Keypom 核心原理

LAK

- NEAR的账户与Key

```
pub enum AccessKeyPermission {
    FunctionCall(FunctionCallPermission),
    FullAccess,
}
```

```
pub struct FunctionCallPermission {
    /// Allowance is a balance limit to use by thi
    /// transaction fees. When this access key is
    /// decreased by the same value.
    /// `None` means unlimited allowance.
    /// NOTE: To change or increase the allowance,
    /// access key should be created.
    pub allowance: Option<Balance>,

    /// The access key only allows transactions wi
    pub receiver_id: AccountId,

    /// A list of method names that can be used. T
    /// function call of one of the given method n
    /// Empty list means any method name can be us
    pub method_names: Vec<String>,
}
```

Key: https://nomicon.io/DataStructures/AccessKey

Accoount: https://nomicon.io/DataStructures/Account

Account

Data for an single account is collocated in one shard. The accou

- Balance
- Locked balance (for staking)
- Code of the contract
- Key-value storage of the contract. Stored in a ordered trie
- Access Keys
- Postponed ActionReceipts
- Received DataReceipts

- 进一步限制FC Key

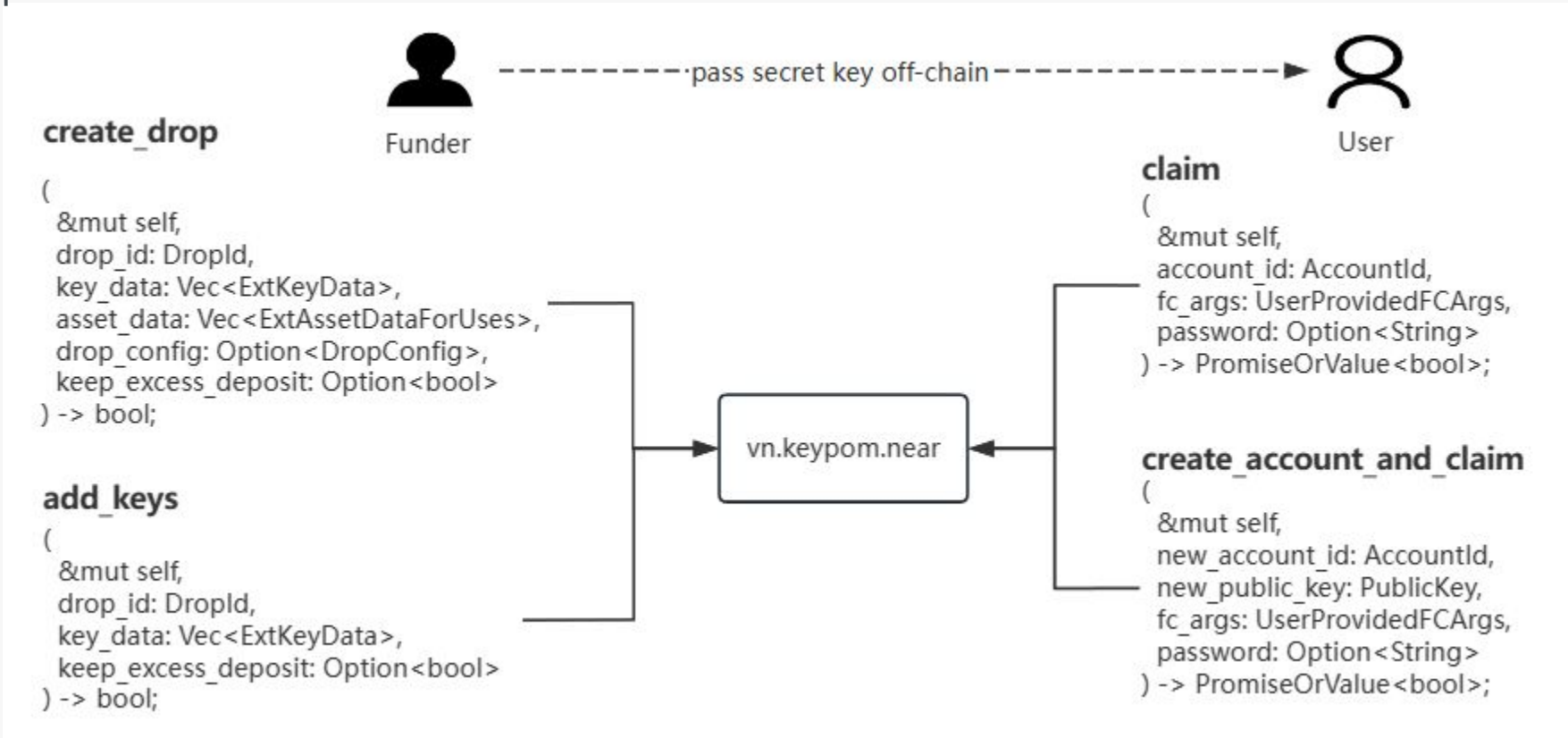
```
Keys for account v2.keypom.near
[
  {
    access_key: {
      nonce: 92222411000000,
      permission: {
        FunctionCall: {
          allowance: '202481569103872000000000',
          method_names: [ 'claim', 'create_account_and_claim' ],
          receiver_id: 'v2.keypom.near'
        }
      }
    },
    public_key: 'ed25519:19xKEKrq1CHQ5K7DQSDKT9LwaBnGUQ3nNVcteV6z
  },
  {
```

Key Interfaces

- Core Smart Contract

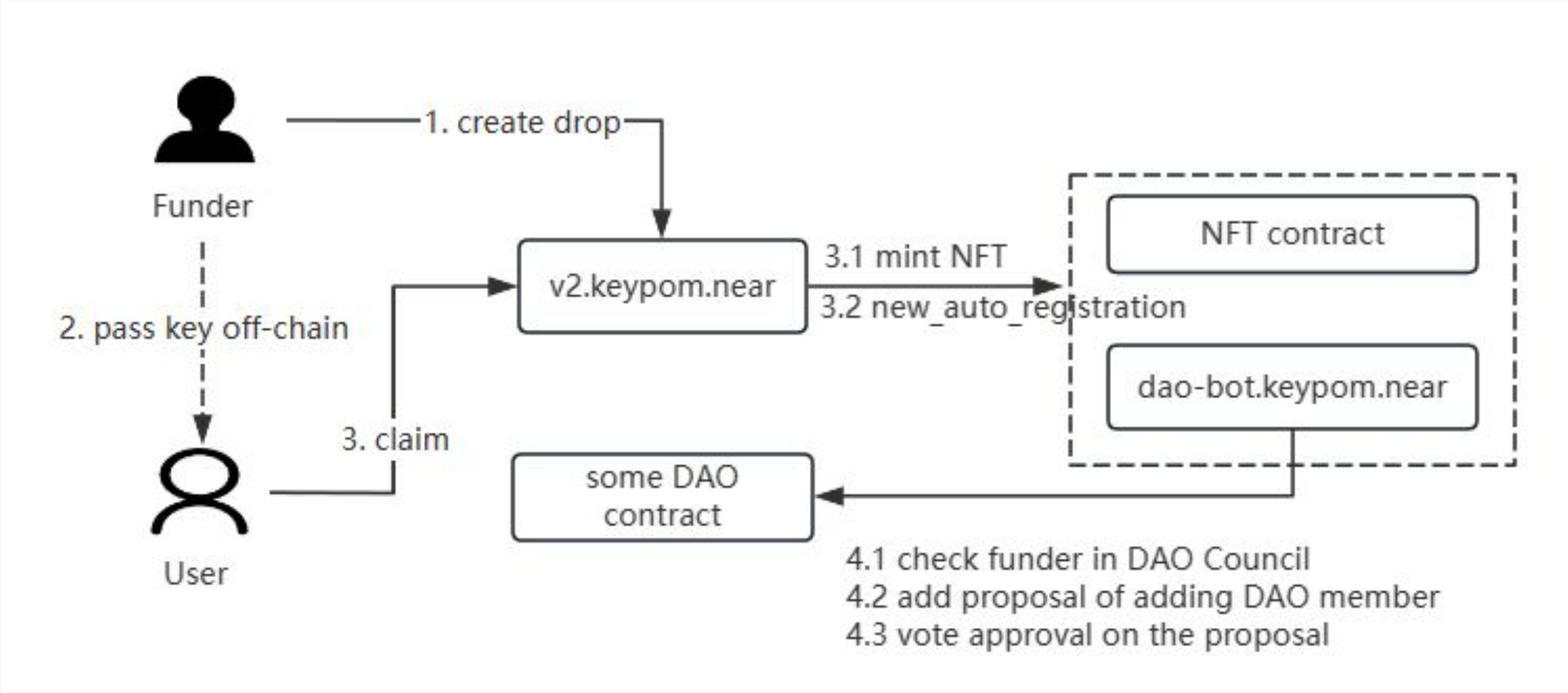
keypom v1.4 v2 v3 https://github.com/keypom/

v2.keypom.near / v2.keypom.testnet



Keypom 用例逻辑分析

Auto-register DAO



Auto-register DAO

- Create Drop

  TX: FF4Ro9A4HZE1wZh7UZDNTScJQK1e2h88R12reFoFFn6


```
{
  "drop_id": "1692215772806",
  "public_keys": [
    "ed25519:FmY28Vpzt6fdttAjD7zB6GUgJLPNi6WYCDoikPR3Rspe"
  ],
  "deposit_per_use": "3185000000000000000000",
  "config": {
    "uses_per_key": 2,
    "usage": {
      "auto_delete_drop": true,
      "auto_withdraw": true,
      "permissions": null,
      "refund_deposit": true,
      "account_creation_fields": {}
    }
  },
  "fc": {
    "methods": [
      [
        {
          "receiver_id": "nft.bluntdao.near",
          "method_name": "nft_mint",
          "args": "{\"id\":\"1\"}",
          "attached_deposit": "8000000000000000000000",
          "account_id_field": "receiver_id"
        },
        {
          "receiver_id": "dao-bot.keypom.near",
          "method_name": "new_auto_registration",
          "args": "{\"dao_contract\":\"blunt.sputnik-dao.near\",\"proposal\":{\"description\":\"Auto-Registering New Member\",\
          "attached_deposit": "10900000000000000000000",
          "account_id_field": "proposal.kind.AddMemberToRole.member_id",
          "funder_id_field": "funder"
        }
```

Auto-register DAO

- Clam Drop

  TX: 2BokfpCkWpTwaTFJTwZWKv3bcMnGHAeR2Fzro6LuVPAA

Called method: 'claim' in contract: v2.keypom.near

Arguments:

```
{
  "account_id": "mintlu.near"
}
```

```
{
  "receiver_id": "dao-bot.keypom.near",
  "method_name": "new_auto_registration",
  "args": "{\"dao_contract\":\"blunt.sputnik-dao.near\",\"proposal\":
  "attached_deposit": "1090000000000000000000",
  "attached_gas": null,
  "account_id_field": "proposal.kind.AddMemberToRole.member_id",
  "drop_id_field": null,
  "key_id_field": null,
  "funder_id_field": "funder",
  "receiver_to_claimer": null,
  "user_args_rule": null
}
```

Called method: 'on_claim_fc' in contract: v2.keypom.near

Arguments:

```
{
  "account_id": "mintlu.near",
  "funder_id": "sharddog.near",
  "balance": "3185000000000000000000",
  "storage_used": 0,
  "fc_data": {
    "methods": [
      [
        {
          "receiver_id": "nft.bluntdao.near",
          "method_name": "nft_mint",
          "args": "{\"id\":\"1\"}",
          "attached_deposit": "8000000000000000000000",
          "attached_gas": null,
          "account_id_field": "receiver_id",
          "drop_id_field": null,
          "key_id_field": null,
          "funder_id_field": null,
          "receiver_to_claimer": null,
          "user_args_rule": null
        },
```

Auto-register DAO

- dao-bot

[actually add member]

```
{
  name: 'council',
  kind: { Group: [ 'bluntdao.near' ] },
  permissions: [
```

```
{
  name: 'keypom',
  kind: { Group: [ 'dao-bot.keypom.near' ] },
  permissions: [
    'add_member_to_role:VoteRemove',
    'remove_member_from_role:AddProposal',
    'vote:AddProposal',
    'add_member_to_role:VoteApprove',
    'add_member_to_role:VoteReject',
    'add_member_to_role:AddProposal',
    'call:AddProposal'
  ],
```

Called method: 'new_auto_registration' in contract: dao-bot.keypom.near

Arguments:


```
{
  "dao_contract": "blunt.sputnik-dao.near",
  "funder": "sharddog.near",
  "keypom_args": {
    "account_id_field": "proposal.kind.AddMemberToRole.member_id",
    "drop_id_field": null,
    "funder_id_field": "funder",
    "key_id_field": null
  },
  "proposal": {
    "description": "Auto-Registering New Member",
    "kind": {
      "AddMemberToRole": {
        "member_id": "mintlu.near",
        "role": "blunts"
      }
```

Password Protected Keys and POAP

- Backend Logic

  One kind of conditional claim, i.e. password protected claim.

  password for a key: ***pwd = hash(base_password + PK)***

  password for a key use: ***pwd = hash(base_password + PK + use_number)***

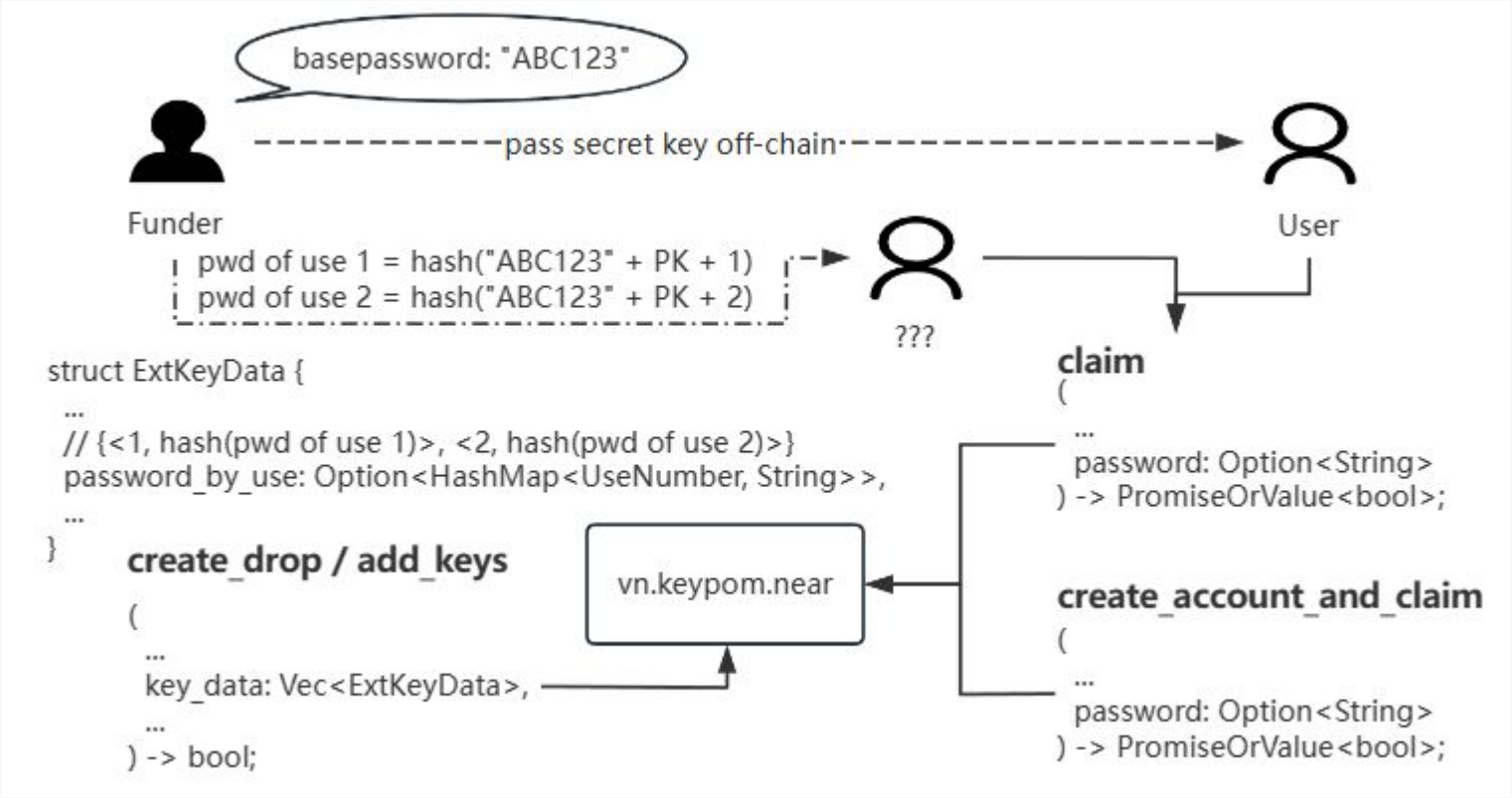
  learn details here.

- On Creating Drop

  Put ***hash(pwd)*** into contract

- On Claiming

  Add ***pwd*** as additional argument



Password Protected Keys and POAP

- POAP Usecase
  - For each key(ticket) in the drop: 2 uses, the first one has password, the second is normal.
  - User has sk, gateman has base password in his ticket checker dapp.
  - Ticket checker process is to calculate password of use 1 and do the 1st claim using the key.
  - POAP proof could be gained by user himself anytime afterward, as the keys 2ed use.
  - Could be extended to marketing and engagement events
- Features
  - Tickets wont go valid until holders actually go to the event
  - No internet connection needed for users at event
  - Near wallet is not necessary for user to attend event

Trial Account

- Combine On SDK Side
  - Organization side: createTrialAccountDrop(...)
  - User side: claimTrialAccountDrop(...)
  - see details on here.
- Hints
  - Account created on claim
  - AA deployed on claim
  - limits defined by org

```
477     * @example
478     * Creating a trial account with any callable methods, an amount of 0.5 $NEAR and 5 keys.
479     * ```js
480     * const callableContracts = [
481     *  `v1.social08.testnet`,
482     *  'guest-book.examples.keypom.testnet',
483     * ]
484     *
485     * const {dropId, keys: {secretKeys: trialSecretKeys, publicKeys: trialPublicKeys}}
486     * = await createTrialAccountDrop({
487     *  numKeys: 1,
488     *  contractBytes: [...readFileSync('./test/ext-wasm/trial-accounts.wasm')],
489     *  startingBalanceNEAR: 0.5,
490     *  callableContracts: callableContracts,
491     *  callableMethods: ['set:grant_write_permission', '*'],
492     *  maxAttachableNEARPerContract: callableContracts.map(() => '1'),
493     *  trialEndFloorNEAR: 0.33
494     * })
495     *
496     * const newAccountId = `${Date.now().toString()}.linkdrop-beta.keypom.testnet`
497     * await claimTrialAccountDrop({
498     *     secretKey: trialSecretKeys[0],
499     *     desiredAccountId: newAccountId,
500     * })
```

Use Trial Account

- Trial Account Smart Contract
  - setup(): setup AA rules. (Beginning)
  - execute(): the proxy entrance for user. (Running)
  - create_account_and_claim(): convert trail to a regular account. (Ending)
  - repo is here.
- SDK Support
  - trialSignAndSendTxns(...)
  - trialCallMethod(...)
  - see details on here.

```
export const trialSignAndSendTxns = async ({
    trialAccountId,
    trialAccountSecretKey,
    txns,
}: {
```

```
export const trialCallMethod = async ({
    trialAccountId,
    trialAccountSecretKey,
    contractId,
    methodName,
    args,
    attachedGas,
    attachedDeposit,
}: {
```

NFT Access Key Ticketing

Useful References

- https://keypom.xyz/
- https://docs.keypom.xyz/
- https://github.com/keypom