



Blockchain Commons

***Advocating for the Creation of Open, Interoperable,
Secure, and Compassionate Digital Infrastructure***

Blockchain Commons #GordianClubs 2025-10-01



What is Blockchain Commons?

- We are a community interested in self-sovereign control of digital assets and identity
- We bring together stakeholders to collaboratively develop interoperable infrastructure
- We design decentralized solutions where everyone wins
- We are a neutral "not-for-profit" that enables people to control their own digital destiny

Thank you to our Sponsors!

 **Bitmark**

chia

 **CROSSBAR**



 **FOUNDATION**

 **Human
Rights
Foundation**

 **unchained
capital**

 **Zcash**
Community Grants

Become a sponsor! Mail us at team@blockchaincommons.com



Last Meeting

FROST Signing on the Command-Line

August 2025

- Demo of FROST signing of Bitcoin transactions
 - with BDK & ZF FROST Tools
- See <https://developer.blockchaincommons.com/meetings/>



Today's Topics

- Project Xanadu: Before Its Time
- Gordian Clubs: An Introduction
- Gordian Clubs: A Demo



Project Xanadu

"In Xanadu did Kubla Khan
A stately pleasure-dome decree:
Where Alph, the sacred river, ran
Through caverns measureless to man
Down to a sunless sea."

—Samuel Taylor Coleridge



Project Xanadu Background

- First Hypertext Project
- "digital repository scheme for world-wide electronic publishing"
 - 1960, named Xanadu in 1966
 - Very active in late 1970s
 - At Autodesk in 1980s
 - I became involved in early 1990s
- Visionary: Ted Nelson
- Other Luminaries: Roger Gregory, Mark S. Miller & Stuart Greene



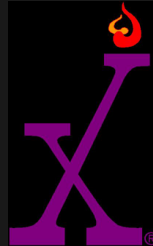
Project Xanadu vs WWW

- Developed before the internet, Xanadu was in many ways superior to the World Wide Web that followed it:
 - Two-Way Links
 - Uniquely Identified Users
 - Uniquely Identified Documents
 - Dynamically Changing Storage
 - Micropayments



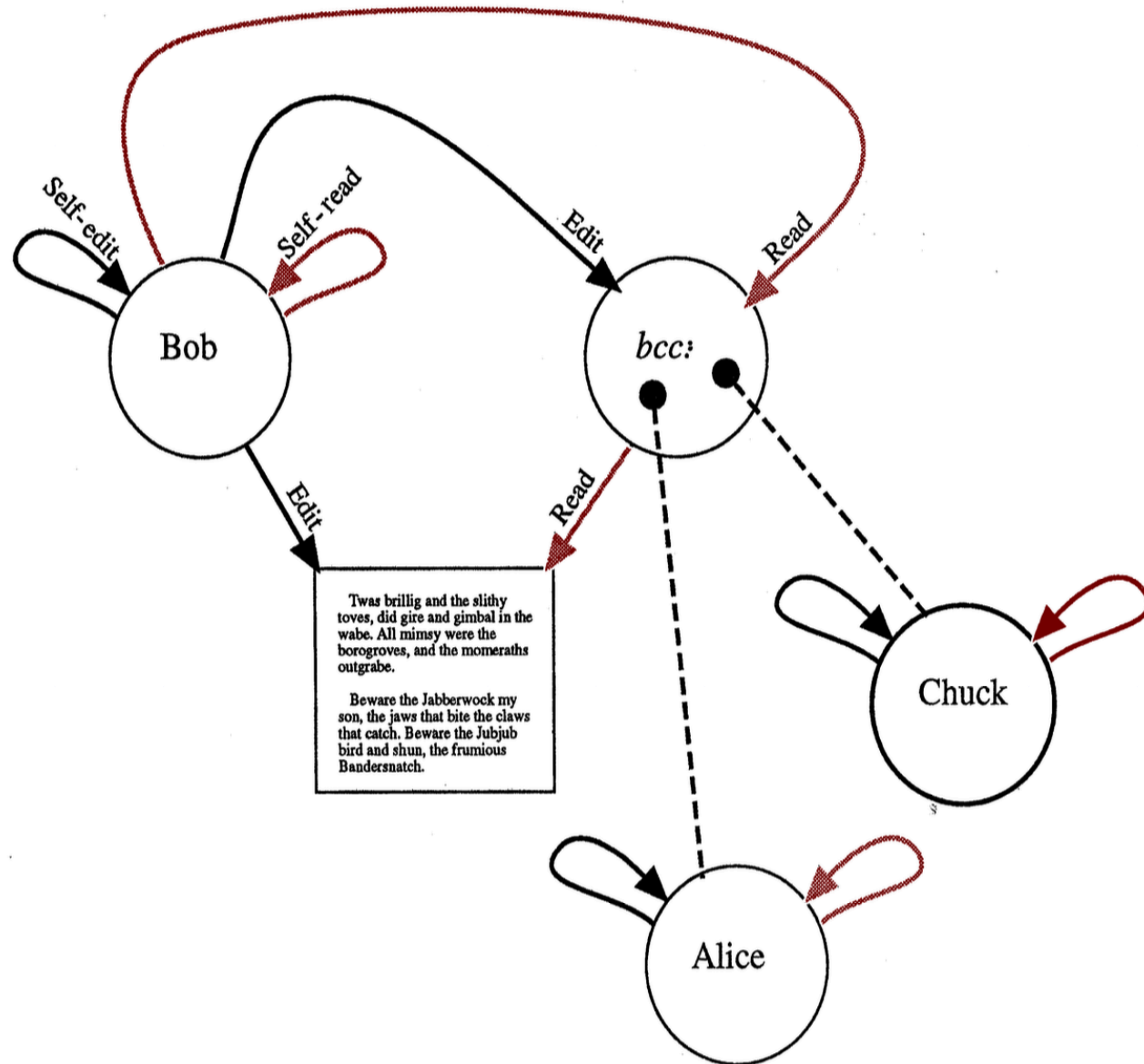
The Xanadu Club System

- I was particularly inspired by the Xanadu Club System:
 - **Public-by-Default:** The root was a public club, openness was default, privacy was a partition
 - **Person as Club of One:** No difference between people and groups
 - **Clubs as Members:** Groups could hold rights and be members of other groups
 - **Recursive Clubs:** Read Clubs and Write Clubs themselves have Read & Write Clubs
 - **"Locksmith" options:** Clubs could grant access through different kinds of locksmiths for different purposes



Why This Was Revolutionary

- **Natural hierarchies** without central administration
- **Flexible governance** through nested permission structures
- **Scriptable access control** using early smart contracts for permissions
- **Unified identity model** with people and groups treated identically
- ***Modeled how human trust hierarchies actually work***





A social system, to a large extent, is a system of rights and responsibilities. Xanadu has an extensive permission system called the *club* system, intended to deal with some of these issues. [Figure 16.6](#) shows a document that Bob can edit. Bob has sent it as a mail message to various people in a blind carbon copy ("bcc") relationship. Alice and Chuck are both members of the *bcc* club of people who have permission to read this document. Bob, though, is the only member who can read or edit the *bcc* club. If this were a *cc* list, Bob would still be the only person who could edit it, but it would be self-reading. Everybody who was a member of such a *cc* club could see who else was a member of that same club.

This demonstrates a principled answer to permissions *meta-issues*--One can distinguish between who can read a document, who can read the list of people who can read a document, and who can read that list, out to any desired degree of distinction (and similarly for the editing dimension). However, infinite regress and needless complexity are avoided by using clubs that are self-reading or self-editing (or both) whenever further distinction is currently not necessary. Should such distinction later become necessary, it can always be introduced by someone with appropriate edit permission to the club in question. Users only grow meta-levels on an as-needed basis.



The Problem with Xanadu Clubs:

~~"Mother may I?"~~ Administrative Fiat

Not

"I can prove I belong." Mathematical Proof



Gordian Clubs: 32 Years in the Making

- **1993:** I proposed cryptographic clubs for Xanadu
 - But: ⚠ "RSA too slow; RSA export-restricted; good code doesn't exist"
- **2001:** ⚠ "Patents too restrictive; patents too expensive"
- **2014:** ⚠ "Schnorr aggregation not safe"
- **2025:** Technology finally caught up to the vision
 - **Gordian Clubs:** Schnorr signatures, FROST, Gordian Envelope, XIDs
 - **Result:** Pure mathematical objects that encode permissions in their structure

Gordian Club Vision: Recursive Cryptographic Capabilities

Mathematical Delegation

- **Clubs granting capabilities to other clubs**
 - No administrative intermediaries
 - Pure cryptographic authorization chains
- **Recursive permission structures**
 - Club A grants read access to Club B
 - Club B can delegate subsets to Club C
 - All enforced by mathematics, not code
- **Complete Xanadu realization**
 - Ted Nelson's hypertext with cryptographic enforcement
 - "You are what you can prove you can access"

Rather than being simple documents, Clubs are autonomous objects.

Gordian Clubs: Autonomous Access

- You receive an **Edition** produced by a Gordian Club. The content of the **Edition** is encrypted and signed.
 - Transport agnostic: Internet; P2P; TOR; NFC; satellite; sneaker-net
- You apply your cryptographic proofs to obtain READ privileges to the **Edition** content
 - The decrypted **Edition** content may reveal nested **Editions** from the same or other Clubs, which you may or may not have access to.
- You update an **Edition** by authoring new **Editions** in particular clubs that must be accepted as authoritative due to your cryptographic proofs.

No Server, No Database, No Phoning Home!

Access Control: Cryptographic OCaps

Meeting these goals required the replacement of traditional (Xanadu) OCaps with cryptographic OCaps.

Traditional Ocaps: Software enforces capability delegation **Cryptographic Ocaps:** Mathematics enforces capability delegation

- **How Gordian Clubs Bridge This:**
 - **Permits** = cryptographic capabilities for read access
 - **Schnorr Signatures** = foundation of composable smart signature protocols
 - **Adaptor signatures** = aka “*Scriptless Scripts*” - conditional authorization protocols
 - **FROST thresholds** = group decision capabilities
- **For the Ocap Community:**
 - Same principles, cryptographic enforcement instead of code
 - **Capabilities become mathematical objects**
 - No confused deputy problem - math doesn't lie

The Schnorr Foundation: Mathematical Elegance

Many of the capabilities that we need are supported through Schnorr signatures.

- **The Key Property: Signature Indistinguishability**
 - Single-party signature = Multi-party signature (same verification)
 - Simple workflow = Complex workflow (same mathematical structure)
- **Mathematical linearity** enables seamless composition
- Why This is Extraordinary
 - Start with basic cryptographic objects
 - Evolve to sophisticated multi-party protocols
 - **External systems never need to change**
 - Same verification logic, infinite internal complexity

The Schnorr Revolution: True Composability

- **Schnorr's Super Power: Composable Cryptographic Primitives**
 - Layer complexity without breaking compatibility
 - All primitives use the same mathematical foundation
 - All primitives produce indistinguishable signatures
 - All primitives can be combined and layered
- **Cryptographic LEGO blocks** that actually fit together

Cryptographic LEGO Blocks: The Schnorr Ecosystem

- **Initial Schnorr LEGO Blocks in Gordian Clubs:**
 - **#1: Adaptor Signatures** → Scriptless Scripts (conditional logic)
 - **#2: FROST** → Threshold operations (group decisions)
- **Future LEGO Blocks:**
 - **MuSig2** → Key aggregation
 - **Blind Signatures** → Privacy
 - **Predicate Logic** → Complex conditional authorization
 - **More!**
- We can build complex systems from simple, interoperable parts!

LEGO Block #1: Scriptless Scripts

- **Adaptor Signatures: Conditional Logic in Pure Math**
 - *"If condition X, then unlock Y"* → Mathematical proof
 - *"Atomic operations across systems"* → Single signature verification
 - *"Complex multi-party workflows"* → Elegant cryptographic composition
- **The Developer Opportunity: Infrastructure-Free Logic**
 - Embed conditions directly into Schnorr signatures
 - Chain operations without external coordination
 - ***Build workflows that are pure mathematics***
- **Use Cases:**
 - Conditional access, delegation, attenuation, atomic swaps
 - Cross-system coordination, delegation chains
 - ***Any logic expressible in math*** - no servers required

Limitations of Schnorr Adaptor Signatures

- **Limited to signature mathematics** - bounded by what elliptic curves can express
- **No loops or iteration** - mathematics is finite, not computational
- **No conditional branching on external events** - logic must be deterministic
- **No mutable state** - each operation creates new mathematical objects
- **These "limitations" offer freedom:**
 - Freedom from platform lock-in, censorship, surveillance
 - ***Mathematical certainty replacing administrative whim***

LEGO Block #2: FROST Threshold Operations

- **FROST: Flexible Round-Optimized Schnorr Threshold**
 - **M-of-N signatures** that look like single signatures
 - **Privacy-preserving** - can't tell who participated
 - **Same verification logic** as single-party signatures
- **Two Core Applications:**
 - **Threshold Signing** → Group authorization for updates
 - **Group-Managed Provenance** → Shared custody of version history
- **Developer Power:**
 - Start with single-party authorization
 - **Seamlessly upgrade** to group governance
 - External systems never know the difference

FROST in Practice: Group Governance Made Simple

- **Signing Ceremonies:**
 - **Coordinate off-chain** → Generate signatures
 - **Single signature result** → Same as individual signing
 - **No coordinator required** → Can be peer-to-peer protocol
- **Provenance Chain Management:**
 - **Shared custody** of version history
 - **No single point of failure** for critical records
 - **Deterministic advancement** without central authority
- **Use Cases:**
 - **Organizational governance** without servers
 - **Community decisions** with cryptographic proof
 - **Succession planning** built into the mathematics

Limitations of FROST Threshold Operations









- **Coordination complexity** - requires secure multi-party communication
- **Network availability** - signing ceremonies need participant connectivity
- **Key management burden** - key distributed across multiple parties
- **No atomic composition** - can't combine with other protocols mid-ceremony
- **These “limitations” offer freedom:**
 - Freedom from single points of failure and control
 - *Democratic governance through cryptographic consensus*

Future LEGO Blocks: The Expanding Toolkit

- **Each new block** builds on the same Schnorr foundation
 - **Infinite composability** from finite primitives
- **Next (2026?):**
 - **MuSig2** → Key aggregation with accountability
 - **Additional object capabilities** such as attenuation
 - **Blind Signatures** → Privacy-preserving authorization
 - **Predicate Logic** → Complex conditional authorization
- **For Developers:**
 - **Build on stable foundation** being adopted by Bitcoin and other projects
 - ***Mathematical certainty in an uncertain world!***


From Schnorr Foundation to Gordian Implementation

Now you understand our goals & general design. Here's how we built them:

- **Selected our data structures:**  dCBOR +  Gordian Envelope
- **Chose our Schnorr LEGO blocks:**  FROST +  adaptor signatures
- **Added identity layer:**  XIDs for decentralized member management
- **Secured our data:**  Envelope permits
- **Created coordination protocols:**  Provenance Marks &  GSTP

Gordian Stack: Data Foundations




Here are our specific technology choices for creating Gordian Clubs within the Gordian Stack:

-  **dCBOR (Deterministic CBOR)**
 - Binary encoding ensures that identical semantics produces identical bytes
 - Foundation for cryptographic verification and content addressing
-  **Gordian Envelope (Smart Document Architecture)**
 - A hash-tree of hash-trees
 - Subject-predicate-object semantic structure
 - Selective disclosure through *elision* or *encryption*
 - Reveal only what's needed
 - Radically recursive — everything is an Envelope

Gordian Stack: Schnorr LEGO Blocks

- 🤝 **FROST (Flexible Round-Optimized Schnorr Threshold)**
 - M-of-N threshold signatures indistinguishable from single signatures for **online** group signing
 - Privacy-preserving: Can't tell which specific members participated
 - Enables group decision-making without revealing individual votes
- ✍️ **Adapter Signatures**
 - Conditions embedded in signatures

Gordian Stack: Identity & Access Control

-  **XIDs (eXtensible IDentifiers)**
 - 32-byte cryptographic identifiers derived from keys ( 7e1e25d7...)
 - Resolve to XID Documents with communication keys, endpoints, permissions, etc.
 - Support key rotation while maintaining stable identity
-  **Permits: Multiple Ways to Access Same Content**
 - **XID permits** - Identity-based access with key rotation support
 - **Public key permits** - Direct cryptographic access to specific keys
 - **SSKR threshold shares** (2-of-3, 3-of-5, etc.) for **offline** key reconstruction
 - **Password permits** for simple access scenarios
 - Each permit type decrypts to the same base symmetric key

Gordian Stack: Coordination & Verification

- **🔒 Provenance Marks: Cryptographic Edition Ordering**
 - Sequential, tamper-evident chains of document versions
 - Each mark cryptographically links to content digest and previous mark
 - **Genesis Mark (#0) → Mark #1 → Mark #2 →** continuous chain
 - Enable write access validation and update authorization
 - Prevent backdating or unauthorized modification
 - Human-readable names (🔒 KNOB BETA AQUA NOON) for easy verification
- **🔒 GSTP: Secure Multi-Party Coordination**
 - Transport-agnostic protocol (works over Bluetooth, QR, NFC)
 - Encrypted message exchange with state preservation
 - Enables secure coordination without infrastructure

How Gordian Clubs Use These Technologies

- **When You Create a Club:**
 - **XID** identifies the club entity (publisher) across updates
 - Content encrypted and stored in **Gordian Envelope** using **dCBOR**
 - Multiple **permits** allow different access methods (keys, passwords, shares)
- **When Members Make Decisions:**
 - **FROST** threshold signatures authorize updates without revealing who voted
 - **Provenance marks** create tamper-evident history of all changes
 - **GSTP** coordinates secure communication between members
- **The Result: Autonomous Cryptographic Objects**
 - No servers, no databases, no central authority
 - Mathematical proofs replace administrative control
 - Unstoppable, private, censorship-resistant collaboration

The Structure of a Gordian Club



A Gordian Club is a layered onion, with some information widely readable and some not:

1. **Club Envelope:** dCBOR-encoded structure containing everything
2. **Public Metadata:** Visible information (club ID, version, etc.)
3. **Encrypted Payload:** The actual club content and member data
4. **Access Layer:** Collection of permits for different entry methods
5. **Governance Layer:** FROST signatures and provenance chain

Result: Self-contained cryptographic object requiring no external infrastructure

Read vs. Write Access Model

Two Distinct Permission Types:

-  **Read Access**
 - Decrypt and view current club content
 - Multiple permit types provide different access methods
 - One-time or ongoing access depending on permit type
-  **Write Access**
 - Create new editions with updated content or membership
 - Requires FROST threshold signatures from existing write group
 - Provenance marks cryptographically ensure ordering

This separation enables flexible governance while maintaining security

The Power of Autonomy

Autonomous Cryptographic Objects enable ...

- **Unblockable Access**
 - No server can be taken down to block access
- **Perfect Privacy**
 - No logs, no tracking, no surveillance possible
- **Disaster Resilience**
 - Works during internet outages or infrastructure failures
- **Censorship Resistance**
 - No authority can revoke mathematical proofs
- **True Ownership**
 - Control rests with keyholders, not platform operators

Platform Independence






- Coordination that can't be algorithmically suppressed
- Communities that can't be deplatformed
- ***Mathematical certainty replacing platform dependency***

Use Cases

- **When Infrastructure Can't Be Trusted:**
 - Dissidents organizing without surveillance
 - Long-term archival that outlives companies
 - Emergency coordination during outages
 - ***Mathematical proofs endure when servers don't***

What You'll See in the Demo

Wolf will demonstrate the proof-of-concept implementation:

1.  **Identity Setup** - Creating XIDs for Publisher, Alice, and Bob
2.  **Genesis Edition** - Encrypting "Welcome to Gordian Club!" with multiple permits
3.  **Access Methods** - Alice decrypts with XID permit, same content via SSKR shares
4.  **Edition Updates** - Publisher creates "Second Edition" with provenance continuity
5.  **Verification** - Proving authenticity and sequence without servers

Watch for: Self-contained data artifacts working without any infrastructure

SSKR Threshold Reconstruction in Action

Demo will show 2-of-3 threshold shares:

- **Publisher creates** 3 SSKR shares when sealing content
- **Any 2 shares** can reconstruct the decryption key
- **Same plaintext** emerges from SSKR as from XID permits
- **Offline capability** - no coordination needed between shareholders

Key insight: Multiple paths to same content, different security models

Edition Structure You'll See

Encrypted envelope contains:

```
{
  ENCRYPTED [
    'isA': "Edition"
    'hasRecipient': SealedMessage // Alice's XID permit
    'hasRecipient': SealedMessage // Bob's pubkey permit
    "club": XID(8f5980be)
    'hasRecipient': SealedMessage // Proves the publishing order of this Edition
    'provenance': ProvenanceMark(c14fafa0)
  ]
} [
  'signed': Signature // Publisher's cryptographic seal
]
```

Plus separate SSKR share envelopes for threshold access

Verification Steps You'll See

Wolf will demonstrate cryptographic validation:

- **clubs edition verify** - Proves edition authenticity and signature validity
- **clubs edition sequence** - Validates provenance chain continuity
- **Content decryption** - Shows multiple permit types accessing same data
- **Provenance progression** - Genesis mark → Mark #1 with content linking

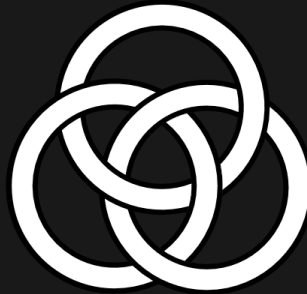
No servers consulted, no external validation needed



clubs-cli-rust Demo

You now have the foundation to understand what you'll see next...

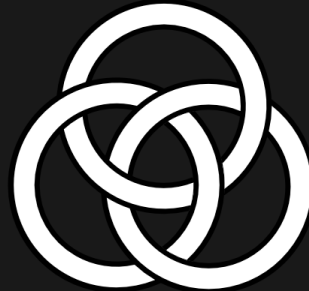
Wolf will demonstrate these concepts working together in a real implementation



Two Rust Repositories

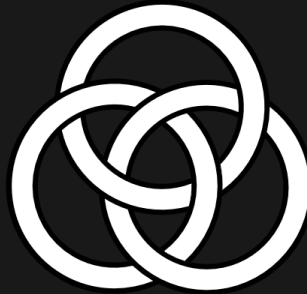
What Wolf just demonstrated is built on:

- 📖 **clubs-rust** - Core cryptographic library
 - Edition creation, FROST signatures, permit handling
 - GitHub: <https://github.com/BlockchainCommons/clubs-rust>
- ⚡ **clubs-cli-rust** - Command-line interface (what you saw in action)
 - Complete demo workflow in `demo-log.md`
 - GitHub: <https://github.com/BlockchainCommons/clubs-cli-rust>



Getting Started

- To install the tools:
 - `cargo install envelope-cli provenance-cli clubs-cli`
- Full walkthrough:
 - <https://github.com/BlockchainCommons/clubs-cli-rust/blob/master/demo-log.md>



Future Work (I) - FROST Editions

- **Using FROST for Editions:**
 - Create the symmetric key to encrypt **Edition** content.
 - Create the next provenance mark in a chain without a secret seed.
 - Sign the **Edition** Gordian Envelope

Status: proof of concept code working for FROST groups to collaboratively conduct FROST rounds for editions.



Future Work (II) - FROST Workflows

- **FROST Publish:** integrate these pieces into a workflow for FROST groups to publish *Editions* indistinguishable from those produced by a single-entity publisher.
- **FROST Coordinator:** create a server, *hubert*, which is a "dumb" message passing hub for GSTP messages.
 - Allows encrypted message passing between participants without having any idea what the messages mean.
 - Supports single-cast, multi-cast, and store-and-forward of GSTP messages.
 - Simplifies the communication needed to conduct the FROST rounds needed to produce new *Editions*.
 - Provides flexible infrastructure for the development of future protocols.

Status: We already have working libraries for FROST and GSTP, just need to integrate with code for Gordian Clubs.



Future Work (III) - Cryptographic Ocaps

What you saw: Static permissions via permits and signatures

What's possible: Dynamic capabilities using adaptor signatures

- Delegate authority without sharing keys
- Attenuation (restricting permissions when delegating)
- Composition (chaining capabilities across systems)

Status: Research phase—novel cryptographic constructions requiring formal proofs

We need cryptographers, engineers, and visionaries to build this together.

Ocap Delegation Without Key Sharing

- **Adaptor signatures enable a powerful pattern:**
 - Alice can delegate her read or write authority to Bob without sharing keys and without creating a new edition.
- **"Naive" = Conceptually Sound, Needs Formal Security Analysis**
 - Like naive Schnorr aggregation (which can leak keys → thus the MuSig2/MuSig-DN protocols), these examples demonstrate the core pattern but require cryptographic proofs before production use.
- **Here's how...:**

Naive Read Delegation via Adaptor Signatures

- **Goal:** Alice (with read access) delegates to Bob without sharing keys OR updating current edition
 1. **Alice creates an incomplete signature**
 - Generates random secret t and commits to it: $T = t \cdot G$
 - Encrypts symmetric key: $k_{enc} = k \oplus \text{hash}(t, B, \text{edition_id})$
 - Creates adaptor signature hiding t that only Bob can complete
 2. **Only Bob can complete it**
 - Bob uses his private key b to complete the signature
 - Completion reveals the secret t to Bob (and only Bob)
 3. **Result: Delegated read authority**
 - Bob can decrypt this edition using revealed t
 - Computes: $k = k_{enc} \oplus \text{hash}(t, B, \text{edition_id})$
 - No key sharing required
 - Alice retains her original access
- **Key insight:** The incomplete signature hides a secret that unlocks decryption—a cryptographic "locked box" that only Bob's key can open.

Naive Write Delegation via Adaptor Signatures

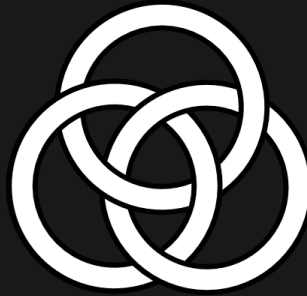
- **Goal:** Alice (with write access) delegates to Bob without sharing keys OR updating current edition
 1. **Alice creates an incomplete signature**
 - Standard Schnorr: $s = r + H(m) \cdot a$
 - Adaptor version: $s' = r + \text{tweak} + H(m) \cdot a$
 - The "tweak" is locked to Bob's public key
 2. **Only Bob can complete it**
 - Bob uses his private key b to derive the tweak to produces a valid signature from Alice
 - Proves both: Alice's authorization + Bob's identity
 3. **Result: Delegated write authority**
 - Bob can create a new edition
 - Each shows Alice's valid signature
 - No key sharing required
 - Alice retains her original authority
- **Key insight:** The incomplete signature is a cryptographic "blank check" that only Bob can fill out, creating mathematical proof of delegation.

Future Work (IV) - Vetting of FROST Provenance Mark VRF Cryptography

- **The Problem:** When a FROST group publishes a new edition's **P** Provenance Mark, we need a signature that produces a unique, unpredictable symmetric key that everyone can verify came from the group.
- **Our Solution:** Use a VRF (Verifiable Random Function) where:
 - The group shares a secret no individual knows
 - Each document creates a unique mathematical challenge
 - The group collaboratively solves it, producing an apparently random number that's:
 - Unique to this group
 - Publicly verifiable
 - Unpredictable yet deterministic
- **Why It Matters:**
 - Creates cryptographic proof of group consensus
 - Forms an unbreakable chain of documents
 - No individual can cheat or predict future keys
- **Analogy:** A combination lock where the group collectively knows the combination, but no individual does—each use produces a new, verifiable number only they could create.

Future Work (IV) - Vetting of FROST Provenance Mark VRF Cryptography (continued)

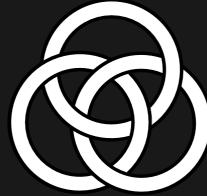
- **Ciphersuite:** ZCash `frost-secp256k1-tr` with secp256k1 curve
- **Group Setup:** t -of- n threshold Schnorr with shared public key $X = x \cdot G$
- **VRF Message:** $m_j = H(\text{"PMVRF-secp256k1-v1"} \parallel X \parallel \text{chain_id} \parallel S_{j-1} \parallel j)$
- **Hash-to-Curve:** $H_j = H2C(m_j)$ using SSWU with domain separation
- **VRF Output:** $\Gamma_j = x \cdot H_j$ (computed via FROST threshold ceremony)
- **DLEQ Proof:** Chaum-Pedersen π_j proving $\log_G(X) = \log_{H_j}(\Gamma_j)$
 - Challenge: $e = H_2(X \parallel \Gamma_j \parallel A \parallel B \parallel \text{"FROST-VRF-DLEQ-2025"})$
 - Response: $z = k + e \cdot x$
- **Key Derivation:** $\text{key}_j = \text{SHA256}(\text{"PMKEY-v1"} \parallel \Gamma_j)[\text{: resolution}]$
- **Ratchet State:** $S_j = \text{SHA256}(\text{"PMSTATE-v1"} \parallel S_{j-1} \parallel \text{expand}(\text{key}_j))$
- **Genesis:** $S_0 = \text{SHA256}(\text{"PM-Genesis"})$
- **Properties:** Deterministic, roster-invariant, publicly verifiable



Get Involved with Gordian Clubs

- Everyone
 - Let us know what your use cases are for this technology!
 - Where practical problems can true autonomous data objects solve today?
- Cryptographers
 - We have novel constructions that need cryptographic proofs!
- Engineers
 - We would like peer review of our protocols and code.
- Companies
 - We need partners interested in this tech!
- Patrons
 - We need financial support!

Mail me at team@blockchaincommons.com



"Some dreams just need the right tools to become real."

Today, we have those tools. Tomorrow, developers like you will use them to build coordination systems that preserve human dignity.

Ready to build uncapturable infrastructure?

www.BlockchainCommons.com



Christopher Allen (@ChristopherA)