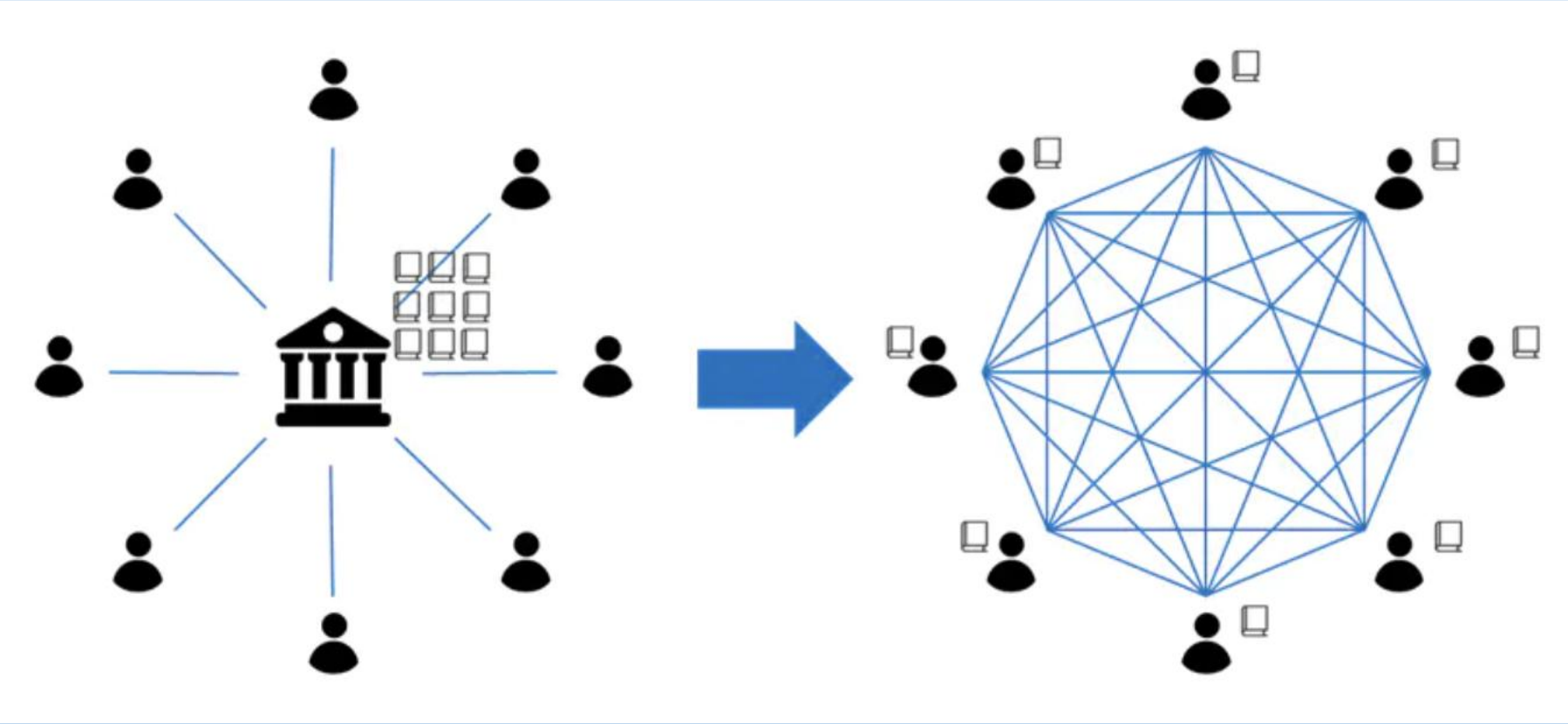
Ethereum P2P network

Student name: Shilin Zhuang
Student No: 46327321
(External)

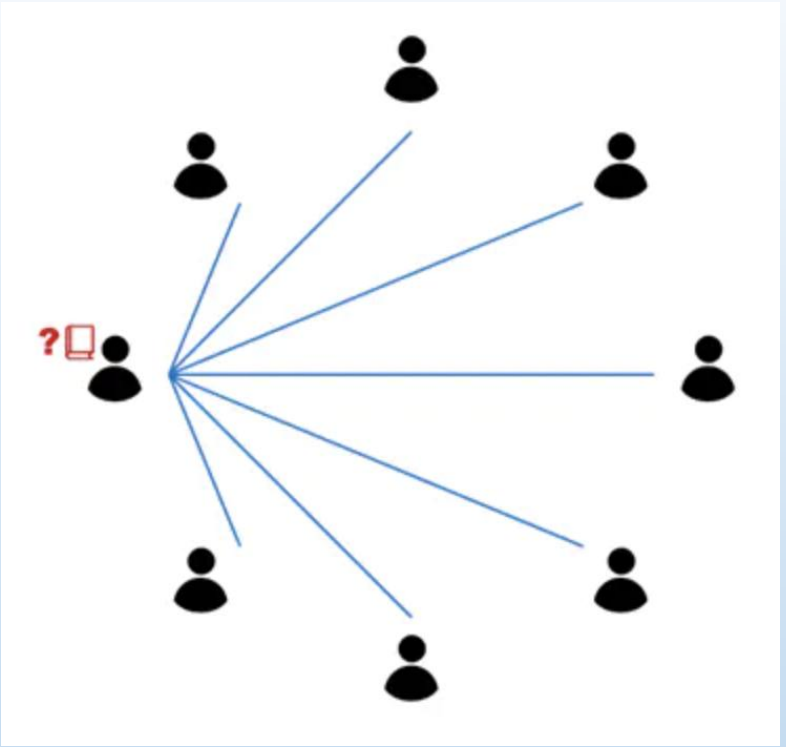
Thinking of a scene



From the central library to the distributed library

Ethereum P2P network

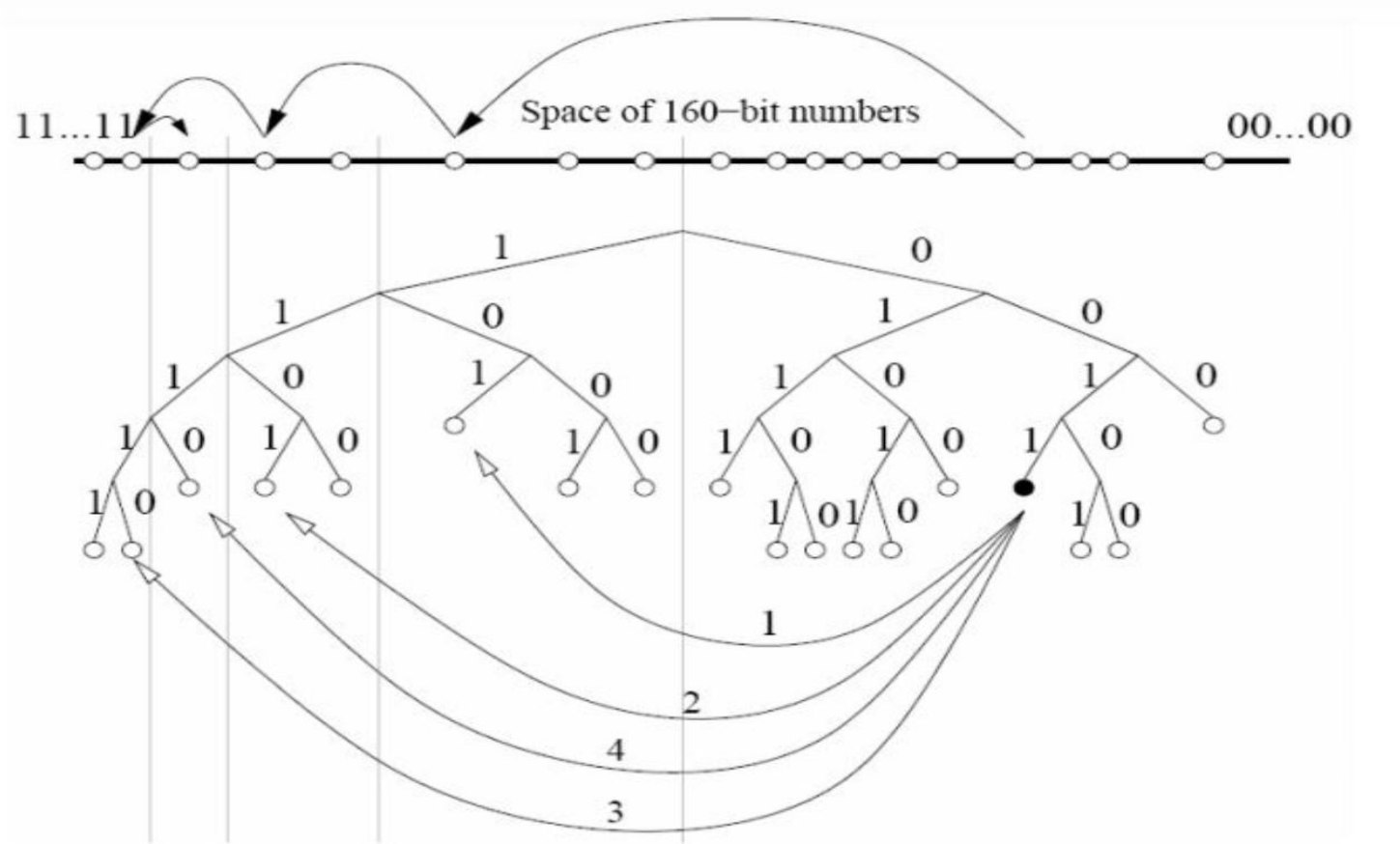
Some questions



1. Which books are assigned to each student?

2. When you need to find a book, how do you know which student has the book?

Kademlia(Kad) Algorithm



KAD network

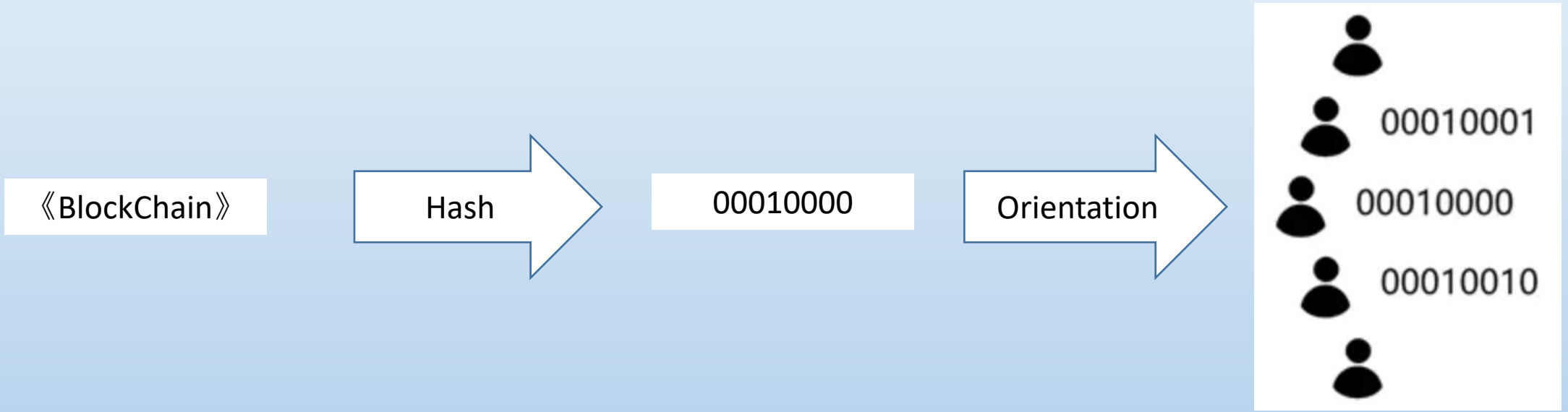
Ethereum P2P network

Element of Node

Kad Concept	Concept in the scene
Node ID (binary data)	Students number
Ip address, port	Student phone number
Key	Hash of a book
Value	book
Routing table, K-bucket	The address book of students(which contains <student.no, student.phone>)

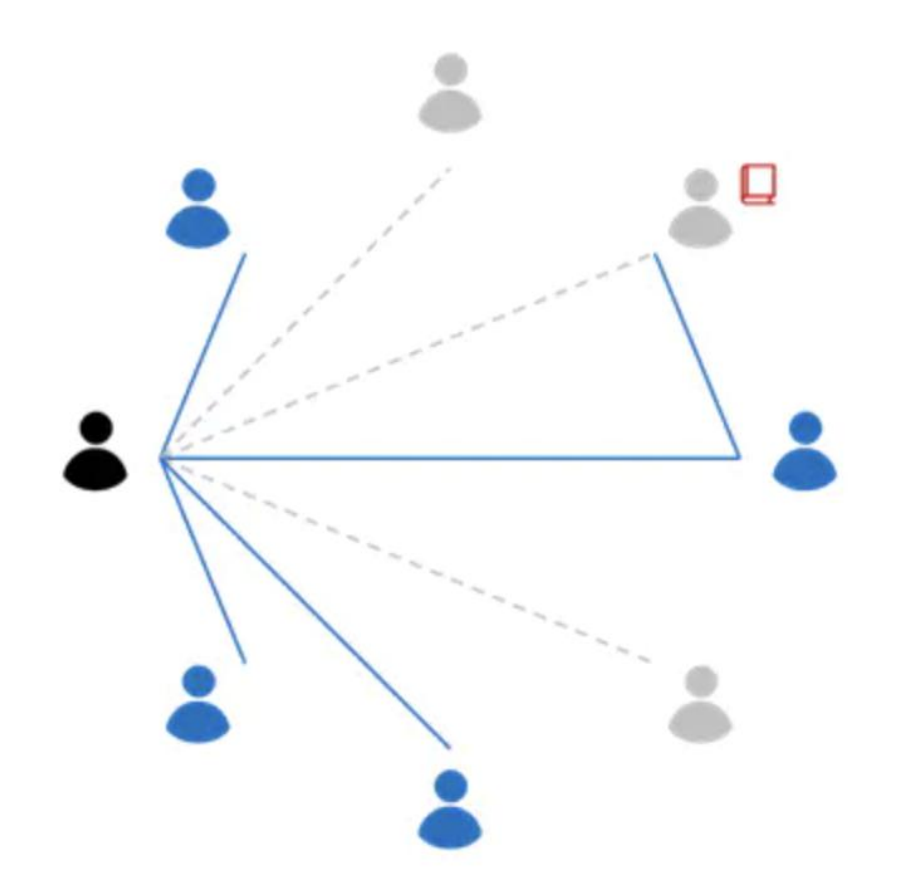
If we need to search for book《BlockChain》, how to do ?

- **KAD Description（1）**

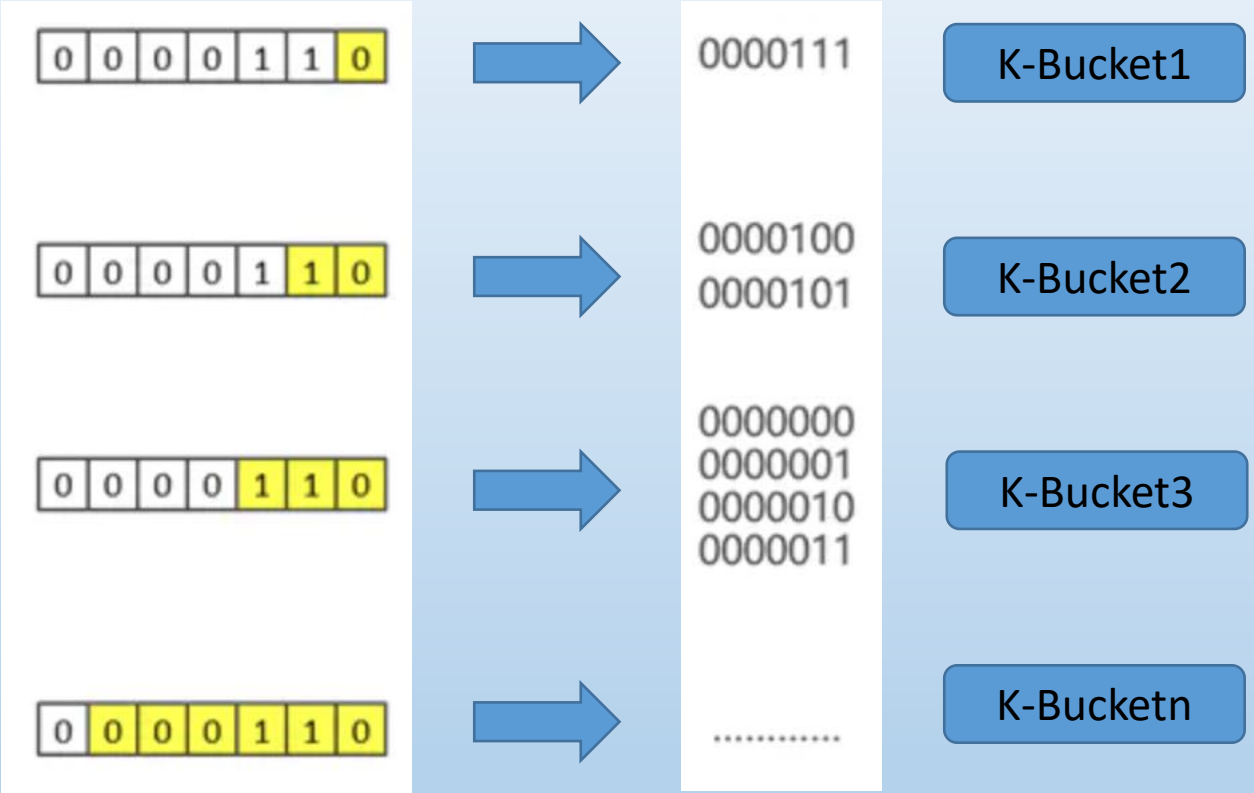


Q: If the 'student'(Node) is absence today?

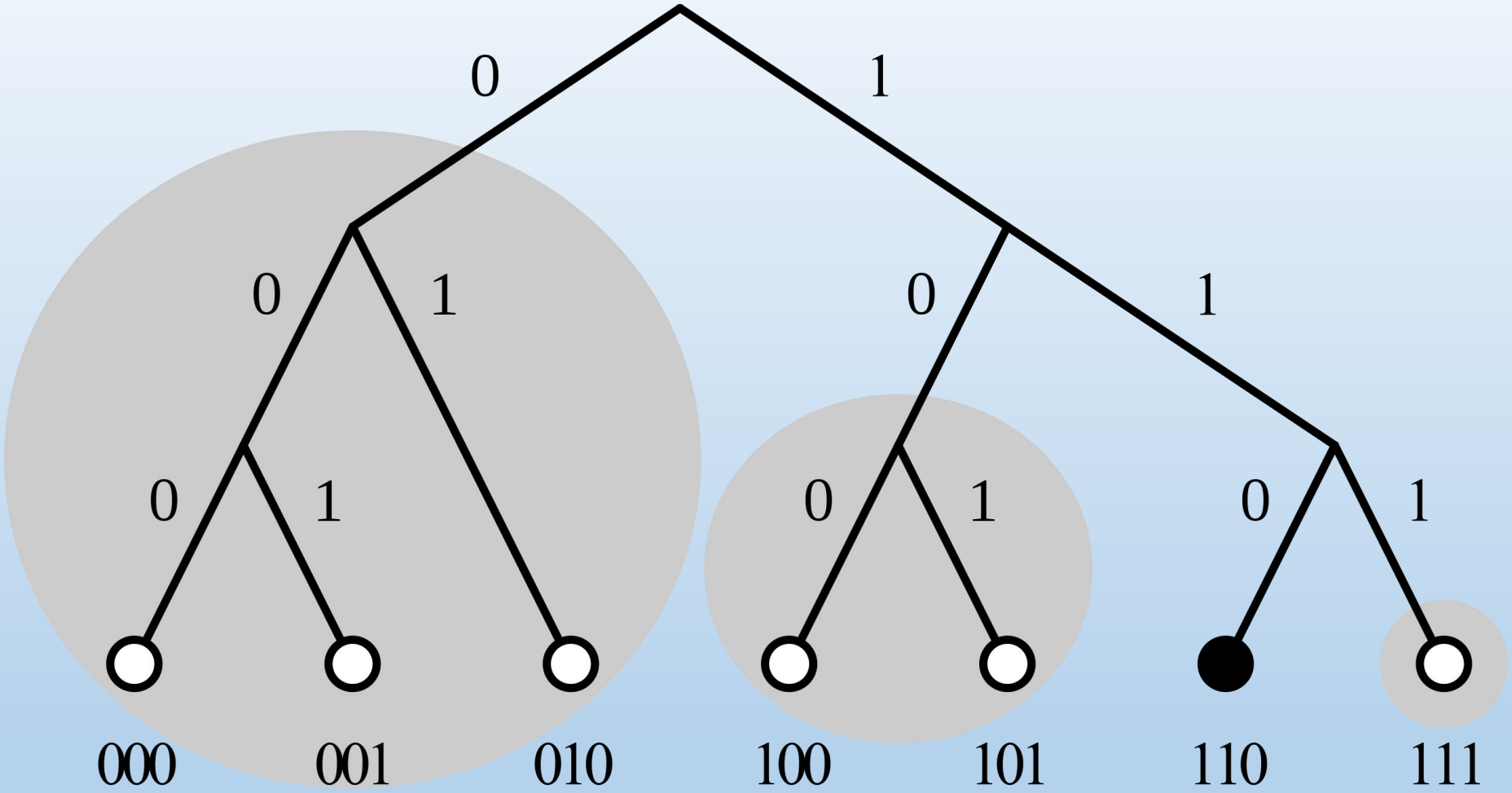
- **KAD Description（2）**



No target in the address book



It may contain 2^(n-1) nodes in the network



K-bucket3 distance
=[2^2, 2^3)

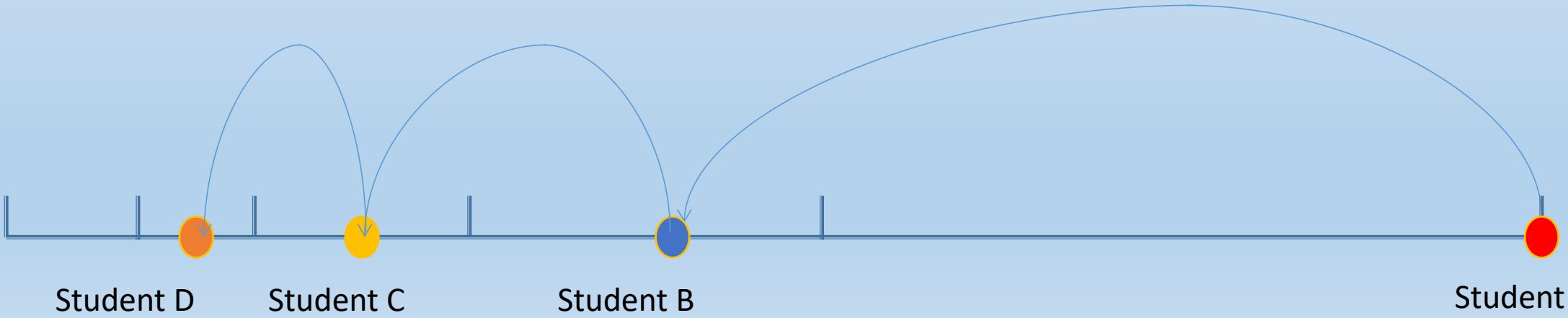
K-bucket2 distance
=[2^1, 2^2)

K-bucket1 distance
=[2^0, 2^1)

Ethereum P2P network

- “Folding paper” && Divided and conquer

Kademlia's query mechanism is a bit like shrinking the search scope by folding a piece of paper in half continuously, ensuring that for any N students, the contact information of the target student can be found only log2(n) times at most (that is, for any network with [2(n−1), 2n) nodes, It takes n steps at most to find the target node.



Ethereum P2P network

- **Four Instruction**

PING
-- Tests whether a node is online
STORE
A node is required to store one copy of data
FIND_NODE
-- Find a node by its ID
FIND_VALUE
To find a data based on a KEY, it is very similar to FIND_NODE

- The advantages of Kademlia

- **For any network with [2(n−1),2[?]) nodes, the target node can be found in n steps at most.**
- **The update mechanism of K-bucket maintains the activity and security of the network to some extent.**

Reference:

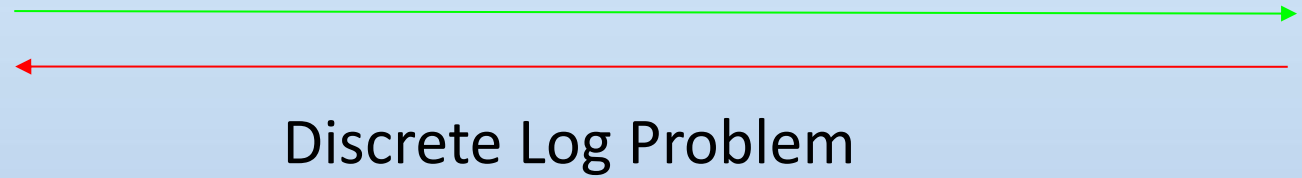
To understand the distributed | Kademlia algorithm
https://www.jianshu.com/p/f2c31e632f1d

Kademlia, Vikipedia
https://zh.wikipedia.org/wiki/Kademlia

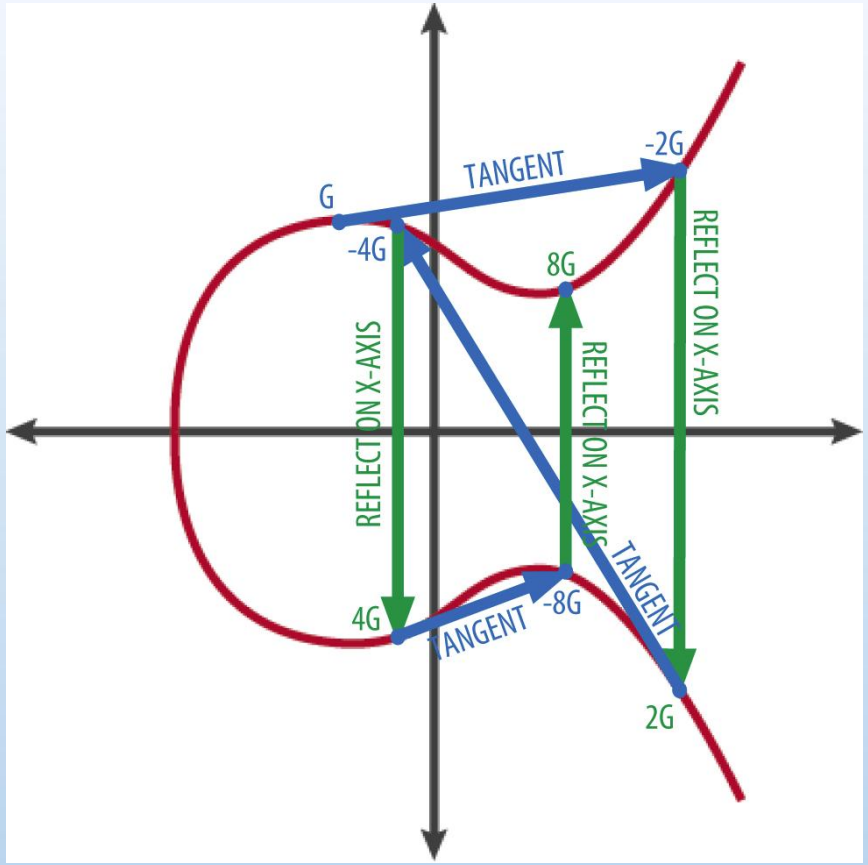
Security and Encryption

secp256k1: y^2 = x^3 + 7. The elliptic curve used by Bitcoin to implement its public key cryptography.

sk(private key) * G(Generator Point) = P(public key)



Discrete Log Problem

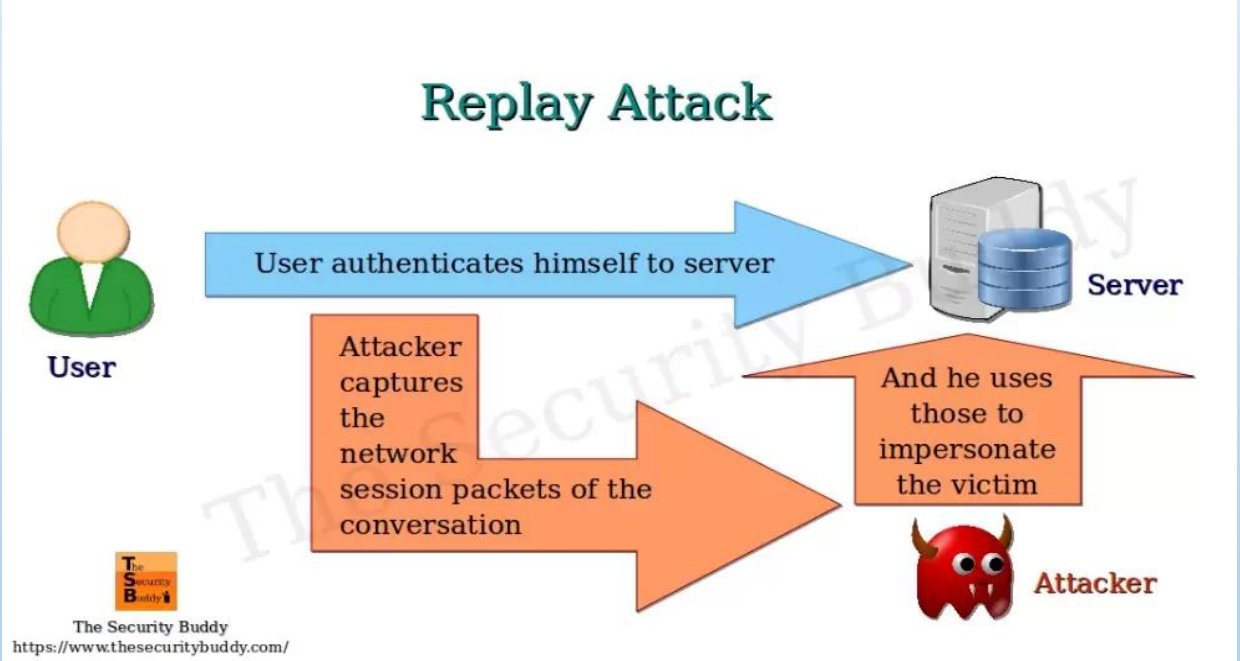


Each node maintains a static private key.

Packets are signed and can be verified with the public key recovered from signature.

Replay Attack Mitigation

Replay Attack: valid data transmission is maliciously or fraudulently repeated or delayed.

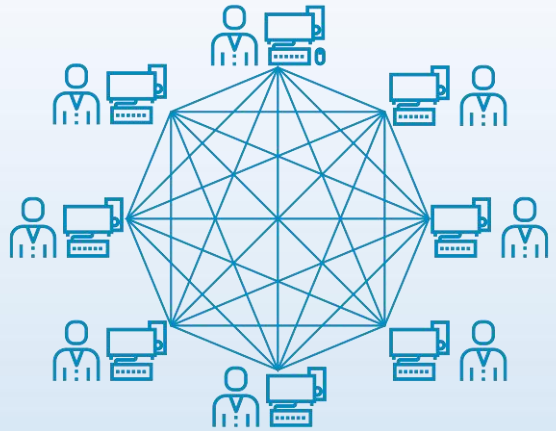


Mitigation by devp2p:

Timestamped Packet

Recommend: only accepts packets created within the last 3 seconds

Messaging



Devp2p messages(**"p2p" Capability**) : Message IDs between 0x00-0x10

“**Hello**” : 0x00

Implemented version, Client software identity, Peer capability name, Peer capability version, Port, Unique Identity of the node

“**Disconnect**” : 0x01

“reason” Parameter: 0x00 to 0x10

“**Ping**” : 0x02

“**Pong**” : 0x03

Sub-protocols: Message IDs of 0x10 onwards

Reference:

https://en.bitcoin.it/wiki/Secp256k1

https://river.com/learn/terms/s/secp256k1/

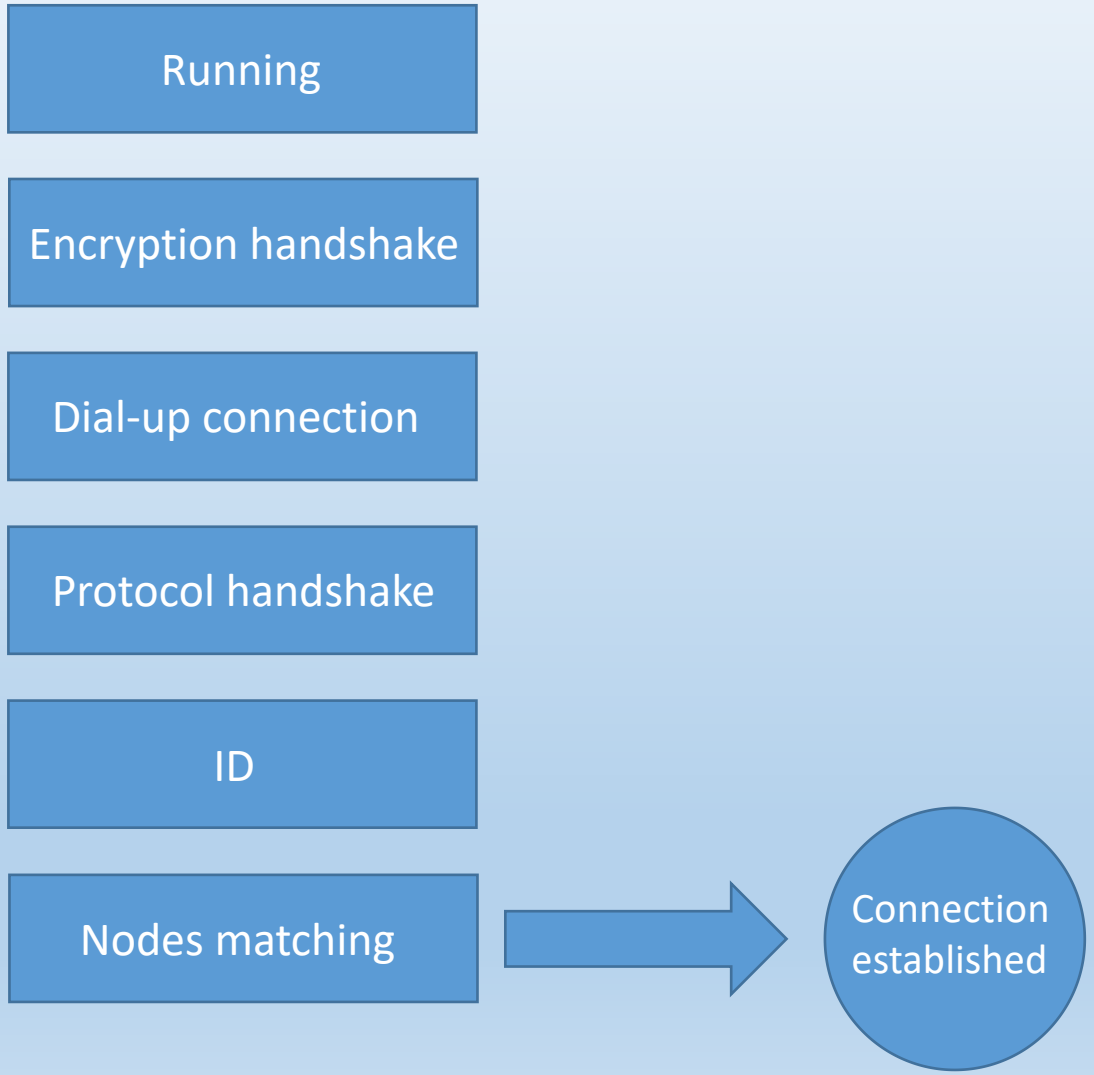
https://github.com/ethereumbook/ethereumbook/blob/05f0dfe6c41635ac85527a60c06ac5389d8006e7/contrib/devp2p-protocol.asciidoc

https://www.thesecuritybuddy.com/vulnerabilities/what-is-replay-attack/

Encryption Handshake

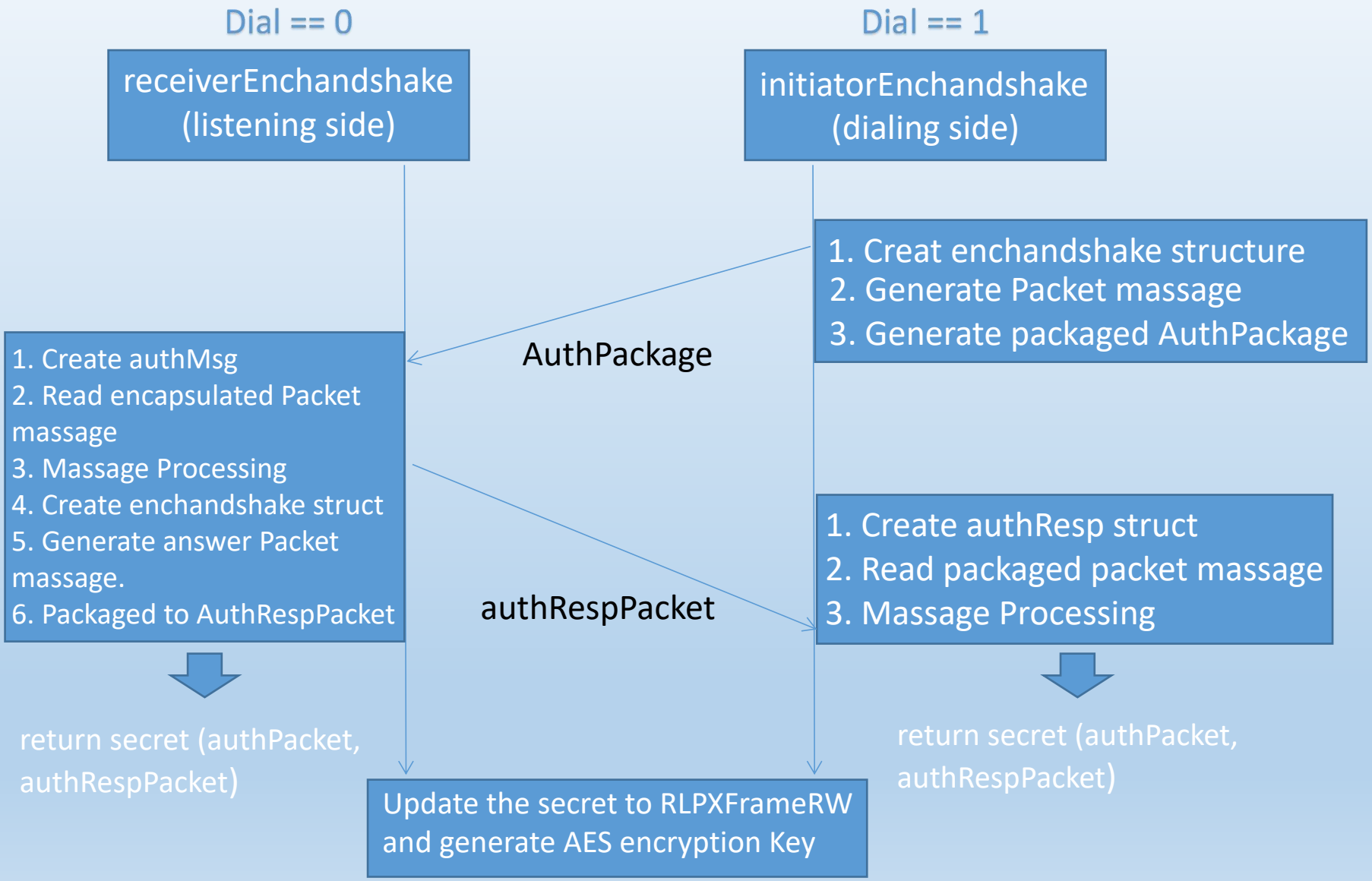
Encryption handshake protocol global location

Check whether the steps pass



Ethereum P2P network

Encryption handshake flow



initiator side:

1. Creat enchandshake structure
2. Generate Packet massage (makeAuthMsg)
3. Generate packaged AuthPackage

Initiator flow:

makeAuthMsg method:

1. initNonce: a random number.

2. ephemeral-privk and ephemeral-pubk （ECIES）.

3. privk & remote_pubk →static-shared-secrets.

4. static-shared-secrets xOR *init-nonce → hash value*

5. ECDSA (ephemeral-privk & hash value) → sig

----------makeAuthMsg Method finish----------

6. *authMsg: sig, pubk, nonce*

7. ECIES (authMsg & remote_pubk) → authPacket

----------AuthPackage----------

8.Wait to read the response from reveiver.

receiver side:

1. Create authMsg
2. Read encapsulated Packet massage
3. Massage Processing
4. Create enchandshake struct
5. Generate answer Packet massage.
6. Packaged to AuthRespPacket

initiator flow:

1. *authPacket* & *privk* → *authMsg*
2. *authMsg: remote_pubk* & *nonce*
3. *ECDH: ephemeral-privk* & *ephemeral-pubk*
4. *remote-pubk* & *privk* → *static-shared-secrets*
5. *nonce* xOR *static-shared-secrets* → *signedMsg*
6. *signedMsg* & *authMsg* → *remote-ephemeral-pubk*
7. *responseNonce* & *ephemeral-pubk* → *authRespMsg*
8. *authRespMsg* & *remote-pubk* → *authresponsePacket*
9. *Send the authresponsePacket to initiator*

initiator side:

1. Create authResp struct
2. Read packaged packet massage
3. Massage Processing

Initiator flow:

9.authresponsePacket & privk → authRespMsg

10.responce nonce & remote ephemeral-pubk

Sub protocol

Using *RLP*, we can encode different types of data, whose types are determined by the integer value used in the first entry of RLP. Thus, DEVp2p, the basic wire protocol, supports arbitrary subprotocols.

Reference:

https://ethereum.stackexchange.com/questions/37051/ethereum-network-messaging
https://www.bookstack.cn/read/ethereum_book-zh/spilt.2.87ffbfa317e55bda.md
https://github.com/sasankh/wiki/blob/master/%C3%90%CE%9EVp2p-Wire-Protocol.md
https://blog.csdn.net/weixin_45859485/article/details/122286000?utm_medium=distribute.pc_relevant.none-task-blog-2~default~baidujs_title~default-0.pc_relevant_default&spm=1001.2101.3001.4242.1&utm_relevant_index=3