Why Account Abstraction is a Game-Changer for Dapps

Scaling self-custody

Julien Niset

co-founder@Argent

argent

Section 1

Self-Custody Matters!

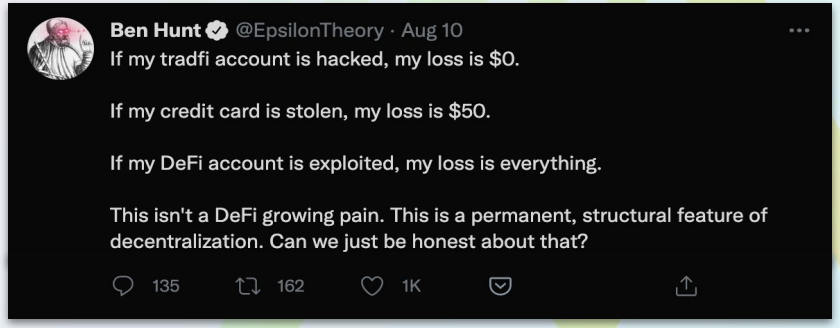
Blockchains enable true digital ownership.
But you don't own an asset if you don't have custody!

THE LEDGER › CRYPTOCURRENCY

Crypto lender Celsius suspends withdrawals, locking up billions in user funds amid 'extreme market conditions'

BY EAMON BARRETT
June 13, 2022 9:20 AM GMT+2

But Self-Custody is hard



The problem: Ethereum Accounts

Two types of Accounts on Ethereum:

- Externally Owned Account (EOA) → can initiate transactions
- Contract Account (CA) → can contain logic

An (**EOA**) Account has:

- an **address** for identification
- a **nonce** to make sure transactions are unique
- a **balance** in ETH to pay fees

A user *owns* an EOA through a pair of cryptographic keys (aka a **Signer**):

- The address of the account is derived from the **public key**
- Transactions from the account must be signed by the **private key**

The signature scheme of Ethereum is ECDSA on the elliptic curve secp256k1.

The problem: Ethereum Accounts

Signer

(k_{priv}, k_{pub})

EOA

address = keccak(k_{pub})[12:32]

nonce + balance

EVM logic to **validate** transactions f(k_{priv})

EVM logic to **execute** transactions

ECDSA on Secp256k1

User *Ethereum*

argent

The problem: Ethereum Accounts

Signer

(k_{priv}, k_{pub})

EOA

address = keccak(k_{pub})[12:32]

nonce + balance

EVM logic to **validate** transactions f(k_{priv})

EVM logic to **execute** transactions

ECDSA on Secp256k1

User *Ethereum*

The concept of **Signer** and the concept of **Account** are merged.

Authorised to spend your tokens

Holds your tokens

argent

This will not work for mainstream adoption!

EOAs are the Problem

If the Signer **IS** the Account, and vice versa, then...

- If you lose your Signer you lose your Account!
- If I have your Signer I have your Account!

The entire security of Ethereum relies on users managing a single secret (k_{priv})

There is no safety net; users can't make mistakes

argent

Section 2

Can we do better?

Account Abstraction!

Decouple the relation between Account and Signer

→ Signer ≠ Account

The Account is a smart-contract that defines what a valid transaction is:

- Different Signature scheme?
- Different elliptic curve?
- Multiple Signers?
- Signer can be replaced?

Account Abstraction means every account is a Smart-contract wallet that can initiate transactions and pay the fee.

No more EOAs!!!

AA was always on the Ethereum roadmap

EIP-86: introduces "forwarding contract" to abstract signature verification and nonce.

Smart-contract wallets: program smart-contracts to secure user assets. Enables social recovery, on-chain fraud monitoring, etc.

EIP-2938: introduces new AA transaction to enable smart-contracts to act as top-level accounts.

EIP-3074: make existing EOAs behave like smart-contracts by allowing users to *delegate* control of their EOA to a contract.

EIP-4337: decentralises the infrastructure needed to write and operate smart-contract wallets. No protocol change needed!

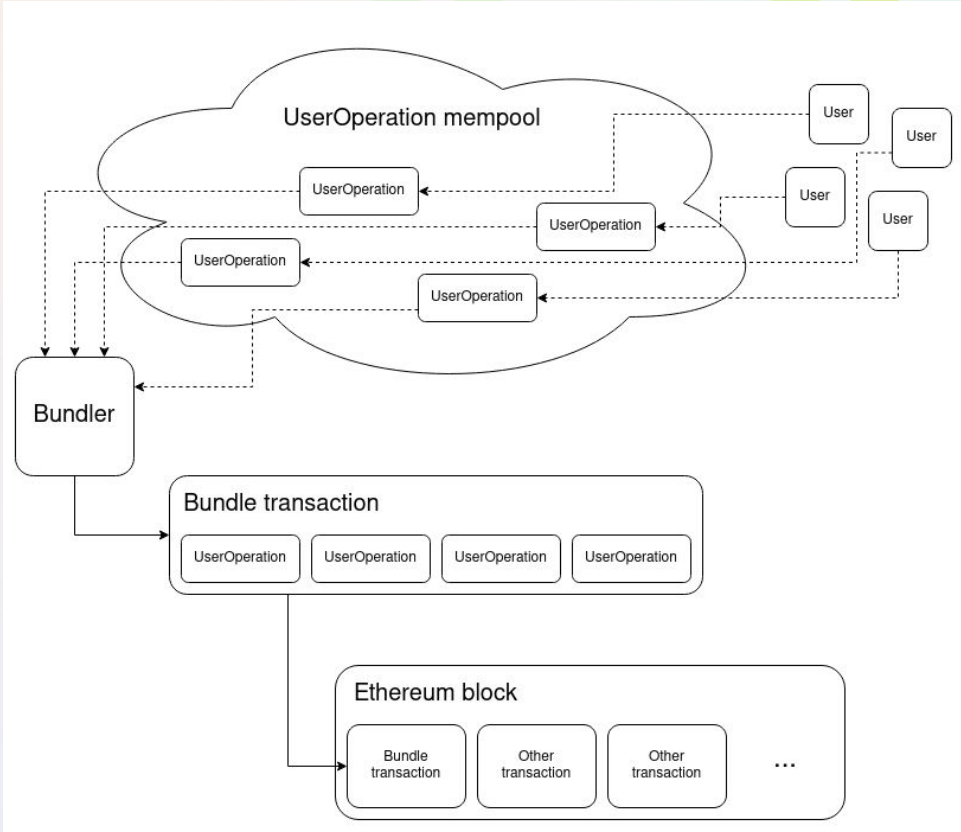
argent

EIP 4337

1. User sends **UserOperation** to higher level mempool

2. **Bundler** bundles UserOperations into a bundle transaction.

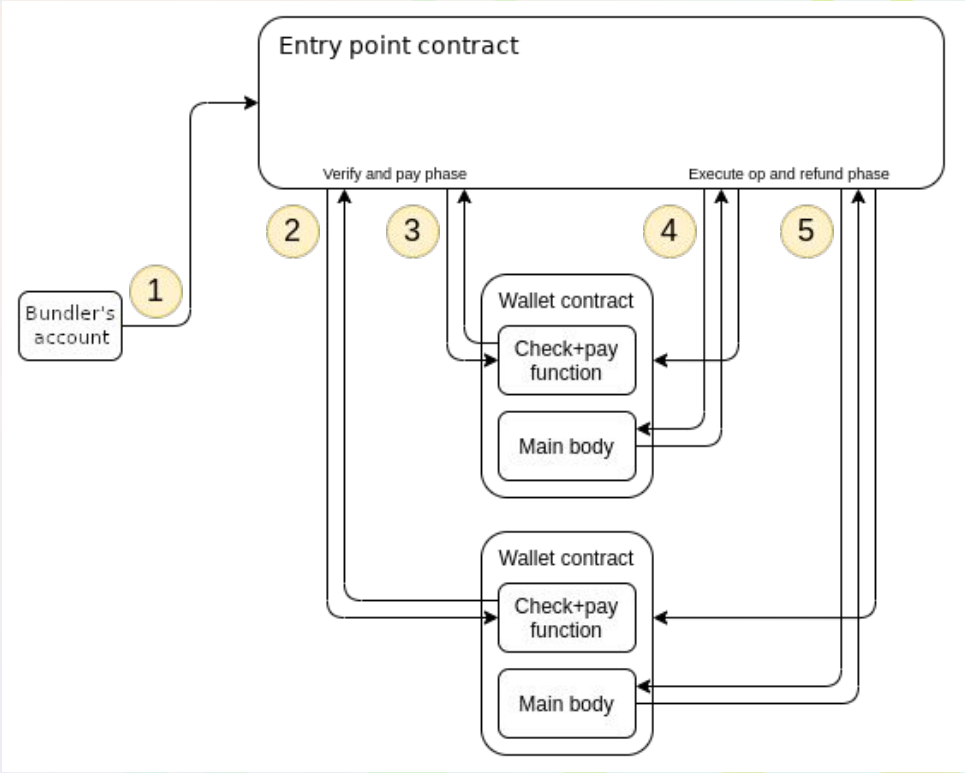
3. Bundle transaction is sent to the **EntryPoint** contract. Bundler pays the transaction fee.



EIP 4337

1. For each UserOperation the EntryPoint contract calls the **validateUserOp** method of the target wallet. The wallet validates the operation (signature, nonce, etc) and pays the fee.

2. For each (validated) UserOperation the EntryPoint contract calls an execution method that can interpret the calldata and **execute** the call(s).



What's next for Account Abstraction?

Each of these EIPs (86, 2938, 3074, 4337) delivers some of the features of Account Abstraction.

EIP-4337 comes close, but like smart-contract wallets, 4337 wallets remain second class citizens on a blockchain designed for EOAs.

What we need is to bring EIP-4337 from the application layer to the protocol layer and remove EOAs!

L2 is the perfect opportunity to fix the limitations of Ethereum and bring AA to users

argent

Section 3

Native Account Abstraction on StarkNet (and ZkSync)

The IAccount interface*

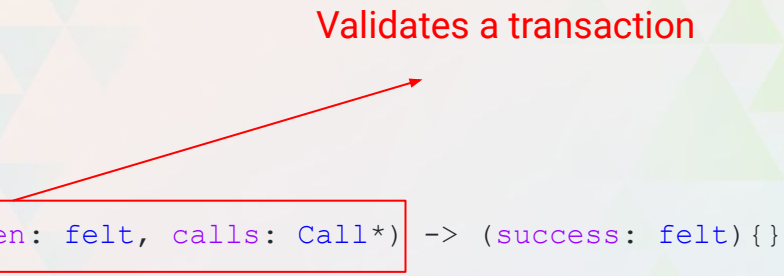
```
namespace IAccount {
    struct Call {
        to: felt,
        selector: felt,
        data_len: felt,
        data: felt*
    }

    func __validate__(calls_len: felt, calls: Call*) -> (success: felt){}

    func __validate_declare__(class_hash: felt){}

    func __validate_deploy__(class_hash: felt, ctr_arg_len: felt, ctr_args: felt*, salt: felt){}

    func __execute__(calls_len: felt, calls: Call*) -> (response_len: felt, response: felt){}

    func isValidSignature(hash: felt, signature_len: felt, signature: felt*) -> (success: felt){}

}
```

* StarkNet

argent

The IAccount interface

```
namespace IAccount {
    struct Call {
        to: felt,
        selector: felt,
        data_len: felt,
        data: felt*
    }

    func __validate__(calls_len: felt, calls: Call*) -> (success: felt){}

    func __validate_declare__(class_hash: felt){}

    func __validate_deploy__(class_hash: felt, ctr_arg_len: felt, ctr_args: felt*, salt: felt){}

    func __execute__(calls_len: felt, calls: Call*) -> (response_len: felt, response: felt){}

    func isValidSignature(hash: felt, signature_len: felt, signature: felt*) -> (success: felt){}
}
```



argent

The IAccount interface

```
namespace IAccount {
    struct Call {
        to: felt,
        selector: felt,
        data_len: felt,
        data: felt*
    }

    func __validate__(calls_len: felt, calls: Call*) -> (success: felt){}

    func __validate_declare__(class_hash: felt){}

    func __validate_deploy__(class_hash: felt, ctr_arg_len: felt, ctr_args: felt*, salt: felt){}

    func __execute__(calls_len: felt, calls: Call*) -> (response_len: felt, response: felt){}

    func isValidSignature(hash: felt, signature_len: felt, signature: felt*) -> (success: felt){}
}
```

Executes a transaction

The IAccount interface

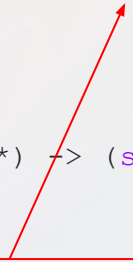
```
namespace IAccount {
    struct Call {
        to: felt,
        selector: felt,
        data_len: felt,
        data: felt*
    }

    func __validate__(calls_len: felt, calls: Call*) -> (success: felt){}

    func __validate_declare__(class_hash: felt){}

    func __validate_deploy__(class_hash: felt, ctr_arg_len: felt, ctr_args: felt*, salt: felt){}

    func __execute__(calls_len: felt, calls: Call*) -> (response_len: felt, response: felt){}

    func isValidSignature(hash: felt, signature_len: felt, signature: felt*) -> (success: felt){}
}
```

Yeah multicalls!!!

The IAccount interface

```
namespace IAccount {
    struct Call {
        to: felt,
        selector: felt,
        data_len: felt,
        data: felt*
    }

    func __validate__(calls_len: felt, calls: Call*) -> (success: felt){}

    func __validate_declare__(class_hash: felt){}

    func __validate_deploy__(class_hash: felt, ctr_arg_len: felt, ctr_args: felt*, salt: felt){}

    func __execute__(calls_len: felt, calls: Call*) -> (response_len: felt, response: felt){}

    func isValidSignature(hash: felt, signature_len: felt, signature: felt*) -> (success: felt){}
}
```

The account can pay for its own deployment



argent

The IAccount interface

```
namespace IAccount {
    struct Call {
        to: felt,
        selector: felt,
        data_len: felt,
        data: felt*
    }

    func __validate__(calls_len: felt, calls: Call*) -> (success: felt){}

    func __validate_declare__(class_hash: felt){}

    func __validate_deploy__(class_hash: felt, ctr_arg_len: felt, ctr_args: felt*, salt: felt){}

    func __execute__(calls_len: felt, calls: Call*) -> (response_len: felt, response: felt){}

    func isValidSignature(hash: felt, signature_len: felt, signature: felt*) -> (success: felt){}
}
```

Verify off-chain signatures (EIP-1271)

Account Abstraction for devs

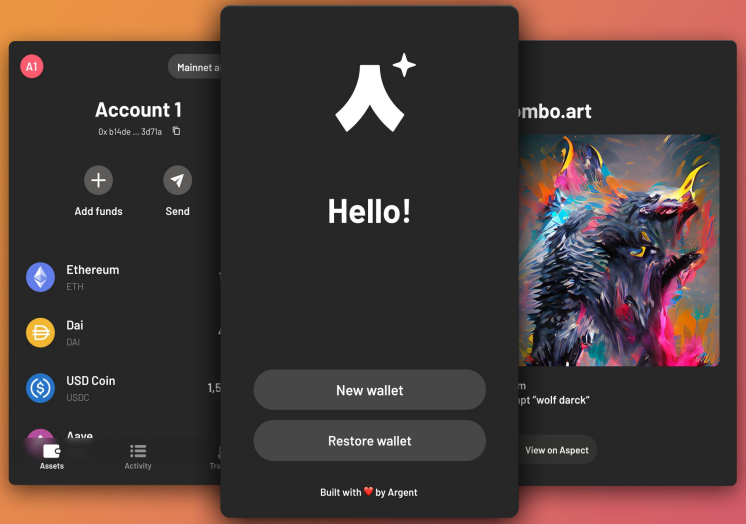
1. Accounts are smart-contracts and must be deployed
2. The address of the Account is computed like a smart-contract (not derived from the Signer)
3. Transactions can have multiple signatures
   → sig = [sig1, sig2, …, sigN]
4. Off-chain signatures (e.g. EIP712) must be validated on-chain by the Account
   → You cannot use ecrecover(m, sig) locally!
   → You must use **account.is_valid_signature(m, sig)**
5. You can (and should!) use multicalls

Account Abstraction for devs

1. Accounts are smart-contracts and must be deployed
2. The address of the Account is computed like [illegible]ntract (not derived from the Signer)
3. Transactions can have [illegible]ures
   → sig = [sig1 [illegible]]
4. Off-chain signa[illegible]es (e.g. EIP712) must be validated on-chain by the Account
   → You cannot use ecrecover(m, sig) locally!
   → You must use **account.is_valid_signature(m, sig)**
5. You can (and should!) use multicalls

All you need to remember!

argent



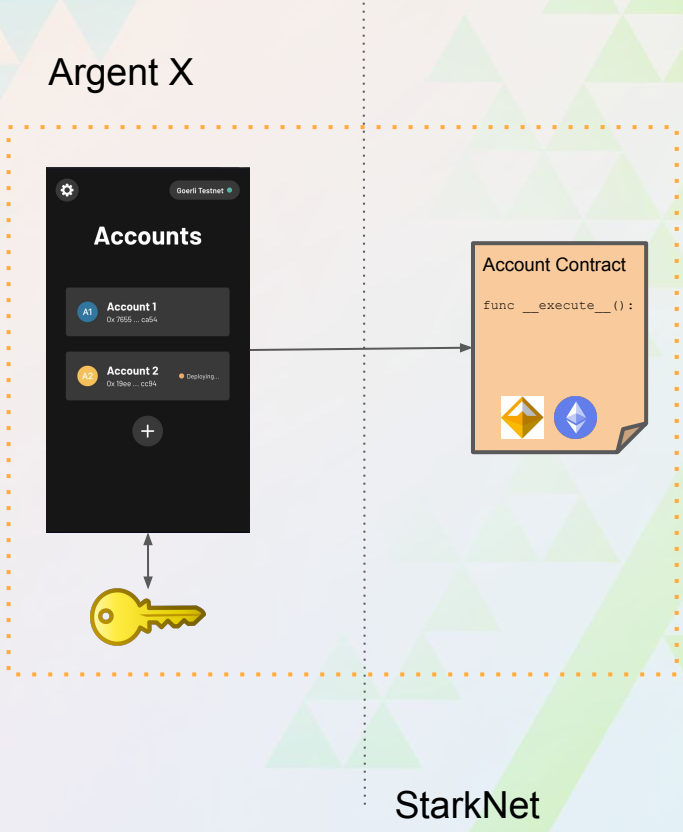
Argent X, first Wallet with native Account Abstraction!

- First Wallet on StarkNet
- Browser extension (Chrome/Firefox)
- Multi-account / multi StarkNet network
- Send and receive tokens/NFTs
- Interact with dapps
- 100% Open source

250k+ downloads

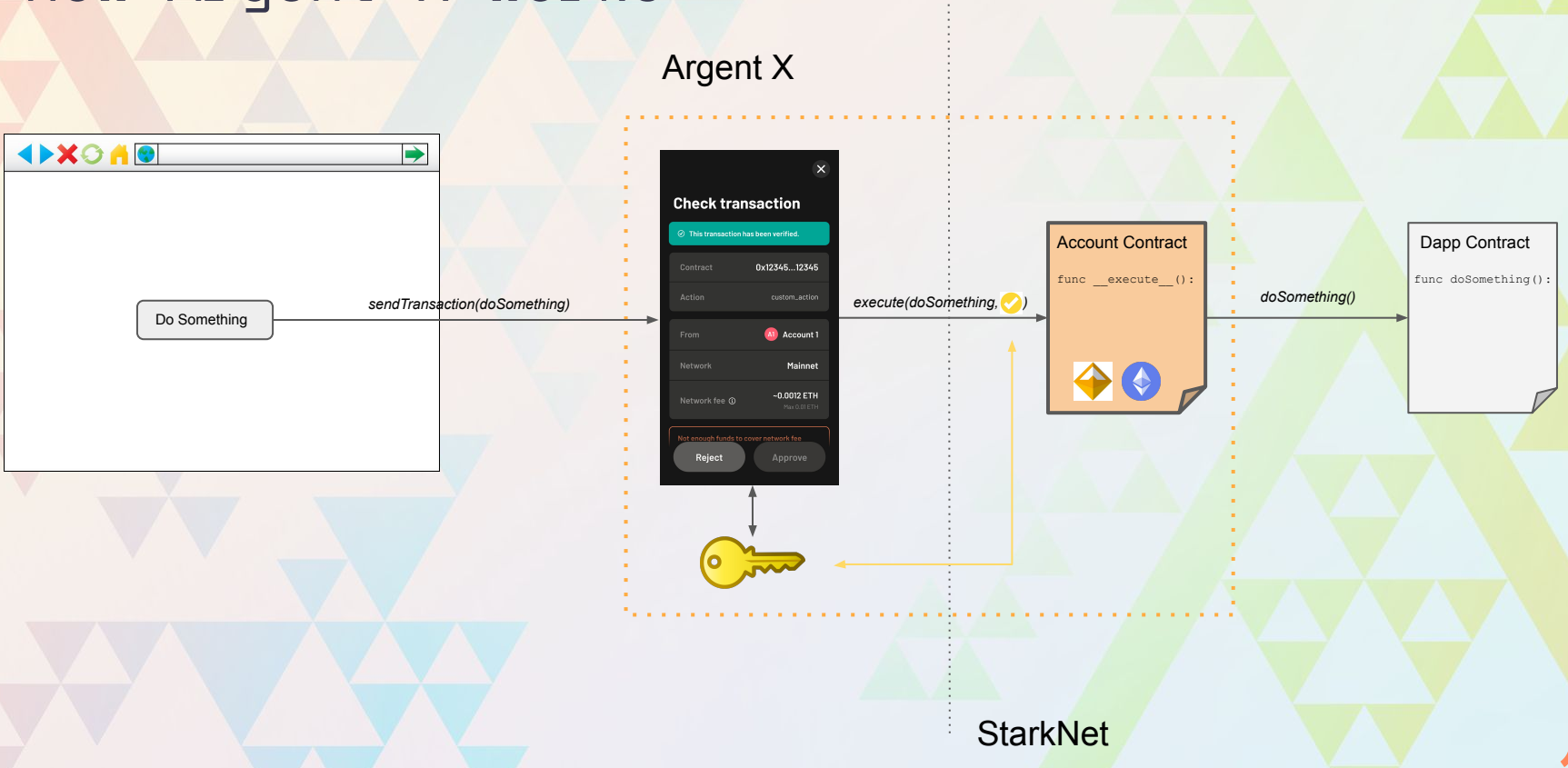
>90% of all StarkNet funds

How Argent X works



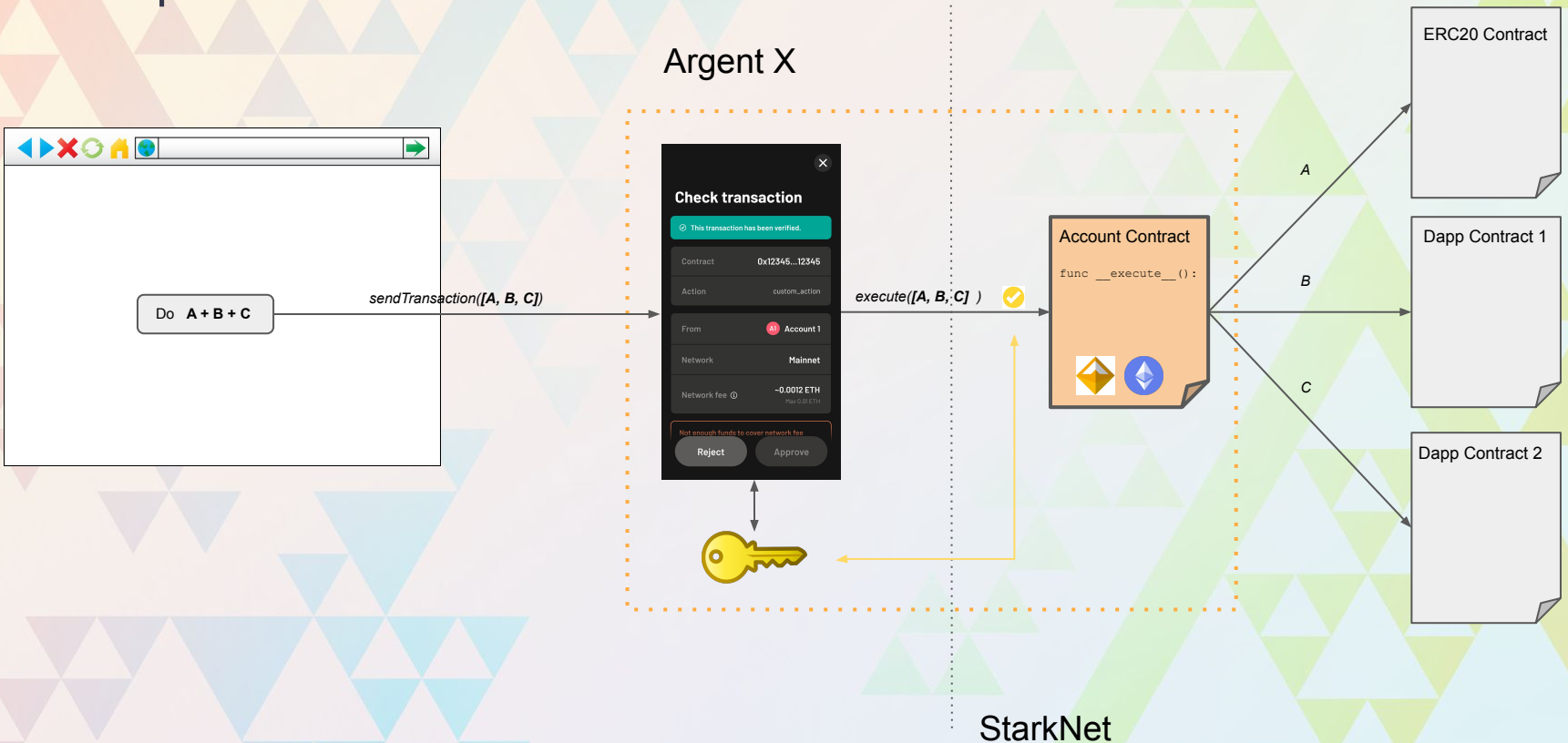
argent

How Argent X works



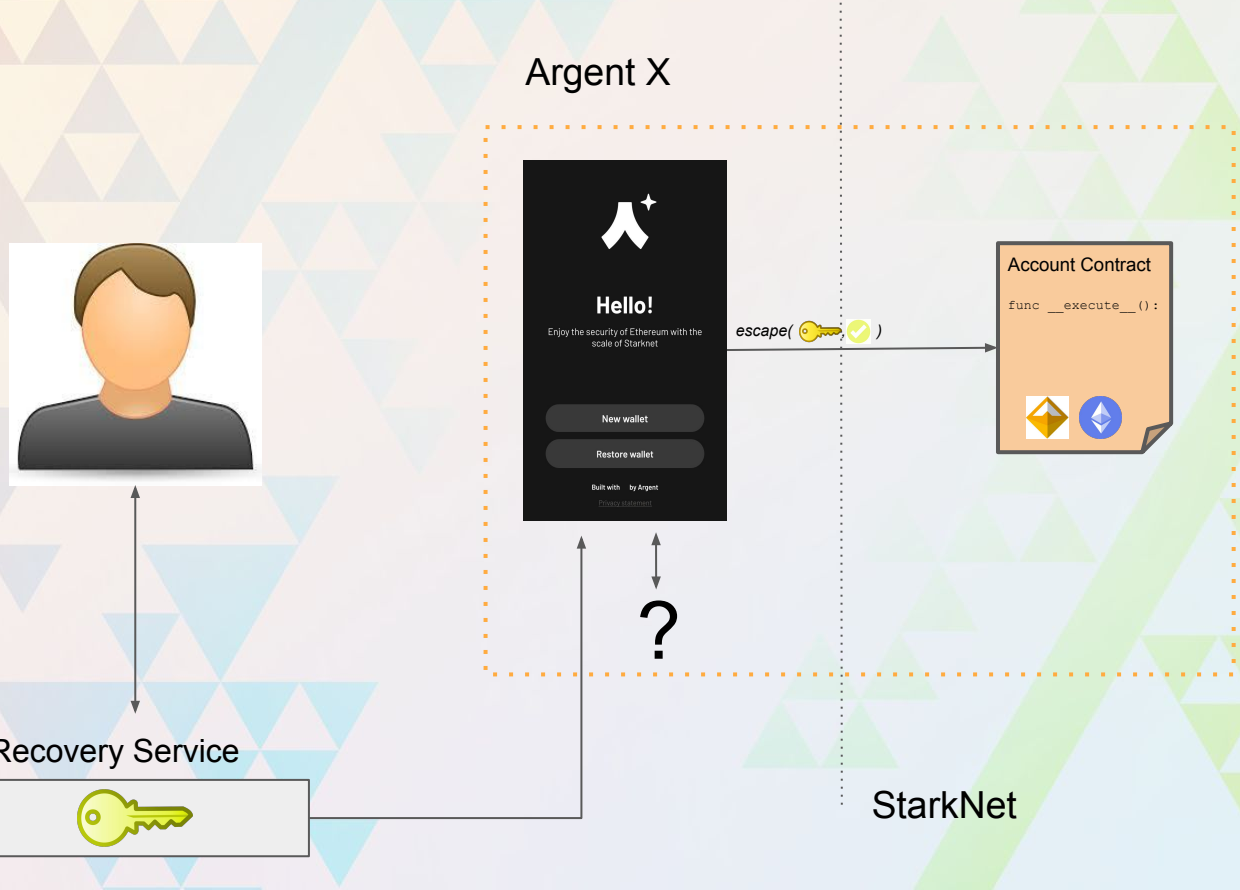
argent

Unique Feature: Multicalls



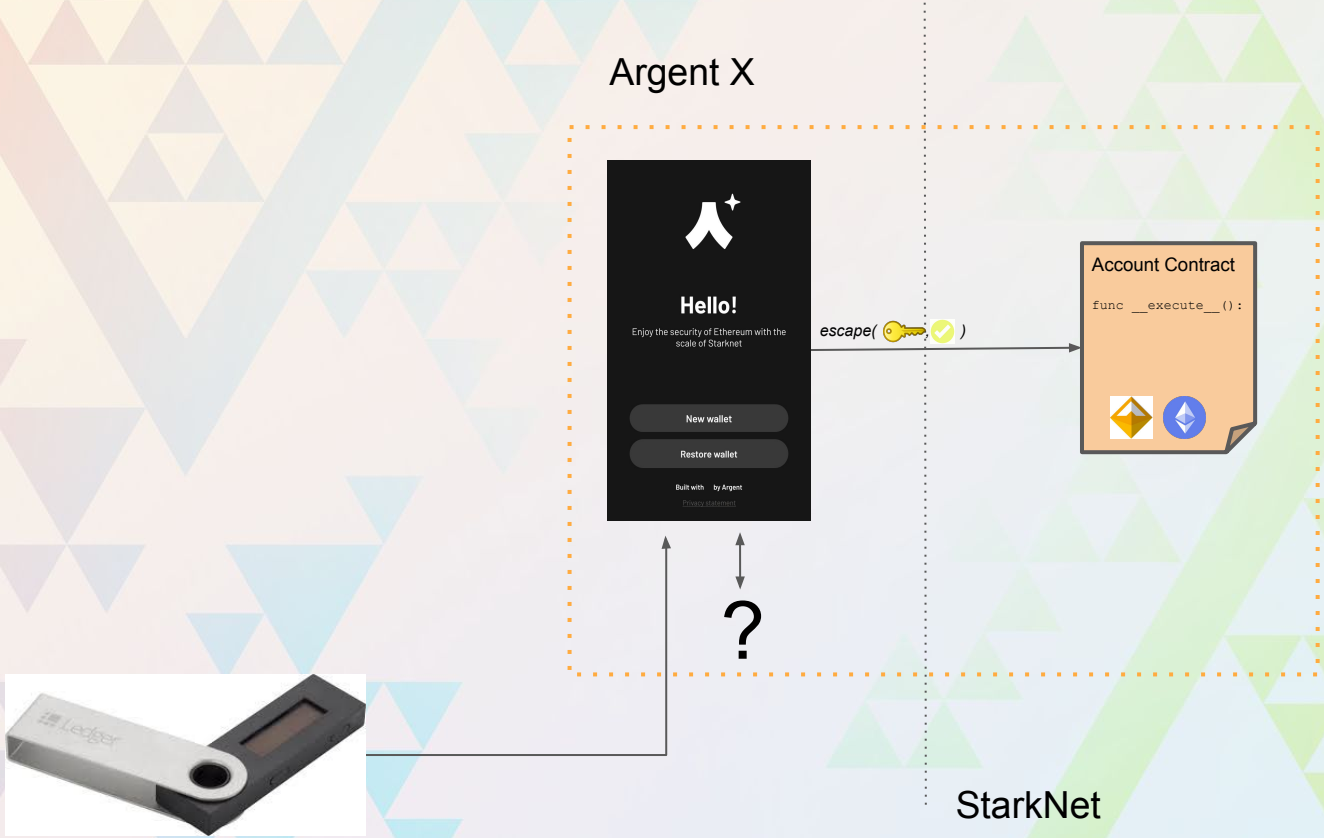
argent

Unique Feature: (Social) Recovery



argent

Unique Feature: (Social) Recovery



argent

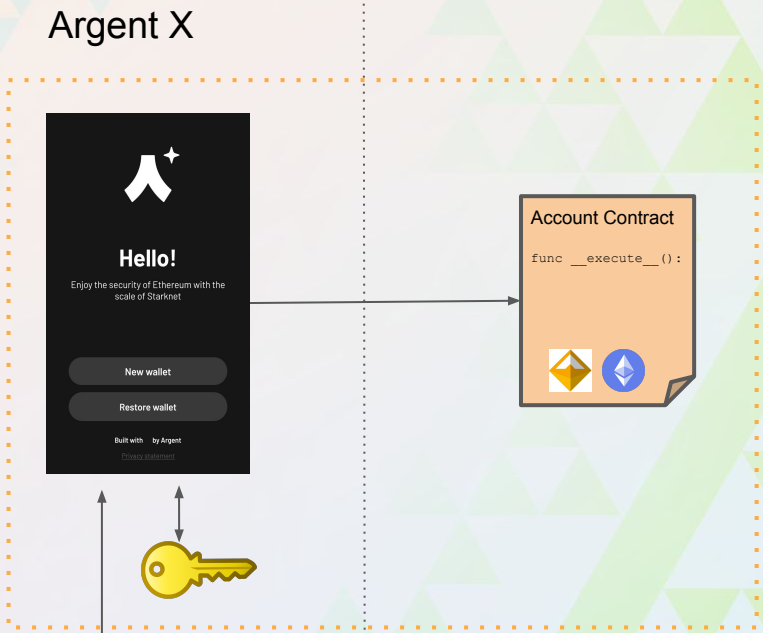
Unique Feature: (Social) Recovery

Time delay of 7 days

Can be canceled

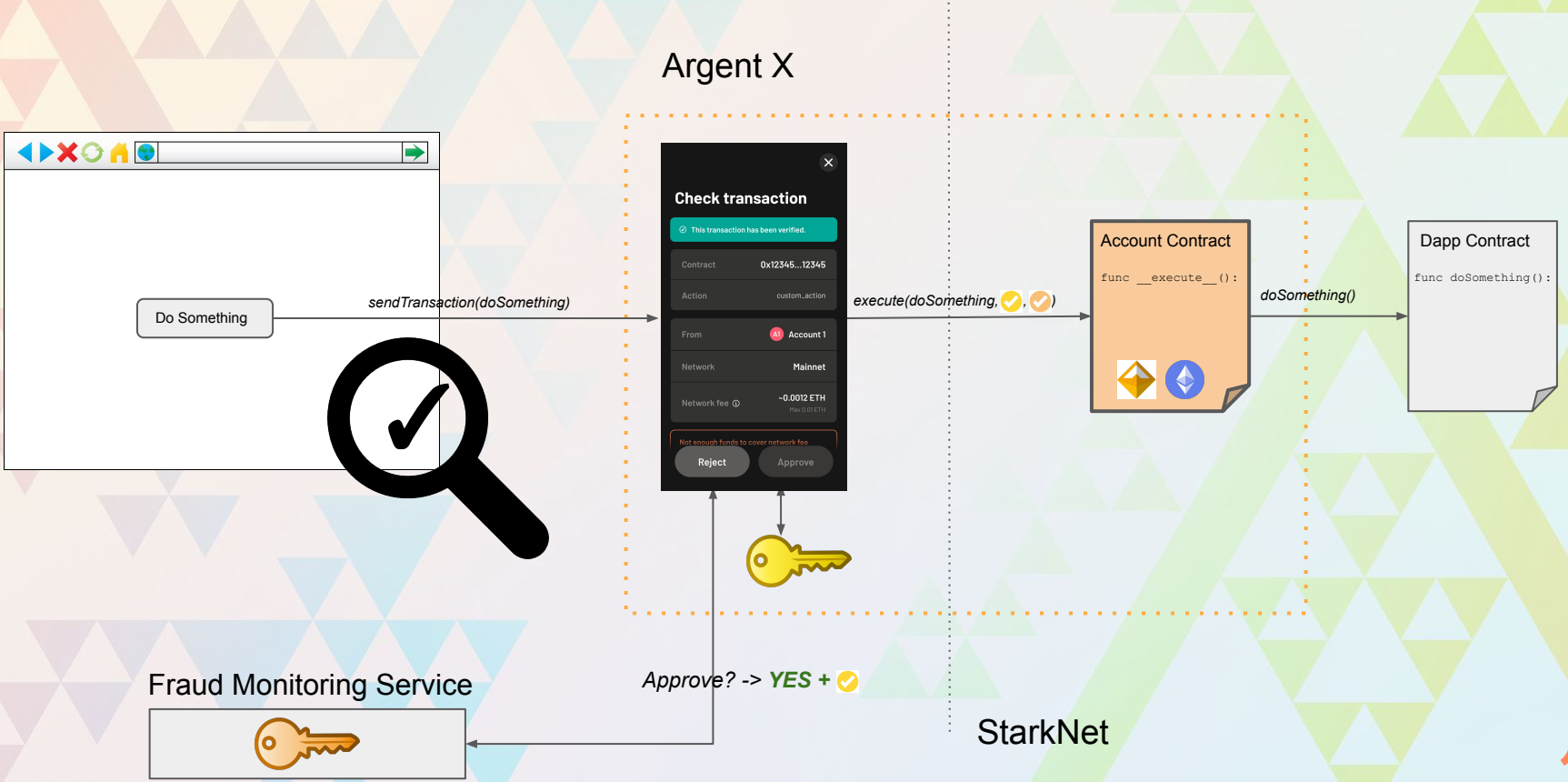
100% non-custodial

No more seed-phrases!



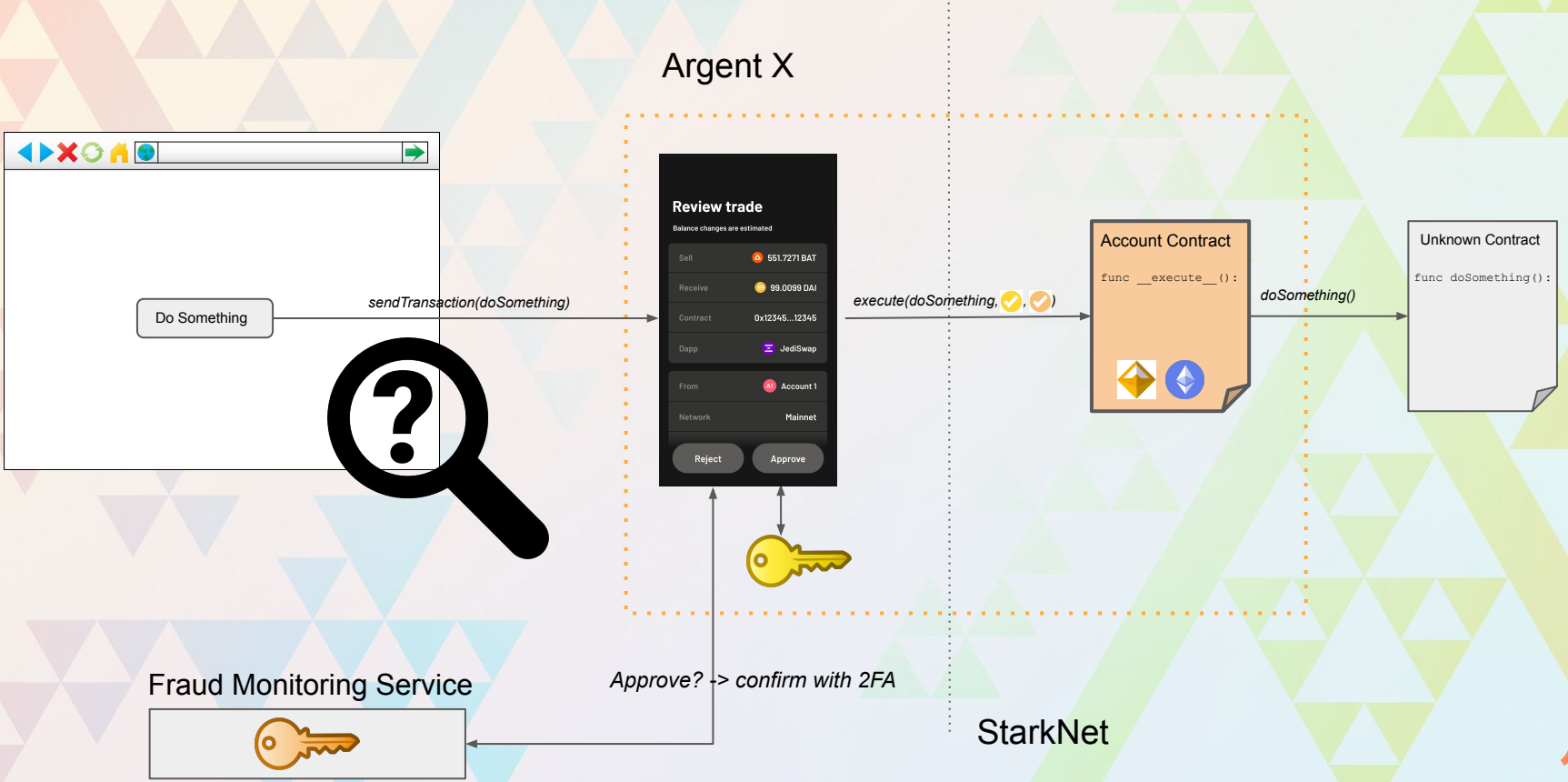
argent

Unique Feature: Fraud Monitoring / 2Fa



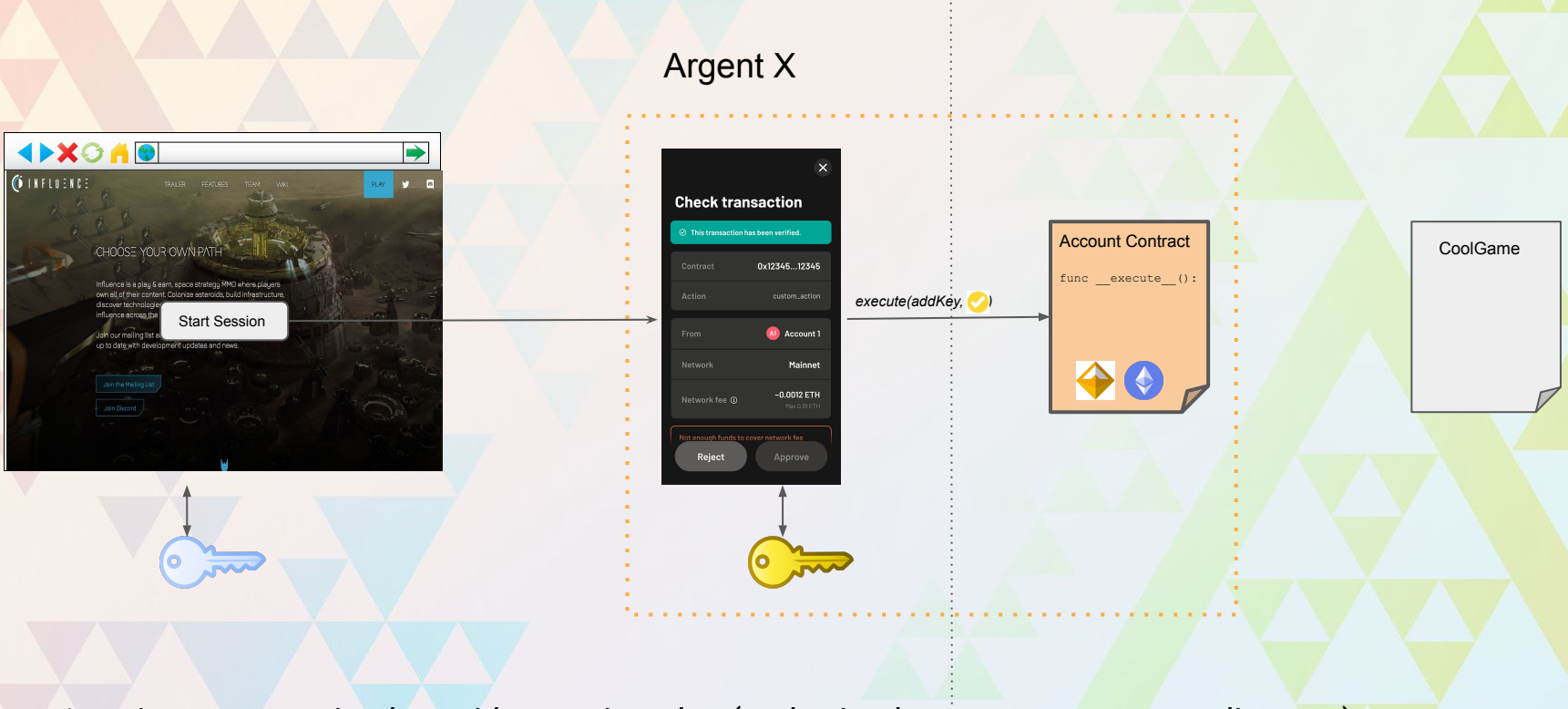
argent

Unique Feature: Fraud Monitoring / 2Fa



argent

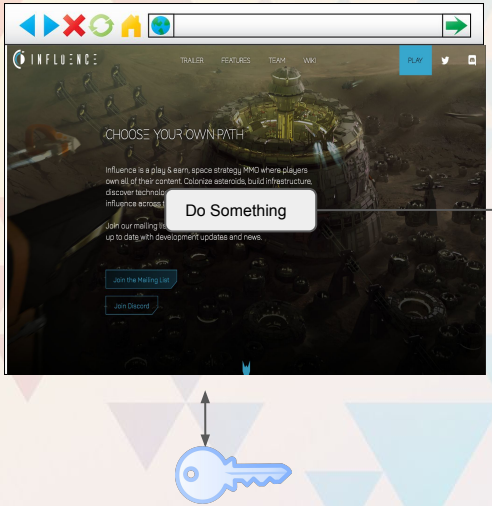
Unique Feature: Session keys for games



1. Approve session key with security rules (authorised contracts, max spending, etc)

argent

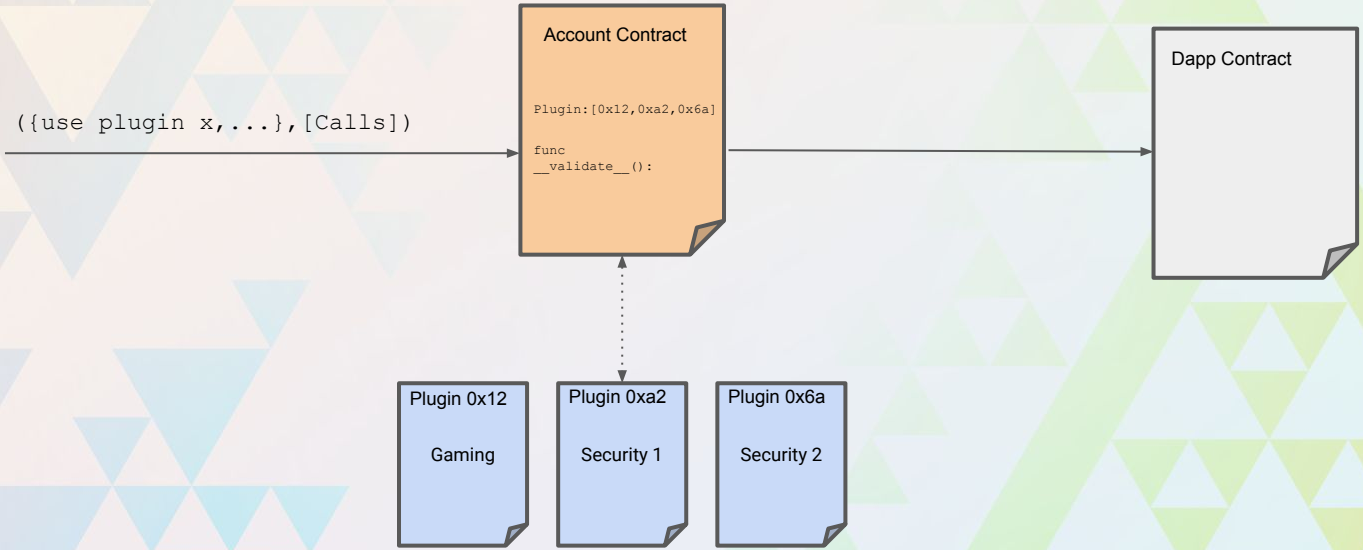
Unique Feature: Session keys for games



2. Execute game actions without the need to approve on Argent X

argent

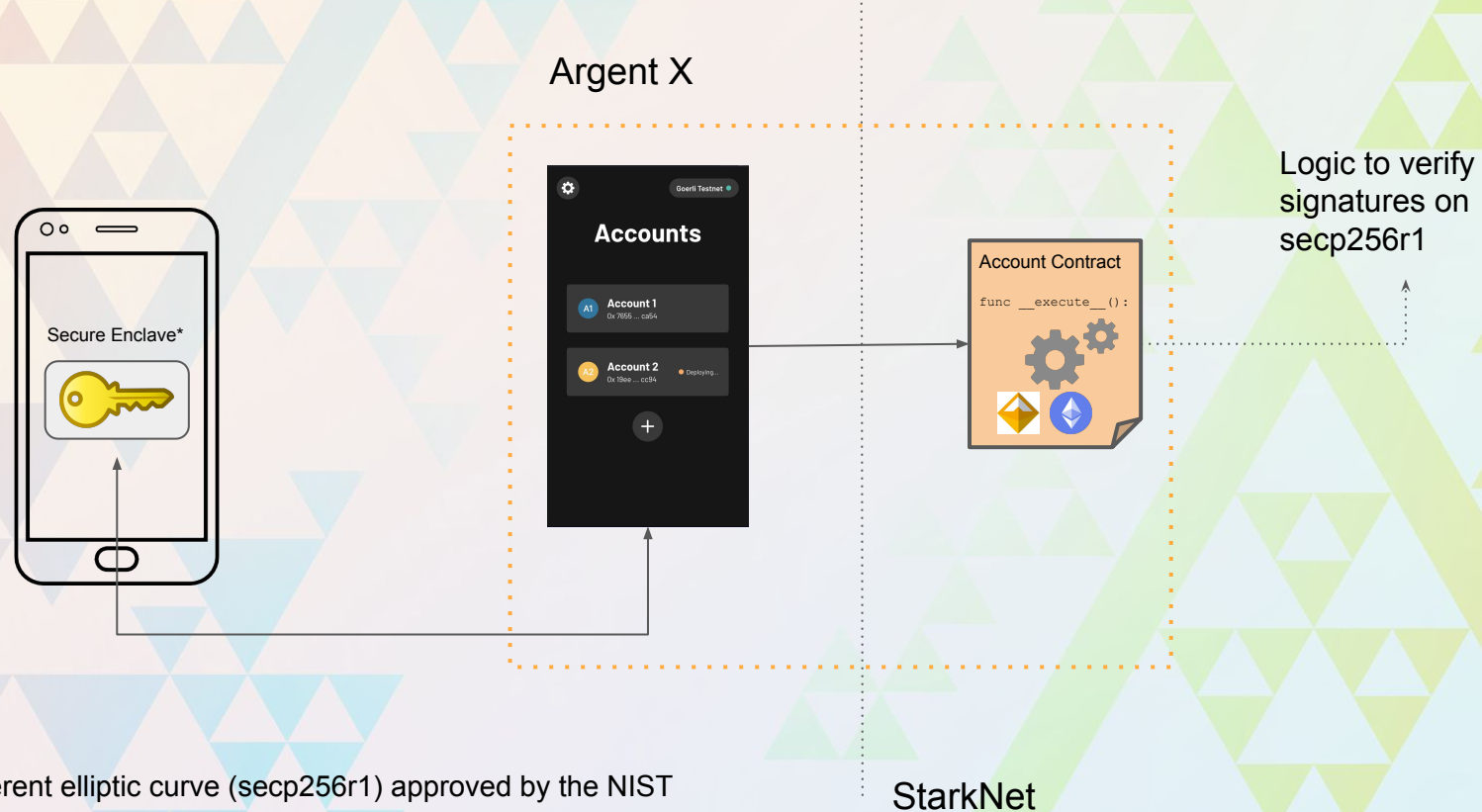
Unique Feature: Plugins*



*Collaboration with Ledger and Cartridge

argent

Unique Feature: Use the Secure Enclave



* Uses a different elliptic curve (secp256r1) approved by the NIST

argent

Account Abstraction is needed to scale the UX of self-custody. Let's make it happen!

Thank you!

Julien Niset

co-founder@Argent

julien@argent.xyz

@jniset

Section 1 details with an image. Enter title here.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Section 4

Section 4 title here.

Section 4 details with a main point. Enter title here.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Enter your main point / statement here.

Here’s the timeline.

Event 1

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam.

Event 2

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam.

Event 3

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam.

Account Abstraction!

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

- Sollicitudin
- Consectetur
  - Condimentum
    - **Magna**
    - **Ligula**

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

- Sollicitudin
- Consectetur
  - Condimentum
    - **Magna**
    - **Ligula**

Section 2 details with an image. Enter title here.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.