Tezos
011 Hangzhou
Upgrade

会社紹介: ダイラムダ株式会社

Tezosブロックチェーン研究開発

- コア開発: ノードやプロトコルの実装
- アプリケーションへのコンサル業務

相場はやりません。

Tezos

- POS ブロックチェーン (LPoS)
- エネルギー消費低。Mining ではなく Baking
- スマートコントラクト (Michelsonスタックマシーン)
- オンチェーンガバナンス
- NFT が流行っているようです

ブロックチェーン状態 (TzStats)

Tezos で dApps をどう開発するか

Tezos は OCaml で書かれている…

OCaml を勉強する必要はありますか?

ありません。dApps は何を使っても良い

- 基本: HTTP/HTTPS によるノード RPC アクセス
- ライブラリとして JavaScript, TypeScript, Python がある
- スマートコントラクト記述: SmartPy, LIGO, SCaml などがある

技術ドキュメントはポータルに集められている

Tezos プロトコルアップグレード

- 2018-07-01 001 Alpha
- 2018-07-22 002 Alpha2
- 2018-11-27 003 Alpha3
- 2019-05-30 004 Athens
- 2019-10-18 005 Babylon
- 2020-03-05 006 Carthage
- 2020-11-12 007 Delphi
- 2021-02-14 008 Edo
- 2021-05-11 009 Florence
- 2021-08-06 010 Granada
- 2021-12-04 011 Hangzhou

Tezos のオンチェーン・ガバナンス

プロトコルのハードフォーク案を投票で承認する

- ステークに比例した投票権
- 二段階投票
    - 複数の案から一件を選ぶ投票
    - 選ばれた案を実際に採用するか投票

投票で承認されいるのでコミュニティーがフォークしにくい

開発者が好き勝手はできない

004 Athens (アテネ)

オンチェーン・ガバナンスの初の社会実験

- 2019-02-26 提案
- 2019-05-29 移行

- Gas limit 緩和
- ステーキング条件緩和

005 Babylon

初の本格的なプロトコルアップグレード

- 2019-07-26 提案
- 2019-10-18 移行

- 合意アルゴリズムの改良 (Emmy から Emmy+ へ)
- スマートコントラクト強化 (エントリポイントのサポート)
- デリゲーション手続の単純化

006 Carthage (カルタゴ)

メンテナンスが主目的

- 2019-11-13 提案
- 2020-03-05 移行

- Block gasリミット +30%
- スマートコントラクトVM命令の強化
- 細かいバグ修正、プロトコルのコードクリーンアップ

007 Delphi (デルフォイ)

- 2020-09-03 提案
- 2020-11-12 移行

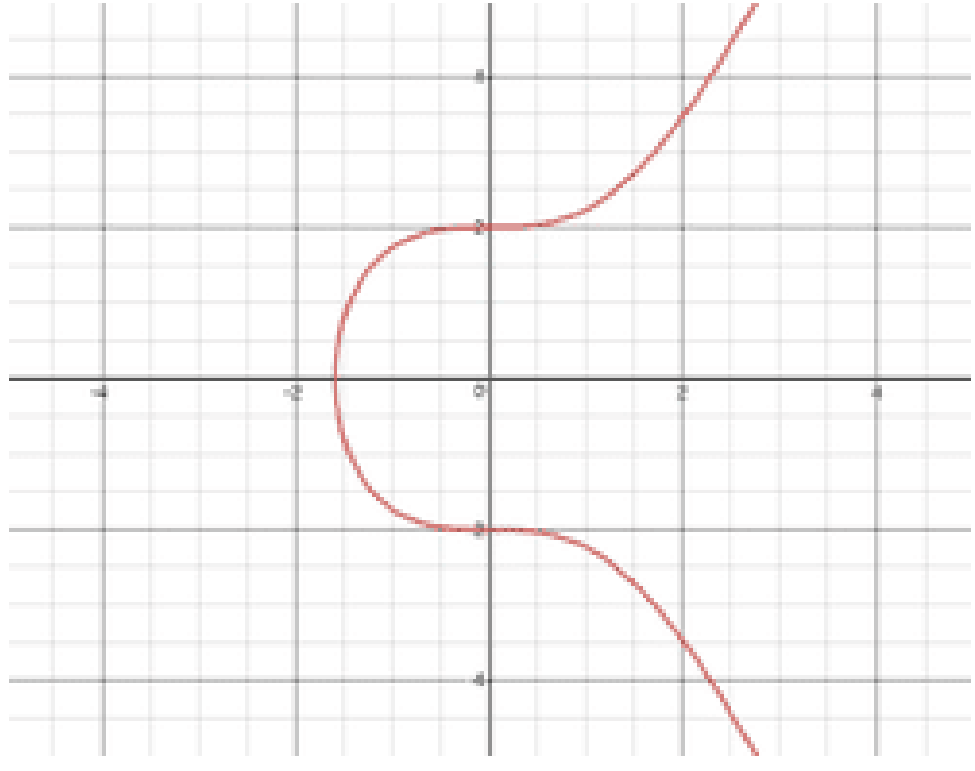
- VM 命令の gas コスト再定義と低減
- ストレージ費用の低減。Gas コスト 1/4 に。

008 Edo

- 2020-11-20 提案
- 2021-02-14 移行

- BLS12-381 ペアリング暗号命令。ゼロ知識証明
- Sapling プロトコルの導入。Shielded Tez による匿名送金
- Keccak, SHA3, PVSS 命令
- Ticket
- ガバナンス: Adoption(移行)期間の導入

BLS12-381 とゼロ知識証明



匿名コイン Zcash の Sapling に使われている楕円曲線と
双線形写像の実装

009 Edo から Tezos/Michelson で使用可能に

BLS12-381 in Tezos

型

- `bls12_381_g1`: G_1 の型
- `bls12_381_g2`: G_2 の型
- `bls12_381_fr`: 係数(fraction)の型

計算

- `INT` : `bls12_381_fr` から `int` への型変換
- `ADD` `MUL` `NEG` : G_1, G_2 と係数の計算に対応

ペアリング検査命令

- `PAIRING_CHECK`

チェーンは検査に特化している。データ作成の便利な命令はない。
本来はオフチェーンでやるべきこと。

BLS12-381 とゼロ知識証明

ゼロ知識証明

ある知識を知っていることを、
その知識そのものを公開せずに証明する。

例: 公開鍵署名

「自分はメッセージ m に対してこの署名 σ を
作成できる秘密鍵 s を知っている」

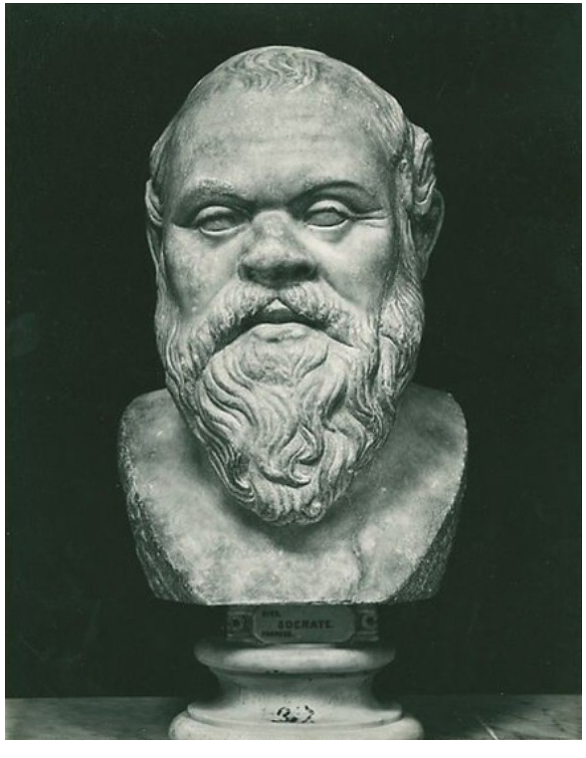
例: 匿名通貨

「自分は残高が n 以上あるUTXOの秘密鍵 s を持っている」
のでその一部 n を送金してほしい

BLS12-381 とゼロ知識証明

Tezos/Michelsonは証明の検証のみ提供。

検証器とそのパラメータ、そこに提供する証明データの
作成は提供されていない



ZoKrates を利用可能

ゼロ知識証明ツールキット

- 証明したい性質を Python 風プログラムとして入力、検証器コード(Solidity)とパラメータを得る
- 性質を満たすデータを入力、証明データを得る

簡単な変更でTezosに **使用可能**

Sapling

Zcash の Sapling protocol を導入

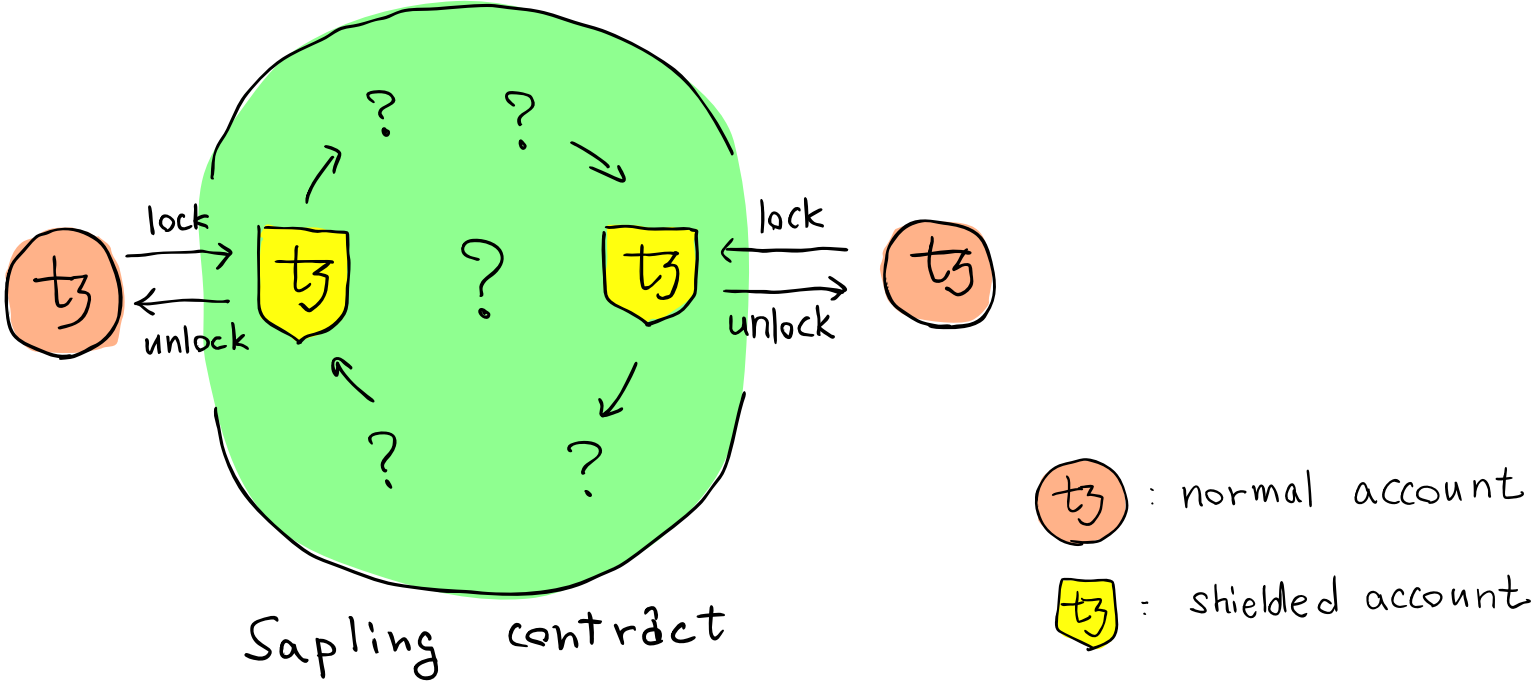
スマートコントラクトでゼロ知識証明を使った
匿名トランザクションが可能に。

Tezos本体が匿名コインになるわけでは無い!!

Shielded tez

`sapling_contract.tz`: 1 shielded tez = 1 tez の匿名コイン

コントラクト内のトランザクションは匿名化



Sapling 命令は Shielded tez のためだけではなく、
匿名トランザクションを使った FT dApps に利用できる

ウォレット対応

`tezos-client sapling` コマンド

各操作に対応するサブコマンドが用意されている

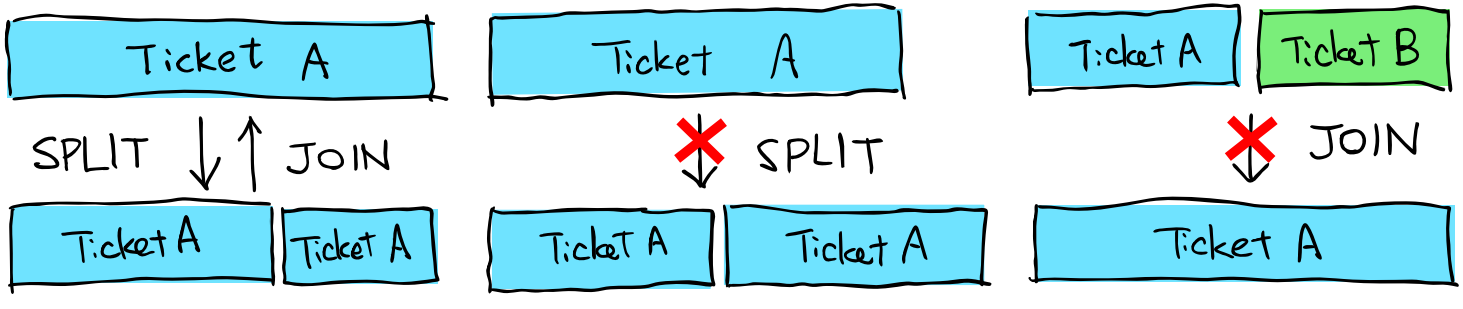
GUIウォレット

AirGap が Sapling トランザクションをサポートしている

Ticket

偽造できない「保存量」の発行が可能

- スマートコントラクトが自由に発行できる
- Ticket には発行者情報が付属、他人は偽造できない
- 量を保存する分割(split)、統合(join)しかできない



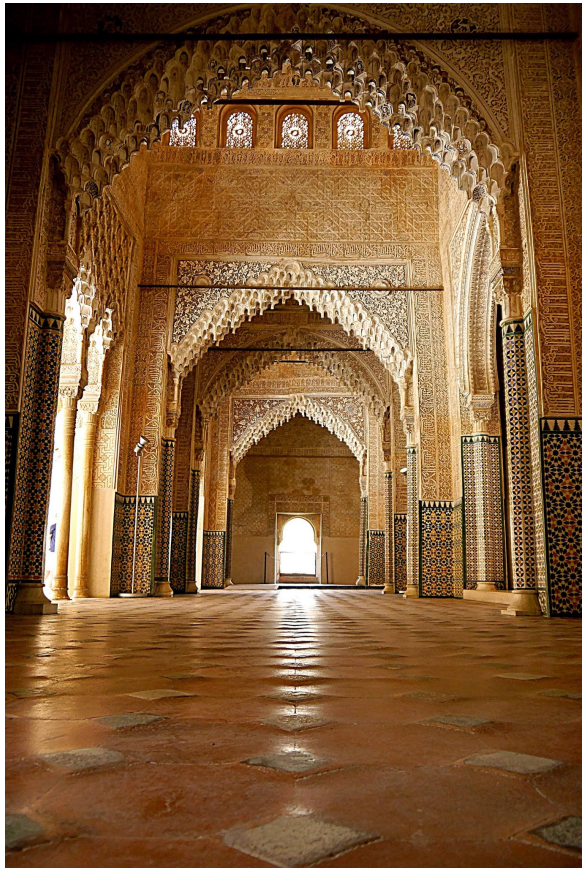
発行総量を保存しない移転操作は自動的に失敗するので、通貨や認証システムを簡単に間違いなく作れる

009 Florence (フィレンツェ)

- 2021-03-04 提案
- 2021-05-11 移行

- オペレーションサイズ x2
- コントラクト呼出順序の修正
- ガス費最適化
- ガバナンス: テスト期間のテストチェーン起動を廃止

010 Granada



- 2021-05-31 提案
- 2021-08-06 移行

- 合意アルゴリズムの改良 (Emmy+ → Emmy*)
- Liquidity Baking
- Gas 費削減: 従来の 1/3 ~ 1/6 に

Emmy*

合意アルゴリズムの改良。

まだ Nakamoto スタイルアルゴリズムなので確立的 finality。

- Block time: 60秒 → 30秒
- Endorsements/block : 32 → 256
- Finality : 12分 → 1.5分 (40% ステークを持つ攻撃者がいた場合)
- 1 cycle: 4096 → 8192

詳しくは Faster finality with Emmy*

BFT スタイル、決定的 finality のある Tenderbake への布石

Liquidity Baking

プロトコルに強化された DEX (Tez ⇆ tzBTC)

Liquidity pool を提供するCPMMコントラクト
KT1TxqZ8QtKvLu3V3JH7Gx58n7Co8pgtpQU5

- プロトコルは各ブロックごとに 2.5tez 鋳造し CPMM コントラクトに足す
- 増えた 2.5tez 分は流動性提供のインセンティブになる
- 全 tez の 1/16 がプールされるまでは baking より儲かる
- インフレ対策: CPMM を使って両替すると 0.1% が burn される
- Baker は Liquidity Baking を止める投票ができる

詳しくは Liquidity baking in Tezos Granada proposal — how it works

011 Hangzhou (杭州)

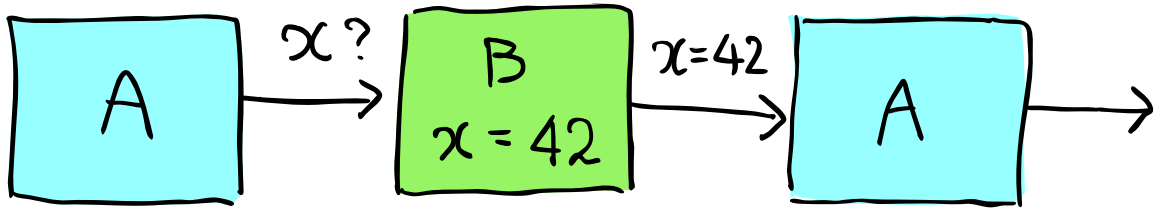
- 2021-09-22 提案
- 2021-12-04 移行

- Views: 他のスマートコントラクトの値を簡単に参照できる
- Timelock: Baker による BPEV (“front runnning”) を防止する
- Cache: 頻繁にアクセスされる値をキャッシュすることでガス消費を抑える
- Global table of constants: よく使われる値やコードの共有化
- Context flattening: ファイル配置変更による最適化

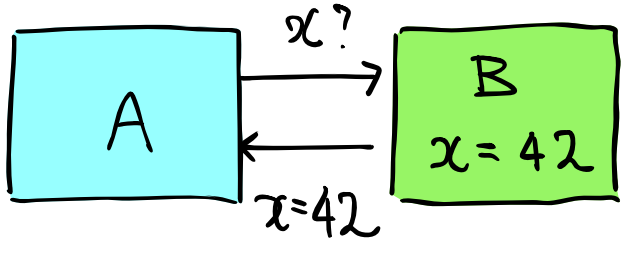
Views

他のコントラクトの値を簡単に取得できるようになった。

View 以前: CPSの形で自分自身を再度呼び出す必要



View: 関数呼び出しの形で値を取得可能



Timelock

スマートコントラクトによるトークン実装のさらなる安全化

BPEV

ブロック作成者が、他人のトランザクションを覗き見て、
自分の利益になるよう自分のトランザクションを挿入したり、
順番を入れ替える

トランザクションの暗号化

二段階の操作

- 暗号化したトランザクションをスマートコントラクトに送付
- 実行順が固定されてから白文を渡し、実行してもらう

Cache

よくアクセスされる値をアクセスコストの低い記憶域に保存し、
アクセスガスコストを削減する。

Global table of constants

コントラクト内のサイズの大きい値(テーブルや関数)を
明示的にアクセスコストの低いテーブルに登録する。

コントラクト間での関数の共有化が可能

- ライブラリ
- コントラクトサイズの削減

Context flattening

記憶域(context)のレイアウトの単純化

過去の実装上の理由からネストされていたディレクトリを平らに

```
/contracts/index/01/34/a7/54/f8/c3/
  a778ba71aee4e7a595b3f500289568157cf79b5be7f2/balance

=>

/contracts/index/
  a778ba71aee4e7a595b3f500289568157cf79b5be7f2/balance
```

- ノードの性能向上 (25%スピードアップ)、記憶域使用量の削減
- 記憶域アクセスガスコストの低減
    - 0.00025 JPY / ディレクトリ (1 tez = 500 JPY の時)

011 Ithaca

- 2021-12-20 提案

- Tenderbake: BFT合意アルゴリズムによる決定的finality
- Precheck of operations: gossip network のスループット向上

Tenderbake

Bizantine 耐性のある合意アルゴリズム

- Tendermint を元にしている
- 確定的 finality
    - 一旦確定すると裏返ることはない → 安心
    - 今までの Nakamoto コンセンサスは確率的 finality