SecHack365

FORMIX

あらゆる**分散ストレージ**の上で、
ユーザー主権のまま秘密鍵を管理・共有できる、
ベンダーフリーかつ**検証可能な**鍵管理プロトコル

開発駆動コース 仲山ゼミ 樅山翔大（もみやましょうだい）

#crypto #blockchain #web3 #secret-as-a-service #Rust #Arweave #AO

世界が不安定になり、情報を送る行為そのものがリスクとなる時代へ

FORMIX とは

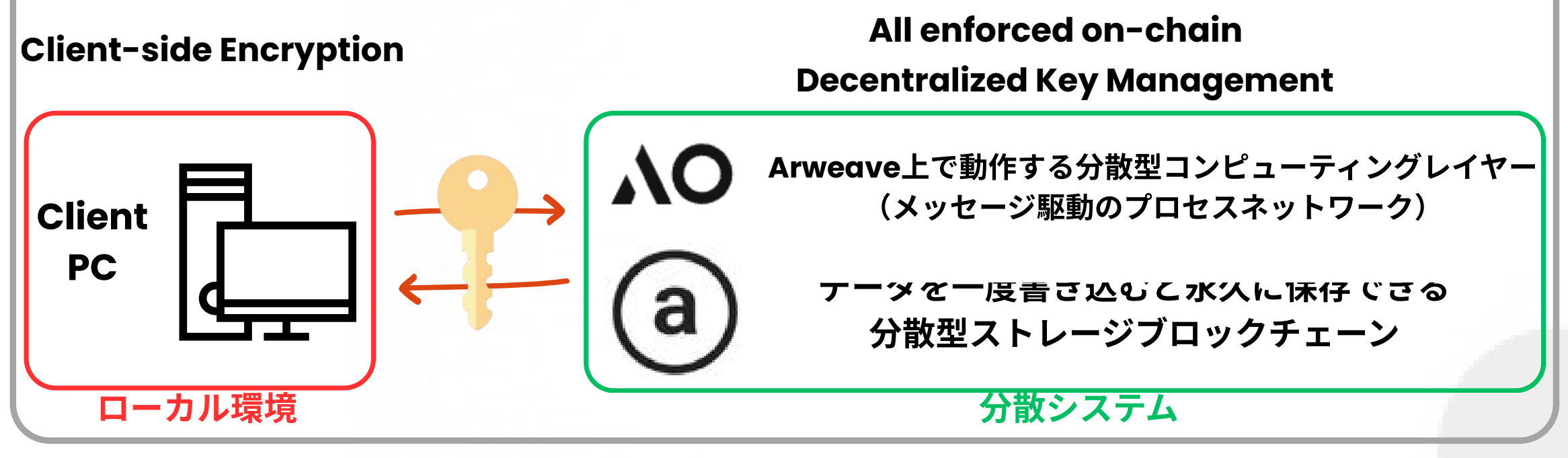
ベンダーを介さない秘密鍵の管理

決定論的かつ検証可能な秘密鍵の共有

WHAT	Arweave と AO Network 上に構築された ベンダーに依存しない分散型の秘密鍵管理・共有プロトコル
LOOK	コンセンサスがストレージレイヤに準拠したコントラクトを アクセス制御におけるプロキシとして機能させることで、 決定論的かつ検証可能な秘密鍵の共有を実現し、 集中型鍵管理のリスクから「信頼」ではなく「構造」で守る。
FUTURE	プロバイダーを介さず、「渡したい人」と「受け取りたい人」の二者間で 安全に暗号化データをやりとりを実現

分散型の鍵管理とは

Decentralized Key Management（分散型鍵管理）は、
秘密鍵の生成・保管・利用・共有といったライフサイクル管理を、
単一のサーバや事業者に集中させず、
複数のノードや主体に分散させる仕組みのこと


Client-side Encryption
All enforced on-chain
Decentralized Key Management
Client PC
ローカル環境
分散システム

AO：Arweave上で動作する分散型コンピューティングレイヤー（メッセージ駆動のプロセスネットワーク）

Arweave：データを一度書き込むと永久に保存できる分散型ストレージブロックチェーン

セキュリティへの貢献と FORMIX の特徴

検閲が構造上可能という問題

Lavabit事件（2013）

米国の暗号化メールサービス Lavabit は、
政府からTLS秘密鍵（マスターキー）の
提出を要求されたが、
鍵を提出すれば特定個人ではなく
全ユーザー通信が傍受可能
創業者はこれを拒否し、
サービス自体を停止

ベンダーが秘密鍵を保持している限り
法令・裁判所命令により
全体を検閲・監視可能な状態が常に内包
→ 善意・悪意以前に **構造上可能**

検閲が「構造上」可能であることが**社会的**リスク

攻撃の標的が一点に定まる問題

LastPass侵害（2022–2023）

パスワード管理サービス LastPass が侵害
暗号化ボルトそのものは守られていたが
鍵素材・メタデータ・バックアップが集中管理
オフライン総当たり・長期攻撃が可能に

「ベンダー鍵管理」という設計により
内部不正 / 侵害時に
攻撃価値が極端に高い一点が生まれる
→ **SPOF**

攻撃の標的が定まり得ることが**技術的**リスク

		秘密鍵	暗号文
SecHack365で開発したプロダクト	FORMIX	自己管理	自己管理
分散型鍵管理の先行事例	NuCypher Lit Protocol	自己管理	委託管理
ベンダーによる鍵管理	aws Key Management Service (KMS)	委託管理	委託管理

自己主権性 ↑ 第三者主権性

FORMIX の使い方

学習コストゼロで、今日から検証可能な分散型の秘密管理を実現

- 簡単にコントラクトをデプロイ

```
$ yarn deploy --wallet <path-to-wallet.json>
```

- シンプルな4つの基本関数 ビルダーパターンのapiを提供

```
let client = FormixClient::init("./wallet.json")
let (owner_sk, owner_pk) = client.generate_keypair()?;
```

```
let share_result = client.share()
    .secret(b"共有する機密データ")
    .threshold(3)
    .total_shares(5)
    .owner_key(&owner_sk)
    .requester_key(&requester_pk)
    .metadata("description", "サンプルの秘密")
    .execute()?;
```

```
let recovered = client.recover()
    .secret_id(&share_result.secret_id)
    .requester_key(&requester_sk)
    .owner_public_key(&owner_pk)
    .execute()?;
```

実装で苦労したポイント

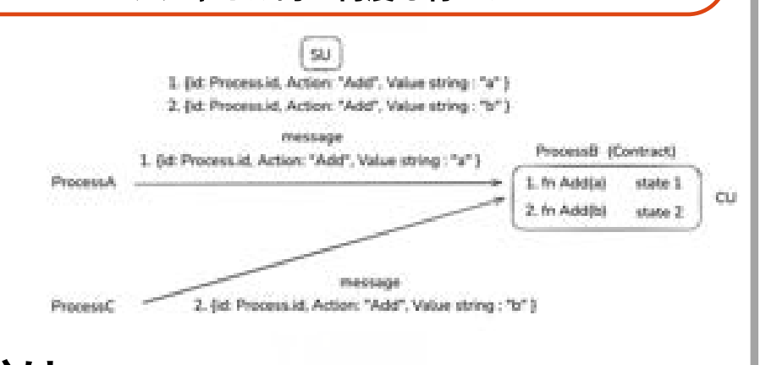
アーキテクチャの設計

秘密鍵の共有はブロックチェーン、保存・管理はAWSといったハイブリッドなニーズも想定。

セキュリティ、疎結合を意識
拡張性を持った秘密管理サービスにするための設計をアジャイル的に何度も行った

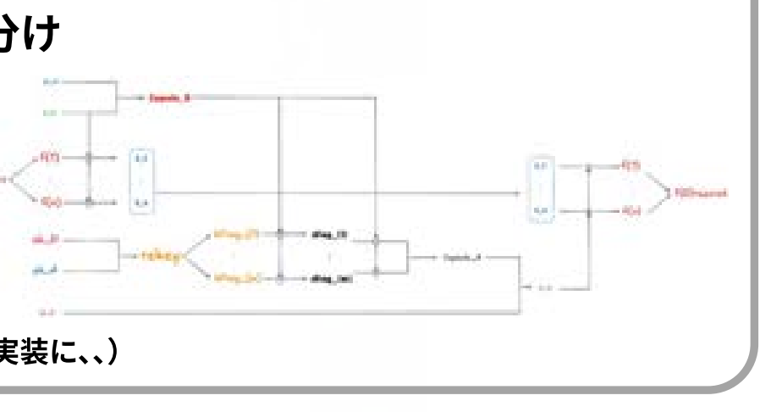
AOのステートレス環境での暗号操作

EVM系のようにstorage変数がなく、実行を順序通りに再現すると最新のステートが決定論的に作られるというAO独自の仕様をベースに、再暗号化処理を実行し続けられるKVSをコントラクトのStateとして実装した


SU
1. {id: Process.id, Action: "Add", Value string : "a" }
2. {id: Process.id, Action: "Add", Value string : "b" }
message
ProcessA
ProcessB (Contract)
1. fn Add(a) state 1
2. fn Add(b) state 2
CU
ProcessC

何がPublicで良くて何がPrivateでないといけないか、クライアントとブロックチェーンの秘密の扱いの棲み分け

FORMIXでは多くのsecretに関連するコンポーネントがあるため、各コンポーネントの性質を理解し、アーキテクチャに直で影響するstateがpublicなブロックチェーンにどこまで処理を切り出して良いかの線引きが試行錯誤だった



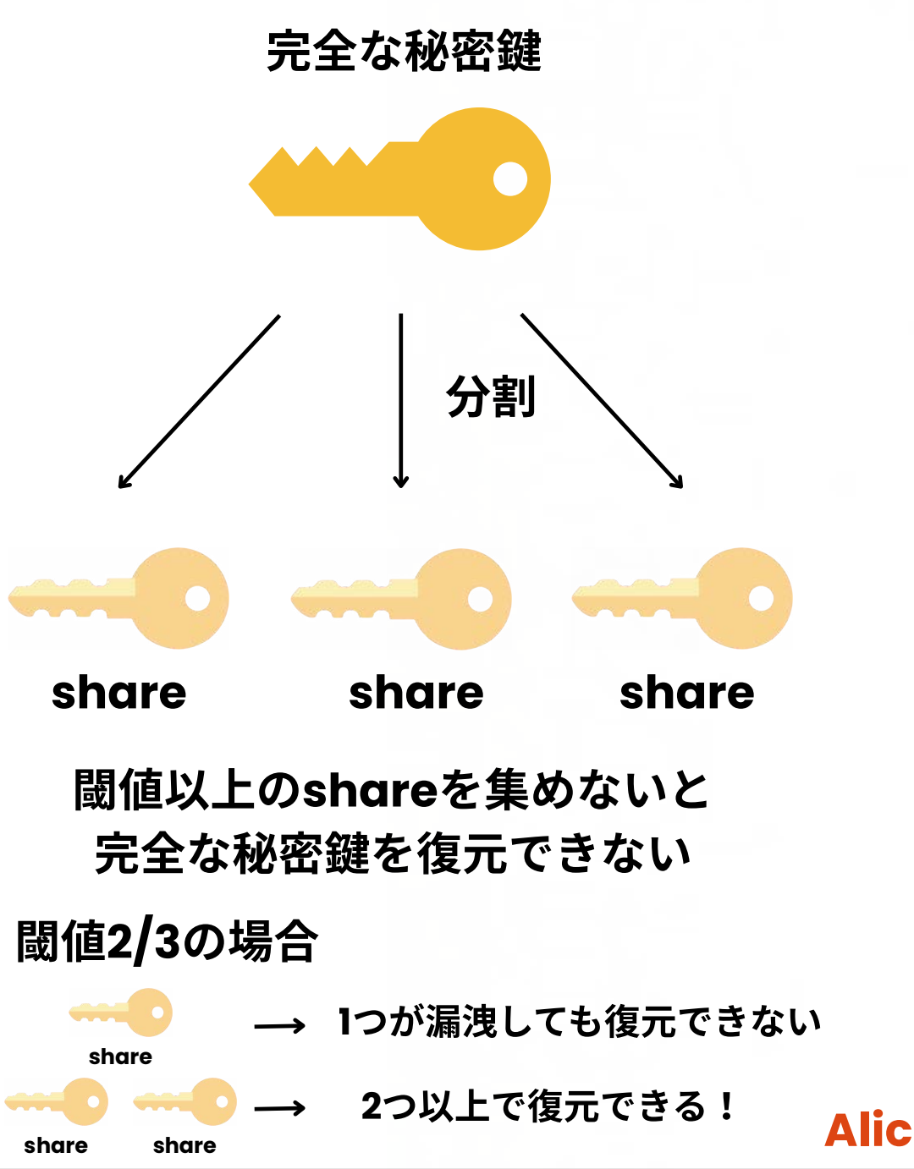
AO ProcessのRust実装の情報がとても少なかった、、、
（doc, ライブラリ等がほぼなかったのでArweaveやAOとの接続はhttpの自前実装に、、）

暗号技術で社会的・技術的リスクの脅威から機密情報を守る

基盤となる暗号技術

シャミアの秘密分散法

「Adi Shamir (1979)」

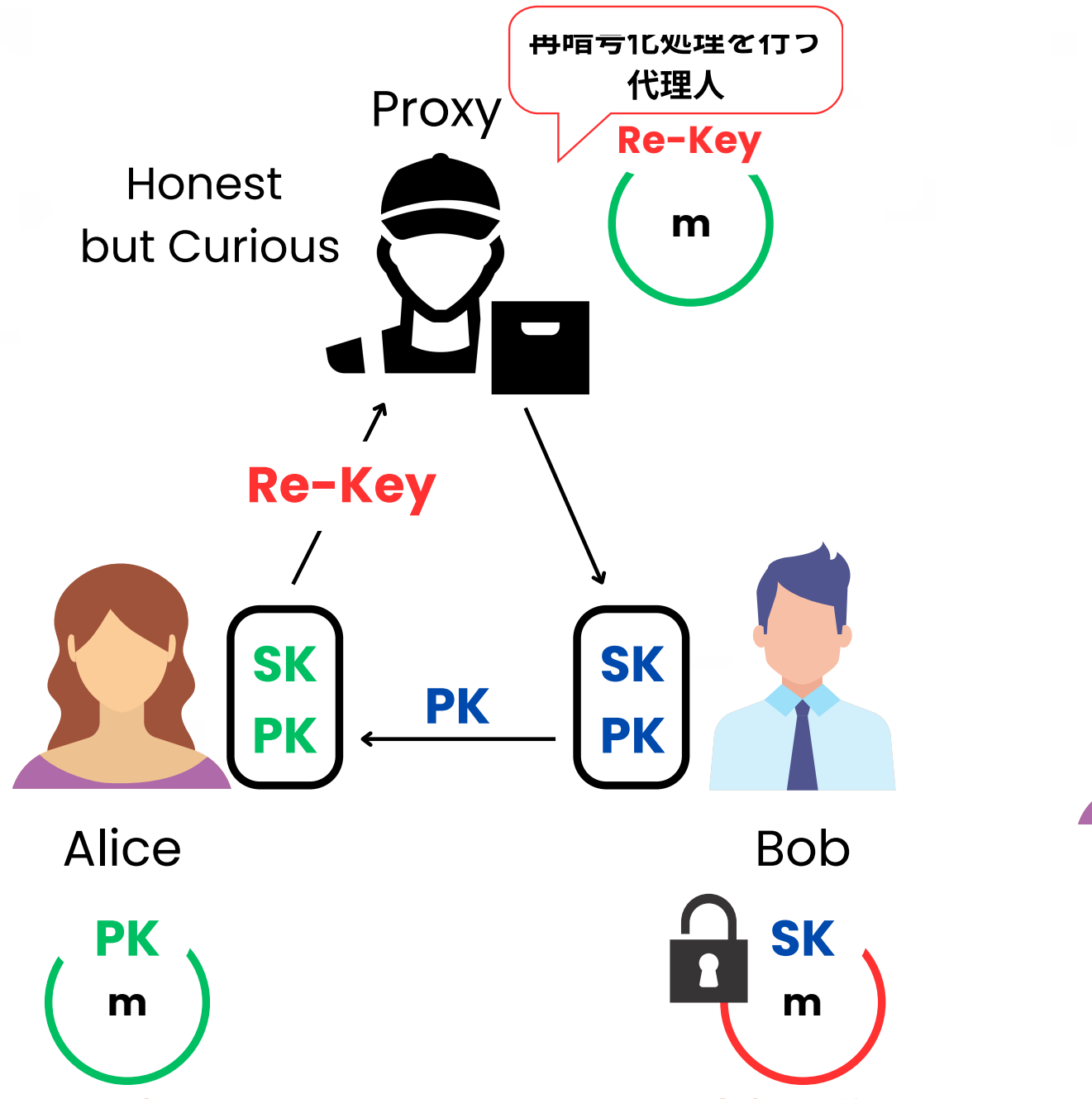

完全な秘密鍵
分割
share
share
share
share
share
share

閾値以上のshareを集めないと
完全な秘密鍵を復元できない

閾値2/3の場合
share → 1つが漏洩しても復元できない
share share → 2つ以上で復元できる！

プロキシ再暗号化

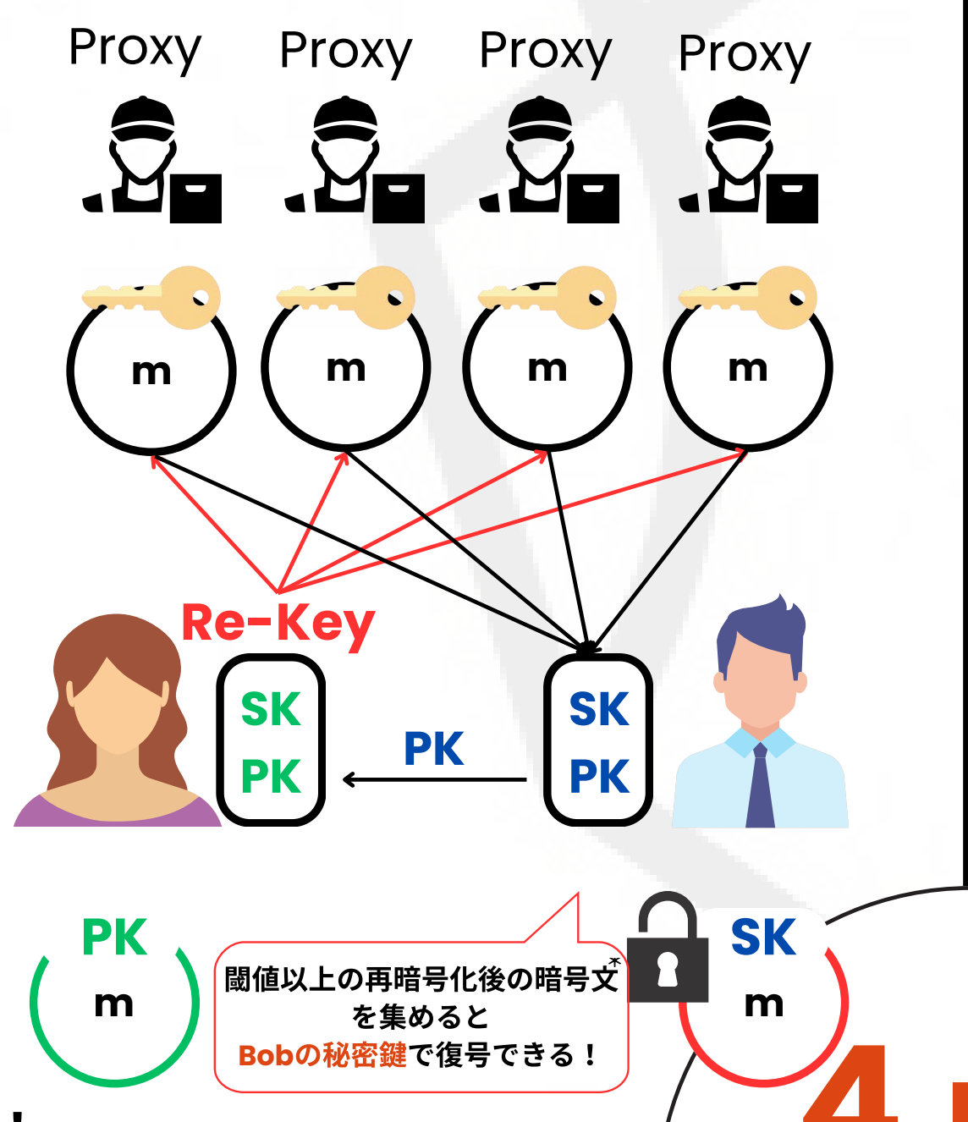
「Giuseppe Atenieseほか (2006)」


Proxy
再暗号化処理を行う代理人
Re-Key
m
Honest
but Curious
Re-Key
SK
PK
PK
SK
PK
Alice
Bob
PK
m
SK
m

Aliceの公開鍵で暗号化 ⟶ Bobの秘密鍵で復号できる！

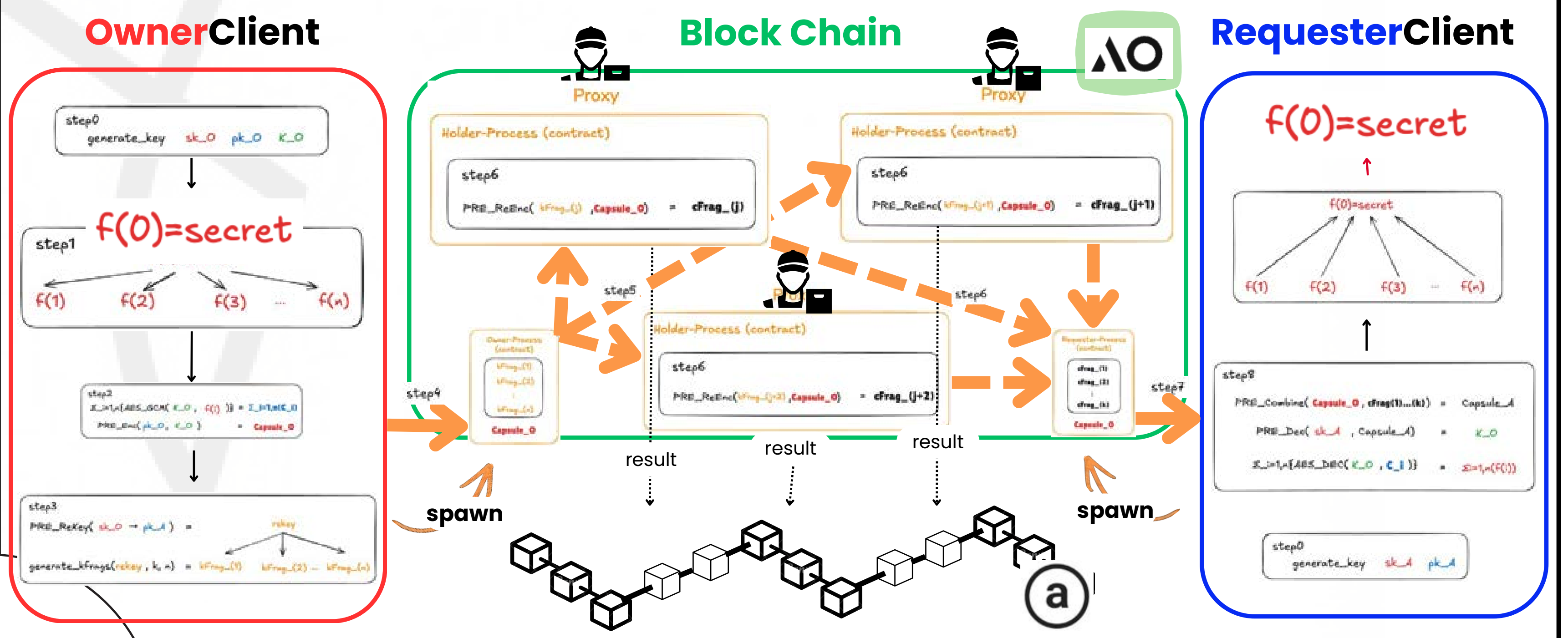
閾値プロキシ再暗号化

「Nuñez, D. et al, Umbral (2018)」


Proxy
Proxy
Proxy
Proxy
m
m
m
m
Re-Key
SK
PK
PK
SK
PK
PK
m
SK
m

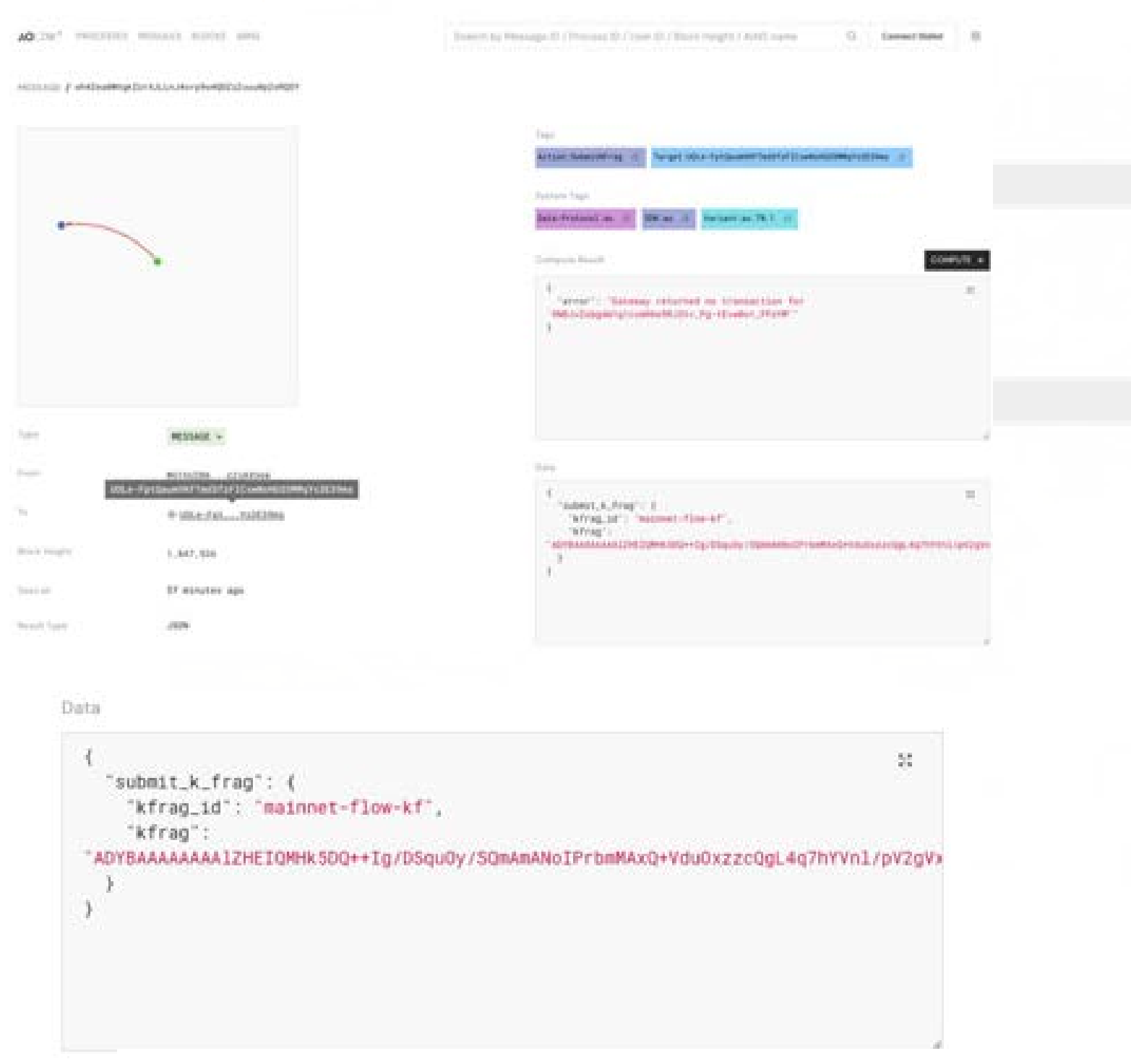
閾値以上の再暗号化後の暗号文を集めると
Bobの秘密鍵で復号できる！

E2EでClientとBlockChainのみで完結


OwnerClient
step0 generate_key sk_O pk_O K_O
step1 f(0)=secret
f(1) f(2) f(3) … f(n)
step2
PRE_Enc(pk_O , K_O) = Capsule_O
step3 PRE_ReKey(sk_O → pk_A) = rekey
generate_kfrags(rekey , k, n) = kFrag_(1) kFrag_(2) … kFrag_(n)
Block Chain
Proxy
Proxy
Holder-Process (contract)
step6 PRE_ReEnc(kFrag_(j) , Capsule_O) = cFrag_(j)
Holder-Process (contract)
step6 PRE_ReEnc(kFrag_(j+1) , Capsule_O) = cFrag_(j+1)
Holder-Process (contract)
step6 PRE_ReEnc(kFrag_(j+2) , Capsule_O) = cFrag_(j+2)
Owner-Process (contract)
Capsule_O
Requester-Process (contract)
Capsule_O
step4
step5
step6
step7
result
result
result
spawn
spawn
RequesterClient
f(0)=secret
f(1) f(2) f(3) … f(n)
step8
PRE_Combine(Capsule_O , cFrag(1)...(k)) = Capsule_A
PRE_Dec(sk_A , Capsule_A) = K_O
step0 generate_key sk_A pk_A

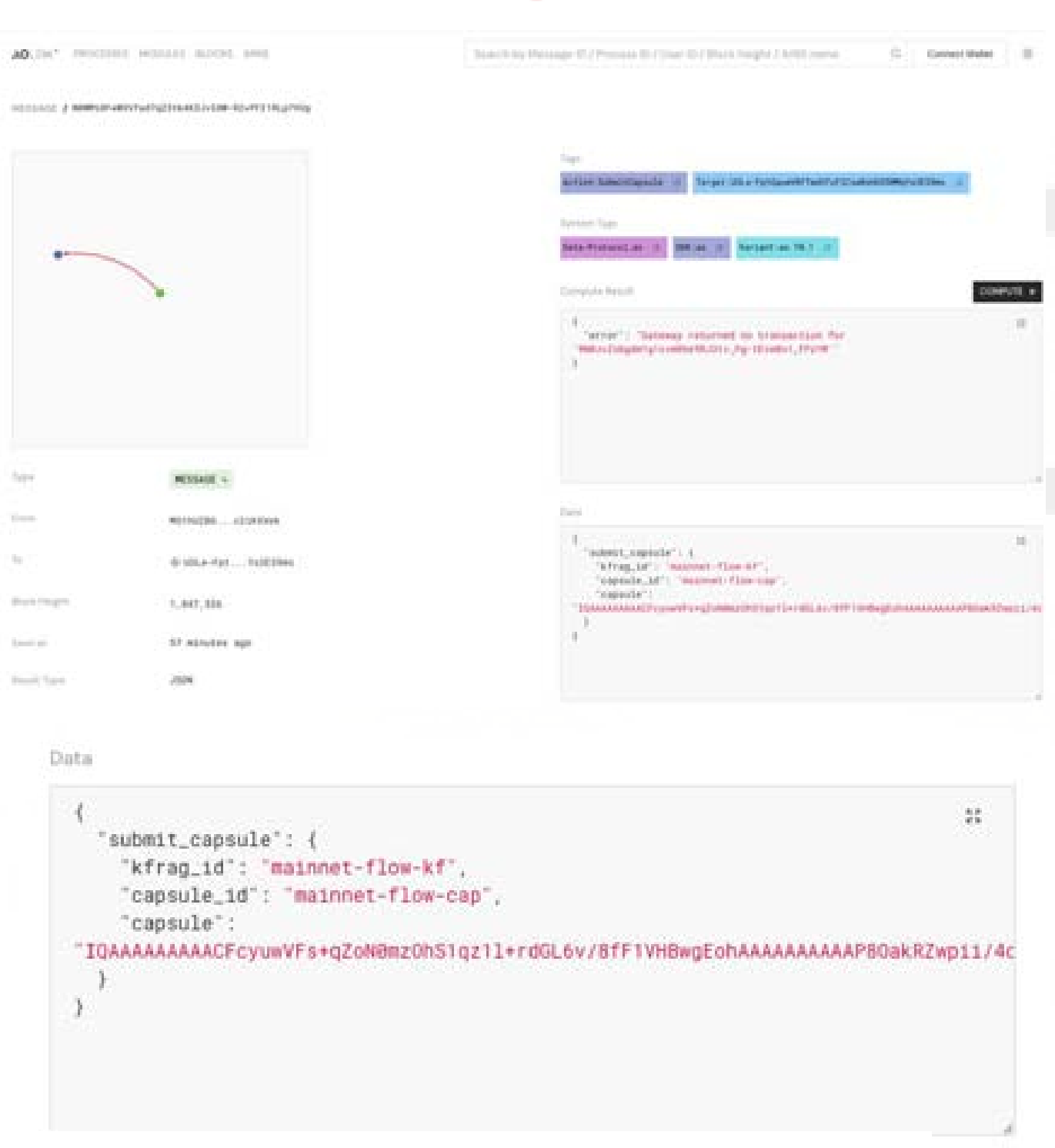
4 Key Features

kFrag を Owner_Processへ移譲



```
{
  "submit_k_frag": {
    "kfrag_id": "mainnet-flow-kf",
    "kfrag":
"ADYBAAAAAAAAlZHEIQMHk5DQ++Ig/DSquOy/SQmAmANoIPrbmMAxQ+VduOxzzcQgL4q7hYVnl/pV2gV
  }
}
```

Holder_Processが Capsule_O を再暗号化



```
{
  "submit_capsule": {
    "kfrag_id": "mainnet-flow-kf",
    "capsule_id": "mainnet-flow-cap",
    "capsule":
"IQAAAAAAAAACFcyuwVFs+qZoN0mzOhS1qz1l+rdGL6v/8fF1VHBwgEohAAAAAAAAAAPB0akRZwpi1/4c
  }
}
```

処理の正当性をいつでもだれでも検証できる

アプローチ①

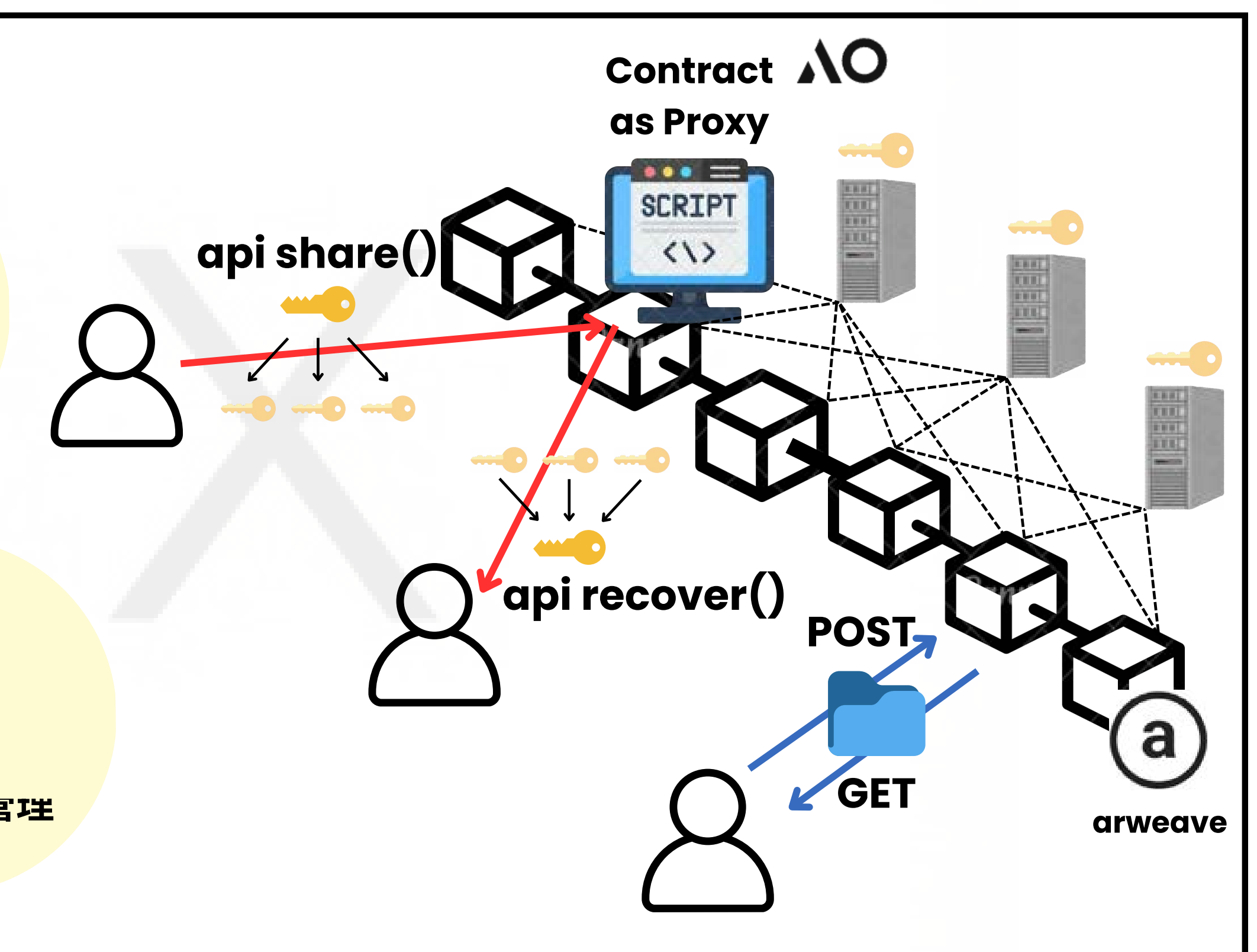
コントラクトを使ったプロキシ再暗号化

第三者の介入なしに決定論的な秘密鍵共有

アプローチ②

AOとArweave上で開発するDapp

鍵とデータを同じチェーンで一体管理


Contract
as Proxy
api share()
SCRIPT
api recover()
POST
GET
arweave

完全な秘密鍵の復元なしに自己主権的な秘密鍵の共有を実現

これからは、預けるから自分で握るデータ管理へ

エコシステム

Application Layer	TOBARI	分散型のGoogle Drive
Middleware Layer	CryptoShepherds Protocol	暗号化キーの共有不要な フォルダ共有ミドルウェア
Protocol(Dapp) Layer	FORMIX	コンセンサスがストレージに 準拠した秘密鍵管理プロトコル
Infrastructure Layer	ARWEAVE	Arweaveブロックチェーン

FORMIX の名前の由来

FORMIXはラテン語で「アリ」を意味するFormicaに、テック系の接尾辞-ixを組み合わせた。
「個体は単純だが、集団で高度な自律分散組織を築く」というアリの生態を、FORMIXの
「プロトコルとして全体を支える分散型鍵管理インフラ」というコンセプトに重ね合わせて命名

SecHack365での1年

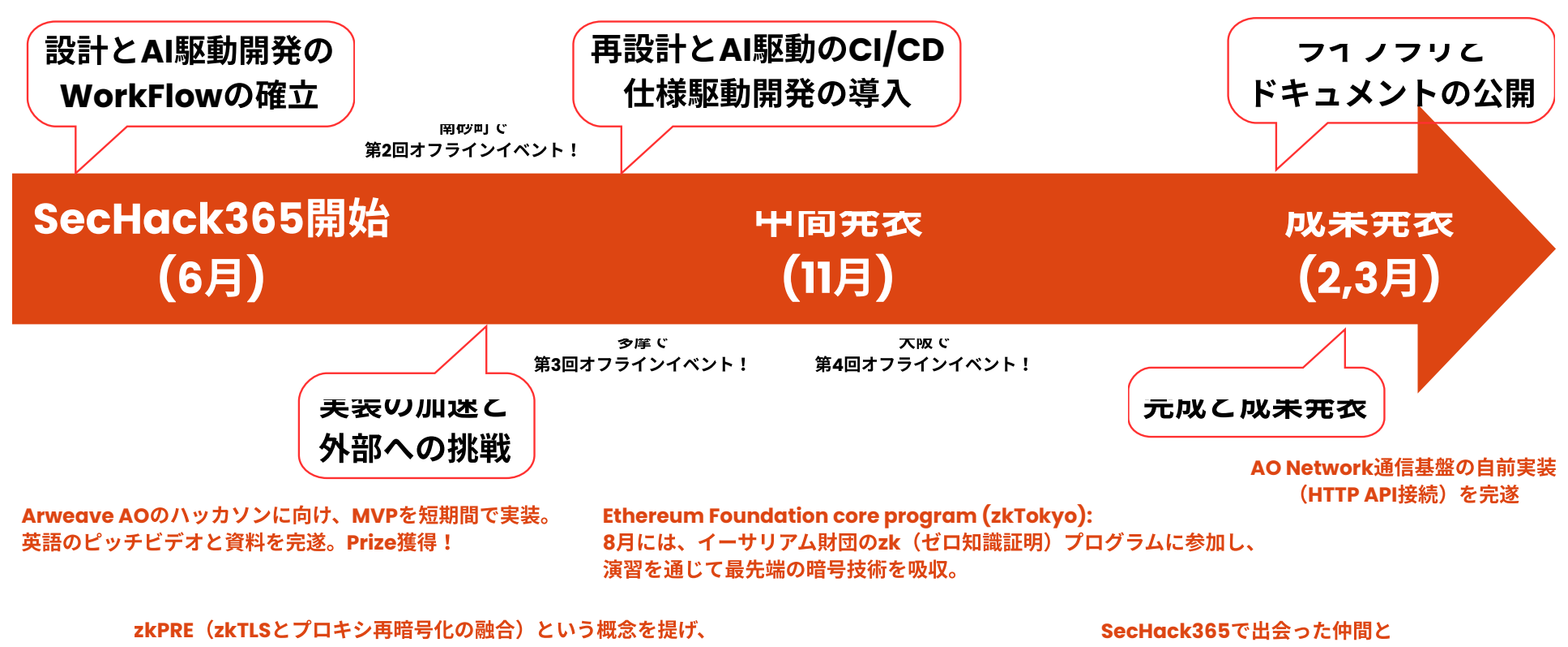
Arweave/AO Network上でのRust開発に着手
AOコミュニティに直接質問を投げながら
ライブラリが未整備な中でKVSベースの
Contractのステート管理を設計

秘密鍵管理とアクセス制御をモジュール化
Shamirs Secret Sharingの実装を完了させるなど
暗号のコアロジックを実装

AI駆動CI/CDの導入: Claude CodeやCodex、
Git Worktreeを組み合わせたAI agentによる並列開発環境を構築。
実装はClaude Code、レビューはCodexという分業体制

仕様駆動開発（SDD）: 不安定なVibe Codingを脱し、
設計書（Spec）から実装を自動生成・検証するSDDワークフローを定着

開発規模の拡大に伴い、依存関係を整理するためレイヤードから
クリーンアーキテクチャへに変更。


設計とAI駆動開発のWorkFlowの確立
南砂町で
第2回オフラインイベント！
再設計とAI駆動のCI/CD
仕様駆動開発の導入
ファイナライズと
ドキュメントの公開
SecHack365開始
(6月)
中間発表
(11月)
成果発表
(2,3月)
多摩で
第3回オフラインイベント！
大阪で
第4回オフラインイベント！
実装の加速と
外部への挑戦
完成と成果発表

Arweave AOのハッカソンに向け、MVPを短期間で実装。
英語のピッチビデオと資料を完遂。Prize獲得！

Ethereum Foundation core program (zkTokyo):
8月には、イーサリアム財団のzk（ゼロ知識証明）プログラムに参加し、
演習を通じて最先端の暗号技術を吸収。

zkPRE（zkTLSとプロキシ再暗号化の融合）という概念を提げ、
イーサリアム創設者Vitalik Buterin氏の前でピッチを実施。
技術の独自性を世に問いました。

AO Network通信基盤の自前実装
（HTTP API接続）を完遂

SecHack365で出会った仲間と
LLM trading botを一緒に開発start

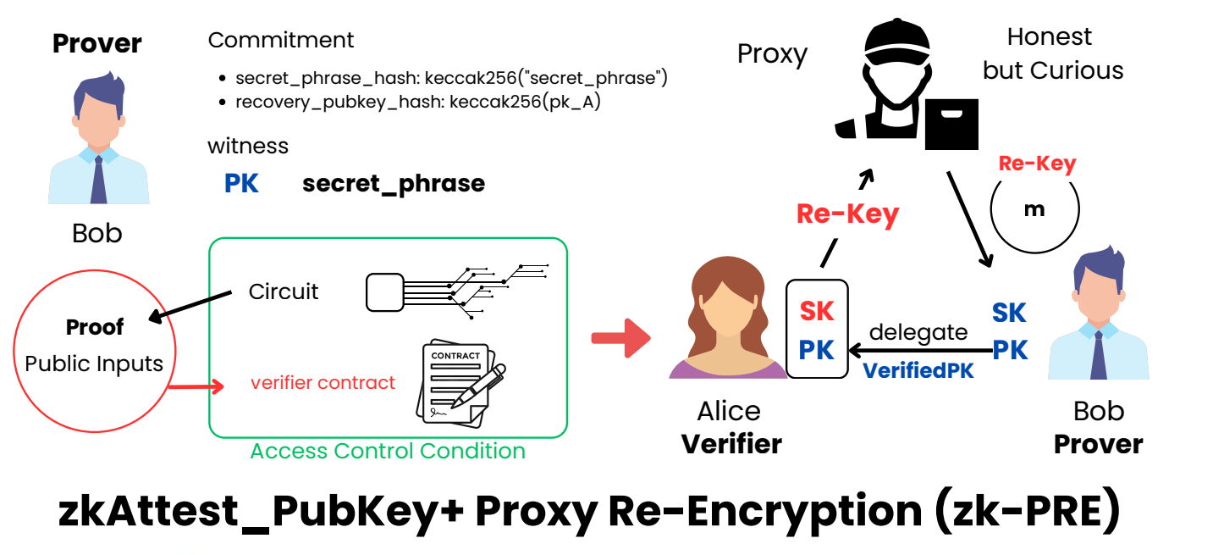
GitHub repository link and QR

https://github.com/shodaimomiyama/FORMIX

今後の展望 Secret-as-a-serviceの実現

プログラマブルな、公開鍵の正当性の検証プロトコルの開発

再暗号化キー生成のプロセスで使用する公開鍵の正当性の検証をzkを用いて行う


Access Control (Verify Pub_Key)
Prover
Commitment
• secret_phrase_hash: keccak256("secret_phrase")
• recovery_pubkey_hash: keccak256(pk_A)
witness
PK secret_phrase
Bob
Proof
Public Inputs
Circuit
verifier contract
Access Control Condition
Proxy Re-Encryption
Proxy
Honest
but Curious
Re-Key
Re-Key
m
SK
PK
delegate
VerifiedPK
SK
PK
Alice
Verifier
Bob
Prover

zkAttest_PubKey+ Proxy Re-Encryption (zk-PRE)

ToC向けサービス TOBARI のリリース

より自由で自己主権的なデータ共有
分散型のGoogle Drive

より自由に	より自己主権的に
リンクかアカウントによる共有は、 実はとても窮屈な共有方法	ローカルPCと ブロックチェーンのみに関与
オンチェーンデータとロジックを 正とした柔軟なアクセス制御	UXはそのままに