Министерство образования Республики Беларусь
Белорусский государственный университет
НАУЧНО-ИССЛЕДОВАТЕЛЬСКИЙ ИНСТИТУТ
ПРИКЛАДНЫХ ПРОБЛЕМ МАТЕМАТИКИ И ИНФОРМАТИКИ

УТВЕРЖДАЮ
Директор НИИ прикладных проблем
математики и информатики

Ю.С.Харин
«___» _____ 2022 г.

МЕТОДИКА ИСПЫТАНИЙ СРЕДСТВ КРИПТОГРАФИЧЕСКОЙ ЗАЩИТЫ ИНФОРМАЦИИ НА СООТВЕТСТВИЕ ТРЕБОВАНИЯМ СТБ 34.101.77-2020

МИ.10177.10.01

Листов 40

Минск 2022

Предисловие

Настоящая методика испытаний предназначена для использования в испытательных лабораториях при проведении сертификационных испытаний средств криптографической защиты информации на соответствие требованиям СТБ 34.101.77-2020 «Информационные технологии и безопасность. Криптографические алгоритмы на основе sponge-функции».

Содержание

1 Нормативные ссылки

В настоящем документе использованы ссылки на следующие стандарты:

ГОСТ 19.202-78 «Единая система программной документации. Спецификация. Требования к содержанию и оформлению».

ГОСТ 19.401-2000 «Единая система программной документации. Текст программы. Требования к содержанию, оформлению и контролю качества».

ГОСТ 19.402-2000 «Единая система программной документации. Описание программы. Требования к содержанию, оформлению и контролю качества».

ГОСТ 19.504-79 «Единая система программной документации. Руководство программиста. Требования к содержанию и оформлению».

СТБ 34.101.31-2020 «Информационные технологии и безопасность. Алгоритмы шифрования и контроля целостности».

СТБ 34.101.77-2020 «Информационные технологии и безопасность. Криптографические алгоритмы на основе sponge-функции».

2 Термины, обозначения и сокращения

В настоящем документе применяются термины и обозначения СТБ 34.101.77, а также следующие сокращения:

ЕСПД единая система программной документации;

СКЗИ средство криптографической защиты информации.

3 Объект и цель испытаний

На испытания представляется средство криптографической защиты информации (СКЗИ), реализующее криптографические алгоритмы СТБ 34.101.77, и документация на СКЗИ.

Целью испытаний является проверка соответствия объекта испытаний требованиям СТБ 34.101.77.

4 Требования к объекту испытаний

К программе объекта испытаний предъявляются следующие требования, подлежащие проверке во время проведения испытаний:

– в программе должны быть точно и полно реализовываны криптографические алгоритмы СТБ 34.101.77, поддерживаемые объектом испытаний;

– программа, реализующая криптографические алгоритмы и требования СТБ 34.101.77, не должна содержать недокументированные возможности.

Документация на объект испытаний должна включать документы «Спецификация», «Текст программы» и может включать документы «Описание программы», «Руководство

программиста» и другие документы. Документация может быть разработана в соответствии с требованиями единой системы программной документации (ЕСПД).

5 Средства и порядок испытаний

5.1 Общие сведения

Испытания программы состоят из трех этапов:

1 Анализ документации.

2 Тестирование программы.

3 Анализ исходных текстов программы.

Выполнение этапа 1 осуществляется экспертами по анализу документации, выполнение этапа 2 — экспертами по тестированию, а выполнение этапа 3 — экспертами по анализу исходных текстов. К проведению испытаний должно быть привлечено не менее двух экспертов по анализу исходных текстов и один или более эксперт по тестированию. К анализу документации должен быть привлечен, по крайней мере, один эксперт по анализу исходных текстов программ.

По результатам выполнения этапа испытаний эксперт оформляет протокол результатов проверок: протокол анализа документации, протокол тестирования, протокол анализа исходных текстов. В протоколе эксперт делает вывод о соответствии (не соответствии) программы требованиям СТБ 34.101.77. Если программа не поддерживает некоторые алгоритмы, определенные в СТБ 34.101.77, то в протоколе делается соответствующее примечание. Примеры оформления протоколов приводятся в приложениях А, Б, В. Допускается оформления протоколов в иной форме, но с обязательным указанием результатов по каждой проводимой проверке и вывода о соответствии (не соответствии).

Если в испытуемой программе используются реализации алгоритмов СТБ 34.101.77, которые в составе других программ имеют действующие сертификаты соответствия требованиям СТБ 34.101.77, то проверки по тестированию и анализу исходных текстов для данных реализаций могут не проводиться. При этом для подтверждения соответствия объекта испытаний требованиям СТБ 34.101.77 экспертом оформляется протокол проверки совпадения контрольных характеристик (хэш-значений) файлов реализации испытуемой программы с контрольными характеристиками соответствующих файлов, указанными в сертификатах соответствия.

На основании протоколов результатов проверок оформляется протокол испытаний, обобщающий результаты испытаний программы. В протоколе испытаний вывод о соответствии программы требованиям СТБ 34.101.77 делается тогда и только тогда, когда вывод о соответствии содержится во всех протоколах результатов проверок. Оформление протокола испытаний проводится в соответствии с требованиями технических нормативно-правовых актов в области сертификации продукции, а также документации, применяемой в испытательной лаборатории.

5.2 Анализ документации

Эксперт проводит анализ документации путем проверки соответствия документации программе объекта испытаний. Такой анализ состоит в получении экспертных заключений, касающихся проверки следующих документов:

– спецификация (см. п. 6.1.1);

– текст программы (см. п. 6.1.2);

– описание программы (см. п. 6.1.3);
– руководство программиста (см. п. 6.1.4).

Анализ документов «Описание программы» и «Руководство программиста» производится в случае их наличия.

5.3 Тестирование

Эксперт проводит тестирование путем выполнения испытуемой программы для некоторого набора проверочных входных значений и сравнения полученных результатов с истинными. Истинные результаты, используемые при тестировании, формируются с помощью эталонной реализации.

Эталонной считается реализация, которая ранее успешно прошла сертификационные испытания на соответствие СТБ 34.101.77 или которая удовлетворяет следующим условиям:

1 Проведен анализ исходных текстов программ эталонной реализации. К анализу привлекались, по меньшей мере, два независимых эксперта. Использовалась методика анализа исходных текстов, определенная в п. 6.3.

2 Проведено тестирование эталонной реализации. При тестировании использовались две другие независимые реализации. Использовались тесты, определенные в п. 6.2, а также тестовые примеры СТБ 34.101.77.

Тестированию подлежат криптографические алгоритмы, реализованные в программе и определенные в СТБ 34.101.77, включая:

– алгоритмы хэширования (см. п. 6.2.1);
– программируемые алгоритмы хэширования (см. п. 6.2.2);
– программируемые алгоритмы аутентифицированного шифрования (см. п. 6.2.3).

Если программа не реализует некоторые из алгоритмов, определенных в СТБ 34.101.77, то тесты для них не выполняются.

Для организации тестирования в исходные тексты программы допускается вносить изменения и дополнения, касающиеся:

– способа чтения входных данных;
– способа записи выходных данных.

При внесении модификаций в исходные тексты должен быть проведен анализ корректности внесенных изменений.

При успешном выполнении тест возвращает признак `УСПЕХ`, иначе — `ОШИБКА`. Если при тестировании программы для некоторых входных значений получены результаты отличные от истинных значений, то эксперт по тестированию должен указать эти входные значения программы и результат ее работы, а также, по требованию, результаты промежуточных вычислений экспертам по анализу исходных текстов.

5.4 Анализ исходных текстов

Эксперт проводит анализ исходных текстов путем проверки корректности реализации в испытуемой программе криптографических алгоритмов СТБ 34.101.77. Такой анализ состоит в получении экспертных заключений, касающихся:

– корректности использования локальных переменных (см. п. 6.3.1);
– корректности использования глобальных переменных (см. п. 6.3.2);
– корректности использования констант (см. п. 6.3.3);

– корректности программной логики функций программы (см. п. 6.3.4);
– корректности вызова стандартных функций (см. п. 6.3.5);
– корректности вызова функций программы (см. п. 6.3.6);
– корректности обработки исключительных ситуаций (см. п. 6.3.7);
– корректности реализации криптографических примитивов (см. п. 6.3.8);
– корректности реализации криптографических алгоритмов (см. п. 6.3.9);
– корректности управления секретными данными (см. п. 6.3.10);
– отсутствия недокументированных возможностей (см. п. 6.3.11).

6 Методы испытаний

6.1 Анализ документации

6.1.1 Документ «Спецификация»

При анализе документа «Спецификация» эксперт проверяет, что в нем указаны компоненты и документация, представляемые на испытания.

Если документ «Спецификация» разработан в соответствии с требованиями ЕСПД, то эксперт проверяет, что содержание и оформление документа соответствует ГОСТ 19.202.

6.1.2 Документ «Текст программы»

При анализе документа «Текст программы» эксперт проверяет, что исходные тексты программы, реализующие определенные в СТБ 34.101.77 криптографические алгоритмы, представлены полностью и в виде, который использовался при сборке программы.

Если документ «Текст программы» разработан в соответствии с требованиями ЕСПД, то эксперт проверяет, что содержание и оформление документа соответствует ГОСТ 19.401.

6.1.3 Документ «Описание программы»

При анализе документа «Описание программы» эксперт проверяет выполнение следующих требований:

– в документе должна быть указана информация, однозначно идентифицирующая вызываемые стандартные функции (версия компилятора, используемые стандартные библиотеки и т.п.);

– документ должен определять программные модули, реализующие определенные в СТБ 34.101.77 криптографические алгоритмы;

– описание программы в терминах программных модулей должно соответствовать исходным текстам программы.

Если документ «Описание программы» разработан в соответствии с требованиями ЕСПД, то эксперт проверяет, что содержание и оформление документа соответствует ГОСТ 19.402.

6.1.4 Документ «Руководство программиста»

При анализе документа «Руководство программиста» эксперт проверяет выполнение следующих требований:

– документ должен содержать описание всех доступных для вызова функций, реализующих определенные в СТБ 34.101.77 криптографические алгоритмы;

– описание функций, реализующих определенные в СТБ 34.101.77 криптографические алгоритмы, и условия их использования должны соответствовать исходным текстам программы.

При описании в документации функций должны выполняться следующие условия:

– каждая функция должна иметь описание назначения;

– каждый параметр функции должен иметь описание назначения, типа и, при необходимости, диапазона допустимых значений;

– каждая функция должна иметь описание возвращаемого результата с указанием типа;

– каждая функция должна иметь описание условий, при выполнении которых в ходе работы функции могут возникать ошибочные ситуации, требующие специальной обработки;

– в случае если при реализации криптографического алгоритма используется более одной доступной для вызова функции, должны быть указаны порядок и условия вызова данных функций.

Если документ «Руководство программиста» разработан в соответствии с требованиями ЕСПД, то эксперт проверяет, что содержание и оформление документа соответствует ГОСТ 19.504.

6.2 Тестирование

6.2.1 Алгоритмы хэширования

При тестировании реализаций алгоритмов хэширования выполняются тесты BASH.HASH128.1 – BASH.HASH128.5, BASH.HASH192.1 – BASH.HASH192.5, BASH.HASH256.1 – BASH.HASH256.5.

Входными данными тестов являются уровень стойкости $l \in \{128, 192, 256\}$ и сообщение $X \in \{0,1\}^*$.

В тестах для хранения результата хэширования используются слова $Y, Y' \in \{0,1\}^{2l}$.

Тест BASH.HASH128.1

1 Задать уровень стойкости: $l \leftarrow 128$.
2 Задать сообщение: $X \leftarrow \perp$.
3 Испытуемой реализацией вычислить хэш-значение X и сохранить результат в Y.
4 Если

$$Y = \begin{array}{l}\texttt{114C3DFAE373D9BC BC3602D6386F2D6A}\\ \texttt{2059BA1BF9048DBA A5146A6CB775709D}_{16},\end{array}$$

то возвратить `УСПЕХ`, иначе — `ОШИБКА`.

Тест BASH.HASH128.2

1 Задать уровень стойкости: $l \leftarrow 128$.

2 Задать сообщение длины 127 октета:

$$X \leftarrow \begin{array}{l}\texttt{B194BAC80A08F53B 366D008E584A5DE4}\\ \texttt{8504FA9D1BB6C7AC 252E72C202FDCE0D}\\ \texttt{5BE3D61217B96181 FE6786AD716B890B}\\ \texttt{5CB0C0FF33C356B8 35C405AED8E07F99}\\ \texttt{E12BDC1AE28257EC 703FCCF095EE8DF1}\\ \texttt{C1AB76389FE678CA F7C6F860D5BB9C4F}\\ \texttt{F33C657B637C306A DD4EA7799EB23D31}\\ \texttt{3E98B56E27D3BCCF 591E181F4C5AB7}_{16}.\end{array}$$

3 Испытуемой реализацией вычислить хэш-значение X и сохранить результат в Y.

4 Если

$$Y = \begin{array}{l}\texttt{3D7F4EFA00E9BA33 FEED259986567DCF}\\ \texttt{5C6D12D51057A968 F14F06CC0F905961}_{16},\end{array}$$

то возвратить `УСПЕХ`, иначе — `ОШИБКА`.

Тест BASH.HASH128.3

1 Задать уровень стойкости: $l \leftarrow 128$.

2 Задать сообщение длины 54 октета:

$$X \leftarrow \begin{array}{l}\texttt{466966747920666F 7572206279746520}\\ \texttt{6F7220666F757220 68756E6472656420}\\ \texttt{7468697274792074 776F20626974206D}\\ \texttt{657373616765}_{16}.\end{array}$$

3 Испытуемой реализацией вычислить хэш-значение X и сохранить результат в Y.

4 Если

$$Y = \begin{array}{l}\texttt{8F866380A7714B53 9DBC9F3D18020BCA}\\ \texttt{EDBD428AECC69F14 05699BE12C19ED02}_{16},\end{array}$$

то возвратить `УСПЕХ`, иначе — `ОШИБКА`.

Тест BASH.HASH128.4

1 Задать уровень стойкости: $l \leftarrow 128$.

2 Задать сообщение X, состоящее из 1000000 октетов $\texttt{61}_{16}$ (шестнадцатеричное представление символа ‘a’ в коде ASCII).

3 Испытуемой реализацией вычислить хэш-значение X и сохранить результат в Y.

4 Если

$$Y = \begin{array}{l}\texttt{F8812E85897421D4 8D56C600D8F76F24}\\ \texttt{F2E0FFC6F879DB8F 3279B4F6C53529FA}_{16},\end{array}$$

то возвратить `УСПЕХ`, иначе — `ОШИБКА`.

Тест BASH.HASH128.5

1 Задать уровень стойкости: $l \leftarrow 128$.

2 Для $i = 1, 2, \ldots, 10000$ выполнить:

1) псевдослучайным методом сгенерировать сообщения X длины 2048 октета;

2) испытуемой реализацией вычислить хэш-значение X и сохранить результат в Y;
3) эталонной реализацией вычислить хэш-значение X и сохранить результат в Y';
4) если $Y \neq Y'$, то возвратить `ОШИБКА`.

3 Возвратить `УСПЕХ`.

Тест BASH.HASH192.1

1 Задать уровень стойкости: $l \leftarrow 192$.
2 Задать сообщение: $X \leftarrow \perp$.
3 Испытуемой реализацией вычислить хэш-значение X и сохранить результат в Y.
4 Если

$$Y = \begin{array}{l}\texttt{296F63CDDF8E4963 A657A8861FAD1D9D}\\ \texttt{75BAF67E747B7E00 AF8E55BAB9E2627B}\\ \texttt{B0E2B752D867E70B EB88D13D495A4ECB}_{16},\end{array}$$

то возвратить `УСПЕХ`, иначе — `ОШИБКА`.

Тест BASH.HASH192.2

1 Задать уровень стойкости: $l \leftarrow 192$.
2 Задать сообщение длины 95 октета:

$$X \leftarrow \begin{array}{l}\texttt{B194BAC80A08F53B 366D008E584A5DE4}\\ \texttt{8504FA9D1BB6C7AC 252E72C202FDCE0D}\\ \texttt{5BE3D61217B96181 FE6786AD716B890B}\\ \texttt{5CB0C0FF33C356B8 35C405AED8E07F99}\\ \texttt{E12BDC1AE28257EC 703FCCF095EE8DF1}\\ \texttt{C1AB76389FE678CA F7C6F860D5BB9C}_{16}.\end{array}$$

3 Испытуемой реализацией вычислить хэш-значение X и сохранить результат в Y.
4 Если

$$Y = \begin{array}{l}\texttt{64334AF830D33F63 E9ACDFA184E32522}\\ \texttt{103FFF5C6860110A 2CD369EDBC04387C}\\ \texttt{501D8F92F749AE4D E15A8305C353D64D}_{16},\end{array}$$

то возвратить `УСПЕХ`, иначе — `ОШИБКА`.

Тест BASH.HASH192.3

1 Задать уровень стойкости: $l \leftarrow 192$.
2 Задать сообщение длины 54 октета:

$$X \leftarrow \begin{array}{l}\texttt{466966747920666F 7572206279746520}\\ \texttt{6F7220666F757220 68756E6472656420}\\ \texttt{7468697274792074 776F20626974206D}\\ \texttt{657373616765}_{16}.\end{array}$$

3 Испытуемой реализацией вычислить хэш-значение X и сохранить результат в Y.
4 Если

$$Y = \begin{array}{l}\texttt{0CC67F5E0D51D7D1 74146E85393C171F}\\ \texttt{3E7B76456589653E D19025C4B2A69601}\\ \texttt{B685F4EC7D8CCB6E E1E7EC8793A82D55}_{16},\end{array}$$

то возвратить `УСПЕХ`, иначе — `ОШИБКА`.

Тест BASH.HASH192.4

1 Задать уровень стойкости: $l \leftarrow 192$.
2 Задать сообщение X, состоящее из 1000000 октетов $\texttt{61}_{16}$ (шестнадцатеричное представление символа ‘a’ в коде ASCII).
3 Испытуемой реализацией вычислить хэш-значение X и сохранить результат в Y.
4 Если

$$Y = \begin{array}{l}\texttt{FE74AC72B33B7306 498393E898D1CAED}\\ \texttt{783276083EA3052F 7897BF9B681B8DC1}\\ \texttt{A9112418133C0165 79F93B22DBAF2977}_{16},\end{array}$$

то возвратить `УСПЕХ`, иначе — `ОШИБКА`.

Тест BASH.HASH192.5

1 Задать уровень стойкости: $l \leftarrow 192$.
2 Для $i = 1, 2, \ldots, 10000$ выполнить:
1) псевдослучайным методом сгенерировать сообщения X длины 2048 октета;
2) испытуемой реализацией вычислить хэш-значение X и сохранить результат в Y;
3) эталонной реализацией вычислить хэш-значение X и сохранить результат в Y';
4) если $Y \neq Y'$, то возвратить `ОШИБКА`.
3 Возвратить `УСПЕХ`.

Тест BASH.HASH256.1

1 Задать уровень стойкости: $l \leftarrow 256$.
2 Задать сообщение: $X \leftarrow \perp$.
3 Испытуемой реализацией вычислить хэш-значение X и сохранить результат в Y.
4 Если

$$Y = \begin{array}{l}\texttt{D3A5F9B655CE3EFC 1C3E6F2FA1E10F39}\\ \texttt{EC1C3950462097CE D1130814868E49E3}\\ \texttt{887581066CD78A97 B6685A410E239D12}\\ \texttt{A357FAFF1B252D63 10AA1F95FD4A0283}_{16},\end{array}$$

то возвратить `УСПЕХ`, иначе — `ОШИБКА`.

Тест BASH.HASH256.2

1 Задать уровень стойкости: $l \leftarrow 256$.
2 Задать сообщение длины 63 октета:

$$X \leftarrow \begin{array}{l}\texttt{B194BAC80A08F53B 366D008E584A5DE4}\\ \texttt{8504FA9D1BB6C7AC 252E72C202FDCE0D}\\ \texttt{5BE3D61217B96181 FE6786AD716B890B}\\ \texttt{5CB0C0FF33C356B8 35C405AED8E07F}_{16}.\end{array}$$

3 Испытуемой реализацией вычислить хэш-значение X и сохранить результат в Y.

4 Если

$$Y = \begin{array}{l}\texttt{2A66C87C189C12E2 5523940612 3BDEDB}\\ \texttt{F19955EAF0808B2A D705E249220845E2}\\ \texttt{0F4786FB6765D0B5 C48984B1B16556EF}\\ \texttt{19EA8192B985E423 3D9C09508D6339E7}_{16},\end{array}$$

то возвратить `УСПЕХ`, иначе — `ОШИБКА`.

Тест BASH.HASH256.3

1 Задать уровень стойкости: $l \leftarrow 256$.

2 Задать сообщение длины 54 октета:

$$X \leftarrow \begin{array}{l}\texttt{466966747920666F 7572206279746520}\\ \texttt{6F7220666F757220 68756E6472656420}\\ \texttt{7468697274792074 776F20626974206D}\\ \texttt{657373616765}_{16}.\end{array}$$

3 Испытуемой реализацией вычислить хэш-значение X и сохранить результат в Y.

4 Если

$$Y = \begin{array}{l}\texttt{38FBE3C7A8F85A64 16E876EF884F6888}\\ \texttt{6336E5214BC189D5 B30079211861B4C8}\\ \texttt{846012005C431631 3B31CB0B1FB5011E}\\ \texttt{1FDADF7E48061283 BCCBD67DE9131DDA}_{16},\end{array}$$

то возвратить `УСПЕХ`, иначе — `ОШИБКА`.

Тест BASH.HASH256.4

1 Задать уровень стойкости: $l \leftarrow 256$.

2 Задать сообщение X, состоящее из 1000000 октетов $\texttt{61}_{16}$ (шестнадцатеричное представление символа ‘a’ в коде ASCII).

3 Испытуемой реализацией вычислить хэш-значение X и сохранить результат в Y.

4 Если

$$Y = \begin{array}{l}\texttt{E3E32CD6E7AB56FD 4BB7D654B93C8325}\\ \texttt{DD7F130ABB99B3B8 DC8AC2BF604D51D0}\\ \texttt{7DD94DB483451D64 33739AE775E4DDF3}\\ \texttt{5154E70E1812A4E0 6EE46E6F02323E41}_{16},\end{array}$$

то возвратить `УСПЕХ`, иначе — `ОШИБКА`.

Тест BASH.HASH256.5

1 Задать уровень стойкости: $l \leftarrow 256$.

2 Для $i = 1, 2, \ldots, 10000$ выполнить:

1) псевдослучайным методом сгенерировать сообщения X длины 2048 октета;
2) испытуемой реализацией вычислить хэш-значение X и сохранить результат в Y;
3) эталонной реализацией вычислить хэш-значение X и сохранить результат в Y';
4) если $Y \neq Y'$, то возвратить `ОШИБКА`.

3 Возвратить `УСПЕХ`.

6.2.2 Программируемые алгоритмы хэширования

При тестировании реализаций программируемых алгоритмов хэширования выполняются тесты BASH.PRGHASH128.1 – BASH.PRGHASH128.6, BASH.PRGHASH192.1 – BASH.PRGHASH192.6, BASH.PRGHASH256.1 – BASH.PRGHASH256.6.

Входными данными тестов являются уровень стойкости $l \in \{128, 192, 256\}$, емкость $d \in \{1, 2\}$, анонс $A \in \{0,1\}^{32*}$, хэшируемое сообщение $X \in \{0,1\}^*$. Во всех тестах используется длина хэш-значения $2l$.

В тестах для хранения результата хэширования используются слова $Y, Y' \in \{0,1\}^{2l}$.

Тест BASH.PRG.HASH128.1

1 Задать уровень стойкости: $l \leftarrow 128$.
2 Задать емкость: $d \leftarrow 2$.
3 Задать анонс: $A \leftarrow \perp$.
4 Задать сообщение: $X \leftarrow \perp$.
5 Испытуемой реализацией вычислить хэш-значение X и сохранить результат в Y.
6 Если

$$Y = \texttt{36FA075EC15721F2 50B9A641A8CB99A3}\\ \texttt{33A9EE7BA8586D06 46CBAC3686C03DF3}_{16},$$

то возвратить `УСПЕХ`, иначе — `ОШИБКА`.

Тест BASH.PRG.HASH128.2

1 Задать уровень стойкости: $l \leftarrow 128$.
2 Задать емкость: $d \leftarrow 2$.
3 Задать анонс: $A \leftarrow \perp$.
4 Задать сообщение длины 127 октета:

$$X \leftarrow \texttt{B194BAC80A08F53B 366D008E584A5DE4}\\ \texttt{8504FA9D1BB6C7AC 252E72C202FDCE0D}\\ \texttt{5BE3D61217B96181 FE6786AD716B890B}\\ \texttt{5CB0C0FF33C356B8 35C405AED8E07F99}\\ \texttt{E12BDC1AE28257EC 703FCCF095EE8DF1}\\ \texttt{C1AB76389FE678CA F7C6F860D5BB9C4F}\\ \texttt{F33C657B637C306A DD4EA7799EB23D31}\\ \texttt{3E98B56E27D3BCCF 591E181F4C5AB7}_{16}.$$

5 Испытуемой реализацией вычислить хэш-значение X и сохранить результат в Y.
6 Если

$$Y = \texttt{C930FF427307420D A6E4182969AA1FFC}\\ \texttt{3310179B8A0EDB3E 20BEC285B568BA17}_{16},$$

то возвратить `УСПЕХ`, иначе — `ОШИБКА`.

Тест BASH.PRG.HASH128.3

1 Задать уровень стойкости: $l \leftarrow 128$.
2 Задать емкость: $d \leftarrow 1$.
3 Задать анонс: $A \leftarrow \perp$.

4 Задать сообщение длины 143 октета:

$$X \leftarrow \begin{array}{l}\texttt{B194BAC80A08F53B 366D008E584A5DE4}\\ \texttt{8504FA9D1BB6C7AC 252E72C202FDCE0D}\\ \texttt{5BE3D61217B96181 FE6786AD716B890B}\\ \texttt{5CB0C0FF33C356B8 35C405AED8E07F99}\\ \texttt{E12BDC1AE28257EC 703FCCF095EE8DF1}\\ \texttt{C1AB76389FE678CA F7C6F860D5BB9C4F}\\ \texttt{F33C657B637C306A DD4EA7799EB23D31}\\ \texttt{3E98B56E27D3BCCF 591E181F4C5AB793}\\ \texttt{E9DEE72C8F0C0FA6 2DDB49F46F7396}_{16}.\end{array}$$

5 Испытуемой реализацией вычислить хэш-значение X и сохранить результат в Y.

6 Если

$$Y = \begin{array}{l}\texttt{D451F9126ADC0467 0441BE01D589809B}\\ \texttt{9717FCF77EE5339E 8318D2CE86A9C390}_{16},\end{array}$$

то возвратить `УСПЕХ`, иначе — `ОШИБКА`.

Тест BASH.PRG.HASH128.4

1 Задать уровень стойкости: $l \leftarrow 128$.

2 Задать емкость: $d \leftarrow 1$.

3 Задать анонс длины 4 октета:

$$A \leftarrow \texttt{06075316}_{16}.$$

4 Задать сообщение длины 54 октета:

$$X \leftarrow \begin{array}{l}\texttt{466966747920666F 7572206279746520}\\ \texttt{6F7220666F757220 68756E6472656420}\\ \texttt{7468697274792074 776F20626974206D}\\ \texttt{657373616765}_{16}.\end{array}$$

5 Испытуемой реализацией вычислить хэш-значение X и сохранить результат в Y.

6 Если

$$Y = \begin{array}{l}\texttt{B8C6742CAE437564 074564BD3B99112E}\\ \texttt{8399CE81E66E7B9A FF7B59ECE5BA2009}_{16},\end{array}$$

то возвратить `УСПЕХ`, иначе — `ОШИБКА`.

Тест BASH.PRG.HASH128.5

1 Задать уровень стойкости: $l \leftarrow 128$.

2 Задать емкость: $d \leftarrow 2$.

3 Задать анонс: $A \leftarrow \perp$.

4 Задать сообщение X, состоящее из 1000000 октетов $\texttt{61}_{16}$ (шестнадцатеричное представление символа ‘a’ в коде ASCII).

5 Испытуемой реализацией вычислить хэш-значение X и сохранить результат в Y.

6 Если

$$Y = \begin{array}{l}\texttt{FA831D7F160A5C58 9F0D0045D4EB4215}\\ \texttt{501075CBFE851CAC 1F45002D0C2BD853}_{16},\end{array}$$

то возвратить УСПЕХ, иначе — ОШИБКА.

Тест BASH.PRG.HASH128.6

1 Задать уровень стойкости: $l \leftarrow 128$.
2 Задать емкость: $d \leftarrow 1$.
3 Задать анонс: $A \leftarrow \perp$.
4 Для $i = 1, 2, \ldots, 10000$ выполнить:
1) псевдослучайным методом сгенерировать сообщения X длины 2048 октета;
2) испытуемой реализацией вычислить хэш-значение X и сохранить результат в Y;
3) эталонной реализацией вычислить хэш-значение X и сохранить результат в Y';
4) если $Y \neq Y'$, то возвратить ОШИБКА.
5 Возвратить УСПЕХ.

Тест BASH.PRG.HASH192.1

1 Задать уровень стойкости: $l \leftarrow 192$.
2 Задать емкость: $d \leftarrow 2$.
3 Задать анонс: $A \leftarrow \perp$.
4 Задать сообщение: $X \leftarrow \perp$.
5 Испытуемой реализацией вычислить хэш-значение X и сохранить результат в Y.
6 Если

$$Y = \begin{array}{l}\texttt{39E58343BA9E2B94 D70D447425CE8AFE}\\ \texttt{B6E019033583B2A0 13687D275E051598}\\ \texttt{6E8408DA9E0C0C08 02111209AA9E1E70}_{16},\end{array}$$

то возвратить УСПЕХ, иначе — ОШИБКА.

Тест BASH.PRG.HASH192.2

1 Задать уровень стойкости: $l \leftarrow 192$.
2 Задать емкость: $d \leftarrow 2$.
3 Задать анонс: $A \leftarrow \perp$.
4 Задать сообщение длины 127 октета:

$$X \leftarrow \begin{array}{l}\texttt{B194BAC80A08F53B 366D008E584A5DE4}\\ \texttt{8504FA9D1BB6C7AC 252E72C202FDCE0D}\\ \texttt{5BE3D61217B96181 FE6786AD716B890B}\\ \texttt{5CB0C0FF33C356B8 35C405AED8E07F99}\\ \texttt{E12BDC1AE28257EC 703FCCF095EE8DF1}\\ \texttt{C1AB76389FE678CA F7C6F860D5BB9C4F}\\ \texttt{F33C657B637C306A DD4EA7799EB23D31}\\ \texttt{3E98B56E27D3BCCF 591E181F4C5AB7}_{16}.\end{array}$$

5 Испытуемой реализацией вычислить хэш-значение X и сохранить результат в Y.

6 Если

$$Y = \begin{array}{l}\texttt{40091E6FE851F45A F77AF952FD42E1B5}\\ \texttt{860B5BB3DE6A8448 2D2EF98642ED0788}\\ \texttt{CB3629472CDCBE2D 3188074515E08B58}_{16},\end{array}$$

то возвратить УСПЕХ, иначе — ОШИБКА.

Тест BASH.PRG.HASH192.3

1 Задать уровень стойкости: $l \leftarrow 192$.

2 Задать емкость: $d \leftarrow 1$.

3 Задать анонс: $A \leftarrow \perp$.

4 Задать сообщение длины 143 октета:

$$X \leftarrow \begin{array}{l}\texttt{B194BAC80A08F53B 366D008E584A5DE4}\\ \texttt{8504FA9D1BB6C7AC 252E72C202FDCE0D}\\ \texttt{5BE3D61217B96181 FE6786AD716B890B}\\ \texttt{5CB0C0FF33C356B8 35C405AED8E07F99}\\ \texttt{E12BDC1AE28257EC 703FCCF095EE8DF1}\\ \texttt{C1AB76389FE678CA F7C6F860D5BB9C4F}\\ \texttt{F33C657B637C306A DD4EA7799EB23D31}\\ \texttt{3E98B56E27D3BCCF 591E181F4C5AB793}\\ \texttt{E9DEE72C8F0C0FA6 2DDB49F46F7396}_{16}.\end{array}$$

5 Испытуемой реализацией вычислить хэш-значение X и сохранить результат в Y.

6 Если

$$Y = \begin{array}{l}\texttt{6166032D6713D401 A6BC687CCFFF2E60}\\ \texttt{3287143A84C78D2C 62C71551E0E2FB2A}\\ \texttt{F6B799EE33B5DECD 7F62F190B1FBB052}_{16},\end{array}$$

то возвратить УСПЕХ, иначе — ОШИБКА.

Тест BASH.PRG.HASH192.4

1 Задать уровень стойкости: $l \leftarrow 192$.

2 Задать емкость: $d \leftarrow 1$.

3 Задать анонс длины 4 октета:

$$A \leftarrow \texttt{06075316}_{16}.$$

4 Задать сообщение длины 54 октета:

$$X \leftarrow \begin{array}{l}\texttt{466966747920666F 7572206279746520}\\ \texttt{6F7220666F757220 68756E6472656420}\\ \texttt{7468697274792074 776F20626974206D}\\ \texttt{657373616765}_{16}.\end{array}$$

5 Испытуемой реализацией вычислить хэш-значение X и сохранить результат в Y.

6 Если

$$Y = \begin{array}{l}\texttt{703DFC9C79A70E39 F75EFC8A9E54AAE6}\\ \texttt{604F21C928A335C4 B5E1B1BC6C252F8B}\\ \texttt{F677302BF6702C19 BC9832E3A237113E}_{16},\end{array}$$

то возвратить УСПЕХ, иначе — ОШИБКА.

Тест BASH.PRG.HASH192.5

1 Задать уровень стойкости: $l \leftarrow 192$.
2 Задать емкость: $d \leftarrow 2$.
3 Задать анонс: $A \leftarrow \perp$.
4 Задать сообщение X, состоящее из 1000000 октетов $\texttt{61}_{16}$ (шестнадцатеричное представление символа ‘a’ в коде ASCII).
5 Испытуемой реализацией вычислить хэш-значение X и сохранить результат в Y.
6 Если

$$Y = \begin{array}{l}\texttt{BED0FE6AF1C639CB DD8DFC57AACA8B35}\\ \texttt{8AA405ADDEAE3C23 CF48356A08F36100}\\ \texttt{908854D326325F05 C901A03D06E65887}_{16},\end{array}$$

то возвратить `УСПЕХ`, иначе — `ОШИБКА`.

Тест BASH.PRG.HASH192.6

1 Задать уровень стойкости: $l \leftarrow 192$.
2 Задать емкость: $d \leftarrow 1$.
3 Задать анонс: $A \leftarrow \perp$.
4 Для $i = 1, 2, \ldots, 10000$ выполнить:
1) псевдослучайным методом сгенерировать сообщения X длины 2048 октета;
2) испытуемой реализацией вычислить хэш-значение X и сохранить результат в Y;
3) эталонной реализацией вычислить хэш-значение X и сохранить результат в Y';
4) если $Y \neq Y'$, то возвратить `ОШИБКА`.
5 Возвратить `УСПЕХ`.

Тест BASH.PRG.HASH256.1

1 Задать уровень стойкости: $l \leftarrow 256$.
2 Задать емкость: $d \leftarrow 2$.
3 Задать анонс: $A \leftarrow \perp$.
4 Задать сообщение: $X \leftarrow \perp$.
5 Испытуемой реализацией вычислить хэш-значение X и сохранить результат в Y.
6 Если

$$Y = \begin{array}{l}\texttt{5FBDF97C622FBB36 F5EE30639C03EF26}\\ \texttt{00105578613401D8 C0877BF38E726845}\\ \texttt{E153094492D13865 CD9527E6B42E2FFA}\\ \texttt{590EE19DB971C327 49A3386D3EC5263B}_{16},\end{array}$$

то возвратить `УСПЕХ`, иначе — `ОШИБКА`.

Тест BASH.PRG.HASH256.2

1 Задать уровень стойкости: $l \leftarrow 256$.
2 Задать емкость: $d \leftarrow 2$.
3 Задать анонс: $A \leftarrow \perp$.

4 Задать сообщение длины 127 октета:

$$X \leftarrow \begin{array}{l}\texttt{B194BAC80A08F53B 366D008E584A5DE4}\\ \texttt{8504FA9D1BB6C7AC 252E72C202FDCE0D}\\ \texttt{5BE3D61217B96181 FE6786AD716B890B}\\ \texttt{5CB0C0FF33C356B8 35C405AED8E07F99}\\ \texttt{E12BDC1AE28257EC 703FCCF095EE8DF1}\\ \texttt{C1AB76389FE678CA F7C6F860D5BB9C4F}\\ \texttt{F33C657B637C306A DD4EA7799EB23D31}\\ \texttt{3E98B56E27D3BCCF 591E181F4C5AB7}_{16}.\end{array}$$

5 Испытуемой реализацией вычислить хэш-значение X и сохранить результат в Y.

6 Если

$$Y = \begin{array}{l}\texttt{6326B650058B5954 C799BF76810A0E6E}\\ \texttt{1DD0900BFA04C2D5 27CC64CD7CE208BA}\\ \texttt{4C7893EAACDC503E F27A5BB052EB1011}\\ \texttt{6E29BA80BCA023FB 28C7808411411865}_{16},\end{array}$$

то возвратить `УСПЕХ`, иначе — `ОШИБКА`.

Тест BASH.PRG.HASH256.3

1 Задать уровень стойкости: $l \leftarrow 256$.

2 Задать емкость: $d \leftarrow 1$.

3 Задать анонс: $A \leftarrow \perp$.

4 Задать сообщение длины 143 октета:

$$X \leftarrow \begin{array}{l}\texttt{B194BAC80A08F53B 366D008E584A5DE4}\\ \texttt{8504FA9D1BB6C7AC 252E72C202FDCE0D}\\ \texttt{5BE3D61217B96181 FE6786AD716B890B}\\ \texttt{5CB0C0FF33C356B8 35C405AED8E07F99}\\ \texttt{E12BDC1AE28257EC 703FCCF095EE8DF1}\\ \texttt{C1AB76389FE678CA F7C6F860D5BB9C4F}\\ \texttt{F33C657B637C306A DD4EA7799EB23D31}\\ \texttt{3E98B56E27D3BCCF 591E181F4C5AB793}\\ \texttt{E9DEE72C8F0C0FA6 2DDB49F46F7396}_{16}.\end{array}$$

5 Испытуемой реализацией вычислить хэш-значение X и сохранить результат в Y.

6 Если

$$Y = \begin{array}{l}\texttt{9E5CBDD50DDC71E3 7482784AAB3BBC1D}\\ \texttt{5E7DC3DF4D201864 2BB98A9583BCF288}\\ \texttt{3B06CCDF5D201C86 CC0564660614CE3C}\\ \texttt{ED90E53C4B2B87B2 3C0B60994FEFE9D6}_{16},\end{array}$$

то возвратить `УСПЕХ`, иначе — `ОШИБКА`.

Тест BASH.PRG.HASH256.4

1 Задать уровень стойкости: $l \leftarrow 256$.
2 Задать емкость: $d \leftarrow 1$.
3 Задать анонс длины 4 октета:

$$A \leftarrow \texttt{06075316}_{16}.$$

4 Задать сообщение длины 54 октета:

$$X \leftarrow \begin{array}{l}\texttt{466966747920666F 7572206279746520}\\ \texttt{6F7220666F757220 68756E6472656420}\\ \texttt{7468697274792074 776F20626974206D}\\ \texttt{657373616765}_{16}.\end{array}$$

5 Испытуемой реализацией вычислить хэш-значение X и сохранить результат в Y.
6 Если

$$Y = \begin{array}{l}\texttt{8F0B8A3A11505F7C DFD7EE033434B847}\\ \texttt{78BA638561D7AB47 568C868AE9FC6496}\\ \texttt{A5E74E8B3C8576F1 2C128181DB4DBDF9}\\ \texttt{15B8D5CE87F8D71F 86DC4A55D7E7DC3B}_{16},\end{array}$$

то возвратить `УСПЕХ`, иначе — `ОШИБКА`.

Тест BASH.PRG.HASH256.5

1 Задать уровень стойкости: $l \leftarrow 256$.
2 Задать емкость: $d \leftarrow 2$.
3 Задать анонс: $A \leftarrow \perp$.
4 Задать сообщение X, состоящее из 1000000 октетов $\texttt{61}_{16}$ (шестнадцатеричное представление символа ‘a’ в коде ASCII).
5 Испытуемой реализацией вычислить хэш-значение X и сохранить результат в Y.
6 Если

$$Y = \begin{array}{l}\texttt{0B38A4135D49C8C9 A2F4145457EFD1D8}\\ \texttt{C1F3D9CEEB3C0AC8 9F5C4331EA6EEBEC}\\ \texttt{7B3FDAB02CA5A41F AB38AAEBD4C37008}\\ \texttt{D04EACE12C31830E 424F5EBBDF678A56}_{16},\end{array}$$

то возвратить `УСПЕХ`, иначе — `ОШИБКА`.

Тест BASH.PRG.HASH256.6

1 Задать уровень стойкости: $l \leftarrow 256$.
2 Задать емкость: $d \leftarrow 1$.
3 Задать анонс: $A \leftarrow \perp$.
4 Для $i = 1, 2, \ldots, 10000$ выполнить:
1) псевдослучайным методом сгенерировать сообщения X длины 2048 октета;
2) испытуемой реализацией вычислить хэш-значение X и сохранить результат в Y;
3) эталонной реализацией вычислить хэш-значение X и сохранить результат в Y';
4) если $Y \neq Y'$, то возвратить `ОШИБКА`.

5 Возвратить УСПЕХ.

6.2.3 Программируемые алгоритмы аутентифицированного шифрования

При тестировании реализаций программируемых алгоритмов аутентифицированного шифрования выполняются тесты BASH.PRGAE128.1 – BASH.PRGAE128.6, BASH.PRGAE192.1 – BASH.PRGAE192.6, BASH.PRGAE256.1 – BASH.PRGAE256.6.

Входными данными тестов являются уровень стойкости $l \in \{128, 192, 256\}$, емкость $d \in \{1, 2\}$, ключ $K \in \{0,1\}^l$, анонс $A \in \{0,1\}^{32*}$, сообщение $X \in \{0,1\}^*$ и ассоциированные данные $I \in \{0,1\}^{8*}$.

В тестах для хранения зашифрованного сообщения используются слова $Y, Y' \in \{0,1\}^{|X|}$, а для хранения имитовставки — слова $T, T' \in \{0,1\}^l$. Дополнительно, для хранения результата расшифрования используется слово $X' \in \{0,1\}^{|Y|}$.

Тест BASH.PRG.AE128.1

1 Задать уровень стойкости: $l \leftarrow 128$.
2 Задать емкость: $d \leftarrow 2$.
3 Задать ключ:

$$K \leftarrow \texttt{2033394D6C320D09 65201A166E62001D}_{16}.$$

4 Задать анонс: $A \leftarrow \perp$.
5 Задать сообщение длины 54 октета:

$$X \leftarrow \begin{array}{l}\texttt{466966747920666F 7572206279746520}\\ \texttt{6F7220666F757220 68756E6472656420}\\ \texttt{7468697274792074 776F20626974206D}\\ \texttt{657373616765}_{16}.\end{array}$$

6 Задать ассоциированные данные длины 32 октета:

$$I \leftarrow \begin{array}{l}\texttt{5468697274792074 776F206279746520}\\ \texttt{6D65737361676520 2874657374203129}_{16}.\end{array}$$

7 Испытуемой реализацией установить защиту и сохранить результат в (Y, T).
8 Если

$$Y = \begin{array}{l}\texttt{3DC15B5B5C44352D 9DD48EB6AF1C7F25}\\ \texttt{AC47F495E2F95DC9 8A395C88FC7BC0FE}\\ \texttt{8D2274F248B34183 F7A19FD3EDF141B8}\\ \texttt{130D3EDF3F13}_{16},\end{array}$$

$$T = \texttt{2091C633C522E51B 1EFF3D704BDFA289}_{16},$$

то возвратить УСПЕХ, иначе — ОШИБКА.

Тест BASH.PRG.AE128.2

1 Задать уровень стойкости: $l \leftarrow 128$.
2 Задать емкость: $d \leftarrow 2$.

3 Задать ключ:

$$K \leftarrow \texttt{5BE3D61217B96181 FE6786AD716B890B}_{16}.$$

4 Задать анонс длины 8 октета:

$$A \leftarrow \texttt{9DEADEC2621747A6}_{16}.$$

5 Задать сообщение длины 32 октета:

$$X \leftarrow \begin{array}{l}\texttt{5468697274792074 776F206279746520}\\ \texttt{6D65737361676520 2874657374203129}_{16}.\end{array}$$

6 Задать ассоциированные данные: $I \leftarrow \perp$.
7 Испытуемой реализацией установить защиту и сохранить результат в (Y, T).
8 Если

$$Y = \begin{array}{l}\texttt{85E004B72641B567 95B8DA31BEE91AD7}\\ \texttt{F82BB6DC05FE9216 A4A71312885FFFB2}_{16},\end{array}$$

$$T = \texttt{2B4C6067265F837A 678EE9803F578817}_{16},$$

то возвратить `УСПЕХ`, иначе — `ОШИБКА`.

Тест BASH.PRG.AE128.3

1 Задать уровень стойкости: $l \leftarrow 128$.
2 Задать емкость: $d \leftarrow 1$.
3 Задать ключ:

$$K \leftarrow \texttt{2033394D6C320D09 65201A166E62001D}_{16}.$$

4 Задать анонс: $A \leftarrow \perp$.
5 Задать сообщение: $X \leftarrow \perp$.
6 Задать ассоциированные данные длины 32 октета:

$$I \leftarrow \begin{array}{l}\texttt{5468697274792074 776F206279746520}\\ \texttt{6D65737361676520 2874657374203129}_{16}.\end{array}$$

7 Испытуемой реализацией установить защиту и сохранить результат в (Y, T).
8 Если $Y = \perp$ и

$$T = \texttt{3501E8ADA1D710A5 986175B5916B0B77}_{16},$$

то возвратить `УСПЕХ`, иначе — `ОШИБКА`.

Тест BASH.PRG.AE128.4

1 Задать уровень стойкости: $l \leftarrow 128$.
2 Задать емкость: $d \leftarrow 1$.
3 Задать ключ:

$$K \leftarrow \texttt{5BE3D61217B96181 FE6786AD716B890B}_{16}.$$

4 Задать анонс длины 8 октета:

$$A \leftarrow \texttt{B194BAC80A08F53B}_{16}.$$

5 Задать сообщение X, состоящее из 192 октетов $\texttt{00}_{16}$.
6 Задать ассоциированные данные длины 49 октета:

$$I \leftarrow \begin{array}{l}\texttt{E12BDC1AE28257EC 703FCCF095EE8DF1}\\ \texttt{C1AB76389FE678CA F7C6F860D5BB9C4F}\\ \texttt{F33C657B637C306A DD4EA7799EB23D31}\\ \texttt{3E}_{16}.\end{array}$$

7 Испытуемой реализацией установить защиту и сохранить результат в (Y, T).
8 Если

$$Y = \begin{array}{l}\texttt{3833B2E4A51E8BFA 91C4E6AC3CA7B3E1}\\ \texttt{5734F392B29474F7 BD8120897A5D13A8}\\ \texttt{A09B32FAC052CE41 D2BBD68D552A93A7}\\ \texttt{AA0CCD9180D000CA DBBD8BA00BAF9E05}\\ \texttt{308E7D559091503B 18CE9303BCF8693E}\\ \texttt{ED776603BBB092C9 C3A9C27A70192092}\\ \texttt{D08398846A2D390F 53DA6938974EE3B4}\\ \texttt{A500BC9DEE21A101 00F51D4C3427B070}\\ \texttt{C5E3507559809196 0DE6AF44EDA36134}\\ \texttt{0E6020B397CB5256 792A600D737C7A66}\\ \texttt{28001749CC5A5B73 63FEFDAE20E4A5FD}\\ \texttt{118A785A42A88812 1761C518D4ACABF0}_{16},\end{array}$$

$$T = \texttt{73BF4BD53A2ED6B1 3B582BC70CA04E57}_{16},$$

то возвратить `УСПЕХ`, иначе — `ОШИБКА`.

Тест BASH.PRG.AE128.5

1 Задать уровень стойкости: $l \leftarrow 128$.
2 Задать емкость: $d \leftarrow 2$.
3 Для $i = 1, 2, \ldots, 10000$ выполнить:
1) псевдослучайным методом сгенерировать ключ K;
2) псевдослучайным методом сгенерировать анонс A длины 8 октета;
3) псевдослучайным методом сгенерировать сообщение X длины 1024 октета;
4) псевдослучайным методом сгенерировать ассоциированные данные I длины 1024 октета;
5) испытуемой реализацией установить защиту и сохранить результат в (Y, T);
6) испытуемой реализацией снять защиту с пары (Y, I) и сохранить результат в X';
7) если алгоритм снятии защиты возвратил $\perp$ или $X \neq X'$, то возвратить `ОШИБКА`.
4 Возвратить `УСПЕХ`.

Тест BASH.PRG.AE128.6

1 Задать уровень стойкости: $l \leftarrow 128$.
2 Задать емкость: $d \leftarrow 1$.
3 Для $i = 1, 2, \ldots, 10000$ выполнить:
1) псевдослучайным методом сгенерировать ключ K;
2) псевдослучайным методом сгенерировать анонс A длины 8 октета;
3) псевдослучайным методом сгенерировать сообщение X длины 1024 октета;
4) псевдослучайным методом сгенерировать ассоциированные данные I длины 1024 октета;
5) испытуемой реализацией установить защиту и сохранить результат в (Y, T);
6) эталонной реализацией установить защиту и сохранить результат в (Y', T');
7) если $Y \neq Y'$ или $T \neq T'$, то возвратить ОШИБКА.
4 Возвратить УСПЕХ.

Тест BASH.PRG.AE192.1

1 Задать уровень стойкости: $l \leftarrow 192$.
2 Задать емкость: $d \leftarrow 2$.
3 Задать ключ:

$$K \leftarrow \texttt{2033394D6C320D09 65201A166E62001D 6779410674740E13}_{16}.$$

4 Задать анонс: $A \leftarrow \perp$.
5 Задать сообщение длины 54 октета:

$$X \leftarrow \texttt{466966747920666F 7572206279746520 6F7220666F757220 68756E6472656420 7468697274792074 776F20626974206D 657373616765}_{16}.$$

6 Задать ассоциированные данные длины 32 октета:

$$I \leftarrow \texttt{5468697274792074 776F206279746520 6D65737361676520 2874657374203129}_{16}.$$

7 Испытуемой реализацией установить защиту и сохранить результат в (Y, T).
8 Если

$$Y = \texttt{7BFC37CEEA096D4B 1A611B92219CCEAB CF5B997ABFE1533C 4CFBF4EB866E5A09 991C98C73B04A614 BE45E471E9287E2D 2A1D7C43C8DD}_{16},$$

$$T = \texttt{B7C89033F37FF0A5 01368542027AF891 DAEC399D9B5A2AB8}_{16},$$

то возвратить УСПЕХ, иначе — ОШИБКА.

Тест BASH.PRG.AE192.2

1 Задать уровень стойкости: $l \leftarrow 192$.
2 Задать емкость: $d \leftarrow 2$.
3 Задать ключ:

$$K \leftarrow \texttt{5BE3D61217B96181 FE6786AD716B890B}$$
$$\texttt{5CB0C0FF33C356B8}_{16}.$$

4 Задать анонс длины 12 октета:

$$A \leftarrow \texttt{9DEADEC2621747A6 2A80A7C3}_{16}.$$

5 Задать сообщение длины 32 октета:

$$X \leftarrow \texttt{5468697274792074 776F206279746520}$$
$$\texttt{6D65737361676520 2874657374203129}_{16}.$$

6 Задать ассоциированные данные: $I \leftarrow \perp$.
7 Испытуемой реализацией установить защиту и сохранить результат в (Y, T).
8 Если

$$Y = \texttt{EC10AB97D0EB88F2 FA32EBB780B2FD9C}$$
$$\texttt{3E011523E53B07C4 D603501D6D3D416A}_{16},$$

$$T = \texttt{2EA75655204A58F7 E7E1AB46974692BC}$$
$$\texttt{9BD2B8E3C00DF8A3}_{16},$$

то возвратить `УСПЕХ`, иначе — `ОШИБКА`.

Тест BASH.PRG.AE192.3

1 Задать уровень стойкости: $l \leftarrow 192$.
2 Задать емкость: $d \leftarrow 1$.
3 Задать ключ:

$$K \leftarrow \texttt{2033394D6C320D09 65201A166E62001D}$$
$$\texttt{6779410674740E13}_{16}.$$

4 Задать анонс: $A \leftarrow \perp$.
5 Задать сообщение: $X \leftarrow \perp$.
6 Задать ассоциированные данные длины 32 октета:

$$I \leftarrow \texttt{5468697274792074 776F206279746520}$$
$$\texttt{6D65737361676520 2874657374203129}_{16}.$$

7 Испытуемой реализацией установить защиту и сохранить результат в (Y, T).
8 Если $Y = \perp$ и

$$T = \texttt{D140D605C4F6BE68 81E9A32766ED2869}$$
$$\texttt{C6F89F35E3B00542}_{16},$$

то возвратить `УСПЕХ`, иначе — `ОШИБКА`.

Тест BASH.PRG.AE192.4

1 Задать уровень стойкости: $l \leftarrow 192$.
2 Задать емкость: $d \leftarrow 1$.
3 Задать ключ:

$$K \leftarrow \texttt{5BE3D61217B96181 FE6786AD716B890B 5CB0C0FF33C356B8}_{16}.$$

4 Задать анонс длины 12 октета:

$$A \leftarrow \texttt{B194BAC80A08F53B 366D008E}_{16}.$$

5 Задать сообщение X, состоящее из 192 октетов $\texttt{00}_{16}$.
6 Задать ассоциированные данные длины 49 октета:

$$I \leftarrow \begin{array}{l}\texttt{E12BDC1AE28257EC 703FCCF095EE8DF1}\\ \texttt{C1AB76389FE678CA F7C6F860D5BB9C4F}\\ \texttt{F33C657B637C306A DD4EA7799EB23D31}\\ \texttt{3E}_{16}.\end{array}$$

7 Испытуемой реализацией установить защиту и сохранить результат в (Y, T).
8 Если

$$Y = \begin{array}{l}\texttt{97198569936C6BC6 ADE8B5BB087C997F}\\ \texttt{076661797B88DB72 DF764E1D0A5C917D}\\ \texttt{6CFB8733451BE071 B52F9F6AEC1C5507}\\ \texttt{F7B966369F767DE5 5E2BF0E7F7240540}\\ \texttt{4D617996D1E4972B FE29315F39F01521}\\ \texttt{7E9579EAEFCC6503 F666ECC25ADBAFD4}\\ \texttt{BBE1E0A13F1B008A 4C29DC153FEE7399}\\ \texttt{D764FD2716A4E43B E52216E22AA0A036}\\ \texttt{ECBDD6426DA9D090 064143941E888E1C}\\ \texttt{CD0154A14116BD3B 6F84F4BC400E6705}\\ \texttt{629DA6E6E1453D1C F706B05F0B095A43}\\ \texttt{880476C494EF5FD6 87764EEDBED4A69A}_{16},\end{array}$$

$$T = \begin{array}{l}\texttt{8A10881AB42C5DD1 80D2970931296ED8}\\ \texttt{9353CBA62C30D7FD}_{16},\end{array}$$

то возвратить `УСПЕХ`, иначе — `ОШИБКА`.

Тест BASH.PRG.AE192.5

1 Задать уровень стойкости: $l \leftarrow 192$.
2 Задать емкость: $d \leftarrow 2$.
3 Для $i = 1, 2, \ldots, 10000$ выполнить:
1) псевдослучайным методом сгенерировать ключ K;
2) псевдослучайным методом сгенерировать анонс A длины 12 октета;
3) псевдослучайным методом сгенерировать сообщение X длины 1024 октета;

4) псевдослучайным методом сгенерировать ассоциированные данные I длины 1024 октета;
5) испытуемой реализацией установить защиту и сохранить результат в (Y, T);
6) испытуемой реализацией снять защиту с пары (Y, I) и сохранить результат в X';
7) если алгоритм снятии защиты возвратил $\perp$ или $X \neq X'$, то возвратить `ОШИБКА`.
4 Возвратить `УСПЕХ`.

Тест BASH.PRG.AE192.6

1 Задать уровень стойкости: $l \leftarrow 192$.
2 Задать емкость: $d \leftarrow 1$.
3 Для $i = 1, 2, \ldots, 10000$ выполнить:
1) псевдослучайным методом сгенерировать ключ K;
2) псевдослучайным методом сгенерировать анонс A длины 12 октета;
3) псевдослучайным методом сгенерировать сообщение X длины 1024 октета;
4) псевдослучайным методом сгенерировать ассоциированные данные I длины 1024 октета;
5) испытуемой реализацией установить защиту и сохранить результат в (Y, T);
6) эталонной реализацией установить защиту и сохранить результат в (Y', T');
7) если $Y \neq Y'$ или $T \neq T'$, то возвратить `ОШИБКА`.
4 Возвратить `УСПЕХ`.

Тест BASH.PRG.AE256.1

1 Задать уровень стойкости: $l \leftarrow 256$.
2 Задать емкость: $d \leftarrow 2$.
3 Задать ключ:

$$K \leftarrow \texttt{2033394D6C320D09 65201A166E62001D 6779410674740E13 6865160D3D730C11}_{16}.$$

4 Задать анонс: $A \leftarrow \perp$.
5 Задать сообщение длины 54 октета:

$$X \leftarrow \texttt{466966747920666F 7572206279746520 6F7220666F757220 68756E6472656420 7468697274792074 776F20626974206D 657373616765}_{16}.$$

6 Задать ассоциированные данные длины 32 октета:

$$I \leftarrow \texttt{5468697274792074 776F206279746520 6D65737361676520 2874657374203129}_{16}.$$

7 Испытуемой реализацией установить защиту и сохранить результат в (Y, T).
8 Если

$$Y = \texttt{1DA58141375539D1 903E4C06AB373859 779355B3C30031A8 918A9D6F97624BA4 ABAE88F2335378B0 FD63295D1E32FCB7 2DF068130A9B}_{16},$$

$$T = \begin{array}{l}\texttt{3DF50074177D5EFA E170E621E6802916}\\ \texttt{6E3EDC7B6A1DB888 394F2C335211BC6C}_{16},\end{array}$$

то возвратить УСПЕХ, иначе — ОШИБКА.

Тест BASH.PRG.AE256.2

1 Задать уровень стойкости: $l \leftarrow 256$.
2 Задать емкость: $d \leftarrow 2$.
3 Задать ключ:

$$K \leftarrow \begin{array}{l}\texttt{5BE3D61217B96181 FE6786AD716B890B}\\ \texttt{5CB0C0FF33C356B8 35C405AED8E07F99}_{16}.\end{array}$$

4 Задать анонс длины 16 октета:

$$A \leftarrow \texttt{9DEADEC2621747A6 2A80A7C3FFA8E347}_{16}.$$

5 Задать сообщение длины 32 октета:

$$X \leftarrow \begin{array}{l}\texttt{94E1191BACD0D021 EA896457DA109A7F}\\ \texttt{BF10576FAAC3675B BD242AED036397A2}_{16}.\end{array}$$

6 Задать ассоциированные данные: $I \leftarrow \perp$.
7 Испытуемой реализацией установить защиту и сохранить результат в (Y, T).
8 Если

$$Y = \begin{array}{l}\texttt{EC10AB97D0EB88F2 FA32EBB780B2FD9C}\\ \texttt{3E011523E53B07C4 D603501D6D3D416A}_{16},\end{array}$$

$$T = \begin{array}{l}\texttt{2EA75655204A58F7 E7E1AB46974692BC}\\ \texttt{9BD2B8E3C00DF8A3}_{16},\end{array}$$

то возвратить УСПЕХ, иначе — ОШИБКА.

Тест BASH.PRG.AE256.3

1 Задать уровень стойкости: $l \leftarrow 256$.
2 Задать емкость: $d \leftarrow 1$.
3 Задать ключ:

$$K \leftarrow \begin{array}{l}\texttt{2033394D6C320D09 65201A166E62001D}\\ \texttt{6779410674740E13 6865160D3D730C11}_{16}.\end{array}$$

4 Задать анонс: $A \leftarrow \perp$.
5 Задать сообщение: $X \leftarrow \perp$.
6 Задать ассоциированные данные длины 32 октета:

$$I \leftarrow \begin{array}{l}\texttt{5468697274792074 776F206279746520}\\ \texttt{6D65737361676520 2874657374203129}_{16}.\end{array}$$

7 Испытуемой реализацией установить защиту и сохранить результат в (Y, T).
8 Если $Y = \perp$ и

$$T = \begin{array}{l}\texttt{839A7287D0BBEE02 31BB909AC24C3FDA}\\ \texttt{FC9EA34921D29CBF 97CCFD0902EC37ED}_{16},\end{array}$$

то возвратить УСПЕХ, иначе — ОШИБКА.

Тест BASH.PRG.AE256.4

1 Задать уровень стойкости: $l \leftarrow 256$.
2 Задать емкость: $d \leftarrow 1$.
3 Задать ключ:

$$K \leftarrow \begin{array}{l}\texttt{5BE3D61217B96181 FE6786AD716B890B}\\ \texttt{5CB0C0FF33C356B8 35C405AED8E07F99}_{16}.\end{array}$$

4 Задать анонс длины 16 октета:

$$A \leftarrow \texttt{B194BAC80A08F53B 366D008E584A5DE4}_{16}.$$

5 Задать сообщение X, состоящее из 192 октетов $\texttt{00}_{16}$.
6 Задать ассоциированные данные длины 49 октета:

$$I \leftarrow \begin{array}{l}\texttt{E12BDC1AE28257EC 703FCCF095EE8DF1}\\ \texttt{C1AB76389FE678CA F7C6F860D5BB9C4F}\\ \texttt{F33C657B637C306A DD4EA7799EB23D31}\\ \texttt{3E}_{16}.\end{array}$$

7 Испытуемой реализацией установить защиту и сохранить результат в (Y, T).
8 Если

$$Y = \begin{array}{l}\texttt{690673766C3E848C AC7C05169FFB7B77}\\ \texttt{51E52A011040E560 2573FAF991044A00}\\ \texttt{4329EEF7BED8E687 5830A91854D1BD2E}\\ \texttt{DC6FC2FF37851DBA C249DF400A0549EA}\\ \texttt{2E0C811D499E1FF1 E5E32FAE7F0532FA}\\ \texttt{4051D0F9E300D9B1 DBF119AC8CFFC48D}\\ \texttt{D3CBF1CA0DBA5DD9 7481C88DF0BE4127}\\ \texttt{85E40988B3158553 7948B80F5A9C49E0}\\ \texttt{8DD684A7DCA871C3 80DFDC4C4DFBE61F}\\ \texttt{50D2D0FBD24D8B9D 32974A347247D001}\\ \texttt{BAD5B16844002569 3967E77394DC088B}\\ \texttt{0ECCFA8D291BA13D 44F60B06E2EDB351}_{16},\end{array}$$

$$T = \begin{array}{l}\texttt{CDE5AF6EF9A14B7D 0C191B869A6343ED}\\ \texttt{6A4E9AAB4EE00A57 9E9E682D0EC051E3}_{16},\end{array}$$

то возвратить УСПЕХ, иначе — ОШИБКА.

Тест BASH.PRG.AE256.5

1 Задать уровень стойкости: $l \leftarrow 256$.
2 Задать емкость: $d \leftarrow 2$.
3 Для $i = 1, 2, \ldots, 10000$ выполнить:
1) псевдослучайным методом сгенерировать ключ K;
2) псевдослучайным методом сгенерировать анонс A длины 16 октета;

3) псевдослучайным методом сгенерировать сообщение X длины 1024 октета;
4) псевдослучайным методом сгенерировать ассоциированные данные I длины 1024 октета;
5) испытуемой реализацией установить защиту и сохранить результат в (Y, T);
6) испытуемой реализацией снять защиту с пары (Y, I) и сохранить результат в X';
7) если алгоритм снятии защиты возвратил $\perp$ или $X \neq X'$, то возвратить `ОШИБКА`.
4 Возвратить `УСПЕХ`.

Тест BASH.PRG.AE256.6

1 Задать уровень стойкости: $l \leftarrow 256$.
2 Задать емкость: $d \leftarrow 1$.
3 Для $i = 1, 2, \ldots, 10000$ выполнить:
1) псевдослучайным методом сгенерировать ключ K;
2) псевдослучайным методом сгенерировать анонс A длины 16 октета;
3) псевдослучайным методом сгенерировать сообщение X длины 1024 октета;
4) псевдослучайным методом сгенерировать ассоциированные данные I длины 1024 октета;
5) испытуемой реализацией установить защиту и сохранить результат в (Y, T);
6) эталонной реализацией установить защиту и сохранить результат в (Y', T');
7) если $Y \neq Y'$ или $T \neq T'$, то возвратить `ОШИБКА`.
4 Возвратить `УСПЕХ`.

6.3 Анализ исходных текстов

6.3.1 Корректность использования локальных переменных

Анализ корректности использования локальных переменных проводится для всех функций программы.

Под функцией понимается часть программы, которая выполняет специфические действия и описывается типом возвращаемого значения, именем функции, формальными параметрами. Выполнение функции осуществляется посредством вызова из программы или другой функции. Данному термину в языках программирования соответствуют такие понятия как «функция», «процедура», «метод» и т.п.

Для каждой локальной переменной v функции f эксперт определяет языковые конструкции f, в которых v встречается, и выполняет следующие проверки:

1 При использовании v в левой части оператора присваивания тип присваиваемого значения должен совпадать с типом v, в противном случае эксперт проверяет корректность результата, учитывая стандартные правила преобразования типов, определенные в используемом языке программирования.

2 Перед использованием значения переменной v должна быть выполнена ее инициализация.

3 Обращение на чтение/запись к переменной v должно происходить в пределах установленных для нее границ, в частности, если v является переменной составного типа, то обращение к элементам v должно происходить в пределах заданных размерностей.

4 Если v является переменной вещественного типа, то ее использование в операциях сравнения запрещено.

5 Если память для v выделяется в динамической области, то перед каждым выходом из f динамическая память должна быть освобождена. После освобождения памяти не должно быть языковых конструкций, ссылающихся на нее.

Примечание — В языках программирования, снабженных средствами «сборки мусора», освобождение динамической памяти, выделяемой для локальной переменной, может быть неявным.

6.3.2 Корректность использования глобальных переменных

Для каждой глобальной переменной v эксперт определяет языковые конструкции программы, в которых v встречается. Далее выполняются проверки 1 – 4 из п. 6.3.1 и следующие проверки:

1 Если память для v выделяется в динамической области, то перед каждым выходом из программы динамическая память должна быть освобождена. После освобождения памяти не должно быть языковых конструкций, ссылающихся на нее.

2 Если v может использоваться в многопоточном режиме работы программы, то должны быть реализованы механизмы, обеспечивающие разграничение доступа к v (механизмы синхронизации доступа к глобальной переменной), при этом данные механизмы не должны блокировать доступ к v на неограниченное время.

Примечание – В языках программирования, снабженных средствами «сборки мусора», освобождение динамической памяти, выделяемой для глобальной переменной, может быть неявным.

6.3.3 Корректность использования констант

Эксперт определяет языковые конструкции программы, в которых встречаются константы, определенных на шагах 1 и 2.5 алгоритма `bash-f` (п. 6.2.3 СТБ 34.101.77). Для каждой языковой конструкции эксперт проверяет, что константы заданы правильно.

6.3.4 Корректность программной логики функций программы

Для каждой функции программы эксперт выполняет следующие проверки:

1 Проверка допустимости переданных параметров и используемых глобальных переменных выполняется до их использования. Проверка может не выполняться, если в документации или в комментариях к функции оговорены ограничения на входные данные, при которых функция работает правильно, и эти ограничения соблюдаются для входных данных во всех вызовах функции.

2 Все заданные варианты условных переходов возможны.

3 Все адреса безусловных переходов доступны.

4 Каждый цикл завершается за конечное число шагов, т.е. завершение цикла гарантировано.

5 После выполнения операторов функции завершение функции гарантировано: достигается одна из точек выхода из функции.

6 Отсутствуют недостижимые участки кода.

7 Цепочки последовательных действий (например, открытие файла, чтение из файла, закрытие файла) корректны. Проверка выполняется, если в функции требуется выполнить некоторое действие, требующее определенной последовательности операций.

6.3.5 Корректность вызова стандартных функций

Эксперт проверяет, что в документации, комментариях исходных текстов программ или конфигурационных файлах указана информация, однозначно идентифицирующая вызываемые стандартные функции (версии компилятора, используемых стандартных библиотек и т.п.).

Для каждого вызова стандартной функции в программе эксперт проверяет:

1 Типы и значения параметров, фактически переданных в функцию, соответствуют типам и допустимым значениям параметров функции, указанным в документации на функцию (с учетом стандартных правил преобразования типов языка программирования).

2 Если в документации на функцию указано, что функция возвращает значение, то проводится анализ корректности использования возвращаемого значения, например, корректность использования в операторе присваивания, допустимость игнорирования возвращаемого значения и т.п.

3 Если в документации на функцию указано, что вызов функции может привести к возникновению исключительной ситуации или ошибки, проверяется наличие и корректность обработки исключительной ситуации.

4 Если в документации на функцию указано, что до и после вызова функции должны выполняться определенные действия, то проверяется наличие и корректность выполнения требуемых действий.

6.3.6 Корректность вызова функций программы

Эксперт проверяет, что в документации или комментариях исходных текстов программ для каждой функции программы указана информация, определяющая:

– допустимые входные параметры и возвращаемые значения функции;

– условия, при выполнении которых в ходе работы функции могут возникать исключительные ситуации (при наличии);

– действия, которые должны выполняться до и(или) после вызова функции (при наличии).

Для каждого вызова функции программы эксперт выполняет следующие проверки:

1 Типы и значения параметров, фактически переданных в функцию, соответствуют типам и допустимым значениям параметров функции (с учетом стандартных правил преобразования типов языка программирования).

2 Если функция возвращает значение, то проводится анализ корректности использования возвращаемого значения, например, корректность использования в операторе присваивания, допустимость игнорирования возвращаемого значения и т.п.

3 Если вызов функции может привести к возникновению исключительной ситуации или ошибки, проверяется наличие и корректность обработки исключительной ситуации.

4 Если до и после вызова функции должны выполняться определенные действия, то проверяется наличие и корректность выполнения требуемых действий.

5 Если функция использует глобальные переменные, то проверяется наличие инициализации данных переменных.

6.3.7 Корректность обработки исключительных ситуаций

Под исключительной ситуацией понимается ошибочная ситуация, возникающая при выполнении программы и требующая специальной обработки. Данному термину в языках программирования соответствует такие понятия как «ошибка», «исключение» и т.п.

Для анализа корректности обработки исключительных ситуаций эксперт формирует список функций, включающий стандартные функции и функции программы, вызов которых может приводить к возникновению исключительной ситуации.

Для каждого вызова функции из составленного списка эксперт проверяет:

1 После каждого вызова функции имеются проверка на случай возникновения исключительной ситуации и соответствующая обработка исключительной ситуации.

2 При проверке и обработке исключительной ситуации учтены все возможные виды исключительных ситуаций, возникновение которых возможно для вызываемой функции.

3 Исключительные ситуации обрабатываются адекватно (возвращаются верные коды ошибок и сообщения об ошибках и т.п.).

6.3.8 Корректность реализации криптографических примитивов

Криптографический примитив — это определенное в СТБ 34.101.77 вспомогательное преобразование, являющееся композиционной частью некоторого криптографического алгоритма.

В СТБ 34.101.77 определены следующие криптографические примитивы:

- алгоритм `bash-s` (п. 6.1 СТБ 34.101.77);
- алгоритм `bash-f` (п. 6.2 СТБ 34.101.77);
- команда `start` (п. 8.3 СТБ 34.101.77);
- команда `commit` (п. 8.4 СТБ 34.101.77);
- команда `restart` (п. 8.5 СТБ 34.101.77);
- команда `absorb` (п. 8.6 СТБ 34.101.77);
- команда `squeeze` (п. 8.7 СТБ 34.101.77);
- команда `encrypt` (п. 8.8 СТБ 34.101.77);
- команда `decrypt` (п. 8.9 СТБ 34.101.77);
- команда `ratchet` (п. 8.10 СТБ 34.101.77).

Анализируя структуру программы и используя документацию, эксперт формирует список криптографических примитивов, реализованных в программе. Для каждого примитива $g : A \to B$, осуществляющего отображение множества A в множество B, эксперт проверяет:

– наличие реализации примитива g в виде отдельной функции, части функции или композиции нескольких функций;

– тождественность реализации примитива g спецификации;

– отсутствие в g операций, не используемых для реализации примитива (наличие операций, не предусмотренных спецификацией на примитив, отражается в приложении к протоколу результатов анализа исходных текстов).

Допускается, что действие отображения g определено на множестве A^*, которое является подмножеством A. В этом случае эксперт дополнительно проверяет, что при выполнении программы прообразы отображения g всегда являются элементами A^*.

6.3.9 Корректность реализации криптографических алгоритмов

В СТБ 34.101.77 определены следующие криптографические алгоритмы:

– алгоритмы хэширования (раздел 7 СТБ 34.101.77);

– программируемые алгоритмы, включая программирумые алгоритмы хэширования и аутентифицированного шифрования (раздел 8 СТБ 34.101.77).

Анализируя структуру программы и используя документацию, эксперт формирует список криптографических алгоритмов, реализованных в программе. Для каждого алгоритма $f : X \times \Theta \to Y$, который ставит в соответствие входным данным $x \in X$ и параметру $\theta \in \Theta$ результат криптографического преобразования $y \in Y$, эксперт проверяет наличие соответствующей реализации алгоритма. Затем эксперт определяет множества функций реализации, в которых:

1) задаются параметры $\theta \in \Theta$;
2) задаются входные данные $x \in X$;
3) реализуется отображение f;
4) возвращается результат $y \in Y$.

Данные множества функций обозначаются соответственно F_1, F_2, F_3, F_4. Множества могут пересекаться или совпадать.

Для функций из множества F_1 эксперт проверяет корректность задания параметров $\theta \in \Theta$. При этом допустимым является использование в программном компоненте множества параметров Θ^*, которое является подмножеством Θ. Однако, использованное сужение множества Θ не должно состоять в ограничении области значений секретных параметров.

Для функций из множества F_2 эксперт проверяет корректность задания входных данных $x \in X$. При этом допускается, что множество входных данных X^* алгоритма является подмножеством X. Однако, использованное сужение множества входных данных должно быть оговорено в документации.

Примечание – Программа может обрабатывать не все допустимые входные данные. Например, может использоваться только определенный уровень стойкости.

Для функций из множества F_3 эксперт проверяет тождественность отображения, реализуемого функциями, спецификации на алгоритм f (при возможных ограничениях на параметры и входные данные, использованные в реализации отображения). Для этого, по результатам анализа элементов множества F_3, составляются использованные в реализации f композиции криптографических примитивов. Затем проверяется тождественность реализованных композиций композициям криптографических примитивов, заданным в спецификации и реализующим анализируемый криптографический алгоритм. Кроме этого, эксперт проводит проверку корректности реализации вспомогательных алгоритмов, использованных в программе и не описанных в спецификации. Если такой анализ провести не удается (алгоритм не описан в документации или описан не полно, без указания использованных источников), то по данному пункту проверки выдается отрицательное заключение по причине недостаточности данных. Если использованы простые вспомогательные алгоритмы, призванные оптимизировать выполнение программы и понятные эксперту, то их описание в документации не требуется.

Для функций из множества F_4 эксперт проверяет корректность выдачи результатов $y \in Y$ выполнения криптографического алгоритма. Сужение в реализации алгоритма f множества результатов Y является недопустимым.

6.3.10 Корректность управления секретными данными

Секретные данные — это ключи, параметры и другие данные криптографических алгоритмов, значения которых в соответствии со стандартом или документацией на СКЗИ должны быть защищены от раскрытия, т.е. должны храниться в секрете.

Секретными данными СТБ 34.101.77 являются:

– ключ;

– сообщение, подлежащее зашифрованию;

– результат расшифрования;

– данные, подлежащее имитозащите, если в соответствии с документацией они могут являться секретным объектом;

– имитовставка, если в соответствии с программной документацией она может интерпретироваться как псевдослучайные числа, используемые для создания секретного объекта;

– сообщение, подлежащее хэшированию, если в соответствии с программной документацией реализация алгоритма хэширования может использоваться для обработки секретных объектов (например, пароля в алгоритмах вычисления ключа по паролю).

Эксперт проверяет, что секретные данные используются в строгом соответствии с криптографическим алгоритмом. Допускается использование секретных данных во вспомогательных операциях с целью повышения быстродействия программной реализации криптоалгоритма. Другие операции с секретными данными не допускаются.

Эксперт проверяет, что все копии секретных данных в открытом виде уничтожаются при завершении работы с ними, при этом:

– значение секретных данных, размещенное в области памяти глобальной переменной, уничтожается перед каждым выходом из программы;

– значение секретных данных, размещенное в области памяти локальной переменной функции, уничтожается перед каждым выходом из данной функции;

– значение секретных данных, размещенное в динамической памяти, уничтожается перед каждым освобождением динамической памяти.

Примечание – Под уничтожением понимается такое изменение данных, хранящихся в электронных устройствах (оперативная память, память на магнитных носителях и др.), которое предотвращает их последующее восстановление. Например, уничтожение может состоять в записи в области памяти, занимаемой значениями секретных данных, фиксированных или случайно выбранных значений.

6.3.11 Отсутствие недокументированных возможностей

Эксперт определяет отсутствие недокументированных возможностей по результатам проверок, выполненных в п. 6.3.1 – 6.3.10.

Обнаруженные недокументированные возможности отражаются в протоколе анализа исходных текстов или в приложении к нему.

Приложение А
Форма протокола анализа документации

Экз. {Поле 1}

Протокол № {Поле 2} **от** {Поле 3}
результатов анализа документации
объекта испытаний {Поле 4}, реализующего криптографические алгоритмы
согласно СТБ 34.101.77-2020

1. Документы:

№	Название документа	Номер
1	{Поле 5}	{Поле 6}
2	{Поле 7}	{Поле 8}
3	{Поле 9}	{Поле 10}
4	{Поле 11}	{Поле 12}

2. При анализе документации были выполнены следующие проверки:

№	Название проверки	Отметка о выполнении
1	Проверка документа «Спецификация»	{Поле 13}
2	Проверка документа «Текст программы»	{Поле 13}
3	Проверка документа «Описание программы»	{Поле 13}
4	Проверка документа «Руководство программиста»	{Поле 13}

3. Заключение по результатам анализа документации: документация {Поле 6}, {Поле 8}, {Поле 10}, {Поле 12} соответствует (не соответствует) программе объекта испытаний в части реализации криптографических алгоритмов согласно СТБ 34.101.77-2020.

Эксперт, {Поле 15} {Поле 16}
{Поле 14}

В поле 1 указывается номер экземпляра протокола.

В поле 2 указывается номер, однозначно идентифицирующий протокол.

В поле 3 указывается дата составления протокола.

В поле 4 указывается название объекта испытаний в соответствии с представленной к испытаниям документацией.

В полях 5 и 6 указываются соответственно полное название документа «Спецификация» и его идентификационный/децимальный номер.

В полях 7 и 8 указываются соответственно полное название документа «Текст программы» и его идентификационный/децимальный номер.

В полях 9 и 10 указываются соответственно полное название документа «Описание программы» и его идентификационный/децимальный номер.

В полях 11 и 12 указываются соответственно полное название документа «Руководство программиста» и его идентификационный/децимальный номер.

В поле 13 указывается результат выполнения проверки: «положительно» — результат проверки положительный, «отрицательно» — результат проверки отрицательный. После завершения анализа документации и заполнения таблицы делается вывод о соответствии (не соответствии) документации программе объекта испытаний в части реализации в части реализации криптографических алгоритмов согласно СТБ 34.101.77. Вывод о соответствии делается только тогда, когда результаты всех проверок являются положительными.

В полях 14 и 16 указываются соответственно должность и Ф. И. О. эксперта.

В поле 15 ставится собственноручная подпись эксперта.

Информация об обнаруженных несоответствиях приводится в протоколе или приложении к протоколу в произвольной форме.

Приложение Б
Форма протокола тестирования

Экз. {Поле 1}

Протокол № {Поле 2} **от** {Поле 3}
результатов тестирования
объекта испытаний {Поле 4}, реализующего криптографические алгоритмы
согласно СТБ 34.101.77-2020

1. Файлы исходных текстов программ:

№	Имя файла	Хэш-значение
1	{Поле 5}	{Поле 6}
2	{Поле 5}	{Поле 6}
...	...	...

Хэш-значения для файлов вычислены согласно {Поле 7}.

2. В ходе тестирования объекта испытаний были выполнены следующие тесты:

№	Название теста	Отметка о выполнении
1	BASH.HASH128.1	{Поле 8}
2	BASH.HASH128.2	{Поле 8}
...	...	...

3. Заключение по результатам тестирования: объект испытаний {Поле 4} соответствует (не соответствует) требованиям, установленным в СТБ 34.101.77-2020.

Эксперт, {Поле 10} {Поле 11}
{Поле 9}

В поле 1 указывается номер экземпляра протокола.

В поле 2 указывается номер, однозначно идентифицирующий протокол.

В поле 3 указывается дата составления протокола.

В поле 4 указывается название объекта испытаний в соответствии с представленной к испытаниям документацией.

В поле 5 указываются имена исходных файлов программ объекта испытаний.

В поле 6 указывается значение функции хэширования для тестируемых файлов, вычисленное в соответствии со стандартом, указанным в поле 7. Разрешается использовать функции хэширования, определенные в СТБ 34.101.31 или СТБ 34.101.77.

В поле 8 указывается результат выполнения теста: «положительно» — тест завершен успешно, «отрицательно» — тест завершен с ошибкой; «не проводился» — тест не проводился, так как программа не поддерживает алгоритм или режим, определенный в тесте.

После завершения тестирования и заполнения таблицы делается вывод о соответствии (не соответствии) программной реализации объекта испытаний СТБ 34.101.77. Вывод о соответствии делается только тогда, когда все проводимые тесты выполнены успешно.

В полях 9, 11 указываются соответственно должность и Ф. И. О. эксперта.

В поле 10 ставится собственноручная подпись эксперта.

Приложение В
Форма протокола анализа исходных текстов

Экз. {Поле 1}

Протокол № {Поле 2} **от** {Поле 3}
результатов анализа исходных текстов программ
объекта испытаний {Поле 4}, реализующего криптографические алгоритмы
согласно СТБ 34.101.77-2020

1. Файлы исходных текстов программ:

№	Имя файла	Хэш-значение
1	{Поле 5}	{Поле 6}
2	{Поле 5}	{Поле 6}
	...	...

Хэш-значения для файлов вычислены согласно {Поле 7}.

2. В ходе анализа исходных текстов программ были выполнены следующие проверки:

№	Название проверки	Результат проверки
1	Корректность использования локальных переменных	{Поле 8}
2	Корректность использования глобальных переменных	{Поле 8}
3	Корректность использования констант	{Поле 8}
4	Корректность программной логики функций программы	{Поле 8}
5	Корректность вызова стандартных функций	{Поле 8}
6	Корректность вызова функций программы	{Поле 8}
7	Корректность обработки исключительных ситуаций	{Поле 8}
8	Корректность реализации криптографических примитивов	{Поле 8}
9	Корректность реализации криптографических алгоритмов	{Поле 8}
10	Корректность управления секретными данными	{Поле 8}
11	Отсутствие недокументированных возможностей	{Поле 8}

3. Заключение по результатам анализа исходных текстов программ: объект испытаний {Поле 4} соответствует требованиям, установленным в СТБ 34.101.77-2020.

Эксперт, {Поле 9} {Поле 10} {Поле 11}

В поле 1 указывается номер экземпляра протокола.

В поле 2 указывается номер, однозначно идентифицирующий протокол.

В поле 3 указывается дата составления протокола.

В поле 4 указывается название объекта испытаний в соответствии с представленной к испытаниям документацией.

В поле 5 указываются имена исходных файлов программ объекта испытаний.

В поле 6 указывается значение функции хэширования для исходных файлов программ, вычисленное в соответствии со стандартом, указанным в поле 7. Разрешается использовать функции хэширования, определенные в СТБ 34.101.31 или СТБ 34.101.77.

В поле 8 указывается результат выполнения проверки: «положительно» — результат проверки положительный, «отрицательно» — результат проверки отрицательный, «не проводилась» — проверка не требуется по причине специфики реализации программ объекта испытаний (например, в программе не используются глобальные переменные). После завершения анализа исходных текстов программ и заполнения таблицы делается вывод о соответствии (не соответствии) объекта испытаний СТБ 34.101.77. Вывод о соответствии делается только тогда, когда результаты всех проводимых проверок являются положительными.

В полях 9, 11 указываются соответственно должность и Ф. И. О. эксперта.

В поле 10 ставится собственноручная подпись эксперта.

Информация об обнаруженных ошибках и недокументированных возможностях приводится в протоколе или приложении к протоколу в произвольной форме и должна включать:

1) описание ошибки или недокументированной возможности;
2) имя файла и номера строк программы, содержащих ошибку.