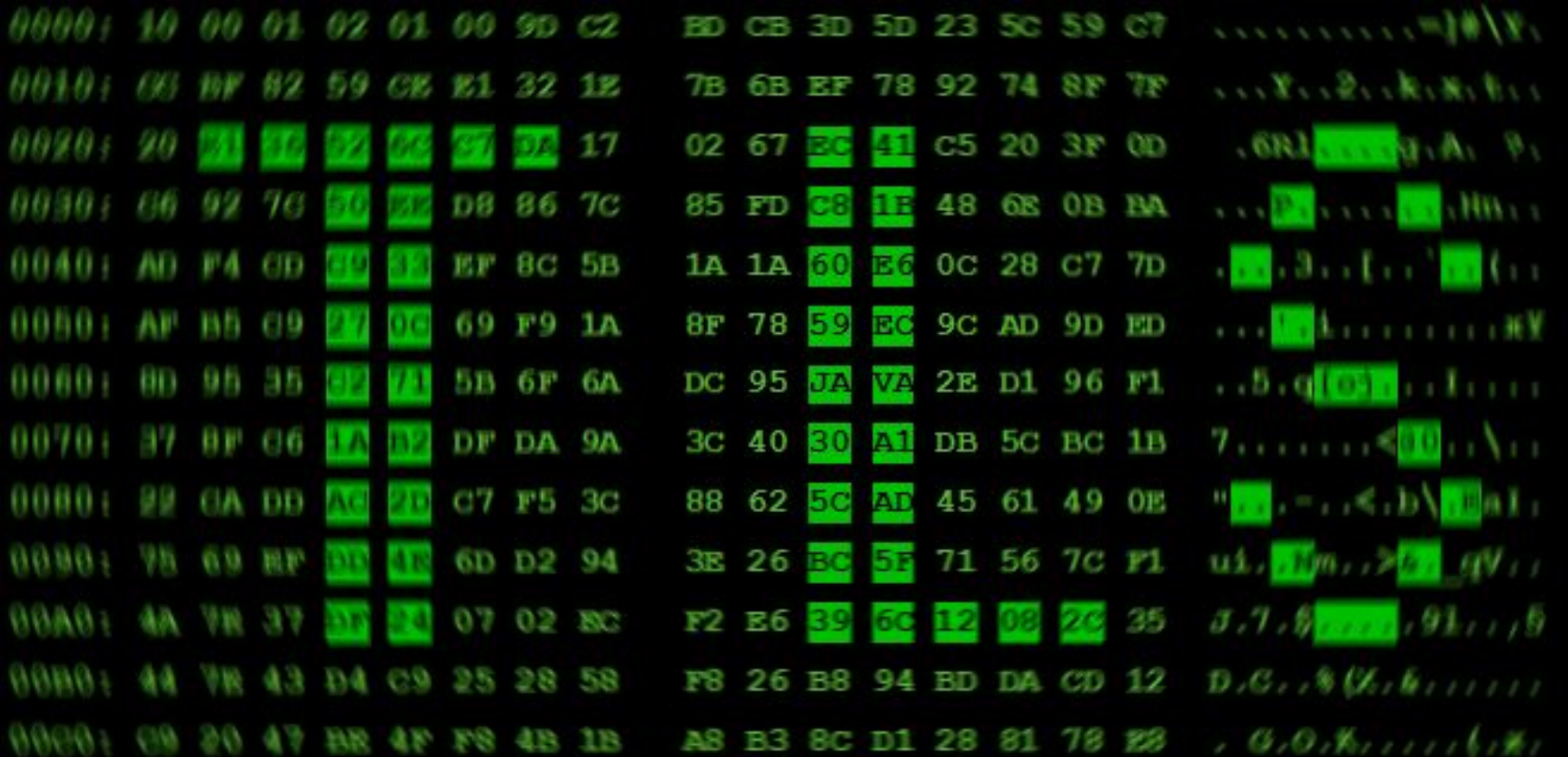
Прикладная криптография для землян

Владимир Плизга́



Вместо плана

1. • Схемы шифрования
2. • Распространение публичных ключей
3. • Протоколы SSL/TLS

Схемы шифрования

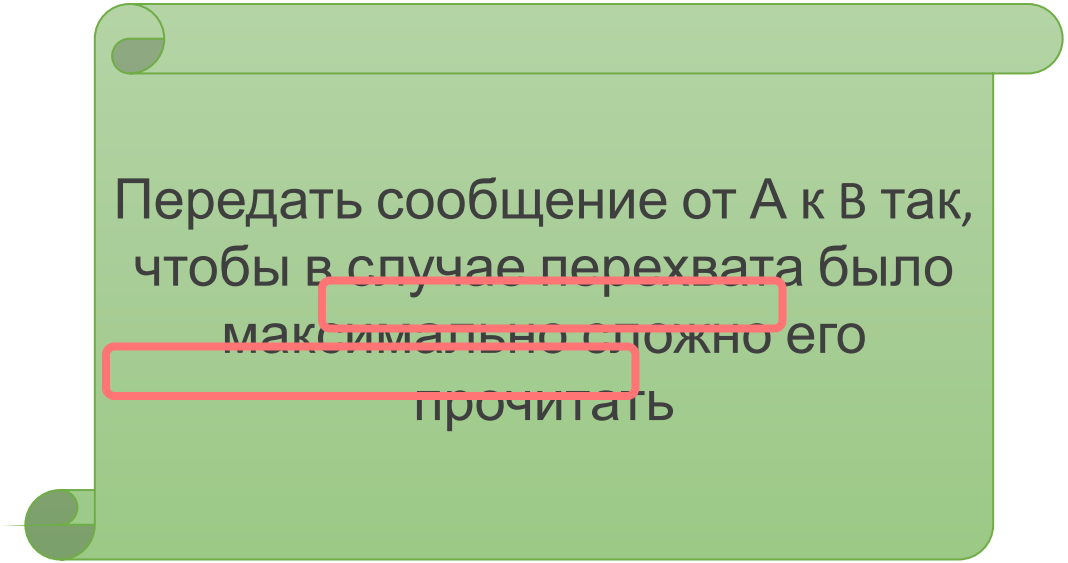
Симметричные и не очень

Сенсация

Криптографические схемы:

- Симметричные
- Асимметричные

Решаемая задача



Симметричная схема

AES, Blowfish, 3DES,
Serpent, RC4, ChaCha

Интернет

code=hJ2&s2

A

B

```
code = encrypt(message, key)
```

```
message = decrypt(code, key)
```

C

=password

Можно передавать по сети
Нельзя передавать по сети

AES: Advanced Encryption Standard

- Разработан в 1998 г. под именем Rijndael
- Победитель конкурса AES от Института стандартов и технологий США (NIST)
- Принят в качестве национального стандарта в 2001 г.
- Симметричный, блочный (128, 192, 256 бит)
- Имеет аппаратную поддержку в большинстве CPU

Асимметричная схема

Основа – не 1, а 2 ключа:

- Однозначно связаны между собой
- Но ~~нельзя~~ сложно вывести один из другого

Где взять такую модель?

- Основная теорема арифметики:

Любое натуральное число однозначно представимо произведением простых чисел

Асимметричная схема

$$21 = ? * 3$$

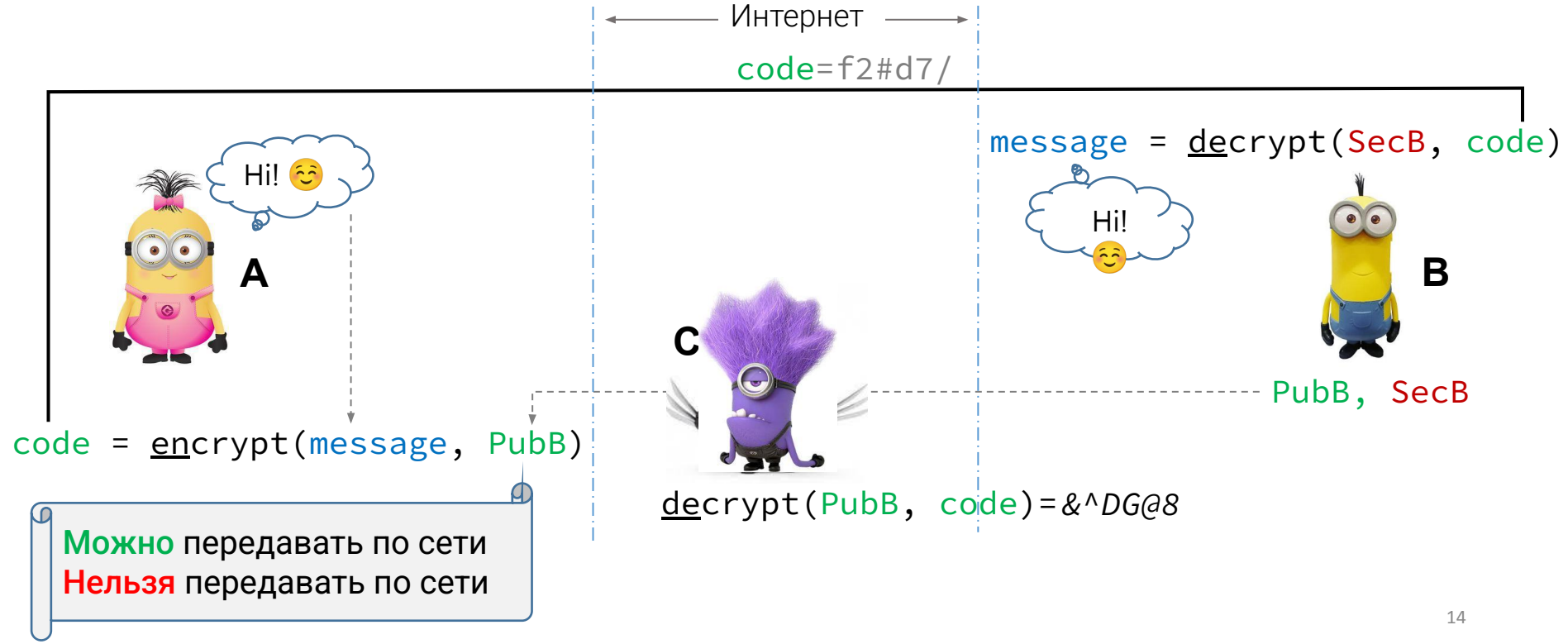
ВЫВЕСТИ

Верно!
Почти…

Пример настоящего ключа

```
0x30820222300D06092A864886F70D0101010500038202 0F003082020A028202
4C112206873D606B8F9B53CAF56AEE35D962DD2EF8539730CF88201EAED45ECE
9DC2702B4EBA3E002C604FEFE29C02139C21B5073ECE9CB6FBF05226641189FD
A4CD8066E3C41FFBA707F3FD3EBC75762474796FD2EBFC80A9BA65EA311DADCF
36CEB5B8D81A19393547274E5A995B1AC92765BD0B115959EAEE05B39993A111
704653F2159DA85C0A7A55A391BD82C58EB826B882A834BB5C6E4B78668A19B5
488301E7FDE39A2D7681A001F7172D55BA35D7C143C33D112D213BCA144B19E3
4C354F4E4D7B4EBF9A5FA682934B383EE093EE43491EAA52A4386B0DB96D9EF4
DB8DCC42A8A0A8905838FFED4DE0251D50CE31F819EB7E8445E53521B0FDA72BEF7DF5C11C4370
BB60CC3029CFAB6AF59255C0D10AF9AAC18D828B2E7B0C7E32248074D90ED0D08A901F937E6054
07751875CE05BF3D0B30F5BB8849B3F050A4C7C604D5DA4E4206139A68CD7E87B52E7F66DC930F
3C8824700A934A5A385B03117C0FCDBFC05B14919473100AE8165C6F80AAABCC7182A0881AEBE6
D9B99C696AD0F618683969543AACABFD7DB8F4AC6CC6254366300452C91BEB9B147D7EAF1167D8
2CA7FE2176048F8A10D605C0809F015B6A49858116F4F33D2AC0972BE0FFEF6D62A505377E3647
0203010001
```

Асимметричная схема: **шифрование**

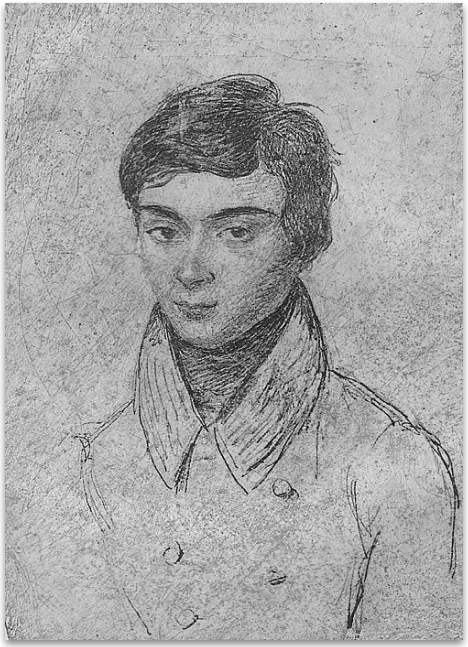


Шифрование: попутное резюме

1. Используются ключи **только получателя**
2. Публичный ключ используется для шифрования а приватный – для дешифрации
3. Для обратной передачи схема **зеркальна**

Реализация асимметричной схемы

- В основе лежат 3 числа
- Одно из них N = P * Q, где P и Q – **случайные** простые числа
- Все операции выполняются по модулю N
  - то есть берутся остатки от деления на N
- Публичный и приватный ключи – это два **других** числа от 0 до N
  - Хранятся вместе с N



Эварист Галуа

Общий порядок действий

- Берем ключи pub & priv, а также число **N**
- Берем секрет – очередной байт шифруемого сообщения
- **Шифрование**:
  - Умножаем секрет сам на себя pub раз по модулю **N**
  - Получаем шифр
- **Дешифрация**
  - Умножаем шифр сам на себя priv раз по модулю **N**
  - Получаем секрет

Проверка боем

Колонка «Математические игры»

Задача: расшифровать сообщение, зашифрованное алгоритмом RSA (425 бит)

Rivest, **S**hamir, **A**dleman

Ronald Rivest

Проверка боем

Шифр

```
9686 9613 7546 2206
1477 1409 2225 4355
8829 0575 9991 1245
7431 9874 6951 2093
0816 2982 2514 5708
3569 3147 6622 8839
8962 8013 3919 9055
1829 9451 5781 5154
```

Публичный ключ

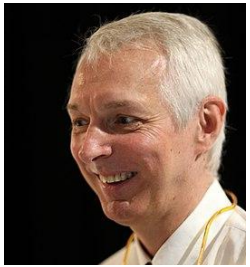
```
n=1143816257578888676692357799761466120102182967212423625625618429357069352457338978305971235639587050589890751475992900268795435 41

e=9007
```

Призовой фонд $100

Проверка боем

Ronald Rivest



Arjen Lenstra

- 1977 год
- на факт

THE MAGIC WORDS ARE SQUEAMISH OSSIFRAGE

Волшебные слова — брезгливый ягнятник

- «Квадр ичное решето»

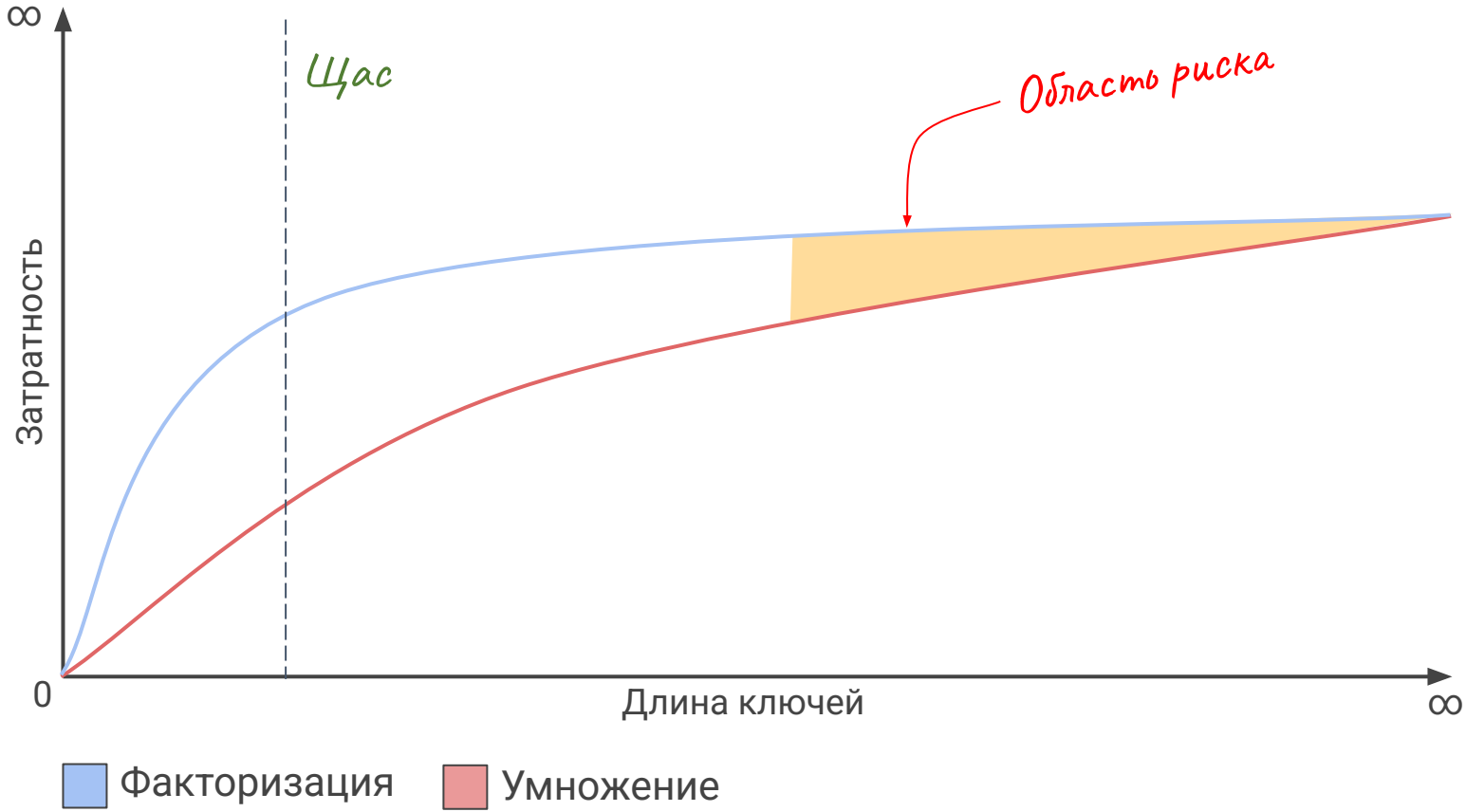
RSA Factor Challenge

RSA number	Decimal digits	Binary digits	Cash prize offered	Factored on	Factored by
RSA100	100	330	US$1,000[8]	April 1, 1991[9]	Arjen K. Lenstra
RSA110	110	364	US$4,429[8]	April 14, 1992[9]	Arjen K. Lenstra and M.S. Manasse
RSA120	120	397	US$5,898[8]	July 9, 1993[10]	T. Denny *et al.*
RSA129[a]	129	426	US$100	April 26, 1994[9]	Arjen K. Lenstra *et al.*
RSA130	130	430	US$14,527[8]	April 10, 1996	Arjen K. Lenstra *et al.*
RSA140	140	463	US$17,226	February 2, 1999	Herman te Riele *et al.*
RSA150	150	496		April 16, 2004	Kazumaro Aoki *et al.*
RSA155	155	512	US$9,383[8]	August 22, 1999	Herman te Riele *et al.*
RSA160	160	530		April 1, 2003	Jens Franke *et al.*, University of Bonn
RSA170[b]	170	563		December 29, 2009	D. Bonenberger and M. Krone[c]
RSA576	174	576	US$10,000	December 3, 2003	Jens Franke *et al.*, University of Bonn

RSA Factor Challenge

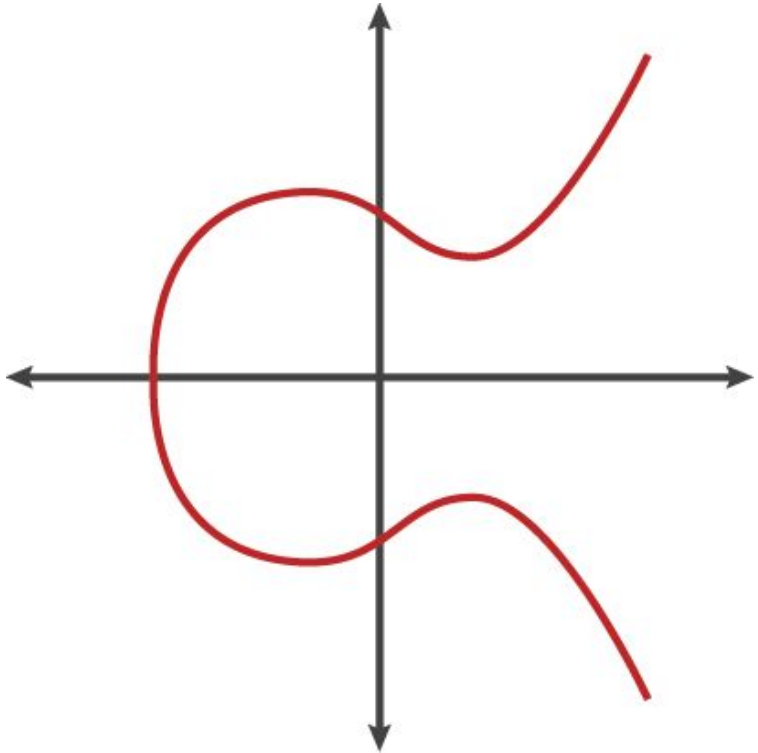
RSA180[b]	180	596		May 8, 2010	S. A. Danilov and I. A. Popovyan, Moscow State University[11]
RSA190[b]	190	629		November 8, 2010	A. Timofeev and I. A. Popovyan
RSA640	193	640	US$20,000	November 2, 2005	Jens Franke *et al.*, University of Bonn
RSA200[b] ?	200	663		May 9, 2005	Jens Franke *et al.*, University of Bonn
RSA210[b]	210	696		September 26, 2013[12]	Ryan Propper
RSA704[b]	212	704	US$30,000	July 2, 2012	Shi Bai, Emmanuel Thomé and Paul Zimmermann
RSA220[b]	220	729		May 13, 2016	S. Bai, P. Gaudry, A. Kruppa, E. Thomé and P. Zimmermann
RSA230[b]	230	762		August 15, 2018	Samuel S. Gross, Noblis, Inc.
RSA232[b]	232	768		February 17, 2020[13]	N. L. Zamarashkin, D. A. Zheltkov and S. A. Matveev.

RSA: Что может пойти не так?



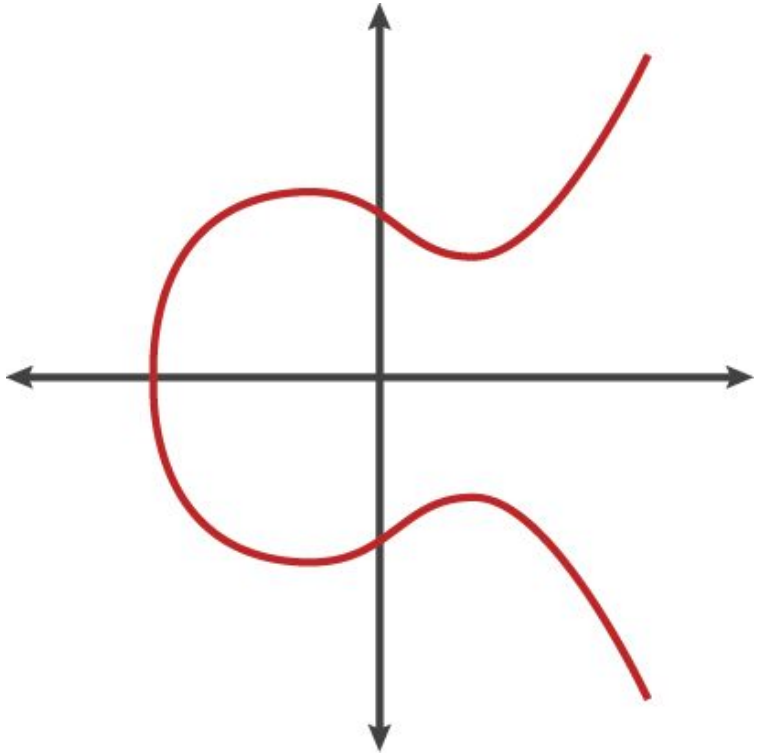
Альтернатива: эллиптические кривые

- Кривые от уравнения вида:
  $y^2 = x^3 + ax + b$
- Могут быть другими, но:
  - Слева степень 2
  - Справа степень 3
- Не связаны с эллипсами
- Стандартизованы



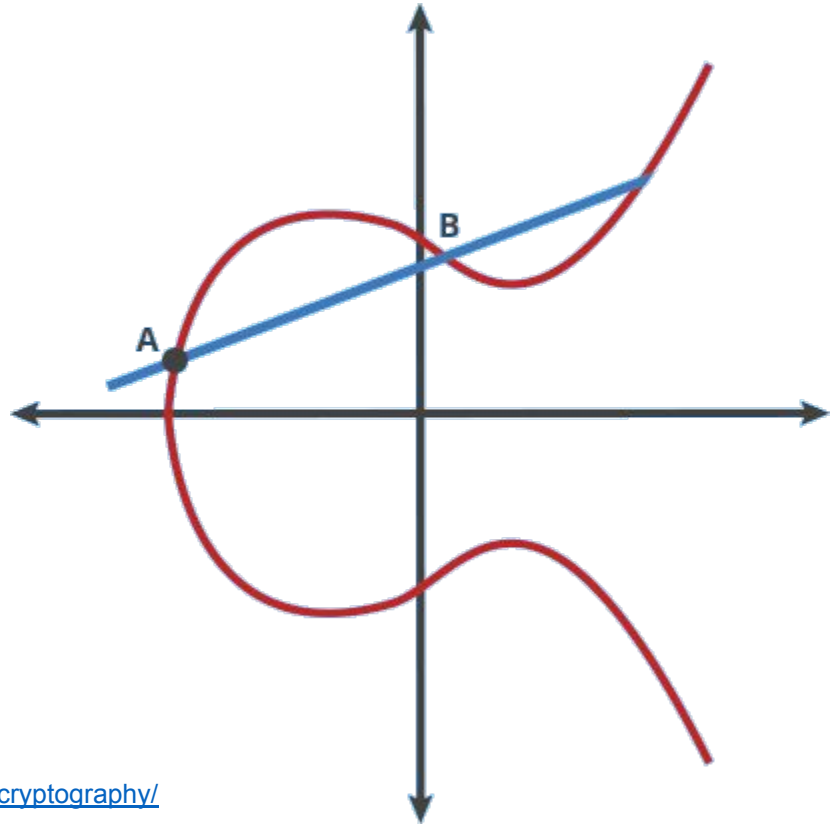
Свойства эллиптических кривых

- Горизонтально симметричны
- Любая не вертикальная прямая пересекает эллиптическую кривую максимум в 3 точках



Сложение точек на кривых

- Проводим прямую из A в B
- Продолжаем до пересечения
- Отражаем через ось X
- Новая точка C = A + B
- Примеры:
  - A + A = B
  - A + B = C
  - A + C = D
  - A + D = E
  - …



https://blog.cloudflare.com/a-relatively-easy-to-understand-primer-on-elliptic-curve-cryptography/

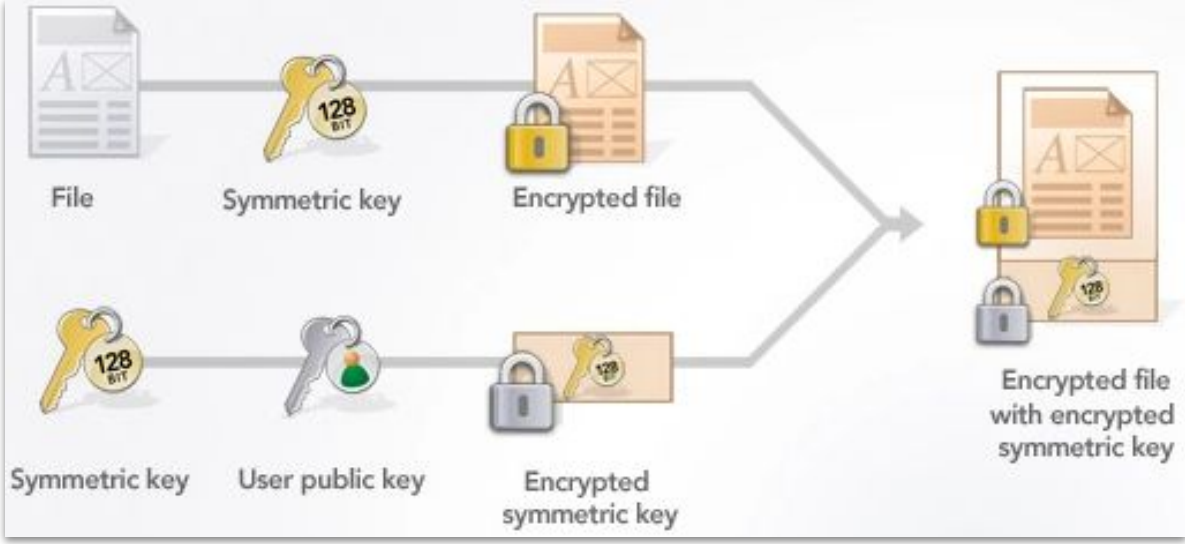
Генерация ключей в ECC

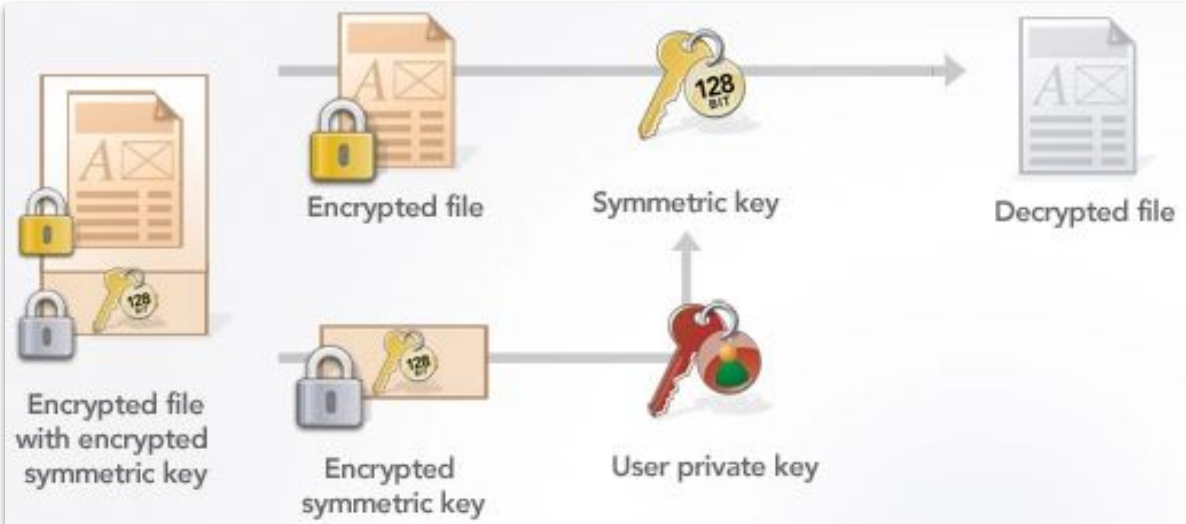
1. Выбрать кривую (параметры a и b)
2. Выбрать точку G на кривой
3. Выбрать случайное число k – секретный ключ
4. Посчитать k * G = Q – публичный ключ

- Зная G и Q, найти k **очень сложно**, т.к. это задача **дискретного логарифмирования** 🤯

Гибридные схемы шифрования и дешифрации

ECC в таких схемах отвечает за выработку симметричного ключа





Источник

И зачем это все?

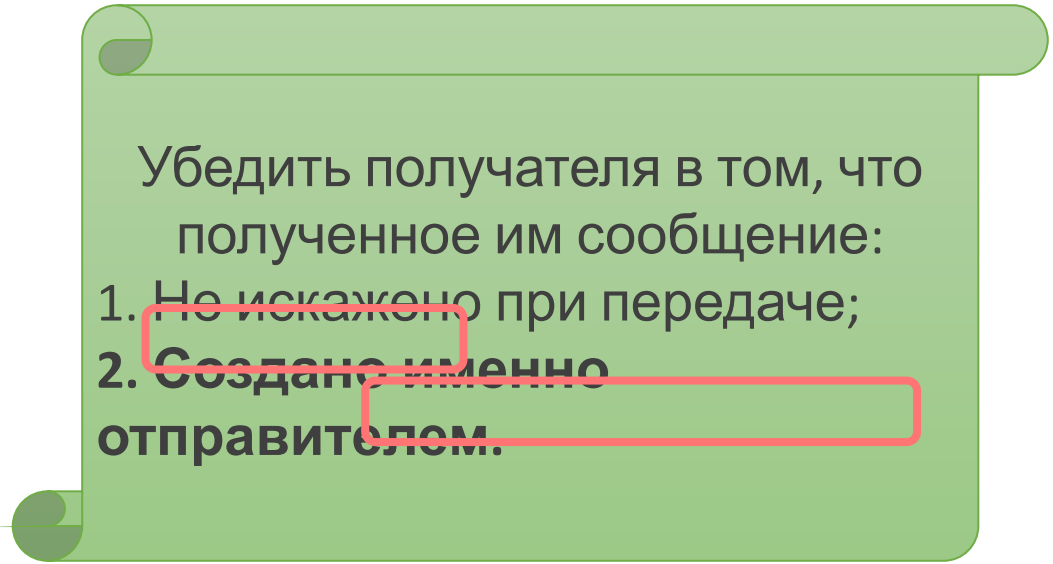
Symmetric Key Size (bits)	RSA Size (bits)	Elliptic Curve Key Size (bits)
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	521

Recommended Key Sizes According to NIST

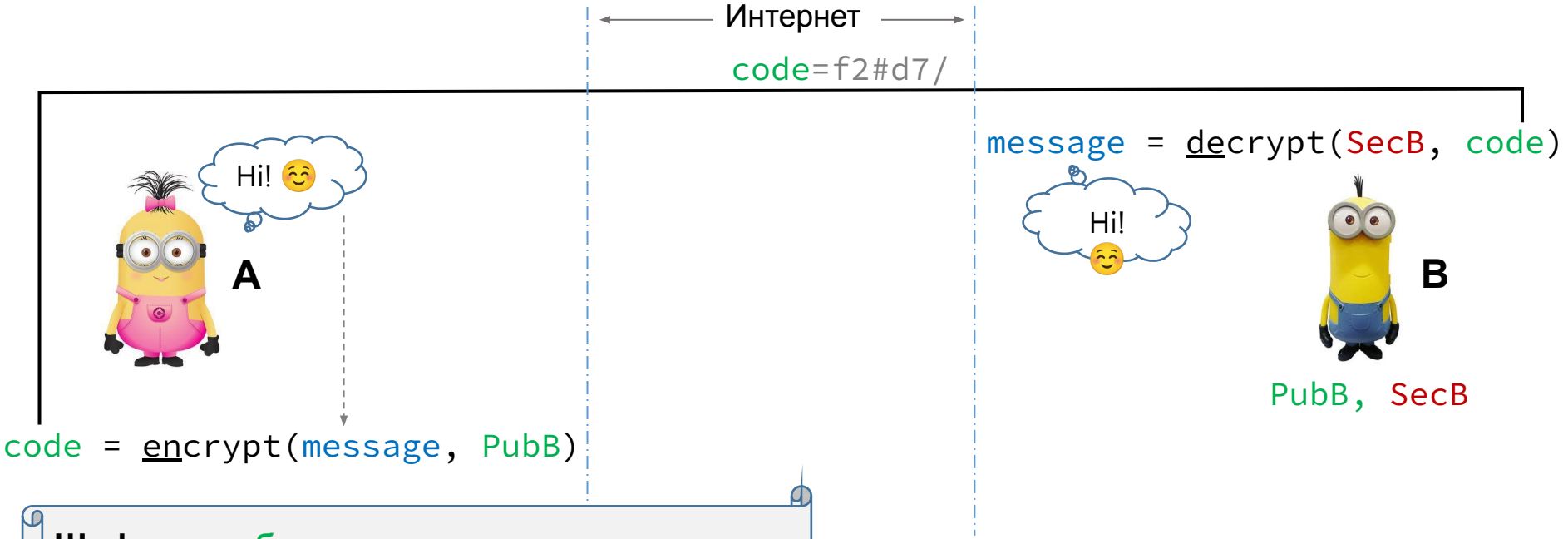
https://www.sectigo.com/resource-library/rsa-vs-dsa-vs-ecc-encryption

Электронная цифровая подпись (ЭЦП)

Решаемая задача

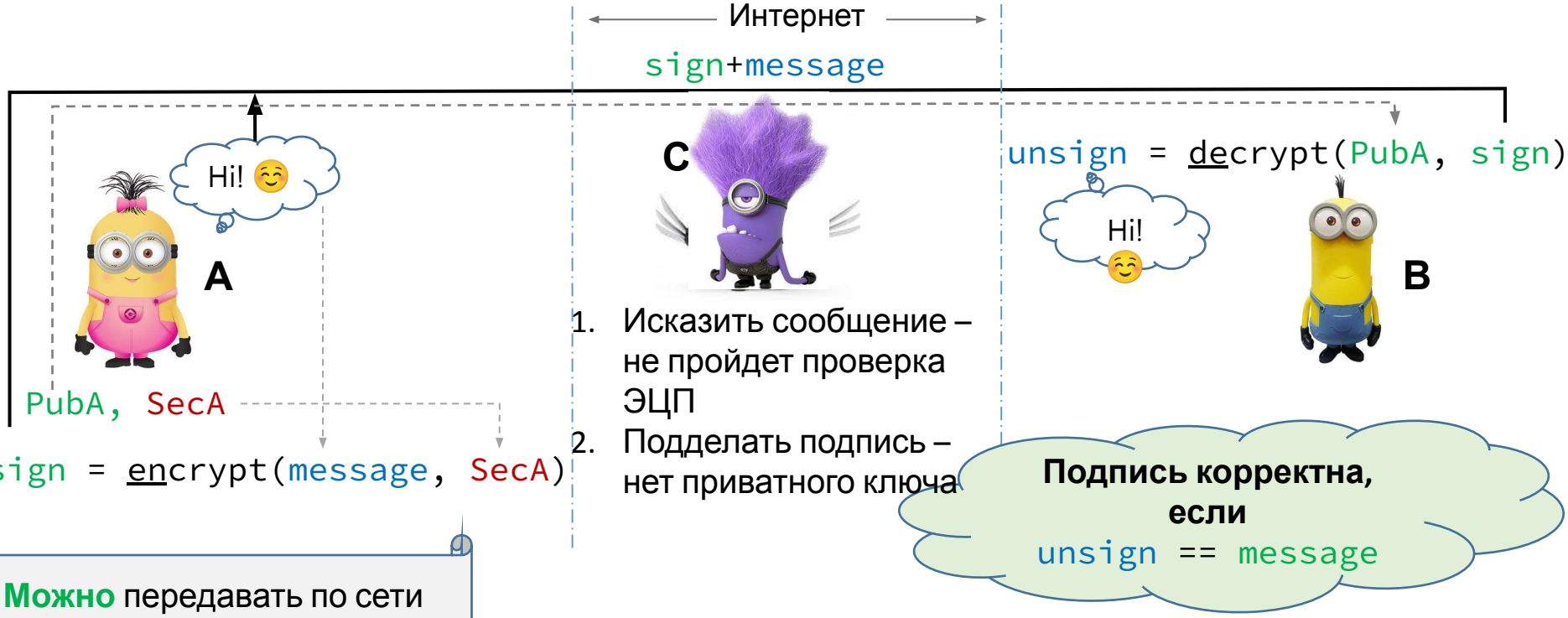


Напоминашка: **шифрование**



Шифрует публичным ключом *получателя*
Дешифрует приватным ключом *получателя*

Асимметричная схема: **подпись**



Асимметричная схема: **подпись**

Шифрует приватным ключом *отправителя*
Дешифрует публичным ключом *отправителя*

Подпись: попутное резюме

В основе – факт **секретности** приватного **ключа** у **отправителя**

Функция схемы – **только индикаторная**, т.е. не позволяет:

- Определить характер сбоя (умышленный/случайный)
- Восстановить искаженное сообщение

Поправка на реальность: хэширование

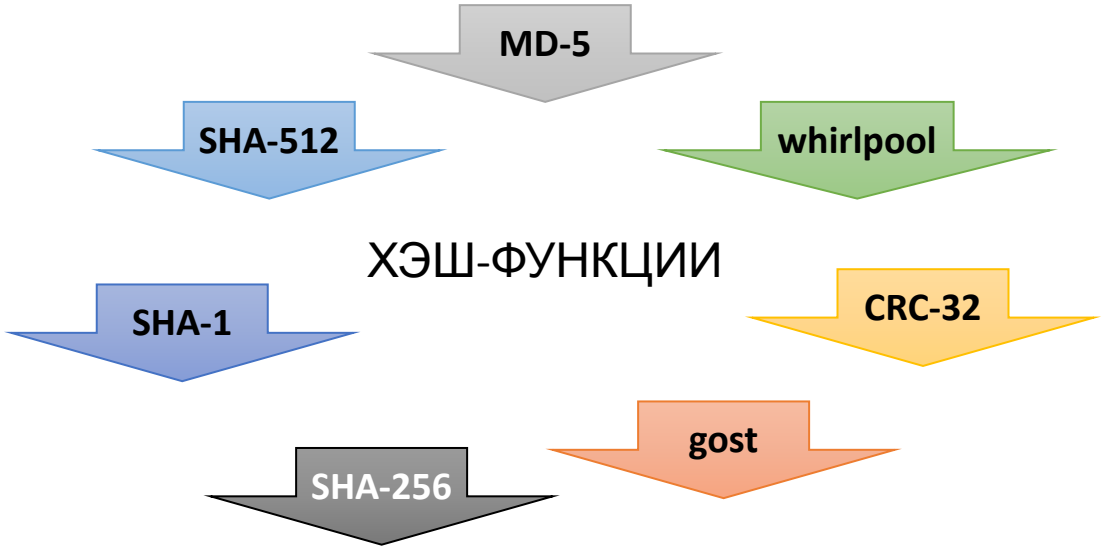
Подпись рассчитывается от **дайджеста** сообщения.

Дайджест = **хэш** – строка **фиксированной** длины,
однозначно описыв[illegible]нное сообщение.

Например:


```
9b99fe3b2c6837f97778dc4d027a49a4
345f32e654d9b84141cfe7d1a46401ae
```

Поправка на реальность: хэширование



Хэширование

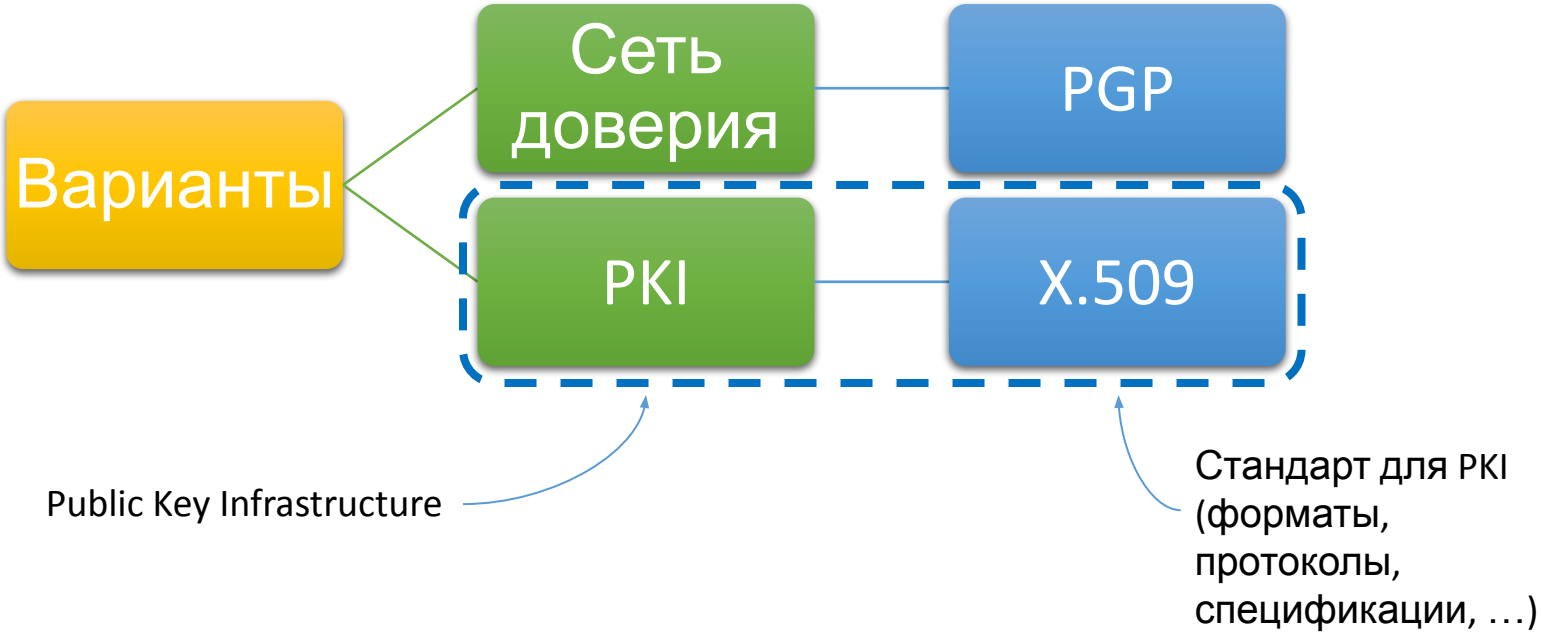
Хэширование – необратимое преобразование сообщения в число фиксированной разрядности, сильно зависящее от исходного сообщения.

Хэширование в ЭЦП делает размер подписи независящим от размера исходного сообщения.

Распространение ключей

Публичные ключи на публике

Как надежно распространять ключи?



Что для этого нужно?

1. Доверенная третья сторона
2. Механизм для «закрепления» доверия

Удостоверяющий центр (УЦ)

Цифровая подпись

Удостоверяющий центр (УЦ)

Удостоверяющий центр (УЦ), **центр сертификации** - сторона
(отдел, организация), чья честность неоспорима,
а **открытый** ключ широко известен. (© Wikipedia)

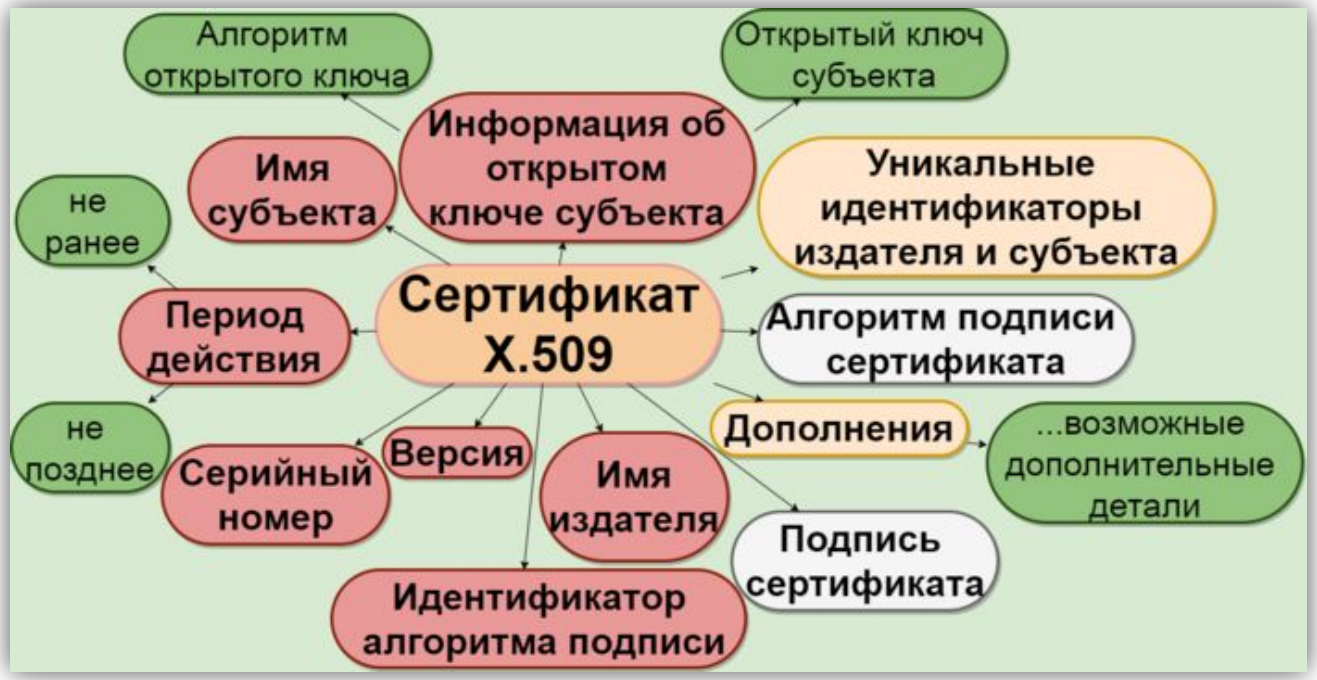
Примеры: Google, Thawte, МинЦифры и любой желающий

Механизм «закрепления» доверия

УЦ заверяет **чужой** **публичный** ключ путем его «подписывания» **своим** **закрытым** ключом.

Формат связки ключа и его подписи:

А как на самом деле



https://commons.wikimedia.org/wiki/File:Структура_сертификата_X.509.png

Защита от компрометации сертификата

Сертификат содержит публичный ключ УЦ

=> ключ может быть **скомпрометирован**

=> ключ нуждается **в защите**

=> ключ нужно **подписать** в другом УЦ

=> а его ключ – в **другом УЦ**, и так далее до…

Корневой УЦ – центр, выдавший сертификат
сам себе и обладающий
абсолютным доверием клиентов

Проверка сертификатов клиентом

Клиенты не обязаны знать сертификаты всех серверов

Если клиент не доверяет серверу,
то он может проверить его УЦ

А если не может доверять его УЦ,
то может проверить УЦ его УЦ

А если не может доверять УЦ его УЦ,
то может проверить УЦ его УЦ его УЦ…

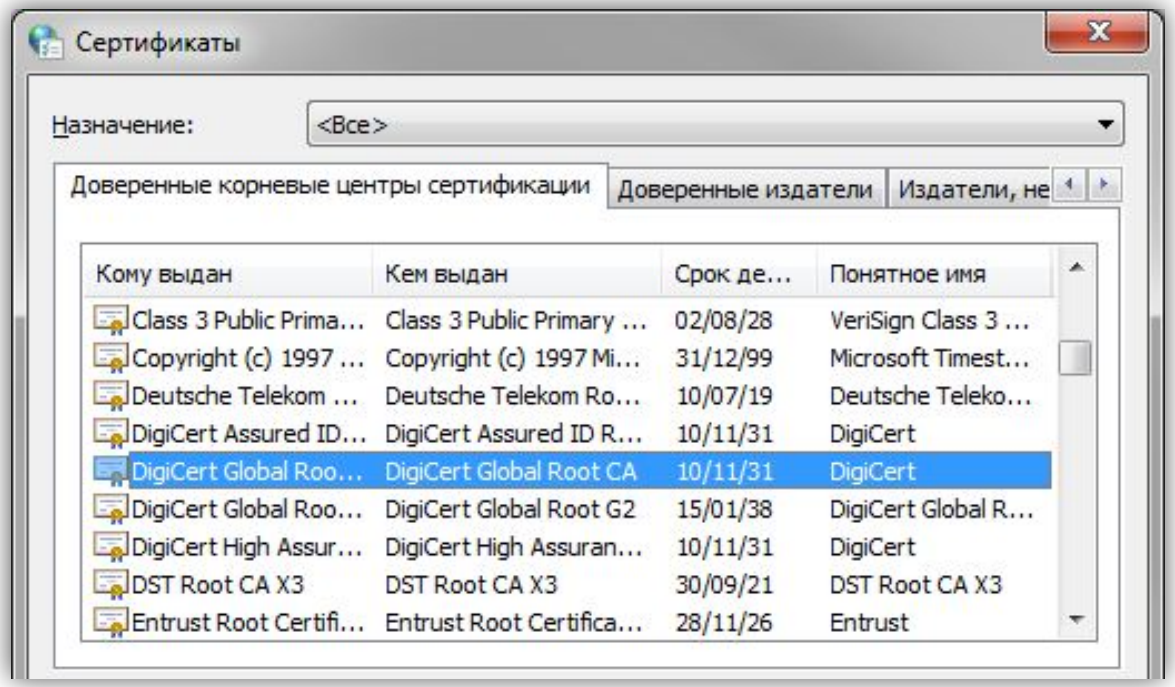
обычно ≈3 шагов

И так пока не найдёт доверенный **корневой УЦ**

Хранилища корневых сертификатов

Chrome, Edge, Opera

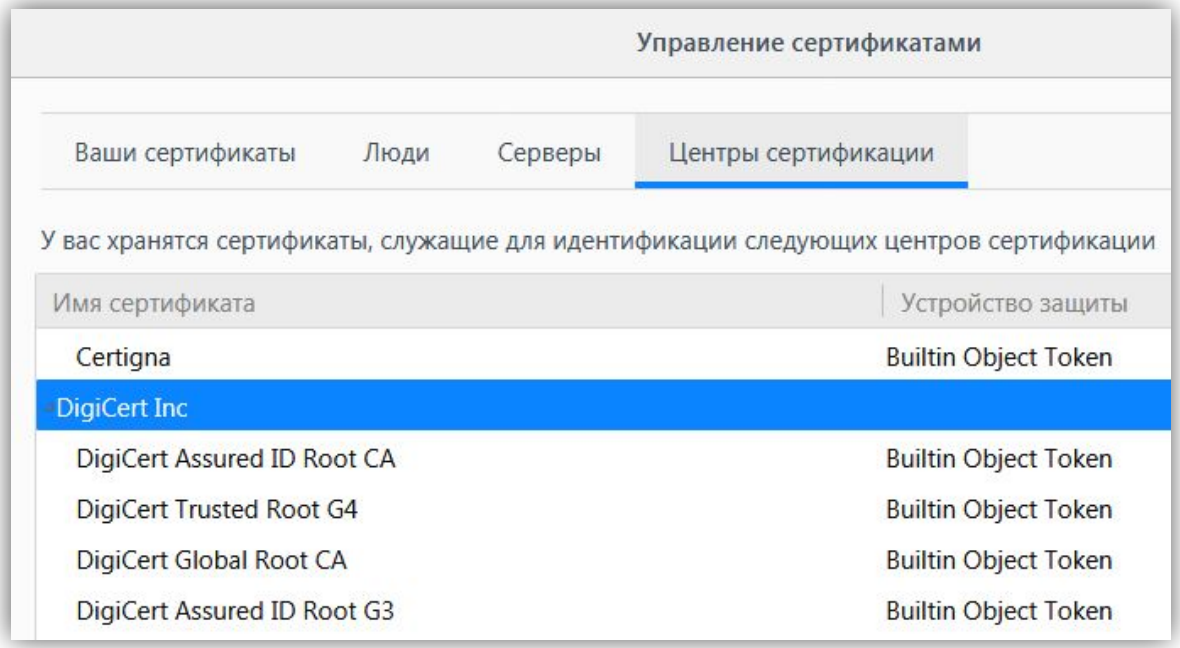
В Windows:
Панель управления
→
Свойства браузера
→
Содержание →
Сертификаты



Хранилища корневых сертификатов

Firefox

Настройки → Приватность и защита → Просмотр сертификатов



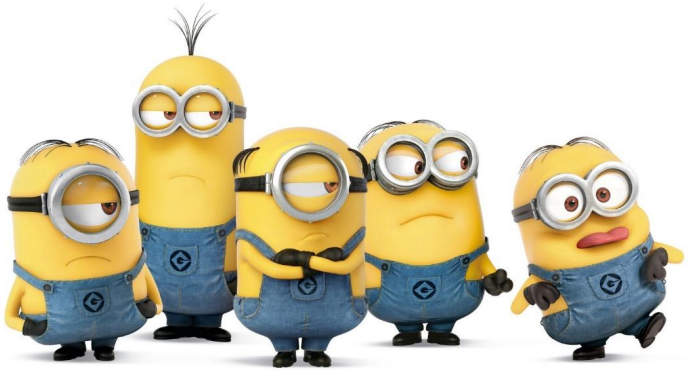
Что мы имеем?

- ✔ Шифрование данных при передаче (RSA, EC)
- ✔ Защита ключей от подделки (ЭЦП)
- ✔ Механизм распространения ключей (PKI + сертификаты X.509)

- ❑ Применимость в масштабах Интернета
  (где с миллионами серверов общаются миллиарды клиентов)
- ❑ Производительность

И как быть?

1. Шифровать прикладные сообщения **симметричным** алгоритмом
2. Генерировать **новый** ключ для каждой **новой** сессии



Установка защищенного канала

1. Клиент обращается к серверу
2. Сервер выдает клиенту свой публичный ключ в составе сертификата
3. Клиент проверяет ЭЦП сертификата
4. Клиент генерирует случайный симметричный сессионный ключ
5. Клиент шифрует его публичным ключом сервера и отправляет серверу
6. Сервер вскрывает сессионный ключ своим приватным ключом
7. Клиент шифрует все сообщения серверу сессионным ключом
8. Сервер шифрует все сообщения клиенту сессионным ключом

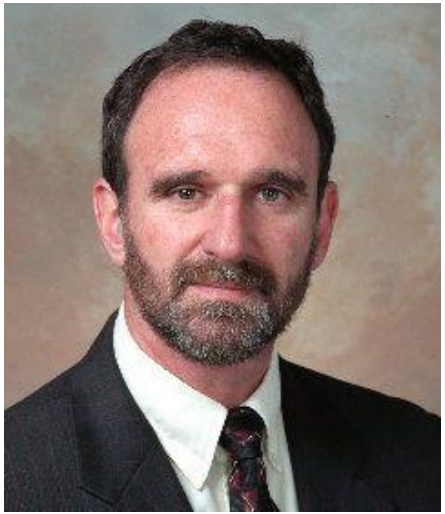

Получение сессионного ключа
Метод обмена
Метод генерации
Алгоритм RSA
Алгоритм Diffie-Hellman

Алгоритм Диффи-Хеллмана (DH)

У. Диффи

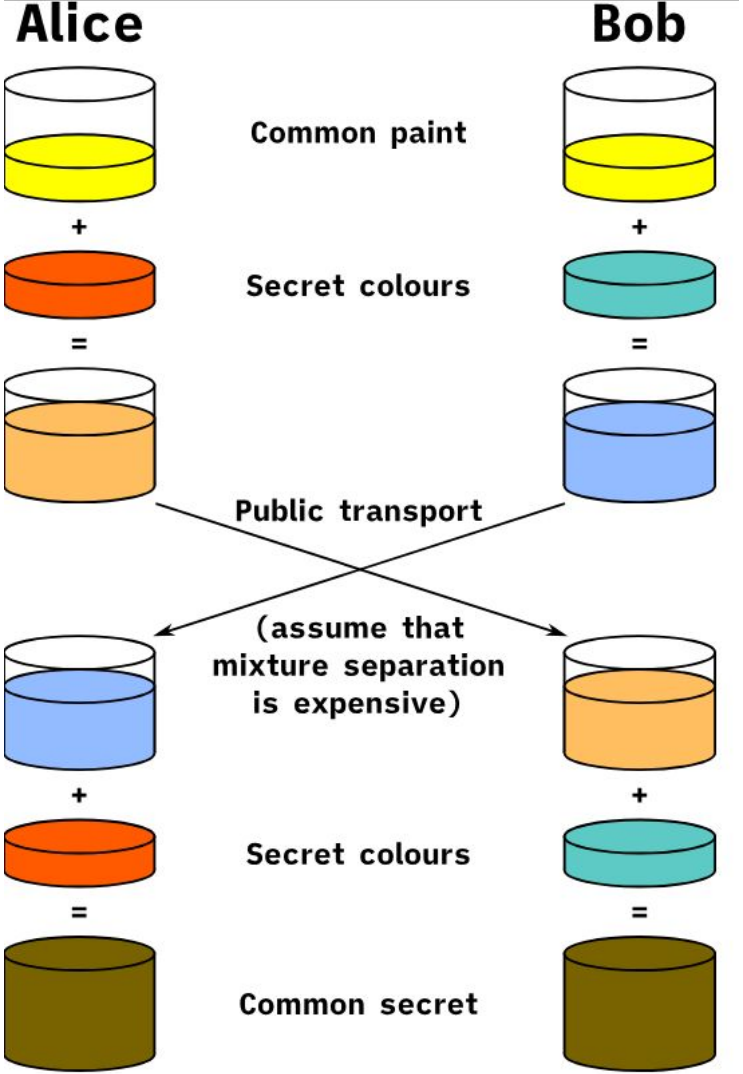
Позволяет двум сторонам выработать общий ключ, никогда не передавая его друг другу

=> Приватный ключ сервера перестаёт быть «Ахиллесовой пятой»



М. Хеллман

DH на пальцах



https://upload.wikimedia.org/wikipedia/commons/4/46/Diffie-Hellman_Key_Exchange.svg

SSL/TLS

И другие ругательства

Протоколы SSL и TLS: в чем разница?

SSL (англ. *Secure Sockets Layer* — уровень защищённых сокетов) — криптографический протокол, который подразумевает более безопасную связь. Он использует асимметричную криптографию для аутентификации ключей обмена, симметричное шифрование для сохранения конфиденциальности, коды аутентификации сообщений для целостности сообщений. Протокол широко использовался для обмена мгновенными сообщениями и передачи голоса через IP (англ. *Voice over IP* — VoIP) в таких приложениях, как электронная почта, интернет-факс и др. В 2014 году правительство США сообщило об уязвимости в текущей версии протокола[1]. SSL должен быть исключён из работы в пользу TLS (см. CVE-2014-3566).

TLS (англ. *transport layer security* — Протокол защиты транспортного уровня[1]), как и его предшественник SSL (англ. *secure sockets layer* — слой защищённых сокетов), — криптографические протоколы, обеспечивающие защищённую передачу данных между узлами в сети Интернет[2]. TLS и SSL используют асимметричное шифрование для аутентификации, симметричное шифрование для конфиденциальности и коды аутентичности сообщений для сохранения целостности сообщений. TLS-протокол основан на спецификации протокола SSL версии 3.0, разработанной компанией Netscape Communications[3]. Сейчас развитием стандарта TLS занимается IETF. Обновления протокола были в RFC 5246 (август 2008), RFC 6176 (март 2011) и RFC 8446 (август 2018).

Короче, это одно и то же.

История версий SSL/TLS

SSL and TLS protocols		
Protocol	**Published**	**Status**
SSL 1.0	Unpublished	Unpublished
SSL 2.0	1995	Deprecated in 2011[19]
SSL 3.0	1996	Deprecated in 2015[20]
TLS 1.0	1999	Deprecated in 2021[21][22][23][24]
TLS 1.1	2006	Deprecated in 2021[21][22][23][24]
TLS 1.2	2008	In use since 2008[25][26]
TLS 1.3	2018	In use since 2018[26][27]
Unsupported	Supported	**Latest version**

https://en.wikipedia.org/wiki/Transport_Layer_Security

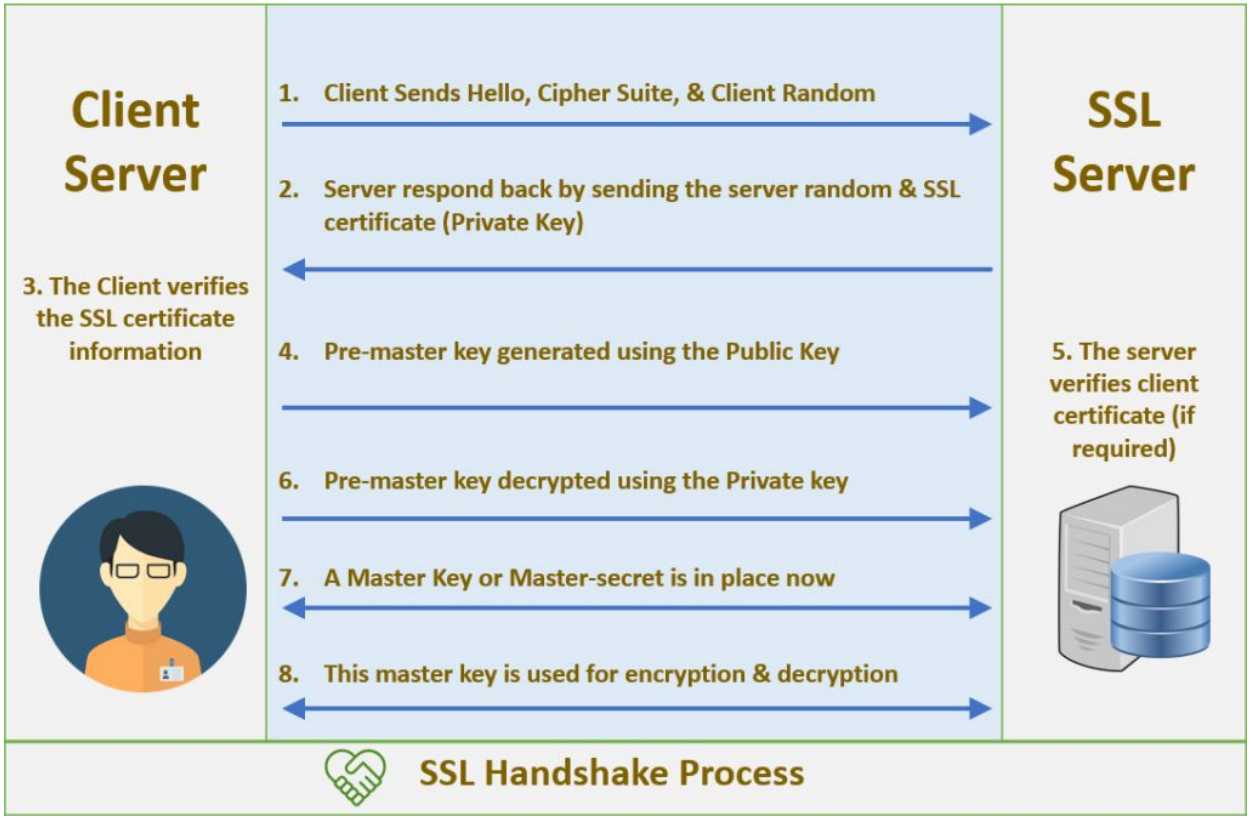
Протокол SSL/TLS

- Описанные ранее схемы – основа SSL/TLS

Соединение по SSL/TLS начинается с рукопожатия (**handshake**):

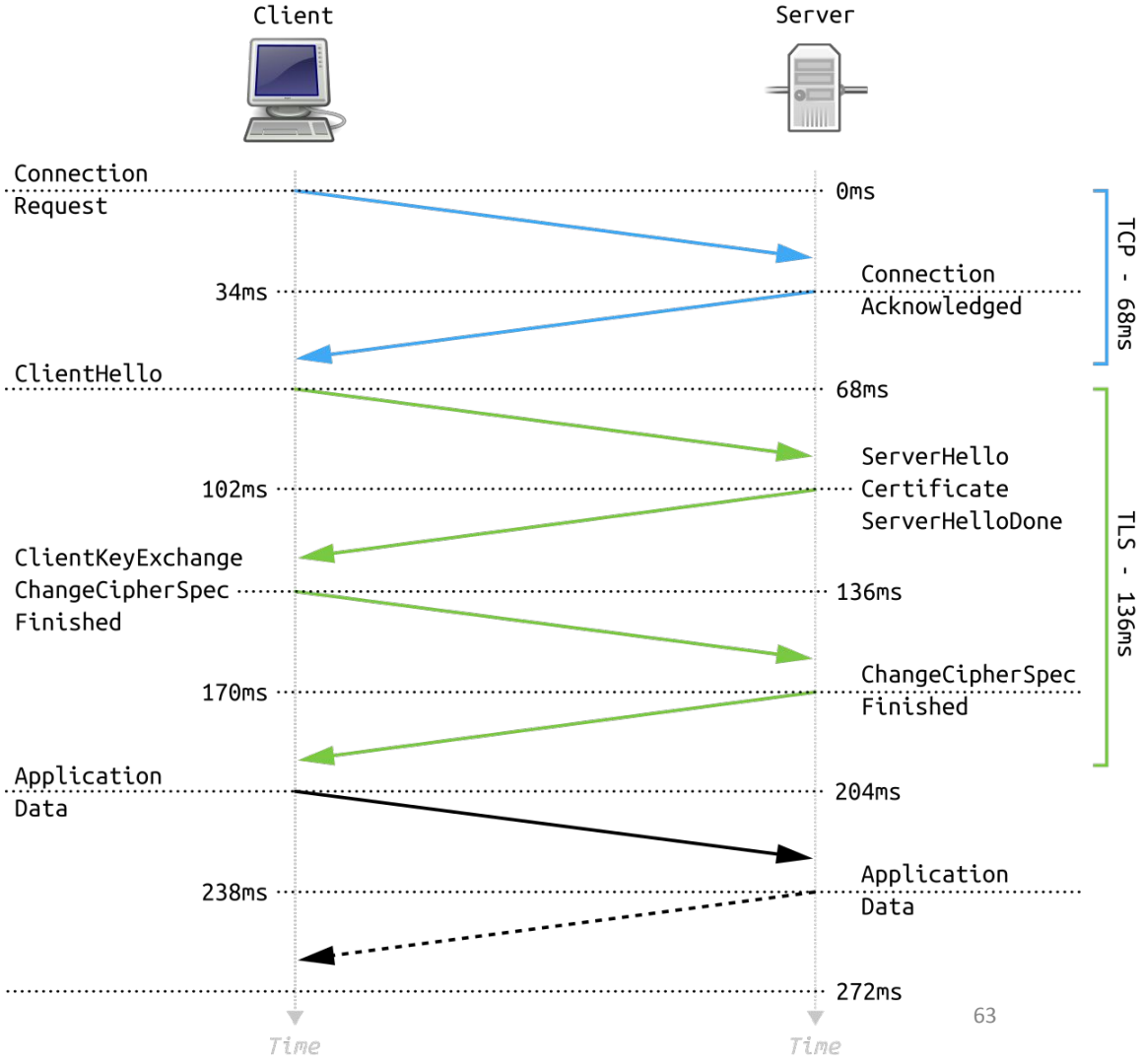
- Обмен сертификатами, их проверка
- Выбор шифронабора (AES, EC, …)
- Выработка сессионного ключа
- Тестирование канала

Обмен сообщениями в TLS 1.2 (**RSA**)



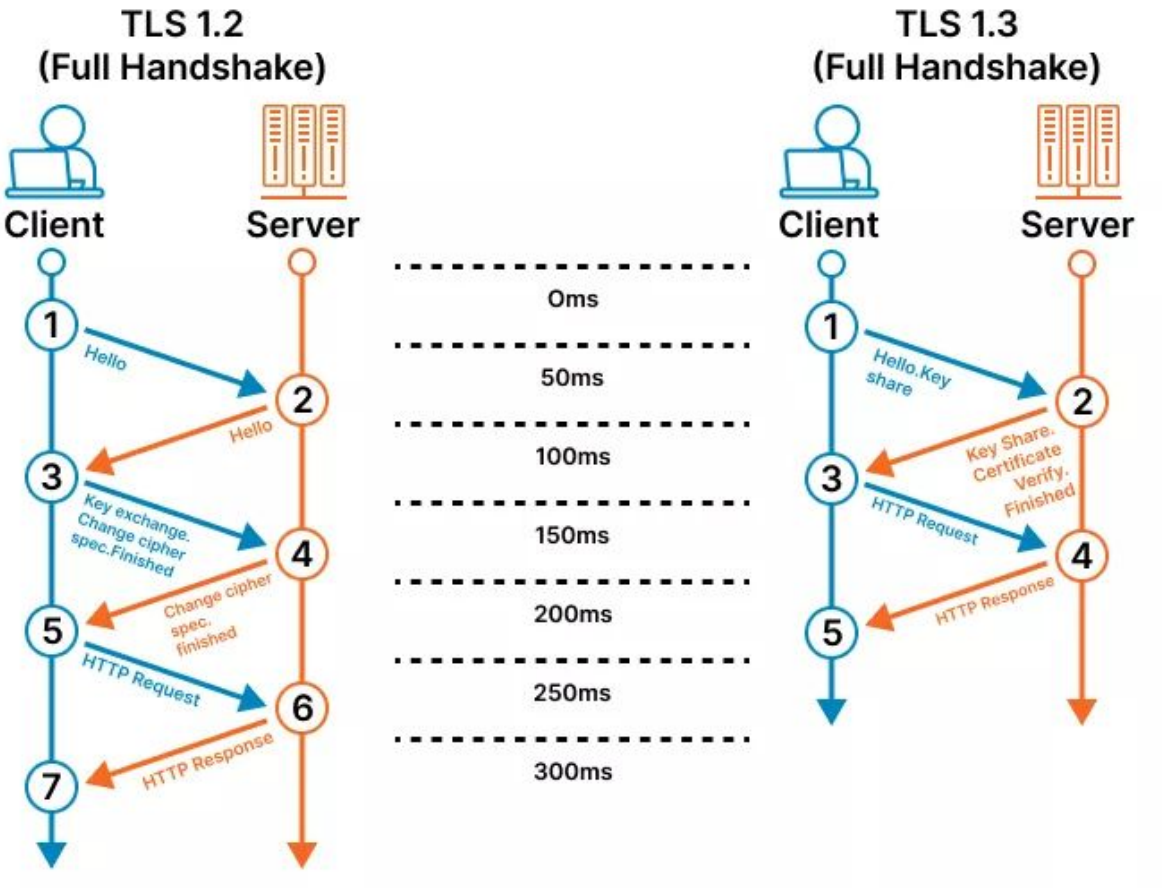
Источник

Обмен сообщениями в TLS 1.2 (**DH**)



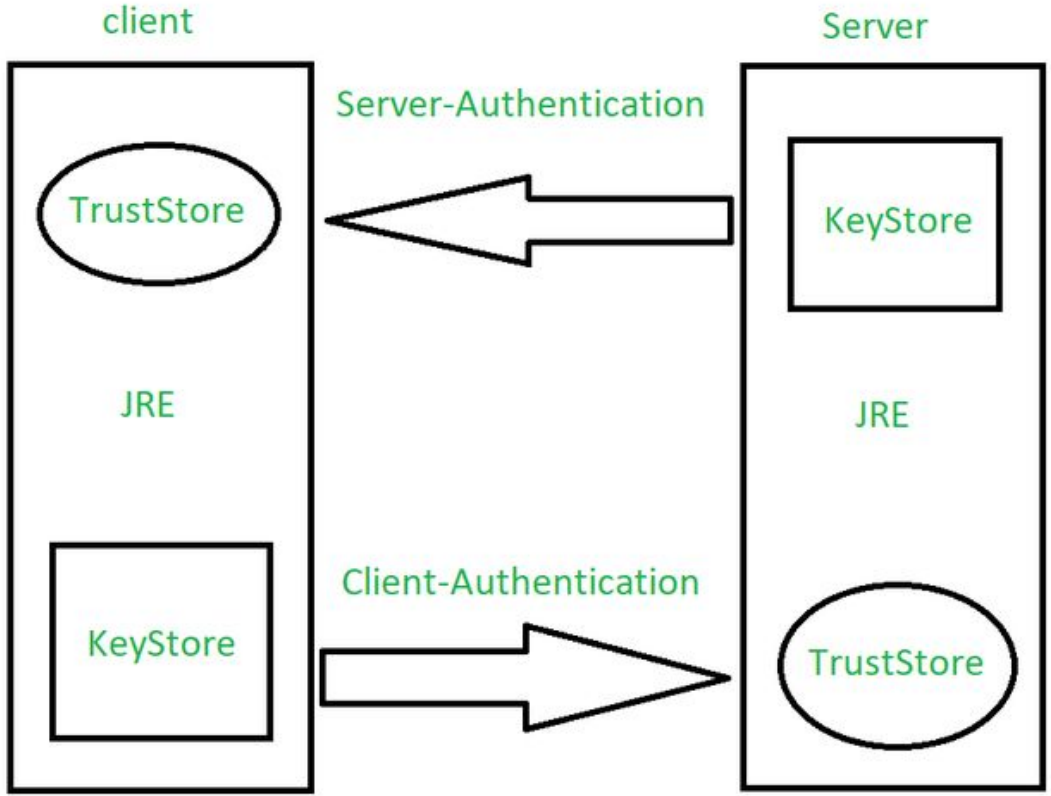
[Источник]

Обмен сообщениями в TLS 1.3



[Источник]

Используемые хранилища ключей и сертификатов



Резюме

Протокол TLS

Асимметричное шифрование
RSA, EC

Только для рукопожатия

Симметричное шифрование
AES, 3DES, Blowfish

Для основного обмена

Хэширование (дайджест)
SHA3, MD5, CRC32

Для фиксации размера подписи

Цифровая подпись (ЭЦП)
RSA+SHA

Для проверки сертификатов

PKI (Public Key Infrastructure)

Удостоверяющий центр (УЦ)
Thawte, Let's Encrypt

Для распространения сертификатов

Цифровой сертификат
X.509, [PGP]

Для защиты ключей от подделки

используется

Прикладная криптография для землян

Владимир Плизгá
@toparvion