Practical Privacy for DEFI

with an Open Source Software Perspective

Hart Montgomery
Linux Foundation

Talk Outline

Previously, Dan taught you the theory and tools for privacy on blockchain.

Goal: today, learn more about the problems facing practical deployments of DEFI and how to solve them.

Agenda:

- Introduction: why is the LF talking about DEFI?
- What's limiting DEFI adoption?
- Limitations of existing privacy technologies
- What extra considerations do we need to make privacy-preserving DEFI protocols practical?
- What's being used today to advance DEFI?
- Open source examples

Linux Foundation: Not Just the Kernel!

Across our foundations: hundreds of projects, lines of code in the billions

11.7M
Lines of Code Added Weekly

9.4M
Lines of Code Removed Weekly

208,660
Contributing Developers

12,435
Contributing Companies

16,618
Repositories

9.1M
Commits

960,610
Pull Requests

1.3M
Builds Monitored

859,150
Logged Issues

5.2B
Container Downloads

1.6M
Group Chat Messages

3.6M
Group Email Messages

4,532
Scanned Repositories

282,342
Open Vulnerabilities

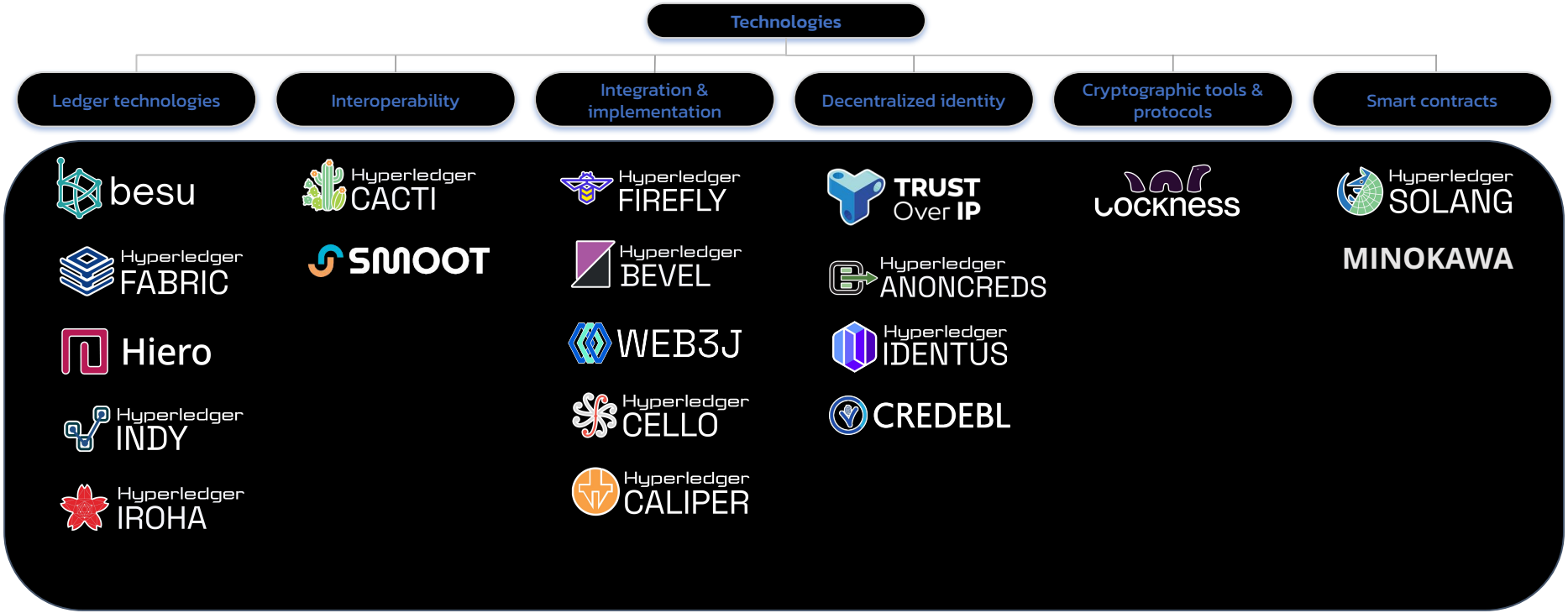
62,379
Recommended Fixes

36,163
Vulnerabilities Fixed

31,496
CLA Contributors

26,998
Community Meetings

Some Decentralized Technologies at the LF



View projects ➤

Besu: All Flavors of Ethereum!

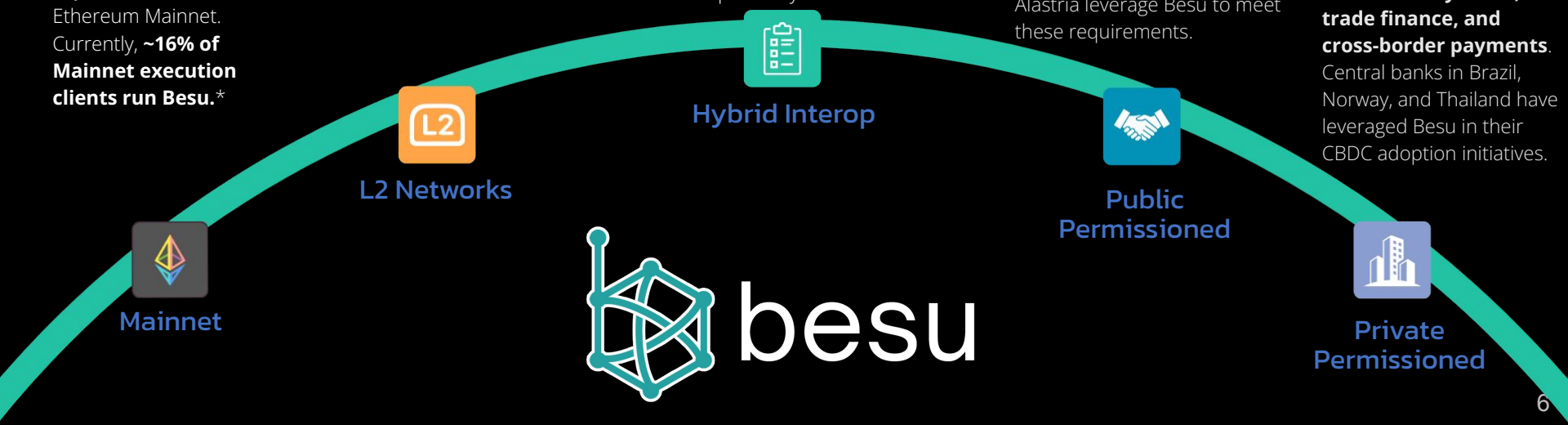
Besu serves as an execution client on **public proof-of-stake Ethereum networks**, including **Ethereum Mainnet, Goerli, and Sepolia**. It can be run as an execution client with any consensus client on Ethereum Mainnet. Currently, **~16% of Mainnet execution clients run Besu.***

Besu is used in Layer 2 (L2) solutions, which are secondary frameworks or **protocols designed to enhance scalability and transaction speed** on Ethereum. **L2s such as LINEA by Consensys** leverage Besu to optimize performance and efficiency.

Blockchains connected to Ethereum Mainnet via a two-way peg enable **secure asset transfers** while operating with **different consensus mechanisms**. The **Asian Development Bank Cross-Border Securities Transaction System** leverages this approach for enhanced interoperability.

Combines public accessibility with permissioned governance, Besu is well-suited for applications requiring **transparency, security, and regulatory compliance**. Public infrastructures such as LACCHAIN, EBSI (Europeum), Rede Blockchain Brasil, and Alastria leverage Besu to meet these requirements.

Besu offers comprehensive permissioning schemes tailored for consortium environments, ensuring controlled access to participants and transactions. It is **widely used in private blockchains for settlement systems, trade finance, and cross-border payments**. Central banks in Brazil, Norway, and Thailand have leveraged Besu in their CBDC adoption initiatives.

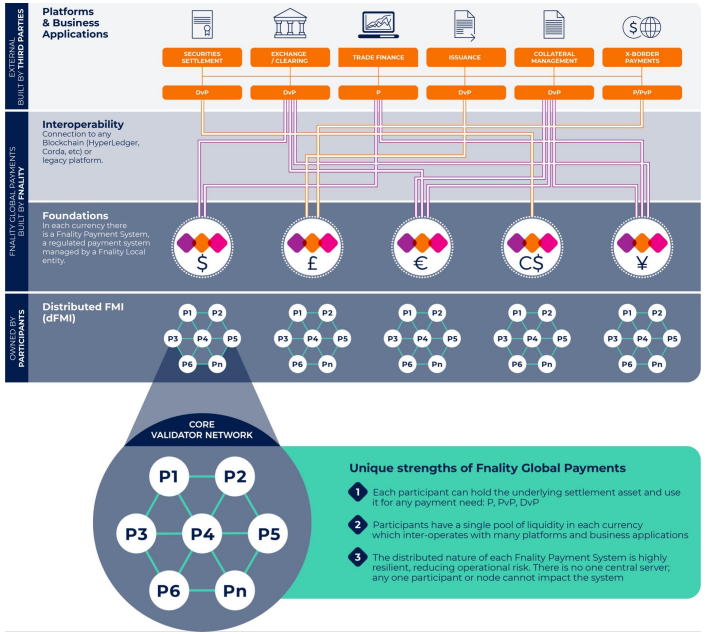


Collateralized Assets

FNALITY

besu

- Fnality is distributed Financial Market Infrastructure built on Hyperledger Besu that interoperates and connects to many existing systems and rails
- Collateral assets are tokenized and pooled among nodes in the network for use across currencies, payment types, and even with other blockchain networks
- The network reduces systemic and credit risk by utilizing the dFMI infrastructure and enabling faster settlement among parties
- Fnality provides a digital representation of money held in a central bank account to ensure banks could rely upon token value as they would fiat currency in transactions



Fnality now boasts 17 major institutions as shareholders: Banco Santander, BNY Mellon, Barclays, CIBC, Commerzbank, Credit Suisse, Euroclear, ING, KBC Group, Lloyds Banking Group, Mizuho Financial Group, MUFG Bank, Nasdaq, Nivaura, Sumitomo Mitsui Banking Corporation, State Street Corporation, and UBS

Tokenized Collateral Management

besu

DTCC launched a next-generation digital platform to improve global collateral mobility and efficiency, marking the first institutional use of AppChain infrastructure for decentralized finance

DTCC

Key Highlights

- Built on Besu, offering enterprise-grade privacy, security, and control
- Leverages DTCC AppChain and ComposerX for open, modular deployment of digital applications
- Demonstrates real-time tokenized collateral transfers across traditional and digital assets

Business Value

- Capital Efficiency: Increases liquidity and reduces collateral drag
- Operational Agility: Automates complex collateral workflows via smart contracts
  Market Connectivity: Bridges traditional and digital asset infrastructures

Why Besu?

- Open-source, Ethereum-compatible enterprise-grade blockchain hosted at neutral foundation under the Linux Foundation
- Only enterprise EVM with complete parity with Ethereum mainnet
- Designed for permissioned networks with privacy controls and interoperability
- Supported by the LF Decentralized Trust community for long-term resilience and innovation

> *"Collateral mobility is the killer app for institutional blockchain."*

— Dan Doney, CTO
DTCC Digital Assets

Read Announcement

Citi Banking Services

MEMBER COMPANY

TECHNOLOGY USED

SUMMARY

Early on, Citi saw blockchain's potential to address long-standing challenges in financial markets. So it isn't surprising that the organization began vetting distributed ledger technology (DLT) over a decade ago.

The announcement of Citi Token Services for Cash moving to a live commercial solution in October 2024 marked a turning point in finance.

RESULTS

- The platform has been connected to multiple Citi systems in the first year of its rollout
- Awarded the 2024 Model bank Award for Digital Asset Innovation by Celent, a global research and advisory firm for the financial services industry

Read case study ➢

CBDC Projects and Experiments

ebook

Norges Bank

Project Rosalind: England

Bundesbank

Project Venus: France & Luxembourg

Banque de France

Iberpay Smart Money

ECB Digital Euro

Project Aber: Sa
Arabia, U.A.E.

Eastern Caribbean
Central Bank DCash

Bank of Nigeria
e-Naira

Banco Central do Brasil
Drex

besu

Hyperledger CACTI

Hyperledger FABRIC

Hyperledger FIREFLY

Hyperledger IROHA

LF DECENTRALIZED TRUST

BANCO CENTRAL DO BRASIL

Bank of England

BANQUE DE FRANCE
EUROSYSTÈME

Central Bank of Nigeria

한국은행
BANK OF KOREA

HONG KONG MONETARY AUTHORITY
香港金融管理局

DEUTSCHE BUNDESBANK
EUROSYSTEM

Norges Bank

MAS
Monetary Authority of Singapore

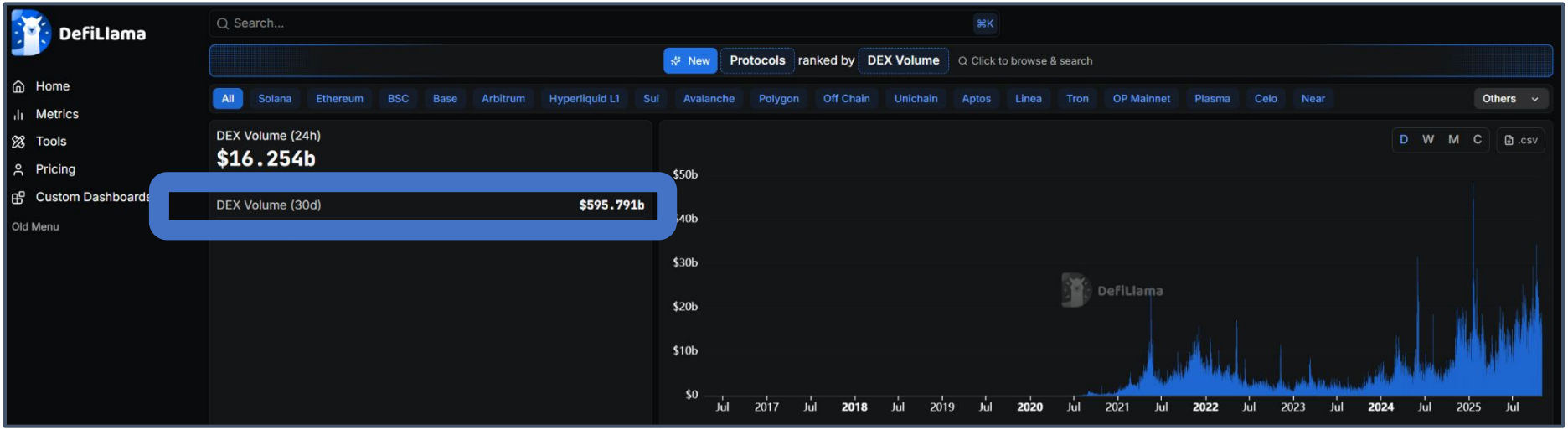
Federal Reserve Bank of Boston®

Global Collaborations

Swift Sandbox

Project Mariana: France, Switzerland, Singapore, & BIS

DEFI Adoption



DEFI Adoption

DTCC

Securing Today. Shaping Tomorrow.℠

Recall: DTCC processes many global securities transactions, providing clearing, settlement, and information services.

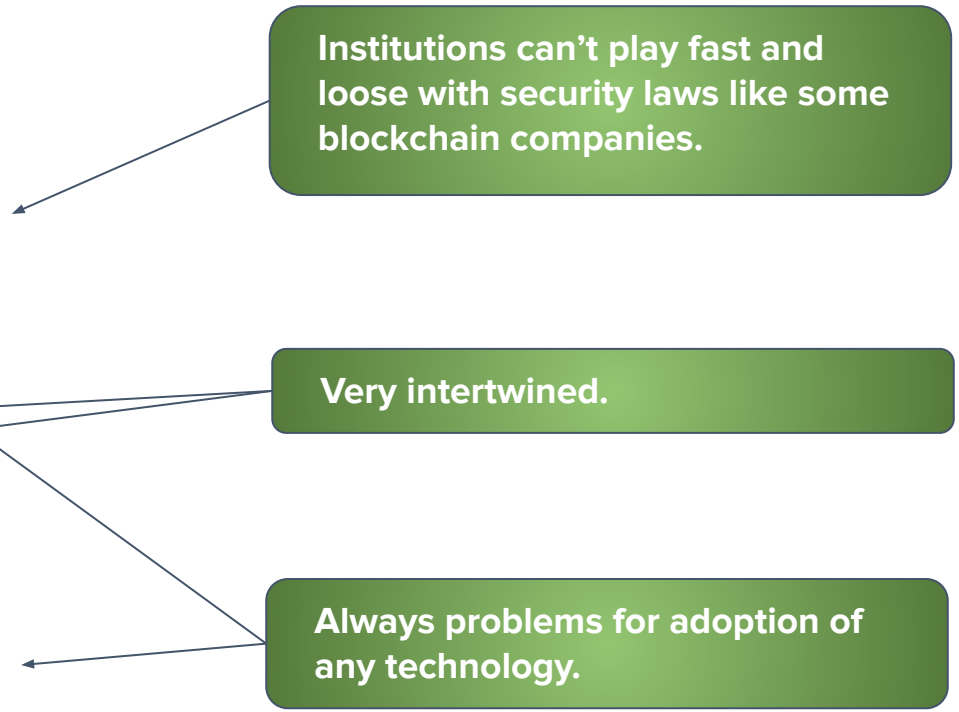
2024 Trades Cleared:

> **$4 Quadrillion**

Why Aren't Institutions Using DEFI?

A Personal List:

- Regulatory Concerns
  - More on this later, but changing for the better (at least in the US recently
- Legacy systems integration
- Scalability
- Privacy
- "Human Factors"
  - Institutions may not have or have access to experts that understand the technology



EUROPEAN COMMISSION

Brussels, 28.6.2023

COM(2023) 369 final

2023/0212(COD)

Proposal for a

REGULATION OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL

on the establishment of the digital euro

{SEC(2023) 257 final} - {SWD(2023) 233 final} - {SWD(2023) 234 final}

“The objective of this proposal is to ensure that central bank money with the status of legal tender remains available to the general public, while offering a state-of-the-art and cost-efficient payment means, **ensuring a high level of privacy in digital payments**, maintaining financial stability and promoting accessibility and financial inclusion.”

FORBES DIGITAL ASSETS

Brazil Abandons Blockchain For Its Drex CBDC Project

By Aaron Stanley, Contributor. ⓘ I write about decentralized infrastructure an... Follow Author

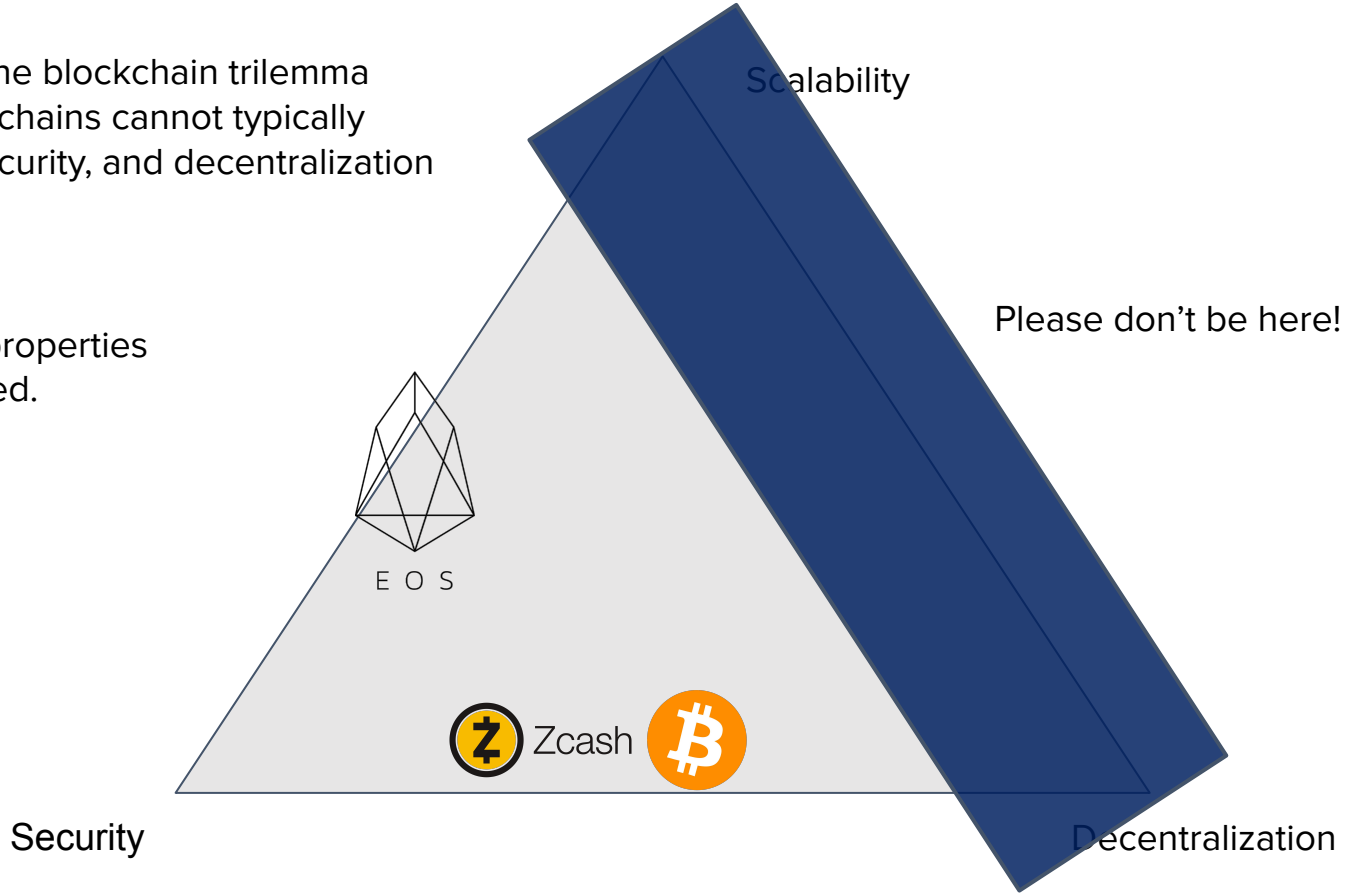
Published Aug 13, 2025, 04:59pm EDT

"In comments to Valor, Drex coordinator Fabio Araujo confirmed that the project's blockchain component would be **discontinued due to scaling and privacy challenges**."

Blockchain Trilemma

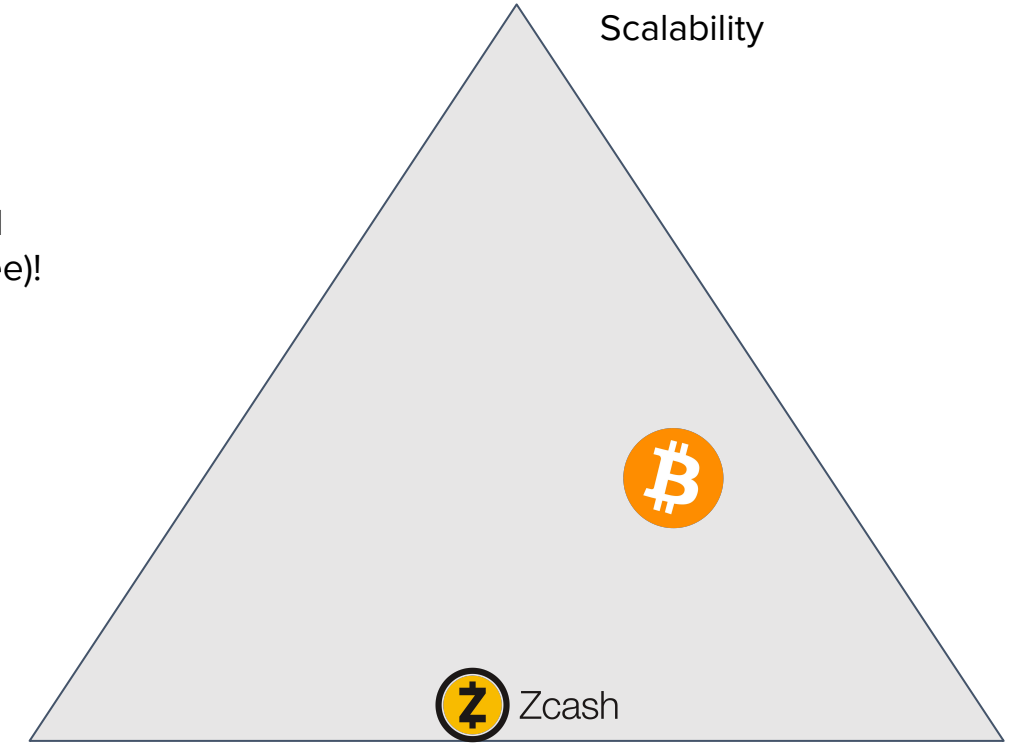
Coined by Vitalik Buterin, the blockchain trilemma refers to the fact that blockchains cannot typically achieve all of scalability, security, and decentralization at the same time.

Tradeoffs between these properties must be carefully considered.



Blockchain Trilemma’

Sometimes decentralization can **help** privacy, but usually only to a very small point (e.g. going from one node to three)!



Blockchain Scalability

From the Base Engineering Blog: **ZK ECDSA Verification**

Suite	Average gas cost	Difference
SnarkJS	347,665	Baseline
Rapidsnark	347,665	Baseline
Gnark	407,664	+17% (extra security)
Noir	2,396,575	+590% (6-7x higher)

Blockchain Scalability

From the Base Engineering Blog: **ZK ECDSA Verification**

Proof Generation Times

Noir dominated in speed, consistently achieving 5-50x faster proof generation than Groth16-based systems:

- c7i.2xlarge (8 vCPU, 16GB RAM): Noir ~2.1s, others 40-50s
- c7i.4xlarge (16 vCPU, 32GB RAM): Noir ~1.1s, others 20-30s
- c7i.8xlarge (32 vCPU, 64GB RAM): Noir ~0.6s, others 15-25s

Among Groth16 systems, Gnark performed best, followed by Rapidsnark and SnarkJS.

Blockchain Scalability

ZK Scalability is **still a problem for many applications** and is an **active area of research!**

If you want to learn more, try CS 171 (Intro to Cryptography) or especially CS 276 or CS 294 (Graduate and Advanced Cryptography, respectively).

In practical deployments today, we typically must b**ring in techniques outside of ZK to scale private blockchains (examples later!)**.

Recall from Dan: Types of Privacy

Pseudonymity: (weak privacy):

- One consistent pseudonym (e.g. reddit)
  - Pros: Reputation
  - Cons: Linkable posts: one post linked to you ⟹ all posts linked to you

Full anonymity:

- User's transactions are unlinkable
- The system cannot tell if two transactions are from the same person
- Maintaining reputation is possible but more complex

03/01/2018	CMSVEND*CV BAY AREA VEND SAN JOSE CA
03/01/2018	FALAFEL BITE SUNNYVALE CA
02/28/2018	CMSVEND*CV BAY AREA VEND SAN JOSE CA
02/28/2018	CMSVEND*CV BAY AREA VEND SAN JOSE CA
02/28/2018	60775 - SFO PARKING IT-G SAN FRANCISCOCA
02/27/2018	A1 CORPORATE CATERING SUNNYVALE CA
02/27/2018	CMSVEND*CV BAY AREA VEND SAN JOSE CA
02/27/2018	SHELL OIL 57444683205 REDWOOD CITY CA
02/27/2018	SAFEWAY #747 REDWOOD CITY CA
02/26/2018	STANFORD AOERC STANFORD CA
02/26/2018	MARTINS WEST GASTR REDWOOD CITY CA
02/26/2018	UBER V4PGT HELP.UBER.COMCA
02/26/2018	UBER TRIP MPYPR HELP.UBER.COMCA
02/26/2018	UBER TRIP V4PGT HELP.UBER.COMCA

Snack Machine
I work in the South Bay

Mediterranean Restaurant
I (probably) work in or near Sunnyvale

Fujitsu Cafeteria
I definitely work in Sunnyvale

Grocery/Gas
I (probably) live in Redwood City

Stanford Gym
I (probably) am a Stanford alum

Redwood City Gastropub
I (probably) live in Redwood City

Uber After Gastropub
I (probably) enjoy drinking


03/01/2018 CMSVEND*CV BAY AREA VEND SAN JOSE CA
03/01/2018 FALAFEL BITE SUNNYVALE CA
02/28/2018 CMSVEND*CV BAY AREA VEND SAN JOSE CA
02/28/2018 CMSVEND*CV BAY AREA VEND SAN JOSE CA
02/28/2018 60775 - SFO PARKING
02/27/2018 A1 CORPORATE CATERING
02/27/2018 CMSVEND*CV BAY AREA VEND SAN JOSE CA
02/27/2018 SHELL OIL 57444683205 REDWOOD CITY CA
02/27/2018 SAFEWAY #747 REDWOOD CITY CA
02/26/2018 STANFORD ACERC STANFORD CA
02/26/2018 MARTINS WEST GASTRO REDWOOD CITY CA
02/26/2018 UBER V4PGT HELP.UBER.COMCA
02/26/2018 UBER TRIP MPYPR HELP.UBER.COMCA
02/26/2018 UBER TRIP V4PGT HELP.UBER.COMCA
Just from my transactions over a few days, you learned that, in 2018:
I worked in Sunnyvale
I lived in Redwood City
I'm a Stanford alum
I enjoy beer/cocktails
This might not totally deanonymize me, but...
If you had my transactions over the course of an entire month, you could probably pinpoint me individually
Being "anonymous" didn't buy me much!
Challenging even in the permissioned setting!
Snack Machine
I work in the South Bay
Mediterranean Restaurant
I (probably) work in or near Sunnyvale
Fujitsu Cafeteria
I definitely work in Sunnyvale
Grocery/Gas
I (probably) live in Redwood City
Stanford Gym
I (probably) am a Stanford alum
Redwood City Gastropub
I (probably) live in Redwood City
Uber After Gastropub
I (probably) enjoy drinking

Recall from Dan: Types of Privacy

Pseudonymity (weak privacy):

- One consistent pseudonym (e.g. reddit)
  - Pros: Reputation
  - Cons: linkable posts: one post linked to you ⟹ all posts linked to you

Full anonymity:

- User's transactions are unlinkable
- The system cannot tell if two transactions are from the same person
- Maintaining reputation is possible but more complex

PRESS RELEASE

Founder Of Tornado Cash Crypto Mixing Service Convicted Of Knowingly Transmitting Criminal Proceeds

Wednesday, August 6, 2025

Share

For Immediate Release

U.S. Attorney's Office, Southern District of New York

Tornado Cash Founder Roman Storm Knowingly Operated a Money Transmitting Business That Transmitted More Than $1 Billion in Criminal Proceeds

United States Attorney for the Southern District of New York, Jay Clayton, announced today the conviction of ROMAN STORM, a co-founder of Tornado Cash, a cryptocurrency mixer that facilitated more than $1 billion in illegal transactions, for willfully conspiring to operate a money transmitting business that moved more than $1 billion in dirty money. The defendant was found guilty following a four-week jury trial before U.S. District Judge Katherine Polk Failla.

Key takeaways

- Existing anti-money laundering (AML) approaches relying on trusted intermediaries have limited effectiveness with decentralised record-keeping in permissionless public blockchains.
- The public transaction history on blockchains can enable AML and other compliance efforts, such as FX regulations, by leveraging the provenance and history of any particular unit or balance of a cryptoasset, including stablecoins.
- An AML compliance score based on the likelihood that a particular cryptoasset unit or balance is linked with illicit activity may be referenced at points of contact with the banking system ("off-ramps"), preventing inflows of the proceeds of illicit activity and supporting a culture of "duty of care" among crypto market participants.

"*An approach to anti-money laundering compliance for cryptoassets*," BIS Bulletin No. 111, 13 August 2025

Common Crypto Misconceptions on KYC/AML

“You can always just solve KYC/AML with a ZK proof.”

- Legally, for many kinds of transactions financial institutions are required to have the information *in the clear*.
- KYC/AML formulas are often secret. Even the inputs to the functions are secret sometimes (Hart: I asked the BIS and they couldn’t tell me!).

“KYC/AML can be reusable with proofs.”

- Again, many financial institutions must have information *in the clear* or they may face legal liability in some cases.

“International KYC/AML is just a simple extension of local policies.”

- Legally handling cross-border transactions is exceedingly complicated.

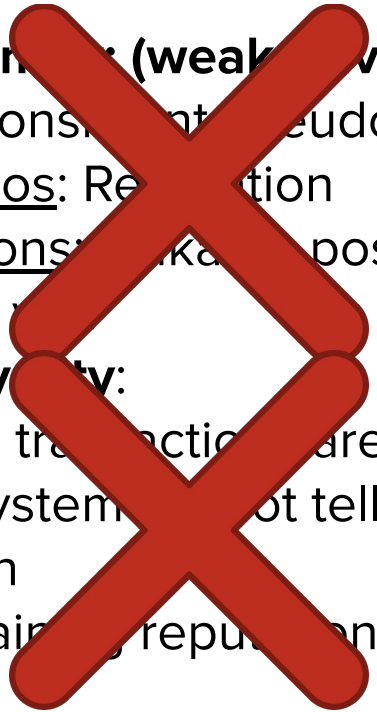
Recall from Dan: Types of Privacy

Pseudonymity (weak privacy):

- One consistent pseudonym (e.g. reddit)
  - Pros: Reputation
  - Cons: linkable posts: one post linked to you ⟹ all posts linked to you

Full anonymity:

- User's transactions are unlinkable
- The system cannot tell if two transactions are from the same person
- Maintaining reputation is possible but more complex



What Can We Do?

From Dan's lecture:

> Can we support positive applications of
> private payments, but prevent the negative ones?
> - Can we ensure legal compliance while preserving privacy?
> - Yes! With proper use of zero knowledge proofs

Well, not exactly.

Cryptography **doesn't solve all of our problems**, but it is **the most important tool** for us to solve these problems!

Architectural Ingredients of PETs

Privacy-Enhancing Token (PET) Requirements for Broad Use:

Onchain states as the **final source of truth** (balances/ownership, entitlements)

Confidentiality: must hide all of the above from public eyes

Anonymity: must hide identity information of asset ownership and transaction counterparties

History masking: must hide asset's lineage

Auditability: "regulator's view"

Anonymous **KYC**: transaction counterparties must be on the approved list

Post-Quantum security

Composability: atomic multi-step flows (DvP, PvP, etc.) support

Architectural Ingredients of PETs

The **BIS (Bank of International Settlements) reports** that are included in the reading material lay out in detail the requirements that are needed for blockchain to replace current financial systems.

Project Aurum: A Prototype for Two-tier Central Bank Digital Currency (CBDC)

Blueprint for the future monetary system: improving the old, enabling the new

Problem: Auditability

Many (regulated) financial markets require an auditor to have oversight of all transactions.

- FFIEC Bank Secrecy Act: banks that have a presence in the US must (securely) keep a record of any checks and deposits over 100 USD and any international payments for more than 10,000 USD for five years for anti-money laundering and anti-terrorism purposes, and these restrictions would almost certainly apply to CBDCs.
  - Would also apply to CBDCs and does apply to stablecoins in the US (due to GENIUS Act).
- Sarbanes-Oxley Act (H.R. 3763) imposes even more restrictions on data retention for accounting purposes.
- Security laws can be even tougher!

Problem: Auditability

Auditability often conflicts with privacy!

Simple Idea: encrypt the secret information in a ZK-based transaction to the auditor's public key. Then add in to the ZK logic a **proof that the secrets in the encryption are the same as the secrets in the ZK transaction**.

Caution in Practice: need to protect the auditor's secret key with **practical (e.g. thresholdizing)** and **legal mechanisms** to preserve privacy!

Problem: Auditability

Recall from Dan's lecture:

H_1, H_2, H_3: cryptographic hash functions.

(1) Shielded address: random $sk \leftarrow X$, $pk = H_1(sk)$

(2) Shielded coin: owned by address **pk**:
- (a) coin owner has (from payer): value **v** and $r \leftarrow R$
- (b) on blockchain: $coin = H_2((pk, v), r)$ (commitment to **pk, v**)

pk: addr. of owner, **v**: value of coin, **r**: random chosen by payer

New: **pk_Aud** public encryption key of auditor.

Problem: Auditability

Blockchain view:

Just Merkle root ... append only tree (coins are never removed)

Explicit list: one entry per spent coin

Problem: Auditability

Owner of **coin = H2((pk, v) , r)** wants to send coin value v to:

- Shielded **pk'**, **v'**
- Transparent **pk''**, **v''**

where **v = v' + v''**

Step 1: construct new **coin: coin' = H2((pk', v') , r')** by choosing random **r' ← R** (and send **(v', r')** to owner of **pk'**).

Step 2: compute nullifier for spent coin **nf = H3(sk, coin.Merkle_index)**

- nullifier **nf** is used to "cancel" coin (no double spends)
- key point: miners learn that some coin was spent, but not which one!

Step 2.5: Encrypt **(pk, v, r, pk', v', r')** under the auditor's public key by setting **ct_coin = Enc(pk_Aud, (pk, v, r, pk', v', r'), ; r_enc)**.

Problem: Auditability

Step 3: construct a zk-SNARK proof **π** for:

statement = **x = (current Merkle root, coin', nf, v', pk_Aud, ct_coin)**

witness = **w = (sk, (v, r), (pk', v', r'), MerkleProof(coin), r_enc)**

C(x, w) outputs **0** if:

Compute **coin:= H_2((pk = H_1(sk), v), r)** and check:

(1) **MerkleProof(coin)** is valid.

(2) **coin' = H_2((pk', v'), r')**

(3) **v = v' + v'', v' ≥ 0, v'' ≥ 0**

(4) **nf = H_3(sk, coin.Merkle_index)**

(5) **ct_coin = Enc(pk_Aud, (pk, v, r, pk', v', r'), ; r_enc)**

Very important that the values (pk, v, r, pk, v', r') are the same in the encryption and the standard ZCash circuit!

Problem: Auditability

Step 4:

Send **(coin', nf, transparent-TXO, proof π, ct_coin)** to blockchain.

Send **(v', r')** to owner of **pk'**.

Step 5:

Blockchain verifies:

(1) Proof **π** and **transparent-TXO**

(2) Verify that **nf** is not in the nullifier list (to prevent double spend)

(3) If so, blockchain:

- (a) Adds **coin'** to the Merkle tree.
- (b) Adds **nf** to the nullifier list.
- (c) Adds **transparent-TXO** to the UTXO set.
- (d) Adds **ct_coin** to audit list.

Problem: Auditability

Simple in Practice, Tricky to Make Efficient: composing an encryption scheme with a zk-SNARK can be **very tricky to do efficiently in practice.**

Other Ideas: in some cases, particularly when information can be shared with the auditor on a per-transaction or batch-transaction case, there are considerably more efficient ways to handle auditing.

Can also use MPC and other techniques to get rid of single point of failure and increase efficiency in certain cases.

Problem: Quantum Security

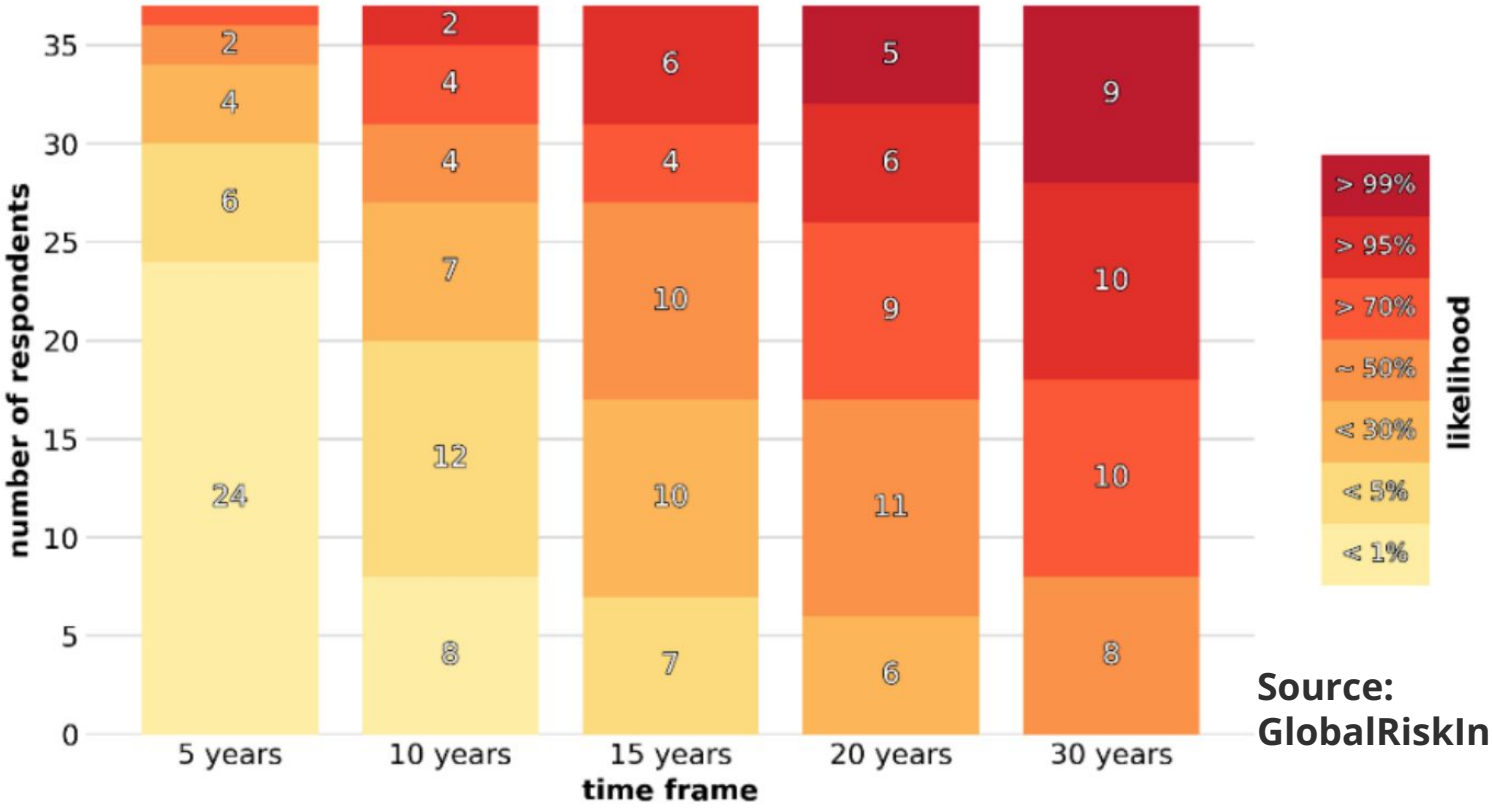
Most Public Blockchains: the only cryptoprimitives used are hashing and signatures, or perfectly hiding commitments. **No private information is stored (in a non-perfectly hiding way) on chain**.

Institutional DEFI chains may need to store secret information: in some cases, like our auditing application previously, **encrypted data or other secret information needs to be stored on chain**.

We may need quantum-secure cryptography for this!

2023 EXPERTS' ESTIMATES OF LIKELIHOOD OF A QUANTUM COMPUTER ABLE TO BREAK RSA-2048 IN 24 HOURS

Number of experts who indicated a certain likelihood in each indicated timeframe



Source: GlobalRiskInsitute.org

Problem: Quantum Security

Harvest Now, Decrypt Later: used to describe attacks where an attacker records (encrypted) data now and then decrypts it once quantum computers become powerful enough to break cryptography.

Data on blockchains is forever so this applies!

For blockchains that store data that needs to be secret for long periods of time, **quantum-secure privacy techniques need to be use**.

This is a demand from banks (and central banks), who often face legal consequences if data is exposed (sometimes data must be secure for up to 30 years!).

Problem: Composability (Atomicity)

In certain financial applications (e.g. PVP, or payment vs. payment, and sometimes DVP, or delivery vs. payment) we need strong atomicity guarantees:

Either all transactions in a set “go through”, or none of them do.

This is a fairly standard requirement for payment systems, but **baking it into privacy-preserving protocols and flows is tricky**.

This is a complicated topic but we’ll spend a little bit of time on it later in our open-source code explanation.

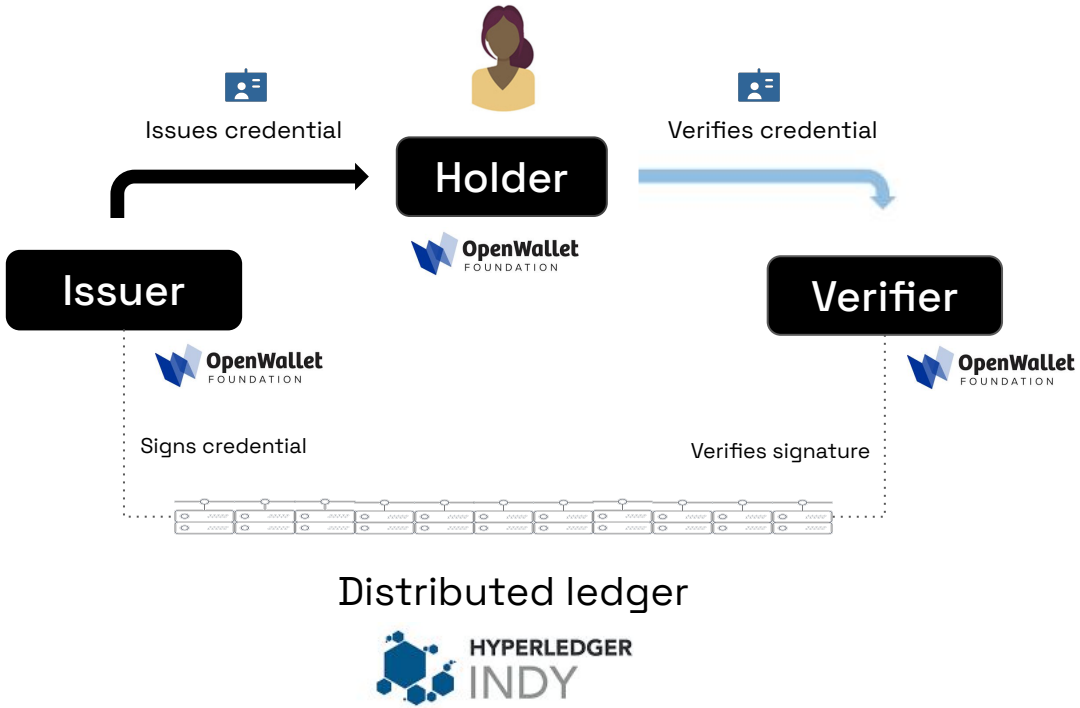
Problem: Digital Identity (KYC)

The most common way to solve KYC is through **anonymous credentials**.

Anonymous credentials could be an entire course, so we won't cover them formally here. **Baking them into zk-SNARKs efficiently is also tricky**.

Instead, we will briefly outline how privacy-preserving blockchain-based digital identity works.

Digital Identity Protocols (through LF Projects)



HYPERLEDGER INDY

The “root of trust” as a ledger

OpenWallet FOUNDATION

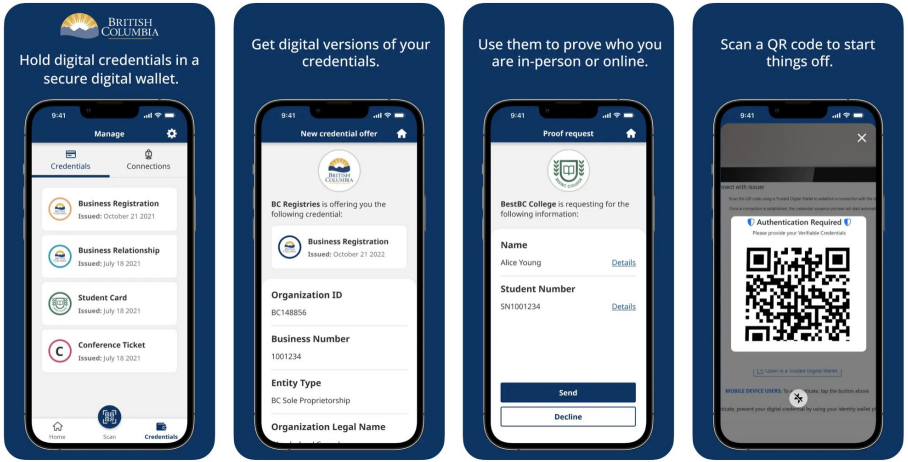
The “wallet” for digital identity

HYPERLEDGER ANONCREDS

The “protocol” for handling proofs

British Columbia Digital Wallet

- Securely receive, store and present digital credentials such as licenses, memberships, IDs, permits, and more
- Only use the parts of your credentials that are needed for any given situation
- Credentials are only used over secure, confidential connections, and you approve each use
- Wallet is built from open-source code by the Government of British Columbia.



HYPERLEDGER INDY

OpenWallet FOUNDATION

> Visit BC Govt Wallet

KYC for Financial Institutions

Benefits related to security, fraud prevention, loss liability shift, and Know Your Customer (KYC) checks for businesses operating in the digital space.

With the help of Hyperledger technologies Instnt Accept™ enables end user to own their identity:

- Distributed Ledger Technology / Blockchain
- Decentralized ID (DID)
- DIDComm protocol
- Verifiable Credential (VC)
- Level of Assurance (LOA)
- JSON-LD Schema

HYPERLEDGER INDY

OpenWallet FOUNDATION

instnt

What to Do?

The above may make the problem of practical privacy seem grim.

It isn't quite that bad though: modern technology gets us pretty close!

We'll spend the rest of the talk going over the state of the art in terms of technology that institutions actually use!

All of what I'll talk about is free, open source software hosted at the Linux Foundation! Use it for your real-world development projects, or even your class projects if you're allowed!

Paladin:
Programmable Privacy for EVM

an LF Decentralized Trust Project

LF DECENTRALIZED TRUST
LABS

PALADIN

EVM

** no modifications necessary*

Privacy preserving tokens

Confidential and anonymous fungible and non-fungible tokens

Private EVM smart contracts

Private EVM worlds running on the Besu EVM backed by a single ledger

Advanced cryptography - including ZKP

Each token can be backed by an notary or fully decentralized ZKP

Ecosystem programmability - with atomicity

DvP, PvP and more all programmed in EVM without token modification

Full enterprise privacy stack

Private data, Wallet functions, Key management, Transports, Registry...

Simple, extensible, cloud native runtime

Single process, modular architecture - with Kubernetes operator

Why EVM?

Why do we rely on the EVM rather than use newer, potentially faster technology?

Financial institutions want to avoid "vendor lock-in". Most other chains and technologies have a single vendor and/or developer, and companies want to avoid being too reliant on a single entity.
Decentralization in software vendors!

Advice: pay attention to what technology people use and adapt accordingly. You can't always get them to use what you want (e.g. ECDSA for the cryptographers).

PALADIN

Programmable Privacy

LF DECENTRALIZED TRUST LABS

Privacy Frameworks

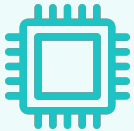
- **ZKP Tokens** — ZKP Logic Frameworks
- **Issuer Backed Tokens** — EVM or off-chain notarization
- **Private EVM** — Smart Contracts
- *... next new privacy model on EVM*

Wallet Functions

Token & Contract APIs

Private Data Store

Key Management



ZKP Proof Engines

Client Functions

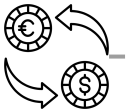
Transaction Orchestration

Indexing

Event streams

Encrypted Data Transfer

Existing (non-private) Tokens / Contracts



Atomic Swap Contracts

Privacy Preserving Smart Contracts

EVM

*no modifications necessary

Node Registry

Recall: Native EVM, Lack of Privacy

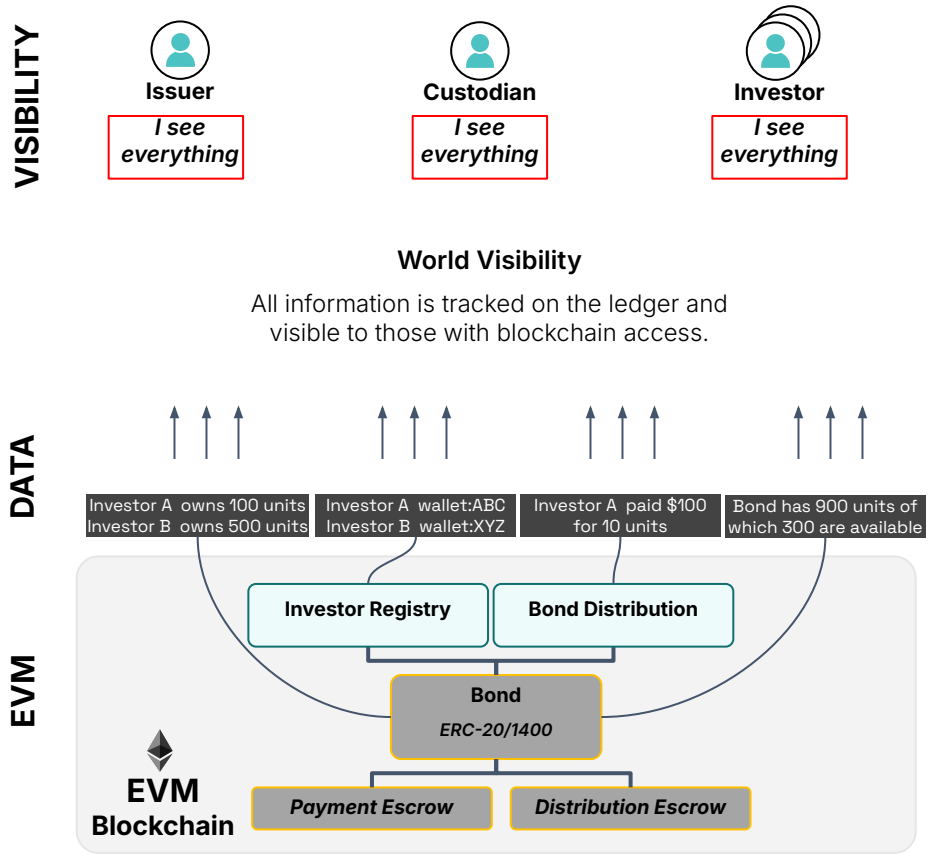
Bond Example

We'll examine a simplified Bond use case throughout this presentation.

The "as is" example shows how a tokenized bond stores all data on the Base EVM.

This means that any party that has access to a node can see all of the data.

This is a significant problem for many use cases.

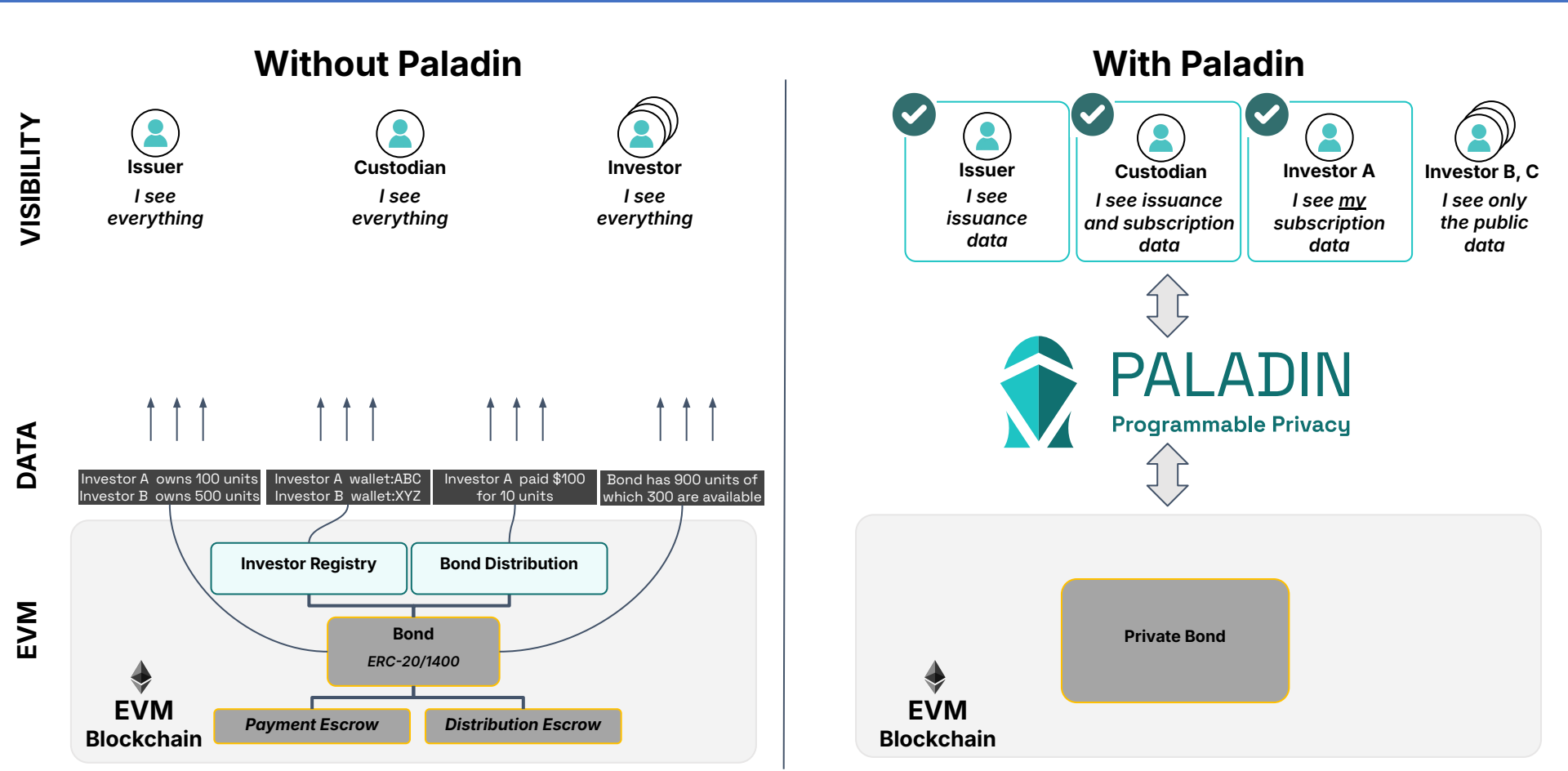


PROBLEM!

Many use cases must have some level of privacy.

In this case, for example, all Investors can see all of the details about Investor A's activity.

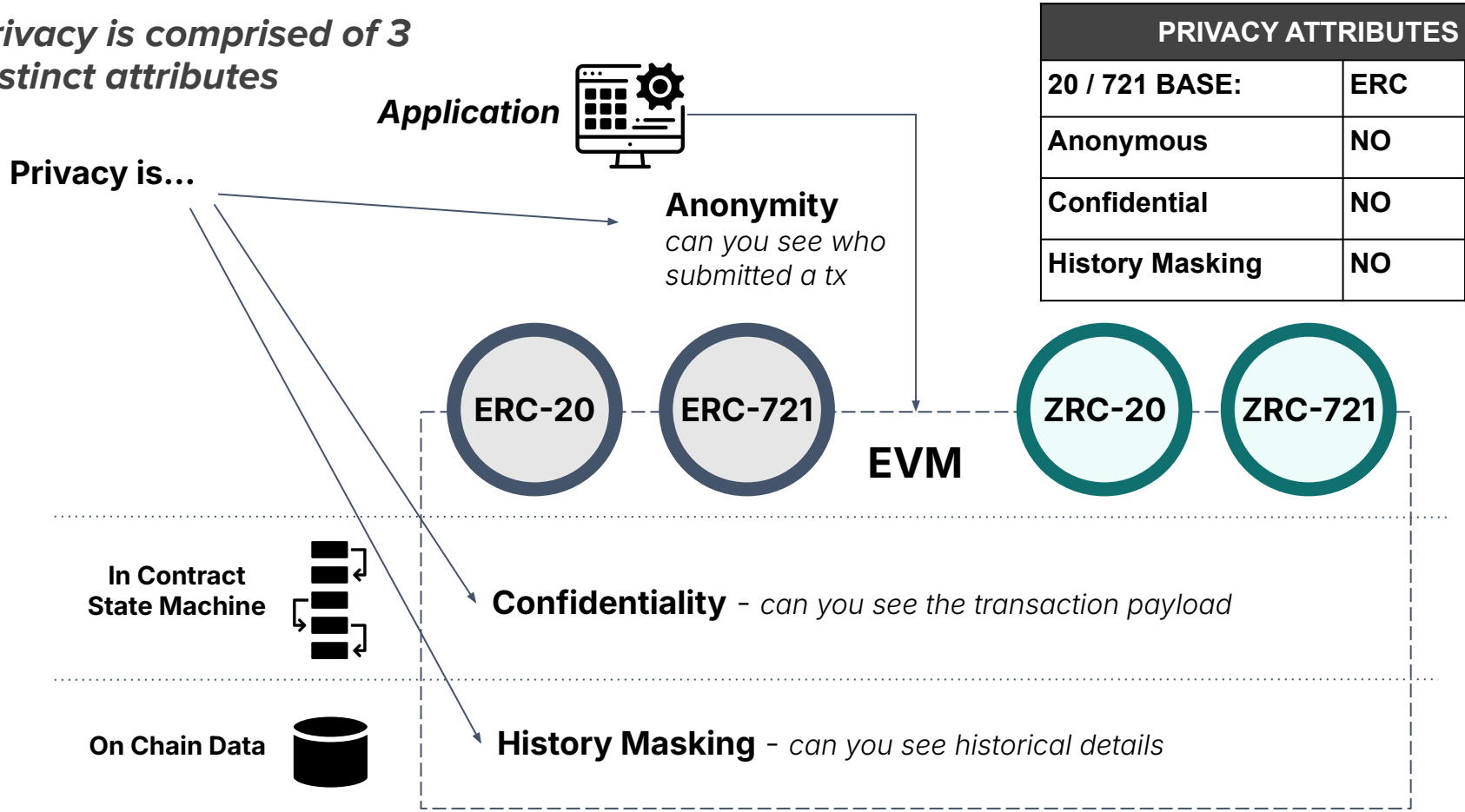
Parties Only See the Intended Information



End to End Privacy: What Is It Anyway?

Privacy is comprised of 3 distinct attributes

PRIVACY ATTRIBUTES		
20 / 721 BASE:	ERC	ZRC
Anonymous	NO	YES
Confidential	NO	YES
History Masking	NO	YES



Trust and Data Model

	Plain ERC	Issuer Backed ZRC	Zero Knowledge ZRC
TRUST MODEL	**TRUSTLESS** *I trust it because I can see everything*	**HYBRID*** *I trust it because I trust the Issuer*	**TRUSTLESS** *I trust it because I trust zk mathematical proofs*
DATA	**OPEN** Stores all data such as balances and transaction participants in the open on chain	**PRIVATE** Stores UTXO state transitions on chain. The Issuer knows all state. Users only know their state.	**PRIVATE OR AUDITED** Stores UTXO state transitions on chain. Token owners generate proofs. Optional pattern to allow 3rd party to *audit* all activities
EXAMPLE USE CASE	Stablecoins, DEFI	Bonds, CBDCs, KYC	Trading Securities, CBDCs, KYC

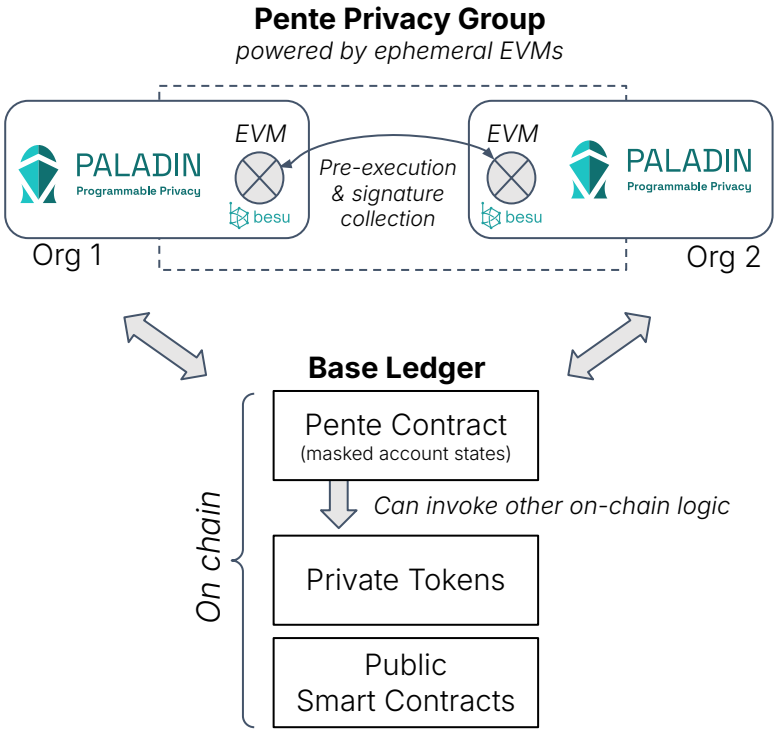
*one can describe NOTO as ***TRUSTLESS COUNTERPARTY + TRUSTED ISSUER***

EVM Private Smart Contracts

Privacy Groups provide an EVM native way to program private workflows

The Pente Domain

- A Paladin Client can instantiate a *Pente Domain* to participate in privacy group workflows
- Pente improves on the architecture of prior Privacy Groups
- Developer friendly and simple to use
- Scalable & efficient design horizontally manages many parallel privacy groups



Pente Privacy Groups

- Each Org transacting in a Privacy Group runs an ephemeral EVM
- A Privacy Group EVM only runs to process a transaction (like Lambda)
- Endorsement signatures are collected and transactions submitted to the base ledger
- The base ledger enforces the inputs, outputs and order of every transaction

How Paladin Works

Business Workflows

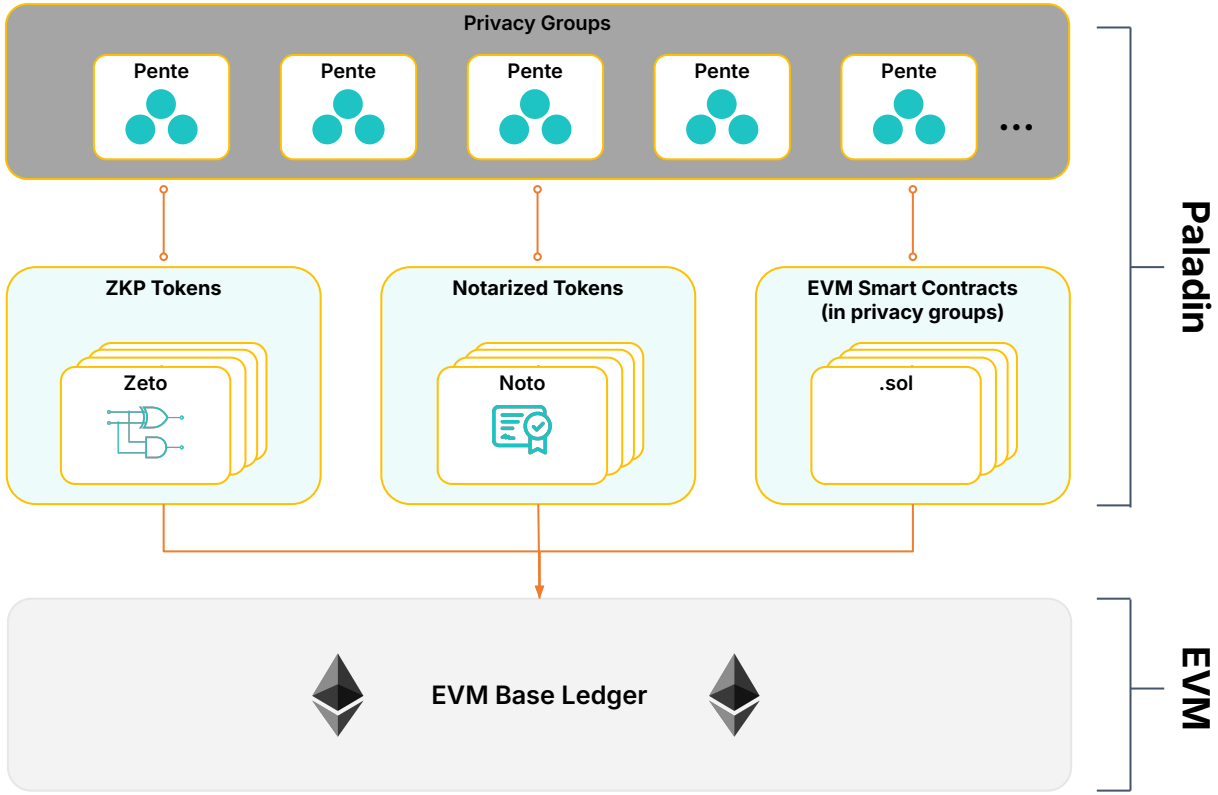
Utilize Pente privacy groups for approved parties to create and execute business logic.

Transaction Privacy

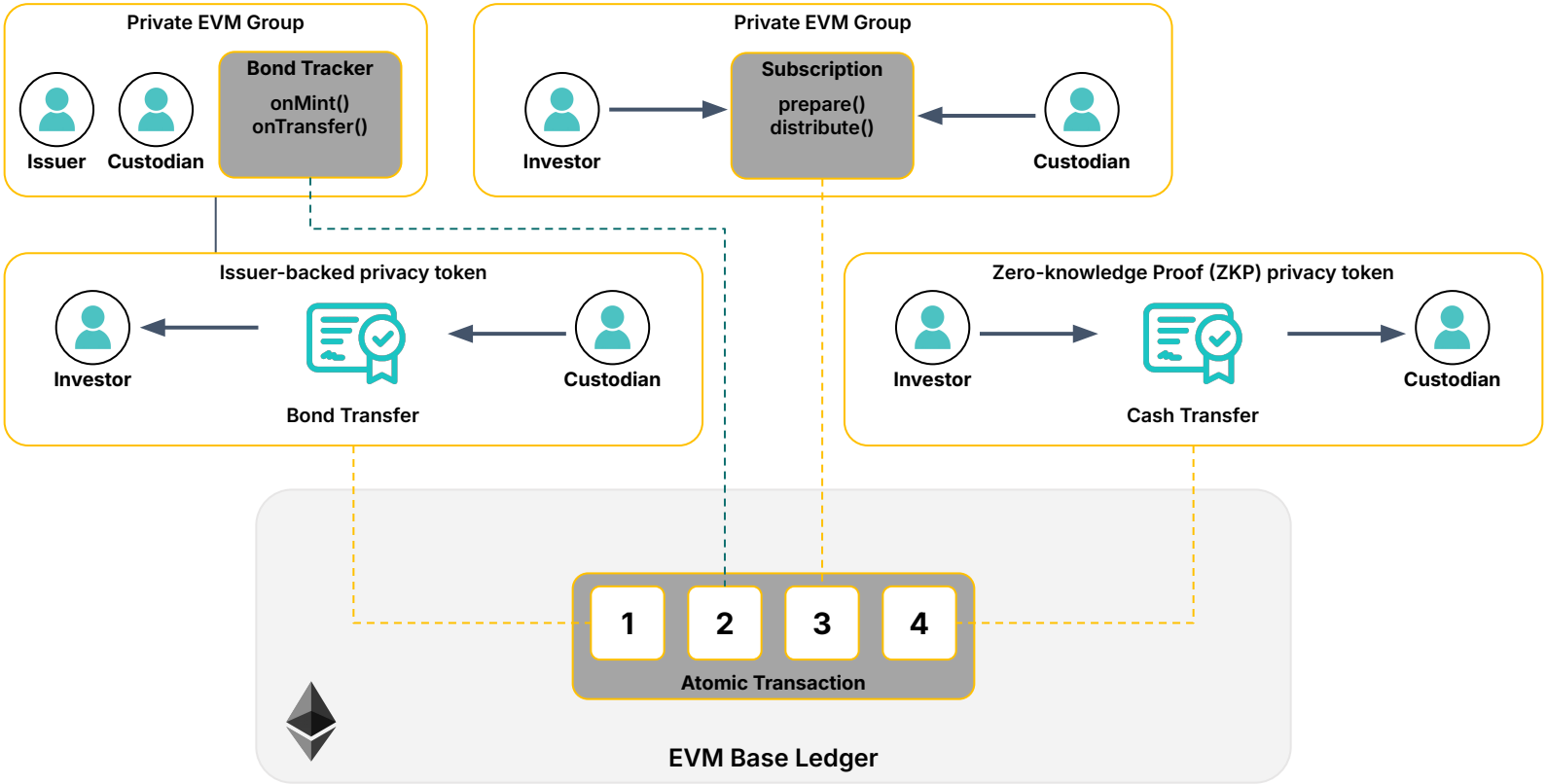
Utilize Zeto/Noto privacy preserving tokens or EVM Smart Contracts in privacy groups to execute confidential value transfer.

Global State

Record private transaction in shared global state while masking transaction details.

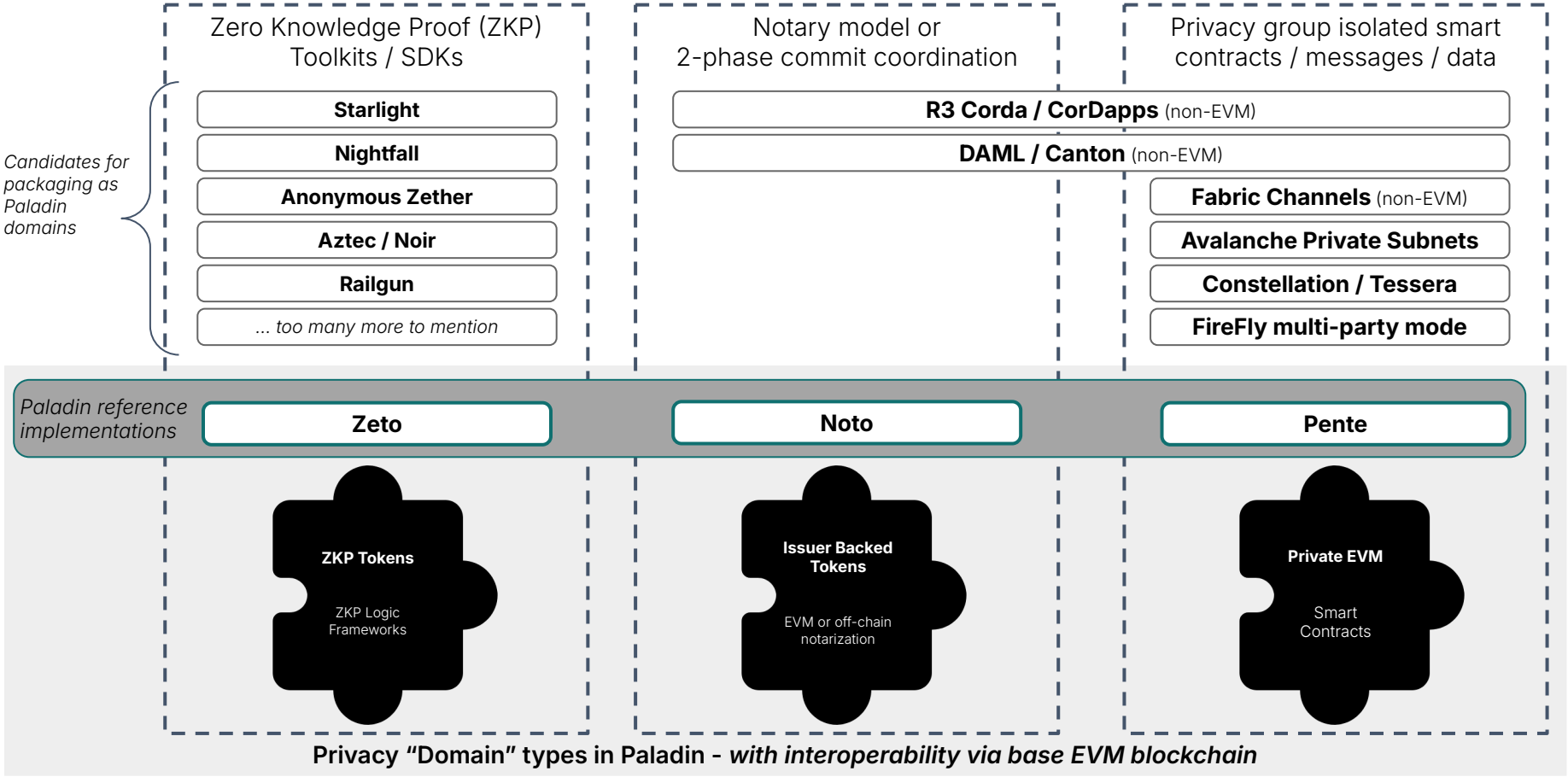


Atomic DvP with sub-transaction privacy



Paladin delivers programmable, composable Atomic DvP across the entire privacy stack

The Enterprise Privacy Landscape



Privacy "Domain" types in Paladin - *with interoperability via base EVM blockchain*

Recall: UTXOs

Unspent **T**ransaction **O**utputs:

Why for Ethereum?

Benefits:

- Highly parallelizable spending transactions (higher throughput compared to account based privacy tokens)

Challenges:

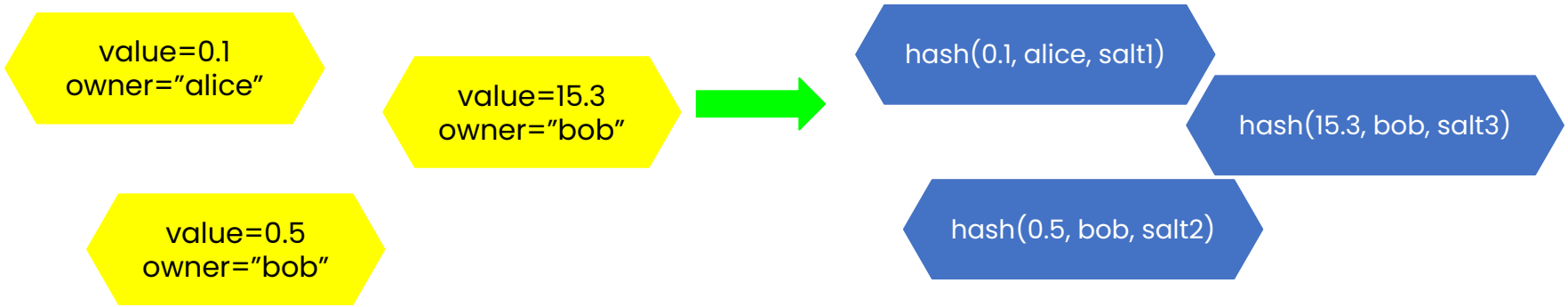
- Requires more complex clients (such as LFDT Paladin: https://github.com/LF-Decentralized-Trust-labs/paladin)

UTXOs as commitments

Use secure hashes to hide the values

`hash(value, owner public key, salt)`

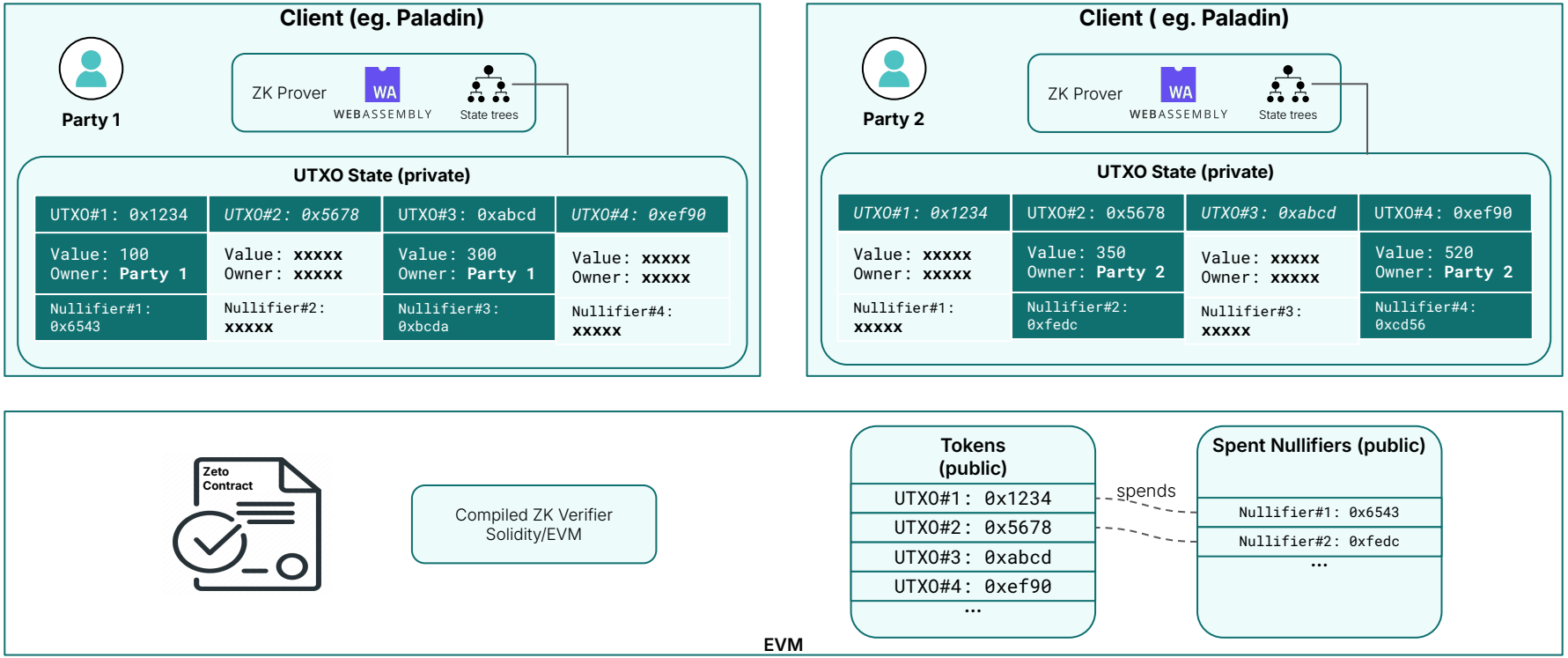
ZKP-friendly hashing algorithms: Poseidon



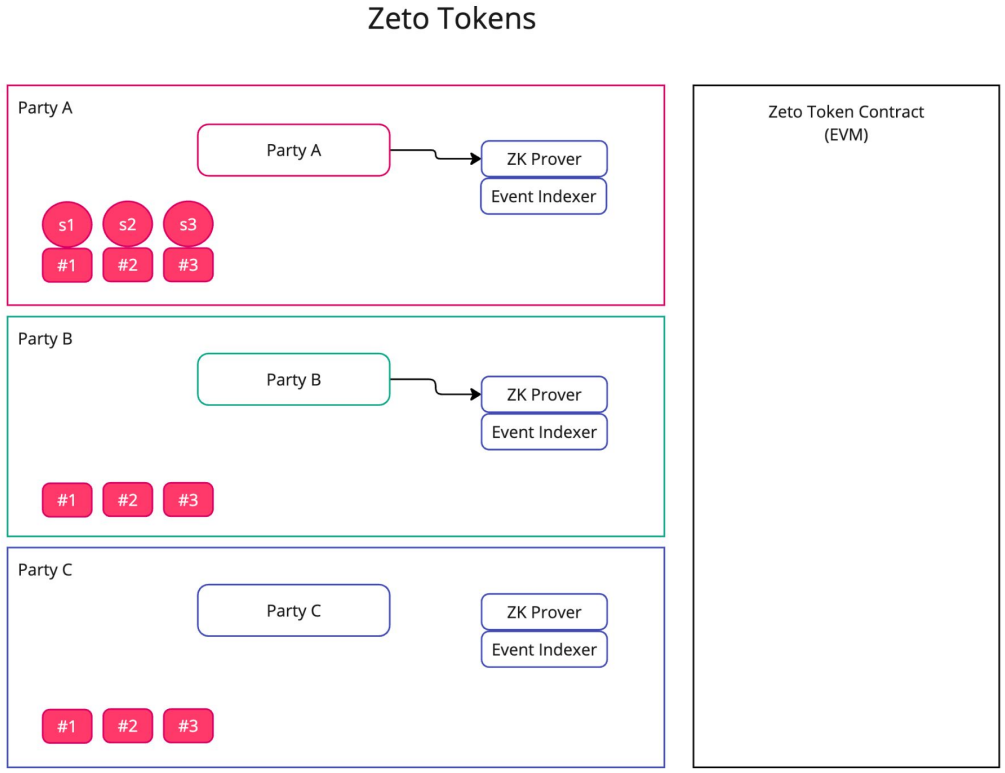
UTXOs with clear-text values, eg. Bitcoin
(alice and bob are pseudonymous addresses)

UTXOs with commitments

UTXO Commitments and private states

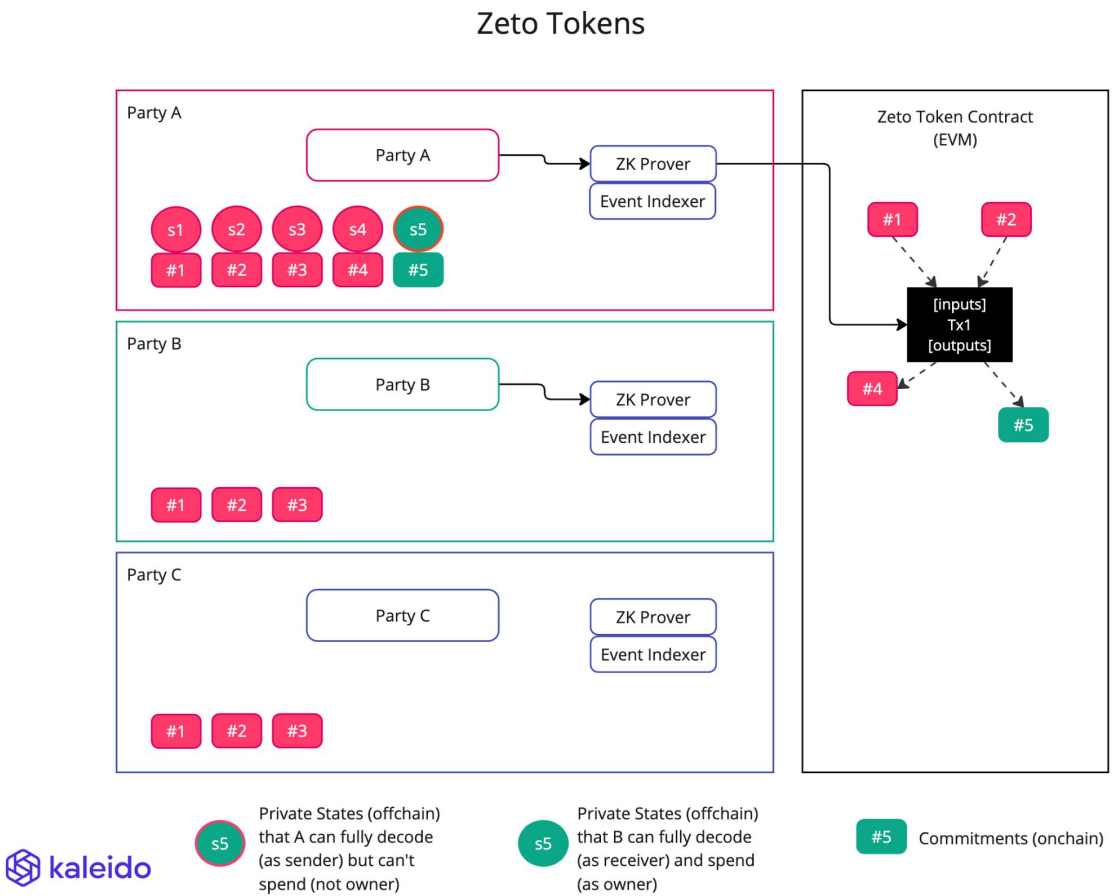


UTXO Commitments and private states

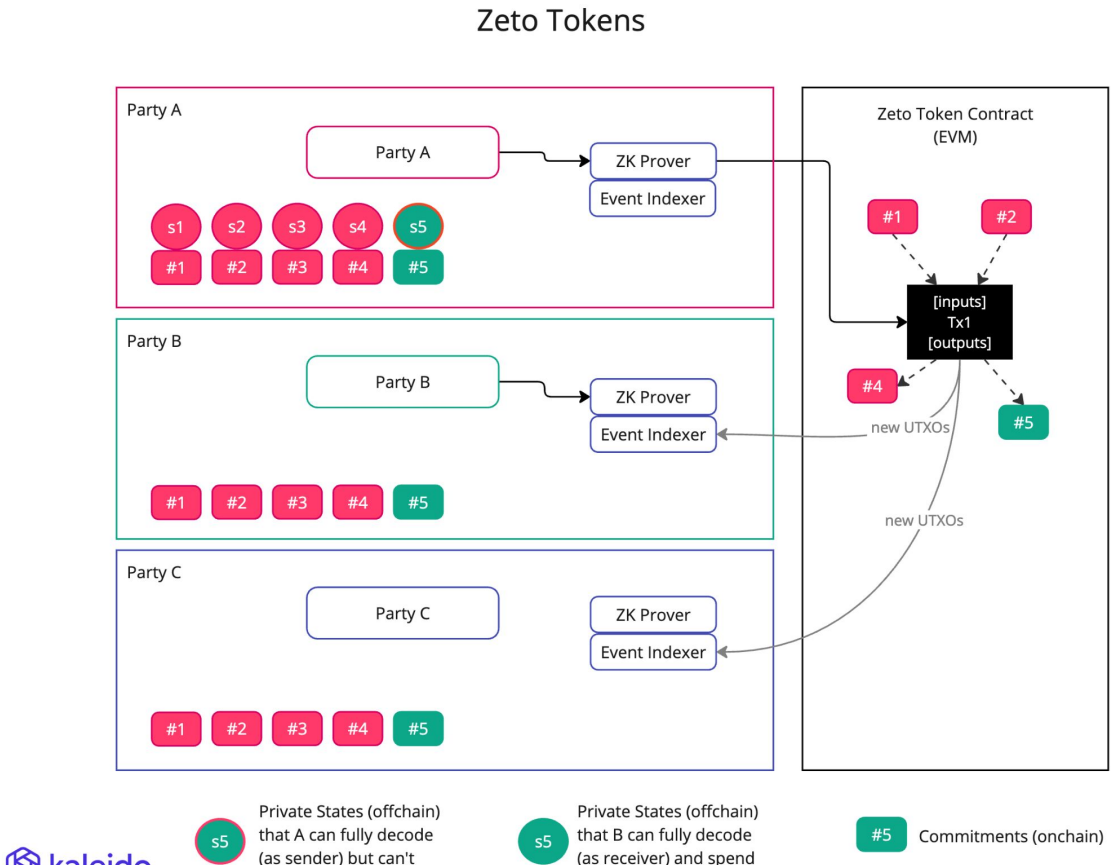


kaleido

UTXO Commitments and private states

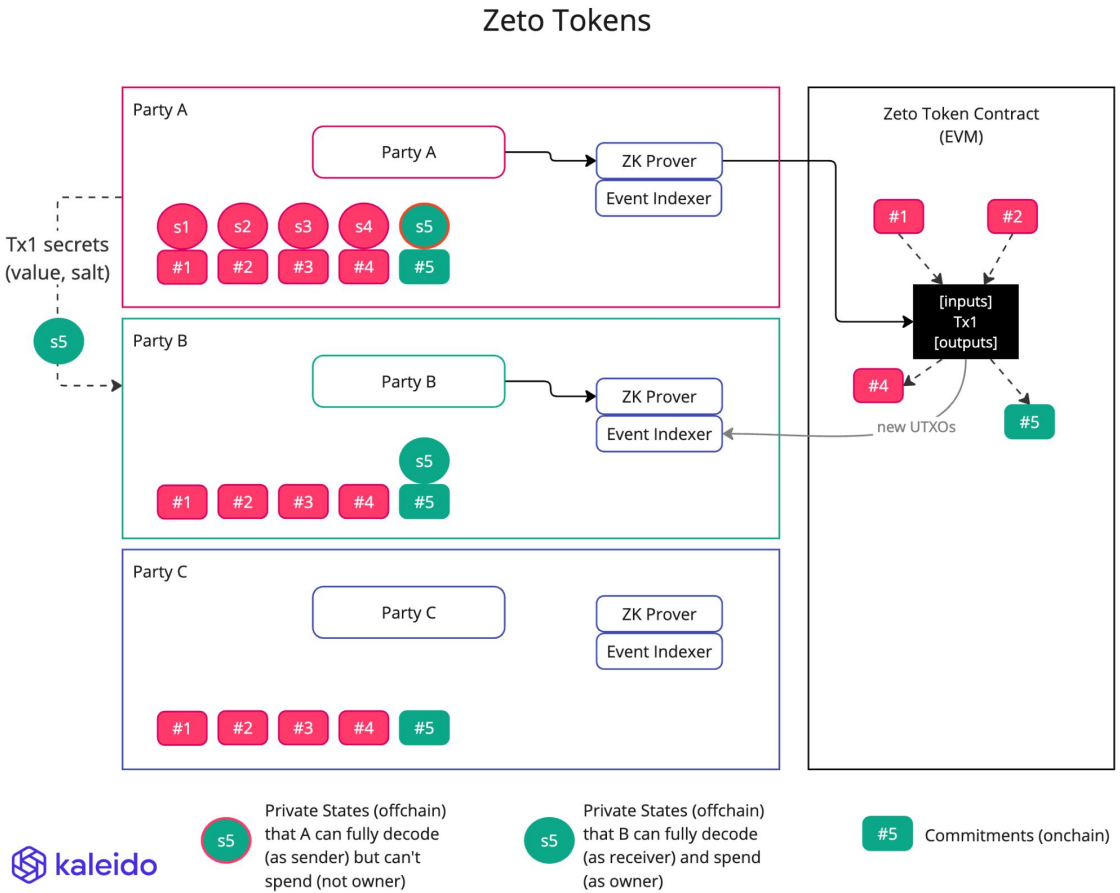


UTXO Commitments and private states

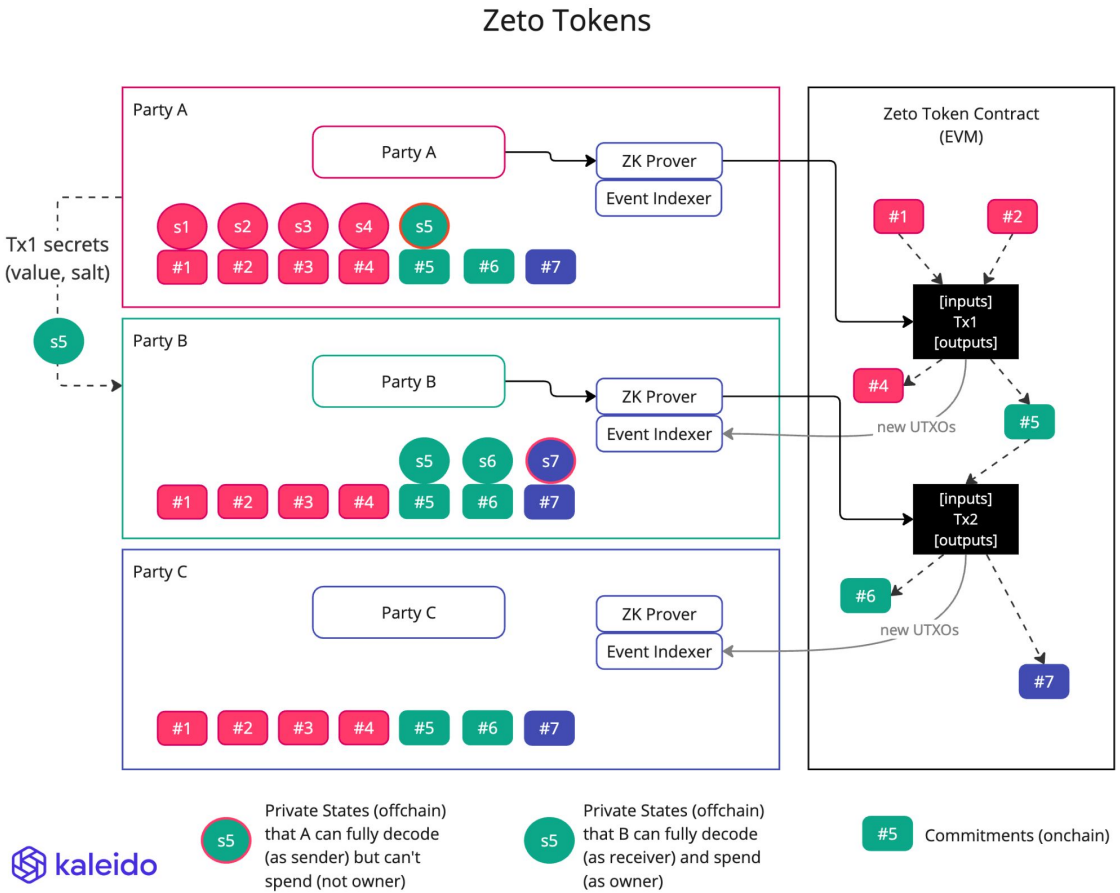


kaleido

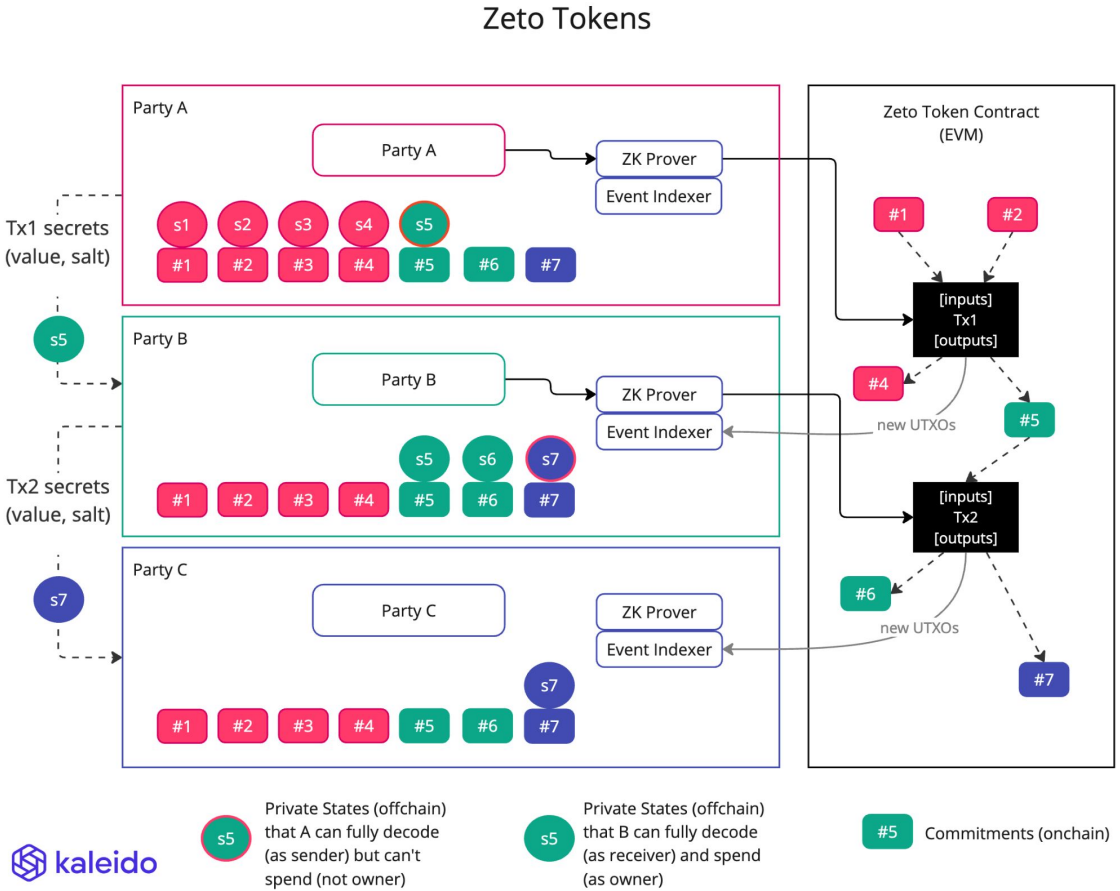
UTXO Commitments and private states



UTXO Commitments and private states



UTXO Commitments and private states



Anonymity

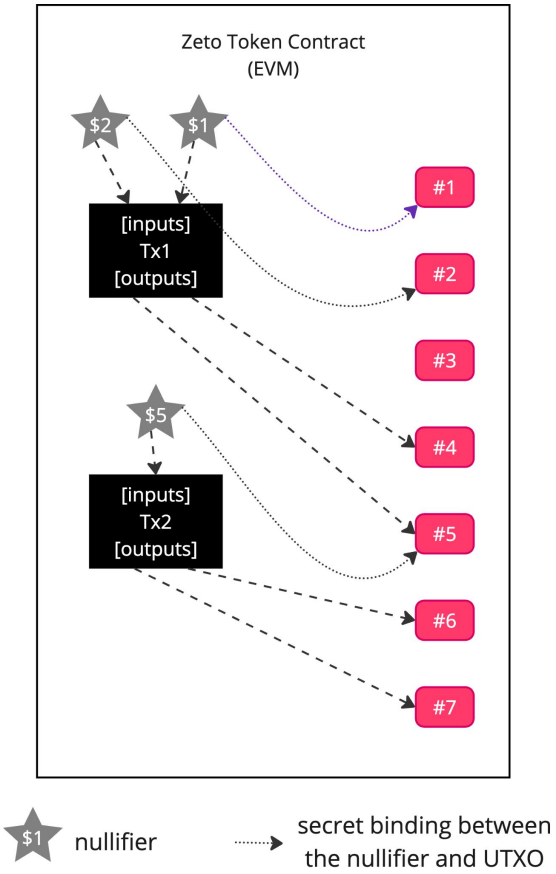
ZKP verification guarantees proper entitlement of the spender (sender)

No need to check for **`msg.sender`** any longer

Transactions can be signed by any address

One-time signing keys can be used

History Masking - Nullifiers



Target commitment:

hash(value, owner public key, salt)

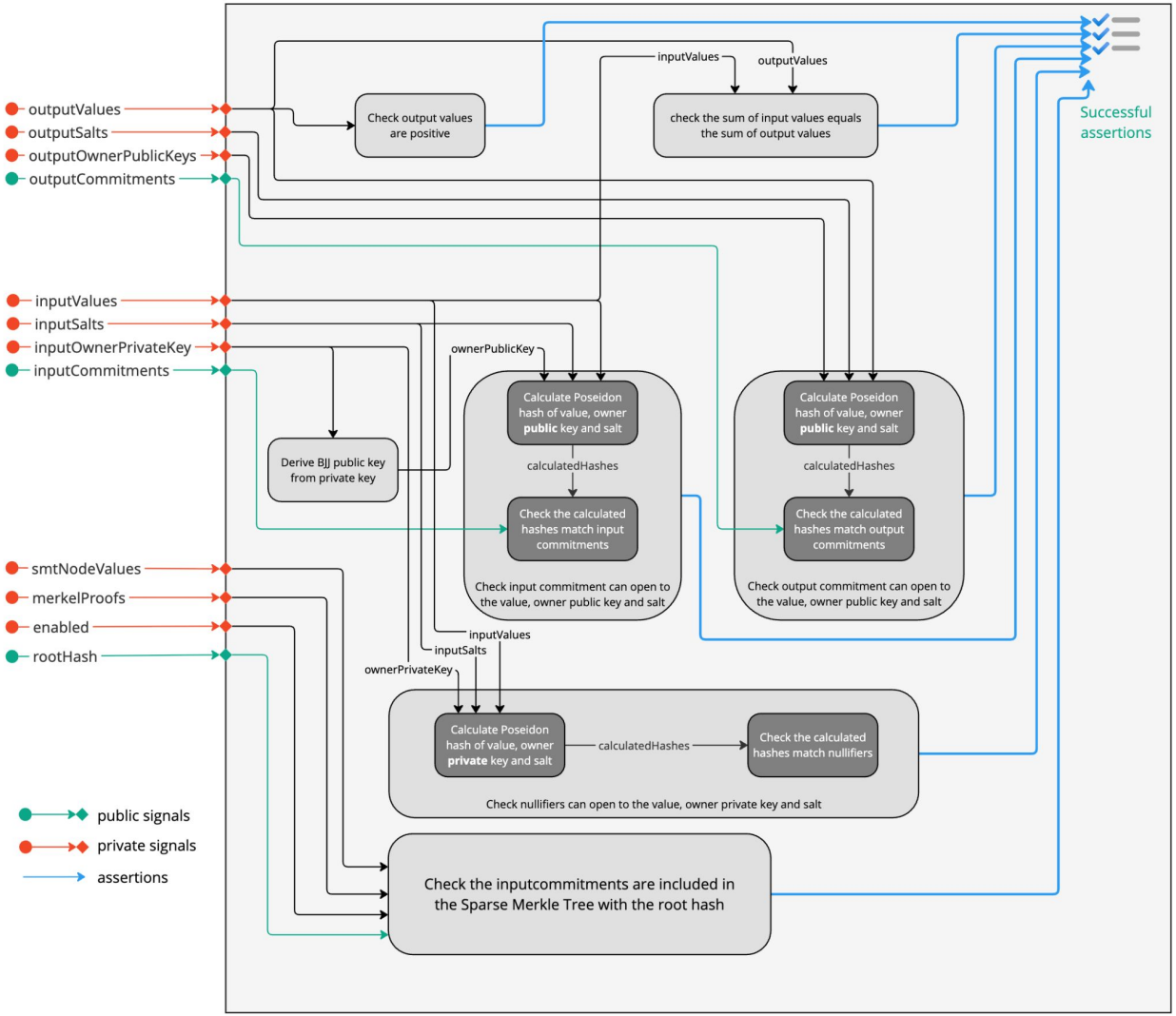
Nullifier:

hash(value, owner private key, salt)

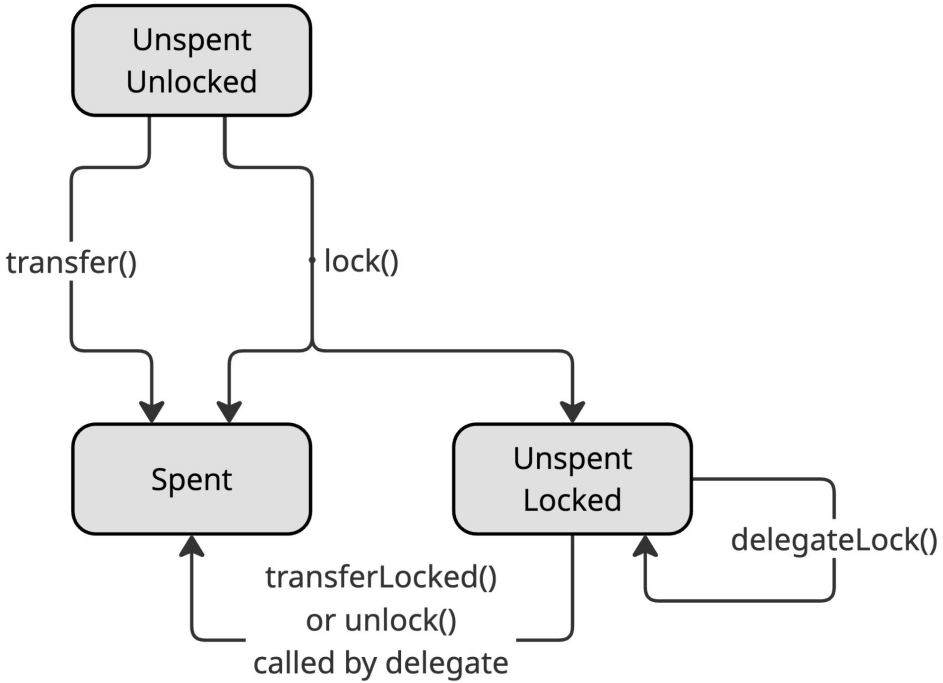
ZKP Circuit

Zeto_AnonNullifer

- Anonymity
- Confidentiality
- History masking

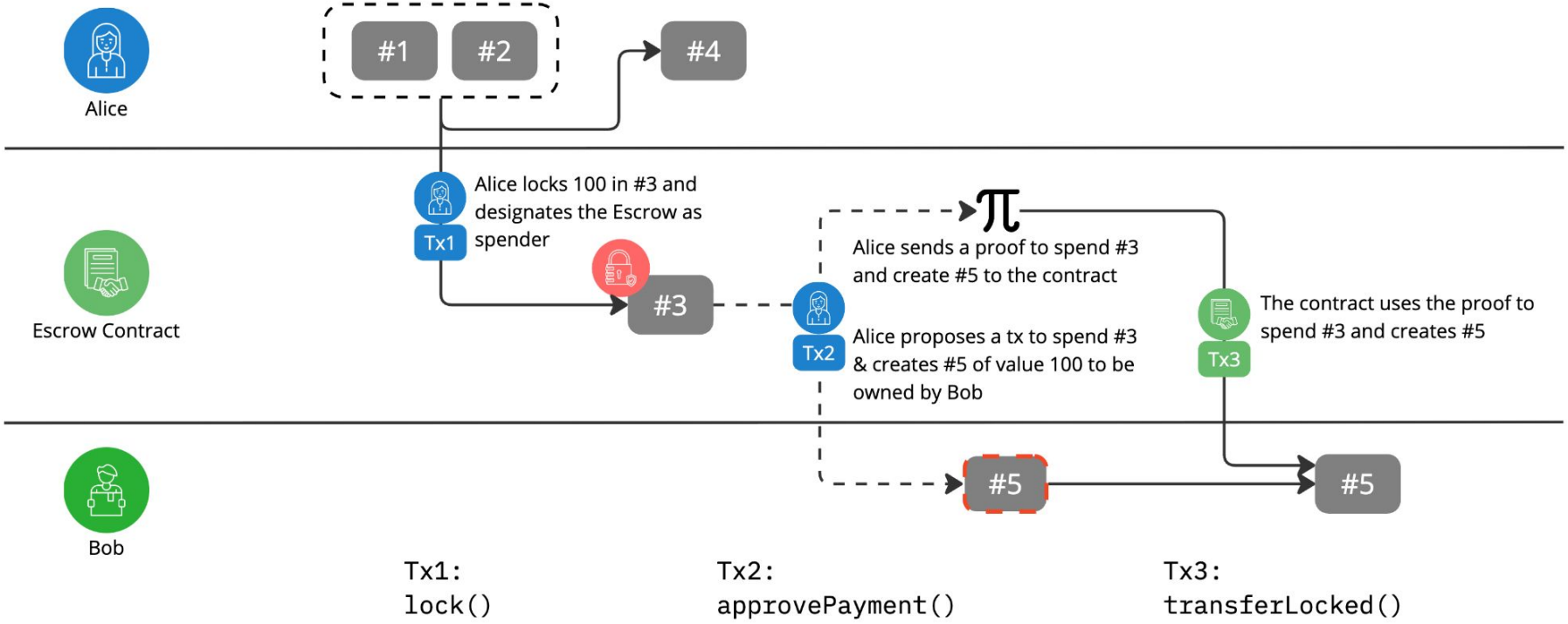


Locking Mechanism for DeFi flows

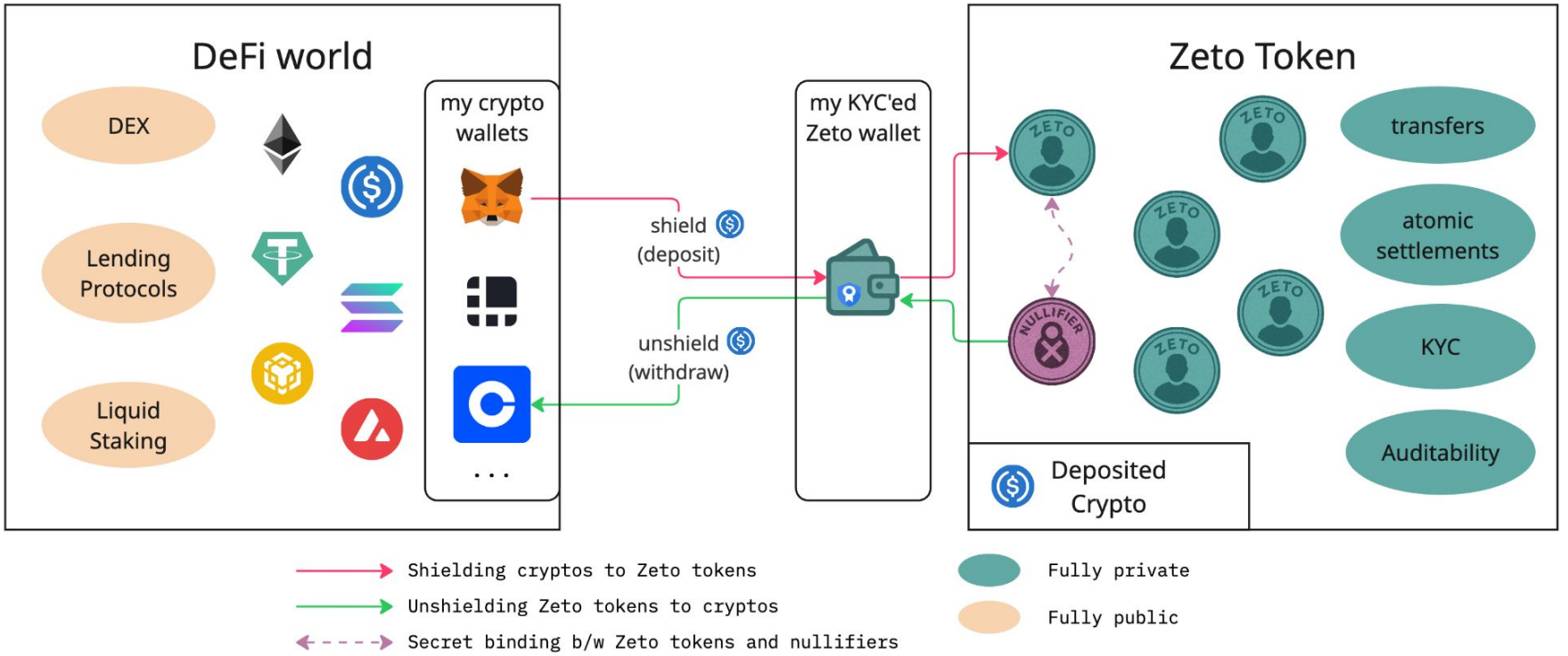


UTXO lifecycle states

Locking Mechanism for multi-step flows



Integration with the DeFi world and Stablecoins



Auditability

Auditing authority must get verifiable data for all transactions

Ideally in real-time

Recall: good solution is sending encrypted secrets (value, ownership, salt) in the transaction payload, decipherable only by the auditing authority

This requires post-quantum security:

- The “Qurrency” protocol, designed by a team of cryptographers including several from UC Berkeley, implemented in Zeto
- Based on NIST standard for Post-Quantum Cryptography FIPS-203 (ML-KEM)

Formal Modeling of Zeto + Qurrency

Sanjam Garg (UC Berkeley)

Guru Vamsi Policharla (UC Berkeley)

Arka Rai Choudhuri (UC Berkeley)

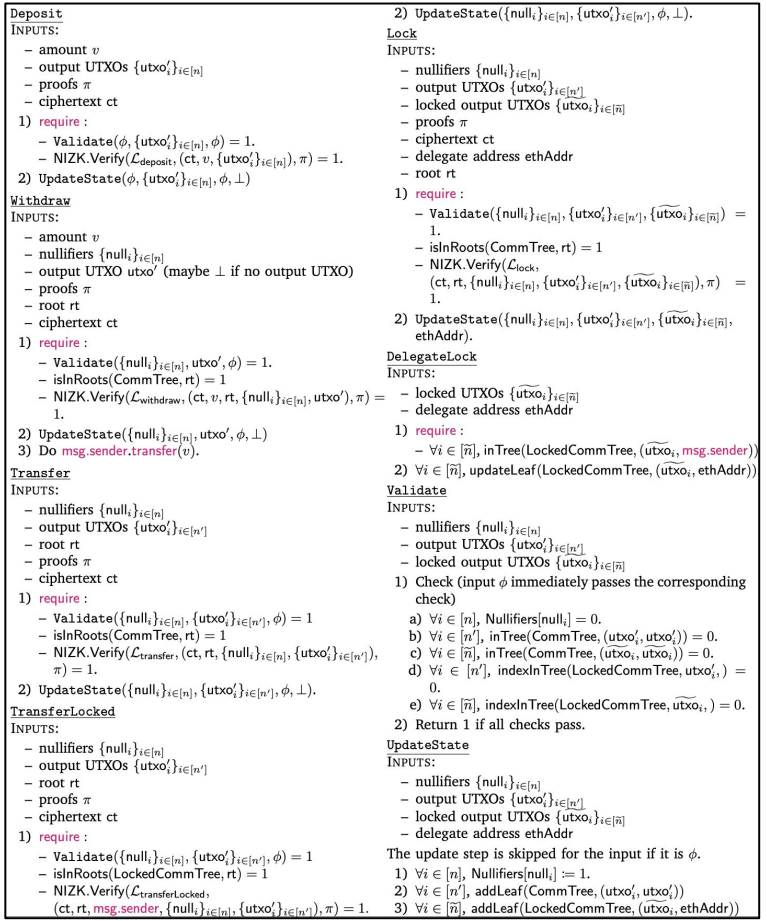
Keewoo Lee (UC Berkeley)

Hart Montgomery (LFDT)

Jim Zhang (Kaleido)

Matthew Gregoire (Kaleido)

Mike Lodder (Lit Protocol)



Deposit

INPUTS:

- amount v
- output UTXOs $\{\text{utxo}'_i\}_{i\in[n]}$
- proofs π
- ciphertext ct

1) require :
   - $\texttt{Validate}(\phi, \{\text{utxo}'_i\}_{i\in[n]}, \phi) = 1$.
   - $\text{NIZK.Verify}(\mathcal{L}_{\text{deposit}}, (\text{ct}, v, \{\text{utxo}'_i\}_{i\in[n]}), \pi) = 1$.
2) $\texttt{UpdateState}(\phi, \{\text{utxo}'_i\}_{i\in[n]}, \phi, \perp)$

Withdraw

INPUTS:

- amount v
- nullifiers $\{\text{null}_i\}_{i\in[n]}$
- output UTXO utxo′ (maybe $\perp$ if no output UTXO)
- proofs π
- root rt
- ciphertext ct

1) require :
   - $\texttt{Validate}(\{\text{null}_i\}_{i\in[n]}, \text{utxo}', \phi) = 1$.
   - $\text{isInRoots}(\text{CommTree}, \text{rt}) = 1$
   - $\text{NIZK.Verify}(\mathcal{L}_{\text{withdraw}}, (\text{ct}, v, \text{rt}, \{\text{null}_i\}_{i\in[n]}, \text{utxo}'), \pi) = 1$.
2) $\texttt{UpdateState}(\{\text{null}_i\}_{i\in[n]}, \text{utxo}', \phi, \perp)$
3) Do msg.sender.transfer(v).

Transfer

INPUTS:

- nullifiers $\{\text{null}_i\}_{i\in[n]}$
- output UTXOs $\{\text{utxo}'_i\}_{i\in[n']}$
- root rt
- proofs π
- ciphertext ct

1) require :
   - $\texttt{Validate}(\{\text{null}_i\}_{i\in[n]}, \{\text{utxo}'_i\}_{i\in[n']}, \phi) = 1$
   - $\text{isInRoots}(\text{CommTree}, \text{rt}) = 1$
   - $\text{NIZK.Verify}(\mathcal{L}_{\text{transfer}}, (\text{ct}, \text{rt}, \{\text{null}_i\}_{i\in[n]}, \{\text{utxo}'_i\}_{i\in[n']}), \pi) = 1$.
2) $\texttt{UpdateState}(\{\text{null}_i\}_{i\in[n]}, \{\text{utxo}'_i\}_{i\in[n']}, \phi, \perp)$.

TransferLocked

INPUTS:

- nullifiers $\{\text{null}_i\}_{i\in[n]}$
- output UTXOs $\{\text{utxo}'_i\}_{i\in[n']}$
- root rt
- proofs π
- ciphertext ct

1) require :
   - $\texttt{Validate}(\{\text{null}_i\}_{i\in[n]}, \{\text{utxo}'_i\}_{i\in[n']}, \phi) = 1$
   - $\text{isInRoots}(\text{LockedCommTree}, \text{rt}) = 1$
   - $\text{NIZK.Verify}(\mathcal{L}_{\text{transferLocked}}, (\text{ct}, \text{rt}, \text{msg.sender}, \{\text{null}_i\}_{i\in[n]}, \{\text{utxo}'_i\}_{i\in[n']}), \pi) = 1$.
2) $\texttt{UpdateState}(\{\text{null}_i\}_{i\in[n]}, \{\text{utxo}'_i\}_{i\in[n']}, \phi, \perp)$.

Lock

INPUTS:

- nullifiers $\{\text{null}_i\}_{i\in[n]}$
- output UTXOs $\{\text{utxo}'_i\}_{i\in[n']}$
- locked output UTXOs $\{\widetilde{\text{utxo}}_i\}_{i\in[\widetilde{n}]}$
- proofs π
- ciphertext ct
- delegate address ethAddr
- root rt

1) require :
   - $\texttt{Validate}(\{\text{null}_i\}_{i\in[n]}, \{\text{utxo}'_i\}_{i\in[n']}, \{\widetilde{\text{utxo}}_i\}_{i\in[\widetilde{n}]}) = 1$.
   - $\text{isInRoots}(\text{CommTree}, \text{rt}) = 1$
   - $\text{NIZK.Verify}(\mathcal{L}_{\text{lock}}, (\text{ct}, \text{rt}, \{\text{null}_i\}_{i\in[n]}, \{\text{utxo}'_i\}_{i\in[n']}, \{\widetilde{\text{utxo}}_i\}_{i\in[\widetilde{n}]}), \pi) = 1$.
2) $\texttt{UpdateState}(\{\text{null}_i\}_{i\in[n]}, \{\text{utxo}'_i\}_{i\in[n']}, \{\widetilde{\text{utxo}}_i\}_{i\in[\widetilde{n}]}, \text{ethAddr})$.

DelegateLock

INPUTS:

- locked UTXOs $\{\widetilde{\text{utxo}}_i\}_{i\in[\widetilde{n}]}$
- delegate address ethAddr

1) require :
   - $\forall i \in [\widetilde{n}]$, $\text{inTree}(\text{LockedCommTree}, (\widetilde{\text{utxo}}_i, \text{msg.sender}))$
2) $\forall i \in [\widetilde{n}]$, $\text{updateLeaf}(\text{LockedCommTree}, (\widetilde{\text{utxo}}_i, \text{ethAddr}))$

Validate

INPUTS:

- nullifiers $\{\text{null}_i\}_{i\in[n]}$
- output UTXOs $\{\text{utxo}'_i\}_{i\in[n']}$
- locked output UTXOs $\{\widetilde{\text{utxo}}_i\}_{i\in[\widetilde{n}]}$

1) Check (input ϕ immediately passes the corresponding check)
   a) $\forall i \in [n]$, $\text{Nullifiers}[\text{null}_i] = 0$.
   b) $\forall i \in [n']$, $\text{inTree}(\text{CommTree}, (\text{utxo}'_i, \text{utxo}'_i)) = 0$.
   c) $\forall i \in [\widetilde{n}]$, $\text{inTree}(\text{CommTree}, (\widetilde{\text{utxo}}_i, \widetilde{\text{utxo}}_i)) = 0$.
   d) $\forall i \in [n']$, $\text{indexInTree}(\text{LockedCommTree}, \text{utxo}'_i,) = 0$.
   e) $\forall i \in [\widetilde{n}]$, $\text{indexInTree}(\text{LockedCommTree}, \widetilde{\text{utxo}}_i,) = 0$.
2) Return 1 if all checks pass.

UpdateState

INPUTS:

- nullifiers $\{\text{null}_i\}_{i\in[n]}$
- output UTXOs $\{\text{utxo}'_i\}_{i\in[n']}$
- locked output UTXOs $\{\widetilde{\text{utxo}}_i\}_{i\in[\widetilde{n}]}$
- delegate address ethAddr

The update step is skipped for the input if it is ϕ.

1) $\forall i \in [n]$, $\text{Nullifiers}[\text{null}_i] := 1$.
2) $\forall i \in [n']$, $\text{addLeaf}(\text{CommTree}, (\text{utxo}'_i, \text{utxo}'_i))$
3) $\forall i \in [\widetilde{n}]$, $\text{addLeaf}(\text{LockedCommTree}, (\widetilde{\text{utxo}}_i, \text{ethAddr}))$

Any Questions?

hmontgomery@linuxfoundation.org