**Модель протокола защищенного соединения**

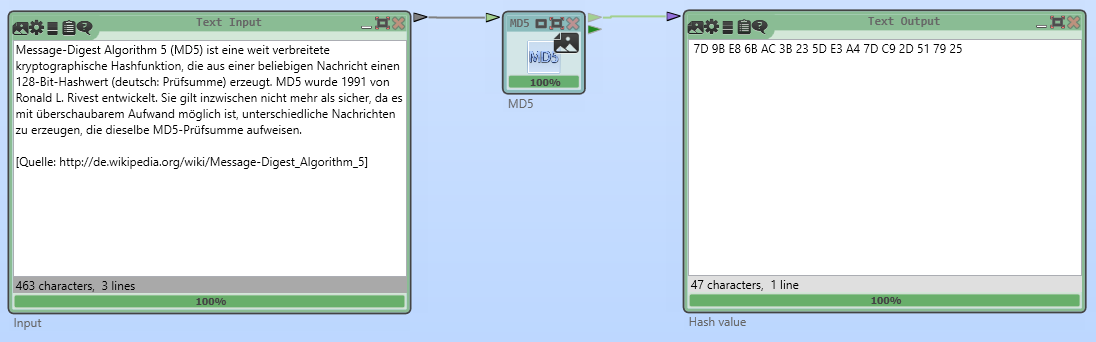
**Цель:** изучить подходы к применению криптопримитивов в рамках протоколов для защищенных соединений.

**Необходимое программное обеспечение:** В рамках задания необходим установленный openssl, который может быть установлен отдельно или как составляющая сборки (например, kali linux)

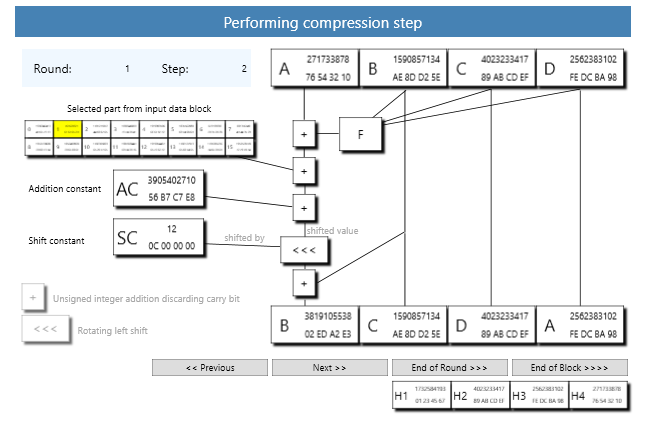
**Ход работы:**

В ходе работы необходимо изучить взаимодействие таких криптопримитивов как алгоритмы симметричного и асимметричного шифрования, имитовставки и электронной подписи. В рамках практического задания предлагается рассмотреть один из способов, как эти криптопримитивы могут взаимодействовать между собой.

1. Выполните визуализацию 1 раунда любого алгоритма хэширования Templates->Hash Functions->MD5 (MD5, SHA-1, SHA-2, SHA-3) или имитовставки. Начальные переменные, раундовые функции определяются согласно стандартам хэширования.



Проанализируйте каждую из операций, входящую в раундовую функцию для хэширования. Отдельно опишите процесс инициализации раундовых констант A,B,C,D, констант для сложения и сдвига 32 битного слова.



1. Отразите в отчете алгоритм для генерации кодов аутентификации сообщений. Путь для шаблонов Cryptool 2 Templates->Hash Functions->HMAC.
2. Протестировать возможности хэширования файлов и сообщений с помощью openssl dgst. Выполнить хэширование как минимум для одного сообщения (задаваемого, например, через echo) и одного файла. Протестировать хэш-алгоритмы –md5, -sha256, -sha512. Проанализируйте длину сгенерированного хэша. Влияет ли на длину хэша длина входного сообщения? Как сильно меняется хэш, если мы незначительно изменим входное сообщение (на 1 бит, на 1 символ)?
3. Генерация Hash based MAC, для для этого воспользуйтесь дополнительным флагом –hmac. После флага –hmac укажите ключ. Сгенерируйте, используя различные флаги и ключи HMAC-MD5, HMAC-SHA256, HMAC-sha512. Проверьте возможность использования ключей различной длины при генерации имитовставки (Возможно ли это? Требуется ли указывать ключ фиксированного размера?).
4. Сгенерируйте пару открытый-закрытый ключ для асимметричной криптосистемы RSA. С помощью данной ключевой пары RSA зашифруйте случайный ключ AES, например, длиной 128 бит. Покажите с помощью каких команд в openssl можно выполнить:

* Генерацию ключевой пары;
* Шифрование симметричного ключа с помощью открытого ключа криптосистемы RSA;
* Дешифрование симметричного ключа;
* Генерацию подписи для зашифрованного сообщения на основе асимметричного алгоритма RSA;
* Проверку подписи RSA.

1. Поскольку для алгоритма симметричного шифрования необходим ключ, то мы должны передать его принимающей стороне. Для этого мы используем асимметричные криптоалгоритмы с электронной цифровой подписью в каждом сообщении. Сообщением для асимметричного криптоалгоритма является ключ симметричной криптосистемы, задаваемый на шаге 1. Выполните генерацию ключей для асимметричного криптоалгоритма (например, RSA или другого), зашифруете сообщение, сгенерируйте подпись и выполните проверку подписи соответствующими командами:

- *openssl genrsa (генерация ключей)*;

- *openssl rsautl –encrypt (шифрование);*

*- openssl dgst и openssl rsautl -sign (для генерации хэш значения, взятого от симметричного ключа, и подписи сгенерированного хэша);*

*-*  *openssl rsautl –verify (для проверки подписи)*.

1. В качестве сообщения может быть указана случайная строка (задаваемая через echo) или файл (example.txt). Продемонстрируете с помощью команд openssl процесс шифрования и дешифрования сообщения с помощью любой симметричной криптосистемы, используя команду **openssl enc** с любыми флагами (т.е. сам алгоритм шифрования, режим шифрования, способ задания ключа и пр. выбираются самостоятельно из набора предоставляемого openssl).
2. Дополнительно с зашифрованным сообщением используем алгоритм имитовставки с тем же ключом шифрования, который использовался при шифровании сообщения. **Команды даны в ПЗ 1.** Генерируем hmac для сообщения из пункта 1. Научитесь сравнивать между собой два разных hmac с помощью флага *-verify* или команды *diff* .

В качестве **альтернативного варианта** можно рассмотреть реализацию хэш-функции или имитовставки. Для лабораторной работы достаточно реализовать одну хэш-функцию или имитовставку. Алгоритмы хэш-функций и имитовставок, основанные на использовании блочных симметричных криптосистем, не рассматриваются в рамках данной лабораторной.

Основные процессы, которые необходимо реализовать в самом алгоритме:

* процесс генерации хэш-суммы
* процесс проверки хэш-значения

Данные процессы будут практически полностью идентичны, однако, каждый из них необходим при использовании реализованного алгоритма на практике.

Примеры алгоритмов хэширования и имитозащиты на выбор:

* MD5
* SHA1
* SHA2
* SHA3(Keccak)
* ГОСТ 34.11
* MAA из ISO 8731
* HMAC. Имитовставки на основе хэшей (MD5, SHA1, SHA2, SHA3, ГОСТ 34.11)

**Вопросы для защиты**

1. Хэш функции. Коллизии 1 и 2 рода
2. Описание внутренней структуры следующих хэш функций: MD5/SHA-1/SHA-2/SHA-3 (keccak)
3. Парадокс дней рождений/атака дней рождений
4. Коды аутентификации сообщений на основе хэш-функций
5. Схема Меркла-Дамгарда. Схема Keccak