**КРИПТОГРАФИЯ**

**ВОПРОСЫ**

[1. Односторонние функции.](#_1akmw1hfyv0o)

[2. Хеш-функции.](#_ezchie1lvuty)

[3. Гипотеза P <> ( НЕ РАВНО) NP.](#_az581n7yqu4e)

[4. Симметричное шифрование.](#_sjhdyyjhrn7e)

[5. Асимметричное шифрование.](#_f326grcc65z8)

[6. Доказательства с нулевым разглашением.](#_5xkc1z7ewghk)

[7. Протоколы аутентификации и электронной подписи.](#_algo41b75zjq)

[8. Неотслеживаемость. Электронная монета.](#_n2o6s96lfb5h)

[9. Протокол привязки к биту.](#_cp9f6bo3emch)

[10. Протоколы электронного голосования.](#_g5jlvol44soh)

[11. Порождение простых чисел и проверка чисел на простоту. Сложность теоретико-числовых алгоритмов.](#_869dptsvihqi)

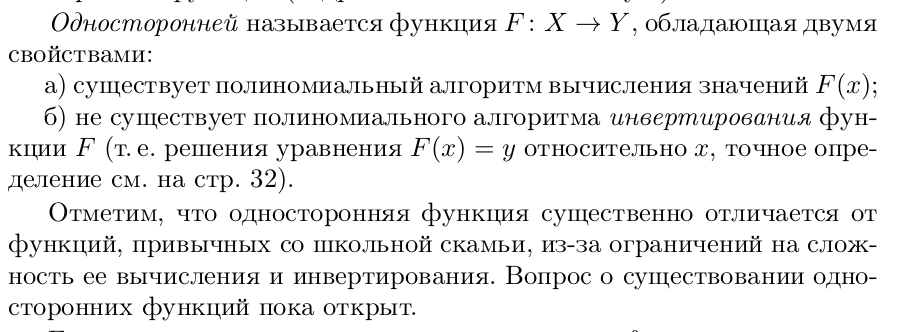
[12. Протоколы разделения секрета.](#_ncuiqy3wu0ux)

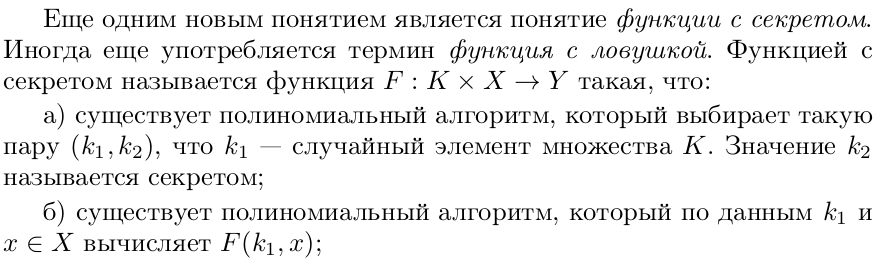
[13. Криптография на эллиптических кривых.](#_y6wxxwxphyhs)

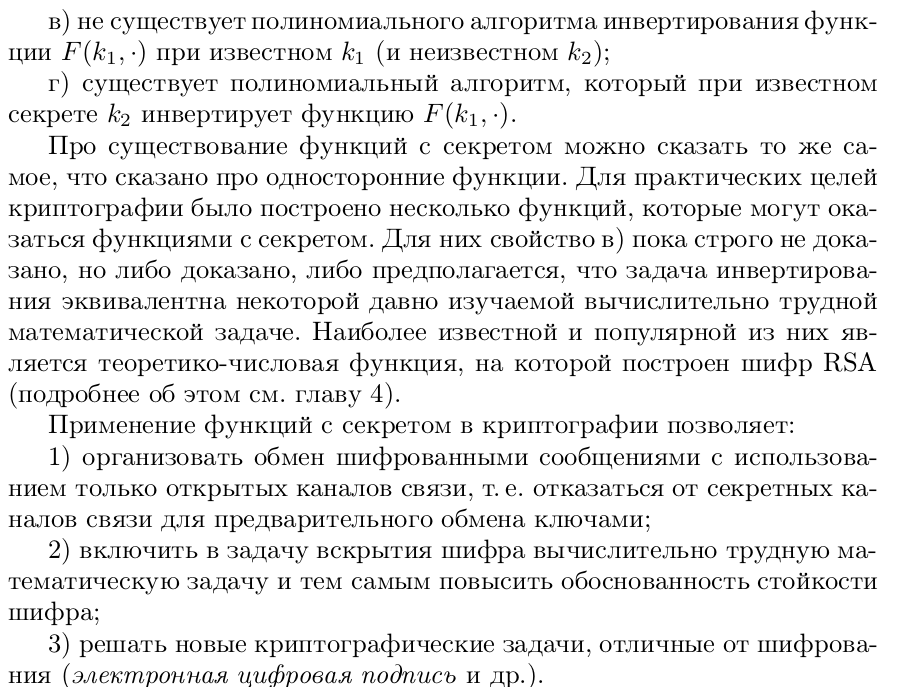
**ОТВЕТЫ**

**кТо? ГдЕ? кОгДа?  
Короче, я буду вставлять то, что нашёл, т.к. это похоже на правду, но я ебал эти вопросы. *ЛаДнО.ПрОхЛаДнО***

## [1. **Односторонние функции.**](#_1akmw1hfyv0o)







## [2. **Хеш-функции.**](#_ezchie1lvuty)

Обычная хеш-функция отображает некоторый массив в битовый массив заданной длины.

Криптографическая хеш-функция - это математический алгоритм, который отображает данные произвольного размера в битовый массив фиксированного размера и отвечает вопросам криптографической стойкости.

Результат, производимый хеш-функцией, называется «хеш-суммой» или же просто «хешем», а входные данные часто называют «сообщением».

Для идеальной хеш-функции выполняются следующие условия:

а) хеш-функция является детерминированной, то есть одно и то же сообщение приводит к одному и тому же хеш-значению

b) значение хеш-функции быстро вычисляется для любого сообщения

c) невозможно найти сообщение, которое дает заданное хеш-значение

d) невозможно найти два разных сообщения с одинаковым хеш-значением

e) небольшое изменение в сообщении изменяет хеш настолько сильно, что новое и старое значения кажутся не коррелирующими.

<https://habr.com/ru/post/534596/>

Сформулируем основные требования, предъявляемые к криптографическим хеш-функциям:

* хеш-функция должна быть применима к сообщению любого размера;
* вычисление значения функции должно выполняться достаточно быстро;\*
* при известном значении хеш-функции должно быть трудно (практически невозможно) найти подходящий прообраз М ;
* при известном сообщении М должно быть трудно найти другое сообщение М’ с таким же значением хеш-функции, как у исходного сообщения;
* должно быть трудно найти какую-либо пару случайных различных сообщений с одинаковым значением хеш-функции.

В настоящее время предложены и практически используются различные специальные алгоритмы для вычисления хеш-функции. Наиболее известными алгоритмами являются MD5, SHA-1, SHA-2 и другие версии SHA, а также отечественный алгоритм, изложенный в ГОСТ Р 34.11-94.

*Алгоритм MD5* появился в начале 90-х годов ХХ века в результате усовершенствования алгоритма формирования хеш-функции MD4. Символы в названии "MD" означают Message Digest – краткое изложение сообщения. Автор алгоритмов MD4 и MD5 – Р. Ривест (R.Rivest). В результате использования MD5 для произвольного сообщения формируется 128-битное хеш-значение. Входные данные обрабатываются блоками по 512 бит. В алгоритме используются элементарные логические операции (инверсия, конъюнкция, сложение по модулю 2, циклические сдвиги и др.), а также обыкновенное арифметическое сложение. Комплексное повторение этих элементарных функций алгоритма обеспечивает то, что результат после обработки хорошо перемешан. Поэтому маловероятно, чтобы два сообщения, выбранные случайно, имели одинаковый хеш-код. Алгоритм MD5 имеет следующее свойство: каждый бит полученного хеш-значения является функцией от каждого бита входа. Считается, что MD5 является наиболее сильной хеш-функцией для 128-битного хеш-значения.

*Алгоритм SHA* (Secure Hash Algorithm – Безопасный хеш-алгоритм) был разработан национальным институтом стандартов и технологии (NIST) США и опубликован в качестве американского федерального информационного стандарта в 1993 году. SHA-1, как и MD5, основан на алгоритме MD4. SHA-1 формирует 160-битное хеш-значение на основе обработки исходного сообщения блоками по 512 бит. В алгоритме SHA-1 также используются простые логические и арифметические операции. Наиболее важным отличием SHA-1 от MD5 является то, что хеш-код SHA-1 на 32 бита длиннее, чем хеш-код MD5. Если предположить, что оба алгоритма одинаковы по сложности для криптоанализа, то SHA-1 является более стойким алгоритмом. Используя атаку методом грубой силы (лобовую атаку), труднее создать произвольное сообщение, имеющее данный хеш-код, а также труднее создать два сообщения, имеющие одинаковый хеш-код.

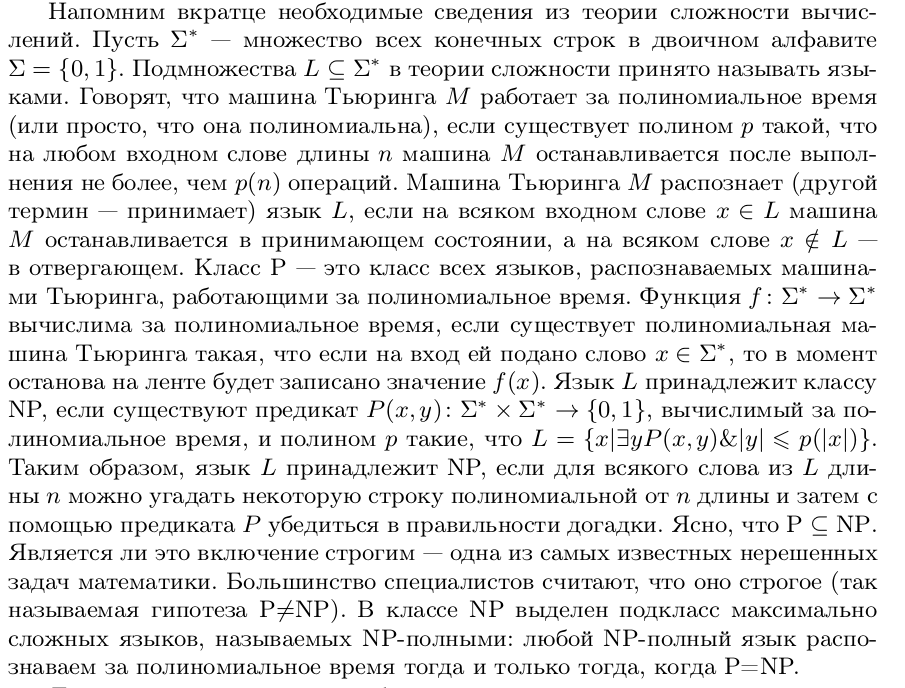
В 2001 году национальный институт стандартов и технологии США принял в качестве стандарта три хеш-функции с большей длиной хеш-кода, чем у SHA-1. Часто эти хеш-функции называют SHA-2 или SHA-256, SHA-384 и SHA-512 (в названии указывается длина создаваемого алгоритмами хеш-кода). Эти алгоритмы отличаются не только длиной создаваемого хеш-кода, но и используемыми внутренними функциями и длиной обрабатываемого блока (у SHA-256 длина блока – 512, а у SHA-384 и SHA-512 длина блока – 1024 бита). Постепенные усовершенствования алгоритма SHA ведут к увеличению его криптостойкости. Несмотря на отличия рассматриваемых алгоритмов друг от друга, все они являются дальнейшим развитием SHA-1 и MD4 и имеют похожую структуру.

В России принят **ГОСТ Р34.11-94**, который является отечественным стандартом для хеш-функций. Его структура довольно сильно отличается от структуры алгоритмов SHA-1,2 или MD5, в основе которых лежит алгоритм MD4. Длина хеш-кода, создаваемого алгоритмом ГОСТ Р 34.11-94, равна 256 битам. Алгоритм последовательно обрабатывает исходное сообщение блоками по 256 бит справа налево. Параметром алгоритма является стартовый вектор хеширования – произвольное фиксированное значение длиной также 256 бит. В алгоритме ГОСТ Р 34.11-94 используются операции перестановки, сдвига, арифметического сложения, сложения по модулю 2. В качестве вспомогательной функции в ГОСТ 34.11-94 используется алгоритм по ГОСТ 28147-89 в режиме простой замены.

## [3. **Гипотеза P <> ( НЕ РАВНО) NP.**](#_az581n7yqu4e)

Сначала дам про равенство двух классов.  
Технически проблему перебора формулируют так: совпадают ли классы P и NP? Класс P — это те задачи, которые могут быть решены «быстро». А класс NP — это переборные задачи. Если P равно NP, то окажется, что все переборные задачи могут быть быстро решены. А если не равно, то не все.

Неформально можно сказать так: переборная задача — эта задача, в которой нужно найти объект, удовлетворяющий каким-то условиям. Причем эти условия легко проверить, но возможных кандидатов много. Пример такой задачи: есть ребус в виде кружков, соединенных линиями, и нужно расставить в кружках буквы А, Б и В (одну букву в каждом), причем требуется, чтобы кружки, соединенные линией, содержали разные буквы. Ясно, что когда буквы уже расставлены, то надо просто проверить условие для всех линий, это несложно. Зато вариантов расстановок очень много, и если перебирать их все, то это будет очень долго. Для этой задачи никакого существенно более быстрого способа пока не открыли. Существует такой способ или нет — в этом как раз и состоит проблема перебора.



## 

## [4. **Симметричное шифрование.**](#_sjhdyyjhrn7e)

<https://encyclopedia.kaspersky.ru/glossary/symmetric-encryption/>

<https://thecode.media/symmetric/> *(более простой материал)*

**Симметричное шифрование** — это способ шифрования данных, при котором один и тот же ключ используется и для кодирования, и для восстановления информации. До 1970-х годов, когда появились первые асимметричные шифры, оно было единственным криптографическим методом.

**Принцип работы**

Симметричное шифрование работает так:

1. Есть данные, которые нужно зашифровать.
2. Есть ключ шифрования. С его помощью данные шифруются по какому-то алгоритму.
3. Тот, кто обладает ключом и знает алгоритм, может расшифровать сообщение
4. Если не знать ключа, то расшифровать данные почти невозможно. Почему почти — расскажем в статье про криптостойкость.

Слово «симметричный» связано с тем, что для шифрования и расшифровки нужен один и тот же ключ. Ключом может быть что угодно: число, слово, фраза, фрагмент текста или файл. Выбор ключа зависит от способа шифрования, но общее правило в симметричном шифровании такое: чем длиннее ключ, тем сложнее его взломать.

Пример: шифрования по Цезарю, где ключом выступает число сдвига алфавита, при помощи данного ключа можно как зашифровать сообщение, так и расшифровать.

## **Виды алгоритмов симметричного шифрования**

В зависимости от принципа работы алгоритмы симметричного шифрования делятся на два типа:***блочные;потоковые*.** *Блочные* алгоритмы шифруют данные блоками фиксированной длины (64, 128 или другое количество бит в зависимости от алгоритма). Если все сообщение или его финальная часть меньше размера блока, система дополняет его предусмотренными алгоритмом символами, которые так и называются *дополнением*. К актуальным блочным алгоритмам относятся:

* [AES](https://encyclopedia.kaspersky.ru/glossary/aes/)
* -ГОСТ 28147-89
* RC5
* Blowfish
* Twofish

Потоковое шифрование данных предполагает обработку каждого бита информации с использованием *гаммирования*, то есть изменения этого бита с помощью соответствующего ему бита псевдослучайной секретной последовательности чисел, которая формируется на основе ключа и имеет ту же длину, что и шифруемое сообщение. Как правило, биты исходных данных сравниваются с битами секретной последовательности с помощью логической операции XOR (исключающее ИЛИ, на выходе дающее 0, если значения битов совпадают, и 1, если они различаются).

Потоковое шифрование в настоящее время используют следующие алгоритмы:

* [RC4](https://encyclopedia.kaspersky.ru/glossary/rc4/)
* Salsa20
* HC-256
* WAKE

**Использование**

Почти всё общение в современных мессенджерах построено на симметричном шифровании.

Все наши банковские платежи, переводы и онлайн-оплата тоже шифруется симметричным методом.

Когда мы ставим замок на дверь и даём ключ всем жильцам — это тоже симметричное шифрование. Замок — это шифр, а ключ — это ключ расшифровки. Если вставить не тот ключ, то ничего не произойдёт. Открыть дверь можно только тем ключом, который был от этого замка, и сделать это может только тот, у кого есть этот ключ.

Любая защищенная линия связи, от правительственной до частной, использует симметричное шифрование.

## 

## [5. **Асимметричное шифрование.**](#_f326grcc65z8)

**\*ВступлениЕ\***

Мы зашифровали письмо, но как его отправить нашему другу? Отправлять в соц. сетях или текстовым сообщением — не самая лучшая затея. И это приводит нас к новому типу шифрования.

**В ассиметричном шифровании используется 2 ключа** — открытый и закрытый(тайный). Открытый ключ для шифрования, закрытый — для дешифрования.

<https://encyclopedia.kaspersky.ru/glossary/asymmetric-encryption/>

## **Асимметричное шифрование** — это метод шифрования данных, предполагающий использование двух ключей — открытого и закрытого. Открытый (публичный) ключ применяется для шифрования информации и может передаваться по незащищенным каналам. Закрытый (приватный) ключ применяется для расшифровки данных, зашифрованных открытым ключом. Открытый и закрытый ключи — это очень большие числа, связанные друг с другом определенной функцией, но так, что, зная одно, крайне сложно вычислить второе.

Асимметричное шифрование используется для защиты информации при ее передаче, также на его принципах построена **работа электронных подписей.**

**Принцип действия**

Схема передачи данных между двумя субъектами (А и Б) с использованием открытого ключа выглядит следующим образом:

* Субъект А генерирует пару ключей, открытый и закрытый (публичный и приватный).
* Субъект А передает открытый ключ субъекту Б. Передача может осуществляться по незащищенным каналам.
* Субъект Б шифрует пакет данных при помощи полученного открытого ключа и передает его А. Передача может осуществляться по незащищенным каналам.
* Субъект А расшифровывает полученную от Б информацию при помощи секретного, закрытого ключа.

В такой схеме перехват любых данных, передаваемых по незащищенным каналам, не имеет смысла, поскольку восстановить исходную информацию возможно только при помощи закрытого ключа, известного лишь получателю и не требующего передачи.

**Алгоритмы**

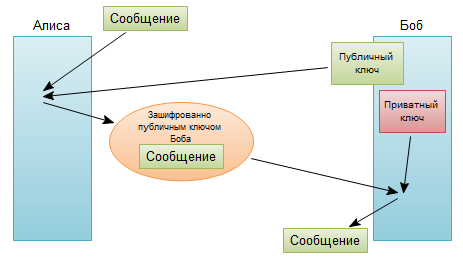
* RSA (аббревиатура от Rivest, Shamir и Adelman, фамилий создателей алгоритма) — алгоритм, в основе которого лежит вычислительная сложность факторизации (разложения на множители) больших чисел. Применяется в защищенных протоколах SSL и TLS, стандартах шифрования, например в PGP и S/MIME, и так далее. Используется и для шифрования данных, и для создания цифровых подписей.
* DSA (Digital Signature Algorithm, «алгоритм цифровой подписи») — алгоритм, основанный на сложности вычисления дискретных логарифмов. Используется для генерации цифровых подписей. Является частью стандарта DSS (Digital Signature Standard, «стандарт цифровой подписи»).
* Схема Эль-Гамаля — алгоритм, основанный на сложности вычисления дискретных логарифмов. Лежит в основе DSA и устаревшего российского стандарта ГОСТ 34.10–94. Применяется как для шифрования, так и для создания цифровых подписей.
* ECDSA (Elliptic Curve Digital Signature Algorithm) — алгоритм, основанный на сложности вычисления дискретного логарифма в группе точек эллиптической кривой. Применяется для генерации цифровых подписей, в частности для подтверждения транзакций в криптовалюте Ripple.

**Плюсы и минусы**

* Плюс таких алгоритмов в том, что для передачи зашифрованных сообщений можно использовать открытый канал связи. Даже если злоумышленник перехватит сообщение, он не сможет прочитать его без секретного ключа. Но чтобы всё было именно так, нужно, чтобы ключ был достаточно длинный — 1024 бит и выше.
* Минус асимметричного шифрования очевиден — оно работает только в одну сторону. Чтобы такое общение было двусторонним, каждый должен предоставить другому свой открытый ключ.

**Применение**

* Когда нужно установить защищённый канал связи в интернете. Мы знаем, что изначально наши данные может перехватить кто угодно, поэтому нам нужно обмениваться только теми ключами, которые можно показывать другим.



Отправитель перед отправкой получает публичный ключ и шифрует сообщение данным ключом, после чего данное сообщение можно расшифровать только приватным ключом, который хранится в секрете у принимающей стороны.

* Для создания цифровых подписей и сертификатов. Это позволяет проверять подлинность цифровых документов, а также убедиться, что его отправил именно владелец сертификата.

## [6. **Доказательства с нулевым разглашением.**](#_5xkc1z7ewghk)

<https://habr.com/ru/post/94901/>

Если коротко, то это протокол доказательства знания секрета без его разглашения. Т.е., например, А знает секрет, а Б хочет проверить, знает ли А секрет, но проверка будет проходить так, чтобы Б не узнал секрета. Идея заключается в том, что Б задаёт ряд вопросов, на которые А может ответить правильно, если знает секрет, или даже если не знает, но с некоторой вероятность. Например, пусть А не знает секрет. Допустим, что вероятность ответить правильно на вопрос 1/2. Тогда вероятность ответить на n вопросов правильно равна 1/2^n, т.е. пренебрежимо мала для больших n. В этом суть данного протокола. Подробнее рассказано в статье выше.

## 

## 7. **Протоколы аутентификации и электронной подписи.**

## 

## **Идентификация** (англ. identification) - процесс распознавания сущностей путем присвоения им уникальных меток (идентификаторов, логинов).

## 

## **Аутентификация** (англ. authentication) - проверка соответствия (подлинности) сущности предъявленному ею идентификатору.

**Протокол аутентификации** – криптографический протокол, в ходе которого одна сторона убеждается:

1) в идентичности другой стороны, вовлеченной в протокол

2) в активности другой стороны во время или непосредственно перед моментом приобретения доказательства.

Аутентификация может быть **односторонней** (когда клиент доказывает свою подлинность серверу) и **двусторонней** или взаимной (это обоюдная аутентификация между сторонами обмена информацией).

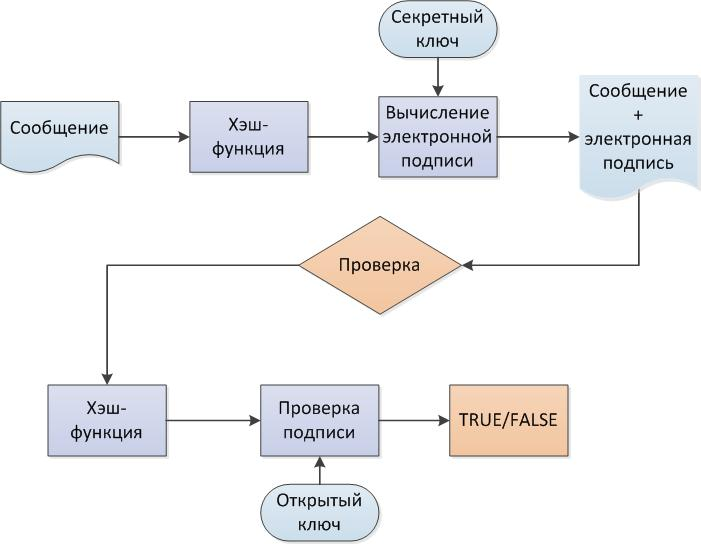
Пример односторонней аутентификации – процедура входа в систему (WINDOWS NT). Пример двусторонней – использование протокола KERBEROS (WINDOWS 2000).

Характеристики протоколов аутентификации:

* вычислительная эффективность – количество операций, необходимых для выполнения протокола;
* коммуникационная эффективность – данное свойство отражает количество сообщений и их длину, необходимую для осуществления аутентификации;
* наличие третьей стороны – примером третьей стороны может служить доверенный сервер распределения симметричных ключей или сервер, реализующий дерево сертификатов открытых ключей;
* основа гарантий безопасности – примером могут служить протоколы, обладающие свойством доказательства с нулевым знанием;
* хранения секрета – имеется в виду способ хранения критичной ключевой информации.

**Электронная цифровая подпись** – реквизит электронного документа, предназначенный для защиты данного документа от подделки, полученный в результате криптографического преобразования информации с использованием закрытого ключа ЭЦП и позволяющий идентифицировать владельца сертификата ключа подписи, а также установить отсутствие искажения информации в электронном документе.

**Как работает??** [**https://www.e-notary.ru/vse-ob-elektronnoj-cifrovoj-podpisi/**](https://www.e-notary.ru/vse-ob-elektronnoj-cifrovoj-podpisi/)



* Программа-шифратор преобразовывает файл в строку символов(или последовательность битов просто) — **хеш**. Разные документы переводятся в разный набор символов, а одинаковые — в одинаковый.
* После хеширования программа зашифровывает строку с помощью закрытого ключа. Процесс аналогичен тому, как что-то кладется в коробку и запечатывается замком. Это и есть ЭЦП.
* Документ пересылается получателю, к нему прикладывается зашифрованный хеш и сертификат, где указаны контактные данные отправителя.
* С помощью сертификата адресат «распечатывает» хеш и может посмотреть документ. Если у него есть ЭЦП, обратное отправление документа происходит также по рассмотренному нами алгоритму.

**Протокол на базе алгоритма RSA** <https://www.sites.google.com/site/anisimovkhv/learning/kripto/lecture/tema12>

Этап 1. Отправитель А генерирует открытый {e=5, n=91} и закрытый {d=29} ключи, публикует открытый ключ.

Этап 2. Отправка сообщения и электронной подписи (выполняет отправитель А).

| **n** | **Описание операции** | **Пример** |
| --- | --- | --- |
| **1** | **Вычисление хеш-образа h = h(T),**  **где T – исходное сообщение;**  **h(T) – хеш-функция (для MD5 длина хеш-образа 128 бит).** | **h = 7** |
| **2** | **Выработка электронной подписи s = hd mod n,**  **где d – закрытый ключ отправителя A;**  **n – часть открытого ключа отправителя A.** | **s = 729 mod 91 = 63** |
| **3** | **Отправка получателю Б исходного сообщения Т и электронной подписи s.** |  |

Этап 3. Получение сообщения и проверка электронной подписи (выполняет получатель Б).

| **№ п/п** | **Описание операции** | **Пример** |
| --- | --- | --- |
| **1** | **Вычисление хеш-образа по полученному сообщению h’ = h(T’),**  **где T’ – полученное сообщение.** | **h' = 7** |
| **2** | **Вычисление хеш-образа из электронной подписи h = se mod n,**  **где e и n – открытый ключ отправителя A.** | **h = 635 mod 91 = 7** |
| **3** | **Если h' = h, то получатель Б делает вывод, что полученное сообщение T’ = T и оно действительно отправлено А.** | **7 = 7** |

## 

## [8. **Неотслеживаемость. Электронная монета.**](#_n2o6s96lfb5h)

<http://citforum.ru/security/cryptography/yaschenko/21.html>

<https://studme.org/215082/informatika/elektronnye_monety>

Обеспечение неотслеживаемости — одна из задач криптографии. Проблема с эл деньгами кредитками и прочим в том, что кредитная карточка представляет собой носитель информации, который при каждом платеже идентифицирует своего владельца. И если он использует ее для покупки билетов на транспорт, то можно отследить все его поездки, для каждого владельца кредитных карточек возможно собирать информацию о том, какие товары и где он покупает, какими услугами пользуется, какие культурно-зрелищные мероприятия посещает и т.д. То же самое с эл документами и компьютерными хранилищами информации - можно собирать на людей досье а это плохо

Для обеспечения целостности и неотслеживаемости есть криптографические протоколы - эти понятия не формализуются

Под *электронными деньгами* мы будем понимать электронные платежные средства, обеспечивающие неотслеживаемость(как наличные)

В автономных системах электронных платежей используются *три основные транзакции.*

1. Снятие со счета. Эта транзакция по своему назначению не отличается от аналогичной транзакции в централизованных системах.

2. Платеж. Один клиент (покупатель) передает другому (продавцу) электронную монету. Последний проверяет подлинность монеты без обращения к банку.

3. Депозит. Клиент кладет электронную монету на свой счет в банке. В отличие от централизованных систем, транзакцию депозита выполняет продавец.

*требования к платежным системам:*

Безопасность для банка. После ^-кратного выполнения транзакции снятия со счета, клиент не может создать более k различных электронных монет.

Неотслеживаемость. Наблюдая только транзакции снятия со счета и депозита, банк не может связать эти транзакции между собой, т.е. определить, кто из клиентов получил какую монету и кому ее заплатил.

Невозможность потратить монету дважды. Если клиент дважды расплатится одной и той же монетой, он будет идентифицирован банком, как только для обеих копий будет выполнена транзакция депозита. — отличает автономную от централизованной!

Защита от ложных обвинений. Клиенты защищены от ошибочных обвинений в повторном платеже одной и той же монетой.

Централизованная система не позволяет клиентам потратить одну и ту же банкноту дважды, а автономная лишь идентифицирует нарушителя в какой-то момент после повторного использования монеты.

*про безопасность транзакций:*

В транзакции *снятия со счета* покупатель получает подписанную

банком электронную банкноту на затребованную сумму. При этом счет

покупателя уменьшается на эту сумму. В транзакции *платежа* покупатель передает банкноту продавцу и указывает сумму платежа. Продавец, в свою очередь, передает эту информацию банку, который проверяет подлинность банкноты. Если банкнота подлинная, банк проверяет, не была ли она потрачена ранее. Если нет, то банк заносит банкноту в специальный регистр, зачисляет требуемую сумму на счет продавца,

уведомляет продавца об этом, и, если достоинство банкноты выше, чем

сумма платежа, возвращает покупателю «сдачу» (через продавца). С

помощью транзакции *депозита*, покупатель может положить «сдачу»

на свой счет в банке.

Безопасность банка основывается на невозможности подделать его

подпись для создания фальшивой банкноты. Для неотслеживаемости покупателя необходимо, чтобы банк, получив банкноту в транзакции платежа, не мог установить, кому она была выдана. То же относится и к «сдаче». Это, казалось бы, парадоксальное требование удовлетворяется с помощью схемы так называемой *подписи вслепую*: в транзакции снятия со счета банк подписывает не банкноту, а некоторую «абракадабру», из которой покупатель восстанавливает подписанную банкноту. Таким образом, неотслеживаемость гарантируется тем,

что банк просто не знает, что именно он подписал.

## 

## [9. **Протокол привязки к биту.**](#_cp9f6bo3emch)

[http://cryptography.ru/ref/протокол\_привязки\_к\_биту/](http://cryptography.ru/ref/%D0%BF%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB_%D0%BF%D1%80%D0%B8%D0%B2%D1%8F%D0%B7%D0%BA%D0%B8_%D0%BA_%D0%B1%D0%B8%D1%82%D1%83/)

[Примитивный криптографический протокол](http://cryptography.ru/docs/%d0%bf%d1%80%d0%b8%d0%bc%d0%b8%d1%82%d0%b8%d0%b2%d0%bd%d1%8b%d0%b9_%d0%ba%d1%80%d0%b8%d0%bf%d1%82%d0%be%d0%b3%d1%80%d0%b0%d1%84%d0%b8%d1%87%d0%b5%d1%81%d0%ba%d0%b8%d0%b9_%d0%bf%d1%80%d0%be%d1%82%d0%be/) с двумя участниками (отправителем и получателем), посредством которого отправитель передает бит информации, упакованный в так называемый [блоб](http://cryptography.ru/docs/%d0%b1%d0%bb%d0%be%d0%b1/) таким образом, что выполняются следующие два условия:

1. После того как блоб передан получателю (выполнен этап привязки), отправитель уже не может изменить значение заключенного в нем бита.

2. Получатель не может самостоятельно извлечь из блоба значение бита и узнает его только после того, как отправитель выполнит этап раскрытия блоба.

В целом протокол привязки к биту напоминает шифрование, но может быть интерактивным и, кроме того, не требуется существования эффективного алгоритма извлечения бита из блоба (при известном ключе).

* Говоря неформально, протоколом привязки к биту называется криптографический протокол с двумя участникам (отправителем и получателем), посредством которого отправитель передает произвольный бит b получателю следующим образом. Вначале выполняется *этап привязки*, на котором отправитель передает получателю (вообще говоря, интерактивно) информацию о b, называемую *блобом*, но не само значение . Затем на *этапе раскрытия* отправитель передает получателю значение b и доказывает ему, что это значение действительно соответствует блобу, полученному на этапе привязки. Этот процесс называется раскрытием блоба в значение b. Протокол привязки к биту должен удовлетворять следующим условиям.

1. Получатель до этапа раскрытия не может узнать по блобу значение передаваемого бита. Это условие называется условием *конфиденциальности*.
2. Отправитель на этапе раскрытия не может успешно доказать отправителю, что один и тот же блоб соответствует как 0, так и 1 Другими словами, отправитель не может раскрыть один и тот же блоб как в 0, так и в 1. Это условие называется условием *однозначности*.

Неформальный смысл протоколов привязки к биту можно проиллюстрировать с помощью следующего «протокола» из реальной жизни. На этапе привязки отправитель записывает значение бита b на листе бумаги, помещает этот лист в сейф и передает этот сейф (без ключа от него) получателю. В данном случае роль блоба играет сейф с находящимся в нем значением b. На этапе раскрытия отправитель передает получателю значение b и ключ от сейфа. Получатель открывает сейф с помощью полученного ключа и признает полученное значение b соответствующим блобу, если и только если оно совпадает со значением, записанным на листе бумаги из сейфа.

## 

## [10. **Протоколы электронного голосования.**](#_g5jlvol44soh)

<https://habr.com/ru/post/436560/>

**Протоколы голосования по существу являются частными случаями протоколов конфиденциального вычисления**

**Большинство классических протоколов электронного голосования работают следующим образом:**

1. **Избиратель получает токен в виде бюллетеня, который изменяет соответственно своему выбору. Разные избиратели получают разные бюллетени.**
2. **Избиратель шифрует бюллетень (используя особый тип шифрования, позволяющий работать магии электронного голосования, и отправляет его так, чтобы организаторы голосования получили зашифрованный бюллетень.**
3. **Организаторы публикуют зашифрованные бюллетени на доске объявлений, «публичном широковещательном канале с памятью», на жаргоне криптографов – говоря проще, на чём-то вроде Pastebin.**
4. **Организаторы комбинируют зашифрованные бюллетени для подсчёта зашифрованного итога. Затем они расшифровывают итог (но не сами бюллетени!) и публикуют результаты.**
5. **Получив результат и зашифрованные голоса, любой может проверить его корректность.**

**На 4-м шаге организаторы комбинируют криптограммы для создания новой криптограммы, шифрующей сумму отдельных голосов. Для этого схемы электронного голосования используют схему шифрования Enc(), в которой можно подсчитать Enc(v1 + v2), имея на руках только Enc(v1) и Enc(v2), и не зная ключа шифрования. Такие схемы шифрования называются гомоморфными.**

К примеру, если сильно упростить, избиратели США производят следующие действия:

1. Получают от организаторов бюллетень «Клинтон» и бюллетень «Трамп»
2. Пишут на одном бюллетене Enc(1), а на другом – Enc(0), используя в качестве ключа публичный ключ, выданный организаторами.

Зашифрованные бюллетени затем публикуются на доске объявлений вместе с ID избирателя. Все знают, кто проголосовал, но невозможно понять, за кого именно, поскольку каждые Enc(0) и Enc(1) уникальны, а мы используем сильное и рандомизированное шифрование. Если бы шифрование было детерминистское, избирателя можно было бы заставить раскрыть его голос, вычислив Enc(0) заново и сравнив его со значением на доске. Наконец, организаторы комбинируют все бюллетени за «Клинтон» и получают результат Enc(количество голосов за Клинтон), а потом то же делают с бюллетенями за «Трампа» и получают Enc(количество голосов за Трампа). Затем они берут ключ расшифровки и расшифровывают два этих значения, объявляя победителя.

***итог:* Основная проблема безопасной системы электронного голосования – публичная проверяемость. Это достигается публикацией зашифрованных бюллетеней на публично доступном форуме. Организаторы голосования также должны описать механизм, проводящий само подтверждение.**

**Правильность голосования подтверждается путём проведения авторизации каждого избирателя с уникальным ID и, давая избирателям доступ, который позволяет им проверить, что их бюллетень 1) засчитан и 2) не изменён.**

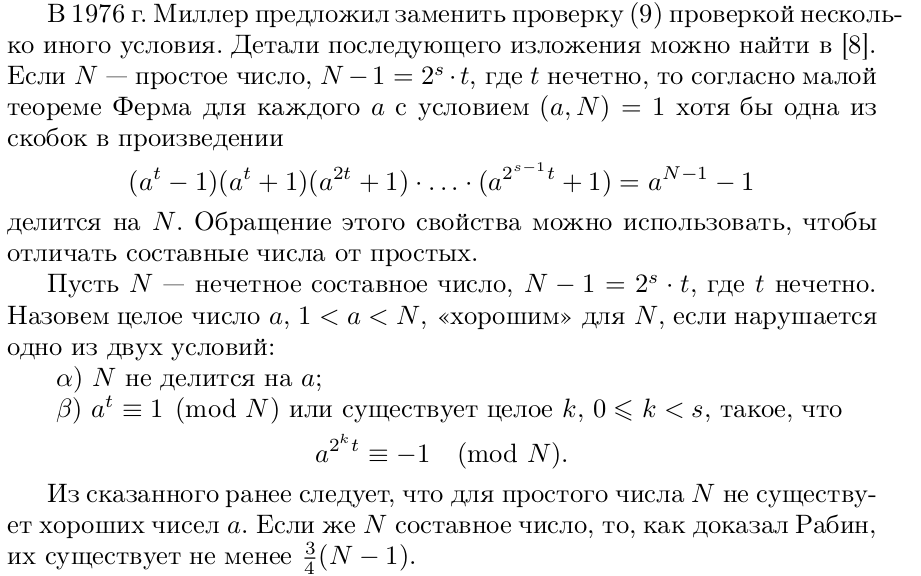
**Избирателей нельзя наказать за голоса за «неправильных» кандидатов благодаря сопротивлению принуждению, которое, в частности, достигается за счёт непредсказуемого и вероятностного шифрования.**

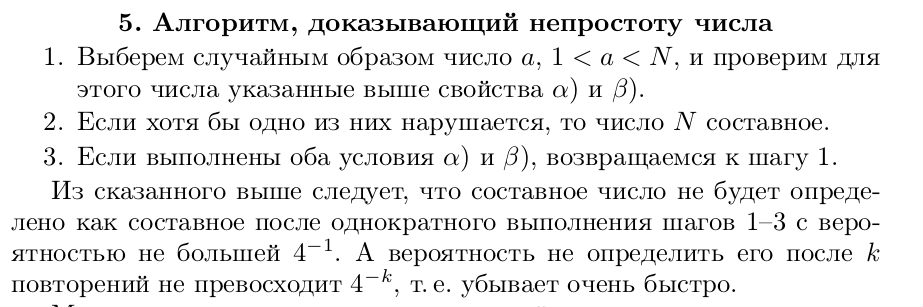
**Приватность бюллетеней гарантируется тем, что зашифрованные голоса не расшифровываются, а расшифровывается только итог, созданный при помощи гомоморфного шифрования.**

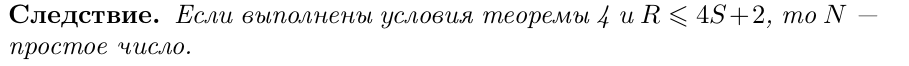
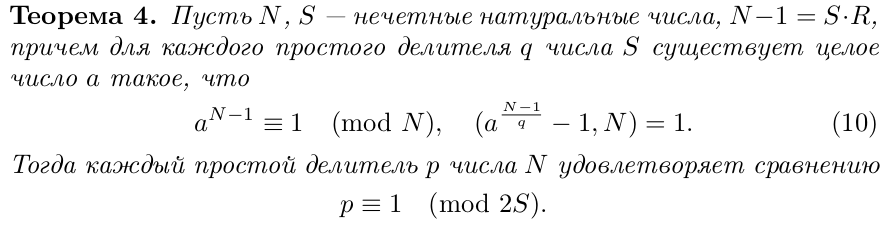
**(Жульничество не проходит, если мы заставляем избирателей публиковать криптографическое доказательство допустимости их бюллетеня при помощи НИНР- не интерактивное нулевое разглашение. (это уже про протокол доказательства с нулевым разглашением информации) ) Для превращения такого интерактивного протокола в неинтерактивный (такой, который можно отправить одним набором данных), строятся НИНР-доказательства. Мы самостоятельно играем в эту игру и симулируем проверяющего так, чтобы реальный проверяющий убедился в том, что мы не можем придумать НИНР-доказательство, не зная доказанного утверждения.**

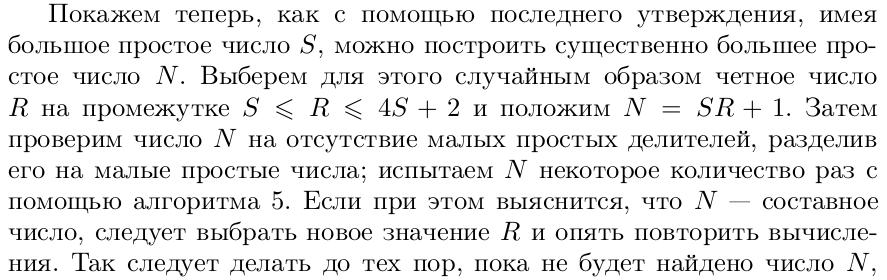
## [11. **Порождение простых чисел и проверка чисел на простоту. Сложность теоретико-числовых алгоритмов.**](#_869dptsvihqi)

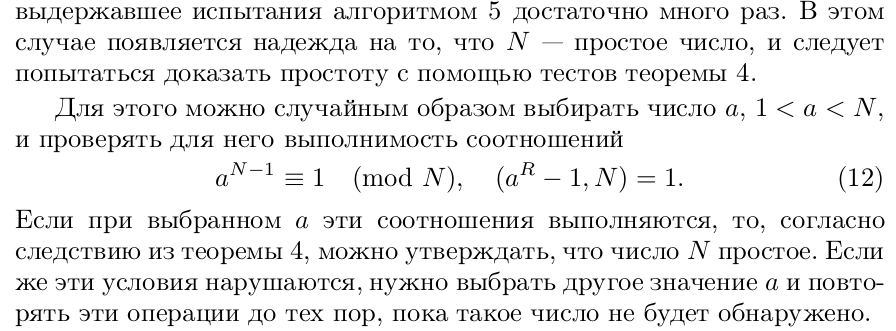
## 

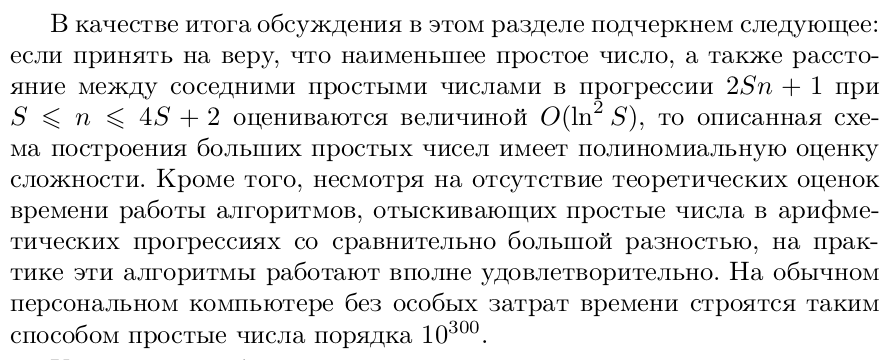


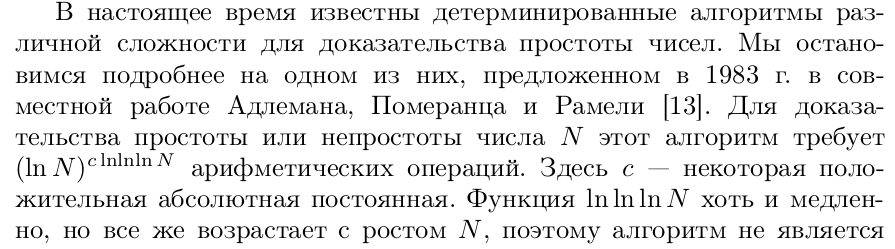


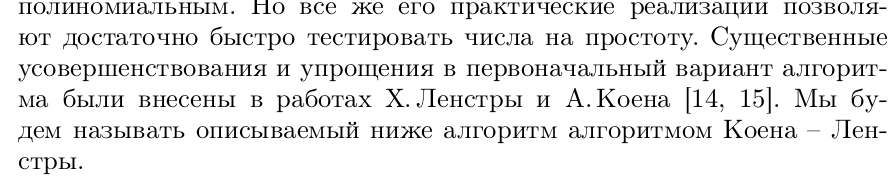


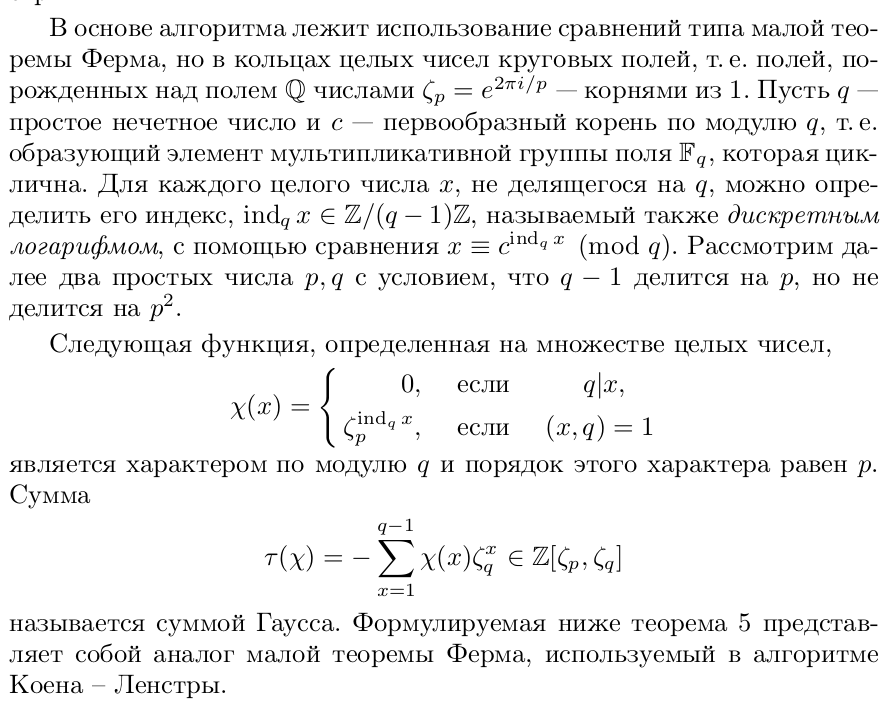


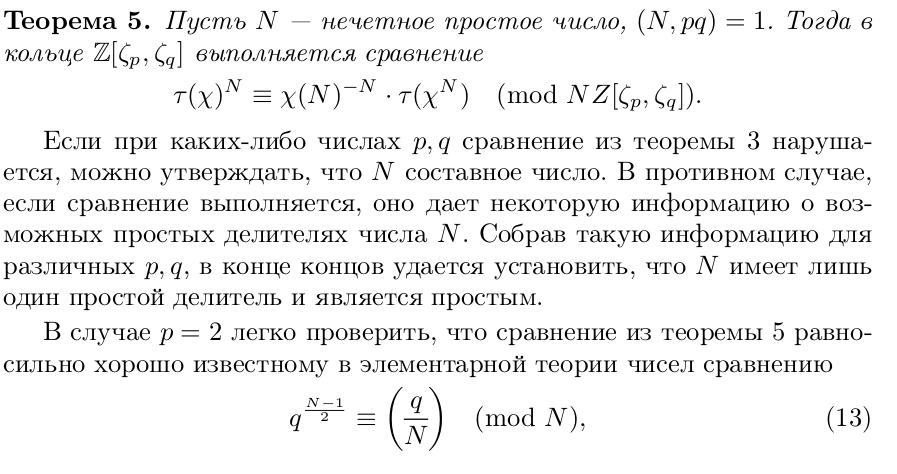


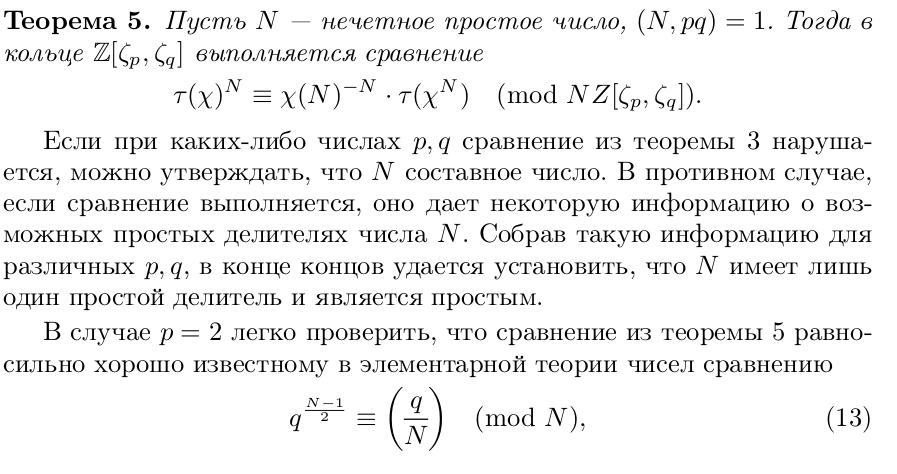


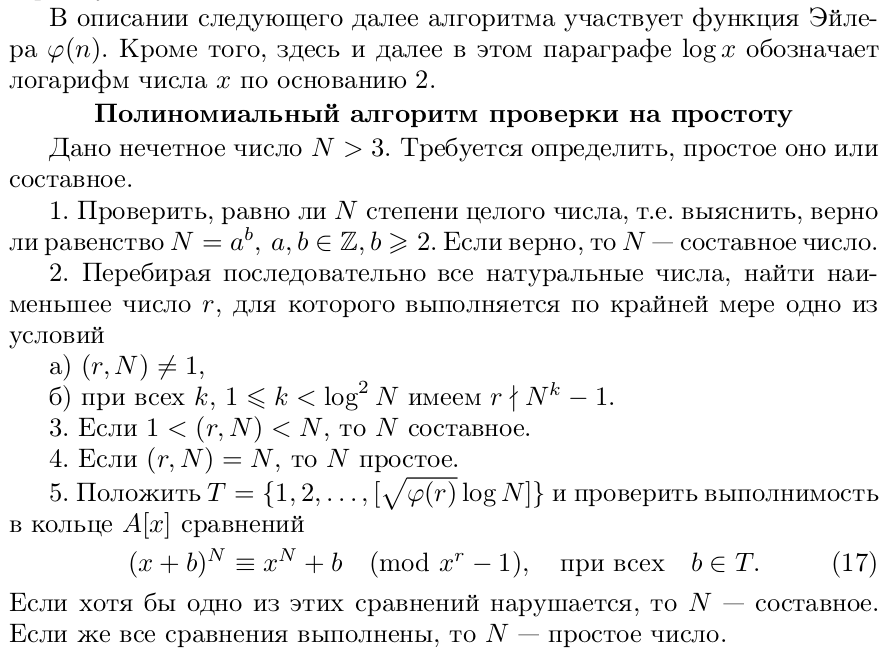












## [12. **Протоколы разделения секрета.**](#_ncuiqy3wu0ux)

В протоколе разделения секрета имеются n участников P1, . . . , Pn, которых мы будем называть процессорами, и один выделенный участник D, называемый дилером (или, иногда, лидером). Протокол состоит из двух фаз. На фазе разделения секрета дилер, знающий некоторый секрет s, генерирует n долей секрета s1, . . . , sn и посылает si процессору Pi по защищенному каналу связи. На фазе восстановления секрета любое подмножество из не менее чем t + 1 процессоров, где t — параметр протокола, однозначно восстанавливает секрет, обмениваясь сообщениями по защищенным каналам связи. А любое подмножество из не более чем t процессоров не может восстановить секрет.

Как и в других типах криптографических протоколов, в протоколе разделения секрета участники, вообще говоря, не доверяют друг другу и каждый из них может оказаться противником. В том числе и дилер. Можно ли обеспечить какую-либо защиту честных участников даже и в этом случае? Безусловно, нечестный дилер может просто саботировать выполнение протокола. Но если дилер пытается обмануть более хитрым способом, то от этого, оказывается, можно защититься следующим образом. Фаза разделения секрета начинается с того, что дилер публикует секрет s в «зашифрованном» виде (точнее было бы сказать — выполняет привязку к строке s, по аналогии с привязкой к биту). С помощью этой информации каждый процессор Pi может проверить, что значение si , полученное им от дилера, действительно является долей секрета s. Такой протокол называется **протоколом проверяемого разделения секрета**. В обычных схемах разделения секрета рассматривается пассивный противник, а именно, противником являются не более чем t участников, которые, объединив свои доли, пытаются получить какую-либо информацию о значении секрета. На фазе восстановления секрета в протоколе проверяемого разделения секрета действует активный противник: нечестные участники могут преследовать цель сорвать восстановление значения s честными участниками, посылая им вместо своих долей секрета любую другую информацию. От протокола требуется, чтобы честные участники, если их по крайней мере t + 1, всегда правильно восстанавливали значение s.

Конструкция основана на задаче дискретного логарифмирования.

В соответствии со схемой Шамира дилер выбирает случайный полином Q(x) = a0 + a1\*x + · · · + at\*x^t степени t, где a0 = s, вычисляет  (i = 0, 1, . . . , t) и публикует r0, . . . , rt. После этого для всякого j = 1, . . . , n дилер вычисляет sj = Q(j) и посылает это значение процессору Pj по защищенному каналу. Процессор Pj , проверяя равенство убеждается, что sj — доля секрета s. В самом деле:

Конструкцию протокола для фазы восстановления секрета рассмотрим в наиболее простом случае, когда дилер честный. На этой фазе каждый процессор Pj посылает каждому другому процессору Pi свою долю sj . Всякий честный участник Pi , получив некоторое значение sj от Pj , проверяет это значение, как описано выше, и отбрасывает все доли sj, не прошедшие проверку. Поскольку честных участников не менее t + 1, Pi получит по крайней мере t + 1 правильных долей секрета. Используя алгоритм восстановления секрета из схемы Шамира, Pi восстановит значение s.

стойкость данного протокола основывается на предположении о вычислительной трудности задачи дискретного логарифмирования. Поэтому, если в обычных схемах разделения секрета требуется, чтобы любое подмножество участников, не составляющее кворума, не получало никакой информации о секрете, то во многих схемах проверяемого разделения секрета такое подмножество лишь «не может восстановить» секрет. В том смысле, что для его восстановления требуется решить некоторую гипотетически трудную вычислительную задачу. В рассмотренном выше примере всякий участник мог бы узнать секрет s, если бы он умел вычислять дискретные логарифмы. Одно из возможных приложений схем разделения секрета — организация хранения криптографических ключей. Свойство проверяемости представляется далеко не лишним для таких приложений. Но круг приложений схем проверяемого разделения секрета существенно шире. Предположим, что с помощью описанной выше схемы разделены два секрета s1 и s2 и что оба эти секреты являются числами. Теперь представим себе ситуацию, что после этого потребовалась разделить секрет s = s1+s2. Конечно, это может сделать дилер с помощью того же протокола. А могут ли процессоры выполнить то же самое без участия дилера?

Пусть Q1(x) = a0 + a1\*x + · · · + at\*x^t и Q2(x) = b0 + b1\*x + · · · · · · + bt\*x^t — полиномы, которые использовались для разделения секретов s1 и s2 соответственно. Пусть  для i = 0, . . . , t. Для любого j = 1, . . . , n пусть  — доли секретов s1 и s2, полученные процессором Pj . Ясно, что Q(x) = Q1(x) + Q2(x) — также полином степени t и Q(0) = s. Поэтому каждый процессор Pj может вычислить долю sj секрета s просто по формулеЭти доли проверяемы с помощью значений 

## 

## [13. **Криптография на эллиптических кривых.**](#_y6wxxwxphyhs)

[https://habr.com/ru/post/188958/](https://vk.com/away.php?to=https%3A%2F%2Fhabr.com%2Fru%2Fpost%2F188958%2F&cc_key=)

[https://habr.com/ru/post/335906/](https://vk.com/away.php?to=https%3A%2F%2Fhabr.com%2Fru%2Fpost%2F335906%2F&cc_key=)

[https://ru.wikipedia.org/wiki/Эллиптическая\_кривая](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%BB%D0%B8%D0%BF%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%BA%D1%80%D0%B8%D0%B2%D0%B0%D1%8F)

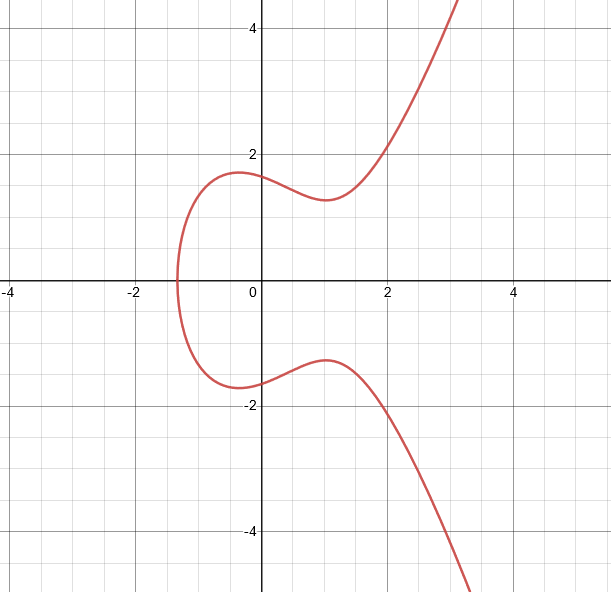
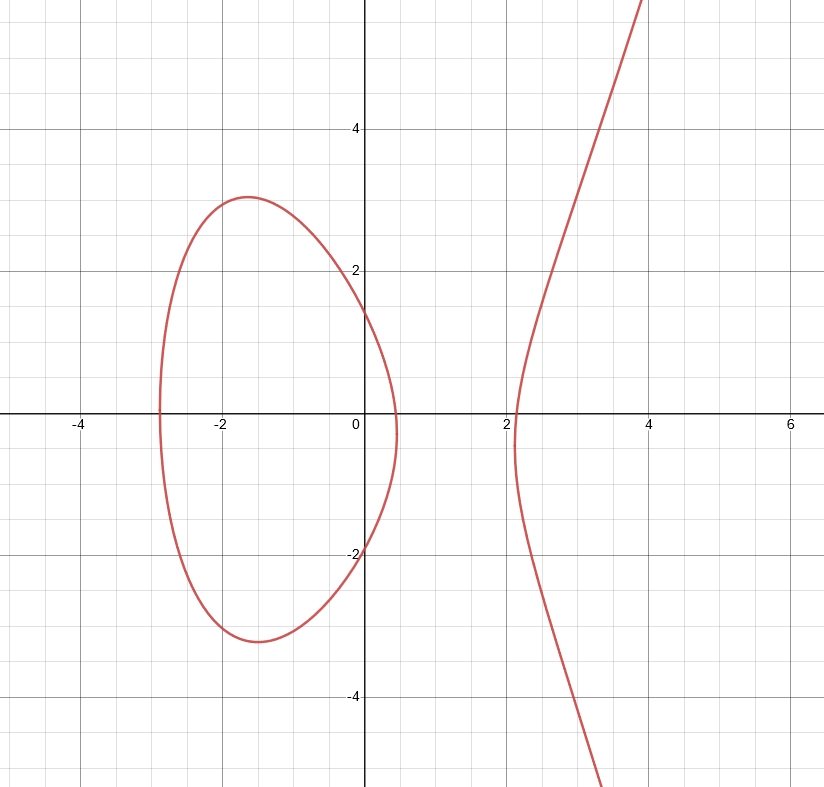
[https://www.youtube.com/watch?v=fWi7sAyGD\_k](https://www.youtube.com/watch?v=fWi7sAyGD_k&t=753s)

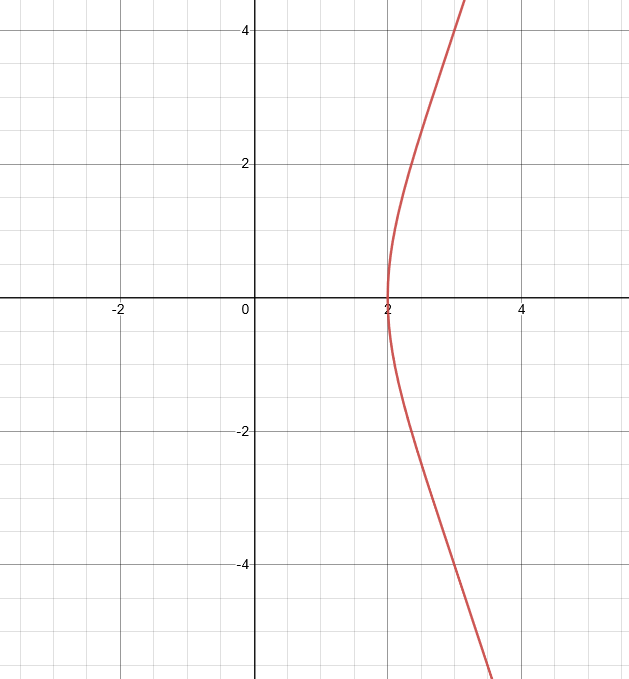
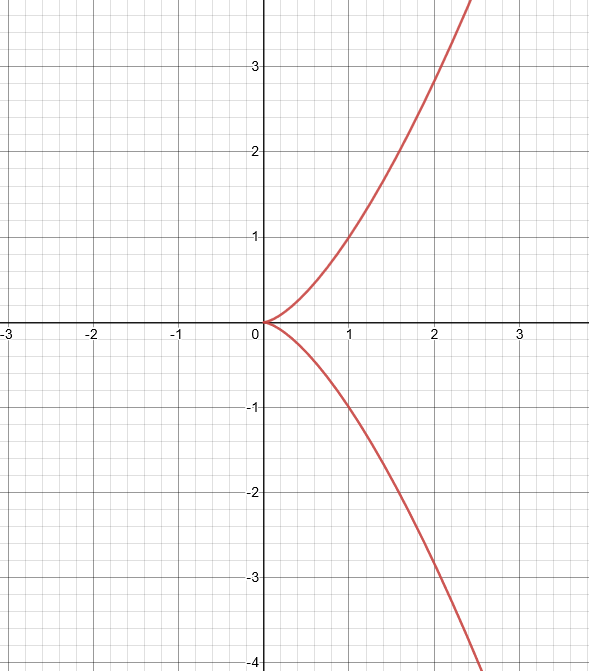
[https://ru.wikipedia.org/wiki/Эллиптическая\_криптография](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%BB%D0%B8%D0%BF%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%8F)

Задается уравнением вида



Как может выглядеть:

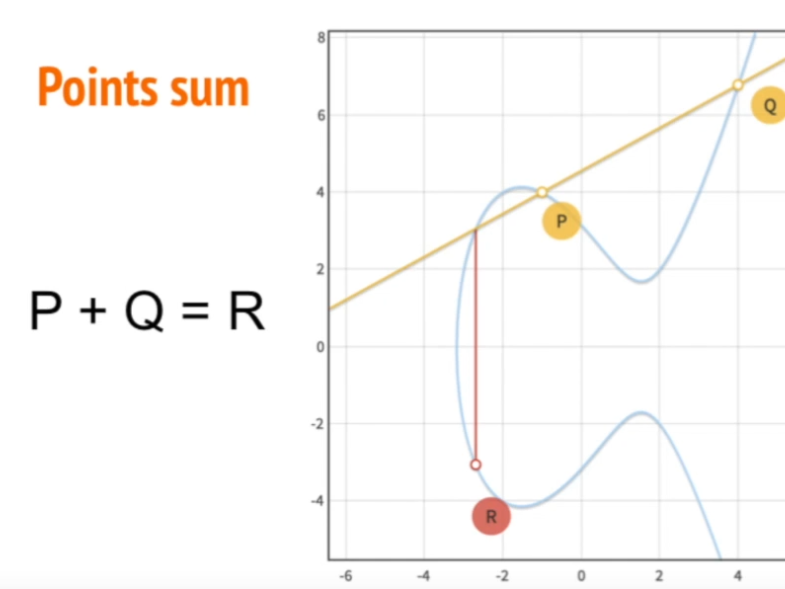


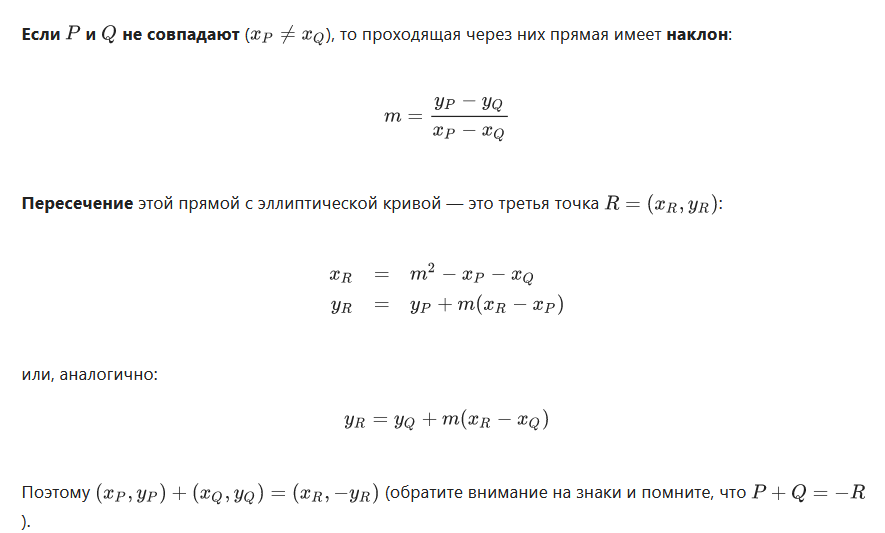


Будем рассматривать кривые вида

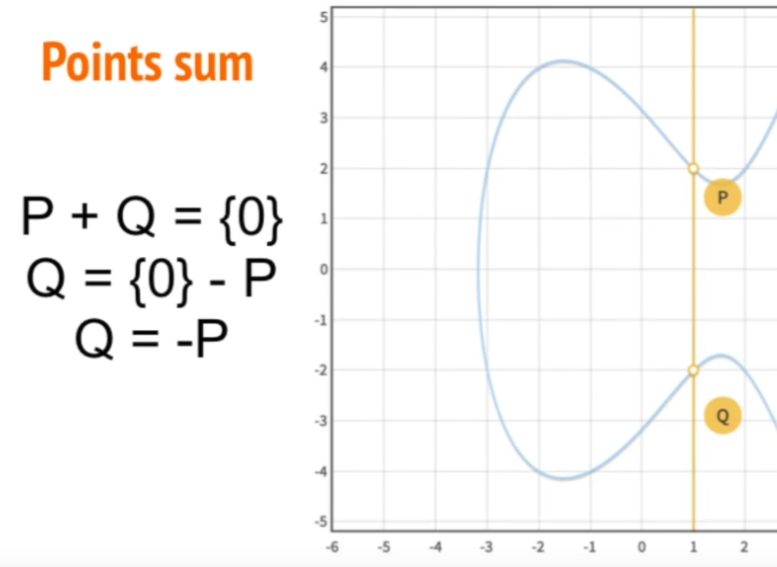
Для использования в криптографии необходимо условие (чтобы был кругляшок, не было острого угла и у каждой точки была пара (см. дальше))

Вводится понятие суммы точек

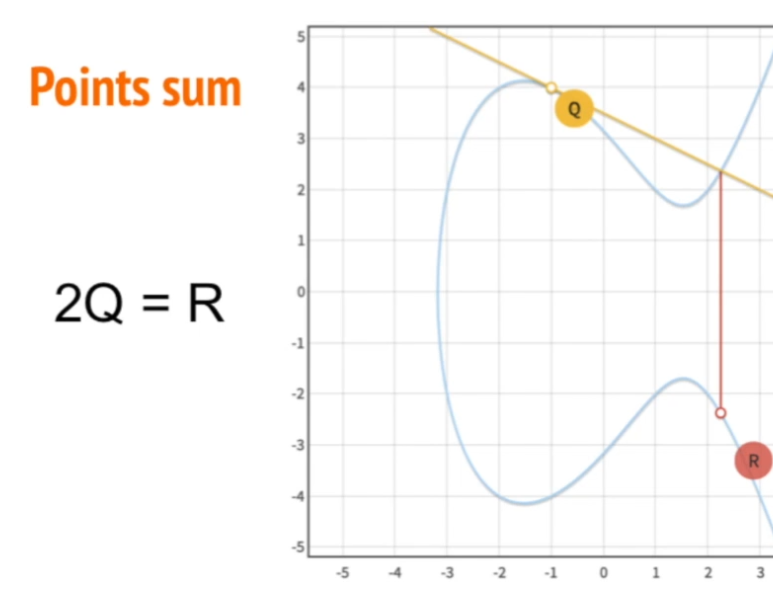




Вводится понятие “нулевой точки” {0}:

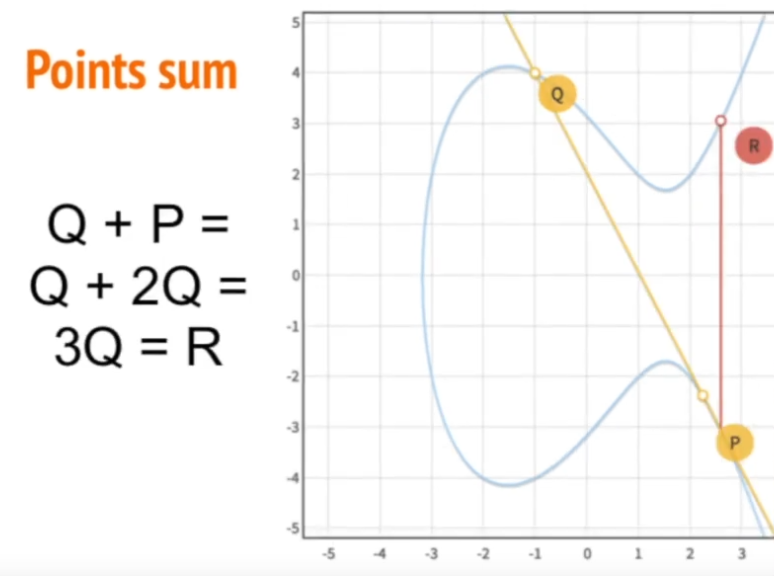


При складывании точки с самой собой проводим касательную

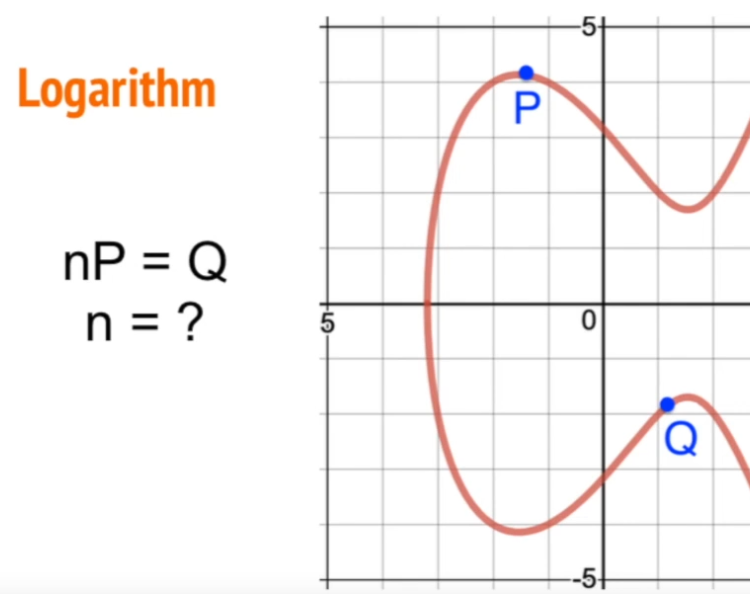


Таким образом получаем точку Q+Q=2Q

Продолжая складывать точку саму с собой получим

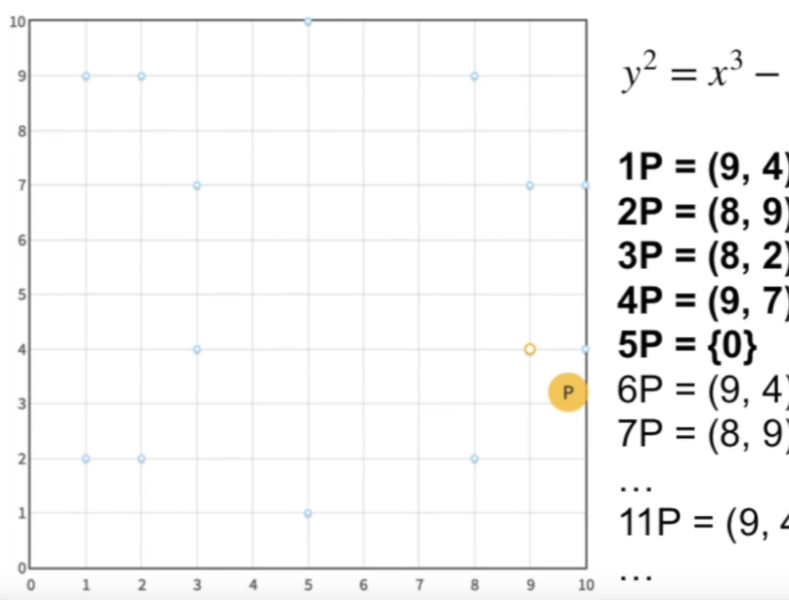


Зная P и Q нельзя быстро восстановить n:



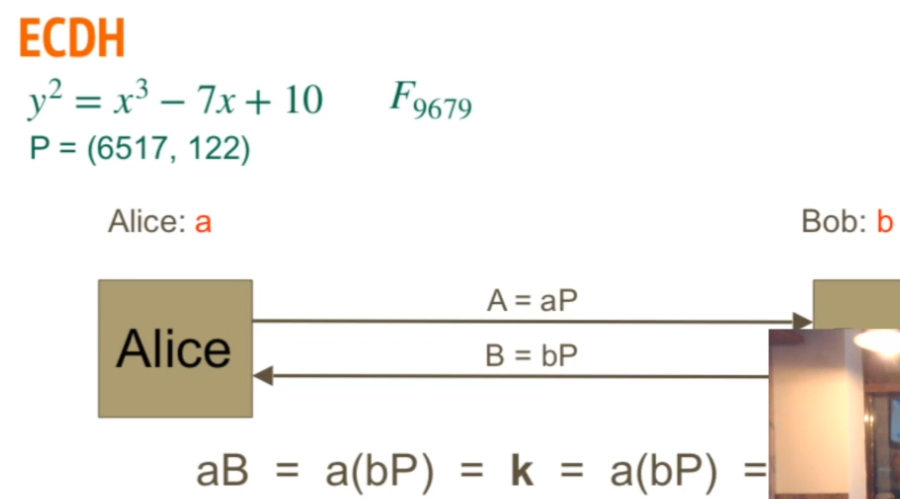
Чтобы избавиться от проблемы точности чисел с плавающей запятой операцию сложения точек рассматривают над определенным полем целых чисел.

Складывая точку саму с собой рано или поздно получим нулевую точку



Порядок этой подгруппы будет не больше количества всех точек в получившейся кривой.

Для шифрования каждый из собеседников выбирает произвольное число a и b (<p), вычисляют произведение aP и bP, обмениваются ими и умножают на свое число. (Зеленым - известно всем, красное - известно только им). Зная A и P нельзя быстро восстановить значение a (задача дискретного логарифма).



Для генерации aP можно умножить P само на себя как в быстром возведении в степень:

a = 18

P = P0

P = P+P (2P)

P = P+P (4P)

P = P+P (16P)

P = P+P0 (17P)

P = P+P0 (18P)

Однако для взлома не зная число **a** придется перебирать все линейным поиском.

