МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных Технологий
Кафедра Информационных систем и технологий
Специальность 1-40 01 01 «Программное обеспечение информационных технологий»
Специализация Программирование интернет-приложений

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА КУРСОВОГО ПРОЕКТ:

по дисциплине «Объектно-ориентированное программирование»
Тема «Электронная цифровая подпись»

Исполнитель
студент (ка) 2 курса группы 8 Иванов И.А.
(Ф.И.О.)

Руководитель
доцент, канд. техн. наук Пацей Н.В.
(учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой _____
Руководитель Пацей Н.В.
(подпись)

Минск 2017

Оглавление

Введение

Информационные ресурсы в современном обществе играют не меньшую, а нередко и большую роль, чем ресурсы материальные. Знания, кому, когда и где продать товар, может цениться не меньше, чем собственно товар и в этом плане динамика развития общества свидетельствует о том, что на "весах" материальных и информационных ресурсов последние начинают превалировать.

С позиций рынка информация давно уже стала товаром, ее производят, хранят, транспортируют, продают и покупают, а значит — воруют и подделывают — и, следовательно, ее необходимо защищать. Современное общество все в большей степени становится информационно—обусловленным, успех любого вида деятельности все сильней зависит от обладания определенными сведениями и от отсутствия их у конкурентов. И чем сильней проявляется указанный эффект, тем больше потенциальные убытки от злоупотреблений в информационной сфере, и тем больше потребность в защите информации. Одним словом, возникновение индустрии обработки информации с железной необходимостью привело к возникновению индустрии средств защиты информации, и это обстоятельство требует интенсивного развития практики, промышленности и теории компьютеризации общества.

Среди всего спектра методов защиты данных от нежелательного доступа особое место занимают криптографические методы. В отличие от других методов, они опираются лишь на свойства самой информации и не используют свойства ее материальных носителей, особенности узлов ее обработки, передачи и хранения. Образно говоря, криптографические методы строят барьер между защищаемой информацией и реальным или потенциальным злоумышленником из самой информации. Конечно, под криптографической защитой в первую очередь — так уж сложилось исторически — подразумевается шифрование данных. Раньше, когда эта операция выполнялось человеком вручную или с использованием различных приспособлений, и при посольствах содержались многолюдные отделы шифровальщиков, развитие криптографии сдерживалось проблемой реализации шифров, ведь придумать можно было все что угодно, но как это реализовать…

Почему же проблема использования криптографических методов в информационно ных системах (ИС) стала в настоящий момент особо актуальна? С одной стороны, расширилось использование компьютерных сетей, в частности глобальной сети Интернет, по которым передаются большие объемы информации государственного, военного, коммерческого и частного характера, не допускающего возможность доступа к ней посторонних лиц. С другой стороны, появление новых мощных компьютеров, технологий сетевых и нейронных вычислений сделало возможным дискредитацию криптографических систем еще недавно считавшихся практически не раскрываемыми.

1. Аналитическая часть

1.1 Понятие Электронной цифровой подписи

Электронная цифровая подпись – реквизит электронного документа, предназначенный для защиты данного электронного документа от подделки, полученный в результате криптографического преобразования информации с использованием закрытого ключа электронной цифровой подписи и позволяющей идентифицировать владельца сертификата ключа подписи, а также установить отсутствие искажений информации в электронном документе. Электронная цифровая подпись в электронном документе равнозначна собственноручной подписи в документе на бумажном носителе при одновременном соблюдении следующих условий:

- сертификат ключа подписи, относящийся к этой электронной цифровой подписи, не утратил силу (действует) на момент проверки или на момент подписания электронного документа при наличии доказательств, определяющих момент подписания;
- подтверждена подлинностью электронной цифровой подписи в электронном документе;
- электронная цифровая подпись используется в соответствии со сведениями, указанными в сертификате ключа подписи.

При этом электронной документ с электронной цифровой подписью имеет юридическое значение при осуществлении отношений, указанных в сертификате ключа подписи.

В скором будущем заключение договора будет возможно в электронной форме, который будет иметь такую же юридическую силу, как и письменный документ. Для этого он должен иметь механизм электронной цифровой подписи, подтверждаемый сертификатом. Владелец сертификата ключа подписи владеет закрытым ключом электронной цифровой подписи, что позволяет ему с помощью средств электронной цифровой подписи создавать свою электронную цифровую подпись в электронных документах (подписывать электронные документы). Для того, чтобы электронный документ могли открыть и другие пользователи, разработана система открытого ключа электронной подписи.

Для того чтобы иметь возможность скреплять электронный документ механизмом электронной цифровой подписи, необходимо обратиться в удостоверяющий центр за получением сертификата ключа подписи. Сертификат ключа подписи должен быть внесен удостоверяющим центром в реестр сертификатов ключей подписей не позднее даты начала действия сертификата ключа подписи. Удостоверяющий центр по закону должен подтверждать подлинность открытого ключа электронной цифровой подписи.

Существует несколько методов построения ЭЦП, а именно:

- шифрование электронного документа (ЭД) на основе симметричных алгоритмов. Данная схема предусматривает наличие в системе третьего лица – арбитра, пользующегося доверием обеих сторон. Авторизацией документа в данной

схеме является сам факт шифрования ЭД секретным ключом и передачи его арбитру.

- Использование ассиметричных алгоритмов шифрования. Фактом подписания документа является шифрование его на секретном ключе отправителя.

Развитием предыдущей идеи стала наиболее распространенная схема ЭЦП – шифрование окончательного результата обработки ЭД хеш-функцией при помощи ассиметричного алгоритма.

Появление этих разновидностей обусловлено разнообразием задач, решаемых с помощью электронных технологий передачи и обработки электронных документов.

При генерации ЭЦП используются параметры трех групп:

- общие параметры,
- секретный ключ,
- открытый ключ.

Отечественным стандартом на процедуры выработки и проверки ЭЦП является СТБ 1176.2-99.

Совместно с ЭЦП обычно применяются хэш-функции. Они служат для того, чтобы помимо аутентификации отправителя, обеспечиваемой ЭЦП, гарантировать, что сообщение не имеет искажений, и получатель получил именно то сообщение, которое подписал и отправил ему отправитель.

Хэш-функция - это процедура обработки сообщения, в результате действия которой формируется строка символов (дайджест сообщения) фиксированного размера. Малейшие изменения в тексте сообщения приводят к изменению дайджеста при обработке сообщения хэш-функцией. Таким образом, любые искажения, внесенные в текст сообщения, отразятся в дайджесте. . Анин «Защита компьютерной информации»

Алгоритм применения хэш-функции заключается в следующем:

- перед отправлением сообщение обрабатывается при помощи хэш-функции. В результате получается его сжатый вариант (дайджест). Само сообщение при этом не изменяется и для передачи по каналам связи нуждается в шифровании описанными выше методами;
- полученный дайджест шифруется закрытым ключом отправителя (подписывается ЭЦП) и пересылается получателю вместе с сообщением;
- получатель расшифровывает дайджест сообщения открытым ключом отправителя;
- получатель обрабатывает сообщение той же хэш-функцией, что и отправитель и получает его дайджест. Если дайджест, присланный отправителем, и дайджест, полученный в результате обработки сообщения получателем, совпадают, значит, в сообщение не было внесено искажений.

Существует несколько широко применяемых хэш-функций: MD5, SHA-1 и др.

Схема электронной подписи обычно включает в себя:

- алгоритм генерации ключевых пар пользователя,

- функцию вычисления подписи,
- функцию проверки подписи.

Функция вычисления подписи на основе документа и секретного ключа пользователя вычисляет собственно подпись. В зависимости от алгоритма функция вычисления подписи может быть детерминированной или вероятностной. Детерминированные функции всегда вычисляют одинаковую подпись по одинаковым входным данным. Вероятностные функции вносят в подпись элемент случайности, что усиливает криптостойкость алгоритмов ЭЦП. Однако, для вероятностных схем необходим надёжный источник случайности (либо аппаратный генератор шума, либо криптографически надёжный генератор псевдослучайных бит), что усложняет реализацию.

В настоящее время детерминированные схемы практически не используются. Даже в изначально детерминированные алгоритмы сейчас внесены модификации, превращающие их в вероятностные (так, в алгоритм подписи RSA вторая версия стандарта PKCS#1 добавила предварительное преобразование данных (OAEP).

Функция проверки подписи проверяет, соответствует ли данная подпись данному документу и открытому ключу пользователя. Открытый ключ пользователя доступен всем, так что любой может проверить подпись под данным документом.

Поскольку подписываемые документы переменной (и достаточно большой) длины, в схемах ЭЦП зачастую подпись ставится не на сам документ, а на его хэш. Для вычисления хэша используются криптографические хэш-функции, что гарантирует выявление изменений документа при проверке подписи. Хэш-функции не являются частью алгоритма ЭЦП, поэтому в схеме может быть использована любая надёжная хэш-функция.

1.2 Обзор существующих схем ЭЦП

1.2.1 Схема цифровой подписи RSA

Криптографические системы с открытым ключом используют так называемые однонаправленные функции, которые обладают следующим свойством:

- Если x известно, то $f(x)$ вычислить относительно просто,
- Если известно $y=f(x)$, то для x нет простого пути вычисления.

Под однонаправленностью понимается не теоретическая однонаправленость, а практическая невозможность вычислить обратное значение, используя современные вычислительные средства, за обозримый интервал времени.

В основу криптографической системы с открытым ключом RSA положена задача умножения и разложения простых чисел на множители, которая является вычислительно однонаправленной задачей.

В криптографической системе RSA каждый ключ состоит из пары целых чисел. Каждый участник создаёт свой открытый и секретный ключ самостоятельно.

Секретный ключ каждый из них держит в секрете, а открытые ключи можно сообщать кому угодно или даже публиковать их.

Система RSA может использоваться не только для шифрования, но и для цифровой подписи.

Поскольку цифровая подпись обеспечивает как аутентификацию автора сообщения, так и подтверждение целостности содержимого подписанного сообщения, она служит аналогом подписи, сделанной от руки в конце рукописного документа

Заметим, что подписанное сообщение не зашифровано. Оно пересылается в исходном виде и его содержимое не защищено. Путём совместного применения представленных выше схем шифрования и цифровой подписи в системе RSA можно создавать сообщения, которые будут и зашифрованы, и содержать цифровую подпись. Для этого автор сначала должен добавить к сообщению свою цифровую подпись, а затем — зашифровать получившуюся в результате пару(состоящую из самого сообщения и подписи к нему) с помощью открытого ключа принадлежащего получателю. Получатель расшифровывает полученное сообщение с помощью своего секретного ключа. Если проводить аналогию с пересылкой обычных бумажных документов, то этот процесс похож на то, как если бы автор документа поставил под ним свою печать, а затем положил его в бумажный конверт и запечатал, с тем чтобы конверт был распечатан только тем человеком, кому адресовано сообщение.

Поскольку генерация ключей происходит значительно реже операций, реализующих шифрование, расшифрование, а также создание и проверку цифровой подписи, задача вычисления $a = b^e \bmod n$ представляет основную вычислительную сложность. Эта задача может быть разрешена с помощью алгоритма быстрого возведения в степень. Таким образом для вычисления $M^e \bmod n$ требуется $O(\ln e)$ операций умножения по модулю.

1.2.2. Схема цифровой подписи DSA

В 1991 г. в США был опубликован проект федерального стандарта цифровой подписи - DSS (Digital Signature Standard, описывающий систему цифровой подписи DSA (Digital Signature Algorithm). Одним из основных критериев при создании проекта была его патентная чистота.

Предлагаемый алгоритм DSA, имеет, как и RSA, теоретико-числовой характер, и основан на криптографической системе Эль-Гамаля в варианте Шнорра. Его надежность основана на практической неразрешимости определенного частного случая задачи вычисления дискретного логарифма. Современные методы решения этой задачи имеют приблизительно ту же эффективность, что и методы решения задачи факторизации; в связи с этим предлагается использовать ключи длиной от 512 до 1024 бит с теми же характеристиками надежности, что и в системе RSA. Длина подписи в системе DSA меньше, чем в RSA, и составляет 320 бит.

С момента опубликования проект получил много критических отзывов, многие из которых были учтены при его доработке. Одним из главных аргументов

против DSA является то, что, в отличие от общей задачи вычисления дискретного логарифма, ее частный случай, использованный в данной схеме, мало изучен и, возможно, имеет существенно меньшую сложность вскрытия. Кроме того, стандарт не специфицирует способ получения псевдослучайных чисел, используемых при формировании цифровой подписи, и не указывает на то, что этот элемент алгоритма является одним из самых критичных по криптографической стойкости.

Функции DSA ограничены только цифровой подписью, система принципиально не предназначена для шифрования данных. По быстродействию система DSA сравнима с RSA при формировании подписи, но существенно (в 10-40 раз) уступает ей при проверке подписи.

Вместе с проектом DSS опубликован проект стандарта SHS (Secure Hash Standard), описывающий однонаправленную хэш-функцию SHA (Secure Hash Algorithm), рекомендованную для использования вместе с DSA. Хэш-функция SHA является модификацией алгоритма MD4, хорошо известного в криптографической литературе.

1.2.3. Схема цифровой подписи ГОСТ Р 34.10-01

В основу безопасности российского стандарта электронной цифровой подписи ГОСТ Р 34.10-2001 на эллиптической кривой $E(F_p)$ над простым конечным полем из p элементов и его американского аналога ECDSS положена задача дискретного логарифмирования на эллиптической кривой. Эллиптическая кривая $E(F_p)$ в форме Вейерштрасса задается уравнением

$$y^2 = f(x) \ (\mathrm{mod}\ p\ ),$$

где кубический многочлен $f(x)$ не имеет кратных корней в поле F_p.

В основу протокола электронной цифровой подписи ГОСТ Р 34.10-2001 положен протокол Эль-Гамаля. Этот протокол обладает вычислимыми морфизмами, позволяющими на основании одного подписанного сообщения создавать произвольное число формально правильных пар сообщение-подпись, поэтому подпись вычисляется для значения хэш-функции от сообщения.

Безопасность протокола подписи непосредственно зависит от сложности обращения хэш-функции и сложности вычисления коллизий хэш-функции, так как в первом случае можно вычислить сообщение для имеющейся подписи, а во втором — заготовить пару сообщений с одинаковыми значениями e, подписать одно из них, а потом заменить одно сообщение другим.

1.2.4. Схема цифровой подписи ESIGN

Esign – это схема подписи, предложенная Okamoto и Shiraishi в 1985 году. Она использует вычисления в кольце вычетов со специальным типом модуля. Главное преимущество схемы Esign – это ее скорость. В сравнении с системами, основанными на схеме RSA или схеме Эль-Гамаля, Esign отличается в несколько раз

более высоким быстродействием процессов подписи документа и проверки подписи. Пусть $N = p^2 q - 3k$ разрядное целое, где p и q - два простых числа примерно одинаковой длины. Секретную часть ключа составляют два k-битных простых числа p и q, а открытую часть ключа – пара (N,e), где e - целое число, большее четырех. Схема Esign использует $(k\text{-}1)$ битную хэш-функцию H, для вычисления хэш-значения от подписываемого сообщения.

Для проверки правильности подписи сообщения M проверяющий просто сравнивает k старших разрядов $s^e \pmod N$ с $0||H(M)$ и на основании этого сравнения делает вывод о принятии (если битовые последовательности совпадают) или отклонении (если битовые последовательности не совпадают) подписи.

Стойкость схемы Esign основывается на разновидности проблемы RSA, на извлечении корня e-й степени в кольце вычетов. Точнее, вычисление 'приближенного' значения корня, что считается сложной задачей.

1.2.5. Схема цифровой подписи RSA PSS-ES

Ниже будет рассмотрена вероятностная схема цифровой подписи на основе PSS-R (PSS - Probabilistic Signature Scheme - вероятностная схема шифрования, PSS-R - PSS with message recovery). В этой схеме рекомендуется, но не навязывается использование криптосистемы RSA, что и отражено в ее названии. Ее основное достоинство заключается в том, что можно использовать один и тот же ключевой набор для шифрования с открытым ключом и для цифровой подписи. То есть, если абонент **A**, посылая сообщение абоненту **B**, шифрует это сообщение с помощью открытого ключа **B** - пары (N,e). Абонент **B** дешифрует зашифрованное сообщение с помощью соответствующего секретного ключа - (N,d). Для того чтобы подписать сообщение **M**, пользователь **B** использует тот же секретный ключ (N,d). Как обычно, любой желающий может проверить эту подпись, используя открытый ключ **B** (N,e).

Схема шифрования и цифровой подписи PSS-ES основана на PSS-R (вероятностная схема цифровой подписи с восстановлением сообщения в процессе проверки подписи) и k-разрядной однонаправленной функции с ловушкой f. Как PSS-R схема подписи PSS-ES восстанавливает сообщение в процессе проверки подписи. Поэтому можно подписывать только сообщения фиксированной длины - $(k - k_0 - k_1)$ бит. Для подписи сообщения M произвольной длины необходимо использовать хэш-функцию.

1.2.6. Схема цифровой подписи Клауса Шнорра(Schnorr)

Стойкость схемы Шнорра основывается на трудной задаче вычисления дискретных логарифмов. Для генерации пары ключей сначала выбираются два простых числа, p и q так, чтобы q было сомножителем p-1. Затем выбирается a, не равное 1, такое что $a^q=1 \pmod p$. Все эти числа могут быть свободно опубликованы и использоваться группой пользователей. Для генерации конкретной пары ключей выбирается случайное число, меньшее q. Оно служит закрытым ключом, s. Затем

вычисляется открытый ключ $v = a^{-s} \pmod p$. Для подписи сообщений используется хэш-функция H(M).

Предварительные вычисления. Пользователь **A** выбирает случайное число r, меньшее q, и вычисляет $x = a^r \pmod p$.

Подпись сообщения *M*. Для того чтобы подписать сообщение *M* пользователю **A** необходимо выполнить следующие действия:

1. Объединить *M* и *x* и хэшировать результат:

$$e = H(M,x)$$

2. Вычислить $y = (r + se) \pmod q$. Подписью являются значения *e* и *y*, их нужно выслать получателю **B**.

Проверка подписи для сообщения *M*. Получатель **B** вычисляет $x' = a^y v^e \pmod p$. Затем он проверяет, что хэш-значение для объединения *M* и *x'* равно *e*. $e = H(M,x')$ Если это так, то он считает подпись верной.

В своей работе Шнорр приводит следующие свойства своего алгоритма:

- Большая часть вычислений, нужных для генерации подписи и независящих от подписываемого сообщения, может быть выполнена на стадии предварительных вычислений. Следовательно, эти вычисления могут быть выполнены во время простоя и не влияют на скорость вычисления подписи.
- При одинаковом уровне безопасности длина подписей для Schnorr короче, чем для RSA. Например, при 140-битовом *q* длина подписей равна всего лишь 212 битам, меньше половины длины подписей RSA. Подписи Schnorr также намного короче подписей ElGamal.

1.2.7Схема цифровой подписи ElGamal

Для генерации ключевой пары выбираются большое простое число p и примитивный элемент g мультипликативной группы поля *GF(p)*. Выбирается случайное число *x* такое, что $x<p-1$. Открытым ключом является $y = g^x \pmod p$, секретным ключем является *x*.

Подпись сообщения *M*. Для того, чтобы подписать сообщение *M* пользователю **A** необходимо выполнить следующие действия:

Выбрать случайно и равновероятно число *k* из группы обратимых элементов кольца $\boldsymbol{Z}_{p-1}^{*}$. То есть НОД(k,p-1) = 1;

Вычислить $a = g^k \pmod q$;

Вычислить $b = (M-xa)k^{-1} \pmod{p-1}$;

Подписью для сообщения M является пара (a,b).

Проверка подписи для сообщения M. Для проверки подписи (a,b) для сообщения M получатель **B** проверяет выполнение равенства $g^M = y^a a^b \pmod p$.

Надо заметить, что для ускорения процесса подписи вычисления на шагах 1,2 можно проводить заранее, вычислять значения k^{-1} тоже. Как и в других похожих схемах подписи значение *k* должно оставаться в секрете и выбираться случайно.

1.2.8Схема Диффи – Лампорта

Для аутентификации пользователя можно применять произвольную симметрическую схему. Пусть отправитель сообщения **A** хочет подписать n-битовое бинарное сообщение $m=m_1..\ m_n$ из V_2^n. Он выбирает $2n$ ключей некоторой криптосистемы, которые хранятся в секрете. Обозначим их так: $a_1,..,a_n;b_1,..,b_n$.

Если E - алгоритм шифрования, то **A** генерирует $4n$ параметров проверки $\{(X_i,Y_i,U_i,V_i)$: $1<= i <= n\}$, где X_i и Y_i - величины, подаваемые на вход E и связанные соотношениями

$U_i = E(X_i,a_i)$, $V_i = E(Y_i,b_i)$, $1 <= i < n$.

Параметры проверки заранее посылаются получателю **B** и третьему доверенному лицу.

Чтобы подписать n-битовое сообщение $m=(m_1,..,m_n)$, пользователь **A** применяет следующую процедуру: его подпись будет цепочкой $S = S_1 ..S_n$, где для каждого i

$S_i = a_i$, если $m_i =0$,

$S_i = b_i$, если $m_i =1$.

Пользователь **B** проверяет подпись следующим образом: для каждого i ($0 <= i <= n$) он использует бит m_i и ключ S_i, чтобы проверить:

если $m_i = 0$, то $E(X_i,S_i) = U_i$,

если $m_i = 1$, то $E(Y_i,S_i) = V_i$.

Пользователь **B** принимает подписанное сообщение за истинное только в том случае, если при процедуре проверки эти равенства выполнены для каждого L.

Эта система проста в использовании, но у нее есть, по крайней мере, два очевидных недостатка. Во-первых, необходима предварительная передача параметров проверки. Во-вторых, что более важно, подпись сильно увеличивает длину сообщения. Если в криптосистеме используются ключи длиной, скажем, 64 бита, то длина подписанного сообщения увеличится в 64 раза.

1.2.9 Вероятностная схема подписи Рабина

В 1978 г. Рабин предложил следующий способ подписи. Пусть E - функция шифрования некоторой криптосистемы, (k_i, $1 <= i <= 2r$) - последовательность случайно выбранных ключей, которые отправитель **A** держит в секрете. Пользователь **B** получает список параметров проверки (X_i,U_i) ($1 <= i <= 2r$), где

$E(X_i,k_i) = U_i$ ($1 <= i <= 2r$).

Эти параметры хранятся в доступном для всех месте.

Предположим, что **A** хочет подписать сообщение m. Его подписью будет цепочка $S = S_1S_2..S_{2r}$, для каждого i ($1<=i<=2r$) $S_i =E(m,k_i)$. **B** делает следующее. Сначала он выбирает случайным образом r ключей, которые **A** должен ему предъявить: k_{i1} , k_{i2} ,..,k_{ir}. При получении этих ключей от **A** он проверяет:

$E(m,k_{i1}) = S_{i1}$, $E(X_{i1},k_{i1}) = U_{i1}$

и далее для всех индексов $i_2, i_3, .., i_r$. Он считает действительной подпись от **A** только в том случае, когда выдерживаются все проверки. Безопасность получателя зависит от его уверенности в том, что только отправитель, знающий секретный ключ, может послать так подписанное сообщение. Что касается **A**, то предположим, что он отказывается от сообщения, которое по утверждению **B** он подписал. Тогда протокол для **A** следующий: он должен предъявить судье свои секретные ключи $k_1,k_2,..,k_{2r}$, и в присутствии обеих сторон делается 2*r* проверок:

$E(m,k_i)=S_i$, $E(X_i,k_i)=U_i$.

Рассмотрим, что это означает в трех возможных случаях.

Корректными являются менее r проверок. Тогда **B** не должен был принимать подписанное сообщение.

Выдерживается точно r проверок, то есть при формировании ключей **B** выбрал именно это подмножество из r ключей. Вероятность, что он угадает это подмножество, равна $p_r = (C_{2r}^{\ r})^{-1}$, для r=18, p_r примерно равно 10^{-10}.

Выдерживается $r+1$ и более проверок: отказ абонента **A** не принимается.

1.2.10 Электронная цифровая подпись СТБ 1176.2-99

Введенный в 1999г. стандарт Республики Беларусь СТБ 1176.2 "Информационная технология. Защита информации. Процедуры выработки и проверки электронной цифровой подписи " базируется на схеме ЭЦП Шнорра.

При выработке и проверке подписи используются l-битовое простое число p и r-битовое простое число q, $q \mid (p-1)$. Допустимые значения указаны в таблице 1. Применяется определяемая СТБ 1176.1 функция хеширования h, параметр L которой устанавливается равным $(r-1)$.

Часть преобразований стандарта выполняется в группе G, определяемой множеством $B_p = \{1,2, ...,p-1\}$ и операцией *:

$u * v = uvR^{-1} \ (mod\ p)$,

где $R = 2^{l+2}$. Использование операции * вместо обычного умножения по модулю p упрощает применение алгоритма Монтгомери. Далее, $u^{(m)}$ - m-я степень числа u из B_p как элемента G. В стандарте приведены алгоритмы генерации чисел p,q и элемента a из B_p, имеющего порядок q в группе G. Числа p,q,a являются долговременными параметрам СТБ 1176.2-99, едиными для группы пользователей.

Всякое сообщение M, подпись к которому вырабатывается или проверяется, задается последовательностью байтов. Если t- n-разрядное число по основанию $2^8 = 256$, то есть

$t = t_0 (256)^0+..+t_{n-1} (256)^{n-1}$,

причем $0 <= t_i <256$, t_{n-1} неравно 0, то t||M --- сообщение, полученное вставкой байтов $t_0,t_1,..,t_{n-1}$ в начало M.

При выработке подписи S к сообщению M используется личный ключ x, 0<x<q. Выполняются следующие шаги.

Выработать случайное секретное число k, 1<k<q.

$t = a^{(k)}$.

$U = h(t||M)$. Если U=0, то вернуться к шагу 1.

$V = (k-xU)$. Если V=0, то вернуться к шагу 1.

$S = U2^r+V$.

При проверке подписи S к сообщению M используется открытый ключ $y = a^{(X)}$. Алгоритм проверки подписи состоит из следующих шагов.

$V = S \ (mod \ 2^r)$.

$U = (S-V)/2^r$.

Проверить условия $0<U<2^{r-1}$, $0<V<q$. Если хотя бы одно из условий нарушается, то подпись признается недействительной и выполнение алгоритма завершается.

$t = a^{(V)} * y^{(U)}$.

$W = h(t||M)$.

Если W неравно U, то подпись признается недействительной. Если $W=U$, то принимаются решения о том, что:

а) подпись S была создана с помощью личного ключа x, связанного с открытым ключем y;

б)подпись S и сообщение M не были изменены с момента их создания.

2. Проектирование

2.1 Постановка задачи

Целью курсового проекта является разработка и создание системы электронной цифровой подписи на основе стандарта СТБ 1176.2-99. Использование ЭЦП необходимо для внутренних целей:

- защита от несанкционированного доступа;
- организации электронного документооборота;
- установления авторства.

Данная информационная система должна решать следующие задачи:

- конфиденциальность документов;
- установление лица, отправившего документ;
- разграничение доступа по сотрудникам.

2.2 Проектирование приложения

В данной работе проектирование сводится к описанию классов, реализующих вычисление функции хеширования, и выработки значения цифровой подписи. Диаграмма классов представлена на рисунке 2.1.

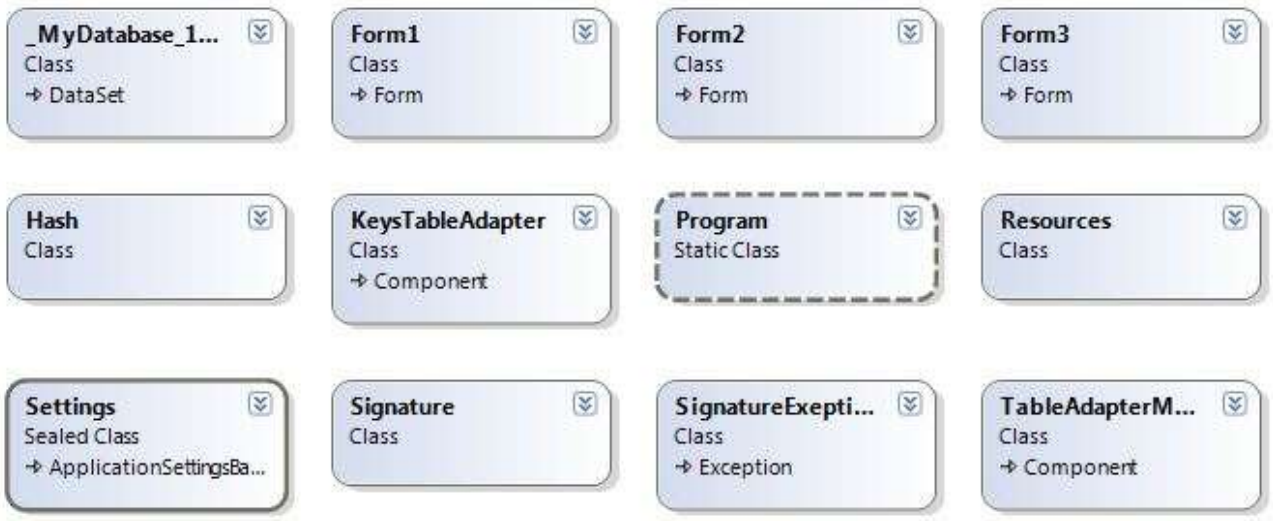

Рисунок 2.1 – Диаграмма классов

Как видно из рисунка в программе используется большое число классов, но наибольший интерес представляют только 2, так как остальные классы реализуются практически независимо от программиста. Класс Hash – был реализован как набор статических функций, необходимых для вычисления функции хэширования. Большинство методов класса являются служебными, и нет необходимости их вызывать. По сути класс предоставляет всего одну функцию – вычисление значения функции хеширования по заданному файлу. Методы класса Signature как и в классе

Hash в общем случае являются служебными. Класс предоставляет пользователю функции, осуществляющие выработку и проверку ЭЦП файла. Остальные классы обеспечивают вывод и управление информацией, необходимой для вычисления значений ЭЦП.

В данном программном средстве не было необходимости проектировать сложные базы данных, так как работа осуществляется с одним и тем же небольшим набором данных. Все данные хранятся в одной компактной таблице (Рисунок 2.2).

Рисунок 2.2 – Таблица базы данных

Так как имена и фамилии пользователей могут повторяться – наиболее целесообразно использовать в качестве первичного ключа таблицы номер паспорта пользователя, так как ни в одном стране мира нет 2-х человек с одинаковыми номерами паспортов. Также было наложено ограничение уникальности на поле «Ключ» дабы избежать коллизии при проверке подписи.

Так как реализация центра сертификации почти не отличается от реализации основного приложение – нет смысла его отдельно описывать. Стоит отметить лишь то, что в отличие от программы выработки подписи центр сертификации может подписывать файлы только одним закрытым ключом – это необходимо для того, чтобы любые пользователи могли проверить подлинность того или иного пользовательского ключа.

В данном проекте центр сертификации ключей и программа выработки и проверки ЭЦП взаимодействуют только через импорт-экспорт базы данных. Так как с центром сертификации постоянно взаимодействует большое количество пользователей в приложение выработки и проверки подписи импортируется вся база данных центра сертификации и заменяет собой существующую базу. Это также позволяет сохранять согласованность информацию между центром и любым пользователем, использующим данное программное средство. Возможности экспорта базы данных также предусмотрена и в пользовательской программе, чтобы в случае потери базы данных ее можно было восстановить без обращения к центру сертификации.

3. Реализация

Так как процесс выработки и проверки цифровой подписи представляет собой решение большого количества математических выражений с числами большой разрядности (больше 100 знаков) – реализация данного программного средства сводится к созданию классов, реализующих эти вычисления.

Для того, чтобы данные классы могли производить операции над большими числами в качестве основного типа дынных была выбрана структура BigInteger, которая появилась с выходом NET.Fraimwork4 и VisualStudio 2010.

Как уже говорилось выше в классах, реализующих вычисление функции хеширования и выработку и проверку ЭЦП, большинство функций являются служебными и не доступны пользователям (Рисунок 3.1).

```
class Signature
{
    //Параметры соответствующме 5-му уровню криптографической стойкости настоящего стандарта
    static int r = 195, l = 1310;
    static public BigInteger p;
    static public BigInteger q;
    static public BigInteger a;

    public static void Initialize()...

    private static BigInteger BinaryOperation(BigInteger a, BigInteger b)...

    private static BigInteger BinaryOperation2(BigInteger C, BigInteger D)...

    private static BigInteger Power(BigInteger C, BigInteger K)...

    private static void GetSumm(BigInteger value, BigInteger osn, out BigInteger[] k)...

    public static void GetKeys(out BigInteger X, out BigInteger Y)...

    public static BigInteger Rand()...

    public static BigInteger CreateSignature(BigInteger x , string Path)...

    public static bool VerifySignature(BigInteger S, BigInteger y , string Path)...

}
```

Рисунок 3.1 – Функции класса Signature

Такая реализация функций необходима по той причине, что все они используются для вычисления промежуточных внутри основных функций значений (Листинг 3.1) и совершенно бесполезны для пользователя.

```
public static BigInteger CreateSignature(BigInteger x , string Path)
{
    Initialize();

    Random rnd = new Random();
    BigInteger t, S;
```

```
        BigInteger[] T, M;

        BigInteger U, V = 0;

        using (FileStream fs = System.IO.File.OpenRead(Path))
        {
        step1:
            BigInteger k = Rand();

            t = BigInteger.ModPow(a, k, p);

            GetSumm(t, BigInteger.Pow(2, 8), out T); //3

            byte[] file = new byte[fs.Length];

            fs.Read(file, 0, (int)fs.Length);

            M = new BigInteger[T.Length + file.Length];

            T.CopyTo(M, 0);
            int j = 0;
            for (int i = T.Length; i < M.Length; i++) //4
                M[i] = file[j++];

            U = Hash.ComputeHash1(M); //5
            if (U == 0) goto step1;

            V = (k - x * U) % q;//6
            if (V == 0) goto step1;

            S = U * BigInteger.Pow(2, r) + V; //7
        }

        return S;
    }
```

Листинг 3.1 – Использование Служебных функций методом CreateSignature

Вызов служебных функций Initialize() перед выполнением основных, обуславливает их правильную работу, так как при вычислений значений основных функций параметры классов постоянно меняются (Листиниг 3.2), вызов методов Initialize() возвращает все значения в исходное состояние гарантируя правильную работу классов.

```
    public static BigInteger ComputeHash(string Path)
    {
        BigInteger K, V = 0;

        using (FileStream fs = System.IO.File.OpenRead(Path))
        {
            BigInteger[] M, T, k, WW, HH, KK, h;
            BigInteger W;
            int d = 0;                                  //1 ( d := 1 )
            BigInteger[][] r = new BigInteger[8][];
            byte[] file = new byte[fs.Length];

            fs.Read(file, 0, (int)fs.Length);
```

```
                        CreateBlocks(file, out M);

                        while (d < M.Length)
                        {

                            K = M[d];                                   //2

                            V = 0;

                            GetSumm(H, 4294967296, out h, 8);
                            GetSumm(K, 4294967296, out k, 8);

                            for (int i = 0; i < 16; i++)                //3
                            {
                                if (i < 8)
                                {
                                    V += BinaryOperation2(v[i], k[i]) * BigInteger.Pow(2,
32 * i);
                                }

                                if (i >= 8)
                                {
                                    V += BinaryOperation2(v[i], h[i - 8]) *
BigInteger.Pow(2, 32 * i);
                                }
                            }

                            for (int i = 0; i < 7; i++)
                            {
                                if (i % 2 == 0)
                                {
                                    r[i] = new BigInteger[4];
                                    V = g0(V); r[i][0] = V;
                                    V = g1(V)r[i][1] = V;
                                    V = g2(V); r[i][2] = V;
                                    V = g3(V); r[i][3] = V;
                                }
                            }

                            T = new BigInteger[8];
                            for (int i = 0; i < T.Length; i++)
                            {
                                T[i] = 0;

                                if (i % 2 == 0)
                                    for (int j = 0; j < r[i].Length; j++)
                                        T[i] += r[i][j] * BigInteger.Pow(2, 512 * j);
                                else
                                    for (int j = 0; j < t[i].Length; j++)
                                        T[i] += t[i][j] * BigInteger.Pow(2, 32 * j);
                            }

                            W = BinaryOperation(K, H); // 5

                            GetSumm(W, BigInteger.Pow(2, 128), out WW, 2);

                            WW[0] = w(WW[0], T[0], T[1], T[2], T[3], T[4], T[5], T[6],
T[7]); //6

                            WW[1] = w(WW[1], T[4], T[1], T[0], T[3], T[6], T[5], T[2],
T[7]); //7
```

```
                    W = 0;

                    for (int i = 0; i < WW.Length; i++)
                        W += WW[i] * BigInteger.Pow(2, 128 * i);

                    W = E(W); //8

                    GetSumm(H, BigInteger.Pow(2, 128), out HH, 2);
                    GetSumm(K, BigInteger.Pow(2, 128), out KK, 2);

                    W = fi(HH[0], fi(HH[0], W));//9
                    W = fi(KK[0], fi(HH[0], W));//10
                    W = fi(HH[1], fi(HH[1], W));//11
                    W = fi(KK[1], fi(HH[1], W));//12

                    W = 0;

                    for (int i = 0; i < WW.Length; i++)
                        W += WW[i] * BigInteger.Pow(2, 128 * i);

                    H = W; // 13

                    d++; //14
                }

                return H % BigInteger.Pow(2, L);
            }

        }
```

Листинг 3.2 – изменение параметра H в функции ComputeHash

Для обработки ошибок, появившихся в ходе выполнения одной из функций был разработан класс исключений SignatureExeption (Листинг 3.3). Он представляет собой обычный производный класс от класса Exception, но генерируемые им исключение упрщают распознавание и обработку ошибок, появившихся в ходе выполнения функций классов Hash и Signature.

```
        class SignatureExeption : Exception
        {
            public SignatureExeption() : base() { }
            public SignatureExeption(string str) : base(str) { }
            public SignatureExeption(string str, Exception inner) : base(str, inner) {
}
            protected SignatureExeption(System.Runtime.Serialization.SerializationInfo
si, System.Runtime.Serialization.StreamingContext sc) : base(si, sc) { }

            public override string ToString()
            {
                return base.ToString();
            }
        }
```

Листинг 3.3 – Класс SignatureExeption

4. Руководство пользователя

Начало работы:

1. Для работы с приложением «Цифровая подпись» в первую очередь обратиться к центру сертификации для получения ключа и базы данных открытых ключей.

2. После получения открытого или закрытого ключа необходимо экспортировать полученную базу данных в свое приложение. Для этого нужно в приложении открыть вкладку «Менеджер ключей» (Рисунок 4.1) и нажать на кнопку «Экспорт Базы данных», после чего в открывшемся диалоге выбрать нужную базу.

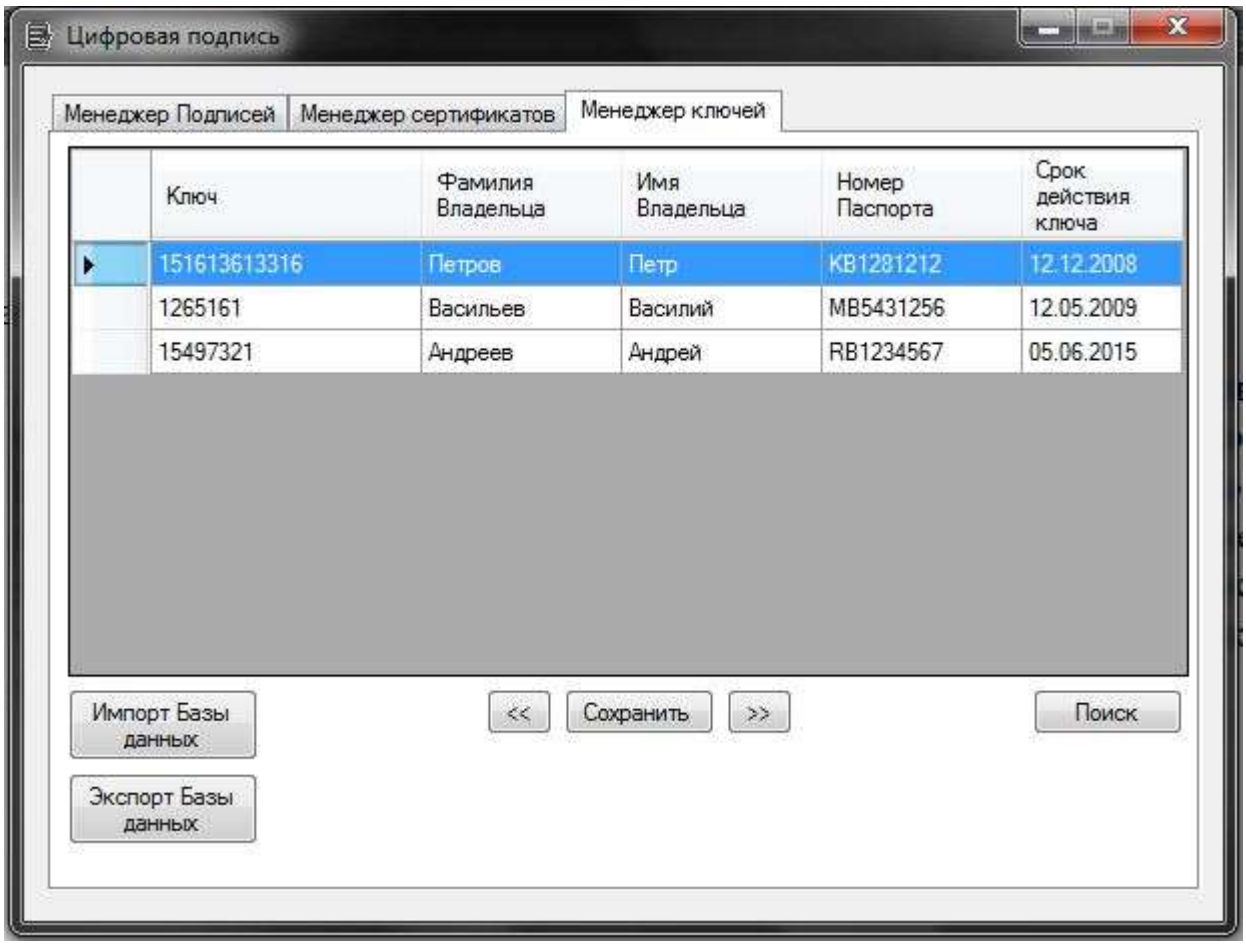

Рисунок 4.1 – Вкладка «Менеджер ключей»

3.Для того, чтобы подписать файл необходимо: перейти на вкладку «Менеджер подписей» (Рисунок 4.2) и в соответствующих полях ввести свои данные (Для облегчения ввода открытого ключа предусмотрена возможность считывания его из сертификата).

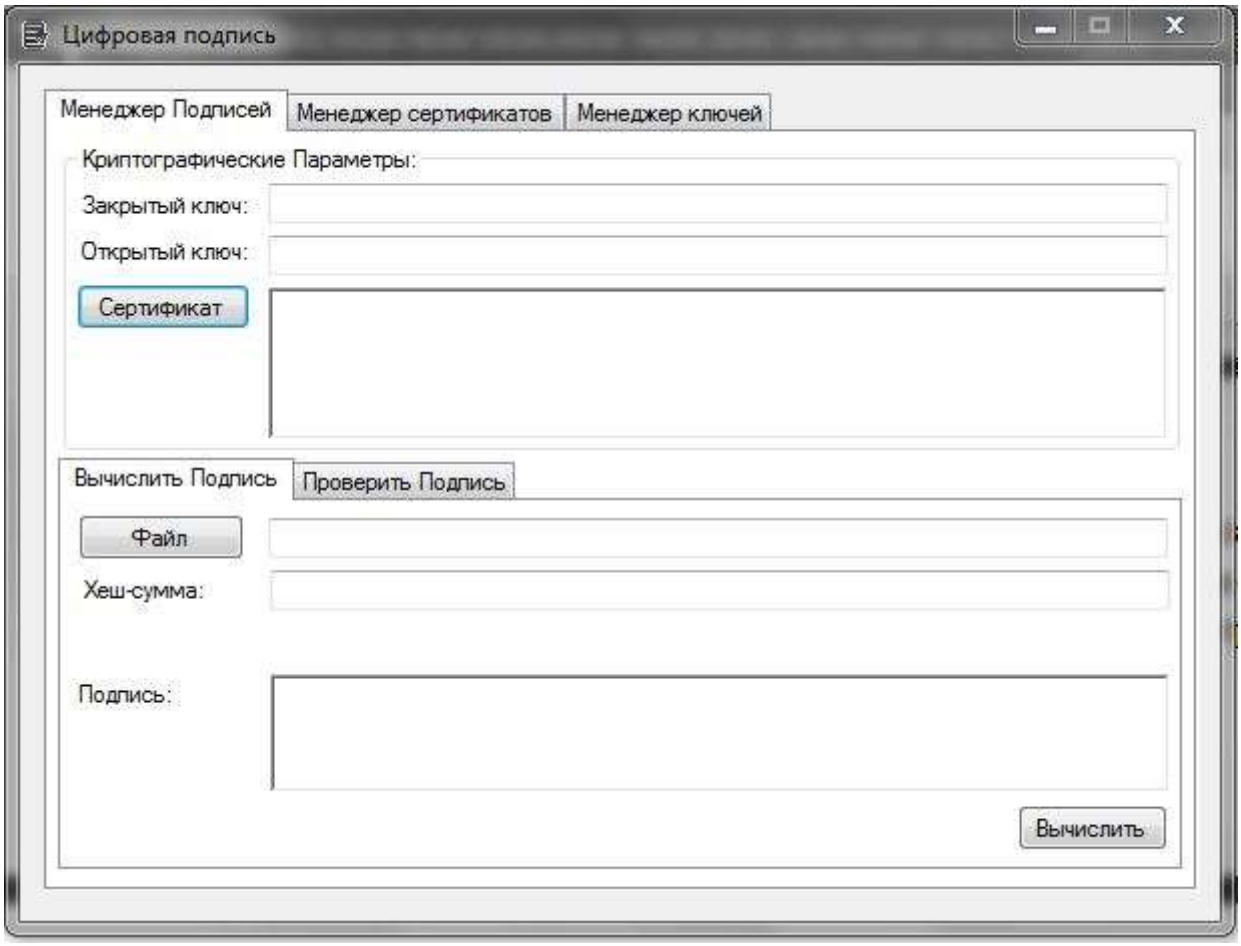

Рисунок 4.2 – Вкладка «Менеджер подписей»

4. Для того чтобы проверить подпись необходимо: перейти по внутренней вкладке, открыть файл и нажать кнопку «Проверить». Результаты проверки будут выведены на информационное окно (Рисунок 4.3).

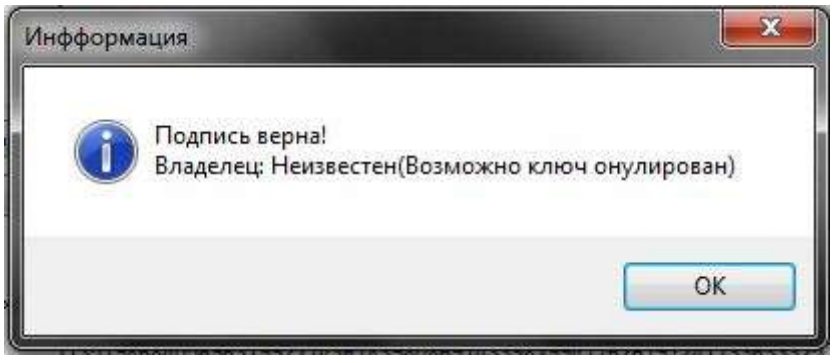

Рисунок 4.3 – Вывод результатов проверки подписи

5. Для проверки сертификата необходимо: выбрать вкладку «Менеджер сертификатов» нажать кнопку «Выбрать» и в появившемся окне (Рисунок 4.4) выбрать учетную, запись, ключ которой вы хотите проверить. Затем нажать кнопку «проверить».

Рисунок 4.4 – Окно выбора проверяемой учетной записи

5. Тестирование

Запуск программы осуществляется со значка на рабочем столе или меню «Пуск». В зависимости от открытого приложения будет открыто окно программы «Цифровая подпись» или «Центр сертификации».

Подписывая файл для того, чтобы избежать разнообразных ошибок был предусмотрен механизм уведомления пользователя о неверном вводе данных. К примеру, при открытии сертификата для поверки сразу проверяется наличие его подписи и в случае ее отсутствия выводится сообщение об ошибке (Рисунок 5.1).

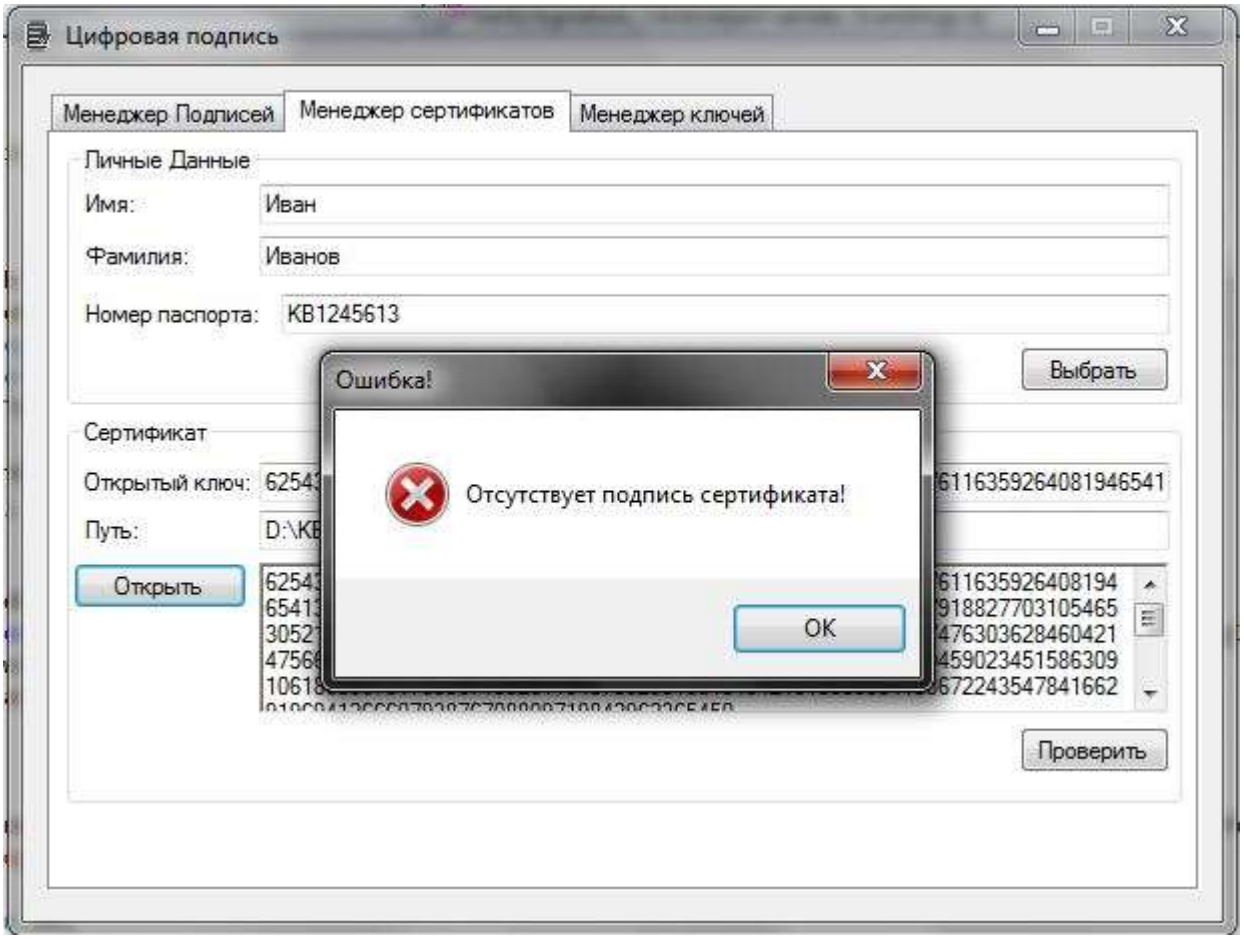

Рисунок 5.1 – Открыт сертификат не имеющий подписи

Такой же механизм реализован в проверке подписанного файла. Также реализован контроль вводимых данных, причем проверяется лишь их наличие, а контролировать данные пользователь должен самостоятельно. Например, при отсутствии открытого ключа при подписании файла вылетает сообщение об ошибке, в то время как при наличии какого-либо числа в поле ключа на соответствие закрытому ключу не проверяется (Рисунок 5.2). То есть, если пользователь введет неверный закрытый или открытый ключ – его подпись в любом случае при проверке будет признана недействительной (Рисунок 5.3).

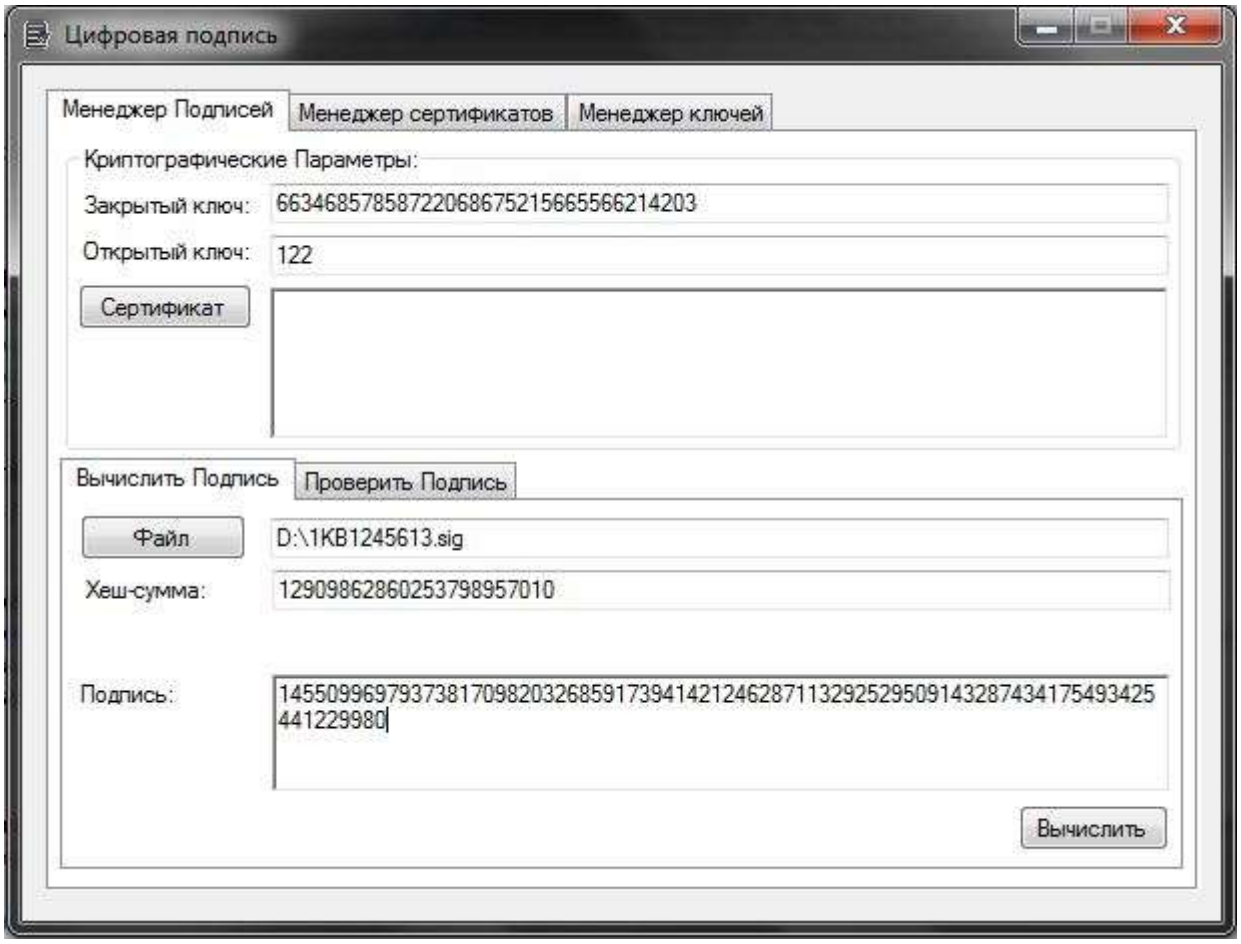

Рисунок 5.2 – Ввод неправильного открытого ключа

Как видно из рисунка файл был успешно подписан, а значит, подпись для него существует и ее можно проверить.

Проверка подписи происходит с помощью открытого ключа, который был получен из закрытого вычислением сложного математического выражения. Поэтому несоответствие любого ключа приведет к неправильному результату вычислений, необходимых для проверки подписи, а это значит, что подпись будет признана недействительной.

Также проверяется сертификат пользователя, который попытается его использовать для получения своего открытого ключа. При отсутствии на сертификате подписи центра сертификации ключей пользователь получает об этом предупреждение и если он решит его проигнорировать, то если сертификат был поврежден - подпись файла может быть признана некорректной.

Также при подписании файла, который уже подписан, предусмотрено уведомление о том, что файл имеет подпись и пользователю предоставляется возможность оставить старую подпись или выставить новую.

Такая система уведомлений позволит не только избежать ошибок связанных с некорректным вводом данных, но и понять пользователям – какие ошибки они допустили при вводе данных.

Также следует отметить, что ввиду большого количества операций, выполняемых при подсчете значения функции хэширования, программа требует достаточно много системных ресурсов. И в зависимости от размера файла скорость вычисления контрольной суммы файла может очень сильно различаться.

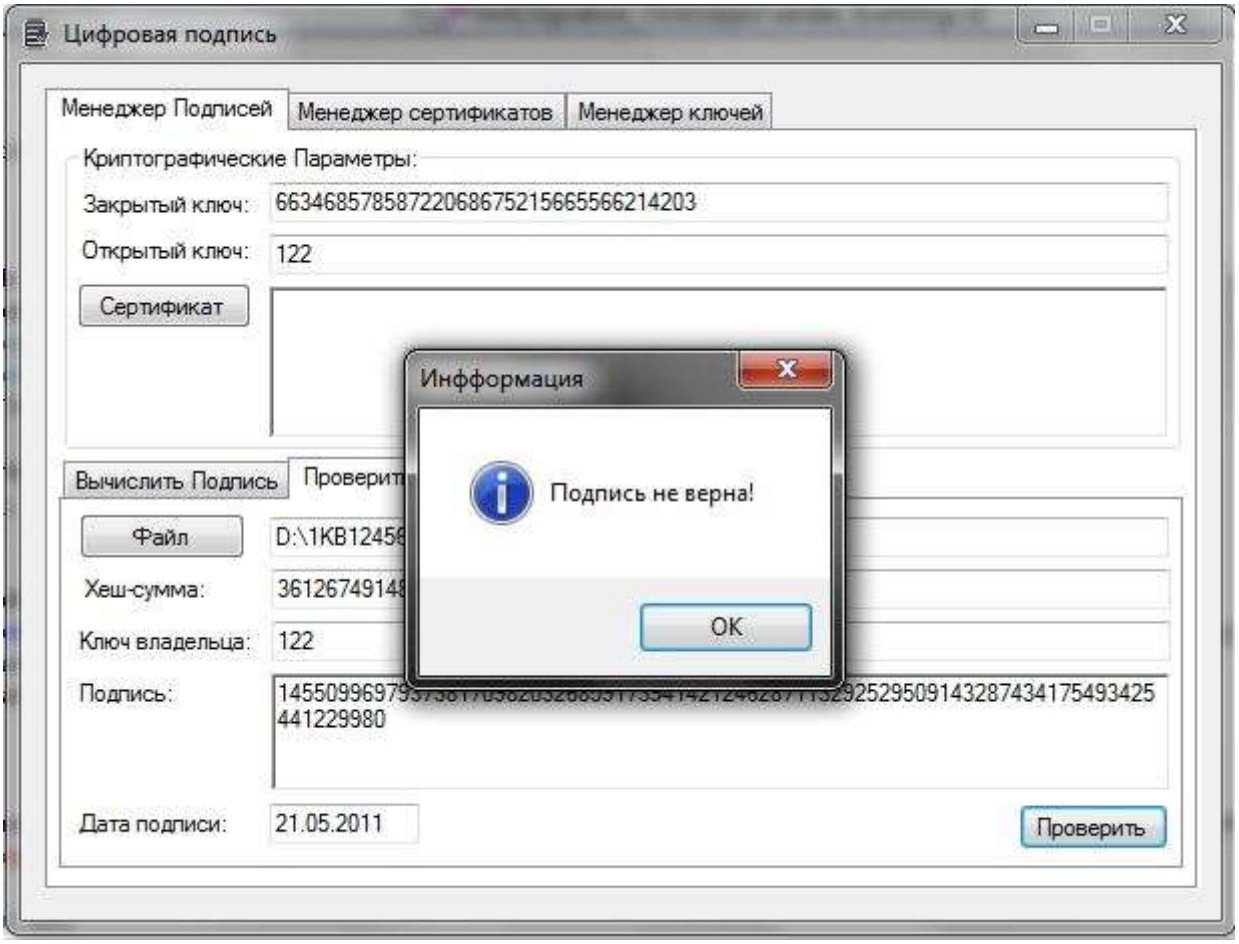

Рисунок 5.3 – Проверка подписи сделанной при помощи некорректного ключа

6. Экспериментальная часть

В данной работе был реализован алгоритм цифровой подписи по Белорусскому стандарту СТБ 1176.2-99, который также включал функцию хэширования по стандарту СТБ СТБ 1176.2-99. Был проведен опыт, для сравнения скорости вычисления значения функции хэширования реализованной в программе и американским алгоритмом MD5. Для проведения эксперимента создавался файл *.txt, в который после каждого измерения дописывалось по 100 букв, тем самым увеличивая его размер на 100 байт. Вычислялось значение функции хэширования по обоим алгоритмам, и замерялось время, необходимое на выполнение каждой функции. График зависимости скорости вычисления от размера файла представлен на графике 6.1

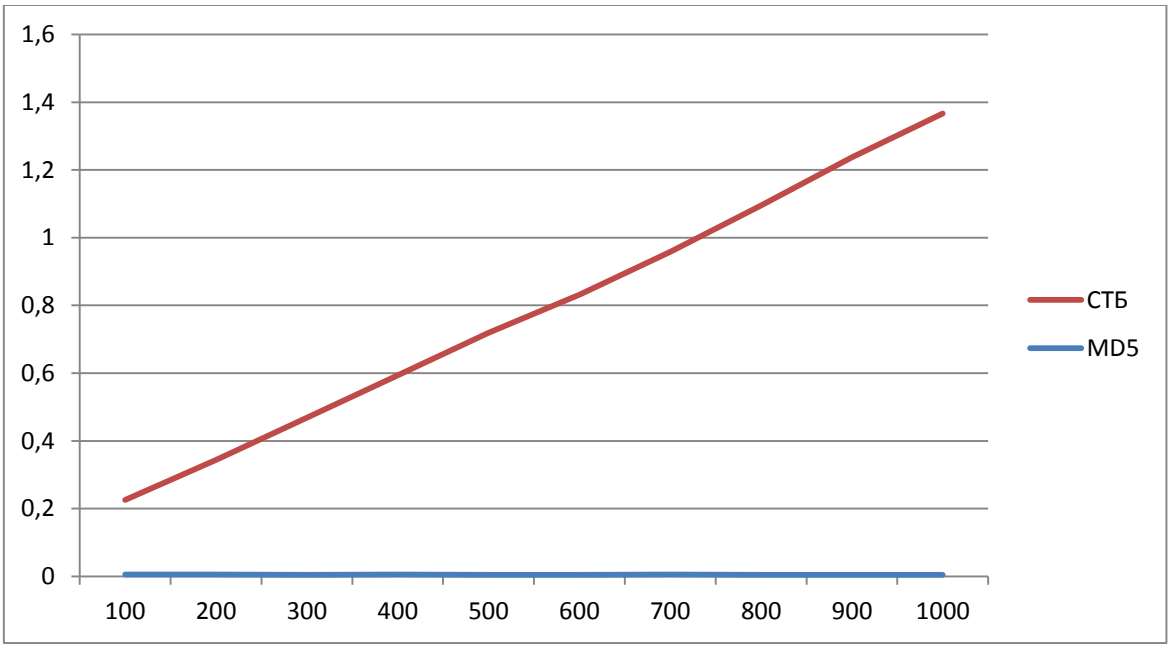

График 6.1 – Зависимости времени вычисления функций хэширования от размера фала

Как видно из Графика 6.1 скорость вычисления значения функции хэшированя линейно зависит от размера фала, что делает нецелесообразным использование этого алгоритма для вычисления контрольной суммы больших файлов.

Особенностью функции хэширования по стандарту СТБ – это длина ее значения, определяемая параметром L, величина которого определяет криптографическую стойкость функции хэширования. В данном эксперименте была изучена зависимость времени вычисления значения функции хэширования от значения параметра L. Результаты представлены на графике 6.2.

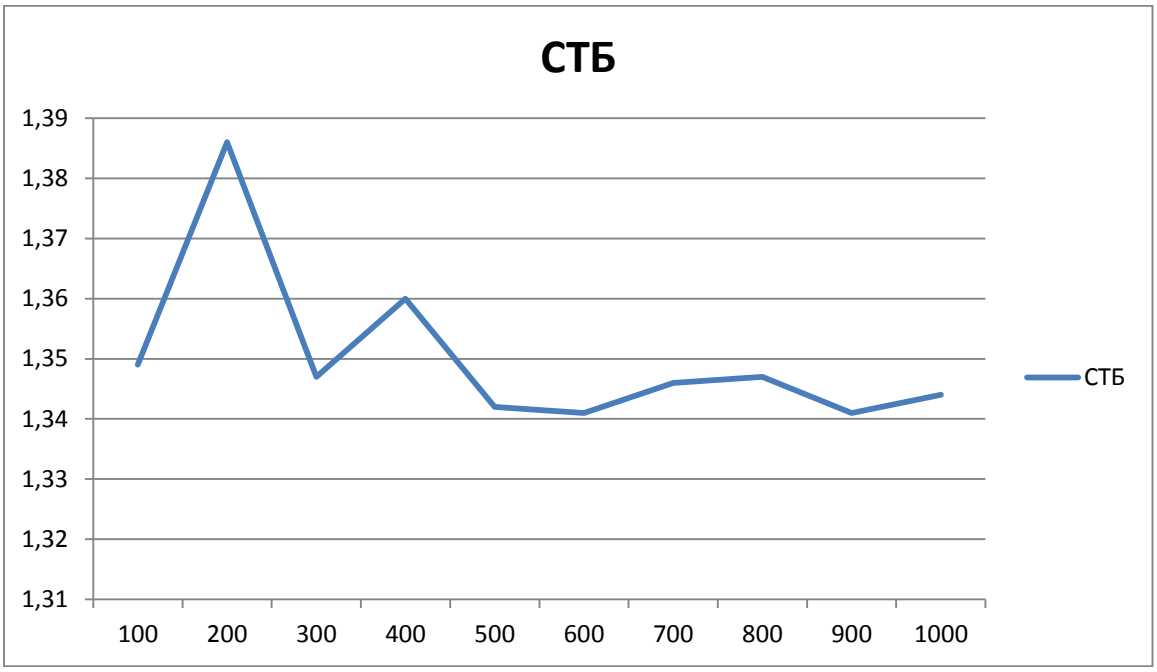

График 6.2 – Зависимость времени вычисления контрольной суммы файла размером 1000 байт от параметра L

Основываясь на полученном графике можно сказать, что увеличение параметра L функции хэширования не уменьшает скорость вычисления значения функции, следовательно, функция позволяет добиться увеличения криптографической стойкости без потери времени вычисления.

В следующем эксперименте в файле были заменены некоторые буквы и снова взяты значения функций хэширования(Рисунок 6.1 – 6.4).

Взяли исходный файл.

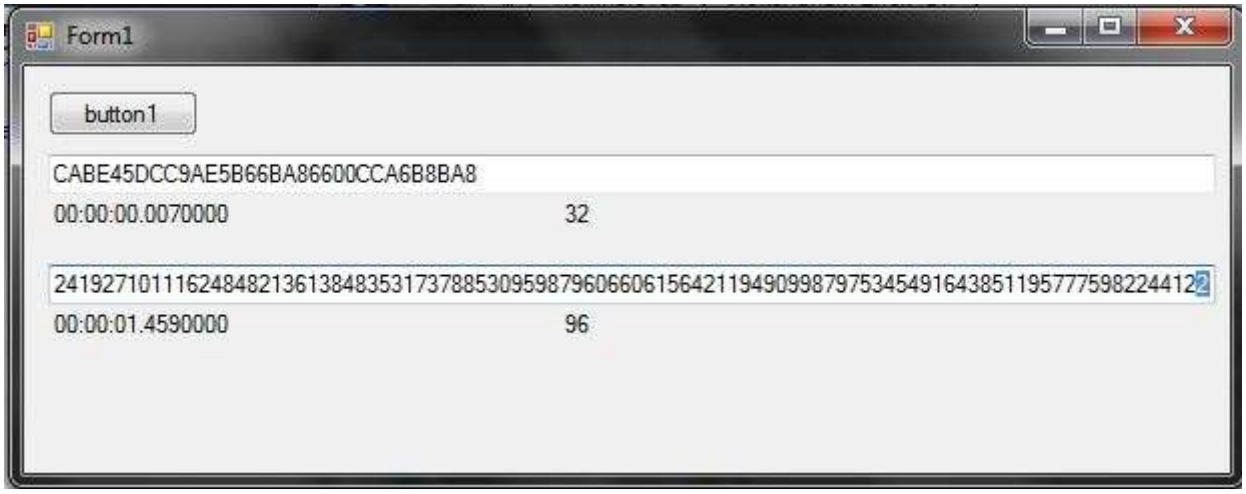

Рисунок 6.1 – Хэш-суммы файла (Вверху MD5, внизу СТБ)

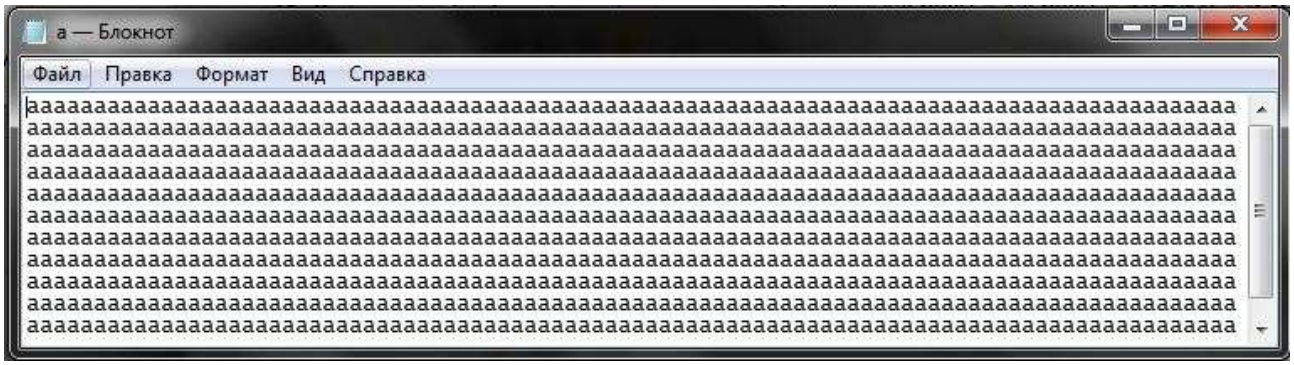

Рисунок 6.2 – Исходный файл

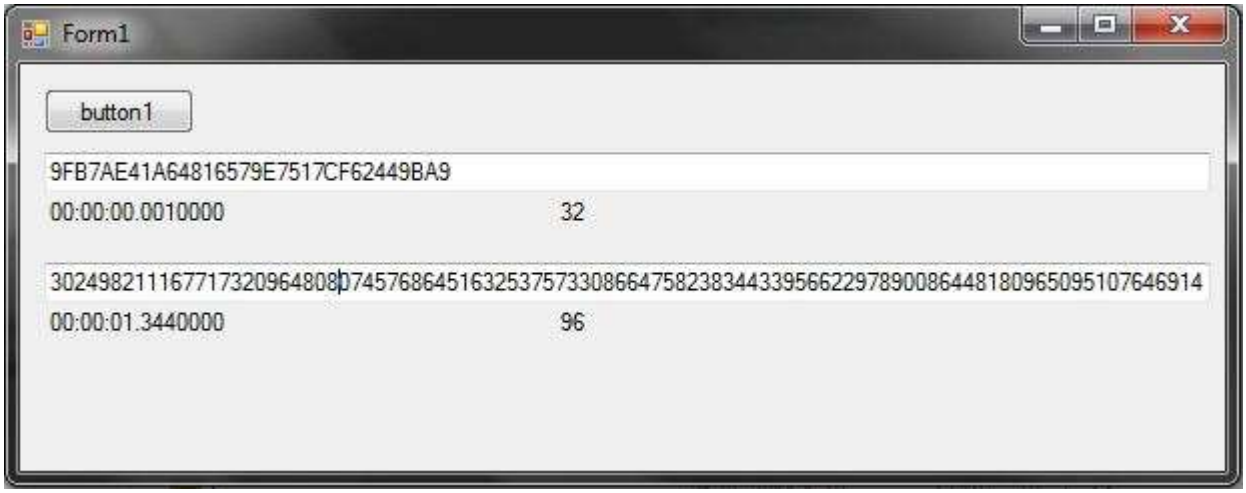

Рисунок 6.3 - Хэш-суммы измененного файла (Вверху MD5, внизу СТБ)

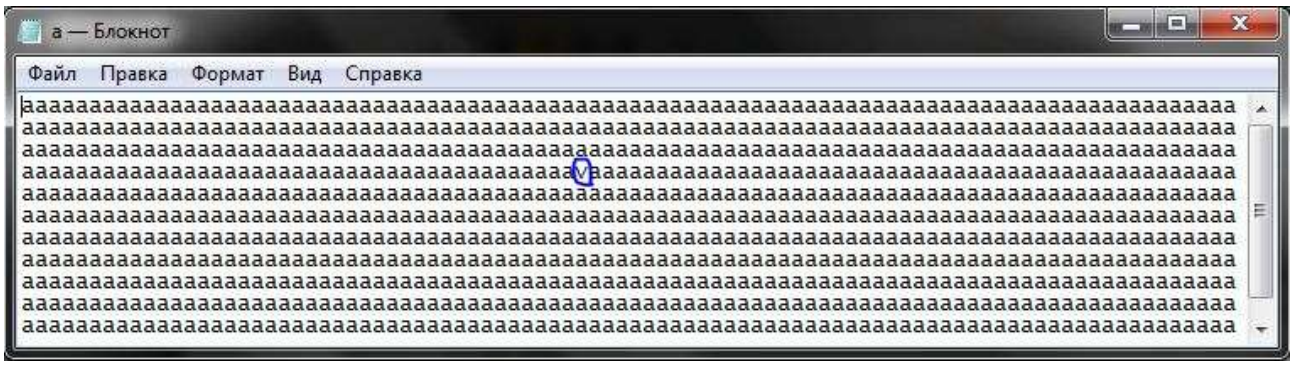

Рисунок 6.4 – Измененный файл.

Из результатов опыта видно, что разработанная функция не уступает по чувствительности MD5, следовательно, она как и MD5 может быть использована для проверки целостности документа.

Вывод: Данная функция медленнее чем MD5, но благодаря возможности менять длину хэш-суммы эта функция может оказаться безопаснее чем MD5.

Заключение

В ходе выполнения данной работы были изучены основные понятия криптографии, проведен анализ существующих схем электронной цифровой подписи. Также был проведен сравнительный анализ разработанной функции хеширования с наиболее популярным на сегодняшний день алгоритмом MD5. Итогом работы является готовое программное средство, которое может быть использовано небольшими группами людей для безопасного обмена информацией.

Список используемой литературы

1. Г. Шилдт. C# 4.0 Полное руководство. – М. : ООО «И.Д. Вильямс», 2011 – 1056с.
2. Конспект лекций по дисциплине: Объектно-ориентированное программирование. Преподаватель: Пацей. Н.В.
3. Панасенко С.П. Алгоритмы шифрования. Специальный справочник – СПб.: БХВ-Петербург, 2009. -576с.: ил.
4. Интернет сайт http://ru.wikipedia.org дата: 15.05.2011 20.20
5. Государственный стандарт Республики Беларусь СТБ 1176.2-99 Процедуры выработки и проверки электронной цифровой подаиси.
6. Государственный стандарт Республики Беларусь СТБ 1176.1-99 Функция Хэширования.

Приложение 1

СТБ 1176.2-99

ГОСУДАРСТВЕННЫЙ СТАНДАРТ РЕСПУБЛИКИ БЕЛАРУСЬ

Информационная технология
Защита информации

ПРОЦЕДУРЫ ВЫРАБОТКИ И ПРОВЕРКИ ЭЛЕКТРОННОЙ ЦИФРОВОЙ ПОДПИСИ

Інфармацыйная тэхналогія
Ахова інфармацыі

ПРАЦЭДУРЫ ВЫПРАЦОЎКІ І ПРАВЕРКІ ЭЛЕКТРОННАГА ЛІЧБАВАГА ПОДПІСУ

Издание официальное

Госстандарт
Минск

ГОСУДАРСТВЕННЫЙ СТАНДАРТ РЕСПУБЛИКИ БЕЛАРУСЬ

Информационная технология
Защита информации
ПРОЦЕДУРЫ ВЫРАБОТКИ И ПРОВЕРКИ ЭЛЕКТРОННОЙ ЦИФРОВОЙ ПОДПИСИ

Інфармацыйная тэхналогія
Ахова інфармацыі
ПРАЦЭДУРЫ ВЫПРАЦОЎКІ І ПРАВЕРКІ ЭЛЕКТРОННАГА ЛІЧБАВАГА ПОДПІСУ

Information technology
Data security
PRODUCE AND CHECK PROCEDURES OF ELECTRONIC DIGITAL SIGNATURE

Дата введения 2000-04-01

1 Область применения

Настоящий стандарт устанавливает алгоритмы и процедуры выработки и проверки электронной цифровой подписи (ЭЦП), которые применяются в криптографических методах обработки и защиты информации.

Настоящий стандарт применяется при разработке средств криптографической защиты информации в автоматизированных системах.

2 Нормативные ссылки

В настоящем стандарте использована ссылка на стандарт:
СТБ 1176.1-99 Информационная технология. Защита информации. Функция хэширования

3 Обозначения

В настоящем стандарте применяют следующие обозначения:

$Z(n)$ – множество всех неотрицательных целых чисел, меньших 2^n,
где n – натуральное число;

B_p – множество, состоящее из чисел $1, 2, ..., p-1$, где p – параметр, определяемый в разделе 7;

$c := d$ – присвоение параметру (переменной) c значения d;

$c \bmod d$ – остаток от деления c на d, где c – натуральное число или ноль, d – натуральное число;

$c^{-1} \bmod d$ – натуральное число b такое, что $b < d$ и $(c \cdot b) \bmod d = 1$, где c и d – взаимно простые числа;

$\lceil c \rceil$ – наименьшее целое число, не меньшее чем c;

$\lfloor c \rfloor$ – наибольшее целое число, не большее чем c;

$c = \sum_{i=0}^{k-1} c_i \cdot \left(2^b\right)^i$ – разложение неотрицательного целого числа c по основанию 2^b,

где k и b – натуральные числа,

c_i – целое число, $0 \le c_i < 2^b$;

$\oplus$ – бинарная операция, определенная на множестве неотрицательных целых чисел по формуле

$$d \oplus b = \sum_{i=0}^{k-1} \left((d_i + b_i) \bmod 2 \right) \cdot 2^i , \qquad (1)$$

где $d = \sum_{i=0}^{k-1} d_i 2^i$, $b = \sum_{i=0}^{k-1} b_i 2^i$, $d_0, \ldots, d_{k-1}, b_0, \ldots, b_{k-1} \in \{0,1\}$.

Издание официальное

$\circ$ – операция $\circ: B_p \times B_p \to B_p$ определяется для любых $c \in B_p$ и $d \in B_p$ по формуле

$$c \circ d = (c \cdot d \cdot (2^{l+2})^{-1}) \bmod p, \tag{2}$$

где l, p – параметры, определяемые в разделе 7;

$c^{(k)}$ – степень числа на основе операции $\circ$, определяется индуктивно по формуле

$$c^{(k)} = \begin{cases} c, & k = 1, \\ c^{(k-1)} \circ c, & k > 1, \end{cases} \tag{3}$$

где k — натуральное число;

h – функция хэширования, процедура вычисления значений которой соответствует СТБ1176.1-99. Значение исходного параметра L для процедуры вычисления значения функции хэширования выбирается равным r-1, где r – исходный параметр процедур выработки и проверки ЭЦП, определяемый в соответствии с разделом 7. Значение исходного параметра H для процедуры вычисления значения функции хэширования выбирается произвольным образом и фиксируется.

4 Общие положения

Настоящий стандарт определяет процедуры выработки и проверки ЭЦП на базе асимметричного криптографического алгоритма с применением функции хэширования, соответствующей СТБ 1176.1-99. ЭЦП позволяет обеспечить целостность сообщений (документов), передаваемых в системах обработки информации различного назначения, с гарантированной идентификацией ее автора (лица, подписавшего документ). Для выработки ЭЦП применяется личный ключ подписи, для проверки – открытый ключ проверки подписи. Используемые в процедурах ЭЦП числа p, l, q, r и a являются открытыми параметрами и генерируются в соответствии с разделом 7. Данные числа обладают следующими свойствами:

– p, q – простые числа;

– q делит p-1;

– l является длиной записи числа p в системе счисления по основанию 2;

– r является длиной записи числа q в системе счисления по основанию 2;

– порядок a в группе, определяемой множеством B_p и операцией $\circ$, равен q.

Конкретный выбор значений указанных параметров должен быть единым в рамках автоматизированной системы или выделенной группы пользователей.

Процедуры ЭЦП допускают как программную, так и аппаратную реализацию.

5 Процедура выработки ЭЦП

5.1 Исходные данные и параметры

5.1.1 В процедуре выработки ЭЦП используются следующие исходные параметры: p, l, q, r и a – числа, генерируемые процедурами, описанными в разделе 7, и являющиеся открытыми параметрами.

5.1.2 Исходными данными для процедуры выработки ЭЦП являются:

M – последовательность чисел $M=(m_1, m_2, ..., m_z)$, где $m_i \in Z(8)$ для $i=1, 2, ..., z$ и z – длина последовательности M;

x – целое число, $0 < x < q$, являющееся личным ключом подписи и хранящееся в тайне, где q – параметр, определяемый в разделе 7.

5.2 Используемые переменные

В процедуре выработки ЭЦП используются следующие переменные:

k – целое число, $1 < k < q$, которое хранится в тайне и должно быть уничтожено сразу после использования;

t – целое число, $0 < t < p$;

M_i – последовательность чисел из $Z(8)$, имеющая конечную длину;

U – целое число, $0 \le U < 2^{r-1}$;

V – целое число, $0 \le V < q$;

S – целое число, $0 < S < 2^{2r-1}$.

5.3 Алгоритм выработки ЭЦП

Алгоритм выработки ЭЦП включает в себя следующие шаги:

1 Выработать с помощью физического датчика случайных чисел или псевдослучайным методом с использованием секретных параметров число k $(1 < k < q)$;

2 $t := a^{(k)}$;

3 Представить число t в виде разложения по основанию 2^8:

$$t = \sum_{i=0}^{s-1} t_i \cdot (2^8)^i;$$

4 $M_i := (t_0, t_1, ..., t_{s-1}, m_1, ..., m_z)$;

5 $U := h(M_i)$.

Если $U = 0$, то перейти к шагу 1;

6 $V := (k - x \cdot U) \bmod q$.

Если $V = 0$, то перейти к шагу 1;

7 $S := U \cdot 2^r + V$.

ЭЦП последовательности M является число S.

6 Процедура проверки ЭЦП

6.1 Исходные данные и параметры

6.1.1 В процедуре проверки ЭЦП используются следующие исходные параметры: p, l, q, r и a – числа, генерируемые в результате выполнения процедур, описанных в разделе 7, и являющиеся открытыми параметрами.

6.1.2 Исходными данными для процедуры проверки ЭЦП являются:

M – последовательность чисел $M = (m_1, m_2, ..., m_z)$, где $m_i \in Z(8)$ для $i = 1, 2, ..., z$ и z – длина последовательности M;

S – ЭЦП последовательности M;

y – открытый ключ проверки подписи лица, подписавшего ЭЦП S последовательность M, равный $y = a^{(x)}$, где x – личный ключ подписи этого лица.

6.2 Используемые переменные

В процедуре проверки ЭЦП используются следующие переменные:

t – целое число, $0 < t < p$;

M_i – последовательность чисел из $Z(8)$, имеющая конечную длину;

W – целое число, $0 \le W < 2^{r-1}$;

V – целое число, $0 \le V < q$;

U – целое число, $0 \le U < 2^{r-1}$

6.3 Алгоритм проверки ЭЦП

Алгоритм проверки ЭЦП включает в себя следующие шаги:

1 $V := S \bmod 2^r$;

2 $U := (S - V) / 2^r$;

3 Проверить условия:

$0 < U < 2^{r-1}$ и $0 < V < q$.

Если хотя бы одно из этих условий не выполнено, то ЭЦП считается недействительной и работа алгоритма завершается;

4 $t := a^{(V)} \circ y^{(U)}$;

5 Представить число t в виде разложения по основанию 2^s

$$t = \sum_{i=0}^{n-1} t_i \cdot (2^s)^i;$$

6 $M_t := (t_0, t_1, ..., t_{n-1}, m_1, ..., m_z)$;

7 $W := h(M_t)$;

8 Проверить условие: $W = U$.

При совпадении значений W и U принимается решение о том, что ЭЦП была создана с помощью личного ключа подписи x, связанного с открытым ключом проверки подписи y, а также ЭЦП и последовательность M не были изменены с момента их создания. В противном случае подпись считается недействительной.

7 Процедуры получения параметров

7.1 Выбор r и l

Выбор значений параметров r и l производится в зависимости от необходимого уровня криптографической стойкости в соответствии с таблицей 7.1 (уровни криптографической стойкости приведены в порядке возрастания стойкости).

Таблица 7.1

Уровень криптографической стойкости	r	l
1	143	638
2	154	766
3	175	1022
4	182	1118
5	195	1310
6	208	1534
7	222	1790
8	235	2046
9	249	2334
10	257	2462

7.2 Процедура генерации параметров p и q

7.2.1 Исходные данные

Процедура генерации параметров p и q использует следующие исходные данные:

l, r – целые числа, выбираемые в соответствии с 7.1;

$z_1, z_2, ..., z_{31}$ – инициализирующие значения датчика случайных чисел, описанного в 7.2.2, $0 < z_i < 65257$ для $i = 1, 2, ..., 31$. Инициализирующие значения датчика случайных чисел выбираются с помощью физического датчика случайных чисел или псевдослучайным методом и сохраняются для проверки того, что параметры p и q были сгенерированы с использованием процедуры настоящего стандарта;

$y_1, y_2, y_3, ...$ – последовательность целых чисел, $0 \le y_i < 256$ для $i = 1, 2, 3, ...$, порождаемая датчиком случайных чисел, описанным в 7.2.2, с использованием инициализирующих значений $z_1, z_2, ..., z_{31}$;

$p_1, ..., p_{\lceil l/4 \rceil}$ – первые $\left\lceil \frac{l}{4} \right\rceil$ нечетных простых чисел, занумерованных в порядке возрастания.

7.2.2 Датчик случайных чисел

Датчик случайных чисел порождает последовательность целых чисел $y_1, y_2, y_3 ...$, определенную в 7.2.1. В качестве инициализирующего значения датчика используются значения $z_1, z_2, ..., z_{31}$, определенные в 7.2.1.

Реализация датчика случайных чисел состоит в пошаговом вычислении следующих последовательностей целых чисел:

– $z_{32}, z_{33}, z_{34}, ...,\ 0 \le z_i < 65257$ для $i = 32, 33, 34, ...$;

– $v_1, v_2, v_3, ...,\ 0 \le v_i < 65536$ для $i = 1, 2, 3, ...$;

– $w_0, w_1, w_2, ...,\ 0 \le w_i < 65536$ для $i = 0, 1, 2, ...$;

– $u_1, u_2, u_3, ...,\ 0 \le u_i < 65536$ для $i = 1, 2, 3, ...$.

Для вычисления данных последовательностей необходимо выполнить следующие шаги:

1 $z_i := (z_{i-31} - z_{i-21}) \bmod 65257,\ i = 32, 33, 34, ...$;

2 $v_i := \sum_{j=1}^{i} z_j \bmod 65536,\ i = 1, 2, 3, ...$;

3 $w_0 := 0,\ w_i := \left[z_{i+20} + \left\lfloor \frac{w_{i-1}}{2} \right\rfloor + 32768 (w_{i-1} \bmod 2) \right] \bmod 65536,$ где $i = 1, 2, 3, ...$;

4 $u_i := w_i \oplus v_i,\ i = 1, 2, 3, ...$;

5 $y_i := \left(u_{i+256} + \left\lfloor \frac{u_{i+255}}{255} \right\rfloor \right) \bmod 256,\ i = 1, 2, 3, ...$.

7.2.3 Переменные алгоритма

Алгоритм генерации чисел p и q использует следующие переменные:

$i, j, t, d_0, d_1, ..., d_t, k_0, k_1, ..., k_t$ – целые числа, большие либо равные нулю и меньшие l ;

$s, r_0, r_1, ..., r_s$ – целые числа, большие либо равные нулю и меньшие r ;

u – целое число, $0 \le u \le 4l + 1$;

w – целое число, $0 < w \le \lceil l / 4 \rceil$;

$M, N, g_0, ..., g_t, p$ – *целые числа, большие либо равные нулю и меньшие* 2^{l+1};

$f_0, ..., f_s, e, q$ – целые числа, большие либо равные нулю и меньшие 2^{r+1};

$A_1, ..., A_w$ и $C_1, ..., C_w$ – целые числа, большие либо равные нулю и меньшие $p_{\lceil l/4 \rceil}$.

7.2.4 Алгоритм генерации параметров p и q

Алгоритм генерации чисел p и q состоит из следующих шагов:

1 $e := 1$;

2 Выбрать последовательности целых чисел $d_0, d_1, ..., d_t$ и $r_0, r_1, ..., r_s$ так, чтобы были выполнены следующие условия:

$$\frac{l}{2} \le d_0 \le \frac{7l}{8} - r,$$

$$d_0 > d_1 > ... > d_t,$$

$$r = r_0 > r_1 > ... > r_s,$$

$$\frac{5}{4} d_{i+1} + 4 < d_i \le 2d_{i+1}, \text{ где } i = 0, 1, ... t-1, \ 16 < d_t \le 32,$$

$$\frac{5}{4} r_{i+1} < r_i \le 2r_{i+1}, \text{ где } i = 0, 1, ... s-1, \ 16 < r_s \le 32.$$

Сохранить выбранные последовательности для проверки того, что параметры p и q были сгенерированы с использованием процедуры настоящего стандарта;

3 Для всех $j = 0, 1, ..., t$ присвоить $k_j := \left\lceil \frac{d_j}{8} \right\rceil$;

4 $M := 2^{d_t - 1} + \left(\sum_{j=0}^{k_t - 1} y_{j+e} \cdot 2^{8j} \right) \bmod 2^{d_t - 1}$, $e := e + k_t$;

5 Найти наименьшее простое число g_t такое, что $M \le g_t < 2^{d_t}$. Если такого простого числа не существует, то перейти к шагу 4;

6 $i := t - 1$;

7 $w := \left\lceil \frac{d_i}{4} \right\rceil$;

8 $M := 2^{d_i - 2} + \left(\sum_{j=0}^{k_i - 1} y_{j+e} \cdot 2^{8j} \right) \bmod 2^{d_i - 2}$, $e := e + k_i$;

9 $N := \left\lceil \frac{M}{g_{i+1}} \right\rceil$;

10 $g_i := 2g_{i+1}N + 1$; $u := 0$;

11 Если $g_i \ge 2^{d_i}$, то перейти к шагу 8;

12 Для всех $j = 1, 2, ..., w$ присвоить $A_j := g_i \bmod p_j$, $C_j := 2g_{i+1} \bmod p_j$;

13 Если существует натуральное j такое, что $j \le w$ и $A_j = 0$, то перейти к шагу 15;

14 Если $2^{2g_{i+1}N} \bmod g_i = 1$ и $2^{2N} \bmod g_i \ne 1$, то перейти к шагу 22;

15 $u := u + 1$;

16 Если $u \ge 4d_i$, то $i := i + 1$ и при $i < t$ перейти к шагу 7, при $i = t$ перейти к шагу 4;

17 $g_i := g_i + 2g_{i+1}$;

18 Если $g_i \ge 2^{d_i}$, то перейти к шагу 8;

19 Для всех $j = 1, 2, ..., w$ присвоить $A_j := (A_j + C_j) \bmod p_j$;

20 $N := N + 1$;

21 Перейти к шагу 13;

22 $i := i - 1$;

23 Если $i \ge 0$, то перейти к шагу 7;

24 Для всех $j = 0, 1, ..., s$ присвоить $k_j := \left\lceil \frac{r_j}{8} \right\rceil$;

25 $M := 2^{r_s - 1} + \left(\sum_{j=0}^{k_s - 1} y_{j+e} \cdot 2^{8j} \right) \bmod 2^{r_s - 1}$, $e := e + k_s$;

26 Найти наименьшее простое число f_s такое, что $M \le f_s < 2^{r_s}$. Если такого простого числа не существует, то перейти к шагу 25;

27 $i := s - 1$;

28 $w := \left\lceil \frac{r_i}{4} \right\rceil$;

29 $M := 2^{r_i - 2} + \left(\sum_{j=0}^{k_i - 1} y_{j+e} \cdot 2^{8j} \right) \bmod 2^{r_i - 2}$, $e := e + k_i$;

30 $N := \left\lceil \frac{M}{f_{i+1}} \right\rceil$;

31 $f_i := 2f_{i+1}N + 1$, $u := 0$;

32 Если $f_i \ge 2^{r_i}$, то перейти к шагу 29;

33 Для всех $j = 1, 2, ..., w$ присвоить $A_j := f_i \bmod p_j$, $C_j := 2f_{i+1} \bmod p_j$;

34 Если существует натуральное j такое, что $j \le w$ и $A_j = 0$, то перейти к шагу 36;

35 Если $2^{2f_{i+1}N} \bmod f_i = 1$ и $2^{2N} \bmod f_i \ne 1$, то перейти к шагу 43;

36 $u := u + 1$;

37 Если $u \ge 4r_i$, то $i := i + 1$ и при $i < s$ перейти к шагу 28, при $i = s$ перейти к шагу 25;

38 $f_i := f_i + 2f_{i+1}$;

39 Если $f_i \ge 2^{r_i}$, то перейти к шагу 29;

40 Для всех $j = 1, 2, ..., w$ присвоить $A_j := (A_j + C_j) \bmod p_j$;

41 $N := N + 1$;

42 Перейти к шагу 34;

43 $i := i - 1$;

44 Если $i \ge 0$, то перейти к шагу 28;

45 $k_i := \left\lceil \frac{l}{8} \right\rceil$;

46 $M := 2^{l-2} + \left(\sum_{j=0}^{k_i - 1} y_{j+e} \cdot 2^{8j} \right) \bmod 2^{l-2}$, $e := e + k_i$;

47 $N := \left\lceil \frac{M}{g_0 f_0} \right\rceil$;

48 $p := 2g_0 f_0 N + 1$, $u := 0$, $w := \left\lceil \frac{l}{4} \right\rceil$;

49 Если $p \geq 2^l$, то перейти к шагу 46;

50 Для всех $j = 1, 2, ..., w$ присвоить $A_j := p \bmod p_j$, $C_j := 2g_0 f_0 \bmod p_j$;

51 Если существует натуральное j такое, что $j \leq w$ и $A_j = 0$, то перейти к шагу 53;

52 Если $2^{2g_0 f_0 N} \bmod p = 1$ и $2^{2f_0 N} \bmod p \neq 1$, то перейти к шагу 60;

53 $u := u + 1$;

54 Если $u \geq 4l$, то перейти к шагу 24;

55 $p := p + 2g_0 f_0$;

56 Если $p \geq 2^l$, то перейти к шагу 46;

57 Для всех $j = 1, 2, ..., w$ присвоить $A_j := (A_j + C_j) \bmod p_j$;

58 $N := N + 1$;

59 Перейти к шагу 51;

60 $q := f_0$;

61 Конец работы алгоритма.

Значениями параметров p и q являются значения переменных p и q соответственно после завершения работы алгоритма.

7.3 Процедура генерации параметра a

7.3.1 Исходные данные

Процедура генерации a использует в качестве исходных данных параметры p и q, процедура генерации которых описана в 7.2.

7.3.2 Переменные алгоритма

Алгоритм генерации параметра a использует переменные d, a – целые числа, $0 < d < p$, $0 < a < p$.

7.3.3 Алгоритм генерации параметра a

Алгоритм генерации параметра a состоит из следующих шагов:

1 Сгенерировать с помощью физического датчика случайных чисел или псевдослучайным методом число d;

2 $a := d^{\left(\frac{p-1}{q}\right)}$;

3 Если $a = 2^{l+2} \bmod p$, то перейти к шагу 1;

4 Сохранить d для проверки того, что параметр a сгенерирован в соответствии с процедурами настоящего стандарта;

5 Конец работы алгоритма.

Значение переменной a после завершения работы алгоритма является искомым значением параметра a.

Приложение 2

СТБ 1176.1-99

ГОСУДАРСТВЕННЫЙ СТАНДАРТ РЕСПУБЛИКИ БЕЛАРУСЬ

Информационная технология
Защита информации

ФУНКЦИЯ ХЭШИРОВАНИЯ

Інфармацыйная тэхналогія
Ахова інфармацыі

ФУНКЦЫЯ ХЭШЫРАВАННЯ

Издание официальное

Госстандарт
Минск

ГОСУДАРСТВЕННЫЙ СТАНДАРТ РЕСПУБЛИКИ БЕЛАРУСЬ

Информационная технология
Защита информации
ФУНКЦИЯ ХЭШИРОВАНИЯ

Інфармацыйная тэхналогія
Ахова інфармацыі
ФУНКЦЫЯ ХЭШЫРАВАННЯ

Information technology
Data security
HASHING FUNCTION

Дата введения 2000-04-01

1 Область применения

Настоящий стандарт устанавливает алгоритм и процедуру вычисления значения функции хэширования, которые применяются в криптографических методах обработки и защиты информации, в том числе для реализации процедур электронной цифровой подписи при передаче, обработке и хранении информации.

Настоящий стандарт применяется при разработке средств криптографической защиты информации в автоматизированных системах.

2 Нормативные ссылки

В настоящем стандарте использована ссылка на стандарт:

СТБ 1176.2-99 – Информационная технология. Защита информации. Процедуры выработки и проверки электронной цифровой подписи

3 Обозначения

В настоящем стандарте применяют следующие обозначения:

$Z(n)$ – множество всех неотрицательных целых чисел, меньших 2^n,

где n – натуральное число;

$a = \sum_{i=0}^{k-1} a_i \cdot (2^b)^i$ – разложение неотрицательного целого числа a по основанию 2^b,

где k и b – натуральные числа,

a_i – целое число, $0 \le a_i < 2^b$;

$\oplus$ – бинарная операция, определенная на множестве неотрицательных целых чисел по формуле

$$a \oplus b = \sum_{i=0}^{k-1} ((a_i + b_i) \bmod 2) \cdot 2^i, \qquad (1)$$

где $a = \sum_{i=0}^{k-1} a_i 2^i$, $b = \sum_{i=0}^{k-1} b_i 2^i$, $a_0, \ldots, a_{k-1}, b_0, \ldots, b_{k-1} \in Z(1)$;

Издание официальное

$\boxplus'$ – сложение неотрицательных целых чисел по модулю 2^{32};
$\times$ – декартово произведение множеств;

η^i – i-ая степень преобразования η. Степень преобразования η определяется индуктивно по формуле

$$\eta^i = \begin{cases} \eta, & i = 1, \\ \eta(\eta^{i-1}), & i > 1; \end{cases} \qquad (2)$$

H – целое число, $0 \le H < 2^{256}$, такое, что

$$H = \sum_{i=0}^{7} h_i \cdot (2^{32})^i = \sum_{i=0}^{1} H_i \cdot (2^{128})^i, \qquad (3)$$

где h_i – целое число, $0 \le h_i < 2^{32}$,

H_i – целое число, $0 \le H_i < 2^{128}$;

L – натуральное число, $142 \le L \le 256$;
M – последовательность чисел из $Z(8)$, имеющая конечную длину;
h – функция хэширования, отображающая последовательность чисел M в число $h(M)$;
$h(M)$ – неотрицательное целое число, определяющее значение функции хэширования h;
$c := d$ – присвоение параметру (переменной) c значения d.

4 Общие положения

Настоящий стандарт определяет функцию хэширования, которая позволяет осуществлять проверку целостности сообщений (документов), передаваемых в системах обработки информации различного назначения, с гарантированной достоверностью. Данная функция хэширования используется в процедурах выработки и проверки электронной цифровой подписи в соответствии с СТБ 1176.2-99.

Для осуществления возможности контроля целостности данных необходимо вычислить от них значение функции хэширования и хранить его достоверным способом. При необходимости проверки целостности данных значение функции хэширования от этих данных вычисляется заново. В случае, если вновь вычисленное значение совпало с достоверно хранимым, целостность данных подтверждается. В противном случае – целостность нарушена. Параметрами алгоритма вычисления функции хэширования являются числа L и H, которые выбираются в соответствии с пунктом 5.1. Конкретный выбор значений параметров должен быть единым в рамках автоматизированной системы или выделенной группы пользователей.

Процедура вычисления значения функции хэширования допускает как программную, так и аппаратную реализацию.

5 Процедура вычисления функции хэширования

5.1 Исходные данные и параметры

Исходными данными для процедуры вычисления значения функции хэширования является последовательность чисел $M = (m_1, m_2, ..., m_z)$, где $m_i \in Z(8)$ для $i=1,2,...,z$ и z – длина последовательности M.

Параметрами являются числа H и L. Значение H выбирается произвольным образом и фиксируется. Значение L влияет на криптографическую стойкость описанного в настоящем стандарте алгоритма вычисления значения функции хэширования и выбирается максимально возможным.

5.2 Используемые преобразования

5.2.1 Преобразования $\rho_i : Z(512) \to Z(512)$, $i = \overline{0,3}$ для любого $x \in Z(512)$, где $x = \sum_{j=0}^{15} x_j \cdot (2^{32})^j$, выполняются следующим образом:

$$\rho_0(x) = \sum_{j=0}^{14} x_{j+1} \cdot (2^{32})^j + ((x_{15} \oplus x_{13} \oplus x_3 \oplus x_0) \boxplus C_0) \cdot (2^{480}), \quad (4)$$

$$\rho_1(x) = \sum_{j=0}^{14} x_{j+1} \cdot (2^{32})^j + ((x_{15} \oplus x_2 \oplus x_0) \boxplus C_1) \cdot (2^{480}), \quad (5)$$

$$\rho_2(x) = \sum_{j=0}^{14} x_{j+1} \cdot (2^{32})^j + ((x_9 \oplus x_4 \oplus x_0) \boxplus C_2) \cdot (2^{480}), \quad (6)$$

$$\rho_3(x) = \sum_{j=0}^{14} x_{j+1} \cdot (2^{32})^j + ((x_{15} \oplus x_8 \oplus x_0) \boxplus C_3) \cdot (2^{480}), \quad (7)$$

где C_0 = 2BDA732E,

C_1 = 3920FE85,

C_2 = BC1641F9,

C_3 = 75FE243B (в шестнадцатеричной системе счисления).

5.2.2 Преобразование $\omega : Z(128) \times Z(2048) \times Z(2048) \times Z(2048) \times Z(2048) \times Z(2048) \times Z(2048) \times Z(2048) \to Z(128)$ для любого $X \in Z(128)$, где $X = \sum_{i=0}^{1} X_i \cdot (2^{64})^i$ и $X_i = \sum_{j=0}^{7} x_{ij} \cdot (2^8)^j$, $i = 0,1$, и для любых $Y_i \in Z(2048)$, где $Y_i = \sum_{j=0}^{255} y_{ij} \cdot (2^8)^j$ и $i = \overline{0,7}$ определяется следующим алгоритмом, использующим переменные m и P, где m – целое число, $0 \le m < 33$, $P \in Z(64)$, $P = \sum_{i=0}^{63} p_i \cdot 2^i$:

1 $m := 0$;

2 $P := \sum_{i=0}^{7} y_{i x_{i1}} \cdot (2^8)^i$;

3 $P := \sum_{i=0}^{60} p_i \cdot 2^{i+3} + p_{63} \cdot 2^2 + p_{62} \cdot 2 + p_{61}$;

4 $P := P \oplus X_1$;

5 $X_1 := X_0$, $X_0 := P$;

6 $m := m+1$;

7 Проверить условие $m=32$.

Если условие выполнено, то перейти к шагу *8*. Если условие не выполнено – к шагу *2*;

8 Конец работы алгоритма.

Значением преобразования $\omega(X, Y_0, Y_1, Y_2, Y_3, Y_4, Y_5, Y_6, Y_7)$ является значение переменной X после завершения работы алгоритма.

5.2.3 Преобразование $\xi: Z(256) \to Z(256)$ для любого $X \in Z(256)$ такого, что $X = \sum_{i=0}^{7} x_i \cdot (2^{32})^i$ выполняется по формуле

$$\xi(X) = \sum_{i=1}^{7} x_{i-1} \cdot (2^{32})^i + (x_0 \oplus x_2 \oplus x_4 \oplus x_7). \quad (8)$$

5.2.4 Преобразование $\varphi: Z(128) \times Z(256) \to Z(256)$ для любого $X \in Z(128)$ такого, что $X = \sum_{i=0}^{3} x_i \cdot (2^{32})^i$, и для любого $Y \in Z(256)$ такого, что $Y = \sum_{i=0}^{7} y_i \cdot (2^{32})^i$ определяется следующим алгоритмом, использующим переменную $S \in Z(128)$:

1 $S := \sum_{i=0}^{3} y_i \cdot (2^{32})^i$;

2 $Y := \sum_{i=4}^{7} y_i \cdot (2^{32})^i + \sum_{i=0}^{3} (x_i \oplus' y_i) \cdot (2^{32})^i$;

3 $Y := S \cdot (2^{32})^4 + \sum_{i=0}^{3} (y_i \oplus y_{i+4}) \cdot (2^{32})^i$;

4 Конец работы алгоритма.

Значением преобразования $\varphi(X,Y)$ является значение Y после завершения работы алгоритма.

5.3 Дополнение и разбиение на блоки

Если длина последовательности M не кратна 32, то она дополняется минимальным количеством нулей таким образом, что $M = (m_1, m_2, ..., m_{l+k}) = (m_1, m_2, ..., m_l, 0, ..., 0)$ для некоторого неотрицательного целого числа k и $l+k$ кратно 32.

Вычисляется значение $n=(l+k)/32$, причем k полагается равным нулю, если l делится на 32. По последовательности M формируются числа $M_1, M_2, ..., M_n$ таким образом, что $M_i = m_{(i-1)32+1} + m_{(i-1)32+2} * 2^8 + ... + m_{i32} * 2^{248}$ для $i=1,2,...,n$. Значение числа M_{n+1} устанавливается равным l.

5.4 Переменные и начальные значения

Процедура вычисления значения функции хэширования использует следующие переменные:

$T_i - T_i \in Z(2048)$,

где $T_i = \begin{cases} \sum_{j=0}^{3} r_{ij} \cdot (2^{512})^j, i = 0,2,4,6, \\ \sum_{j=0}^{63} t_{ij} \cdot (2^{32})^j, i = 1,3,5,7. \end{cases}$

Начальные значения коэффициентов t_{ij} в шестнадцатеричной системе счисления определяются таблицей 5.1.

$V - V \in Z(512)$, $V = \sum_{i=0}^{15} v_i (2^{32})^i$. Начальные значения коэффициентов v_i в шестнадцатеричной системе счисления определяются таблицей 5.2.

Таблица 5.1 – **Значения коэффициентов t_{ij} в шестнадцатеричной системе счисления**

Значение индекса j	Значение индекса i			
	1	3	5	7
0	AA2AA82E	4DCDCF4F	5557455D	B2B03212
1	8A0A088E	69E9EB6B	4715547C	921A13BF
2	A222A026	65E5E767	5644465C	BAB83A18
3	82020086	41C1C343	1416177D	9A101BB7
4	AE2CAC28	49C9CB4B	51534159	B6B43616
5	8C0E0C88	6DEDEF6F	43115078	961E17BB
6	A624A420	61E1E363	7072607A	BEBC1C3E
7	84061E9A	45C5C747	F8FAF9D1	9E143F11
8	AB2BA92F	5DDDCC4C	5F4D4F05	B3B13303
9	8B0B098F	79F9E868	1D1F5EF4	931902AE
10	A323A127	F17140C0	585A4852	BDAF3715
11	B32133A7	D55564E4	4A185BF1	9F0706AA
12	8D0F9F1B	51D1C242	494B1913	B5A73505
13	AF2D3FBB	75F5EC6C	1B091AF0	970E04B9
14	07870583	FD7D4ECE	7E6C6E74	ABA90131
15	17859D19	D95960EA	F6E4F50D	9B0A3057
16	BA3AB83E	5CDCDE5E	7577657F	A2A02200
17	98381ABE	6AF8FA78	6735764C	900853AD
18	B231B025	66E6E062	64626668	A8FA380C
19	806312A5	48C88C0C	3430710F	98520BA3
20	AD3DBF29	7AA8AA28	7332616A	A6A43455
21	89B94B6F	8E0E4ACA	63313840	943C1F82
22	BD3CEF6A	53D3C444	420710D8	ACFE5623
23	CFBC1532	FF7F6EEE	DADCD5FB	9C263D24
24	EB3BE96D	1C9CDF5F	FDFFEDF7	A1217391
25	9B396BCD	2AB8FB7B	EFBDDE36	839D502A
26	B130E336	E2700181	E8FCEAE6	FC2E7680
27	E11003B6	9E0D7CAD	B8FEDD33	8ED2473B
28	9E1C9C18	72F48357	E9BAEBA8	954599F6
29	EACA4A6E	46D0A92D	D9DBF212	A58A2C74

Окончание таблицы 5.1

Значение индекса *j*	Значение индекса *i*			
	1	3	5	7
30	4EDF4C99	8D5B0FDB	4E0B2408	885A288F
31	5E7FB481	F3747E91	B4E0C83C	F8FF092F
32	E868FA6C	54C6D456	D7C5C726	F2F02042
33	C8485ACE	20F0F222	9597D03A	D05851FD
34	E262E034	2CFC1EFE	D4D6C41C	EAE87A4A
35	92424096	58D838BA	C694D379	DA405BF7
36	EE7EEC7A	50D28052	C1C3917B	F4F5668B
37	DD0D1FCB	30A0A232	9381963E	861D890F
38	E666E460	3A825ADA	E2B0F369	7E2DD8FB
39	971D4F93	08889092	282A2C84	5E87F984
40	FB697BED	04D6D705	DFCDCFE1	F3F17143
41	C949DB4D	73A1ACAE	CCD2B99D	8159D3EE
42	F36137F1	BC0AAB1D	AA9FB28D	EFE7E58D
43	0173B771	8F2F842E	BBE58521	EC7CDFDB
44	5B5C0459	188A8909	C9CB9901	DDE62754
45	CC5F5D16	29B9F63D	CA86BF9B	850D4625
46	47945291	A406F721	CE9C87A9	3972776E
47	35FEF813	853F8677	6D03ABC0	78E2416C
48	F272787C	07879515	E7B5B76F	E06A6444
49	907058FC	23BBA32B	A5A7A00A	C248D9EB
50	147946B5	A5AF1A98	B6A29E2E	685FCB2B
51	DAD9C267	009A25BD	A4A6E38C	CAC95D29
52	E765E511	31B19303	B18EB389	E44FC870
53	C0F0437D	0B19A7B5	A1A3EC04	7BEDE975
54	F764C344	3C3E14BF	C2EE8B98	4D7F7DC0
55	C6FFF9C4	B33927B7	9A880020	8C67DED7
56	F5775654	1F117624	372527AD	E36365D5
57	9545FD75	26948B35	8A6B223D	D1C76FE1
58	C1D8D376	991B9B97	02061E90	C379625C
59	C77441DE	9D34963B	BC0C0E23	4ED6606D
60	D25351D0	3337B0BE	BE928FAC	CFDC6B4B
61	C5D7D5F6	02101716	80AF3F83	69CDD4CE
62	50D6D157	12A69FB4	39AE823B	C649614C
63	D455DCF4	B61336B2	2D292F2B	C5CCC1C4

Таблица 5.2 – **Начальные значения коэффициентов v_i в шестнадцатеричной системе счисления**

Значение индекса i	v_i
0	D1845AC6
1	AC3D25C6
2	F467247D
3	079294AB
4	F19A24CD
5	B47D25C6
6	D4522491
7	0D817489
8	87D45A6F
9	3D5721C6
10	573714C8
11	078274DB
12	2A8A1A76
13	DC6715C6
14	B4F1257D
15	0B1294AC

$K - K \in Z(256),\ K = \sum_{i=0}^{7} k_i \cdot (2^{32})^i = \sum_{i=0}^{1} K_i \cdot (2^{128})^i$;

$W - W \in Z(256), W = \sum_{i=0}^{1} W_i \cdot (2^{128})^i$;

d – целое число, $0 < d \le n + 2$.

5.5 Алгоритм вычисления значения функции хэширования

Алгоритм вычисления значения функции хэшированиявключает в себя следующие шаги:

1 $d := 1$;

2 $K := M_d$;

3 $V := \sum_{i=0}^{7} (v_i \oplus' k_i) \cdot (2^{32})^i + \sum_{i=8}^{15} (v_i \oplus' h_{i-8}) \cdot (2^{32})^i$;

4 Для $i = 0,2,4,6$ выполнить следующую последовательность вычислений:

$V := \rho_0^{29}(V),\ r_{i0} := V$;

$V := \rho_1^{18}(V),\ r_{i1} := V$;

$V := \rho_2^{19}(V),\ r_{i2} := V$;

$V := \rho_3^{17}(V),\ r_{i3} := V$;

5 $W := K \oplus H$;
6 $W_0 := \omega(W_0, T_0, T_1, T_2, T_3, T_4, T_5, T_6, T_7)$;
7 $W_1 := \omega(W_1, T_4, T_1, T_0, T_3, T_6, T_5, T_2, T_7)$;
8 $W := \xi^{31}(W)$;
9 $W := \varphi(H_0, \varphi(H_0, W))$;
10 $W := \varphi(K_0, \varphi(K_0, W))$;
11 $W := \varphi(H_1, \varphi(H_1, W))$;
12 $W := \varphi(K_1, \varphi(K_1, W))$;
13 $H := W$;
14 $d := d + 1$;
15 Проверить условие $d=n+2$.
Если условие выполнено, то перейти к шагу 16. Если условие не выполнено – к шагу 2;
16 $h(M) := H \bmod 2^L$;
17 Конец работы алгоритма.