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 34.101.60-2014

МИ.10160.10.01

Листов 33

Минск 2022

Предисловие

Настоящая методика испытаний предназначена для использования в испытательных лабораториях при проведении сертификационных испытаний средств криптографической защиты информации на соответствие требованиям СТБ 34.101.60-2014 «Информационные технологии и безопасность. Алгоритмы разделения секрета».

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 34.101.47-2017 «Информационные технологии. Защита информации. Криптографические алгоритмы генерации псевдослучайных чисел».

СТБ 34.101.60-2014 «Информационные технологии и безопасность. Криптографические алгоритмы разделения секрета».

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 34.101.60, 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 34.101.60, и документация на СКЗИ.

Целью испытаний является проверка соответствия объекта испытаний требованиям СТБ 34.101.60.

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 34.101.60, поддерживаемые объектом испытаний;

– программа, реализующая криптографические алгоритмы и требования СТБ 34.101.60, 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 34.101.60. Если программа не поддерживает некоторые алгоритмы, определенные в СТБ 34.101.60, 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 34.101.60, которые в составе других программ имеют действующие сертификаты соответствия требованиям СТБ 34.101.60, то проверки по тестированию и анализу исходных текстов для данных реализаций могут не проводиться. При этом для подтверждения соответствия объекта испытаний требованиям СТБ 34.101.60 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 34.101.60 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 34.101.60 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 34.101.60.

Тестированию подлежат криптографические алгоритмы, реализованные в программе и определенные в СТБ 34.101.60, включая:

– алгоритм генерации общего открытого ключа (см. п. 6.2.1);

– алгоритм генерации открытых ключей пользователей (см. п. 6.2.2);

– алгоритм генерации открытого ключа пользователя по идентификатору (см. п. 6.2.3);

– алгоритмов разделения и восстановления секрета (см. п. 6.2.4).

Если программа не реализует некоторые из алгоритмов, определенных в СТБ 34.101.60, 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 34.101.60. 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 34.101.60 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 34.101.60 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 34.101.60 криптографические алгоритмы;

– описание функций, реализующих определенные в СТБ 34.101.60 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 Алгоритм генерации общего открытого ключа

При тестировании реализации алгоритма генерации общего открытого ключа выполняются тесты BELS.GMC.1 – BELS.GMC.3.

Входными данными тестов является длина секрета $l \in \{128, 192, 256\}$.

В тестах для хранения результата генерации общего открытого ключа используется слово $M_0 \in \{0,1\}^l$.

Для каждого теста при выполнении алгоритма генерации общего открытого ключа для выбора открытого ключа (см. шаг 1 алгоритма) используются последовательные l-битовые слова, сформированные из выходных данных алгоритма `brng-ctr-hbelt` (см. СТБ 34.101.47), полученных на ключе, синхропосылке и дополнительном входном слове, состоящих из всех нулевых октетов.

Тест BELS.GMC.1

1 Задать длину секрета: $l \leftarrow 128$.

2 Испытуемой реализацией выполнить генерацию общего открытого ключа и сохранить результат в M_0.

3 Если

$$M_0 \neq \texttt{15F05ED7 396D0385 9B5E40F3 A9B21C09}_{16},$$

то возвратить `ОШИБКА`.

4 Испытуемой реализацией выполнить генерацию общего открытого ключа и сохранить результат в M_0.

5 Если

$$M_0 \neq \texttt{BD1C34F7 E53A65FF BB865B62 3E2B557B}_{16},$$

то возвратить `ОШИБКА`.

6 Испытуемой реализацией выполнить генерацию общего открытого ключа и сохранить результат в M_0.
7 Если

$$M_0 \neq \texttt{9730B43A D38B1614 F8022C2E 75DFD8A9}_{16},$$

то возвратить ОШИБКА.
8 Испытуемой реализацией выполнить генерацию общего открытого ключа и сохранить результат в M_0.
9 Если

$$M_0 \neq \texttt{21B9CEAB 39C7AB0C BAB956A5 6604BE09}_{16},$$

то возвратить ОШИБКА.
10 Испытуемой реализацией выполнить генерацию общего открытого ключа и сохранить результат в M_0.
11 Если

$$M_0 \neq \texttt{D35CD525 DCC5A119 62175212 38B4A934}_{16},$$

то возвратить ОШИБКА.
12 Испытуемой реализацией выполнить генерацию общего открытого ключа и сохранить результат в M_0.
13 Если

$$M_0 \neq \texttt{3592FFBC 58EDF663 9DEBB66A E64A64EE}_{16},$$

то возвратить ОШИБКА.
14 Возвратить УСПЕХ.

Тест BELS.GMC.2

1 Задать длину секрета: $l \leftarrow 192$.
2 Испытуемой реализацией выполнить генерацию общего открытого ключа и сохранить результат в M_0.
3 Если

$$M_0 \neq \texttt{5B5FA4B5 C5F09E1A CCD9C9C2 74B19D2F AE9AFC18 6B21D035}_{16},$$

то возвратить ОШИБКА.
4 Испытуемой реализацией выполнить генерацию общего открытого ключа и сохранить результат в M_0.
5 Если

$$M_0 \neq \texttt{EFE3BAEC 4CE34734 23329D2E E7BCA7C2 AFE89B7E A72353EB}_{16},$$

то возвратить ОШИБКА.
6 Испытуемой реализацией выполнить генерацию общего открытого ключа и сохранить результат в M_0.
7 Если

$$M_0 \neq \texttt{952BABEE 387CE1EF 892345B0 5DD51358 BF2F5B9F E546D7BF}_{16},$$

то возвратить ОШИБКА.

8 Испытуемой реализацией выполнить генерацию общего открытого ключа и сохранить результат в M_0.
9 Если

$$M_0 \neq \texttt{47F34073 8EF8F560 231C03AE 1B47EBC0 3AC5F6F4 99078787}_{16},$$

то возвратить ОШИБКА.
10 Испытуемой реализацией выполнить генерацию общего открытого ключа и сохранить результат в M_0.
11 Если

$$M_0 \neq \texttt{9532A1E0 AE557C25 2ED71395 A6A092B0 D256A608 A2F2D733}_{16},$$

то возвратить ОШИБКА.
12 Испытуемой реализацией выполнить генерацию общего открытого ключа и сохранить результат в M_0.
13 Если

$$M_0 \neq \texttt{E7F56BDB 0F1DFCC5 6A60D1BA 184917B9 DACFEB75 E6C320CA}_{16},$$

то возвратить ОШИБКА.
14 Возвратить УСПЕХ.

Тест BELS.GMC.3

1 Задать длину секрета: $l \leftarrow 256$.
2 Испытуемой реализацией выполнить генерацию общего открытого ключа и сохранить результат в M_0.
3 Если

$$M_0 \neq \texttt{5904634D 106B91CF 2D701C1E DD1F3372 E201E705 07AB2A15 C275E2C9 1CD333DD}_{16},$$

то возвратить ОШИБКА.
4 Испытуемой реализацией выполнить генерацию общего открытого ключа и сохранить результат в M_0.
5 Если

$$M_0 \neq \texttt{B758A022 511902EE 0A15A7F6 A7A1C37E DFF9B0CE 85BDC834 CA97CB2C EE321558}_{16},$$

то возвратить ОШИБКА.
6 Испытуемой реализацией выполнить генерацию общего открытого ключа и сохранить результат в M_0.
7 Если

$$M_0 \neq \texttt{5FD73B35 853C160A AC929953 0D34BB4C F9D79D21 5F03377B 4CA6A551 E39E528B}_{16},$$

то возвратить ОШИБКА.
8 Испытуемой реализацией выполнить генерацию общего открытого ключа и сохранить результат в M_0.

9 Если

$$M_0 \neq \texttt{E587B1F9 13A90D4B 3F07782A 57F868FE 28E0B970 2CCAE02B 47873DFF 98857CAA}_{16},$$

то возвратить ОШИБКА.

10 Испытуемой реализацией выполнить генерацию общего открытого ключа и сохранить результат в M_0.

11 Если

$$M_0 \neq \texttt{E5597A5F 19B8541A 1798E921 273E46DC 85717EF4 2CD01B3C 61F13711 0F6CFCDA}_{16},$$

то возвратить ОШИБКА.

12 Испытуемой реализацией выполнить генерацию общего открытого ключа и сохранить результат в M_0.

13 Если

$$M_0 \neq \texttt{B16C4A66 EF79E22A 73EECF3B B728243B 2E8AD661 7C348B64 5115B429 EC5A06B3}_{16},$$

то возвратить ОШИБКА.

14 Возвратить УСПЕХ.

6.2.2 Алгоритм генерации открытых ключей пользователей

При тестировании реализации алгоритма генерации открытых ключей пользователей выполняются тесты BELS.GMU.1 – BELS.GMU.3.

Входными данными тестов являются число пользователей n, длина секрета $l \in \{128, 192, 256\}$ и общий открытый ключ $M_0 \in \{0,1\}^l$.

В тестах для хранения результата генерации ключей пользователей используются слова $M_1, \ldots, M_n \in \{0,1\}^l$.

При выполнении алгоритма генерации открытых ключей пользователей для выбора значений переменной u (см. шаг 2.1 алгоритма) используются последовательные l-битовые слова, сформированные из выходных данных алгоритма `brng-ctr-hbelt` (см. СТБ 34.101.47), полученных на ключе, синхропосылке и дополнительном входном слове, состоящих из всех нулевых октетов.

Тест BELS.GMU.1

1 Задать длину секрета: $l \leftarrow 128$.

2 Задать число пользователей: $n \leftarrow 5$.

3 Задать общий открытый ключ:

$$M_0 \leftarrow \texttt{87000000 00000000 00000000 00000000}_{16}.$$

4 Испытуемой реализацией выполнить генерацию открытых ключей пользователей и сохранить результат в $M_1, \ldots, M_5$.

5 Если

$$M_1 = \texttt{2BF4BC81 4F09AC3C 5C81C5AD 1C058C69}_{16},$$

$$M_2 = \texttt{811B3586 00C30363 431AD42C 5D97069A}_{16},$$

$M_3 =$ `CF4EB863 A7E16D3B 004A9528 E3C58819`$_{16}$,

$M_4 =$ `A51094D2 78988CD2 96A92F3E 4AB7BB38`$_{16}$,

$M_5 =$ `4DE68FCB 7257B2D7 1BF606D2 FBFAF109`$_{16}$,

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

Тест BELS.GMU.2

1 Задать длину секрета: $l \leftarrow 192$.
2 Задать число пользователей: $n \leftarrow 5$.
3 Задать общий открытый ключ:

$M_0 \leftarrow$ `87000000 00000000 00000000 00000000 00000000 00000000`$_{16}$.

4 Испытуемой реализацией выполнить генерацию открытых ключей пользователей и сохранить результат в $M_1, \ldots, M_5$.
5 Если

$M_1 =$ `9D47904A B8BED946 6BA84189 E7919C19 D31616AB 5CE2AFAD`$_{16}$,

$M_2 =$ `53213D30 6898FBB9 BD8F47C7 BD6CFF01 75EA7FDE 9795DA4A`$_{16}$,

$M_3 =$ `D727B57D 180146A7 EE43423C C68E208A D859983B 69041CBC`$_{16}$,

$M_4 =$ `A558A658 CCCFAD75 B158B339 8A7CB033 F3719F09 1BBB25D2`$_{16}$,

$M_5 =$ `3FF1A2FC 359353AC F5A046E1 E87215C1 A8EAA2F7 AB77A413`$_{16}$,

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

Тест BELS.GMU.3

1 Задать длину секрета: $l \leftarrow 256$.
2 Задать число пользователей: $n \leftarrow 5$.
3 Задать общий открытый ключ:

$M_0 \leftarrow$ `25040000 00000000 00000000 00000000 00000000 00000000 00000000 00000000`$_{16}$.

4 Испытуемой реализацией выполнить генерацию открытых ключей пользователей и сохранить результат в $M_1, \ldots, M_5$.
5 Если

$M_1 =$ `2D767F7C C08F1B9F 75D29B8C B089A9E4 02C0741E A95A6180 0137158B 22791C58`$_{16}$,

$M_2 =$ `2B04A350 ACADAA5A 82F9E81C A35EE250 87C2B3FB C87CAD77 1306A4E5 2C22B519`$_{16}$,

$M_3 =$ `1BFF42C2 6CE743E6 BB25C9DF ABE679D3 D8B8D0E0 F820EFAA AE160007 92CCFC96`$_{16}$,

$M_4 =$ 6D63A717 D3F2681B C2ABF89A 824FC1B4 A4FE49FE 0282E08A 0921B070 BA30E561$_{16}$,

$M_5 =$ F3E41DB6 178B3DF0 FE681C78 F541D30F CE0481CE C916727B 92B6CD3A 59233C99$_{16}$,

то возвратить УСПЕХ, иначе — ОШИБКА.

6.2.3 Алгоритм генерации открытого ключа пользователя по идентификатору

При тестировании реализации алгоритма генерации открытого ключа пользователя по идентификатору выполняются тесты BELS.GMI.1 – BELS.GMI.6.

Входными данными тестов являются длина секрета $l \in \{128, 192, 256\}$, общий открытый ключ $M_0 \in \{0,1\}^l$ и идентификатор пользователя $Id \in \{0,1\}^*$.

В тестах для хранения результата генерации открытого ключа пользователя по идентификатору используются слова $M, M' \in \{0,1\}^l$.

Тест BELS.GMI.1

1 Задать длину секрета: $l \leftarrow 128$.
2 Задать общий открытый ключ:

$$M_0 \leftarrow \texttt{87000000 00000000 00000000 00000000}_{16}.$$

3 Задать идентификатор пользователя:

$$Id \leftarrow \texttt{416C696365}_{16}.$$

4 Испытуемой реализацией выполнить генерацию открытого ключа пользователя по идентификатору и сохранить результат в M.
5 Если

$$M = \texttt{F9D6F31B 5DB0BB61 F00E17EE F2E6007F}_{16},$$

то возвратить УСПЕХ, иначе — ОШИБКА.

Тест BELS.GMI.2

1 Задать длину секрета: $l \leftarrow 128$.
2 Задать общий открытый ключ:

$$M_0 \leftarrow \texttt{87000000 00000000 00000000 00000000}_{16}.$$

3 Для $i = 1, 2, \ldots, 10000$ выполнить:
1) псевдослучайным методом сгенерировать идентификатор Id длины 5 октета;
2) испытуемой реализацией выполнить генерацию открытого ключа пользователя по идентификатору и сохранить результат в M;
3) эталонной реализацией выполнить генерацию открытого ключа пользователя по идентификатору и сохранить результат в M';
4) если $M \neq M'$, то возвратить ОШИБКА.
4 Возвратить УСПЕХ.

Тест BELS.GMI.3

1 Задать длину секрета: $l \leftarrow 192$.
2 Задать общий открытый ключ:

$$M_0 \leftarrow \texttt{87000000 00000000 00000000 00000000 00000000 00000000}_{16}.$$

3 Задать идентификатор пользователя:

$$Id \leftarrow \texttt{416C696365}_{16}.$$

4 Испытуемой реализацией выполнить генерацию открытого ключа пользователя по идентификатору и сохранить результат в M.
5 Если

$$M = \texttt{09EA7929 7F94A3E4 3A3885FC 0D1BB8FD D0DF86FD 313CEF46}_{16},$$

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

Тест BELS.GMI.4

1 Задать длину секрета: $l \leftarrow 192$.
2 Задать общий открытый ключ:

$$M_0 \leftarrow \texttt{87000000 00000000 00000000 00000000 00000000 00000000}_{16}.$$

3 Для $i = 1, 2, \ldots, 10000$ выполнить:
1) псевдослучайным методом сгенерировать идентификатор Id длины 5 октета;
2) испытуемой реализацией выполнить генерацию открытого ключа пользователя по идентификатору и сохранить результат в M;
3) эталонной реализацией выполнить генерацию открытого ключа пользователя по идентификатору и сохранить результат в M';
4) если $M \neq M'$, то возвратить `ОШИБКА`.
4 Возвратить `УСПЕХ`.

Тест BELS.GMI.5

1 Задать длину секрета: $l \leftarrow 256$.
2 Задать общий открытый ключ:

$$M_0 \leftarrow \texttt{25040000 00000000 00000000 00000000 00000000 00000000 00000000 00000000}_{16}.$$

3 Задать идентификатор пользователя:

$$Id \leftarrow \texttt{416C696365}_{16}.$$

4 Испытуемой реализацией выполнить генерацию открытого ключа пользователя по идентификатору и сохранить результат в M.
5 Если

$$M = \texttt{D53CC51B E1F976F1 032A00D9 CD0E190E 62C37FFD 233E8A9D F14C85F8 5C51A045}_{16},$$

то возвратить УСПЕХ, иначе — ОШИБКА.

Тест BELS.GMI.6

1 Задать длину секрета: $l \leftarrow 256$.
2 Задать общий открытый ключ:

$$M_0 \leftarrow \texttt{25040000 00000000 00000000 00000000 00000000 00000000 00000000 00000000}_{16}.$$

3 Для $i = 1, 2, \ldots, 10000$ выполнить:
1) псевдослучайным методом сгенерировать идентификатор Id длины 5 октета;
2) испытуемой реализацией выполнить генерацию открытого ключа пользователя по идентификатору и сохранить результат в M;
3) эталонной реализацией выполнить генерацию открытого ключа пользователя по идентификатору и сохранить результат в M';
4) если $M \neq M'$, то возвратить ОШИБКА.
4 Возвратить УСПЕХ.

6.2.4 Алгоритмы разделения и восстановления секрета

При тестировании реализации алгоритмов разделения и восстановления секрета выполняются тесты BIGN.SHR.1 – BIGN.SHR.18.

Входными данными тестов являются число пользователей разделяющих секрет n, число пользователей восстанавливающих секрет r, пороговое число t, длина секрета $l \in \{128, 192, 256\}$, секрет $S \in \{0,1\}^l$, открытые ключи $M_0, \ldots, M_n \in \{0,1\}^l$, одноразовый ключ $k \in \{0,1\}^{(t-1)l}$ и частичные секреты $S_1, \ldots, S_n \in \{0,1\}^l$,

В тестах для хранения результата разделения секрета используются слова $S_1, \ldots, S_n \in \{0,1\}^l$ и $S'_1, \ldots, S'_n \in \{0,1\}^l$, а для хранения результата восстановления секрета — слово $S' \in \{0,1\}^l$.

В тестах используются открытые ключи и секреты из таблиц 1 – 3.

Таблица 1 — Открытые ключи и секреты ($l = 128$)

M_0	$\texttt{2F6AC750 E933994D 88440331 7622B258}_{16}$
M_1	$\texttt{2B040C30 764EC717 BB1F9131 7F0BF947}_{16}$
M_2	$\texttt{2503875C 52111755 3F7C1B7C 076B871D}_{16}$
M_3	$\texttt{63270A4F E1118D02 AE5C1D59 8943605B}_{16}$
M_4	$\texttt{1F1A171F 646DB35A 7072B93B EF6A560A}_{16}$
M_5	$\texttt{0F21C06F EA019E51 C333CC3C 50163477}_{16}$
S	$\texttt{48015787 4F248E46 2B81DE5A 1975022C}_{16}$
S_1	$\texttt{56E7BCC8 B08E291E D8BF64E2 A5E9261D}_{16}$
S_2	$\texttt{98258055 03147A21 AB7741B6 96F97D44}_{16}$
S_3	$\texttt{60607E94 EF1B5EB2 E4C54530 C13905BF}_{16}$
S_4	$\texttt{3D56B470 49F406BE 7CAD0707 6E564288}_{16}$
S_5	$\texttt{AF030B9D 9503458B EE05FDC2 C71D961E}_{16}$

Таблица 2 — Открытые ключи и секреты ($l = 192$)

M_0	EF52AA4F 8D539D2F 57137E31 F833FB2F FD5AF978 02250B76$_{16}$
M_1	8147777A 5C43C16F E46C2734 AB767F77 1745DD23 1A072B2C$_{16}$
M_2	5B438A70 7144F560 273CD129 827DF727 01589E6D 673DEA01$_{16}$
M_3	6364A157 9B707835 9F35BE36 A570525D 5A731B71 B93DEB08$_{16}$
M_4	B3727D5E 911AA246 A720416B 62274623 211ED359 D6148736$_{16}$
M_5	F34EE77A 2321EA5B C508CC36 64172F51 F24E1917 0E65AF38$_{16}$
S	C64A7291 6486BA02 6E8C7D79 09FBAFAA 694ABAE2 F15CC540$_{16}$
S_1	470BFE9C 1722F027 75CAE0A7 1B782B47 8BCAFF7B 53CFBD49$_{16}$
S_2	1BC40FD9 AE11ED72 C7657F10 CD609DA6 7234370F D73C9103$_{16}$
S_3	06E78363 D3E234B8 8E92E3CC 4BA9F568 0107ADAC 20C82838$_{16}$
S_4	6275E3C1 66B6DAD5 926167EB 515AF9AB 87E7BEA8 78577C55$_{16}$
S_5	D0F718B9 D587E55D 609DBBC1 5D0219A4 BE4DC505 E98A10EA$_{16}$

Таблица 3 — Открытые ключи и секреты ($l = 256$)

M_0	794BC27C 9324A1A5 1ECE5CB7 CA36C096 3C55E0DB 46D9CC0A EF123FFC 53811243$_{16}$
M_1	9101C43E 7D348974 CA2C34F2 F273854F 10CF0C75 767AD05E DFF279A4 0F45690C$_{16}$
M_2	076673AC BECD8847 8216BCE5 90E9B3E7 9B23F14B 3B0BCB22 E9CB0881 469B6CC7$_{16}$
M_3	DD78F5F2 2D62B128 FD5E180A 4272AC84 2A9B6710 40A6E915 CFC2EE9A 12FAFD73$_{16}$
M_4	6FDBEC49 C48E09E8 B6D7C940 2EF73AB4 3BA1239D 8436FE4E AF36D4F3 55766BC7$_{16}$
M_5	537071CA 876B7D56 8E77562D 73FCCAC3 2CC24C84 F0D806A4 3AC0A63A 7D2193E5$_{16}$
S	5F891BE8 340B60FC 95E70A93 0635B525 F8A5C610 A7A7CE95 82BCDA6A 12C86047$_{16}$
S_1	864C3275 4331B868 0DF497AF B26A5DA2 6C694D7C 89833116 5097B3A2 5126BE64$_{16}$
S_2	EDD67862 260CEC45 7B33D9AE BE3A8213 4584A03C 441794C3 6623DA00 EC4285A2$_{16}$
S_3	06DB03FC B3E7AAE0 EDECEC8B ABAA675B 2DAC6FAF B3691882 7EFA1566 A63F18E5$_{16}$
S_4	F8AEB99B DAFDD1E7 A98AC9F1 BAE2EB98 9F94105D D23B0163 97EE821F 5C952D77$_{16}$
S_5	96D3E320 E1731793 6D841CC9 212C465E 1E5DB7EA 40B53549 B987A2D2 1400E98D$_{16}$

Тест BELS.SHR.1

1 Задать длину секрета: $l \leftarrow 128$.
2 Задать число пользователей: $n \leftarrow 5$.
3 Задать пороговое число: $t \leftarrow 3$.
4 Задать открытые ключи $M_0, \ldots, M_5$ и секрет S из таблицы 1.
5 Задать одноразовый ключ:

$$k \leftarrow \begin{array}{l} \texttt{2ED3523C 3C35D213 F58F6E79 530A84E7} \\ \texttt{370FA33F 50A193BB 3EF19523 F72CEB44}_{16}. \end{array}$$

6 Испытуемой реализацией выполнить разделение секрета и сохранить результат в $S_1, \ldots, S_5$.
7 Если значения $S_1, \ldots, S_5$ совпадают со значениями из таблицы 1, то возвратить `УСПЕХ`, иначе — `ОШИБКА`.

Тест BELS.SHR.2

1 Задать длину секрета: $l \leftarrow 128$.
2 Задать число пользователей: $n \leftarrow 5$.
3 Задать пороговое число: $t \leftarrow 3$.
4 Задать открытые ключи $M_0, \ldots, M_5$ из таблицы 1.
5 Псевдослучайным методом сгенерировать секрет S.
6 Задать одноразовый ключ k, состоящий из 32 октетов 00_{16}.
7 Испытуемой реализацией выполнить разделение секрета и сохранить результат в $S_1, \ldots, S_5$.
8 Если $S_1 = S_2 = \ldots = S_5 = S$, то возвратить `УСПЕХ`, иначе — `ОШИБКА`.

Тест BELS.SHR.3

1 Задать длину секрета: $l \leftarrow 128$.
2 Задать число пользователей: $n \leftarrow 5$.
3 Задать пороговое число: $t \leftarrow 3$.
4 Задать общий открытый ключ M_0 из таблицы 1.
5 Задать число пользователей: $r \leftarrow 3$.
6 Для всевозможных подмножеств $\{i_1, i_2, i_3\} \subset \{1, 2, \ldots, 5\}$ выполнить:
1) задать открытые ключи $M_{i_1}, M_{i_2}, M_{i_3}$ и частичные секреты $S_{i_1}, S_{i_2}, S_{i_3}$ из таблицы 1;
2) испытуемой реализацией выполнить восстановление секрета и сохранить результат в S;
3) если значение S не совпадает со значением из таблицы 1, то возвратить `ОШИБКА`.
7 Возвратить `УСПЕХ`.

Тест BELS.SHR.4

1 Задать длину секрета: $l \leftarrow 128$.
2 Задать число пользователей: $n \leftarrow 5$.
3 Задать пороговое число: $t \leftarrow 3$.
4 Задать общий открытый ключ M_0 из таблицы 1.
5 Задать число пользователей: $r \leftarrow 2$.
6 Для всевозможных подмножеств $\{i_1, i_2\} \subset \{1, 2, \ldots, 5\}$ выполнить:
1) задать открытые ключи M_{i_1}, M_{i_2} и частичные секреты S_{i_1}, S_{i_2} из таблицы 1;
2) испытуемой реализацией выполнить восстановление секрета и сохранить результат в S;
3) если значение S совпадает со значением из таблицы 1, то возвратить `ОШИБКА`.
7 Возвратить `УСПЕХ`.

Тест BELS.SHR.5

1 Задать длину секрета: $l \leftarrow 128$.
2 Задать число пользователей: $n \leftarrow 5$.
3 Задать пороговое число: $t \leftarrow 3$.
4 Задать открытые ключи $M_0, \ldots, M_5$ из таблицы 1.
5 Для $i = 1, 2, \ldots, 10000$ выполнить:

1) псевдослучайным методом сгенерировать секрет S;
2) псевдослучайным методом сгенерировать одноразовый ключ k;
3) испытуемой реализацией выполнить разделение секрета и сохранить результат в $S_1, \dots, S_5$;
4) эталонной реализацией выполнить разделение секрета и сохранить результат в $S'_1, \dots, S'_5$;
5) если $S_i \neq S'_i$ для любого $i \in \{1, \dots, 5\}$, то возратить `ОШИБКА`.
6 Возвратить `УСПЕХ`.

Тест BELS.SHR.6

1 Задать длину секрета: $l \leftarrow 128$.
2 Задать число пользователей: $n \leftarrow 5$.
3 Задать пороговое число: $t \leftarrow 3$.
4 Задать общий открытый ключ M_0 из таблицы 1.
5 Для $i = 1, 2, \dots, 10000$ выполнить:
1) выбрать произвольным образом число пользователей $r \in \{2, \dots, 5\}$;
2) выбрать произвольным образом подмножество $\{i_1, \dots, i_r\} \subset \{1, \dots, 5\}$;
3) задать открытые ключи $M_{i_1}, \dots, M_{i_r}$ из таблицы 1;
4) псевдослучайным методом сгенерировать частичные секреты $S_{i_1}, \dots, S_{i_r}$;
5) испытуемой реализацией выполнить восстановление секрета и сохранить результат в S;
6) эталонной реализацией выполнить восстановление секрета и сохранить результат в S';
7) если $S \neq S'$, то возратить `ОШИБКА`.
6 Возвратить `УСПЕХ`.

Тест BELS.SHR.7

1 Задать длину секрета: $l \leftarrow 192$.
2 Задать число пользователей: $n \leftarrow 5$.
3 Задать пороговое число: $t \leftarrow 3$.
4 Задать открытые ключи $M_0, \dots, M_5$ и секрет S из таблицы 2.
5 Задать одноразовый ключ:

$$k \leftarrow \begin{array}{l} \texttt{9C543621 9BAA75DC 407FF3B0 1A2D6CFA} \\ \texttt{622369AB 95F008FA 639DFF56 C840DF9F} \\ \texttt{79CE9695 81313891 2C9AFF73 241990DB}_{16}. \end{array}$$

6 Испытуемой реализацией выполнить разделение секрета и сохранить результат в $S_1, \dots, S_5$.
7 Если значения $S_1, \dots, S_5$ совпадают со значениями из таблицы 2, то возвратить `УСПЕХ`, иначе — `ОШИБКА`.

Тест BELS.SHR.8

1 Задать длину секрета: $l \leftarrow 192$.
2 Задать число пользователей: $n \leftarrow 5$.
3 Задать пороговое число: $t \leftarrow 3$.
4 Задать открытые ключи $M_0, \ldots, M_5$ из таблицы 2.
5 Псевдослучайным методом сгенерировать секрет S.
6 Задать одноразовый ключ k, состоящий из 48 октетов 00_{16}.
7 Испытуемой реализацией выполнить разделение секрета и сохранить результат в $S_1, \ldots, S_5$.
8 Если $S_1 = S_2 = \ldots = S_5 = S$, то возвратить `УСПЕХ`, иначе — `ОШИБКА`.

Тест BELS.SHR.9

1 Задать длину секрета: $l \leftarrow 192$.
2 Задать число пользователей: $n \leftarrow 5$.
3 Задать пороговое число: $t \leftarrow 3$.
4 Задать общий открытый ключ M_0 из таблицы 2.
5 Задать число пользователей: $r \leftarrow 3$.
6 Для всевозможных подмножеств $\{i_1, i_2, i_3\} \subset \{1, 2, \ldots, 5\}$ выполнить:
1) задать открытые ключи $M_{i_1}, M_{i_2}, M_{i_3}$ и частичные секреты $S_{i_1}, S_{i_2}, S_{i_3}$ из таблицы 2;
2) испытуемой реализацией выполнить восстановление секрета и сохранить результат в S;
3) если значение S не совпадает со значением из таблицы 2, то возвратить `ОШИБКА`.
7 Возвратить `УСПЕХ`.

Тест BELS.SHR.10

1 Задать длину секрета: $l \leftarrow 192$.
2 Задать число пользователей: $n \leftarrow 5$.
3 Задать пороговое число: $t \leftarrow 3$.
4 Задать общий открытый ключ M_0 из таблицы 2.
5 Задать число пользователей: $r \leftarrow 2$.
6 Для всевозможных подмножеств $\{i_1, i_2\} \subset \{1, 2, \ldots, 5\}$ выполнить:
1) задать открытые ключи M_{i_1}, M_{i_2} и частичные секреты S_{i_1}, S_{i_2} из таблицы 2;
2) испытуемой реализацией выполнить восстановление секрета и сохранить результат в S;
3) если значение S совпадает со значением из таблицы 2, то возвратить `ОШИБКА`.
7 Возвратить `УСПЕХ`.

Тест BELS.SHR.11

1 Задать длину секрета: $l \leftarrow 192$.
2 Задать число пользователей: $n \leftarrow 5$.
3 Задать пороговое число: $t \leftarrow 3$.
4 Задать открытые ключи $M_0, \ldots, M_5$ из таблицы 2.
5 Для $i = 1, 2, \ldots, 10000$ выполнить:

1) псевдослучайным методом сгенерировать секрет S;
2) псевдослучайным методом сгенерировать одноразовый ключ k;
3) испытуемой реализацией выполнить разделение секрета и сохранить результат в $S_1, \ldots, S_5$;
4) эталонной реализацией выполнить разделение секрета и сохранить результат в $S'_1, \ldots, S'_5$;
5) если $S_i \neq S'_i$ для любого $i \in \{1, \ldots, 5\}$, то возратить `ОШИБКА`.
6 Возвратить `УСПЕХ`.

Тест BELS.SHR.12

1 Задать длину секрета: $l \leftarrow 192$.
2 Задать число пользователей: $n \leftarrow 5$.
3 Задать пороговое число: $t \leftarrow 3$.
4 Задать общий открытый ключ M_0 из таблицы 2.
5 Для $i = 1, 2, \ldots, 10000$ выполнить:
1) выбрать произвольным образом число пользователей $r \in \{2, \ldots, 5\}$;
2) выбрать произвольным образом подмножество $\{i_1, \ldots, i_r\} \subset \{1, \ldots, 5\}$;
3) задать открытые ключи $M_{i_1}, \ldots, M_{i_r}$ из таблицы 2;
4) псевдослучайным методом сгенерировать частичные секреты $S_{i_1}, \ldots, S_{i_r}$;
5) испытуемой реализацией выполнить восстановление секрета и сохранить результат в S;
6) эталонной реализацией выполнить восстановление секрета и сохранить результат в S';
7) если $S \neq S'$, то возратить `ОШИБКА`.
6 Возвратить `УСПЕХ`.

Тест BELS.SHR.13

1 Задать длину секрета: $l \leftarrow 256$.
2 Задать число пользователей: $n \leftarrow 5$.
3 Задать пороговое число: $t \leftarrow 3$.
4 Задать открытые ключи $M_0, \ldots, M_5$ и секрет S из таблицы 3.
5 Задать одноразовый ключ:

$$k \leftarrow \begin{array}{l}\texttt{6746D27E 011F379E 96EBDE72 2FB79BF2}\\ \texttt{91D63560 6E04E933 79A77732 C65DFF7F}\\ \texttt{CC860839 FCE207AA CC670452 0D1BD5EB}\\ \texttt{A557DD23 E71A4278 B6F82241 43D165AD}_{16}.\end{array}$$

6 Испытуемой реализацией выполнить разделение секрета и сохранить результат в $S_1, \ldots, S_5$.
7 Если значения $S_1, \ldots, S_5$ совпадают со значениями из таблицы 3, то возвратить `УСПЕХ`, иначе — `ОШИБКА`.

Тест BELS.SHR.14

1 Задать длину секрета: $l \leftarrow 256$.
2 Задать число пользователей: $n \leftarrow 5$.
3 Задать пороговое число: $t \leftarrow 3$.
4 Задать открытые ключи $M_0, \ldots, M_5$ из таблицы 3.
5 Псевдослучайным методом сгенерировать секрет S.
6 Задать одноразовый ключ k, состоящий из 64 октетов 00_{16}.
7 Испытуемой реализацией выполнить разделение секрета и сохранить результат в $S_1, \ldots, S_5$.
8 Если $S_1 = S_2 = \ldots = S_5 = S$, то возвратить `УСПЕХ`, иначе — `ОШИБКА`.

Тест BELS.SHR.15

1 Задать длину секрета: $l \leftarrow 256$.
2 Задать число пользователей: $n \leftarrow 5$.
3 Задать пороговое число: $t \leftarrow 3$.
4 Задать общий открытый ключ M_0 из таблицы 3.
5 Задать число пользователей: $r \leftarrow 3$.
6 Для всевозможных подмножеств $\{i_1, i_2, i_3\} \subset \{1, 2, \ldots, 5\}$ выполнить:
1) задать открытые ключи $M_{i_1}, M_{i_2}, M_{i_3}$ и частичные секреты $S_{i_1}, S_{i_2}, S_{i_3}$ из таблицы 3;
2) испытуемой реализацией выполнить восстановление секрета и сохранить результат в S;
3) если значение S не совпадает со значением из таблицы 3, то возвратить `ОШИБКА`.
7 Возвратить `УСПЕХ`.

Тест BELS.SHR.16

1 Задать длину секрета: $l \leftarrow 256$.
2 Задать число пользователей: $n \leftarrow 5$.
3 Задать пороговое число: $t \leftarrow 3$.
4 Задать общий открытый ключ M_0 из таблицы 3.
5 Задать число пользователей: $r \leftarrow 2$.
6 Для всевозможных подмножеств $\{i_1, i_2\} \subset \{1, 2, \ldots, 5\}$ выполнить:
1) задать открытые ключи M_{i_1}, M_{i_2} из таблицы 3;
2) испытуемой реализацией выполнить восстановление секрета и сохранить результат в S;
3) если значение S совпадает со значением из таблицы 3, то возвратить `ОШИБКА`.
7 Возвратить `УСПЕХ`.

Тест BELS.SHR.17

1 Задать длину секрета: $l \leftarrow 256$.
2 Задать число пользователей: $n \leftarrow 5$.
3 Задать пороговое число: $t \leftarrow 3$.
4 Задать открытые ключи $M_0, \ldots, M_5$ из таблицы 3.
5 Для $i = 1, 2, \ldots, 10000$ выполнить:

1) псевдослучайным методом сгенерировать секрет S;
2) псевдослучайным методом сгенерировать одноразовый ключ k;
3) испытуемой реализацией выполнить разделение секрета и сохранить результат в $S_1, \ldots, S_5$;
4) эталонной реализацией выполнить разделение секрета и сохранить результат в $S'_1, \ldots, S'_5$;
5) если $S_i \neq S'_i$ для любого $i \in \{1, \ldots, 5\}$, то возратить `ОШИБКА`.
6 Возвратить `УСПЕХ`.

Тест BELS.SHR.18

1 Задать длину секрета: $l \leftarrow 256$.
2 Задать число пользователей: $n \leftarrow 5$.
3 Задать пороговое число: $t \leftarrow 3$.
4 Задать общий открытый ключ M_0 из таблицы 3.
5 Для $i = 1, 2, \ldots, 10000$ выполнить:
1) выбрать произвольным образом число пользователей $r \in \{2, \ldots, 5\}$;
2) выбрать произвольным образом подмножество $\{i_1, \ldots, i_r\} \subset \{1, \ldots, 5\}$;
3) задать открытые ключи $M_{i_1}, \ldots, M_{i_r}$ из таблицы 3;
4) псевдослучайным методом сгенерировать частичные секреты $S_{i_1}, \ldots, S_{i_r}$;
5) испытуемой реализацией выполнить восстановление секрета и сохранить результат в S;
6) эталонной реализацией выполнить восстановление секрета и сохранить результат в S';
7) если $S \neq S'$, то возратить `ОШИБКА`.
6 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 стандартные открытые ключи (прил. А СТБ 34.101.60). Для каждой языковой конструкции эксперт проверяет, что стандартные открытые ключи 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 34.101.60 вспомогательное преобразование, являющееся композиционной частью некоторого криптографического алгоритма.

В СТБ 34.101.60 определены следующие криптографические примитивы:

– арифметические и логические операции над многочленами над двоичным полем (сложение, сравнение, умножение, деление, сложение по модулю, умножение по модулю, обращение по модулю, возведение в степень по модулю, нахождение наибольшего общего делителя);

– алгоритм `BuildIrred` (п. 6.3 СТБ 34.101.60);

– алгоритм решения системы сравнений (п. 7.2 СТБ 34.101.60). Если реализован другой алгоритм решения системы сравнений, то необходимо провести его проверку в соответствии с документацией на реализованный алгоритм. Этот алгоритм должен быть математически обоснован;

– алгоритмы проверки многочленов на неприводимость (п. Е.1 СТБ 34.101.60). Если реализован алгоритм, отличный от представленных в СТБ 34.101.60, то необходимо провести его проверку в соответствии с документацией на реализованный алгоритм. Этот алгоритм должен быть математически обоснован;

– расширенный алгоритм Евклида (п. Е.2 СТБ 34.101.60). Если используется реализация расширенного алгоритма Евклида, отличная от указанной в СТБ 34.101.60, то необходимо провести ее проверку в соответствии с документацией на реализованный алгоритм. Этот алгоритм должен быть математически обоснован.

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 34.101.60 определены следующие криптографические алгоритмы:

– алгоритм генерации общего открытого ключа (п. 6.4 СТБ 34.101.60);

– алгоритм генерации открытых ключей пользователей (п. 6.5 СТБ 34.101.60);

– алгоритм генерации открытого ключа пользователя по идентификатору (п. 6.6 СТБ 34.101.60);

– алгоритмов разделения и восстановления секрета (п. 7.3, 7.4 СТБ 34.101.60).

Дополнительно в СТБ 34.101.60 определены криптографические механизмы:

– проверка секрета (прил. В СТБ 34.101.60);

– разделение секрета без участия дилера (прил. Д СТБ 34.101.60).

Анализируя структуру программы и используя документацию, эксперт формирует список криптографических алгоритмов, реализованных в программе (включая алгоритмы выполнения криптографических механизмов). 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 с использованием документации 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, могут использоваться секреты только одной длины.

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 34.101.60 являются:

– секрет S;

– промежуточный секрет C;

– одноразовый ключ k;

– частичные секреты $S_1, \ldots, S_t$.

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 34.101.60-2014

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 34.101.60-2014.

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 криптографических алгоритмов согласно СТБ 34.101.60. 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 34.101.60–2013

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	BELS.GIR.1	{Поле 8}
2	BELS.GIR.2	{Поле 8}
...	...	...

3. Заключение по результатам тестирования: объект испытаний {Поле 4} соответствует (не соответствует) требованиям, установленным в СТБ 34.101.60–2013.

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 34.101.60. 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 34.101.60-2014

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 34.101.60-2014.

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 34.101.60. 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.