ブロックチェーン応用講座

Vol.3: ウォレット

小林 聖弥 / Seiya Kobayashi

目次

目次

1. さまざまなブロックチェーンウォレット

Ledger

ブロックチェーンウォレットの分類

A. ウォレット所有者・管理者による分類（🤔 アカウント管理 = ○○の管理？）

- **カストディアルウォレット（custodial wallet）**：第三者が管理
  → e.g.）Coinbase、BitFlyer、Coincheck
- **ノンカストディアルウォレット（non-custodial wallet）**：ユーザー自身が管理
  → e.g.）MetaMask、Trust Wallet、Phantom

B. インターネット接続性による分類

- **ホットウォレット（hot wallet）**：インターネットに接続できる（上記はすべて該当）
- **コールドウォレット（cold wallet）**：インターネットに接続できない（🤔 署名は？）
  → e.g.）Trezor、Ledger

1. さまざまなブロックチェーンウォレット

ブロックチェーンウォレットの分類

C. 秘密鍵の管理方法（生成・保管・電子署名生成 & 検証）による分類

- **HD（Hierarchical Deterministic）ウォレット** → 個人単位で最も利用されている
  - 秘密鍵生成: シードフレーズ（seed/mnemonic phrase）から階層的に生成
  - 秘密鍵保管: ユーザーパスワードを間接的に用いて暗号化した状態で保管
  - 電子署名生成: メモリー上で復号された秘密鍵を用いて生成

    → ⚠️ 生の秘密鍵がインターネット接続された機器のメモリー上に載る点がリスクの一つ
  - 電子署名検証: **プロトコルレベル**で検証（ECDSA）

    → 🤔 誰が検証する？
  - e.g.）MetaMask、Trezor

1. さまざまなブロックチェーンウォレット

ブロックチェーンウォレットの分類

C. 秘密鍵の管理方法（生成・保管・電子署名生成 & 検証）による分類

- **マルチシグ（Multi-Sig）ウォレット** → 透明性を必要とする組織等での利用に最適
  - 秘密鍵生成 & 保管: **複数人がそれぞれ**のウォレットを用いて生成・保管
  - 電子署名生成: **複数人がそれぞれ**のウォレットを用いて生成

    → 電子署名方式は検証用スマートコントラクトが対応しているものであればなんでも良い
  - 電子署名検証: **独自のスマートコントラクト**で検証

    → ⚠️ n人の署名者がいればn個の署名が**それぞれ独立して検証される**（🤔 課題？）
  - e.g.）Safe、BitGo

1. さまざまなブロックチェーンウォレット

ブロックチェーンウォレットの分類

C. 秘密鍵の管理方法（生成・保管・電子署名生成 & 検証）による分類

- **MPC（Multi Party Computation）ウォレット** → 大企業等での利用に最適
  - 秘密鍵生成: 複数人が**秘密鍵のシェア（share）**を持ち合う

    → **分散鍵生成（DKG）**と呼ばれるプロトコルを用いる（🤔 生成 → 分割はなぜNG？）
  - 秘密鍵保管: 複数人がそれぞれのシェアを保管
  - 電子署名生成: 複数人が一定数以上（e.g., 5-of-7）のシェアを持ち合って生成

    → **閾値署名（Threshold Signature）**と呼ばれるプロトコルを用いる
  - 電子署名検証: **プロトコルレベル、あるいは独自のスマートコントラクト**で検証

    → ⚠️ n人の署名者がいる場合も**一つの署名に集約**され、検証される
  - e.g.）OKX、Zengo

1. さまざまなブロックチェーンウォレット

ブロックチェーンウォレットの分類

C. 秘密鍵の管理方法（生成・保管・電子署名生成 & 検証）による分類

- **FHE（Fully Homomorphic Encryption）ウォレット** → まだ実験段階で実用的ではない
  - 秘密鍵生成 & 保管
    - **完全準同型暗号（FHE）:=** $Enc(x \cdot y) = Enc(x) \cdot Enc(y)$ を応用したい

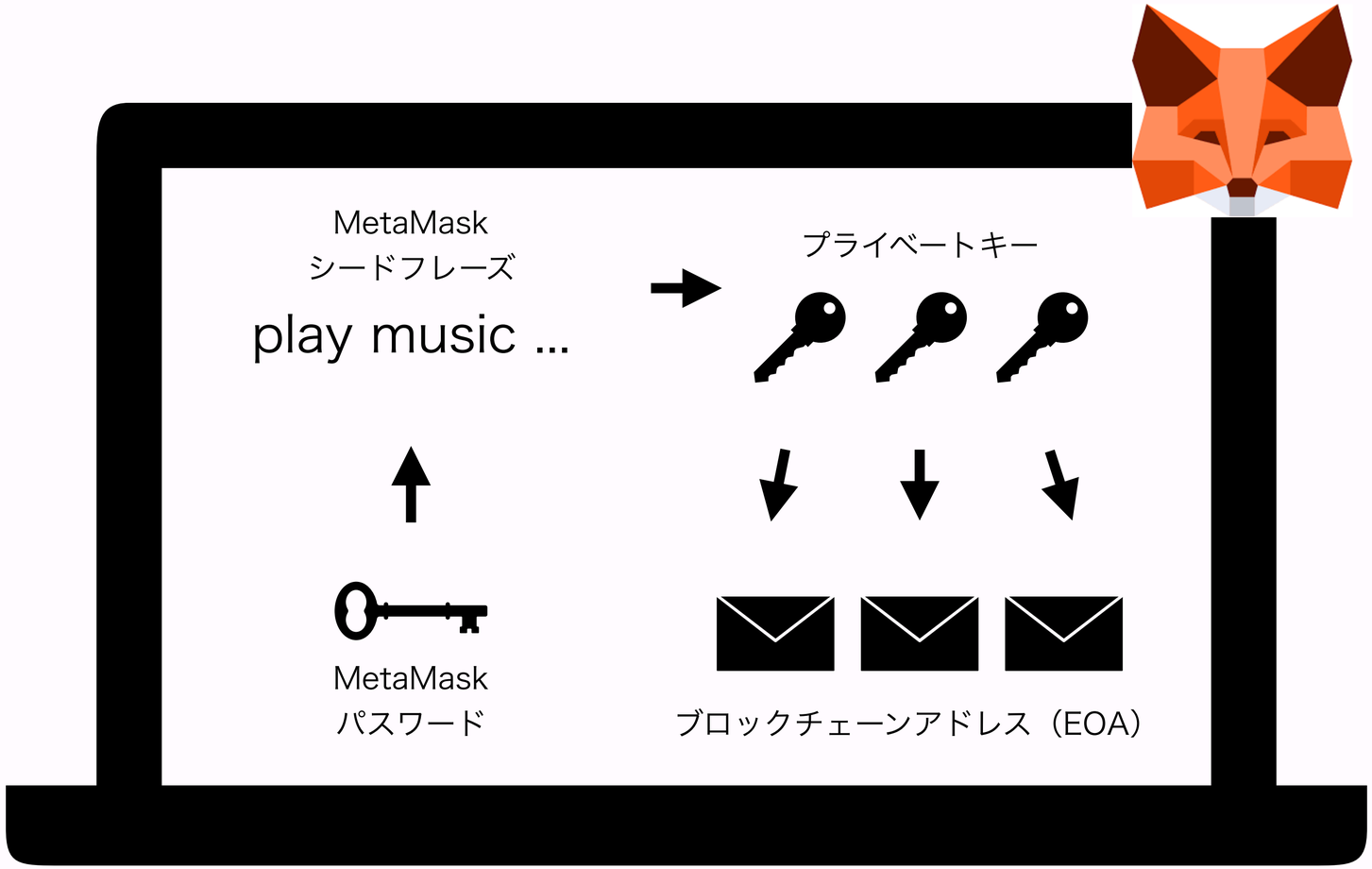
      → データを暗号化したまま**正しく**計算（足し算あるいは掛け算）を行うことができる

      → 🤔 暗号化すべきは秘密鍵か、トランザクションか？
  - 電子署名生成 & 検証: 🤔 誰がどのようにして生成 & 検証できそうか？課題？
  - e.g.）Zama

目次

2. HDウォレットの仕組み

HDウォレットの実体 := BIP-39・BIP-32・BIP-44の実装



2. HDウォレットの仕組み

HDウォレットの実体 := BIP-39・BIP-32・BIP-44の実装

- BIP-39: シードフレーズ → シード値

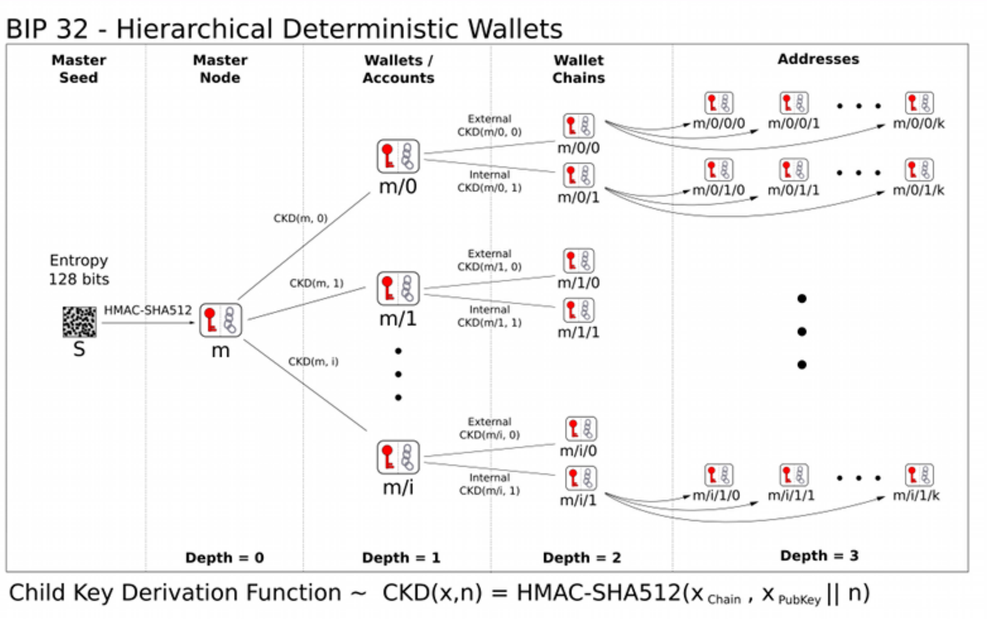
1. **12単語**のシードフレーズを**ランダムに**生成する: 132 bits（128 bits + 4 bits）
   → 💡 人間が扱いやすいように自然言語化している
2. シードフレーズから**シード値**（BIP-32 seed）を導出する: 512 bits
   → PBKDF2-HMAC-SHA512 と呼ばれるハッシュ関数ベースの**鍵導出関数（KDF）**を用いる
   → ビット長を拡大している（132 → 512 bits）と捉えることもできる
3. （MetaMaskの場合）ユーザーパスワードを間接的に用いて**シード値を暗号化**
   → ユーザーパスワードを KDF を用いて秘密鍵化（🤔 なぜ？）
   → 対称暗号（AES-256-GCM）を用いてシード値を含むデータ（keyring）を暗号化して保管

2. HDウォレットの仕組み

HDウォレットの実体 := BIP-39・BIP-32・BIP-44の実装

- BIP-32: 鍵ペアの階層型決定的な導出

1. シード値から**マスター秘密鍵（master private key）**を生成する: 512 bits
   → HMAC-SHA512 と呼ばれる
   暗号学的ハッシュ関数を用いる
   - 前半 256 bits = マスター秘密鍵
   - 後半 256 bits = チェーンコード
     → 子秘密鍵の導出におけるエントロピー
2. チェーンコードから**子秘密鍵**を生成する
   → 再帰的に同じ導出方法を用いる



BIP-32

2. HDウォレットの仕組み

HDウォレットの実体 := BIP-39・BIP-32・BIP-44の実装

- BIP-44: アドレス導出パス

  - Ethereumでは、デフォルトで「**m / 44’ / 60’ / 0’ / 0 / 0**」という導出パスを用いている
    - m: マスター秘密鍵から導出する、の意
    - 44’: BIP-44ベース
    - 60’: コインタイプ（= Ethereum）
    - 0’: アカウントインデックス → **変動する（指定可能）**
    - 0: 外部の導出パスか否か（UTXOの名残り）
    - 0: アドレスインデックス → **変動する（指定可能）**

  → ‘ がついている階層ではセキュリティ向上のため、**ハードニング（hardening）**が行われる

2. HDウォレットの仕組み

HDウォレットの性質

- **1つのシードフレーズから**ほぼ無限に近い（~2^{128}）鍵ペアが導出できる
  - ⚠️ シードフレーズが流出すると導出可能なすべての鍵ペアが流出してしまう

    → 🤔 同じ鍵ペアになる確率はゼロではないのではないか？

- デバイスを変更した場合は、**インデックスを変動させて**鍵ペアを再導出する
  - MetaMaskの場合: アカウントインデックスのみを変動させる
  - ⚠️ 全ての鍵ペアが自動的に導出されるわけではなく、1つ（index = 0）のみ導出する

    → 残りの鍵ペアはユーザーがマニュアルで操作して導出する

    → 🤔 100万個の鍵ペアを導出していた場合は大変ではないか？

目次

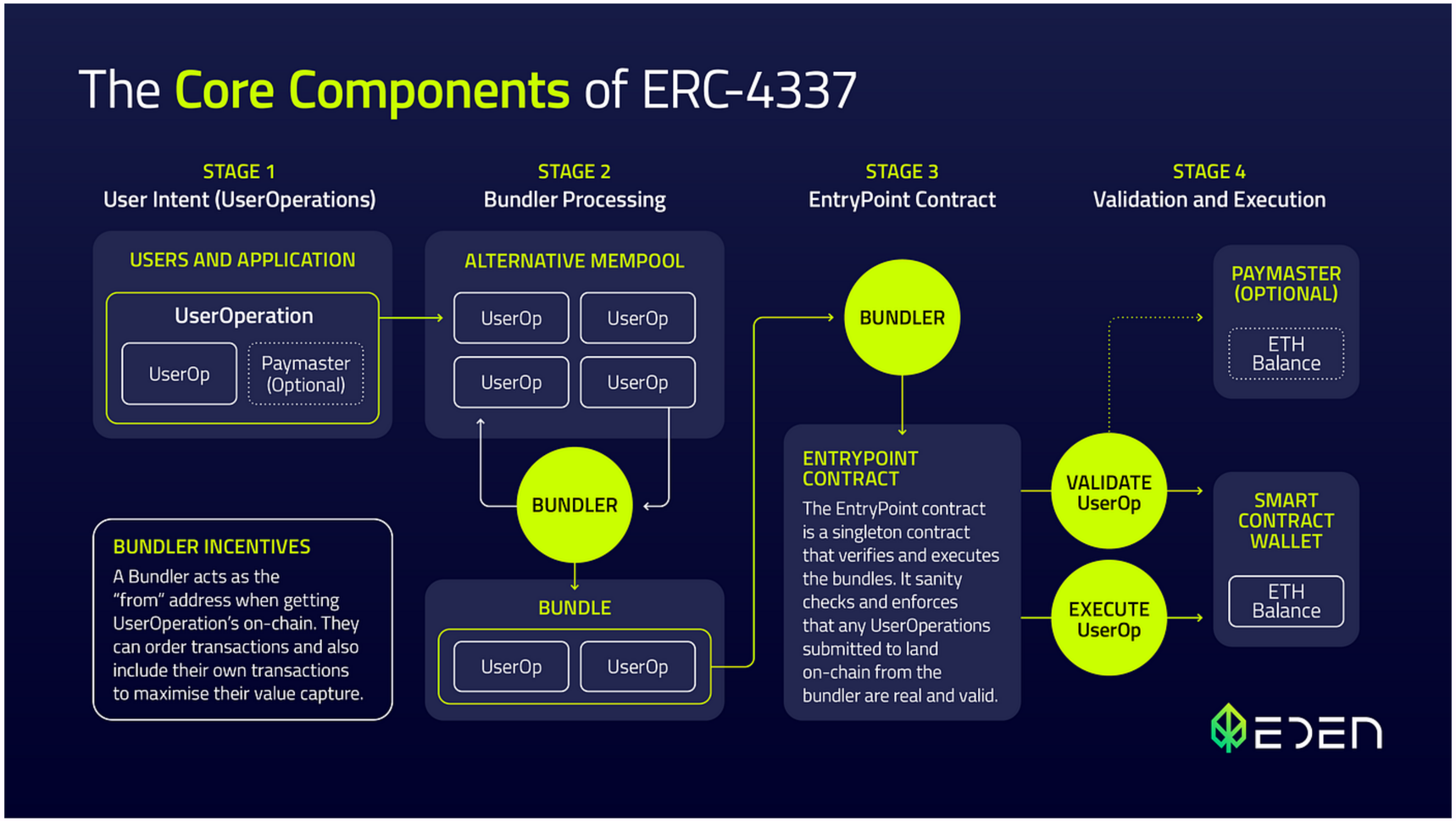
3. 近未来のウォレット: スマートコントラクトウォレット

スマートコントラクトウォレット・アカウント := ERC-4337

- プロトコルを変更しない **SCA（Smart Contract Account）**の実現方法
  → 💡 EOAを一時的にスマートコントラクト化するような提案もある（EIP-7702）

- 2つの抽象化によってUX向上を図る
  - **アカウント抽象化**: EOAに依存しないアカウント概念
    - スマコンを用いてウォレットに対して任意のロジック（e.g., 電子署名方式）を実装できる
      → 🤔 **EOAのみ**しかトランザクションは開始できないのでは？
  - **ガス抽象化**: ユーザーにガス代を意識させない仕組み
    - **スポンサー**: ユーザーはガス代を支払わなくて良い
    - **ERC-20トークン払い**: ユーザーはネイティブトークン（e.g., ETH）を保有しなくて良い

3. 近未来のウォレット: スマートコントラクトウォレット



Medium

目次

🗣️ ワークショップ #1: 理想的なウォレットについて考えよう

グループ毎に**理想とするウォレットの条件**を洗い出し、その**実現方法**を考案しましょう

- ⏳ **時間配分**: グループ内ディスカッション 15分 +（発表 3分 + Q&A 5分）× 3
- 📣 **発表に最低限含めたい事項**
  - 理想条件
    - 想定するユースケース（誰がどういった場面で使うウォレットか）
    - トラストモデル（どこに、あるいは何に信頼をおく必要があり、なぜ信頼に足るのか）
  - 実現方法（ハイブリッド型も検討する）
    - 条件に対する妥当性
    - 具体的なウォレットサービス名（e.g., MetaMask）
    - リスク・懸念点

目次

💻 ワークショップ #2: MetaMaskをローカルチェーンに接続しよう

MetaMask ウォレットの作成・初期化

1. ウォレットの作成
    - 「新規ウォレットを作成」をクリック
    - 「MetaMask ポリシーに同意」をクリック
    - MetaMask パスワード（⚠️ これは秘密鍵ではない）の設定
2. シードフレーズ（リカバリーフレーズ）の取得
    - （一般的には）オフラインの媒体（e.g., 紙）に書き写すことが推奨される

💻 ワークショップ #2: MetaMaskをローカルチェーンに接続しよう

MetaMask ウォレットの作成・初期化

3. ローカルネットワークの追加

    - 手動で以下のネットワークを追加
        - Network Name: **Localhost**
        - Default RPC URL: **http://127.0.0.1:8545**
        - Chain ID: **31337**
        - Currency Symbol: **ETH**

4. いろいろ試してみる

    - e.g.）アカウントの追加、プライベートキーのインポート、ロック & アンロック