

Основы и принципы криптографии

JS
COURSE
ORT DNIPRO

ORT**DNIPRO**.ORG/**JS**

1. Двоичная система счисления 0/1

BIN: 0; DEC: 0; HEX: 0

BIN: 1; DEC: 1; HEX: 1

BIN: 10; DEC: 2; HEX: 2

BIN: 11; DEC: 3; HEX: 3

BIN: 100; DEC: 4; HEX: 4

BIN: 101; DEC: 5; HEX: 5

BIN: 110; DEC: 6; HEX: 6

BIN: 111; DEC: 7; HEX: 7

BIN: 1000; DEC: 8; HEX: 8

BIN: 1001; DEC: 9; HEX: 9

BIN: 1010; DEC: 10; HEX: a

BIN: 1011; DEC: 11; HEX: b

BIN: 1100; DEC: 12; HEX: c

BIN: 1101; DEC: 13; HEX: d

BIN: 1110; DEC: 14; HEX: e

BIN: 1111; DEC: 15; HEX: f

BIN: 10000; DEC: 16; HEX: 10

BIN: 10001; DEC: 17; HEX: 11

BIN: 10010; DEC: 18; HEX: 12

BIN: 10011; DEC: 19; HEX: 13

BIN: 10100; DEC: 20; HEX: 14

BIN: 10101; DEC: 21; HEX: 15

BIN: 10110; DEC: 22; HEX: 16

Двоичная система счисления

```
1  
2   for(let i = 0; i <= 100; i++){  
3  
4       console.log(`  
5           BIN: ${i.toString(2)};  
6           DEC: ${i.toString(10)};  
7           HEX: ${i.toString(16)};  
8       `);  
9  
10  }
```

Разрядность системы счисления зависит от количества цифр используемых для формирования чисел, в остальном отличий от привычной нам десятичной системы нет. В **JavaScript** метод **.toString(N)** позволяет вывести число в нужной системе счисления (разрядной которой задаётся параметром **N**).

2. Битовые операции

Битовые операции

```
1
2 let a = 67;
3
4 let b = 23;
5
6 let x = a & b; //Bitwise AND
7 let y = a | b; //Bitwise OR
8 let z = a ^ b; //Bitwise XOR
9
```

		AND	OR	XOR
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Битовые операторы выполняют операции над битами числа

3. Симметричная шифрование

Симметричное шифрование

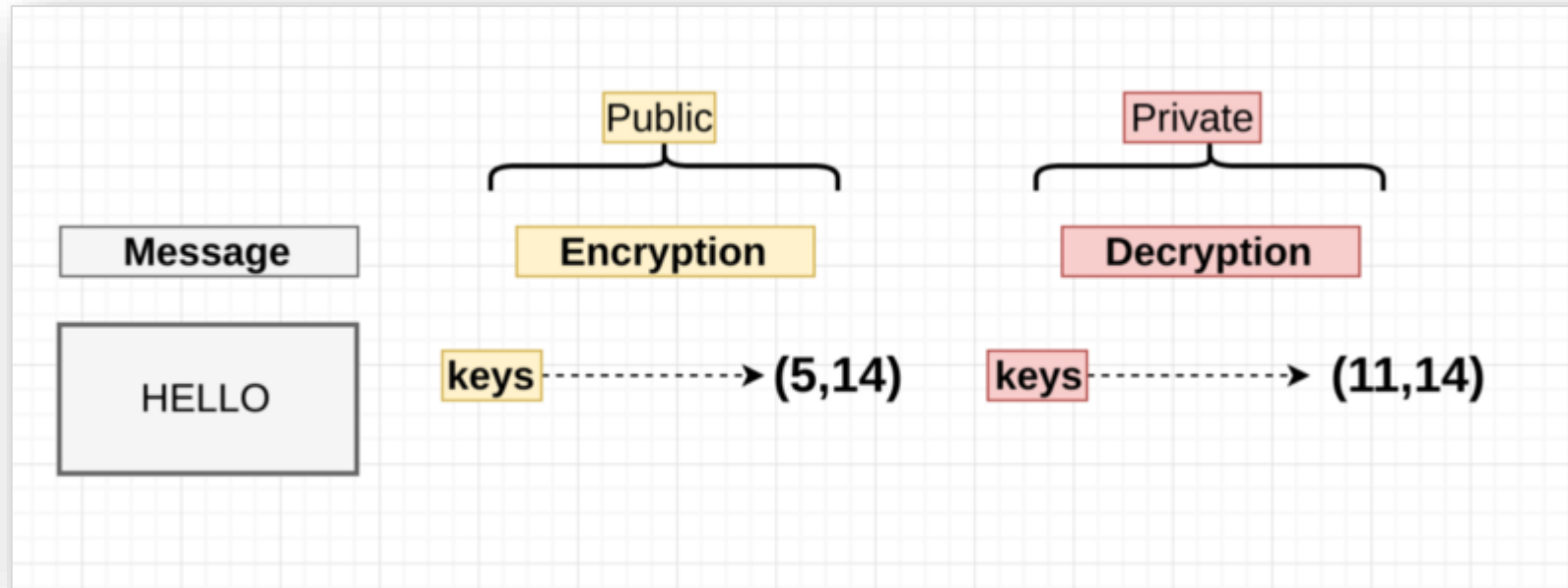
```
1
2 let myData = 'Forza Ferrari!';
3
4 const KEY = 42;
5
6 let chars = [...myData]
7   .map(i => i.charCodeAt(0));
8
9 console.log('Original Chars: ', chars);
10
11 /* Encrypt */
12
13 let encryptedChars = chars.map(i => i ^ KEY);
14
15 console.log('Encrypted Chars: ', encryptedChars);
16
17 console.log('Encrypted Text: ', String
18   .fromCharCode(...encryptedChars));
19
20 /* Decrypt */
21
22 let decryptedChars = encryptedChars.map(i => i ^ KEY);
23
24 console.log('Decrypted Chars: ', decryptedChars);
25
26 console.log('Decrypted Text: ', String
27   .fromCharCode(...decryptedChars));
28
29
```

Симметричное шифрование – использует один и тот же ключ для шифровки и расшифровки данных.

https://ru.wikipedia.org/wiki/Симметричные_криптосистемы

4. Ассиметричное шифрование

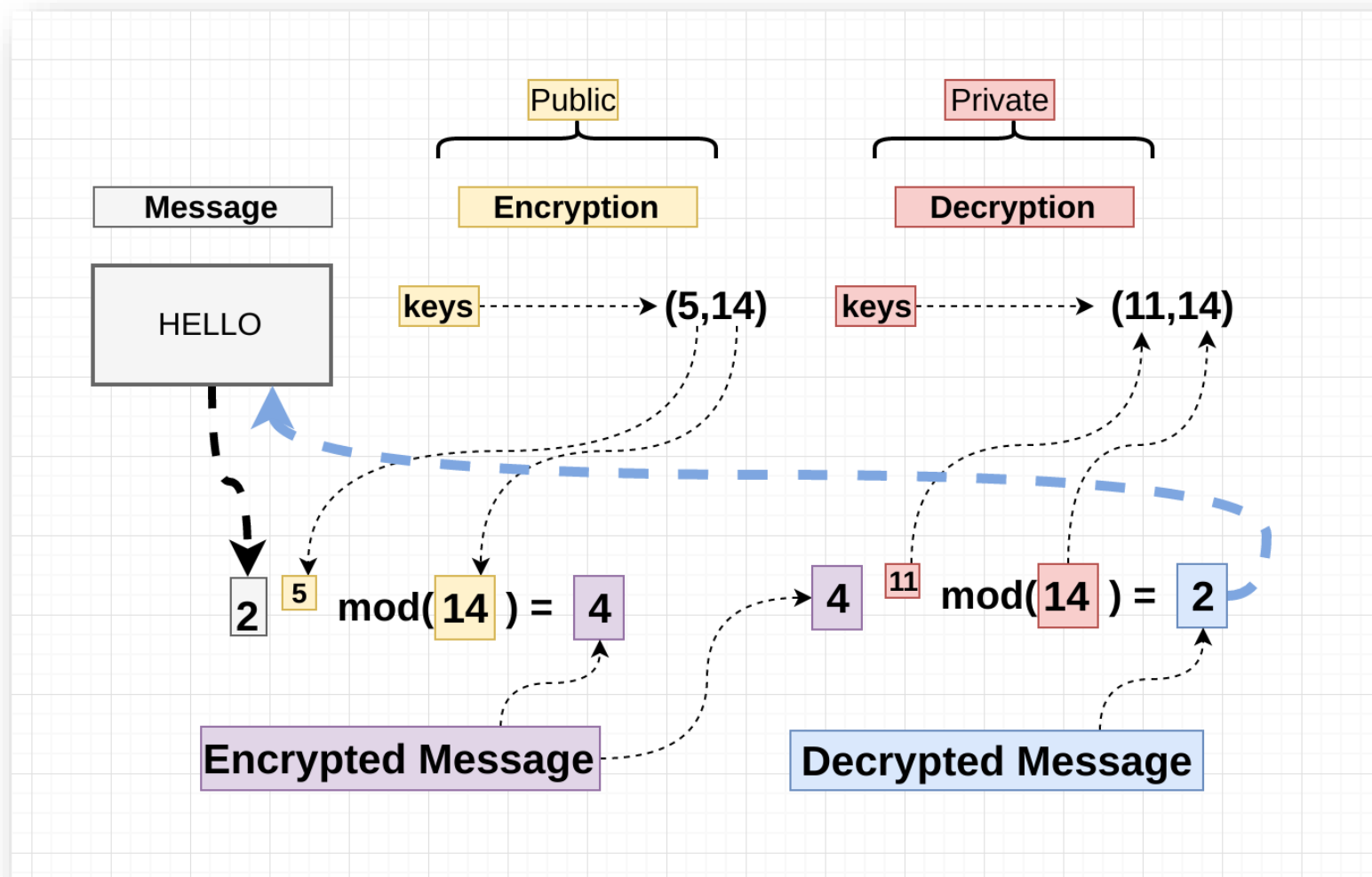
Ассиметричное шифрование (алгоритм RSA)



Ассиметричное шифрование – использует разные ключи (открытый и закрытый) для шифрования и расшифровки данных.

https://ru.wikipedia.org/wiki/Криптосистема_с_открытым_ключом

Ассиметричное шифрование (алгоритм RSA)



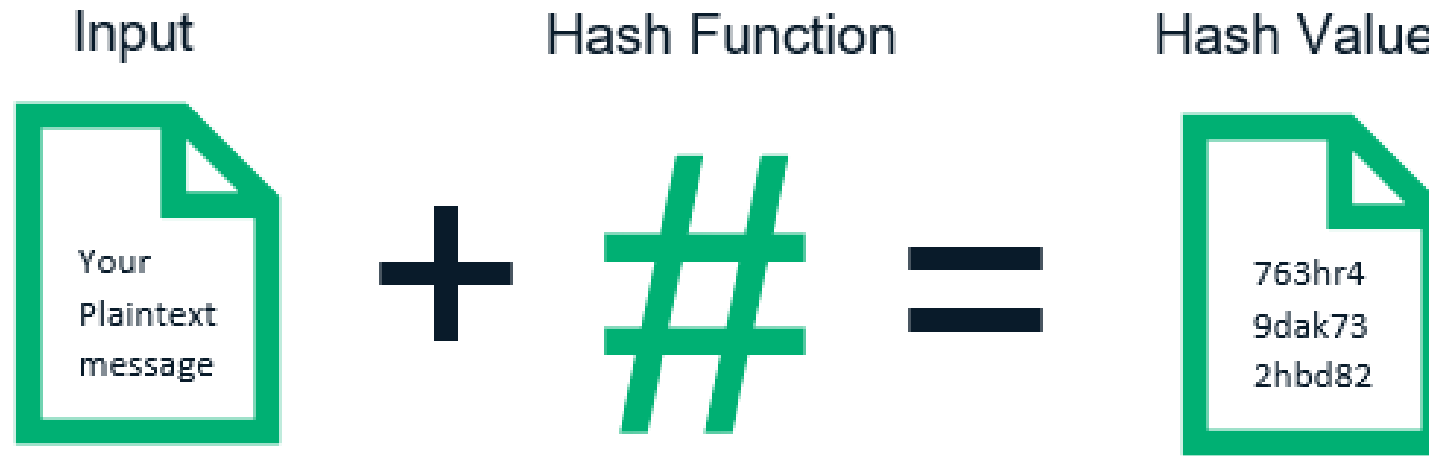
Ассиметричное шифрование – использует разные ключи (открытый и закрытый) для шифрования и расшифровки данных.

Алгоритм RSA / Генерация ключей

1. Выбираем два простых числа P и Q ;
2. Находим $N = P * Q$;
3. Находим $F = (Q - 1) * (P - 1)$;
4. Подбираем число E , которое должно быть простым, быть меньше F и их максимальный общий делитель был 1;
5. Выбираем число D удовлетворяющее $D * E \% F == 1$;
6. Теперь у нас есть пара ключей (E, N) и (D, N) ;

5. Хеширование

Хеширование / Хеш-функция



Преобразование входного набора данных любого (как правило большого) размера в данные фиксированного размера. Существует множество алгоритмов хеширования.

<https://ru.wikipedia.org/wiki/Хеширование>

Хеширование

Hello world!! => (SHA256) =>

4354dfda70c8f0d3991b9de3d56dcb6e9f2fc6c0316d235b63afeb388471ada4

Hello world!! => (SHA256) =>

bbca77170621e018f9b8d17c850d2c7efe3cf9998cf741edf8e7dffbaeeb160e

Хеширование по алгоритму SHA256 (калькулятор):

<http://www.xorbin.com/tools/sha256-hash-calculator>

Преобразование входного набора данных любого (как правило большого) размера в данные фиксированного размера. Существует множество алгоритмов хеширования.

<https://ru.wikipedia.org/wiki/Хеширование>

Хеширование / SHA256

```
1
2   import SHA256 from './sha256.js';
3
4   const hash = SHA256.hash('abc');
5
6   console.log(hash);
7
```

в **NPM**'е есть библиотека **sha256-es** функцию выполняющую расчёт хеша по алгоритму **SHA256** и с поддержкой **ECMAScript Модулей**.

<https://www.npmjs.com/package/sha256-es>

6. Цифровая подпись

Цифровая подпись

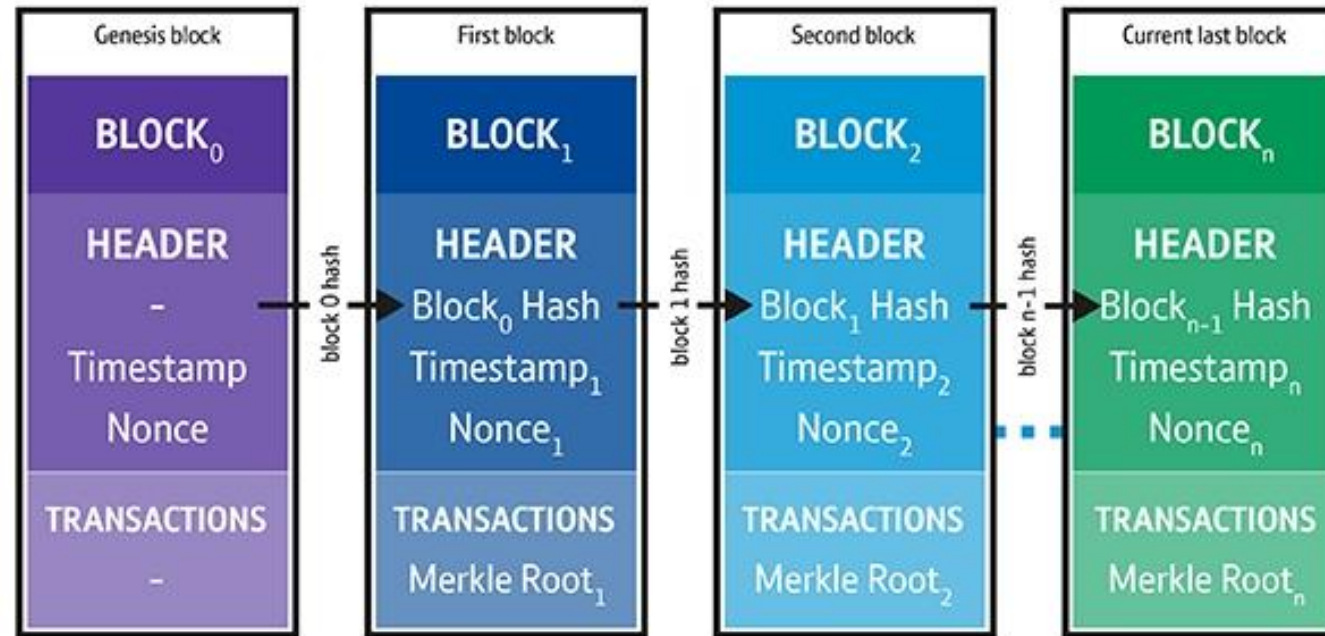


Цифровая подпись – технология на базе хеширования и асимметричного шифрования задача которой подтвердить достоверность передаваемых данных от отправителя к получателю.

https://ru.wikipedia.org/wiki/Электронная_подпись

7. Blockchain

Blockchain



Идея **Blockchain** (цепочки блоков) - в формировании последовательности блоков, в которой каждый блок помимо полезных данных содержит еще и **хеш** предыдущего блока. Такой подход не позволяет сделать подмену какого-либо из блоков цепочки, т.к. в таком случае **хеш** блоков перестанет сходиться. **Blockchain** хорош для систем с открытыми данными где требуется надёжная защита от подмены (фальсификации) данных.

https://en.bitcoin.it/wiki/Block_hashing_algorithm

Blockchain

Генерируем набор блоков в котором каждый последующий помимо данных содержит **хеш** предыдущего, это даёт гарантию невозможности подмены данных в одном из блоков, поскольку это вызовет несовпадение **хешей** от «*искажённого*» блока и далее по цепочке. **Однако** это не помешает злоумышленнику пересчитать все **хеши**, это потребует не таких уж больших трудозатрат...

8. Prof-of-Work

Proof-of-Work

```
1 "Hello, world!0" => 1312af178c253f84028d480a6...
2 "Hello, world!1" => e9afc424b79e4f6ab42d99c81...
3 "Hello, world!2" => ae37343a357a8297591625e71...
4 ...
5 "Hello, world!4248" => 6e110d98b388e77e9c6f04...
6 "Hello, world!4249" => c004190b822f1669cac8dc...
7 "Hello, world!4250" => 0000c3af42fc31103f1fdc...
```

Proof-of-Work механизм обеспечивающий невозможность подделки цепочки блоков за короткий промежуток времени. Суть механизма: блок считается созданным если его **хеш** меньше чем определено «сложностью», т.е. **хеш** должен быть меньше заранее известного числа. Упрощённо можно сказать, что в начале **хеша** должно быть определённое количество нулей. Достичь этого можно добавляя к **хешируемым** данным случайное значение.

https://en.bitcoin.it/wiki/Proof_of_work

Prof-of-Work


Proof-of-work – технология обеспечивающая «сложность» расчёта **хеша** для блока ввиду того, что **хеш** должен быть меньше заранее определённого числа. Поскольку найти такой **хеш** возможно только перебором случайных значений (добавляемых в блок дабы **хеши** изменялись) поиск может занять значительное время. Что не даст злоумышленнику быстро подменить всю цепочку **хешей**.

Bitcoin

Block #499925

BlockHash 00000000000000000006c534903383d583cb9a7d4a1b5e0a071f8a4380b5c7384 

Summary

Number Of Transactions	1449	Difficulty	1590896927258.0786
Height	499925 (Mainchain)	Bits	1800b0ed
Block Reward	12.5 BTC	Size (bytes)	941172
Timestamp	Dec 18, 2017 9:57:29 AM	Version	536870912
Mined by		Nonce	2781613532
Merkle Root	 7a674104ba5657d9c4d20e35e92b2...		
Previous Block	499924		

<https://btc.com/>

В основе **криптовалюты** – генерация блоков за каждый найденный блок даётся награда. Также в каждом блоке имеется информация о транзакциях между кошельками. За транзакции те кто находит блок (и включают в него информацию о транзакциях) получают комиссию. **Кошелёк** – пара ключей для **асимметричного** шифрования. **Номер кошелька** – **открытый ключ**.

Bitcoin



<https://ru.bitcoin.it/>

<https://habrahabr.ru/post/204008/>

<https://habrahabr.ru/company/intel/blog/205524/>

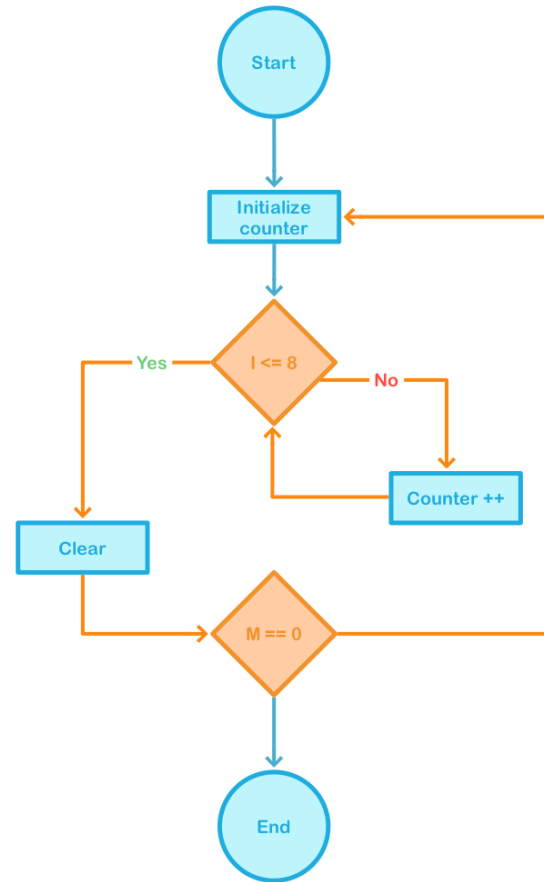
Bitcoin построен на базе **хеша**, **асимметричного шифрования**, **блокчейн** и **proof-of-work**.

Время подводить **ИТОГИ**



**Инструментов, у
JavaScript
разработчика, много...**

Но инструменты не решают задачи...



...задачи решают алгоритмы

Куда двигаться дальше?

Фреймворки & инструменты

Angular / **React** / **Vue.js** – популярные фреймворки для построения пользовательского интерфейса.

Развитие JavaScript

TypeScript – язык программирования построенный на базе **ECMAScript**;

Back-end

Express – **Node.JS** фреймворк для построения серверной части веб-приложений;

Storage

MongoDB – Не реляционная (**NoSQL**) система управления базами данных, управляемая при помощи **ECMAScript** диалекта;

Бёрстка

CSS – друг и помощник JavaScript-разработчика.

Ваше мнение о курсе Front end разработка на JavaScript

ORT DNIPRO

Насколько просто и понятно был изложен материал курса?

- ☐ Практически всё было понятно.
- ☐ Большая часть материала была понятна.
- ☐ Много чего было не понятно.
- ☐ Практически ничего не понял.

Оцените продолжительность курса:

- ☐ Курс нужно сделать длиннее.
- ☐ Продолжительность курса достаточна.
- ☐ Курс можно и сократить.

**Просьбы оставить
своё мнение о курсе
(всё анонимно 😊)**

**[https://forms.gle/YN
TyiHemkbSJJ8aQ6](https://forms.gle/YN
TyiHemkbSJJ8aQ6)**