00 自己紹介

Solana財団のExternal Devrel Japanのユウキです。（本名：高橋　祐貴）

また、CACのブロックチェーン推進グループで働いています。

2021年に市役所の保育課勤務からブロックチェーン業界にエンジニアとして転職し、現在は５つの保育園を運営する社会福祉法人の監事も務めています。

レンディングプロトコル入門

Solana財団 External Devrel Japan
Yuki Takahashi

00 免責事項

- 本資料は技術解説目的であり、投資を勧めるものではありません。
- DeFi の利用には価格変動・スマコンのバグ等のリスクが存在します。
- 投資判断はご自身の責任で行なってください。
- 内容は発表時点の情報です。(正確性は保証しません。)

01 今日話すこと

本日の概要

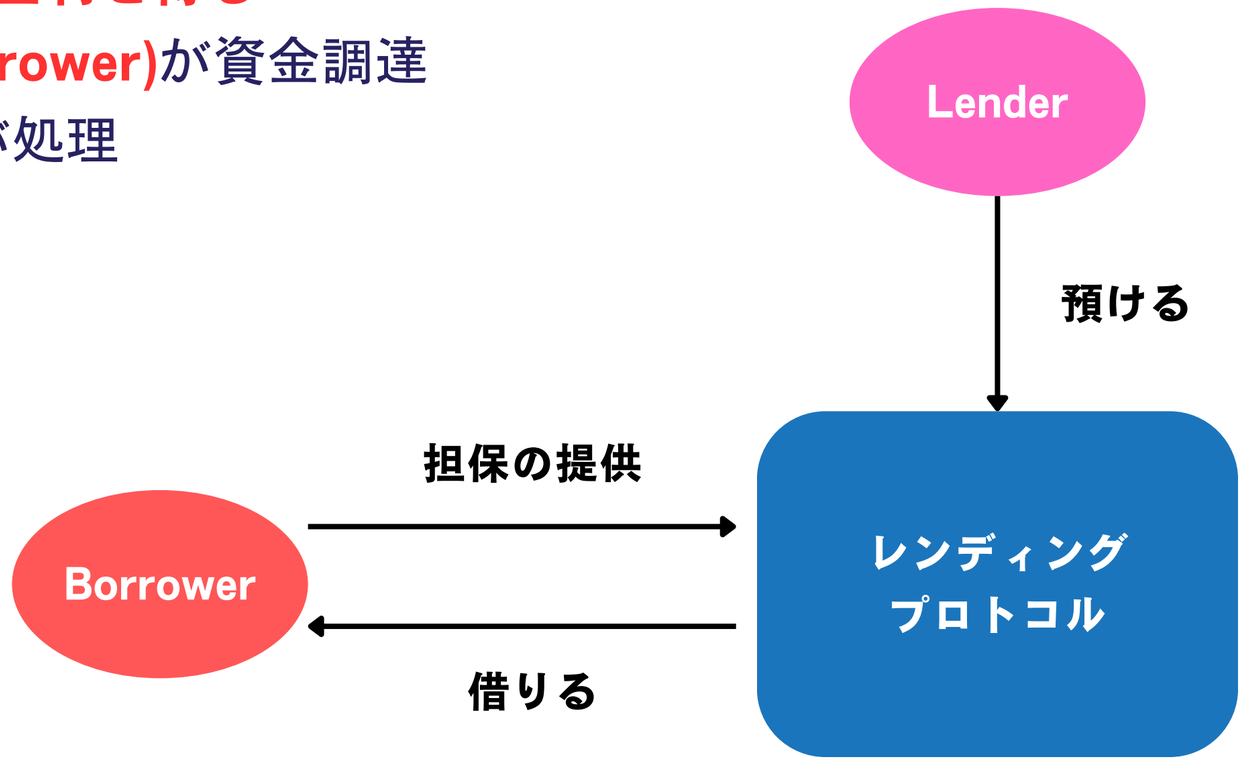
1. レンディングプロトコルとは何か
2. 登場人物は3人 — 預入者・借入者・清算者
3. Solana特有のアカウント構造(Bank / User / Token Account)
4. 預ける・借りる・清算する のデータの動き
5. オラクルとリスクパラメータ

「Solanaだとどうデータが置かれているのか」が今日のメインです。

02 レンディングプロトコルとは

「銀行をスマートコントラクトに置き換えた仕組み」

- 余っている**資産を預ける人(Lender)が金利を得る**
- 担保を入れて**別の資産を借りる人(Borrower)**が資金調達
- 全部 Solana のプログラム(スマコン)が処理



03 登場人物は3人

3人目の「清算者」がいることが重要

役割	やること	インセンティブ
預入者	資産を供給	金利収入
借入者	担保を入れて借りる	レバレッジ・流動性確保
清算者	**危ないポジションを処理**	**清算ボーナス**

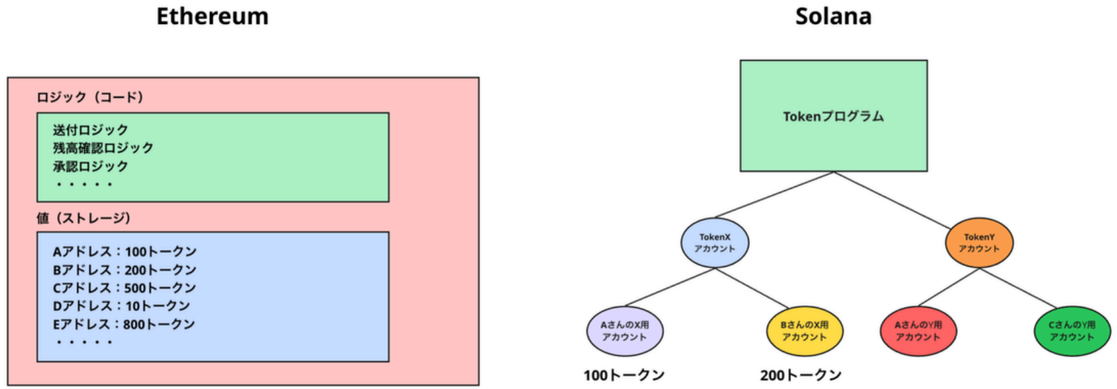
清算者がいないとプロトコルが破綻する。今日の主役の一人。

04 Solanaでは「すべてがアカウント」

ここからSolana 特有の話に入ります。

EVM(Ethereum系)との違い

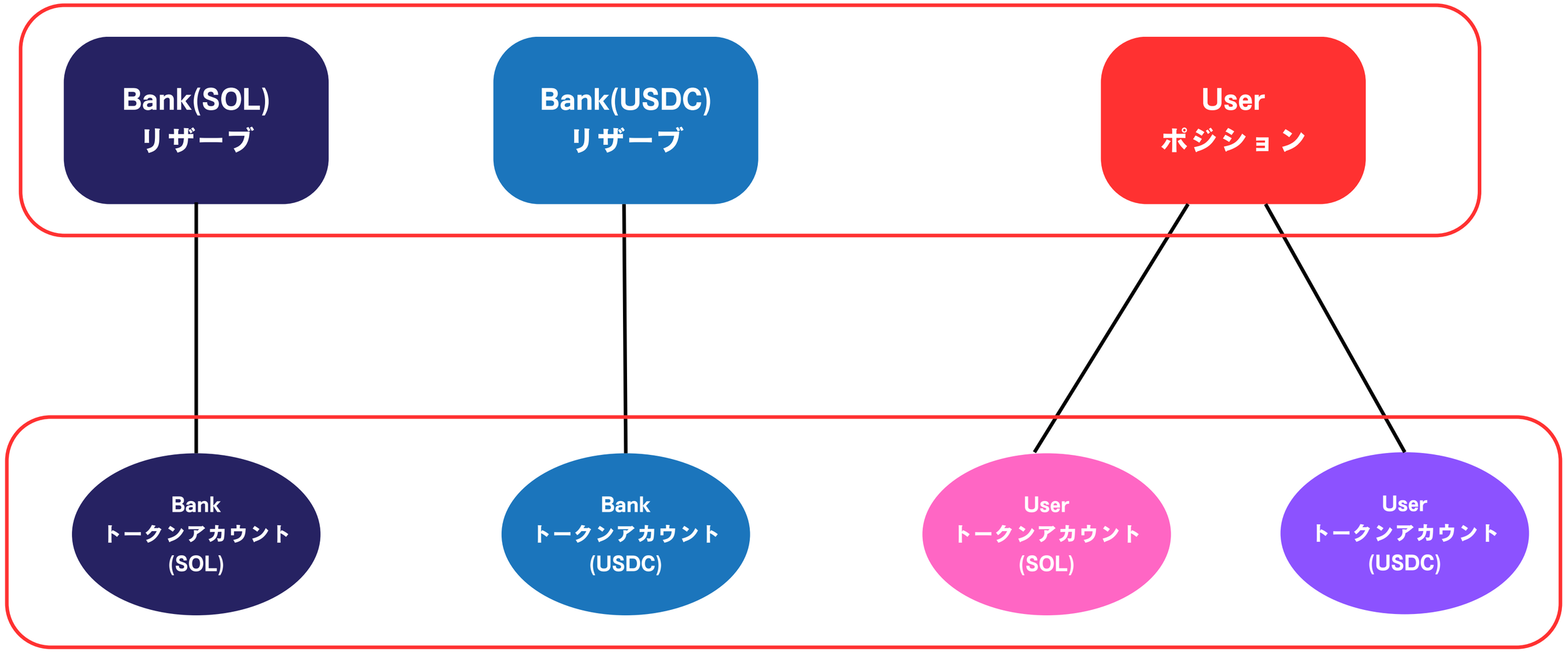
- EVM:コントラクトが内部ストレージに状態を持つ
- Solana:プログラムは状態を持たず、全部「アカウント」に切り出す



レンディングプロトコルも、**「どんなアカウントを設計するか」**がプロトコルの設計そのもの になります。

05 登場するアカウント(全体像)

預かり額などの情報（帳簿）



実際のトークン（金庫）

06 Bankアカウントの中身

資産ごと(SOLとUSDCで別)に1つ存在する「プールの司令塔」

```
Bank(SOL)                    ← seeds: mint

┌──────────────────────────────────────┐
│ 1  authority                         │ ← 管理者
│ 2  mint_address                      │ ← どの資産か
│ 3  total_deposits                    │ ← 預入総額
│ 4  total_deposits_share              │ ← 預入シェア総数
│ 5  total_borrowed                    │ ← 借入総額
│ 6  total_borrowed_shares             │ ← 借入シェア総数
│ 7  liquidation_threshold             │ ← 清算閾値(例:85%)
│ 8  liquidation_bonus                 │ ← 清算ボーナス(例:5%)
│ 9  liquidation_close_factor          │ ← 一度に清算できる割合
│ 10 max_ltv                           │ ← 借入可能上限(例:75%)
│ 11 last_updated                      │ ← 最終更新時刻
│ 12 interest_rate                     │ ← 金利
└──────────────────────────────────────┘
```

メタ情報

リスク
パラメータ

プールの状態とリスクパラメータが同じ場所に置かれる。

07 リスクパラメータとは?

「貸し倒れを防ぐための4つのつまみ」
プロトコル管理者が資産ごとに設定する安全装置。

① **max_ltv** — どこまで借りられるか
② **liquidation_threshold(清算閾値)** — どこから清算されるか
③ **liquidation_bonus(清算ボーナス)** — 清算者へのご褒美
④ **liquidation_close_factor(清算クローズファクター)** — 一度にどこまで処理するか

08

リスクパラメータとは?
① max_ltv — どこまで借りられるか

① max_ltv — **どこまで借りられるか**

Loan-to-Value:担保価値に対する借入上限

```
担保: 100 USDC相当のSOL
max_ltv = 75%
→ 借入可能額の上限: 75 USDC
```

「借りすぎ」を入口で防ぐ門番。 ここを超える借入は Tx が失敗する。

09 リスクパラメータとは? ② liquidation_threshold — どこから清算されるか

② liquidation_threshold — **どこから清算されるか**

清算が発動する境界線

```
[当初]
担保: 1 XYZ($100相当)を預けて 70 USDC 借りた
借入比率: 70 / 100 = 70%  → 安全圏

[XYZ価格が $85 に下落]
担保価値: $85
借入比率: 70 / 85 = 82.4%  → まだセーフ

[XYZ価格が $80 に下落]
担保価値: $80
借入比率: 70 / 80 = 87.5%  → 清算閾値 85% を超過!
→ 清算可能
```

max_ltv(75%)< liquidation_threshold(85%) の差が**安全マージン**。価格変動に耐えるバッファ。

10 リスクパラメータとは? ③ liquidation_bonus ④ liquidation_close_factor

④ liquidation_close_factor — **一度にどこまで処理するか**

```
[借入残 70 USDC のポジション]
liquidation_close_factor = 50%
→ 1回の清算で最大 35 USDC まで返済可能
→ 残り 35 USDC は借入者に残る
```

③ liquidation_bonus — **清算者へのボーナス**

```
[XYZ価格 $80 まで下落、借入 70 USDC のポジション]
清算者が 35 USDC を肩代わり返済
liquidation_bonus = 5%
→ 36.75 USDC相当の XYZ(≒ 0.459 XYZ)を受け取る
→ 差額 1.75 USDC が清算者の利益
```

11 パラメータ設計のジレンマ

「ユーザーが嬉しい設定」と「プロトコルが安全な設定」は逆を向く

パラメータ	緩める (↑)	厳しくする (↓)
max_ltv	借入者: 嬉しい	プロトコル: 安全
清算閾値	借入者: 猶予あり	プロトコル: バッドデット減
清算ボーナス	清算者: 嬉しい	借入者: 損失小
close_factor	清算者: 効率的	借入者: ソフトランディング

→ プロトコル運営は3者のバランスで値を決める綱渡り。市場環境に応じてガバナンスで調整されることもある。

12 なぜ「shares」が出てくるのか?

もしユーザーごとに金額を直接記録すると**金利が乗るたびに全ユーザーの残高を書き換える必要**がある。

10万人いたら10万件の更新 → Solanaでは現実的に不可能。
そこで「取り分の口数」で記録する。

```
Bank(SOL)                ← seeds: mint
┌──────────────────────────────────┐
| 1  authority                    | ← 管理者
| 2  mint_address                 | ← どの資産か
| 3  total_deposits               | ← 預入総額
| 4  total_deposits_share         | ← 預入シェア総数
| 5  total_borrowed               | ← 借入総額
| 6  total_borrowed_shares        | ← 借入シェア総数
| 7  liquidation_threshold        | ← 清算閾値(例:85%)
| 8  liquidation_bonus            | ← 清算ボーナス(例:5%)
| 9  liquidation_close_factor     | ← 一度に清算できる割合
| 10 max_ltv                      | ← 借入可能上限(例:75%)
| 11 last_updated                 | ← 最終更新時刻
| 12 interest_rate                | ← 金利
└──────────────────────────────────┘
```

13 投資信託の「口数」と同じ仕組み

[投資信託]
あなた:10口 保有
基準価額:10,000円 → 10,500円に上昇
→ 資産:10口 × 10,500円 = 105,000円
口数は変わらない。**基準価額が上がるから資産が増える。**

投資信託	レンディング
口数	Shares
基準価額	Share単価（USDC換算レート）
収益の反映	金利でShare単価が上昇

share は USDC ではなく、プール独自の単位(初期値だけ 1 share = 1 USDC)。

14 数字で追ってみる

share単価 = 1.00 USDC
Aさん.shares = 100

1年経過
金利5%が乗る

share単価 = 1.05 USDC
Aさん.shares = 100

← 単価が上がる

← Aさんの記録は変わらない

Aさんが引き出すとき
引き出し額 = 100 shares × **1.05** = 105 USDCとなる。

15 Userアカウントの中身

担保・借入・健全性をまとめて持つ

User

1	owner	← ウォレットアドレス
2	deposited_sol	← 預けたSOL量
3	deposited_sol_shares	← SOL預入シェア
4	borrowed_sol	← 借りたSOL量
5	borrowed_sol_shares	← SOL借入シェア
6	deposited_usdc	← 預けたUSDC量
7	deposited_usdc_shares	← USDC預入シェア
8	borrowed_usdc	← 借りたUSDC量
9	borrowed_usdc_shares	← USDC借入シェア
10	usdc_address	
11	health_factor	← ポジションの健全性
12	last_updated	

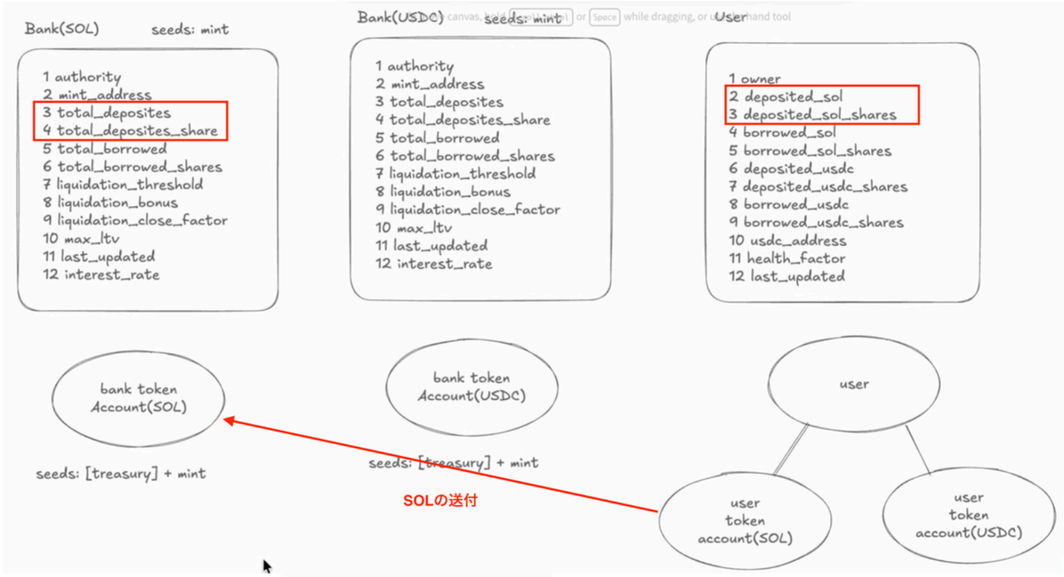
health_factor が 1.0 を割ると清算可能になる、というのが鍵。

16 預ける(Deposit)時の動き

ユーザーが SOL を預けるとき

1. ユーザーの user token account(SOL) から → bank token account(SOL) へ送金
2. Bank(SOL) の **total_deposits を増加**
3. シェア単価から発行シェア数を計算 → **total_deposits_share を増加**
4. User の **deposited_sol と deposited_sol_shares を更新**

1回の Tx で 4つのアカウントが更新される

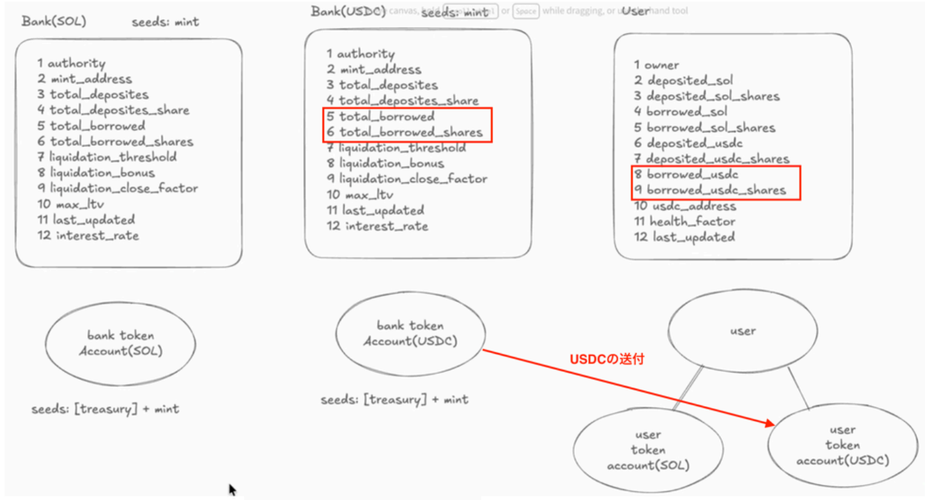


17 借りる(Borrow)時の動き

担保がある状態で USDC を借りたいとき:

1. **借入可否チェック**
   - ~~借入後の health_factor > 1.0 か?~~
   - 担保価値 × max_ltv ≧ 借入合計価値 を満たすか
2. Bank(USDC) の **total_borrowed、total_borrowed_shares** を増加
3. bank token account(USDC) から → user token account(USDC) へ**送金**
4. User の **borrowed_usdc と borrowed_usdc_shares を更新**

ここで オラクル価格が必要



18 オラクルが命綱

価格はどこから来るのか?

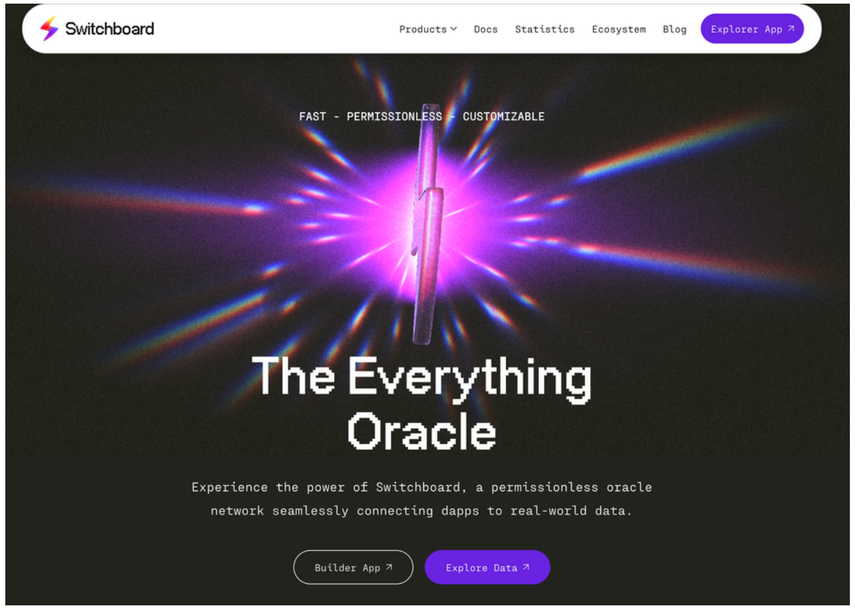
Solana の代表的なオラクル:

- Pyth Network:Solana ネイティブ。低レイテンシで定評
- Switchboard:汎用オラクル

借入可否チェックをするには**担保価値の取得**が必要。
その価格はオラクルの専用アカウントから読み取る。
オラクルが遅延・誤動作すると…

- 本来清算されるべきでないポジションが清算される
- 逆に放置されてバッドデット(回収不能債権)が発生

→ レンディングプロトコルの安全性は**オラクル品質に依存**



https://switchboard.xyz/

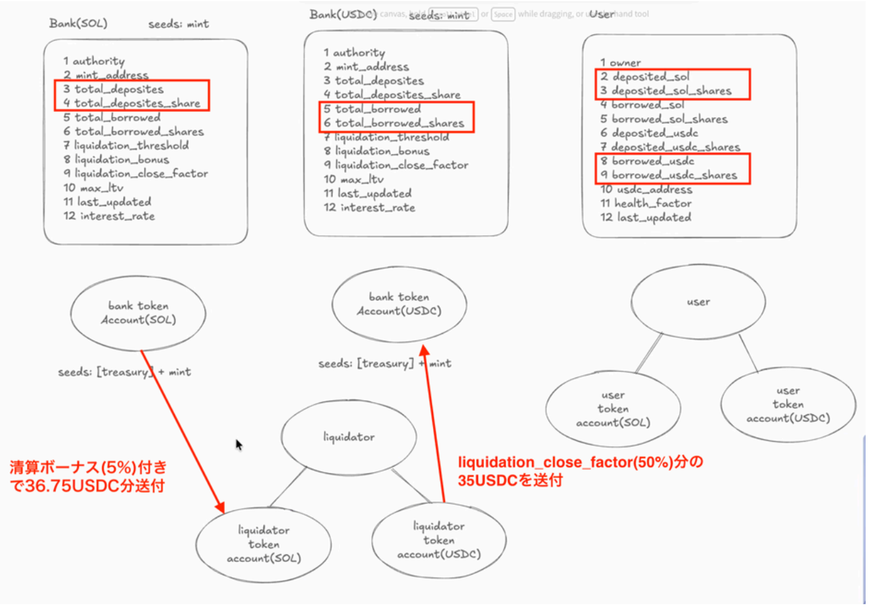
19 清算(Liquidation)— ここが見せ場

ポジションが危険水域(health_factor < 1.0)に入ったとき

清算者(liquidator)の動き:

1. 自分の **liquidator token account(USDC) から**借金の一部を肩代わり返済
2. liquidation_close_factor(例:50%)まで一度に処理可能 → **35 USDC返済**
3. **清算ボーナス5%付きで SOL 担保を受領** → 36.75 USDC 相当の SOL
4. 差額 **1.75 USDC が清算者の利益**

借入者の borrowed_usdc と deposited_sol が両方減る。



20 パラメータ設計のジレンマ（再掲）

「ユーザーが嬉しい設定」と「プロトコルが安全な設定」は逆を向く

パラメータ	緩める (↑)	厳しくする (↓)
max_ltv	借入者: 嬉しい	プロトコル: 安全
清算閾値	借入者: 猶予あり	プロトコル: バッドデット減
清算ボーナス	清算者: 嬉しい	借入者: 損失小
close_factor	清算者: 効率的	借入者: ソフトランディング

→ プロトコル運営は3者のバランスで値を決める綱渡り。

→ 市場環境に応じてガバナンスで調整されることもある。

21 Solanaは「緩める」方向に有利

清算が速く確実なほど、パラメータを攻めた設定にできる

パラメータ	**緩める (↑) = 攻めた設定**	**厳しくする (↓) = 守りの設定**
max_ltv	**Solana 向き**	他のチェーン
清算閾値	**Solana 向き**	他のチェーン
清算ボーナス	厚くしなくて済む(小さくてOK)	厚くしないと清算者が来ない
close_factor	一気に処理OK	慎重に処理する必要

→ Solanaのレンディングは同じリスクレベルで資本効率が高い設定が可能。

P-TOKENの成功と今後の可能性

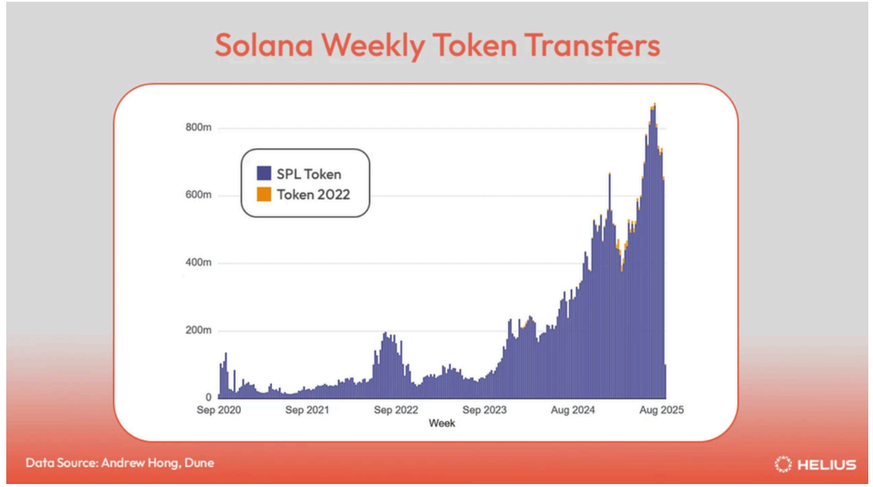
Solana財団 External Devrel Japan
Yuki Takahashi

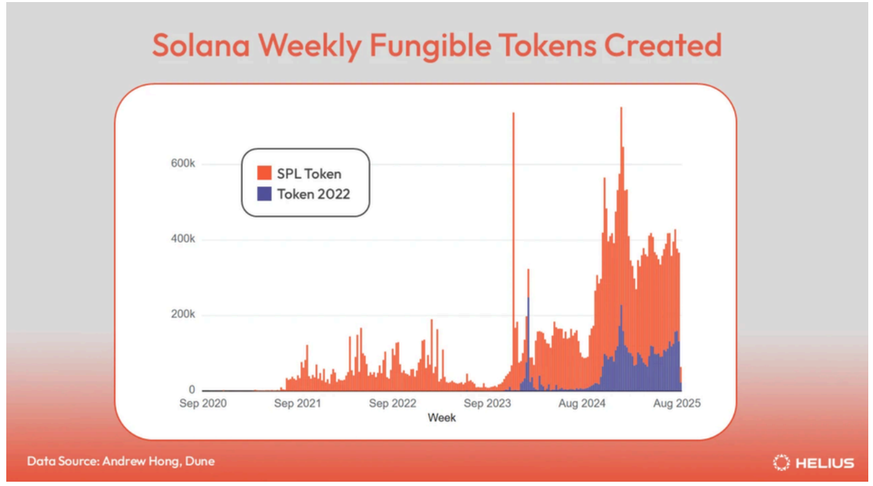
01 Solana におけるSPL Tokenの重要性

Solana 上で最も使われているプログラムは、System Programを除けばSPL Tokenプログラムです。

- 週あたり20〜30万個 の新規トークンが発行
- **週あたりの送金回数は6.5億回超**
- Solana のブロック全体の CU 消費のうち、約10% をトークンプログラムが占める

新規トークン発行は 2年前と比べて約20倍、送金回数は 約17.5倍 に急増しており、Solana の中核となるプログラムです。





https://www.mki.co.jp/knowledge/innovation/column77.html

02 なぜインパクトが大きいのか

P-Token は、SPL Token プログラムを置き換える、新しい軽量版トークンプログラム です。

ポイントは、これが**「新しいトークン規格」ではない**ということです。

命令セットもアカウント構造も既存のSPL Tokenと完全に同じ で、**内部実装だけが最適化**されています。

ユーザーや開発者側で対応する必要はなく、**ある時点から既存の全 SPL トークンが自動的に**高速・低コストになる、という性質のアップグレードです。

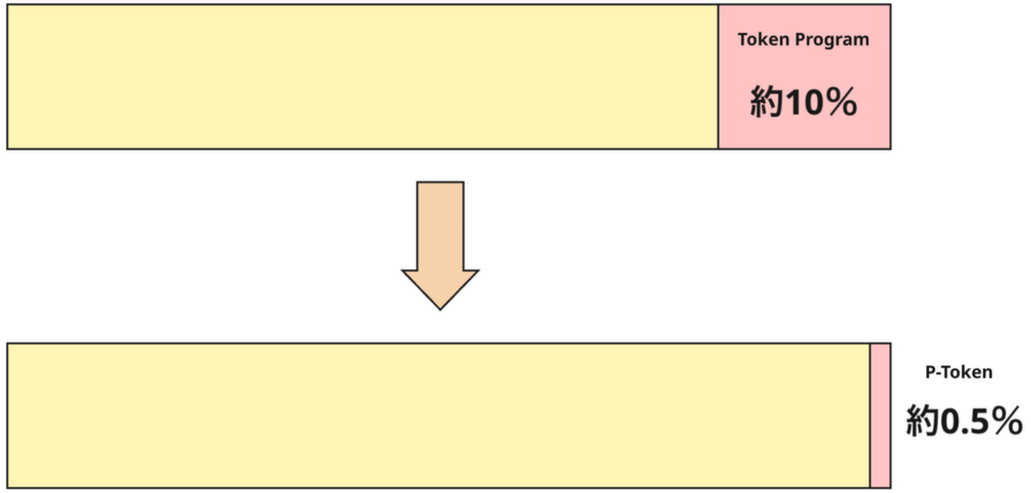
https://www.mki.co.jp/knowledge/innovation/column77.html

03 CU削減の効果

P-Token に置き換わると、Solanaのブロック全体の
CU 消費に占めるトークンプログラムの割合が
約10%から約0.5%まで下がる試算です。

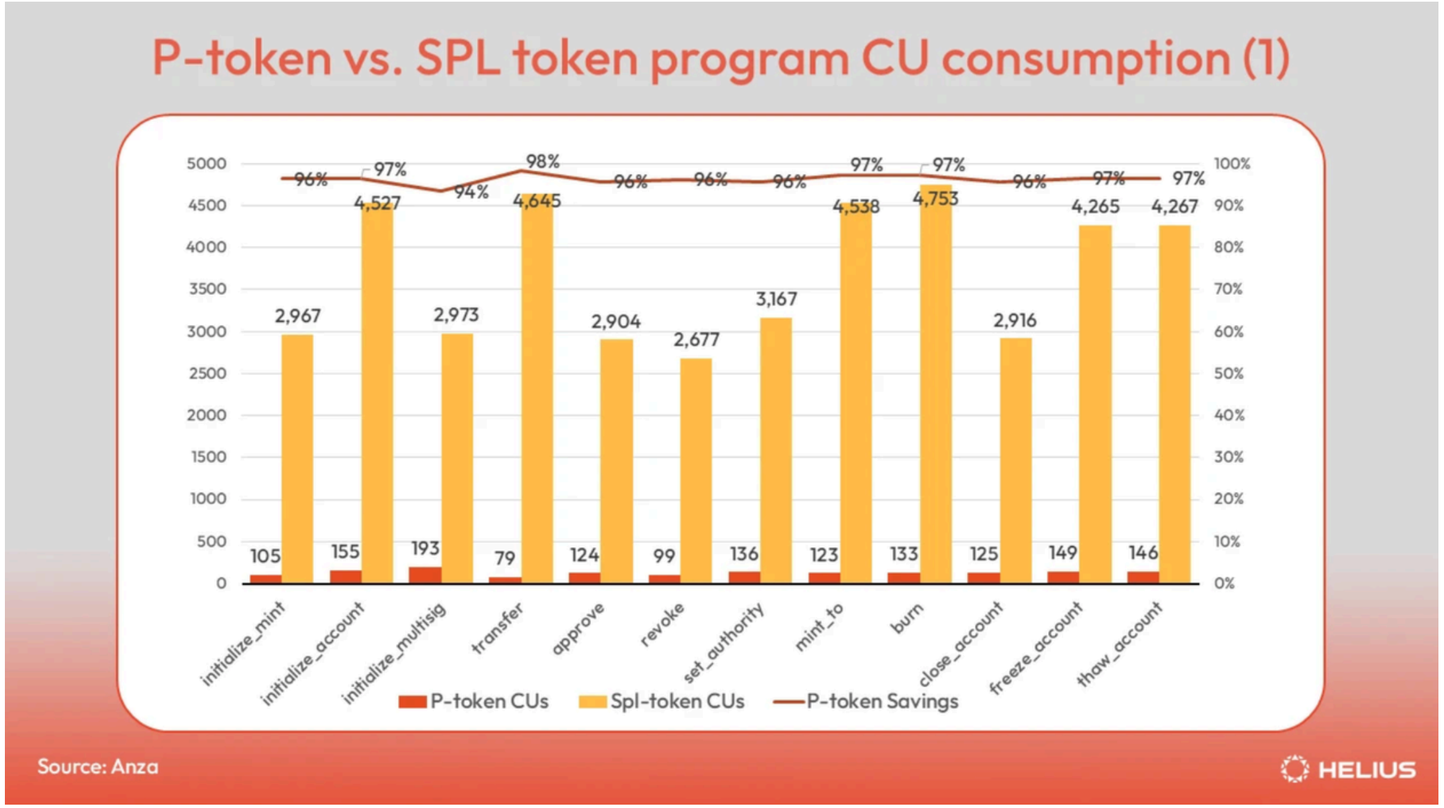
結果として、**ブロック容量の約9.5%が他の用途に解放**されます。

これは ネットワーク全体のスループット向上に直結する話で、**最も使われているプログラムが軽くなることで、Solana全体のキャパシティが底上げ**されます。



04 効率化の具体例

具体的な命令ごとの CU（Compute Unit）削減を見ると、効率化の規模が一目で分かります。



https://www.mki.co.jp/knowledge/innovation/column77.html

05 技術的な肝「Pinocchio」

「P」は Pinocchioという Anza チーム開発のライブラリの頭文字で、ここに削減の核心があります。

1　ゼロコピー設計

通常の Solana プログラムは、データを読み書きする際にメモリ上にコピーを作りますが、**Pinocchioはポインタで直接データを参照**するため、メモリ操作のオーバーヘッドが消えます。

2　no_std（標準ライブラリ非依存）

Rustの標準ライブラリやヒープ割り当てに依存しません。

SVM がランタイムを提供している ため、標準ライブラリは不要、という発想です。

https://solana.com/developers/templates/pinocchio-counter

06 新しく追加される3つの命令

P-Token には、既存にはなかった命令も追加されます。

① withdraw_excess_lamports

ミントアカウントに誤って送られロックされたSOLを救出する命令です。
現在、約87万個のミントアカウントに合計 17.6万 SOLが眠っており、これが解放可能になります。

② batch

複数のトークン命令を 1回の CPIでまとめて実行する命令です。
CPI には**基礎コスト 1,000 CU がかかる**ため、AMM のスワップや流動性プールへのデポジットなど、複数操作を伴う DeFi で特に効果 を発揮します。

③ unwrap_lamports

Wrapped SOL を**アンラップする際**、これまで必要だった **一時的な ATA の作成・クローズを省略**し、直接 lamports を送れるようになります。

07 エレガントな移行プロセス

新アドレスに別途デプロイしてユーザーに移行を促す方式 だと、採用が進まず効果が薄れてしまいます。

そこで採用されたのが、**機能ゲート（Feature Gate）を使った一斉切り替え**です。

エポック境界で機能ゲートが起動した瞬間、全バリデータのランタイムが**既存のトークンプログラムの中身を P-Token に置き換える**、という流れになります。

つまり、**ユーザーも開発者も何もする必要はなく**、ある日突然 Solana 全体がトークン操作で軽くなる、という仕組みです。

https://x.com/solana_devs/status/2054318104822300757

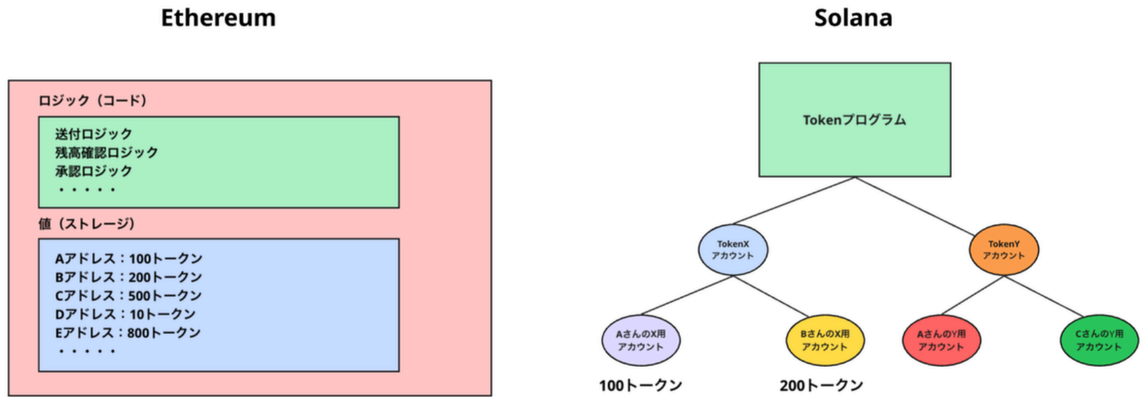
08 これはSolanaだからこそ可能

コードと状態の分離

Solanaでは、**プログラムとアカウントが完全に分離** されています。
全 SPL トークンの状態は Tokenプログラムの外側に保存されているため、**プログラム本体を置き換えても既存のトークンデータはそのまま使え**ます。

他チェーンとの違い

Ethereum などの コントラクト内に状態を持つモデル では、各トークンが個別のコントラクトとしてデプロイされているため、こうした 一括置き換えはそもそも不可能 です。



09 そしてその先へ

今回の成功により、**ATA プログラムや System Program といった他のコアプログラム**も順次置き換えていく青写真 が描かれています。

① 効率的な CU 消費に変えられる
内部実装を最適化することで、**ネットワーク全体のスループット向上に直結する効率化**が可能です。

② ユーザーの手間はない
機能ゲートによる一斉切り替えで、**ユーザーや開発者は何もする必要がなく、**ある日突然、全 SPL トークンが軽くなる という体験になります。

③ 機能も新しく追加できる
今回のように、完全な下位互換を保ちつつ、**新しい命令を追加することも可能**です。

量子コンピューティングと SOLANAの方向性

Solana財団 External Devrel Japan
Yuki Takahashi

00 免責事項

本資料の内容は 2025年12月時点の情報 に基づいて作成されています。

量子コンピュータおよびポスト量子暗号の分野は 進化が非常に速く、本資料の公開後に状況が変化している可能性があります。
最新の動向については、各自で一次情報をご確認ください。

また、発表者は 本テーマについて学習はしているものの、量子暗号・暗号理論の専門家ではありません。

本資料は学習・共有を目的としたものであり、内容には誤りや不正確な記述が含まれる可能性があります。
技術的判断や実装にあたっては、専門家の見解や原典をご参照ください。

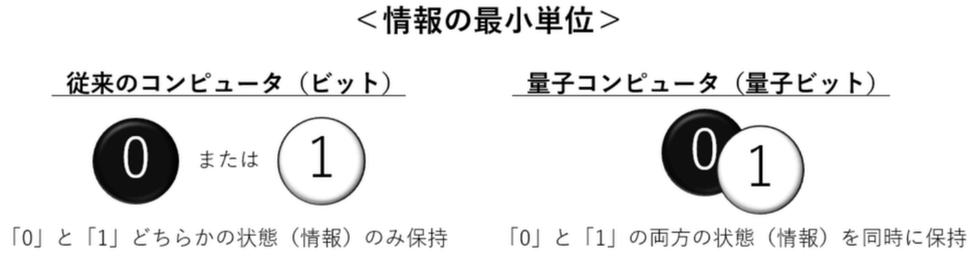
01 量子コンピュータとは何か

量子コンピュータは、0と1で処理する古典コンピュータとは異なり、**「量子ビット（qubit）」** を使う計算機です。

量子ビットは **「重ね合わせ」** により複数の状態を同時に取れるため、多数の解を並列に探索できます。

ただし すべての計算が速くなるわけではなく、**特定の問題でのみ威力を発揮**します。

現状は制約が多く、実用的な暗号解読には要件に対して数桁（100～1000倍以上）足りない水準です。



https://www.mki.co.jp/knowledge/innovation/column77.html

02 暗号を脅かす2つのアルゴリズム

量子コンピュータが暗号にもたらす脅威は、主に2つのアルゴリズムから来ます。

① Shor のアルゴリズム（致命的な脅威）
RSA・Diffie–Hellman・**楕円曲線暗号（Solana の Ed25519 を含む）**を効率的に破ります。
公開鍵から秘密鍵を逆算できる ため、誰でも他人のトランザクションに署名できてしまい、
Solana にとって致命的です。

② Grover のアルゴリズム（限定的な脅威）
総当たり探索を N回 → √N回 に高速化するアルゴリズムです。
これにより SHA-256 の安全性は 256ビット → 128ビット に低下しますが、**128ビットは依然として現実的な攻撃の射程外**です。

03 ポスト量子暗号という解決策

量子攻撃下でも安全な暗号方式を作る分野が **「ポスト量子暗号（PQC）」** です。

NIST は2つの署名方式を標準化しています。

① ML-DSA
② SLH-DSA

方式 (Scheme)	公開鍵サイズ (Public key size)	署名サイズ (Signature size)	セキュリティ (Security)	耐量子対応 (PQ ready)
Ed25519	32 B	64 B	128 ビット	いいえ
ML-DSA	1312 B	2560 B	128 ビット	はい
FN-DSA	897 B	666 B	128 ビット	はい
SLH-DSA	64 B	7856 B	128 ビット	はい

PQ方式の最大の課題は**サイズが大きいこと**です。

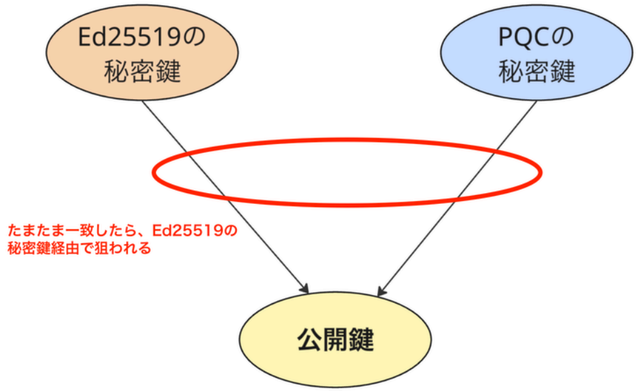
Ed25519 が切手なら、SLH-DSA は段ボール箱ほどの差です。

毎秒数千件を処理するSolana にとって、**署名サイズの肥大化は 帯域・ストレージ・検証コストすべてに直撃**します。

04 Solana が向き合う具体的な課題（1/3）
アドレスと署名

Solana は 「Ed25519 の公開鍵（32B）＝アドレス」 というシンプルな設計ですが、PQ では公開鍵が巨大になるため維持できません。

従来：　公開鍵（32B）= アドレス
PQ後：ハッシュ(PQ公開鍵 + 方式識別子)（32B）= アドレス



1　衝突回避の問題

PQ アドレスのハッシュが Ed25519 曲線上の有効な点と一致すると、
「PQ で守っているはずなのに古典の Ed25519 で奪われる」 事態が起きます。
PDA の bump seed と同じ発想で対処が必要です。

2　権限鍵もすべて Ed25519

アカウント所有者、ミント・アップグレード権限、バリデータID鍵、投票権限鍵など、Solana のあらゆる権限機構が Ed25519 で動いており、量子の脅威下では一斉に影響を受けます。

05 Solana が向き合う具体的な課題（2/3）Alpenglow Votor①

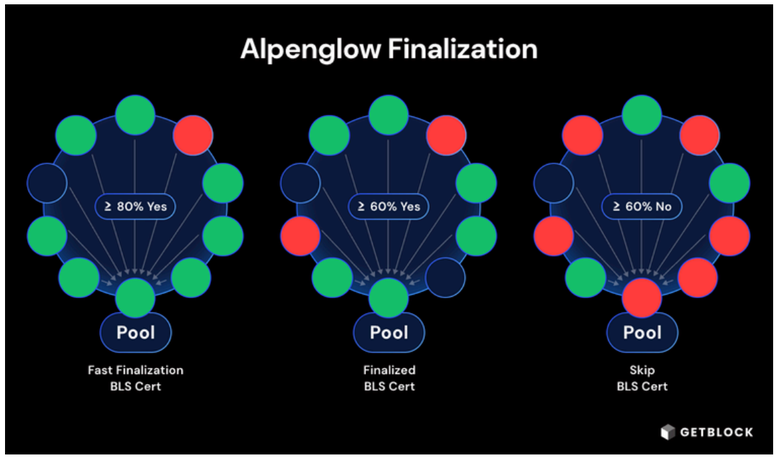
Votor は、**バリデータが全員対全員で投票を交換**し、定足数（2/3）に達したら「証明書」を生成する仕組みです。

これを支えているのが **BLS集約署名**です。

何百～何千もの個別署名を 数学的に足し合わせて1つにまとめることで、**賛成票を1枚に圧縮し、ネットワーク帯域を圧迫しません。**

しかし BLS は楕円曲線ペアリングに依存しているため、**Shorで破られます。**

現状、**BLS 集約に相当する実用的な PQ 方式は存在せず、**研究レベルの代替案も Solana のリアルタイム要件には追いつきません。



https://getblock.io/blog/what-is-solana-alpenglow-a-getblock-review/

06 Solana が向き合う具体的な課題（2/3）Alpenglow Votor②

BLS が使えなくても、Votor を完全に作り直す必要はありません。
配信戦略を変えるだけで性能を許容範囲に保てます。

具体的には：

- 完全な証明書を 全ピアにブロードキャストしない
- **次の1〜2人のリーダーとステーク全体の小さなランダムサブセットにだけ送信**
- そこから ゴシップ的に自然伝播させる

帯域消費は増えますが、Alpenglow 導入前のゴシップ層と同程度 に収まります。

過去の Solana が捌けていた水準なので、現実的に成立可能な範囲です。

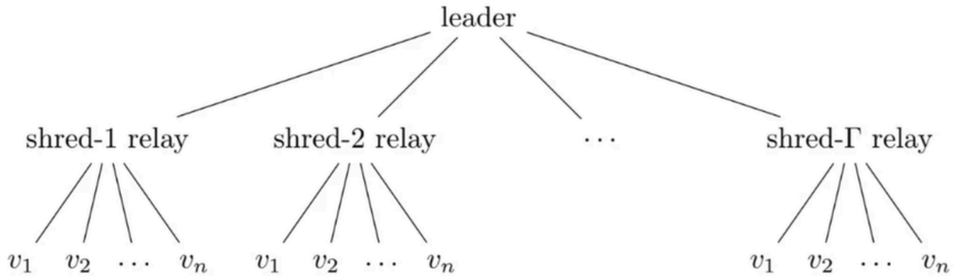
07 Solana が向き合う具体的な課題（3/3）
Alpenglow Rotor①

Rotor はブロック配信プロトコル で、リーダーがブロックを**「スライス」**に分割し、さらに MTUサイズ（約1500B）の**「シュレッド」** に分けてツリー状にリレーします。

ここで効いてくるのが**FEC（前方誤り訂正）**です。

一部のシュレッドが届かなくても元のブロックを復元できる技術です。
しかし、シュレッドごとに署名できない問題があります。

各シュレッドには改ざん検出用にリーダーの署名が必要ですが、**Ed25519 は64Bと小さく問題ない**一方、PQでは署名だけでMTU上限を超えてしまうため、シュレッド本体を入れる余地がなくなります。



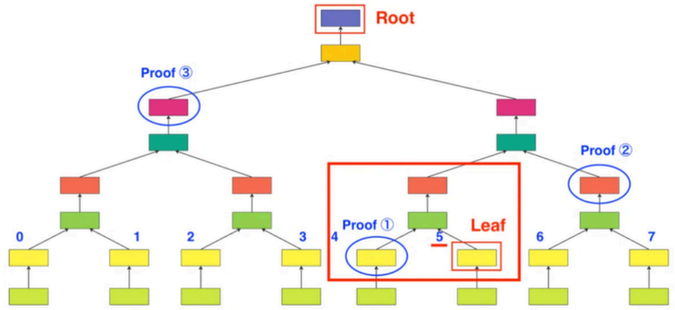
https://journalducoin.com/solana/alpenglow-changement-majeur-algorithme-consensus-solana/

08 Solana が向き合う具体的な課題（3/3）Alpenglow Rotor②

解決策：スライス単位で1署名

シュレッド1つずつに署名できないなら、スライス全体でまとめて1署名 にする方式です。

- スライス内の全シュレッドから Merkle 木 を作る
- **リーダーはルートにだけ署名**
- **各シュレッドには軽い 「Merkle 証明」 を付ける**
- **バリデータはルート署名と Merkle 証明でシュレッドの正当性を検証**



Amazon の発送に例えると、1000個の小包に重いシール1個を共有させ、各小包には「私はロットの537番目です」という軽い証明書を添えるイメージです。

Solana は既に Merkle 構造でシュレッドを束ねているため、ゼロからの再設計ではなく既存の延長線で実現可能 です。

09 よくある誤解①「全部壊れるならブロックチェーンだけ気にしても仕方ない」

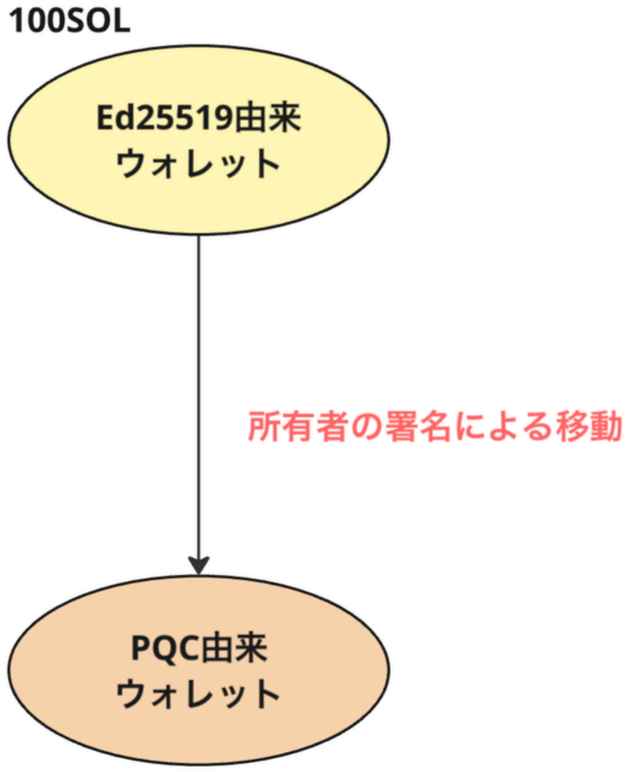
実は逆で、中央集権的なシステムより**ブロックチェーンの方が移行は難しい**と考えられます。

銀行や政府などは、管理者が内部で 鍵をローテーション・インフラ更新・通信路を強制すれば移行できます。

一方、パブリックブロックチェーンでは**何百万人ものユーザー全員が自分で移行トランザクションに署名する必要**があります。

しかも その署名に使う秘密鍵こそが脆弱になりつつある古い鍵であり、攻撃者に移行の瞬間を狙われるリスクすらあります。

つまり移行は、**分散型システムにおいてこそ複雑で切迫する課題**なのです。



10 よくある誤解②「量子コンピュータの実用化はもう目前」

実際にはまだかなり遠いと考えられています。

Shorのアルゴリズムによる素因数分解の実例は、**21（=3×7）や35 などの極小の数**にとどまっています（2025年時点）。

RSA-2048 を破るには 600桁以上の合成数 を扱う必要があり、規模の差は天文学的です。

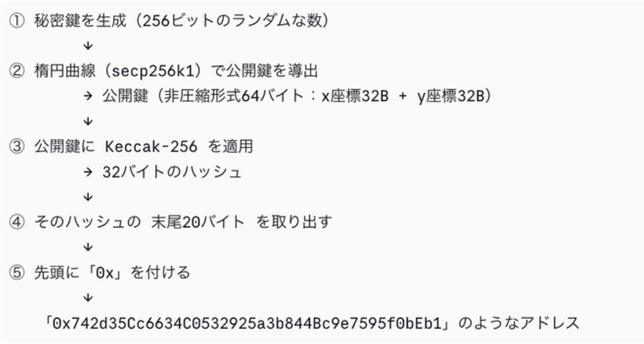
「1000量子ビット達成！」も実は**物理量子ビット**の話で、**実用的な計算に使える論理量子ビットは誤り訂正のため数千〜数万の物理量子ビットを束ねて1個**になります。
論理量子ビットの数では、まだ1個未満 が現状です。

実用化時期は専門家の間でも 「数年」から「永遠に来ない」 まで意見が分かれており、誰にもわかりません。

https://eprint.iacr.org/2025/1237.pdf

11 よくある誤解③「Bitcoin や Ethereum はアドレスがハッシュだから安全」

半分正しいですが、実態としてはほぼ誤解 です。
確かに Bitcoin や Ethereum はアドレスが公開鍵のハッシュであり、**Solana のように公開鍵がそのまま露出してはいません。**



しかし、**送金トランザクションを送った瞬間に公開鍵が露出**します。
署名検証のため、**ネットワークに公開鍵を提示する必要**があるためです。

つまり、**一度でも送金したアドレスは、Shorで秘密鍵を逆算される対象**になります。
特に Ethereum はアドレスを再利用する文化 のため、アクティブなアドレスのほとんどで「ハッシュ化による保護」は機能していません。

実際、約400万BTC(流通量の約25%)が既に公開鍵が露出した状態にあるとの研究結果もあります。

12 Solana は今すぐ移行する必要はない（2025年12月時点Helius執筆者観点）

Solana は近い将来、ポスト量子暗号への移行を 急いで行う必要はないと考えられます。

実現にはフォーマット変更・衝突回避ロジック・複数PQ方式の検証対応などが必要で、現状の PQ 署名は Solana にとってまだ大きすぎ、遅すぎる可能性があります。

ただし、無視してよい問題でもない
いざ必要になった際の道筋は見えており、構造的な検討は今のうちから進める価値があります。

待つことにも合理性がある
量子の脅威が現実化する頃には、HAWK 署名や STARK ベースの集約証明など、**より効率的な PQ 技術が登場している可能性** もあります。

つまり Solana にとっての現実的な立ち位置は「今すぐ慌てる話ではないが、無視もできない」 という、バランスの取れた姿勢ではないでしょうか。