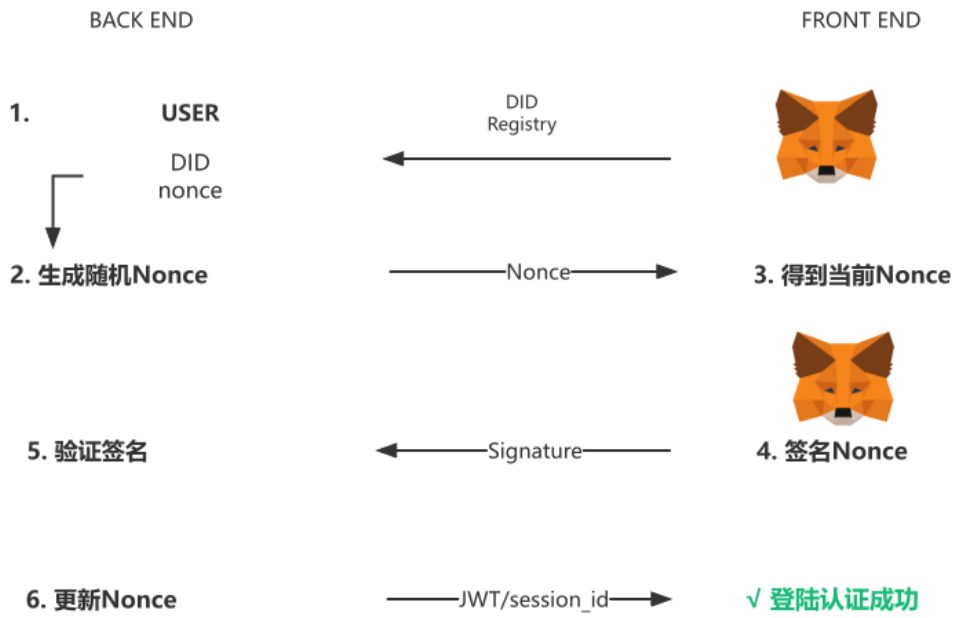
Metamask签名登陆流程

1. 流程
    1. 用户在前端网页上点击登录按钮，并且前端代码与 MetaMask 进行连接。
    2. 前端代码通过调用 `window.ethereum.request({ method: 'eth_requestAccounts' })` 来请求用户连接到 MetaMask。
    3. 后端通过Metamask的masterAccount以及DApp的信息，通过DID Contract查询到相应的用来登陆DID
    4. 前端向后端发送请求以获取与用户钱包地址关联的随机数（nonce）。
    5. 前端调用 `web3.personal.sign(nonce, web3.eth.coinbase, callback)` 来提示 MetaMask 显示签名确认弹窗，其中随机数会显示在弹窗中。
    6. 用户在 MetaMask 中确认签名，并通过回调函数将带有签名的消息（signature）传递给前端。
    7. 前端将签名消息与用户钱包地址一起发送到后端的身份验证接口。
    8. 后端根据用户钱包地址获取对应的随机数（nonce），并使用签名验证算法来验证签名的有效性。
    9. 如果签名验证成功，后端可以进行用户身份认证，并返回给前端一个身份标识符（如 JWT）以进行后续的授权访问。
    10. 为了防止重放攻击，后端会改变下次用户登录时所需的随机数（nonce）。
2. 流程图



3. js代码

```
(async () => {
    try {

        await window.ethereum.enable()
        const provider = new ethers.providers.Web3Provider(window.ethereum)
        const accounts = await provider.send("eth_requestAccounts", []);
```

```
        const defaultAccount = accounts[0]
        console.log(`default account: ${defaultAccount}`)
        //签名
        const signer = await provider.getSigner()
        const msg = web3.utils.asciiToHex("Hello Did!")
        const signature = await signer.signMessage(msg)
        console.log(signature)
        //验证
        const signStatus = verify(defaultAccount, signature, msg);
        console.log(`signStatus: ${signStatus}`)

    } catch (e) {
        console.log(e.message)
    }
  })()

  function verify(address, signature, msg) {
    let signValid = false
    console.log(`address: ${address}`)
    const decodedAddress = ethers.utils.verifyMessage(msg, signature)
    console.log(`decodedAddress: ${decodedAddress}`)
    if (address.toLowerCase() === decodedAddress.toLowerCase()) {
        signValid = true
    }
    return signValid
}
```

sdk

- [x] 封装DID Document格式
- [ ] 提供接口服务

name_service

合约

1. did->名
2. 名->did

- [x] 部署：合约地址 0x0eFCC08cD9831CE3d72c362A5b92011AAD26B23e
- [ ] 计费服务，（创建）周期管理
- [ ] 测试

其它

1. DID注册表：创建一个DID注册表智能合约，该合约将DID与名称进行映射并存储在区块链上。用户可以通过输入名称查询对应的DID，并进行验证和身份识别。
2. DNS解析：利用现有的域名系统（DNS）来实现DID和名称的映射。通过将DID作为特定域名的子域名，可以通过域名解析查询获得与该DID相关的信息。
3. 去中心化命名服务（Decentralized Naming Service）：创建一个去中心化的命名服务，类似于以太坊的ENS（Ethereum Name Service）或Handshake等。该服务将DID与易记的名称进行映射，并提供去中心化的解析和查询功能。
4. DID别名服务：创建一个DID别名服务，类似于电子邮件中的别名。用户可以将易记的名称与其DID关联，并在通信中使用该别名，而不是复杂的DID。别名服务将负责将别名映射到正确的DID上。

DID_service

- [x] 注册添加了权限，仅限owner

6.29

GitHub

跨链场景：NFT、存储

完整：SDK，did管理应用

重复创建（合约）

Name Service：传统

1

1. did合约修改
2. SDK的部分
3. name服务
    1. 生命周期管理
    2. 计费的相关
4. metamask did的登陆认证的流程
    1. 测试
    2. 完善
5. GitHub

---

github-Metamask签名示例

VC的项目流程：https://github.com/OriginProtocol/origin-playground

子did结构

ENS合约

朱永亮:
关于DID的事件存储，可以在合约里维护一个累加的hash，用来标记到当前位置所有的变更

朱永亮:
这样用户使用offchain服务提供的文档的时候，可以直接验证文档的正确性，不用遍历所有的event

0724

1. v2文档格式对应更新代码/注册合约
   - [x] 文档
   - [ ] 注册合约+hash
2. DNS服务确定接口文档
   - [ ]
3. sdk相关开发是否需要分工

3. 这块麻烦列下完整的接口文档和目前的 已实现/待实现

0728

DNS

DNS DID Name Service / 互联网名称服务，域名服务

Name Service:
  contract: DIDRegistry1.sol
  test: testname1.js
  content: basic function of resolution

ENS
  分级，不同的域用.分隔
  top layer（.eth .test...s) 由register contract注册

DNS注册合约

维护所有名称和子名称列表，并存储关于每个名称的三个关键信息:

- 名称的所有者：外部账户或合约地址
    - 为名称设置解析器和 TTL
    - 将名称的所有权转让给另一个地址
    - 更改子名称的所有权
- 名称的解析器：拥有顶级名称的智能合约
- 名称下所有记录的缓存存活时间（即 TTL）

DNS解析

首先，询问注册表是哪个解析器负责解析该名称，然后，向该解析器查询解析结果。

DNS部署

`DNSRegistry.sol` 部署到Goerli网络，地址为：`0x527Fd250a6bF9b6B5A5D1109Cc00ce93FB3D980e`

1. 注册相关函数：合约实现了一系列用于注册和管理ENS节点的函数，包括setRecord、setSubnodeRecord、setOwner、setSubnodeOwner、setResolver、setTTL等。
2. 查询相关函数：合约还实现了一系列用于查询ENS node的函数，包括getOwner、getResolver、getTtl、recordExists等。
3. 授权操作函数：合约实现了setApprovalForAll函数，用于授权其他地址（操作者）来管理调用者的ENS记录。
4. 内部函数：合约定义了一些内部函数，例如*setOwner*和setResolverAndTTL，用于在其他函数中进行节点所有者和解析器的设置。

- 测试函数

```
function setRecord(
        bytes32 node,
        address owner,
        address resolver,
        uint64 ttl
    ) external virtual override {
        setOwner(node, owner);
        _setResolverAndTTL(node, resolver, ttl);
    }
```

```
node:0x6574640000000000000000000000000000000000000000000000000000000000
owner:0x1cc5dc930549740508b253cd19a37f9b48ed79b7
resolver:0x36891a9D3c64d257D6C6bdff55CA0b2302983CEb
ttl:1722270681
```

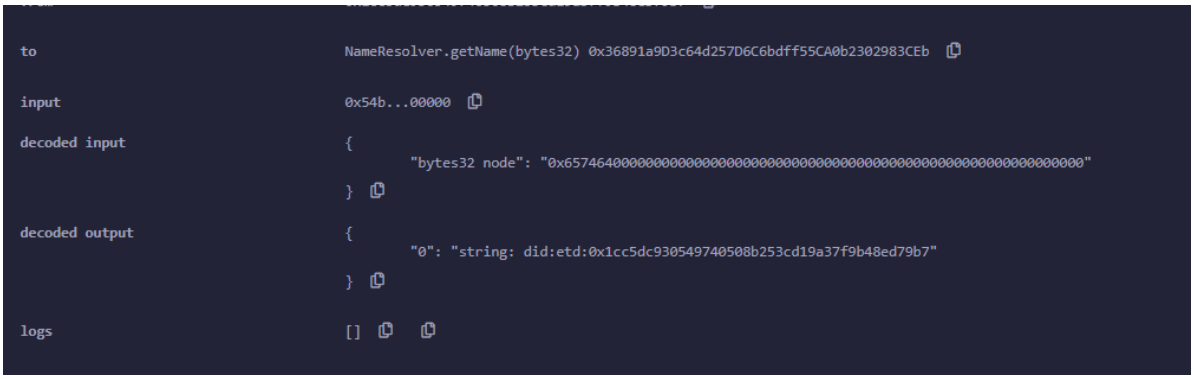
`NameResolver.sol` 部署到Goerli网络，地址为：`0x36891a9D3c64d257D6C6bdff55CA0b2302983CEb`

```
function setName(
        bytes32 node,
        string calldata newName
    ) external {
        names[node] = newName;
        emit NameChanged(node, newName);
    }


function name(
        bytes32 node
    ) external view returns (string memory) {
        return names[node];
    }
```


```
{

"blockHash":"0xec2d3c9c6601ded27c9071b46849cc0262ac35cdb4ddf8fd7e7ec2f9a35cd8e5"
,
   "blockNumber":9434295,
   "contractAddress":null,
   "cumulativeGasUsed":164685,
   "effectiveGasPrice":2500000040,
   "from":"0x1cc5dc930549740508b253cd19a37f9b48ed79b7",
   "gasUsed":92750,

"logsBloom":"0x00000000000000000000000000000000000000000000000010000000000000
00080000000000000000000000000000000000000000000000000000800000000000000000028
80040000080000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000200000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000",
   "status":true,
   "to":"0x36891a9d3c64d257d6c6bdff55ca0b2302983ceb",

"transactionHash":"0x730c7d54a42b724cdb2b8be7340f88d603c659d0d279c87beec818c9d5b
da4df",
   "transactionIndex":3,
   "type":"0x2",
   "events":{
      "NameChanged":{
```

```
          "address":"0x36891a9D3c64d257D6C6bdff55CA0b2302983CEb",

"blockHash":"0xec2d3c9c6601ded27c9071b46849cc0262ac35cdb4ddf8fd7e7ec2f9a35cd8e5"
,
          "blockNumber":9434295,
          "logIndex":1,
          "removed":false,

"transactionHash":"0x730c7d54a42b724cdb2b8be7340f88d603c659d0d279c87beec818c9d5b
da4df",
          "transactionIndex":3,
          "id":"log_4781c699",
          "returnValues":{

 "0":"0x6574640000000000000000000000000000000000000000000000000000000000000000",
             "1":"did:etd:0x1cc5dc930549740508b253cd19a37f9b48ed79b7",

 "node":"0x6574640000000000000000000000000000000000000000000000000000000000000000",
             "name":"did:etd:0x1cc5dc930549740508b253cd19a37f9b48ed79b7"
          },
          "event":"NameChanged",

"signature":"0xb7d29e911041e8d9b843369e890bcb72c9388692ba48b65ac54e7214c4c348f7"
,
          "raw":{

 "data":"0x0000000000000000000000000000000000000000000000000000000000000020000000
0000000000000000000000000000000000000000000000000000000326469643a6574643a30783
1636335646339333035343937343035303862323533636431396133376639623438656437396237 0
0000000000000000000000000",
             "topics":[

"0xb7d29e911041e8d9b843369e890bcb72c9388692ba48b65ac54e7214c4c348f7",

"0x6574640000000000000000000000000000000000000000000000000000000000000000"
             ]
          }
       }
    }
}
```


test result of getName



0807

DID

1. Enumerating Contracts Events to build the DID Document (2 Events)
    1. `DIDOwnerChanged` (indicating a change of `controller`)
    2. `DIDAttributeChanged`

```
event DIDAttributeChanged(
  address indexed identity,
  bytes32 name, //由于gas效率的原因不使用string
  bytes value,
  uint validTo, //有效期时间
  uint previousChange   //记录上一个属性改变的区块号
);
```

2. 方便查询事件
    1. 策略
        1. 合约维护一个名为`changed`的映射，记录最新的改变所在的区块号
        2. 每个Event都有`previousChanged`的属性，用来存储上一个更改事件的块号
    2. 查询Event所有历史的步骤
        1. eth_call `changed(address identity)` on the contract to get the latest block where a change occurred
        2. If result is `null` return.
        3. Filter for events for the above 3 types with the contract address on the specified block
        4. If event has a previous change then go to 3
3. 虽然可以存储任何属性，但对于 DID 文档，我们支持添加到 DID 文档的以下每个部分：
    - verificationMethod
    - authentication
    - proxywallet
    - service
4. Attribute Name 遵循以下命名规则，以满足DID Document属性的层次要求
    - verificationMethod
        - `vfm/id`: id
        - `vfm/type`: type
            - EcdsaSecp256k1VerificationKey
            - Ed25519VerificationKey2020
        - `vfm/con`: controller
        - `vfm/pkm`: publicKeyMultibase
        - `vfm/eth`: ethereumAddress
    - authentication
        - `atc/vfm/id`: verificationMethod id
        - `atc/id`: id

        - `atc/type`: type
        - `atc/con`: controller
        - `atc/pkm`: publicKeyMultibase
    - proxywallet
        - `pxw/vfm/id`: verificationMethod id
        - `pxw/id`: id
        - `pxw/type`: type
        - `pxw/con`: controller
        - `pxw/eth`: ethereumAddress
    - service
        - `svc/id`: id
        - `svc/type`: type
            - Linked Domains
            - DIDCommMessaging
            - CredentialRegistry
        - `svc/ep`: endpoint
5. 权限管理
    1. 创建不需要权限需要指定owner（一般钱包地址）
    2. 更改/新增属性需要权限，owner来调用合约
6. 记录创建时间
    - Attribute Name：created

DID合约部署

合约代码：

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.6;

contract EtdDIDReg{

    bytes32 constant private KEY_CREATED = "created";

    mapping(address => address) public owners;
    mapping(address => uint) public changed;

    modifier onlyOwner(address identity, address actor) {
    require (actor == identityOwner(identity), "bad_actor");
    _;
  }


    event AttributeChanged(
        address indexed identity,
        bytes32 name,
        bytes value,
        uint previousBlock,
        uint validTo
    );
```

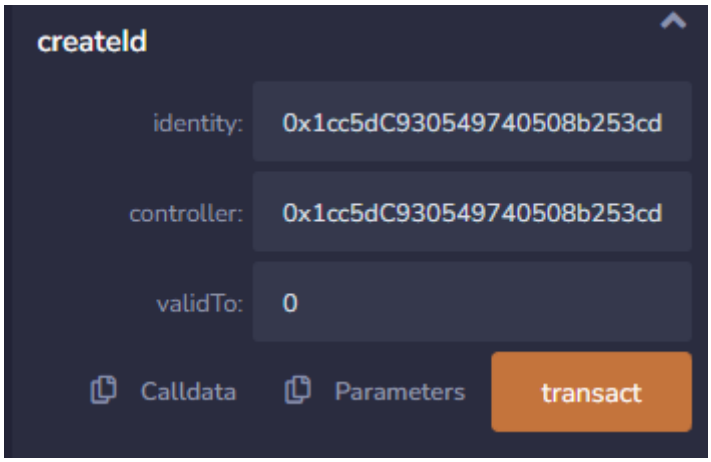
```
    event OwnerChanged(
        address indexed identity,
        address owner,
        uint previousBlock
    );

    function createId(
        address identity,
        address controller,
        uint validTo
    )
        public
    {
        uint currentTime = block.timestamp;
        bytes memory creationTime = abi.encodePacked(currentTime);
        owners[identity] = controller;
        changed[identity] = block.number;
        emit OwnerChanged(identity, controller, 0);
        emit AttributeChanged(identity, KEY_CREATED, creationTime, 0, validTo);
    }

    function changeOwner(address identity, address newOwner) public
onlyOwner(identity, msg.sender) {
        owners[identity] = newOwner;
        changed[identity] = block.number;
        emit OwnerChanged(identity, newOwner, changed[identity]);
    }

    function setAttribute(address identity, bytes32 name, bytes memory value,
uint validTo ) public onlyOwner(identity, msg.sender) {
        emit AttributeChanged(identity, name, value, changed[identity],
validTo);
        changed[identity] = block.number;
    }

    function identityOwner(address identity) public view returns(address) {
        address owner = owners[identity];
        if (owner != address(0x00)) {
            return owner;
        }
    return identity;
    }

}
```

ETD合约地址：`0x85858fe01A9EF40f956B259c5709942Ffa074F5B`

任务分配

DID SDK

- [x] 测试合约
    - createId



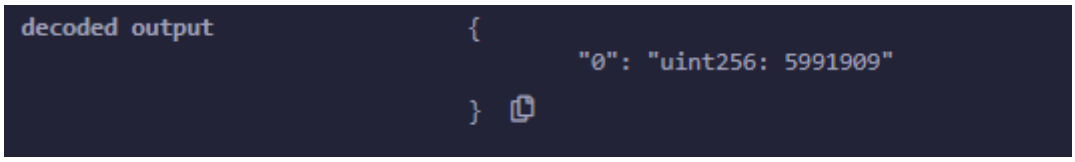

```
//block number
5991880

//log
[
    {
        "from": "0x3a1bb4f1826A555CC5B0b20B61e1137E178dc4Cc",
        "topic":
"0x4c37b24b600916176446859ec41fb06842ec1dfaeeb0bee28784b51f24b8c308",
        "event": "OwnerChanged",
        "args": {
            "0": "0x1cc5dC930549740508b253cd19a37f9b48eD79b7",
            "1": "0x1cc5dC930549740508b253cd19a37f9b48eD79b7",
            "2": "0",
            "identity": "0x1cc5dC930549740508b253cd19a37f9b48eD79b7",
            "owner": "0x1cc5dC930549740508b253cd19a37f9b48eD79b7",
            "previousBlock": "0"
        }
    },
    {
        "from": "0x3a1bb4f1826A555CC5B0b20B61e1137E178dc4Cc",
        "topic":
"0x024a588a8e0cc47a69d57e77c56bc4525491dafb13d7d2553025177d4aeb9746",
        "event": "AttributeChanged",
        "args": {
            "0": "0x1cc5dC930549740508b253cd19a37f9b48eD79b7",
            "1":
"0x6372656174656400000000000000000000000000000000000000000000000000",
            "2":
"0x00000000000000000000000000000000000000000000000000000064d111f6",
            "3": "0",
            "4": "0",
            "identity": "0x1cc5dC930549740508b253cd19a37f9b48eD79b7",
            "name":
"0x6372656174656400000000000000000000000000000000000000000000000000",
```

```
            "value":
"0x000000000000000000000000000000000000000000000000000000000064d111f6",
            "previousBlock": "0",
            "validTo": "0"
        }
    }
]
```


- setAttribute
  - input


```
//blocknumber
5991909
//log
[
    {
        "from": "0x3a1bb4f1826A555CC5B0b20B61e1137E178dc4Cc",
        "topic":
"0x024a588a8e0cc47a69d57e77c56bc4525491dafb13d7d2553025177d4aeb9746"
,
        "event": "AttributeChanged",
        "args": {
            "0": "0x1cc5dC930549740508b253cd19a37f9b48eD79b7",
            "1":
"0x76666d2f696400000000000000000000000000000000000000000000000000000"
,
            "2": "0x2370726f787977616c6c65742d31",
            "3": "5991880",
            "4": "0",
            "identity":
"0x1cc5dC930549740508b253cd19a37f9b48eD79b7",
            "name":
"0x76666d2f696400000000000000000000000000000000000000000000000000000"
,
            "value": "0x2370726f787977616c6c65742d31",
            "previousBlock": "5991880",
            "validTo": "0"
        }
    }
]
```


- changed



- [ ] 封装DID Document
  - metadata
  - event OwnerChanged

- event AttributeChanged

- [ ] 创建DID API
  - 提供创建数据
  - 输出封装DID Document
- [ ] 更改DID Document Attribute API
  - 提供属性name & value
  - 输出封装DID Document

DNS

- [ ] DNS合约完善