

ブロックチェーン公開講座 第16回 L2技術

芝野恭平

東京大学大学院工学系研究科技術経営戦略学専攻

ブロックチェーンイノベーション寄付講座

特任研究員

shibano@tmi.t.u-tokyo.ac.jp





Agenda

- L2技術について概要
- Optimistic Rollupの仕組み
 - Arbitrum Oneの事例より解説
- Proto-dankshardingについて
 - 概要
 - KZGコミットメント
- 注意：本講義で取り扱う情報は、技術的な情報の共有を目的としており、投資等を推奨するものではありません。
- 講義で取り扱う各プロジェクトの情報は不十分な場合があります。



L2技術について：概要

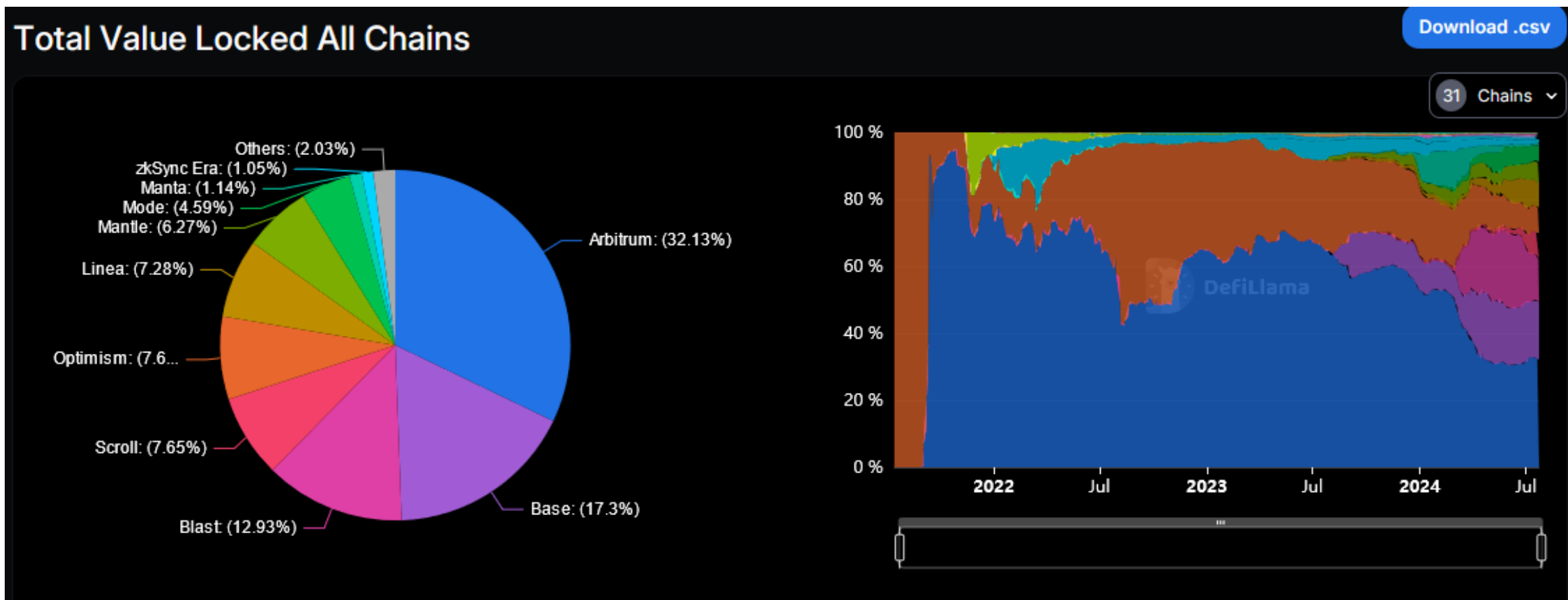
- BitcoinやEthereumは、ブロック生成速度・ブロックあたりのTx（トランザクション）数の制限が多く以下の2つの問題が生じている。
 - TPS（Transactions Per Second）が遅い。
 - 限られたTxしか処理できない。
 - トランザクション手数料（ガス代）の高騰。
 - 処理できるTx量に対して、多くのTxリクエストが生じていることが原因。
- これを解決するのが、スケーリングソリューションと呼ばれる。
 - L2はスケーリングソリューションの一つの方法。
 - Txを安く、たくさん処理できるように。
- L2とは、Leyer2の略で、BitcoinやEthereumなどLayer1のブロックチェーンに対し2層目のレイヤーという意味。
 - L2はL1に付随して初めて機能する。
 - そのため、L2単体では実行不可。

L2：事例

- Bitcoin
 - Lightning Network
- Ethereum
 - Rollup
 - Optimistic Rollup → 今回はここを取り扱います。
 - 各Txの計算結果の検証はしない.
 - 不正なTxも入り得てしまうので, 特定の期間に異議申し立て (フラウドプルーフ) ができる.
 - ZK Rollup → ZKの回で触れます.
 - ゼロ知識証明という暗号学的技術を用いて計算結果の保証する.
 - 計算自体をしなくとも計算結果の検証が $O(1)$ で可能.
 - 以下, 過去に議論されていたもの:
 - Plasma
 - Raiden Network

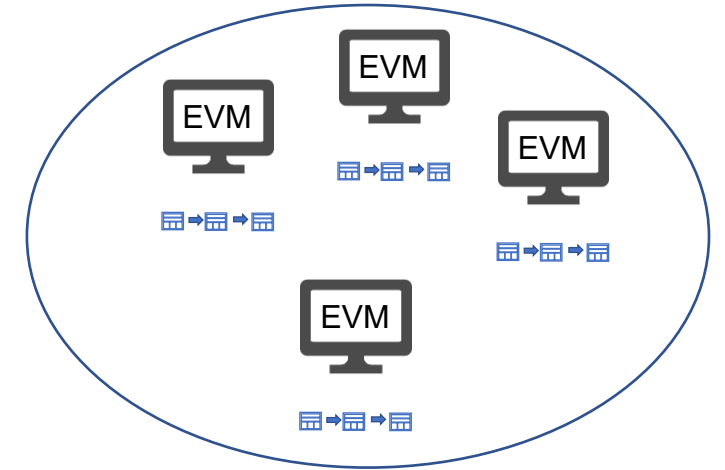
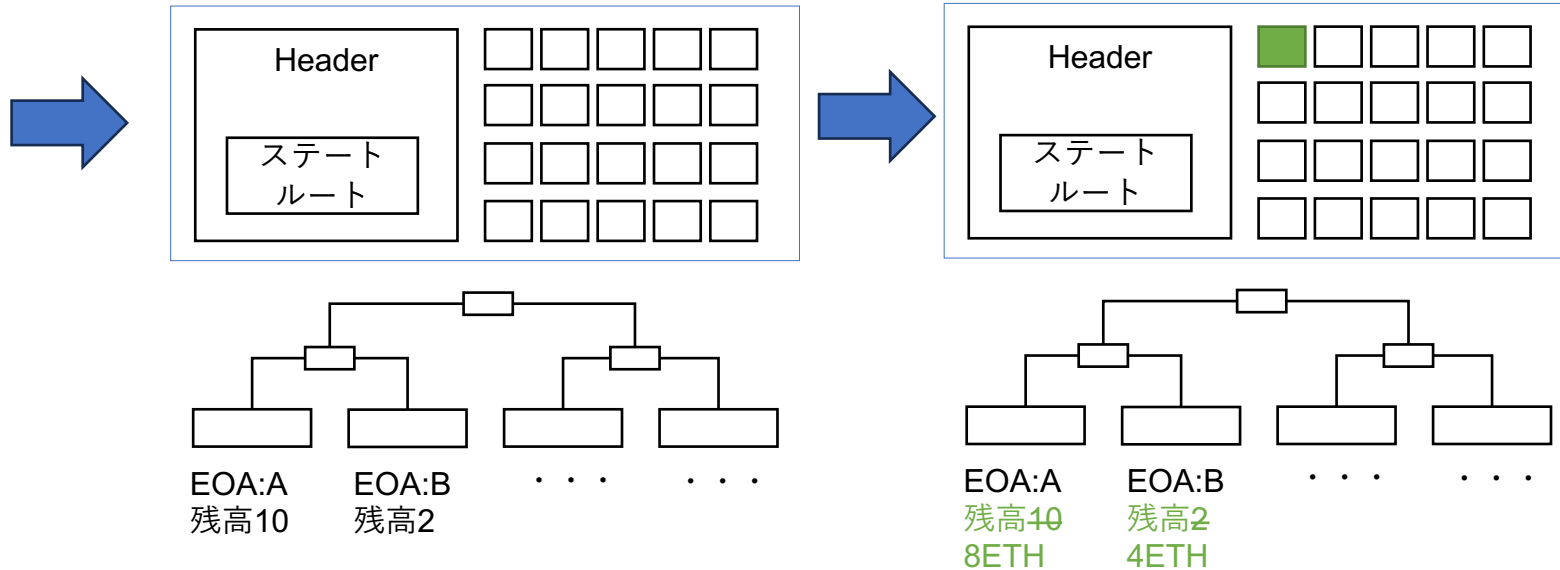


各種RollupプロジェクトのTVL



<https://defillama.com/chains/Rollup>

Ethereumのセキュリティについて：トランザクションとステート



EVMにてコントラクトの処理結果が一意になることが保証される。

グローバルステートで状態を管理している。
ブロック内のTxを実行してステートを更新している

- 安全性：
 - PoSのバリデータが投票：ステーキングしている額に応じて不正をしなくなる。
 - 世界中のフルノードで各々のTxの実行をしてそのステートになる，という「答え合わせ」をしている。

Rollupの考え方

- つまり, Txとそれに伴うステートを記録するのがブロックチェーン.
- この「Tx」と「ステート」をEthereumと違うものをコントラクトを使って実現したい.
 - L1と同様のセキュリティを確保するため, Ethereumを使用して実現をする.
- そしてL1よりも安くしたい.
 - 「速く」は難しい. なぜならばRollupも通常のL1のTxと同様にL1への書き込みが行われることによって処理を確定している.
 - L1に一度に書き込まれるTxが多くなるためにスループット (1秒あたりに処理できるTx数) は増えるが, 1txの承認速度はやはりL1のブロック生成時間 (約12秒) と同様になる.

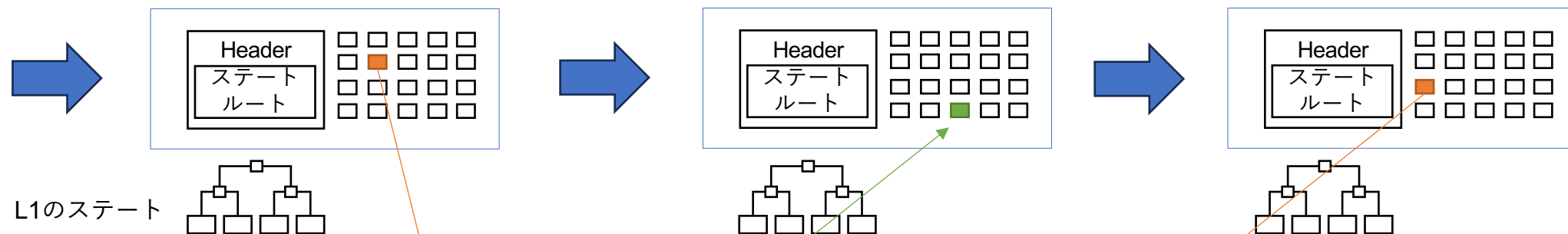


Rollup : どうやって実現していく ?

- 単純な方法
 - L1の一つのTx (L1Tx) にたくさんL2Txを詰め込み, L1コントラクトですべての計算検証を行いL2ステートの更新を行う.
 - 実現できるが, ガス代が安くない
- L2Tx列だけ詰め込んで記録しておくが計算の検証とL2ステートの更新はL1コントラクト内部では行わない
 - calldata領域に記録
- がしかし, それだと本当にそのL2Txを実行してそのL2ステートに変更されるのか, 署名が有効なのかどうかわからない
 - Optimistic Rollupでは, 事後的に不正なL2Txを検出できるようになっている (フラウドプルーフ)
 - ZK Rollupでは, オフチェーンで署名の検証とL2ステート更新を行い, それをZKでコントラクトで検証可能に.
- イメージとしては, L1TxにL2Txを詰め込めるだけ詰め込み, それをひとかたまり (L2のブロック) にしてL2ステートを更新している

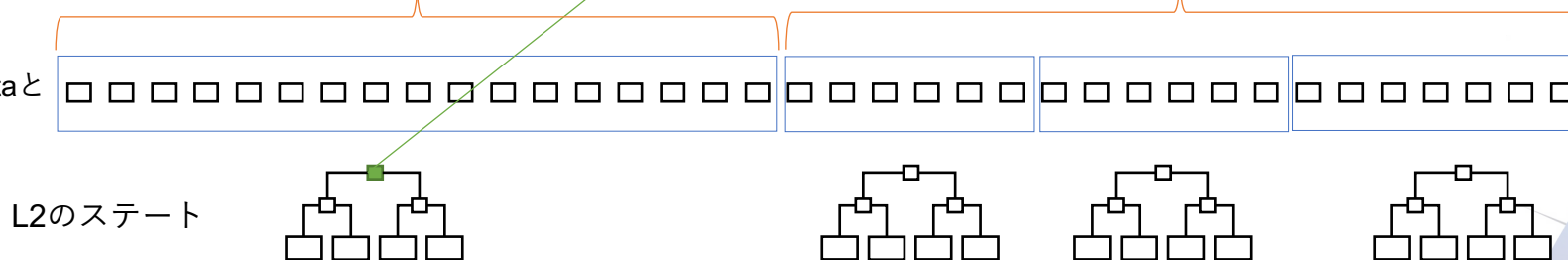
Rollupでの処理イメージ図

L1ブロックチェーン

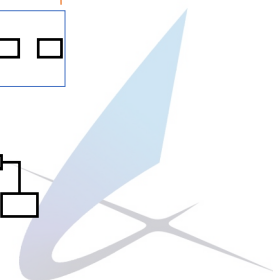


L2


L2TxはL1にcalldataとして書き込まれる。



L2ステートは、L2ブロック内のL2Txを実行して更新。
何らかのタイミングでL1にステートルートが記録。



どれだけガスコストが安くなるか？

Name	Send ETH	Swap tokens
 Arbitrum One	< \$0.01	< \$0.01 ▾
 Optimism	< \$0.01	\$0.02 ▾
 Polygon zkEVM	< \$0.01	\$0.10 ▾
 zkSync Era	< \$0.01	- ▾
 Loopring	\$0.01	\$0.85 ▾
 zkSync Lite	\$0.02	\$0.05 ▾
 DeGate	\$0.05	\$0.19 ▾
 Ethereum	\$1.04	\$5.19 ▾

およそ1/100から1/1000まで1 Txあたりの手数料が低くできる。

<https://l2fees.info/>

Arbitrum Oneについて

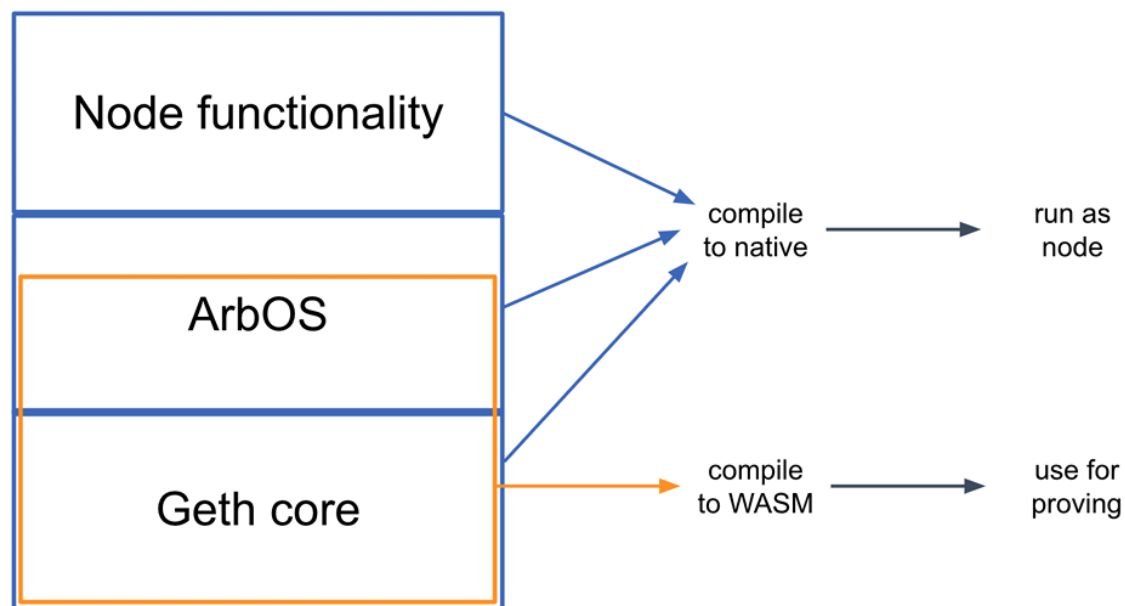


- Optimistic Rollupの詳細を理解するため、現在稼働中の人気のあるOptimistic Rollupの一つであるArbitrum Oneの仕組みを理解する。
- Arbitrumの概要：
 - EthereumのL2でOptimistic Rollupを実現している。
 - Arbitrum OneとArbitrum Novaの2つの異なるL2がある
 - Offchain Labsが開発し、2021年Arbitrum Oneをリリース。
 - <https://www.oklink.com/ja/arbitrum>
 - 2023年からはArbitrum DAOが運営。ARBトークン。
 - https://diamond.jp/crypto/market/arbitrum/#index_id0



クライアントソフトウェアアーキテクチャ

- Arbitrumのフルノードのクライアントソフトウェアのアーキテクチャ
- EVMの互換性を保つためにgethを内包している
 - geth: Ethereumの有名なクライアントソフトウェアの一つ。Go言語で作られている。



トップレイヤー：

RPC接続など外部との接続。Ethereum互換のノードとして動作。ほとんどの機能がgethから提供されている。

ミドルレイヤー：

L2のノードとして必要な機能が搭載。Txデータの圧縮解凍やL1とのブリッジ機能など。

ベースレイヤー：

EVM互換のコントラクトの実行を担う。ノード内部で動作。
WASM互換のWAVMで動作するようになっており、gethはWASMにコンパイル後ビルトインされている。

状態遷移関数

L2Txを実行しL2ステートを更新する関数。

Arbitrumのエンジン部分。

MbSC2030

Arbitrum One : Txの処理の流れ

ユーザー

L1と同等のL2Txをウォレットで作成

シーケンサー

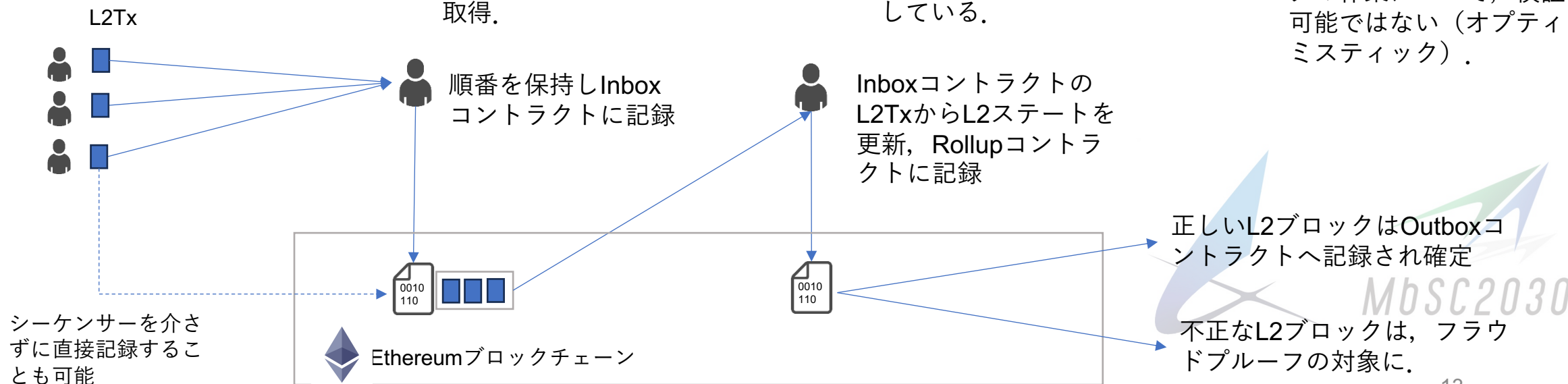
- ユーザーからL2Txを受取り, 順番を保持しL1のコントラクトに記録する担当.
 - calldataに記録
- シーケンサーは世界で一人のみ.
- トラストが必要.
- シーケンサーは手数料を取得.

バリデータ

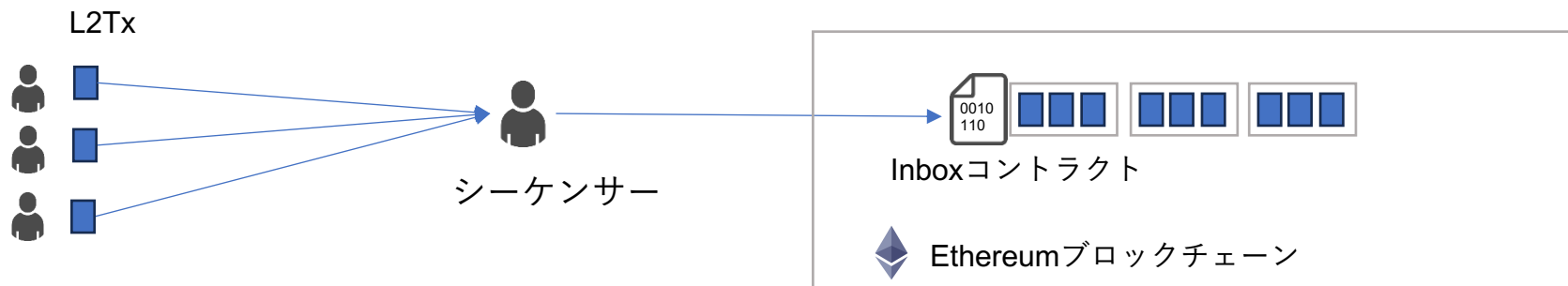
- シーケンサーが記録したL2Txをもとに, L2ステート更新.
- バリデータは世界に複数人.
- パーミッシュド.
- フラウドプルーフを行える.
- バリデータがL2ブロックチェーンの合意形成を担当している.

※ ユーザーがL2Txを作成し, L2ステートの更新がされる基本的な流れは左.

- シーケンサーとバリデータという2つの役割を担う人が必要.
 - 現在は集権的に運用しているが徐々に分散化していく予定.
- シーケンサー, バリデータの作業について, 検証可能ではない (オプティミスティック).

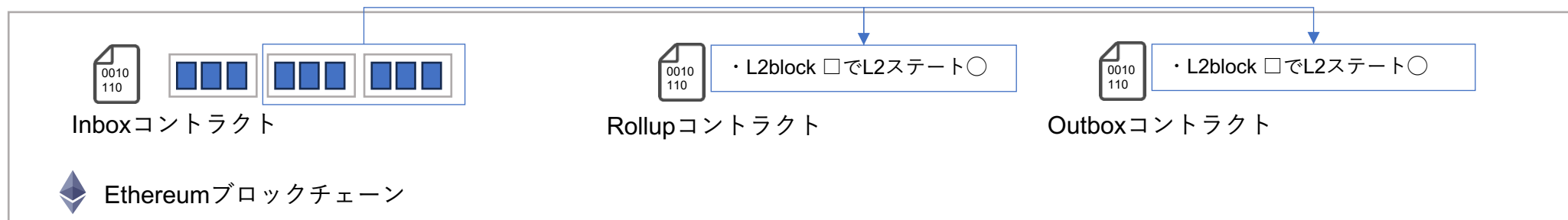
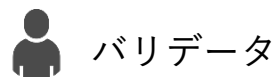


Arbitrum One：シーケンサーの処理



- シーケンサーの役割は、ユーザーから受け取ったL2Txをその順番のままL1に書き込むこと。
 - トラストポイントとなる。
 - 現在はOffchain Labsが運営。
- シーケンサーもL2フルノードであり、L2Txの実行、検証したあとにL1に書き込む。（が正しい保証はされない）
- ユーザーから受け取るL2Txは署名がついているのでシーケンサーによる改ざんは不可能。
- 書き込まれるL1コントラクトはInboxコントラクトと呼ばれる。
- Inboxにcalldataとしてデータを入れる。
 - L1Txに含まれるデータなのでL1ブロックチェーンには記録されるがL1ステートの更新を伴わない
 - ガス代が比較的安くすむ。
- 一つ一つのL2Txはbrotliという圧縮形式で圧縮されて書き込まれる。
- 書き込む際、L2でのブロック単位で区切って書き込まれる。
 - その後バリデータがそのL2ブロック単位でL2ステートの更新を行う。

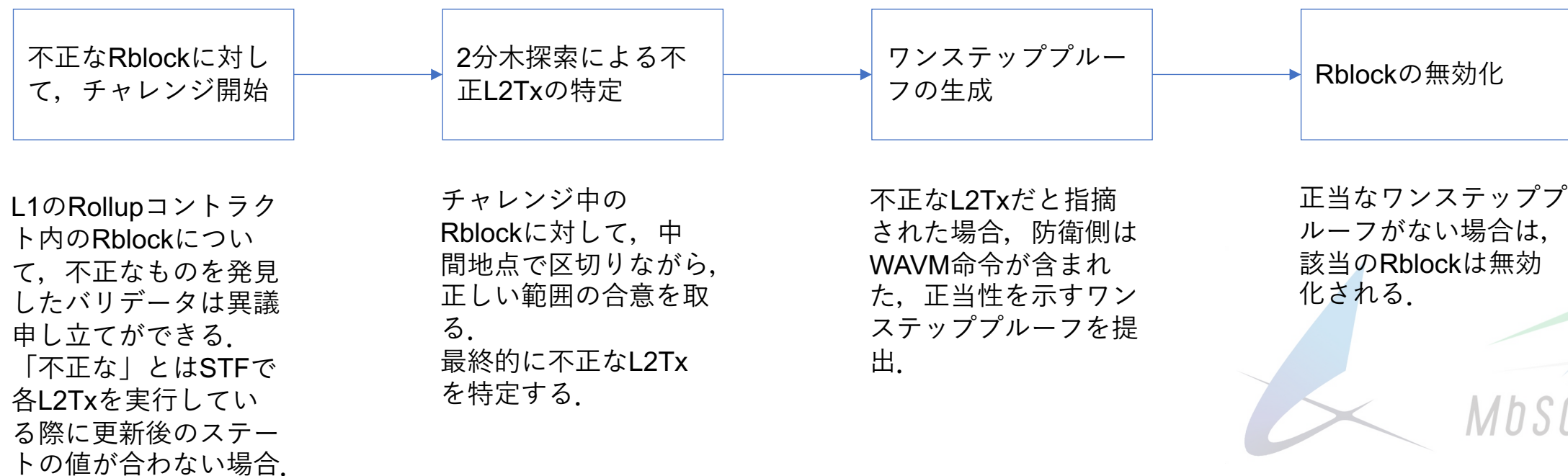
Arbitrum One : L2ブロック生成, L2ステートの更新 (バリデータが実施)



- バリデータは, L1のInboxコントラクトにあるL2ブロックをいくつかまとめてRblockという単位でブロック生成.
 - RblockではInboxにあるL2ブロック単位で複数個同時に入れることができる
 - RblockではL2ステートの更新を行う (before, afterのL2ステートを記録)
- RblockはまずはL1のRollupコントラクトに記録される
 - チャレンジ期間中 (7日間) はフラウドプルーフされる可能性がある.
 - 期間終了後L1のOutboxコントラクトに記録
- Outboxコントラクトへの承認 (confirm) が終わると状態が確定
 - L2資産をL2からL1に移動可能に.

Arbitrum One：フラウドプルーフ（チャレンジ）

- L2ステートの更新はすべてL2ノード内の状態遷移関数(State Transition Function ,STF)で行う。
- Rblock単位でL2ステートの更新を行う際、含まれるL2ブロック内のL2Txすべてを実行する。
- 無効なL2Txが含まれていた場合はSTFで処理が無視され、L2ステートの更新には関わらない。



Arbitrum One：ガス代の支払いと報酬の配分

- ガス代は以下の2つを足し合わせたもの：
 - L2 gas
 - TxをArbitrumで実行するためのガス代で，Ethereumと同等の計算式で算出される.
 - L1 calldata:
 - シーケンサーがL1にcalldataとしてTxを記録するための料金. シーケンサー経由



Optimistic Rollup (Arbitrum One) まとめ

- L2ソリューションの一種
- L2TxはシーケンサーによってL1のcalldataに書き込まれる.
- ステートの更新はRblock単位で, バリデータによって行われる.
- 最終的にOutboxコントラクトに記録されることで確定.
- 不正なRblockはフラウドプルフの対象になる.

Proto-danksharding : 概要

- Ethereum L1のデータアベイラビリティソリューション
- Ethereumプロトコル上に、データが保持できる安価なデータ領域が作られた。
- Dencun Update により実装 2024/3
- フル Danksharding の一歩手前のソリューションとして実装
- EIP-4844にて実現。
- リリース後、多くのL2がProto-dankshardingを利用するように。
- 特徴：
 - Blobデータと呼ばれるデータ形式の導入。
 - calldataで保管するよりガス代が安い。
 - Blobはブロック外に保管される。
 - コンセンサスレイヤーで保管
 - 約18日間（4096エポック）は保管される。
 - それ以降のデータの保管はプロトコルでは保証されない。
 - Optimistic Rollupでの利用を想定。チャレンジ期間を超えた期間保存される。
 - コントラクトからBlobデータにはアクセスできない。コントラクトとは独立したデータ保管用領域。
 - 1つのBlobは約127KBで、1ブロックに最大6つまで保管できる。
- Blobデータの例：
 - <https://blobscan.com/block/20153255>





データ保管性能のスケラビリティ

データ保管

ブロックあたりの最大データ保管量

→ 約1.79MB

1byteあたりの保管費用 (calldata)

→ 16 gas/byte

※ calldataにデータ保管 → 16 gas/byte (EIP-2028)
ブロックのガス上限は3000万なので1ブロックへの最大保管データ量は
 $30,000,000 / 16 = 1,875,000 = 1.79\text{MB}$

Proto-danksharding

→ 約 762 KB追加 (計約

2.53MB= 1.79MB + 762KB)

約1 gas/byte

※ 1 blob: 127KB
1 block: 6 blob
 $2^{17} = 131,072$ gas/blob で計算.

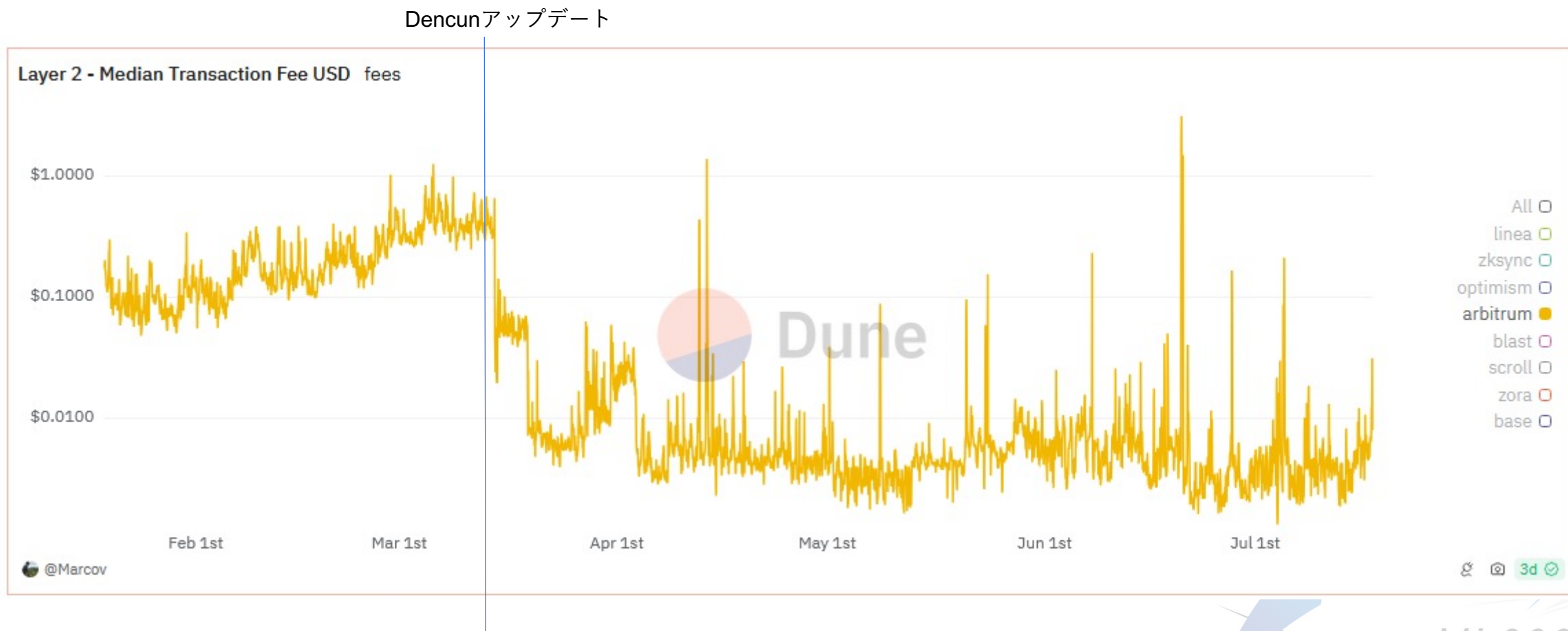
(参考) 計算性能

0.71 MIPS
(1970年代後半のCPU性能)

※ 1命令3ガスとして、ブロックガス上限3000万、ブロック生成時間を14秒とすると
 $30\text{M} / 3 / 14 = 0.71$ MIPS



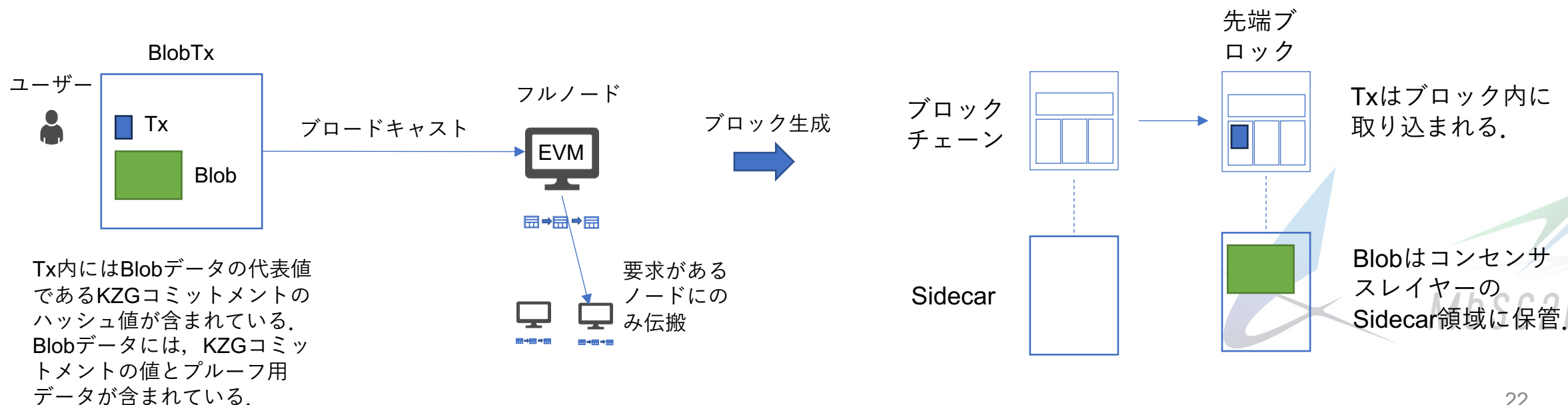
Dencun前後でのArbitrumのトランザクション手数料の変化



https://dune.com/queries/3521133/5921142?aggregation_e15077=&interval_days_e15077=180

Blobはどのように実現されているのか

- コンセンサスレイヤーでBlobデータが保管され、その代表値をTxに保存.
 - Txは通常のTx. つまりブロックチェーン内に保管されるため半永久的に記録される.
- 代表値はKZGコミットメントのハッシュ値
 - ローデータが正しくその時に記録されたものであるかどうかをチェックするのに利用.
- ユーザーは通常のTxと同様にウォレットでBlob用Txを生成し、ブロードキャストする.



KZGコミットメント：概要

- KZGコミットメントとは，多項式コミットメントの一種.
- 多項式コミットメントとは，証明者が自分が知っている多項式 $f(x)$ が

$$f(\alpha) = \beta$$

- を満たすことを，検証者に $f(x)$ そのものを伝えることなく示すことができる手法.
- Blobでは，データを多項式の係数に変換し，KZGコミットメントを利用してデータをあとから検証可能にしています.
 - KZGコミットメントが使いたいがゆえに，Blobデータを多項式に変換し利用している.
 - 元データを取得した人は，公開されているコミットメント/ウィットネスの値を用いて自分のデータの正しさを検証可能.
 - Blobの用途では，データは公開されているためKZGコミットメントを使用せずともハッシュなどでも代替可能.
 - プロトダंकシャーディングでは必ずしもKZGコミットメントを使う必要はないが，その後のフルダंकシャーディングでは必要になるため，このタイミングで実装されている.
- また，次回以降の講義でのゼロ知識証明のプロトコルでも使用されている.
- ここでは「コミットメントとはなにか」「多項式コミットメントとは」「KZGコミットメントの仕組み」の順番で解説していく.

コミットメントとは？

コミットメント

多項式コミットメント

KZGコミットメント

- 多項式コミットメントはコミットメントの一種である。
- コミットメントとは、封筒と同じような性質を持つ処理を実現するもの。
 - 以下の2つの性質がある：
 - Hiding：中身を秘匿化。
 - Binding：中身を変更できない。
 - 処理としては、コミットメントを作成するコミットフェーズと、その中身を公開するオープンというフェーズがある。
- 例えばこれを用いるとオンラインでのじゃんけんを実現できる。上記HidingとBindingの2つの性質があるので、以下の要件を満たせる。
 - 各プレイヤーは、自分の手を出すまで他のプレイヤーの手を知ることとはできない（Hiding）
 - 各プレイヤーは、一度自分の手を出した後、その手を入れ替えることはできない（Binding）
- 各プレイヤーは自分の手のコミットメントを作成し、相手に共有する。
- 2人共コミットメントの共有が終わったあとに、オープンすることで自分の手が公開され、じゃんけんが成立する。

コミットメントとは？

- このケースの場合は、例えばハッシュ関数SHA256を用いて計算できる.
- 手 h を
 - グー：“0”
 - チョキ：“1”
 - パー：“2”
- という文字列で定義する.
- そして、例えば乱数として100文字のアルファベットからなる文字列 r を生成し，
 - $\text{SHA256}(h+s)$
- を計算しこれをコミットメント c として提出. ここで $h+s$ は文字列としての結合.
- 2人が c を出し終わったあと、それぞれの人は自分の手と乱数 h, s をそれぞれ公開する.
- 相手はもうひとりの h, s を用いて提出された c と一致するか確認することで、本当に手が h だったかどうかを検証できる.

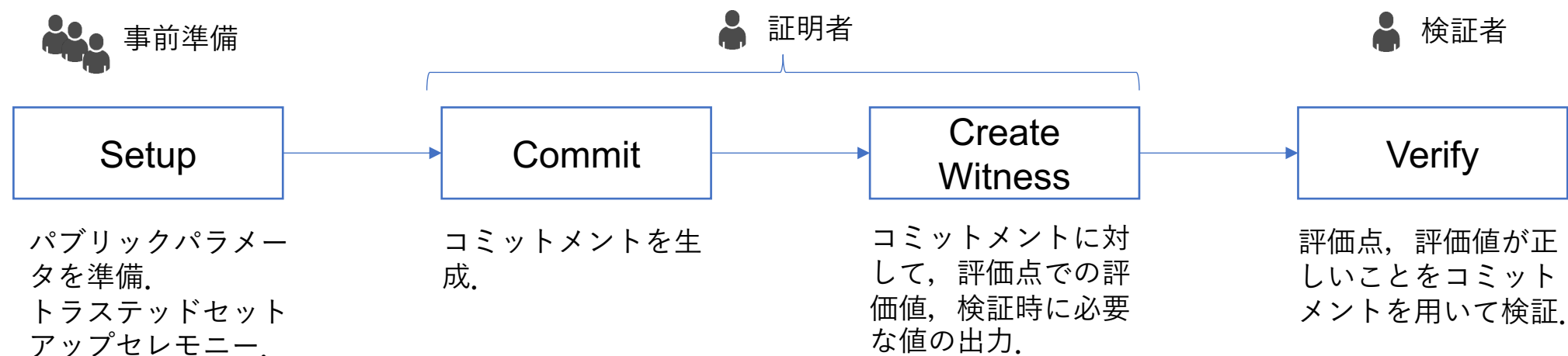
多項式コミットメントとは？

- 多項式コミットメントとは，証明者が自分が知っている多項式

$$f(x) = f_n x^n + f_{n-1} x^{n-1} + f_{n-2} x^{n-2} + \cdots + f_1 x + f_0$$

- に対して係数ベクトル (f_0, f_1, \dots, f_n) をメッセージとするコミットメントスキーム.
- 加えて，任意の評価点 α における値が $f(\alpha) = \beta$ であることも係数ベクトルを開示せずに示すことも可能.
- KZGコミットメントは多項式コミットメントの一種で以下の2つの用途として用いられる.
 - コミットメントを提示し，そのコミットメント提示時点の多項式そのものを開示する．検証者は提示されているコミットメントと多項式から導出されるコミットメントの値を一致しているか確認することで確認する．
 - コミットメントを提示し，多項式そのものを開示せずに，コミットメントを用いてある点 α における値が $f(\alpha) = \beta$ であることを示す．検証者は，証明者から送られてきたウィットネスとコミットメントを用いて $f(\alpha) = \beta$ であることが確認できる．

KZGコミットメントの処理の流れ



- Setupフェーズ, コミットメントフェーズ, ウィットネス作成フェーズ, そして検証フェーズで構成される.
- Setupフェーズは, KZGコミットメントを利用する前に必須となる事前処理である.
- 証明者 (多項式を知っている側の人) が, コミットメントフェーズ, ウィットネス作成フェーズを実行する.
- 検証者は, 証明者が多項式を知っておりとある点 α での多項式の値が β であることを検証フェーズを通じて確認可能.
- KZGコミットメントでは, 任意の x における多項式を取り扱うのではなく, とあるランダムな一点 s で評価する.

KZGコミットメント準備：剰余の定理と，ランダムな点 s での評価

- 剰余の定理は，多項式 $f(x) - f(\alpha)$ が $(x - \alpha)$ で割り切れることを保証する．
- つまり以下の関係を満たす $n - 1$ 次の多項式 $q(x)$ が一意に存在：

$$f(x) - f(\alpha) = q(x)(x - \alpha)$$

- 多項式について，任意の x でこの等式が成立することを示す代わりに，ランダムな一点 s を評価した結果が等しいこと，つまり

$$f(s) - f(\alpha) = q(s)(s - \alpha)$$

- が成り立つことを示す．
- このとき証明者に s の値を知られてはいけない．
 - ※ Schwartz-Zippel の補題を使うと，もし悪意のある証明者が s を知らない場合， $f(\alpha) \neq \beta$ であっても不正な多項式を構築し $f(s), q(s)$ を提出することが困難であることが示せる．
 - KZGコミットメントでの検証プロセスで使用．
- この s を証明者に隠した状態で $f(s), q(s)$ を計算させる部分が KZGコミットメントの重要な箇所．
- 具体的には，楕円曲線とペアリングという方法を用いる．

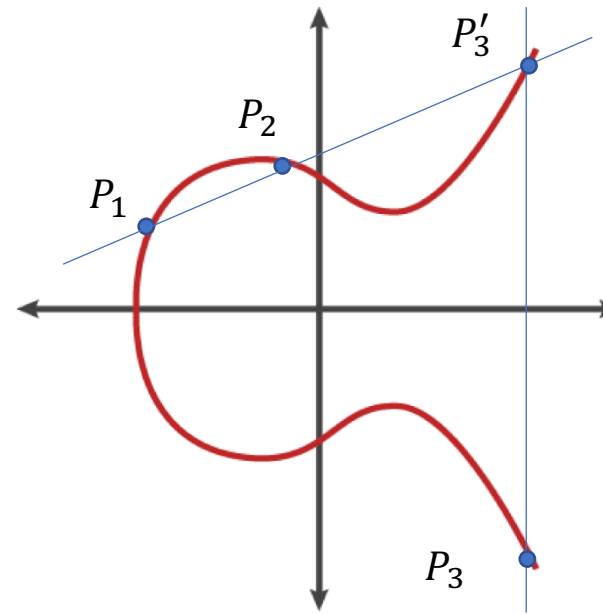
KZGコミットメント準備：有限体と楕円曲線

- 位数 q の有限体を F_q , F_q 上で定義される楕円曲線を $E(F_q)$ と表記します.
- 楕円曲線上の点には加算 $P + Q \in E(F_q)$ と2倍演算 $2P \in E(F_q)$, これらを組み合わせたスカラー倍 $nP \in E(F_q)$ が定義される.

有限体 F_q 上の楕円曲線とは,

$$y^2 \equiv x^3 + ax + b \pmod{p}$$

の解 $(x, y) \in (\{0, 1, \dots, p-1\}, \{0, 1, \dots, p-1\})$ である.



$$P_3 = P_1 + P_2$$

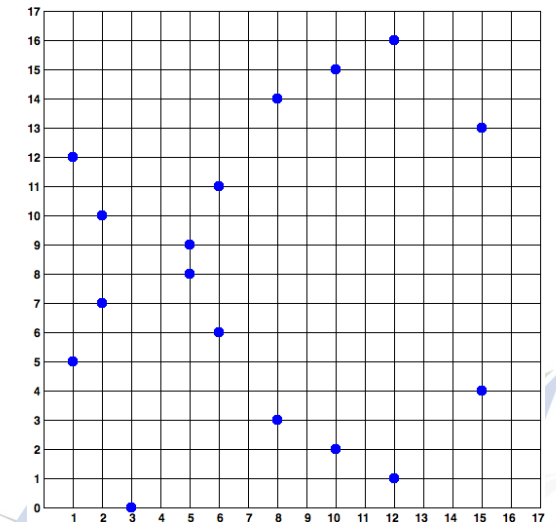


Figure 3. Elliptic curve cryptography: visualizing an elliptic curve over $F(p)$, with $p=17$

KZGコミットメント準備：ペアリング

- BN curveやBLS curveなどいくつかの楕円曲線では、楕円曲線の点同士の乗算に該当するペアリングと呼ばれる演算が可能.
- 具体的には、巡回群 G_1, G_2 と G_T に対して以下の性質を満たす写像 e が取れる：

$$e: G_1 \times G_2 \rightarrow G_T$$
$$e(aP, bQ) = e(P, bQ)^a = (e(P, Q)^b)^a = e(P, Q)^{ab}$$

※ 楕円曲線ごとに、これら G_1 等はよく知られたものがある.

G_1, G_2 はそれぞれある点 $P \in G_1, Q \in G_2$ の加算を繰り返して得られる集合として定義され、それぞれ r 倍すると 0 に該当する点 O になる. すなわち：

$$\begin{aligned} rP &= O, \\ rQ &= O, \\ G_1 &= \{O, P, 2P, \dots, (r-1)P\}, \\ G_2 &= \{O, Q, 2Q, \dots, (r-1)Q\} \end{aligned}$$

同様に G_T はある有限体上の整数 g の乗算を繰り返して得られる集合として定義され r 乗すると 1 になる. つまり：

$$\begin{aligned} g^r &= 1 \\ G_T &= \{1, g, g^2, \dots, g^{r-1}\} \end{aligned}$$

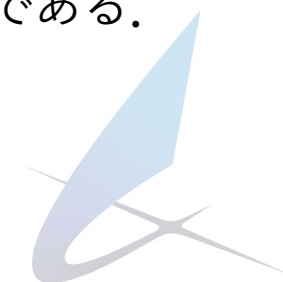
KZGコミットメント：セットアップフェーズ

- Setupフェーズ
 - まずはセットアップを行う。
 - ここでは、誰にも知られてはいけない乱数 s を安全に生成しつつ、KZGコミットメントプロセス全体に必要な値（パブリックパラメータ）を生成する。
 - これは一般にはトラステッドセットアップセレモニーと呼ばれ、複数人で実行される。
 - 各々の乱数をかけて s を作る。そして、自分の乱数を破棄する。
 - 参加者の少なくとも一人が破棄すれば s の秘匿性が守られる。

Setup

$(1^\lambda, n) \rightarrow \text{pp}$:

1. ランダムな s を F_r から取る。
2. $(P, sP, s^2P, \dots, s^nP) \in G_1^{n+1}$ を計算する。ただし P は巡回群 G_1 の生成元である。
3. $(Q, sQ) \in G_2^2$ を計算する。ただし Q は巡回群 G_2 の生成元である。
4. $\text{pp} = ((P, sP, s^2P, \dots, s^nP), (Q, sQ)) \in G_1^{n+1} \times G_2^2$ を出力する。



KZGコミットメント：コミットフェーズ

- セットアップフェーズで得られたパブリックパラメータ pp を用いて、証明者がコミットメントを生成する.
- コミットメントは公開される.
- コミットメントは、多項式に対して一つ.

Commit

$(pp, f(x)) \rightarrow c:$

1. $f(x)$ から係数ベクトル $(f_0, f_1, \dots, f_n) \in \mathbb{F}_r^{n+1}$ を抜き出す.
2. $c \leftarrow \sum_{i=0}^n f_i s^i P$ を計算する.
3. $c \in G_1$ を出力する.

KZGコミットメント：ウィットネスの生成

- 証明者は、コミットメントと評価点を用いて検証用のウィットネスを出力する
- 多項式のコミットメントに対して、評価点ごとにウィットネスを作成する。

Create witness

$(pp, f(x), \alpha, r) \rightarrow (\omega_\alpha, \beta):$

1. $\beta = f(\alpha)$ を計算する。

2. $q(x) = \frac{f(x) - \beta}{x - \alpha}$ を計算する。

3. $q(x)$ から係数ベクトル $(q_0, q_1, \dots, q_n) \in \mathbb{F}_r^{n+1}$ を抜き出す。

4. $\omega_\alpha \leftarrow \sum_{i=0}^n q_i s^i P$ を計算する。

5. $\omega_\alpha \in G_1$ を出力する。

6. β も出力する。

KZGコミットメント：検証

- 検証者は、コミットメント、ウィットネスを用いて検証する。

Verify

$(pp, c, \alpha, \beta, \omega_\alpha) \rightarrow 1/0:$

$e(c - \beta P, Q) = e(\omega_\alpha, sQ - \alpha Q)$ が成り立てば1を、そうでなければ0を出力する。

なぜならば以下のように式変形ができる：

$$\begin{aligned} e(c - \beta P, Q) &= e(\sum_{i=0}^n f_i s^i P - \beta P, Q) & (\because c = \sum_{i=0}^n f_i s^i P) \\ &= e((f(s) - \beta)P, Q) & (\because f(s) = \sum_{i=0}^n f_i s^i) \\ &= e(P, Q)^{f(s) - \beta} & (\because e(aP, bQ) = e(P, Q)^{ab}) \end{aligned}$$

$$\begin{aligned} e(\omega_\alpha, sQ - \alpha Q) &= e(\sum_{i=0}^n q_i s^i P, (s - \alpha)Q) & (\because \omega_\alpha = \sum_{i=0}^n q_i s^i P) \\ &= e(q(s)P, (s - \alpha)Q) & (\because q(s) = \sum_{i=0}^n q_i s^i) \\ &= e(P, Q)^{q(s)(s - \alpha)} & (\because e(aP, bQ) = e(P, Q)^{ab}) \end{aligned}$$

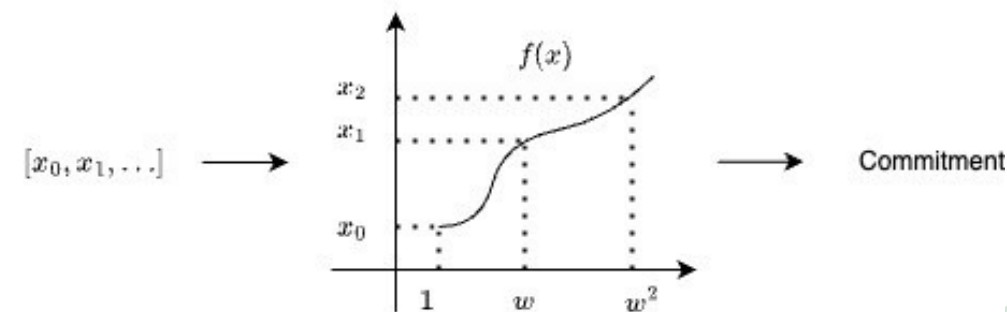
左辺 = 右辺なので、 $f(s) - \beta = q(s)(s - \alpha)$ が成り立つ。
すなわち $f(\alpha) = \beta$ が言える。

KZGコミットメントの利用

- コミットメント中で利用する楕円曲線を選ぶ：
 - ペアリング写像が存在する楕円曲線じゃないと計算できない.
 - ペアリングフレンドリーな曲線が選ばれる
- 適応したい問題に対して, その楕円曲線で計算可能な有限体の元となる係数を設定し, 多項式を作る.
 - G_1, G_2 やペアリング写像 e , 有限体 F_r などは曲線ごとによく使われる効率的なものが存在

Blobデータの生成

- 流れとして、データのバイト列を有限体 F_r の点列 $(x_0, x_1, \dots, x_{4095})$ で表現し直して、そこから多項式 $P(x)$ を作る.
- BLS12-381曲線が使用される.
- Blobは4096個からなる有限体上の元で構成される.
 - 位数 $r =$
52435875175126190479447740508185965837690552500527637822603658699938581184513
 - 255bit. 各値は r 以下.
- Blobデータを表現するために多項式を構築する.
 - Blobの各要素 x_i を用いて多項式 $P(x)$ を構築.
 - $\omega^{4096} = 1 \bmod r$ を満たす整数 ω を使用し (ω^0, x_0) から $(\omega^{4095}, x_{4095})$ を通過する4095次の多項式 $P(x)$ を作る.



<https://zenn.dev/qope/articles/b8d09ae260f1aa>

1つのBlobで取り扱えるデータ容量は
$$\frac{4096 \times \log_2 r}{8 \times 1024} = 127.4 \dots KB$$

Blobデータの構成

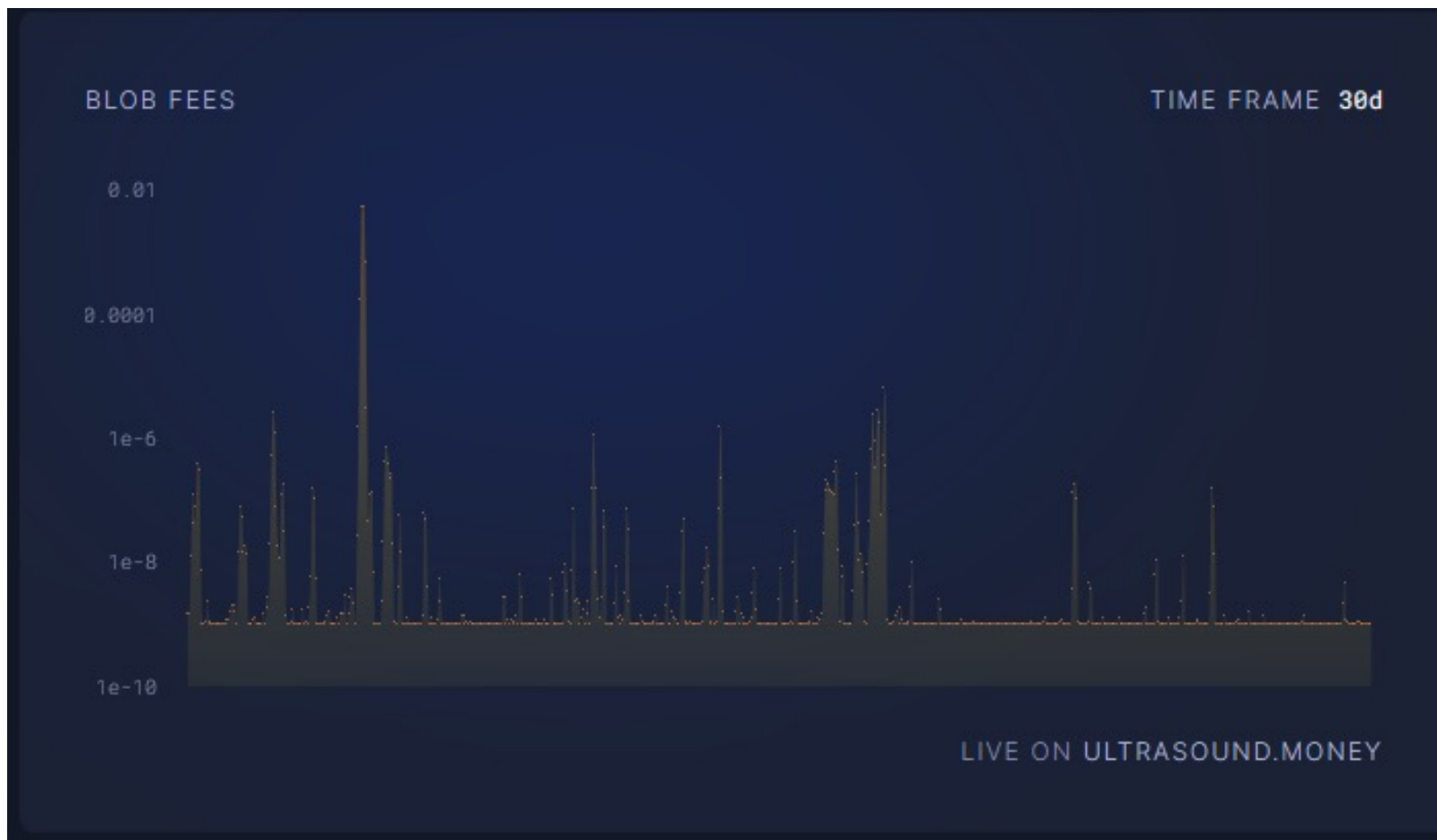
- Blobの約127KBのデータから多項式を構成し、KZGコミットメントを計算する。そして、とある点 α で評価したウィットネスも同時に出力しておく。
 - ※ KZGコミットメント値はkzg, ウィットネスはproofと表現されている。
 - 評価点 α はランダムな値（だけどBlobデータとコミットメントから決定論的に導出可能）。
- つまりBlobデータは以下の3つで構成される。
 - Blob
 - kzg
 - Proof
- これがチェックされ、伝搬されていく。

Blob Carrying Transaction(Shared Blob Transaction)の導入

- Ethereumの通常のTxの形式が、 Blobのデータを取り扱えるように拡張されたもの.
- Txのフィールドに以下の2つが追加：
 - max_fee_per_blob_gas
 - これはsenderがBlobに対して支払う意思のある最大ガス代.
 - blob_version_hashes
 - 複数の versioned_hash.
- Versioned_hashというのが、 KZGコミットメントのSHA256ハッシュ値の、 最初の1バイトを0x01に書き換えた32byteの値.



ガス料金



<https://ultrasound.money/#blobs>

まとめ

- Optimistic Rollupの仕組みをArbitrum Oneの事例をもとに理解できた.
 - Txはcalldataに
 - シーケンサー, バリデータ
 - フラウドプルーフ
- Proto-danksharding
 - Ethereumプロトコル上の, 安くデータを保管できる領域.
 - ダンクシャーディングの一手手前の実装.
 - KZGコミットメントを使用.



-
- 本スライドの著作権は、東京大学ブロックチェーンイノベーション寄付講座に帰属しています。 自己の学習用途以外の使用、無断転載・改変等は禁止します。