**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра информационной безопасности**

отчет

**по лабораторной работе №8**

**по дисциплине «Криптография и защита информации»**

Тема: Изучение цифровой подписи

|  |  |  |
| --- | --- | --- |
| Студент гр. 9381 |  | Колованов Р.А. |
| Преподаватель |  | Племянников А.К. |

Санкт-Петербург

2022

**Цель работы.**

Исследовать алгоритмы создания и проверки цифровой подписи, алгоритмы генерации ключевых пар для алгоритмов цифрой подписи RSA, DSA, ECDSA и получить практические навыки работы с ними, в том числе с использованием приложения Cryptool 1 и 2.

**Основные теоретические положения.**

***Генератор ключевых пар.***

*Генерация ключевых пар для алгоритма RSA.*

1. Генерация двух больших простых чисел *p* и *q*;
2. Вычисление *n* = *p* \* *q*;
3. Выбор произвольного *e* (*e* < *n*), взаимно простого с 𝜑(𝑛);
4. Вычисление 𝑑: 𝑒 ∗ 𝑑 = 1 𝑚𝑜𝑑 𝜑(𝑛);
5. Числа (*e*, *n*) – открытый ключ, *d* – закрытый ключ, *p* и *q* уничтожаются.

*Генерация ключевых пар для алгоритма DSA.*

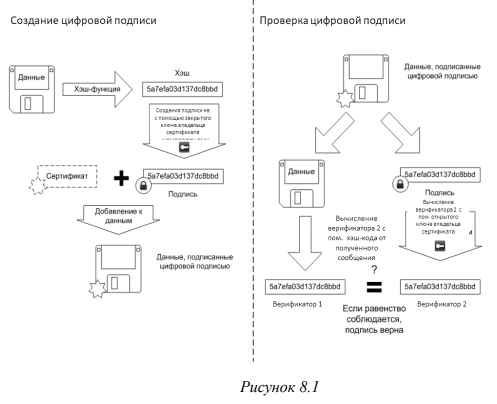
1. Выбирается число *p*: длина – [512, 1024] битов и число битов в *p* должно быть кратно 64;
2. Выбирается число *q*, которое имеет тот же самый размер дайджеста 160 битов, такое, что: (𝑝 − 1) = 0 𝑚𝑜𝑑 𝑞;
3. Выбирается 𝑒1: 𝑒1𝑞 = 1 𝑚𝑜𝑑 𝑝;
4. Выбирается целое число 𝑑 < 𝑞 и вычисляется 𝑒2 = 𝑒1𝑑 𝑚𝑜𝑑 𝑝;
5. Числа (𝑒1, 𝑒2, 𝑝, 𝑞) – открытый ключ, d – закрытый ключ.

*Генерация ключевых пар для алгоритма ECDSA.*

1. Выбирается эллиптическая кривая 𝐸𝑝(𝑎, 𝑏), 𝑝 − простое число;
2. Выбирается точка на кривой 𝑒1 = (𝑥1, 𝑦1);
3. Выбирается простое число *q* – порядок одной из циклических подгрупп группы точек эллиптической кривой: 𝑞 × (𝑥1, 𝑦1) = 𝑂;
4. Выбирается закрытый ключ *d*;
5. Вычисляется точка на кривой 𝑒2 = 𝑑 × 𝑒1;
6. Открытый ключ - (𝑎, 𝑏, 𝑞, 𝑝, 𝑒1, 𝑒2).

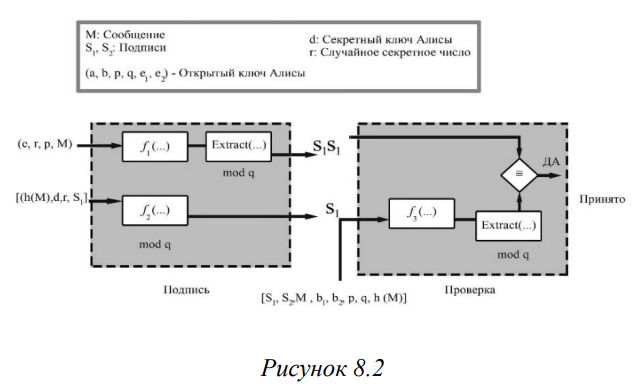
***Процессы создания и проверки цифровой подписи.***

Обобщенные схемы подписания и проверки цифровой подписи представлены на рисунке 8.1.



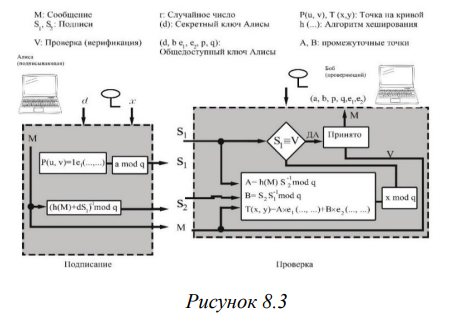
***Схемы цифровой подписи на эллиптических кривых.***

Схема цифровой подписи ECDSA (рисунок 8.2).



В процессе подписания две функции 𝑓1 и 𝑓2 и экстрактор *Extract* создают две части подписи. В процессе проверки (верификации) обрабатывают выход одной функции 𝑓2 (после прохождения через экстрактор) и сравнивают ее с первой частью подписи.

После того, как сгенерирована ключевая пара (закрытый ключ – d, и открытый ключ – (𝑎, 𝑏, 𝑞, 𝑝, 𝑒1, 𝑒2), осуществляется подписание документа, затем на принимающей стороне осуществляется проверка (рисунок 8.3).



Алгоритм подписания ECDSA состоит из следующих операций:

1. Выбирается секретное случайное число 𝑟: 𝑟 ∈ (1, 𝑞 − 1);
2. Выбирается третья точка на кривой: 𝑃(𝑢, 𝑣) = 𝑟 × 𝑒1;
3. Вычисляется первая часть подписи по формуле:



где *u* – абсцисса;

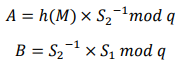
1. Вычисляется вторая часть подписи по формуле:



где ℎ(𝑀) – дайджест сообщения, d – закрытый ключ.

Алгоритм проверки цифровой подписи ECDSA включает следующие операции:

1. Вычисляем промежуточные результаты A и B:



1. Восстанавливаем третью точку:



1. Верификатор 𝑉 = 𝑥 𝑚𝑜𝑑 𝑞 сравнивается с первой частью цифровой подписи 𝑆1.

***Демонстрация процесса подписи в среде PKI.***

Инфраструктура открытых ключей (ИОК, PKI – Public Key Infrastructure) – набор средств (технических, материальных, организационных и т. д.), распределённых служб и компонентов, в совокупности используемых для поддержки решения основных задач криптографии, а именно:

1. Обеспечение конфиденциальности информации;
2. Обеспечение целостности информации;
3. Обеспечение аутентификации пользователей и ресурсов, к которым обращаются пользователи;
4. Обеспечение возможности подтверждения совершенных пользователями действий.

Решение перечисленных задач основано на использовании сертификатов открытых ключей. Сертификат открытого ключа – это электронный документ, который содержит:

1. Открытый ключ пользователя;
2. Информацию о пользователе, которому принадлежит сертификат;
3. Информацию о сроке действия сертификата;
4. Информацию об издателе сертификата;
5. Другие атрибуты;
6. Цифровую подпись этих данных, созданную удостоверяющим центром, издавшим и выдавшим этот сертификат.

Существует несколько вариантов использования сертификатов открытых ключей:

1. Для зашифрования и расшифрования электронных документов;
2. Для подписания электронного документа и проверки подписи;
3. Для аутентификации отправителя документа.

**Ход работы.**

***Генераторов ключевых пар.***

*Задание.*

1. Перейти к утилите «Digital Signatures / PKI -> PKI -> Generate/Import Keys»;
2. Сгенерировать ключевые пары по алгоритмам RSA-2048, DSA-2048, EC-239. Зафиксируйте время генерации в таблице;
3. С помощью утилиты «Digital Signatures / PKI -> PKI -> Display/Export Keys» вывести сгенерированный открытый ключ и сохранить соответствующий скриншот.

*Описание алгоритмов генерации.*

Генерация ключей RSA-2048:

1. Генерация двух больших простых чисел *p* и *q* (*p* и *q* держаться в секрете);
2. Вычисление *n* = *p* *\** *q*;
3. Выбор произвольного *e* (*e < n*), взаимно простого с 𝜑(𝑛) = (*p* – 1)(*q* – 1) – функцией Эйлера;
4. Вычисление закрытого ключа 𝑑: 𝑒 ∗ 𝑑 = 1 𝑚𝑜𝑑 𝜑(𝑛);
5. Числа (*e*, *n*) – открытый ключ, *d* – закрытый ключ, *p* и *q* уничтожаются.

Генерация ключей DSA-2048:

1. Выбирается простое число 𝑝, длиной между 512 и 1024 битами. Число битов в *p* должно быть кратно 64;
2. Выбирается другое простое число 𝑞, которое имеет тот же самый размер, что и дайджест – 160 битов, такое , что (𝑝 − 1) = 0 𝑚𝑜𝑑 𝑞;
3. Выбирается 𝑒1, такое, что 𝑒1𝑞 = 1 𝑚𝑜𝑑 𝑝 путем вычисления 𝑒1 = 𝑒0(𝑝−1)/𝑞 𝑚𝑜𝑑 𝑝, где 𝑒0 ∈ 𝑍𝑝 (теорема Ферма);
4. Выбирается целое 𝑑 < 𝑞 и вычисляется 𝑒2 = 𝑒1𝑑 𝑚𝑜𝑑 𝑝;
5. Числа (𝑒1, 𝑒2, 𝑝, 𝑞) – открытый ключ, d – закрытый ключ.

Генерация ключей ECDSA (EC-239):

1. Выбирается эллиптическая кривая 𝐸𝑝(𝑎, 𝑏), 𝑝 − простое число;
2. Выбирается точка на кривой 𝑒1 = (𝑥1, 𝑦1);
3. Выбирается простое число *q* – порядок одной из циклических подгрупп группы точек эллиптической кривой: 𝑞 × (𝑥1, 𝑦1) = 𝑂;
4. Выбирается целое число *d* (1 < *d* < q – 1), и назначается закрытым ключом;
5. Вычисляется точка на кривой 𝑒2 = 𝑑 × 𝑒1;
6. Открытый ключ - (𝑎, 𝑏, 𝑞, 𝑝, 𝑒1, 𝑒2).

*Генерация ключевых пар.*

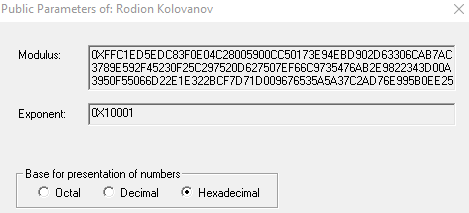
Для начала при помощи утилиты «Digital Signatures / PKI -> PKI -> Generate/Import Keys» были сгенерированы ключевые пары по алгоритмам RSA-2048, DSA-2048 и EC-239, при этом так же было зафиксировано время генерации ключевых пар. Результаты представлены в таблице 1.

*Таблица 1 – Время генерации ключевых пар.*

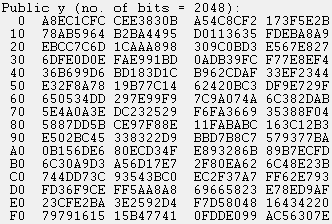
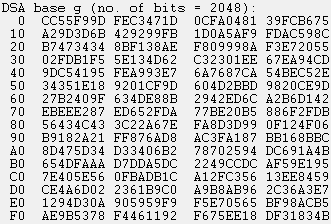
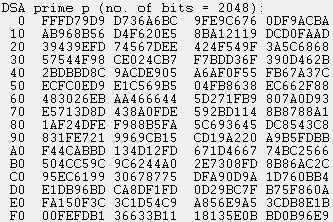
|  |  |
| --- | --- |
| Алгоритм | Время генерации пары ключей |
| RSA-2048 | 0.894 секунды |
| DSA-2048 | 2.899 секунды |
| EC-239 (prime239v1) | 0.012 секунды |

Как видно из результатов, алгоритм EC-239 показал наименьшее время генерации пары ключей, а DSA-2048 – наибольшее.

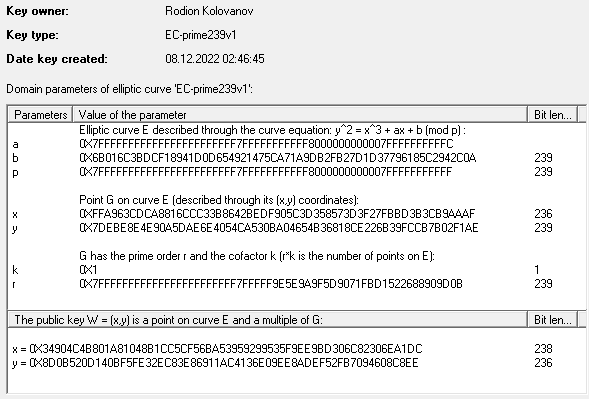
Сгенерированные открытые ключи представлены на рисунках 1, 2 и 3.



*Рисунок 1 – Открытый ключ RSA-2048.*



*Рисунок 2 – Открытый ключ DSA-2048.*



*Рисунок 3 – Открытый ключ EC-239.*

***Процессы создания и проверки цифровой подписи.***

*Задание.*

1. Открыть текст не менее 5000 знаков. Перейти к приложению «Digital Signatures/PKI -> Sign Document»;
2. Задайте хэш-функцию, и другие параметры цифровой подписи;
3. Создайте подпись ключами, сгенерированными в предыдущем задании. Зафиксируйте время создания цифровой подписи для каждого ключа;
4. Сохраните скриншот цифровой подписи с помощью приложения «Digital Signatures/PKI -> Extract Signature»;
5. Выполните процедуру проверки подписи «Digital Signatures/PKI -> Verify Signature» для случаев сохранения и нарушения целостности исходного текста. Сохраните скриншоты результатов.

*Обобщенная схема создания и проверки цифровой подписи.*

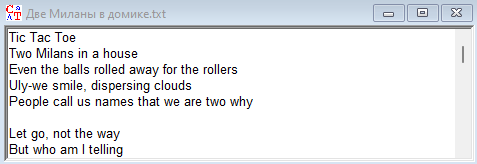
Обобщенная схема создания и проверки цифровой подписи представлена на рисунке 4.



*Рисунок 4 – Схема создания и проверки цифровой подписи.*

*Создание и проверка цифровой подписи.*

В качестве исходного текста был взят текст (размер текста превышает 5000 символов и составляет около 6300 символов), представленный на рисунке 5.



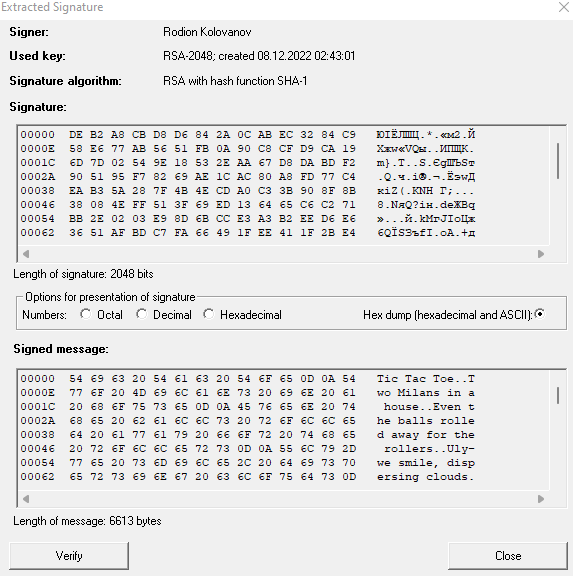
*Рисунок 5 – Исходный текст.*

Далее при помощи ранее сгенерированных ключевых пар были созданы цифровые подписи, при этом так же было зафиксировано время генерации цифровых подписей. Результаты представлены в таблице 2.

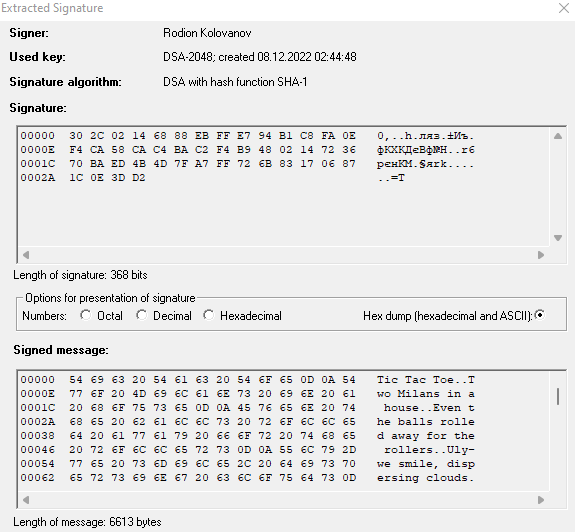
*Таблица 2 – Время генерации цифровых подписей.*

|  |  |  |
| --- | --- | --- |
| Алгоритм | Хэш-функция | Время генерации цифровой подписи |
| RSA-2048 | SHA-1 | 0.010 секунд |
| DSA-2048 | SHA-1 | 0.002 секунды |
| ECSP-DSA | SHA-1 | 0.002 секунды |
| ECSP-NR | SHA-1 | 0.002 секунды |

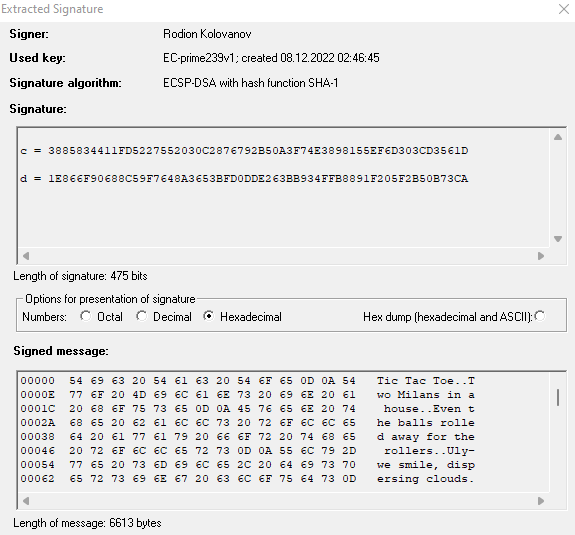
Сгенерированные цифровые подписи представлены на рисунках 6, 7, 8 и 9.



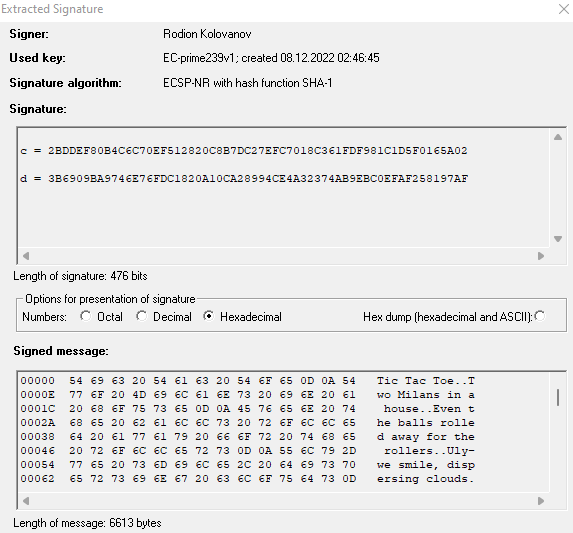
*Рисунок 6 – Цифровая подпись, сгенерированная алгоритмом RSA с использованием хэш-функции SHA-1 и ключа RSA-2048.*



*Рисунок 7 – Цифровая подпись, сгенерированная алгоритмом DSA с использованием хэш-функции SHA-1 и ключа DSA-2048.*



*Рисунок 8 – Цифровая подпись, сгенерированная алгоритмом ECSP-DSA с использованием хэш-функции SHA-1 и ключа EC-prime239v1.*



*Рисунок 9 – Цифровая подпись, сгенерированная алгоритмом ECSP-NR с использованием хэш-функции SHA-1 и ключа EC-prime239v1.*

Далее была выполнена процедура проверки цифровой подписи «Digital Signatures/PKI -> Verify Signature» для случаев сохранения и нарушения целостности. Результаты представлены на рисунках 10-17.

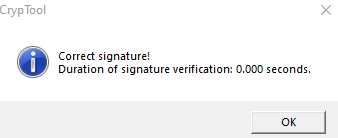


Рисунок 10 – Проверка цифровой подписи для алгоритма RSA при сохранении целостности.

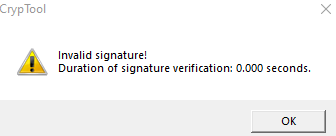


Рисунок 11 – Проверка цифровой подписи для алгоритма RSA при нарушении целостности.

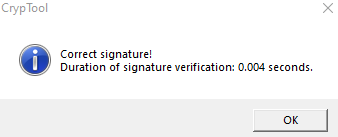


Рисунок 12 – Проверка цифровой подписи для алгоритма DSA при сохранении целостности.

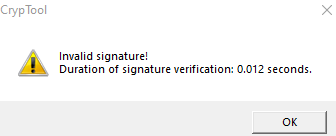


Рисунок 13 – Проверка цифровой подписи для алгоритма DSA при нарушении целостности.

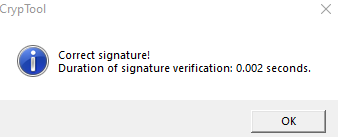


Рисунок 14 – Проверка цифровой подписи для алгоритма ECSP-DSA при сохранении целостности.

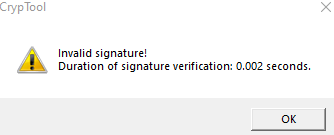


Рисунок 15 – Проверка цифровой подписи для алгоритма ECSP-DSA при нарушении целостности.

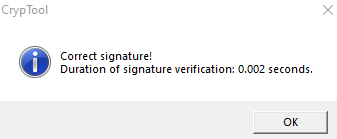


Рисунок 16 – Проверка цифровой подписи для алгоритма ECSP-NR при сохранении целостности.

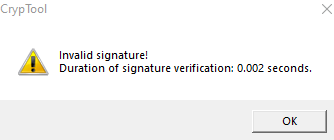


Рисунок 17 – Проверка цифровой подписи для алгоритма ECSP-NR при нарушении целостности.

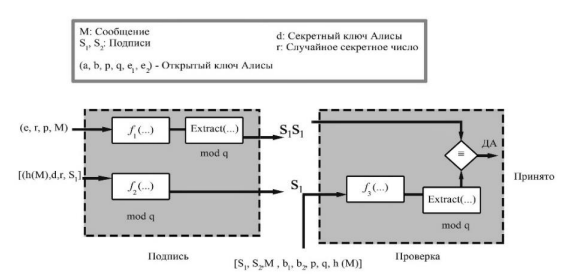
***Схемы цифровой подписи на эллиптических кривых.***

*Задание.*

1. Выполните процедуру создание подписи «Digital Signatures / PKI -> Sign Document» алгоритмом ECSP-DSA в пошаговом режиме (Display inter. results=ON). Зафиксируйте скриншоты последовательности шагов;
2. Выполните процедуру проверки подписи ECSP-DSA для случаев сохранения и нарушения целостности исходного текста. Сохраните скриншоты результатов;
3. Проверить лекционный материал по ECDSA, выполнив создание и проверку подписи сообщения M (принять M=h(M)) приложением «Indiv.Procedures -> Number Theory -> Point Addition on EC».

*Описание алгоритма формирования и проверки подписи ECDSA.*

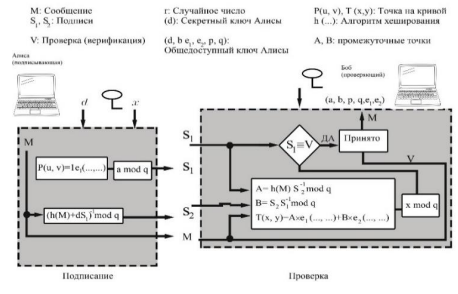
Схема цифровой подписи ECDSA представлена на рисунке 18.

****

*Рисунок 18 – Схема цифровой подписи ECDSA.*

В процессе подписания две функции 𝑓1 и 𝑓2 и экстрактор *Extract* создают две части подписи. В процессе проверки (верификации) обрабатывают выход одной функции 𝑓2 (после прохождения через экстрактор) и сравнивают ее с первой частью подписи.

После того, как сгенерирована ключевая пара (закрытый ключ – d, и открытый ключ – (𝑎, 𝑏, 𝑞, 𝑝, 𝑒1, 𝑒2), осуществляется подписание документа, затем на принимающей стороне осуществляется проверка. Схема представлена на рисунке 19.



*Рисунок 19 – Проверка цифровой подписи ECDSA.*

Алгоритм подписания ECDSA состоит из следующих операций:

1. Выбирается секретное случайное число 𝑟: 𝑟 ∈ (1, 𝑞 − 1);
2. Выбирается третья точка на кривой: 𝑃(𝑢, 𝑣) = 𝑟 × 𝑒1;
3. Вычисляется первая часть подписи по формуле:



где *u* – абсцисса;

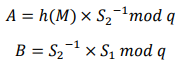
1. Вычисляется вторая часть подписи по формуле:



где ℎ(𝑀) – дайджест сообщения, d – закрытый ключ.

Алгоритм проверки цифровой подписи ECDSA включает следующие операции:

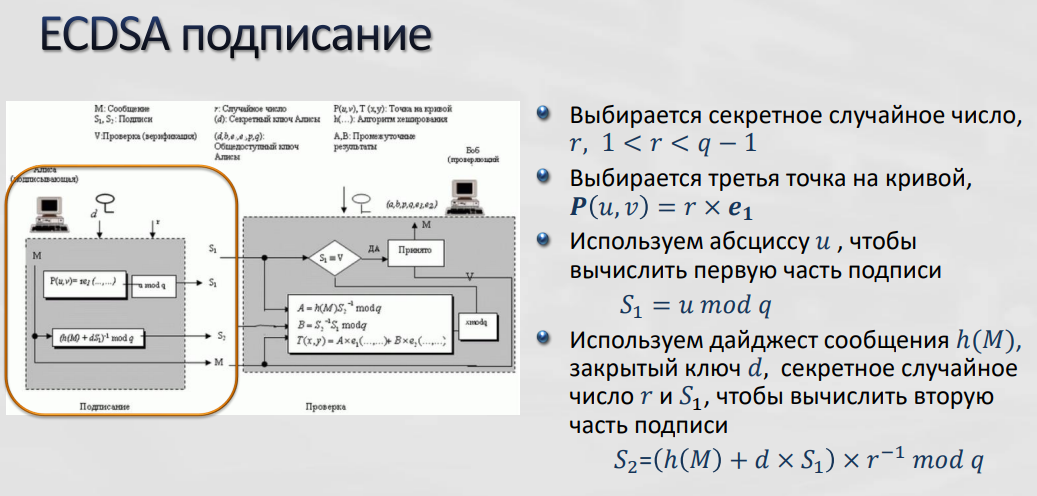
1. Вычисляем промежуточные результаты A и B:

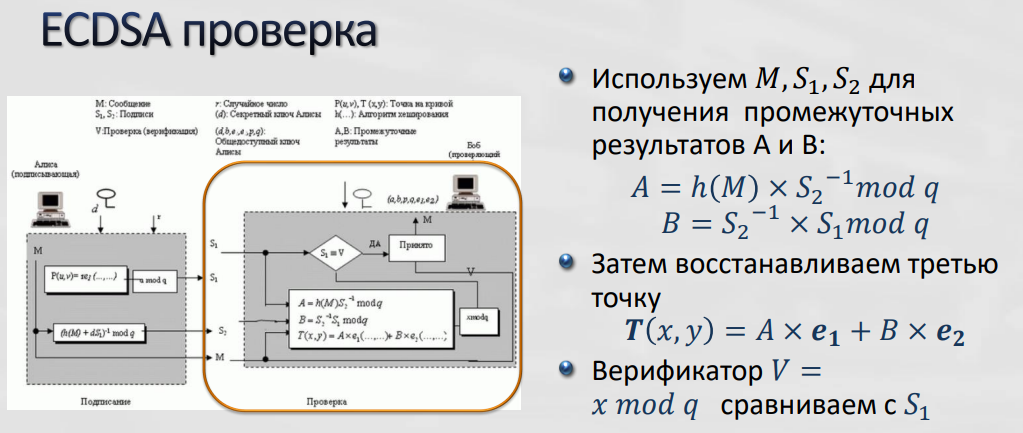


1. Восстанавливаем третью точку:



1. Верификатор 𝑉 = 𝑥 𝑚𝑜𝑑 𝑞 сравнивается с первой частью цифровой подписи 𝑆1.





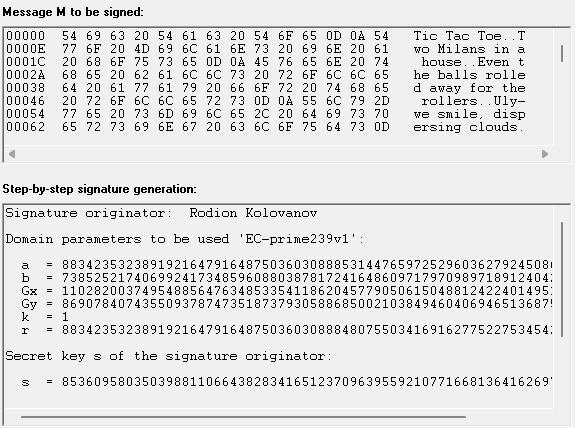
Сравним лекционную версию ECDSA и реализацию из CrypTool 1. Результаты представлены в таблице 3.

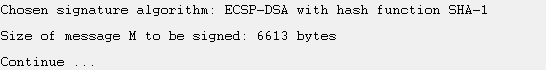
*Таблица 3 – Сравнение лекционной версии ECDSA и реализации из CrypTool.*

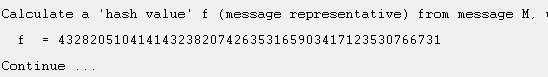
|  |  |
| --- | --- |
| Параметр из CrypTool 1 | Параметр из лекции |
| *a* | *a* |
| *b* | *b* |
| *(Gx, Gy)* | *e1 = (x, y)* |
| *r* | *q* |
| *s* | *d* |
| *f* | *h(M)* |
| *u* | *r* |
| *i* | *u* |
| *(Vx, Vy)* | *P(u, v)* |
| *c* | *S1* |
| *d* | *S2* |

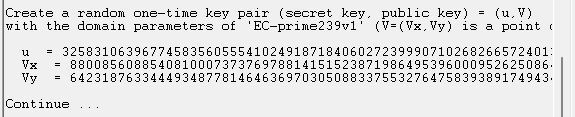
*Выполнение алгоритма формирования и проверки подписи ECDSA.*

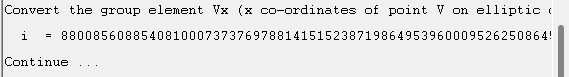
Далее при помощи утилиты «Digital Signatures / PKI -> Sign Document» была создана подпись алгоритмом ECSP-DSA в пошаговом режиме. Последовательность шагов представлена на следующих рисунках:

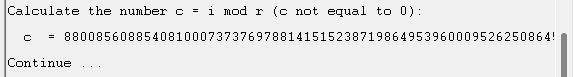
******

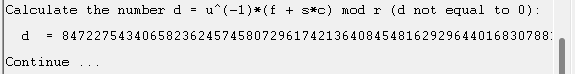
***\***

******

******

******

******

******

******

Далее при помощи утилиты «Digital Signatures/PKI -> Verify Signature» была выполнена проверка подписи для случаев сохранения и нарушения целостности. Результаты представлены на рисунках 20 и 21.

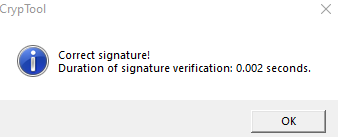


Рисунок 20 – Проверка цифровой подписи для алгоритма ECSP-DSA при сохранении целостности.

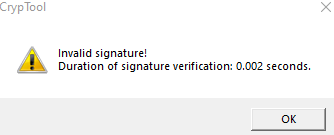


Рисунок 21 – Проверка цифровой подписи для алгоритма ECSP-DSA при нарушении целостности.

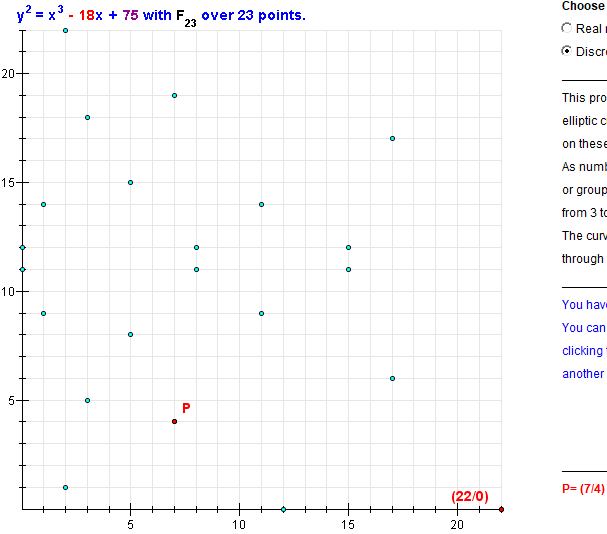
Далее были выполнены создание и проверка подписи сообщения *M = h(M)* утилитой «Indiv.Procedures -> Number Theory -> Point Addition on EC».

1) Генерация ключей.

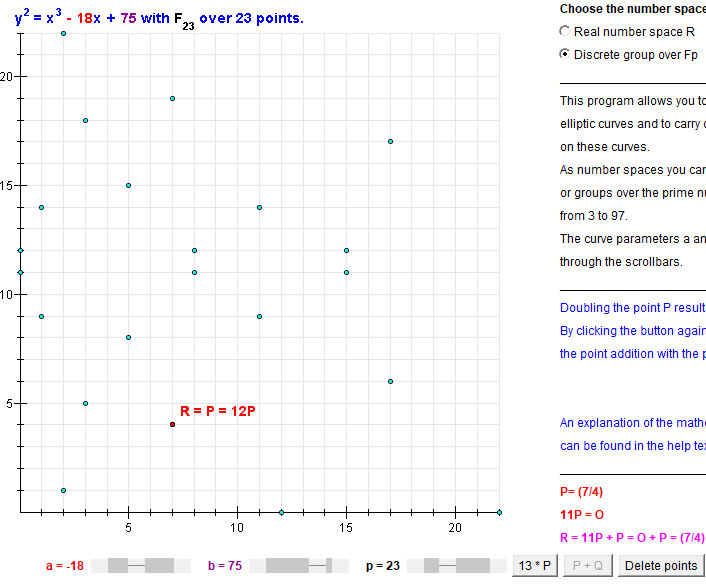
Были взяты следующие параметры эллиптической кривой:

*a* = -18, *b* = 75, *p* = 23.

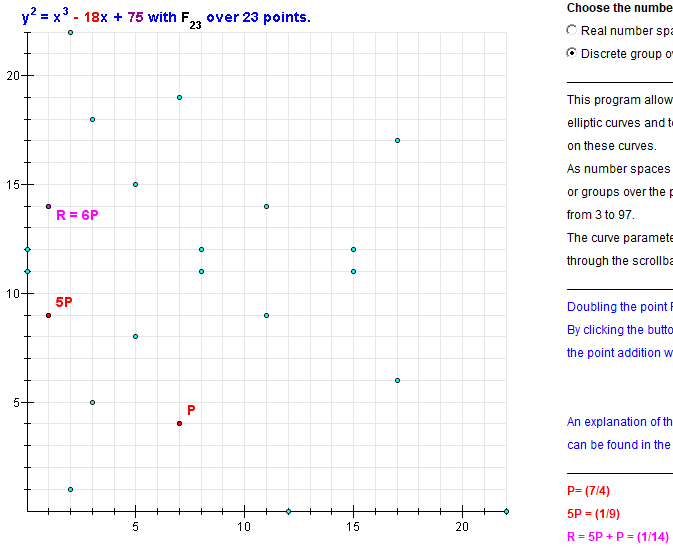
Далее была выбрана точка на кривой *P* = e1 = (7, 4).



Далее было подобрано такое число *q*, что 𝑞 × (𝑥1, 𝑦1) = 𝑂: *q* = 11.



Далее было выбрано целое число d (1 < d < q – 1 = 10): d = 5 – закрытый ключ, после чего была вычислена точка e2 = *d* × e1 = (1, 9).



Открытый ключ получен:

(a = -18, b = 75, p = 23, q = 11, e1 = (7, 4), e2 = (1, 9)).

2) Подписание.

Выбираем секретное случайное число *r* = 3. Далее выбираем третью точку на кривой *P* = *r* × e1 = (15, 11). Далее, используя абсциссу *u = 15*, вычисляем первую часть подписи S1 = *u* mod *q* = 15 mod 11 = 4.

Пусть исходное сообщение *M* *=* *h(M)* = 66. Тогда теперь можно вычислить вторую часть подписи 𝑆2 = (ℎ(𝑀) + 𝑑 × S1) × 𝑟−1 𝑚𝑜𝑑 𝑞 = (66 + 5 × 4) × 4 mod 11 = 344 mod 11 = 3.

Итого получаем (*M* = 66, *h(M)* = 66, S1 = 4, S2 = 3).

3) Проверка подписи.

Вычислим промежуточные результаты *A* и *B*:

*A = h(M)* × 𝑆2-1 mod q = 66 × 4 mod 11 = 0.

*B =* 𝑆2-1 × 𝑆1 mod q = 4 × 4 mod 11 = 5.

Далее восстанавливаем третью точку *T*:

*T* = (*x*, *y*) = A × e1 + *B* × e2 = 0 × (7, 4) + 5 × (1, 9) = (15, 11).

Далее находим верификатор *V*:

*V = x* mod *q* = 15 mod 11 = 4.

Сравниваем верификатор *V* = 4c S1 = 4 – они равны, значит подпись корректна.

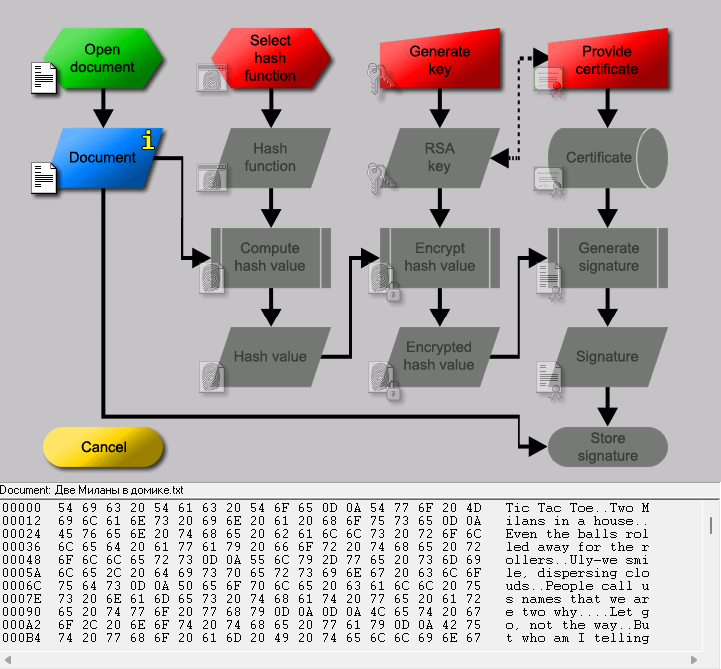
***Демонстрация процесса подписи в среде PKI.***

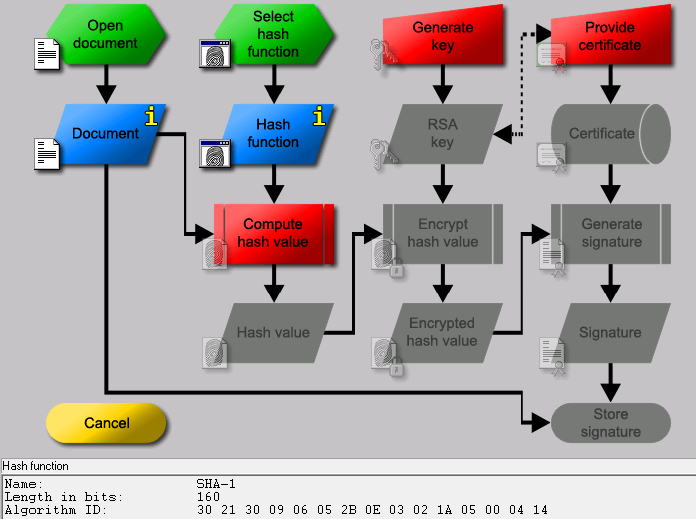
*Задание.*

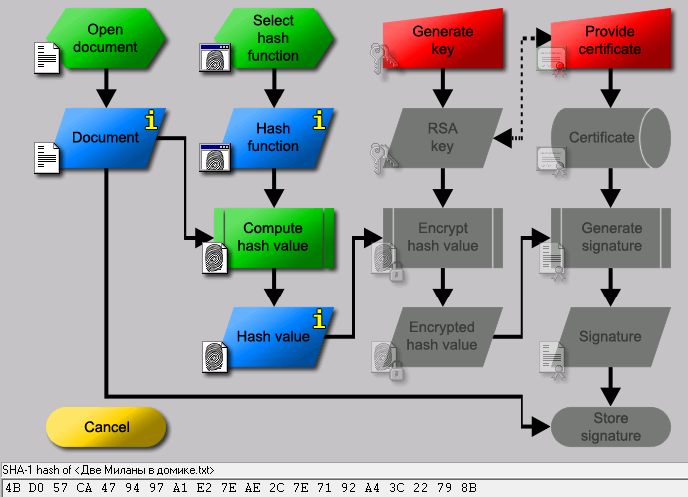
1. Запустить демонстрационную утилиту «Digital Signatures / PKI -> Signature Demonstration»;
2. Получите сертификат на ранее сгенерированную ключевую пару RSA-2048;
3. Выполните и сохраните скриншоты всех этапов создания цифровой подписи документа;
4. Сохраните скриншот сертификата для проверки этой цифровой подписи.

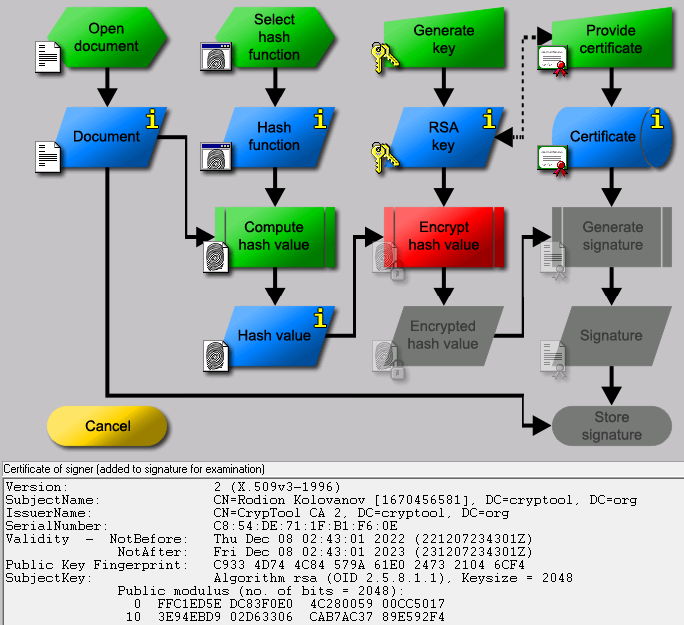
*Схема процедуры подписания из CrypTool.*

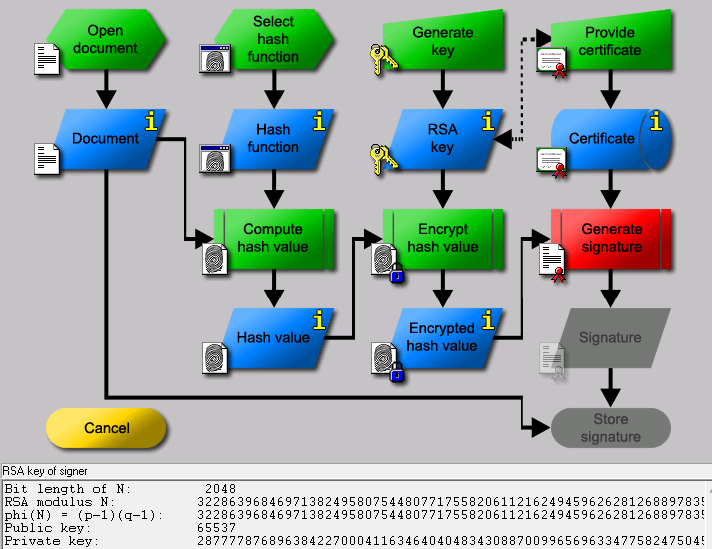
Была запущена демонстрационная утилита «Digital Signatures / PKI -> Signature Demonstration». При помощи утилиты получим сертификат для ранее сгенерированной ключевой пары RSA-2048.

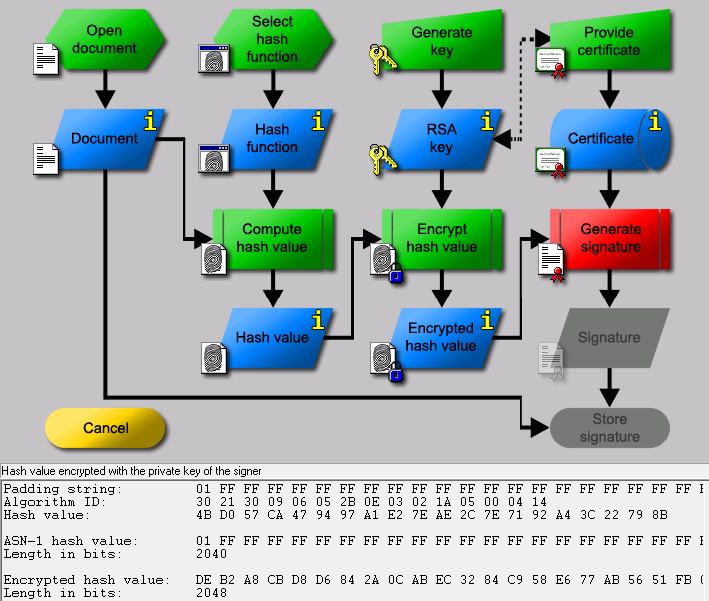
****

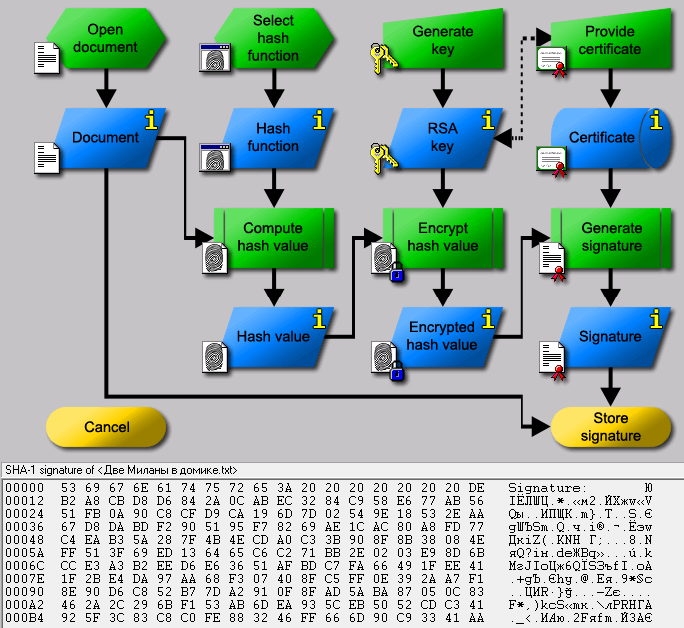


**

**

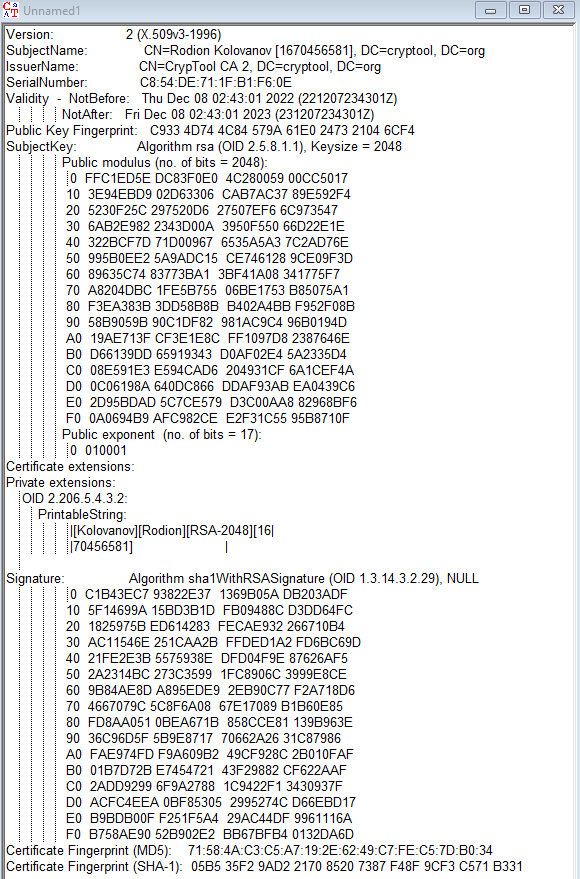
**

**

**

*Сравнение структуры сертификата из лекции и сертификата из CrypTool.*

Полученный на предыдущем шаге сертификат, а также его структура представлены на рисунке 22.

****

*Рисунок 22 – Сертификат из CrypTool 1.*

Структура сертификата из лекций представлена на рисунке 23.



*Рисунок 23 – Структура сертификата из лекций.*

Результаты сравнения двух сертификатов:

* Версия присутствует в обоих сертификатах;
* Серийный номер присутствует в обоих сертификатах;
* Идентификатор алгоритма подписи присутствует в обоих сертификатах;
* Имя издателя и субъекта присутствуют в обоих сертификатах;
* Период действия присутствует в обоих сертификатах;
* Информация об открытом ключе присутствуют в обоих сертификатах;
* Поле с хэшом открытого ключа есть только в версии сертификата CrypTool;
* Уникальные идентификаторы издателя и субъекта присутствуют только в версии сертификата из лекций;
* Дополнения присутствуют в обоих сертификатах;
* Подпись присутствует в обоих сертификатах;
* Поля с хэшами SHA-1 и MD5 сертификата есть только в версии сертификата CrypTool;

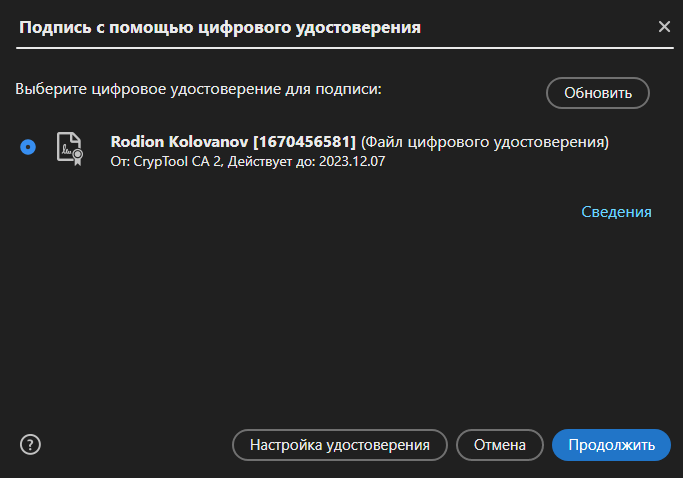
***Подписание своего отчета.***

*Задание.*

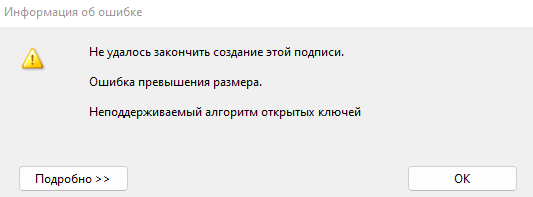
1. Сконвертируйте отчет в формат pdf;
2. Экспортируйте ранее созданный сертификат ключевой пары RSA «Digital Signatures / PKI -> PKI -> Display / Export Keys -> Export PSE(#PKCS12)»;
3. Откройте pdf-версию отчета и попытайтесь подписать с использованием этого сертификата;
4. Создайте собственный самоподписанный сертификат в среде Adobe Reader и используйте его для подписи отчета;
5. Сохраните скриншоты свойств подписи и сертификата;
6. Внесите изменения (маркеры, комментарии) в отчет и проверьте подпись.

*Подписание своего отчета.*

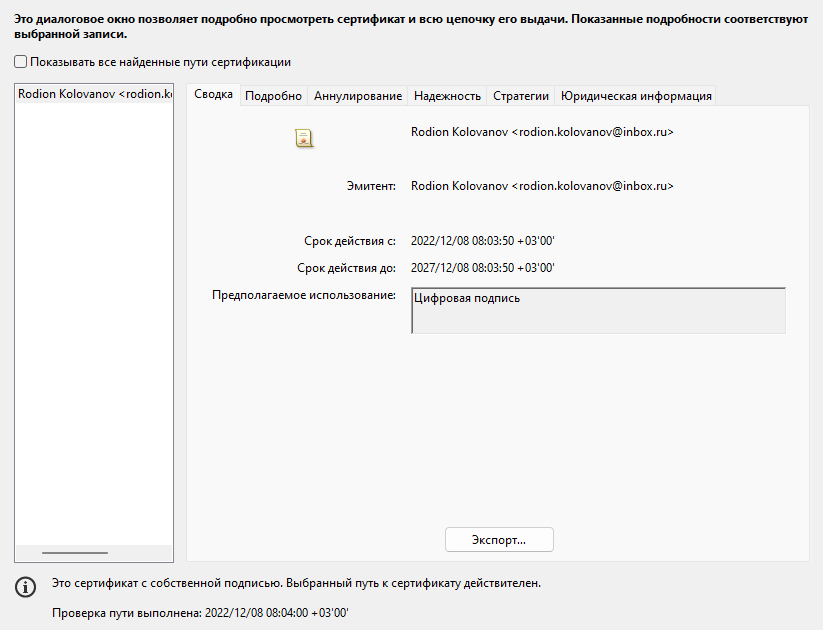
Для начала текущее состояние отчета было сконвертировано в формат pdf. Далее при помощи утилиты «Digital Signatures / PKI -> PKI -> Display / Export Keys -> Export PSE(#PKCS12)» был экспортирован ранее созданный сертификат ключевой пары RSA. Далее через программу Adobe Reader была осуществлена попытка подписать сохраненный pdf.



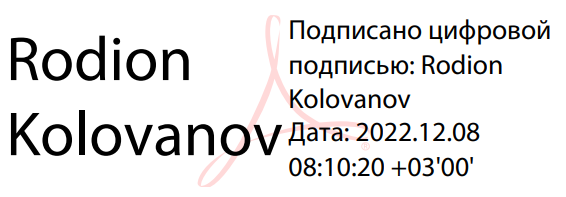
В результате, подписать сохраненный pdf не удалось.



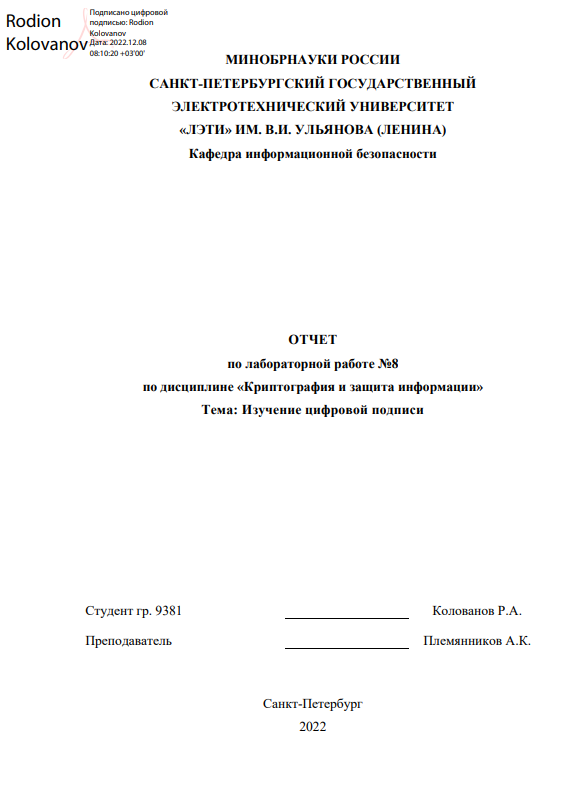
Далее в среде Adobe Reader был создан собственный самоподписанный сертификат, который в дальнейшем был использован для повторного подписания сохраненного pdf. Свойства сертификата представлены на рисунке 24.

*Рисунок 24 – Свойства созданного сертификата.*

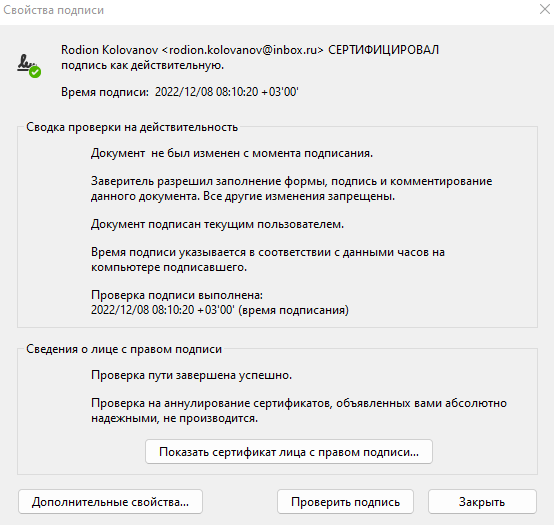
Далее pdf-файл был подписан при помощи созданного сертификата. Подпись и ее свойства представлены на рисунках 25, 26 и 27.



*Рисунок 25 – Подпись pdf-файла.*

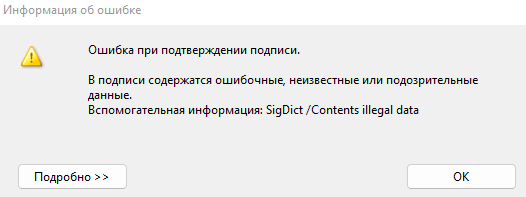


*Рисунок 26 – Титульный лист pdf-файла с подписью.*



*Рисунок 26 – Свойства подписи pdf-файла.*

Далее в pdf-файл были внесены изменения (добавлены маркеры и комментарии), после чего подпись файла была проверена. Результат представлен на рисунке 28.



*Рисунок 28 – Подтверждение подписи измененного pdf-файла.*

**Выводы.**

В ходе выполнения данной лабораторной работы были исследованы алгоритмы создания и проверки цифровой подписи, а также алгоритмы генерации ключевых пар для алгоритмов цифрой подписи RSA, DSA, ECDSA.

1. Алгоритмы генерации ключевых пар:
   1. Была рассмотрена работа алгоритма RSA для генерации ключевых пар. Было определено, что в качестве открытого ключа алгоритм генерирует пару значений (*e*, *n*), а в качестве закрытого – значение *d*. Для генерации пары ключей алгоритм использует два больших простых числа *p* и *q* (*n* = *p* \* *q*), которые после окончанию генерации уничтожаются. В качестве e выбирается любое число, взаимно простое с 𝜑(𝑛). А значение d вычисляется из уравнения 𝑒 ∗ 𝑑 = 1 𝑚𝑜𝑑 𝜑(𝑛);
   2. Была рассмотрена работа алгоритма DSA для генерации ключевых пар. Было определено, что в качестве открытого ключа алгоритм генерирует набор значений (*e1*, *e2*, *p, q*), а в качестве закрытого – значение *d*. Для генерации пары ключей алгоритм использует число *p*, размер которого составляет 512-1024 бита и кратен 64, и число *q*, размер которого совпадает с размером хэша и которое удовлетворяет равенству (𝑝 − 1) = 0 𝑚𝑜𝑑 *q*. В качестве e1 выбирается такое число, что e1q = 1 mod *p*. В качестве *d* выбирается любое число, меньшее q, а значение e2 вычисляется как e1q mod *p*;
   3. Была рассмотрена работа алгоритма ECDSA для генерации ключевых пар. Было определено, что в качестве открытого ключа алгоритм генерирует набор значений (*a*, *b*, *q*, *p*, *e1*, *e2*), а в качестве закрытого – значение *d*. Для генерации пары ключей алгоритм использует эллиптическую кривую, ее параметры *a*, *b* и *p* выбираются случайно, при этом *p* – простое число. В качестве *e1* выбирается любая точка на кривой. Значение *q* равно порядку циклической подгруппы группы точек эллиптической кривой (*q* × *e1* = *O*). В качестве *d* выбирается любое число, меньшее *q* – 1. Точка e2 вычисляется как *d* × *e1*.
   4. Было измерено время работы рассматриваемых алгоритмов генерации ключевых пар. Алгоритм EC-239 показал наименьшее время генерации (0.012 секунд), а DSA-2048 – наибольшее (2.899 секунд). Алгоритм RSA-2048 показал время 0.894 секунд.
2. Процесс создания и проверки цифровой подписи:
   1. Была рассмотрена обобщенная схема создания и проверки цифровой подписи. Было определено, что для создания подписи требуется вычислить дайджест данных и зашифровать его закрытым ключом владельца сертификата. После ее создания сертификат вместе с подписью добавляется к данным. Для проверки данных достаточно вычислить дайджест полученных данных и сравнить его верификатором, который вычисляется при помощи расшифровки подписи открытым ключом сертификата.
   2. Было исследовано время генерации цифровых подписей при помощи алгоритмов RSA, DSA, ECSP-DSA и ECSP-NR. Алгоритм RSA показал наибольшее время генерации (0.01 секунд), а остальные алгоритмы – наименьшее время (0.002 секунды).
   3. Было исследовано время проверки цифровых подписей при помощи алгоритмов RSA, DSA, ECSP-DSA и ECSP-NR. Алгоритм DSA показал наибольшее время проверки (0.004 секунды при сохранении целостности и 0.012 секунд при нарушении целостности), алгоритмы ECSP-DSA и ECSP-NR показали время 0.002 секунд, а алгоритм RSA – наименьшее время (0 секунд).
3. Создание и проверка цифровой подписи алгоритмом ECDSA, основанным на эллиптических кривых:
   1. Был рассмотрен процесс создания цифровой подписи. Было определено, что в качестве цифровой подписи алгоритм ECDSA генерирует набор значений (*M*, *S1*, *S2*) при помощи известного открытого ключа (*a*, *b*, *q*, *p*, *e1*, *e2*) и закрытого ключа *d*, где *M* – это подписанные данные. Для вычисления *S1* и *S2* выбирается секретное случайное число *r*, лежащее в диапазоне 1 < *r* < *q* – 1. Далее находится точка на кривой *P* = (*u*, *v*) = *r* × *e1*. Значение *S1* вычисляется как *u* mod *q*, а значение *S2* – как (ℎ(𝑀) + 𝑑 × S1) × 𝑟−1 𝑚𝑜𝑑 𝑞, где ℎ(𝑀) – дайджест *M*.
   2. Был рассмотрен процесс проверки цифровой подписи. Было определено, что для проверки цифровой подписи необходимо вычислить верификатор *V* и сравнить его со значением *S1* – если их значения совпадают, значит подпись прошла проверку. Значения (*M*, *S1*, *S2*) получаются из цифровой подписи. Для начала вычисляется точка *T* = (*x*, *y*) = (*h(M)* × 𝑆2-1 mod q) × e1 + (𝑆2-1 × 𝑆1 mod q) × e2. Далее значение верификатора *V* вычисляется как *x* mod *q.*
4. Процесс создания цифровой подписи в среде PKI:
   1. Был рассмотрен процесс создания цифровой подписи в среде PKI с использованием демонстрационной утилиты. Было определено, что схема создания цифровой подписи в среде PKI совпадает с обобщенной схемой создания цифровых подписей, рассматриваемой ранее. Цифровая подпись содержит зашифрованный дайджест вместе с информацией о алгоритме создания цифровой подписи и данными.
   2. Была рассмотрена структура сертификата. Было определено, что сертификат содержит в себе версию сертификата, серийный номер от издателя, идентификатор алгоритма подписи сертификата, имя издателя и имя субъекта, период действия сертификата, открытый ключ (параметры и алгоритм генерации), уникальные идентификаторы издателя и субъекта, дополнительную информацию об использовании ключа, а также цифровую подпись сертификата.
5. Подписание своего отчета:
   1. Был рассмотрен способ подписи PDF-документа и проверки его подписи при помощи сертификата средствами Adobe Acrobat Reader. При сохранении целостности PDF-документа подпись успешно проходила проверку, а после изменения PDF-документа – проверка на целостность проходила неуспешно.

Были получены практические навыки работы с рассматриваемыми алгоритмами с использованием приложения CrypTool 1 и 2.