных:

1. **Глобальные**:
   * failed\_attempts — счетчик неудачных попыток входа,
   * blocked\_until — время разблокировки доступа.
2. **Локальные в**main():
   * plaintext/ciphertext/decrypted — буферы для тестовых данных,
   * initial\_iv — начальный вектор инициализации,
   * keys — массив сегментных ключей,
   * modes — конфигурация сегментов (1000 блоков на ключ).

ж) Алгоритмы функционирования:

1. **Аутентификация**:
   * Блокировка доступа на 10 секунд после 5 неудачных попыток.
2. **CTR-шифрование**:
   * Разделение данных на сегменты по 1000 блоков (16 КБ).
   * Генерация уникального ключа для каждого сегмента.
   * Инкремент счетчика в IV для каждого сегмента.
   * Последовательное шифрование/дешифрование сегментов.
3. **Проверка корректности**:
   * Сравнение исходных и расшифрованных данных через memcmp().
4. **Безопасность**:
   * Обнуление ключей, IV и буферов через secure\_zero().

и) Критерии и методика тестирования:

1. **Критерии**:
   * Идентичность данных до/после шифрования.
   * Корректная работа инкремента IV для больших объемов данных.
   * Отсутствие утечек памяти при многократной смене ключей.
2. **Методика**:
   * Замер производительности при обработке 1 млн блоков.
   * Тестирование граничных случаев (последний неполный сегмент).
   * Проверка устойчивости к ошибкам (например, неверный размер IV).
3. **Результаты**:
   * Успешное шифрование/дешифрование 16 МБ данных.
   * Корректная работа инкремента IV при переполнении счетчика.
   * Отсутствие остатков ключевых данных в памяти после secure\_zero().