Give and Take: The Evolving Relationship between Security and Blockchains

Ghada Almashaqbeh

University of Connecticut

October 2024

The Decentralized Internet—Web 3.0

Decentralized currency exchange medium	Global virtual computer	Decentralized Internet
Bitcoin	Ethereum	Web 3.0
2009	2015	2015 & onward?!

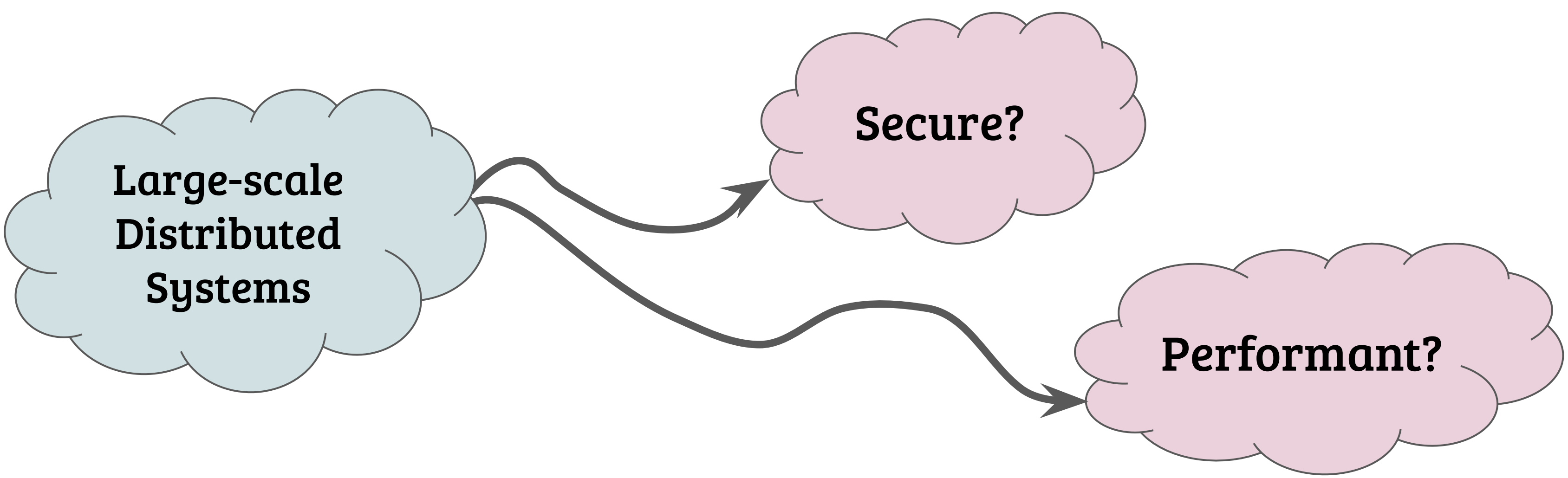
The Decentralized Internet—Web 3.0

Decentralized currency exchange medium	Global virtual computer	Decentralized Internet
Bitcoin	Ethereum	Web 3.0

Large-scale Distributed Systems

The Decentralized Internet—Web 3.0

Decentralized currency exchange medium	Global virtual computer	Decentralized Internet
Bitcoin	Ethereum	Web 3.0



Research Frontiers in Cryptography

The Blockchain Model

Append-only log

Secure distributed ledger

Automated contract term enforcement

Monetary incentives

Open access and dynamic participation

- Implement broadcast channel, indirect communication, sending messages to the future.
- New flavors of MPC: Gage MPC, YOSO MPC, Fluid MPC.
- Circumventing impossibility results.
- …

Two Instances: Give and Take

This talk

- Secure performance boosting for Web 3.0
  - chainBoost
- Cryptographic primitives with new features
  - RelaySchnorr

chainBoost: A Secure Performance Booster for Blockchain-based Resource Markets

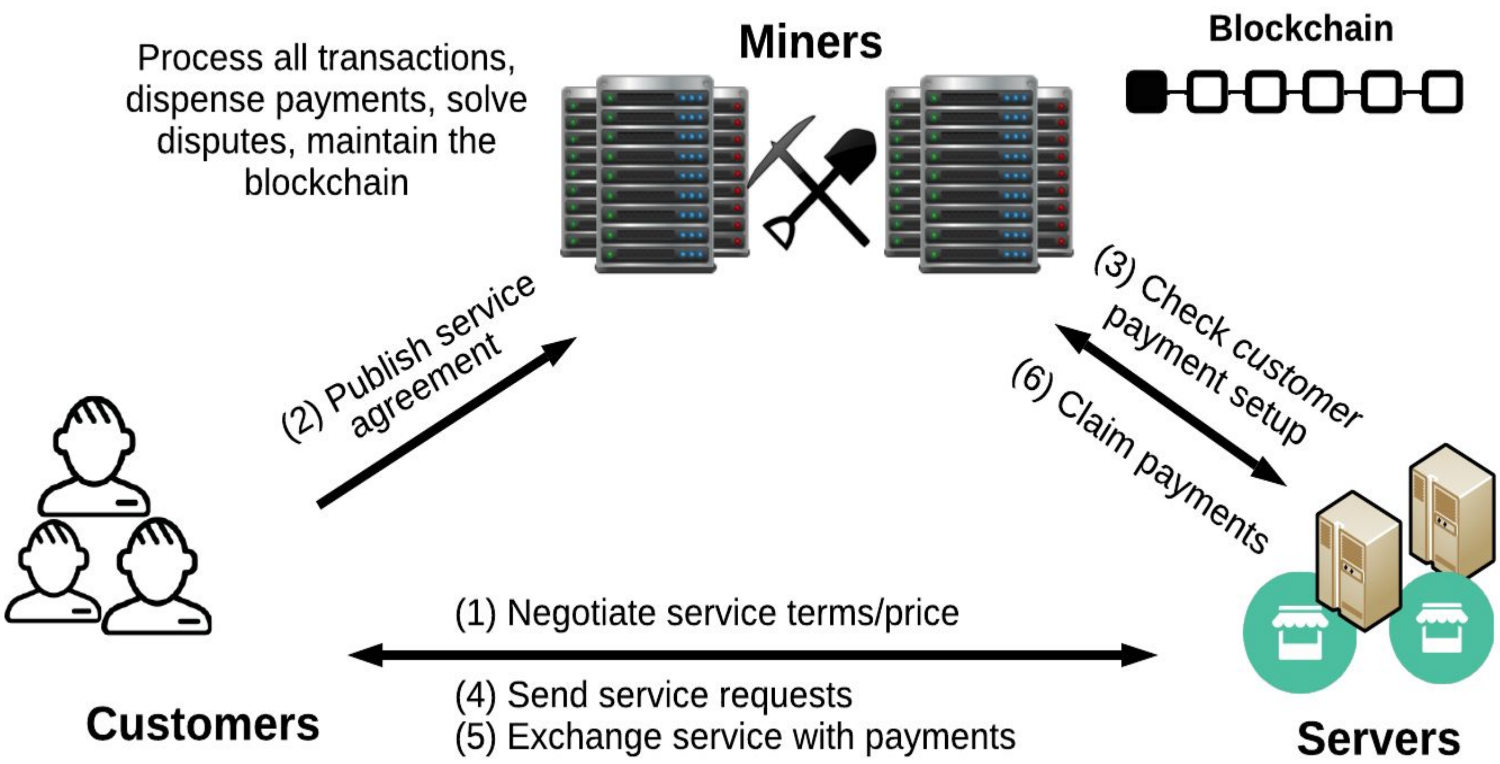
*Z. Motaqy, M. Najd, and G. Almashaqbeh, *chainboost: A secure performance booster for blockchain-based resource markets*, in IEEE EuroS&P 2024 (https://arxiv.org/abs/2402.16095).

Our focus

Decentralized Resource Markets

- Provide distributed services on top of the currency exchange medium.
  - E.g., computation outsourcing, file storage and retrieval, video transcoding, etc.
- They create open-access markets for trading resources.

Decentralized Resource Markets



They are a Large Industry …

Filecoin

livepeer

golem

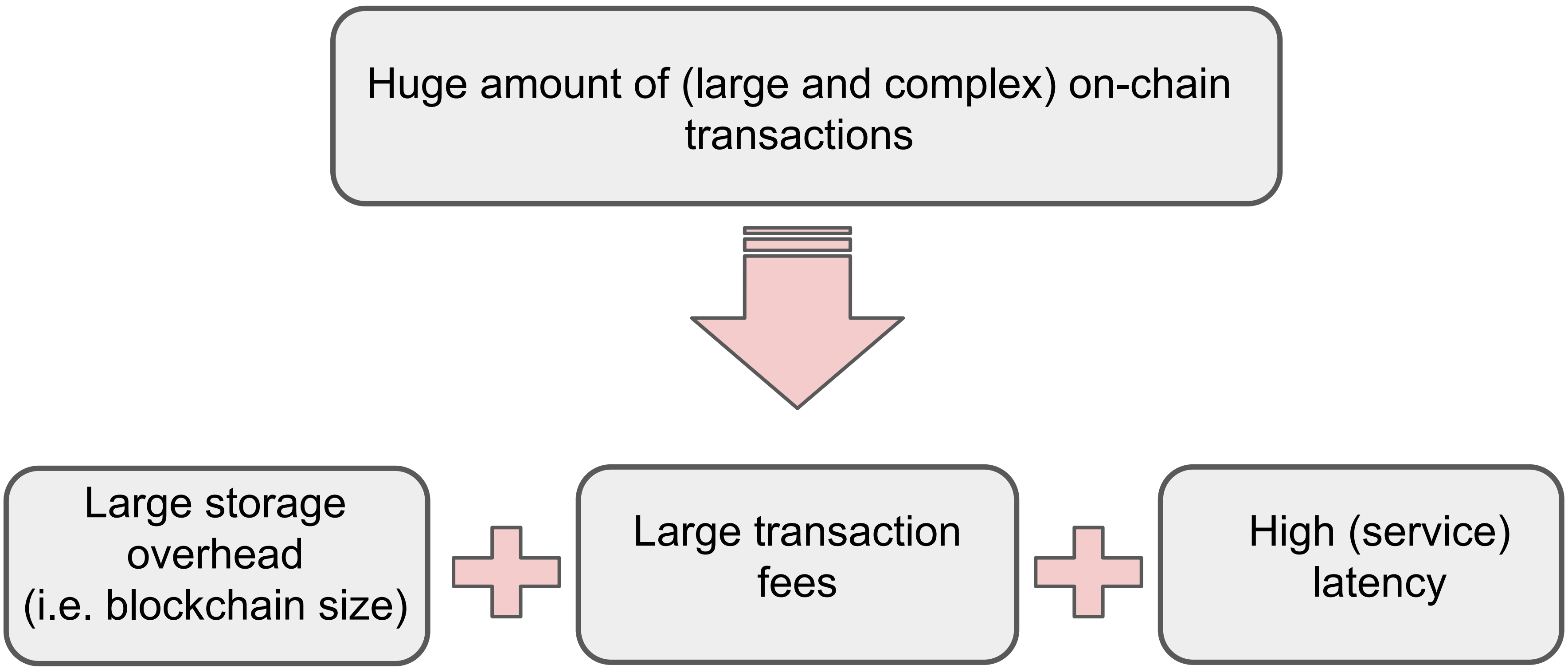
STORJ

Interesting Topics

- Market matching strategies
- Fair exchange protocols
- Proof of service delivery
- Collateral management policies
- Dispute solving
- Privacy
- …

… and a Huge Scalability Problem!

Our focus



Can we build a generic and secure efficiency solution for decentralized resource markets that

1. has a unified architecture and interfaces, and
2. allows for service-specific semantics, while
3. preserving the public verifiability, decentralization, transparency, etc., that are expected of a Web 3.0 protocol?

Limitations of Existing Solutions

Limitations of Existing Solutions

- *Sharding ⇒ High volume of cross-shard transactions!*

Limitations of Existing Solutions

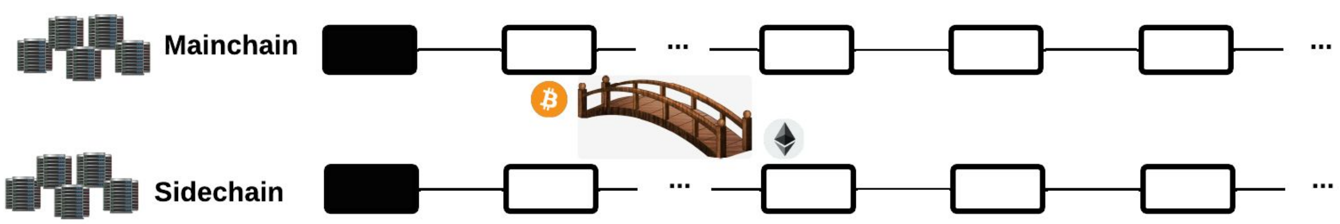
- *Sharding ⇒ High volume of cross-shard transactions!*
- *Zero-knowledge (ZK) rollups ⇒ ZK proofs are expensive!*
- *Optimistic rollups ⇒ Long contestation periods + incentive compatibility issues!*

Limitations of Existing Solutions

- *Sharding ⇒ High volume of cross-shard transactions!*
- *Zero-knowledge (ZK) rollups ⇒ ZK proofs are expensive!*
- *Optimistic rollups ⇒ Long contestation periods + incentive compatibility issues!*
- *Sidechains ⇒ Mainly focused on two-way peg and independent sidechains!*

Independent Sidechains

- Each chain has its own domain, users, network protocol, etc.
- This prevents workload sharing, arbitrary data exchange, or reacting to events happening on the other chain.
- Two-way peg is basically sending currency from chain A to chain B and vice versa.

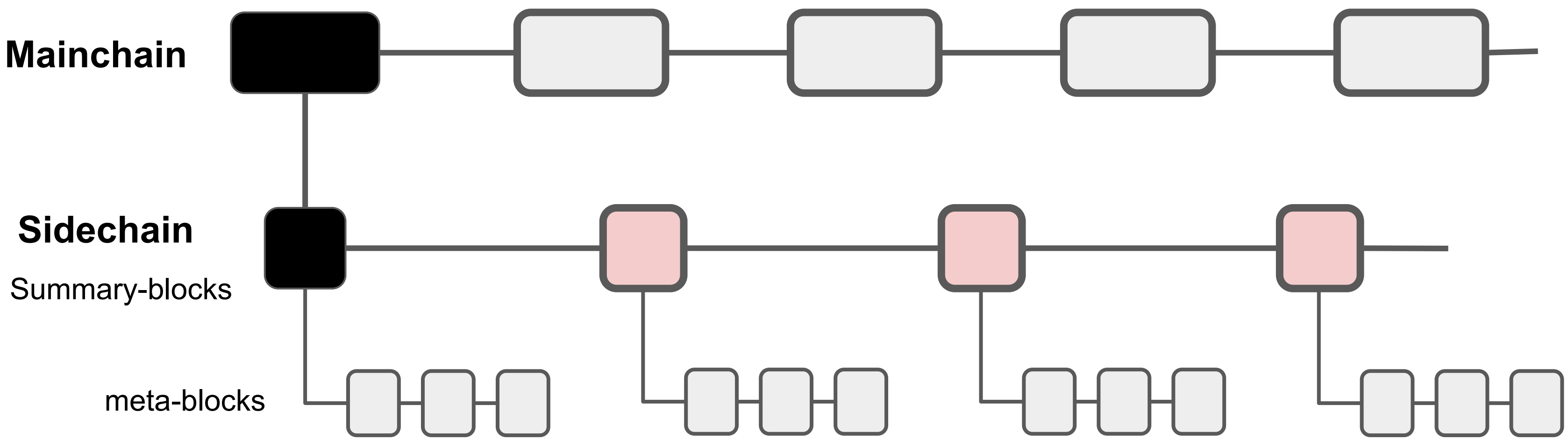


Still, sidechains have the potential to solve the problem!

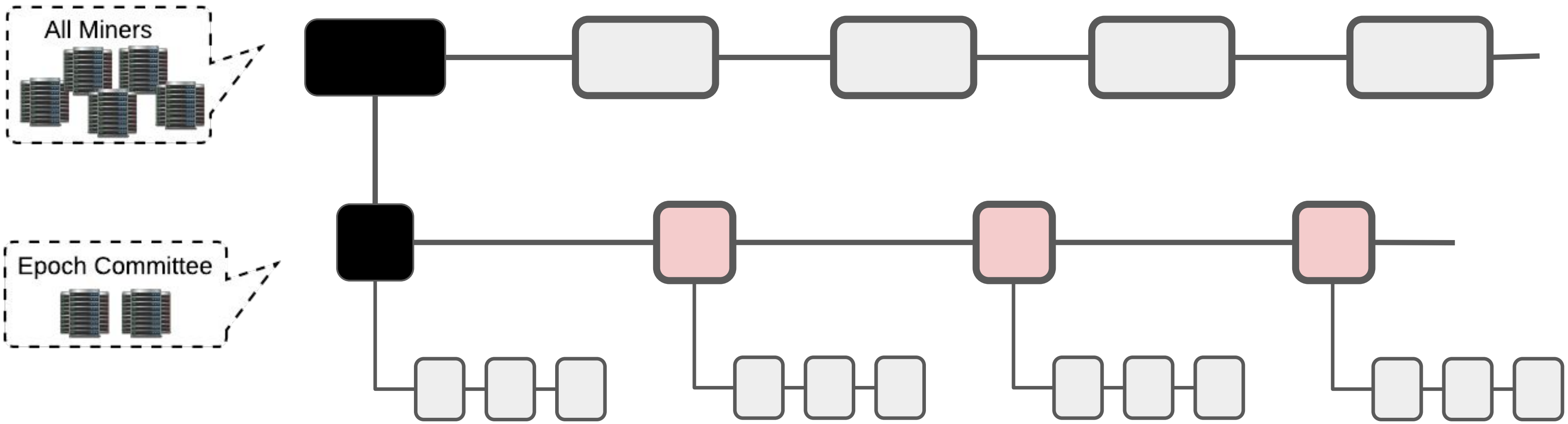
This work

chainBoost—a new dependent sidechain architecture

chainBoost Framework



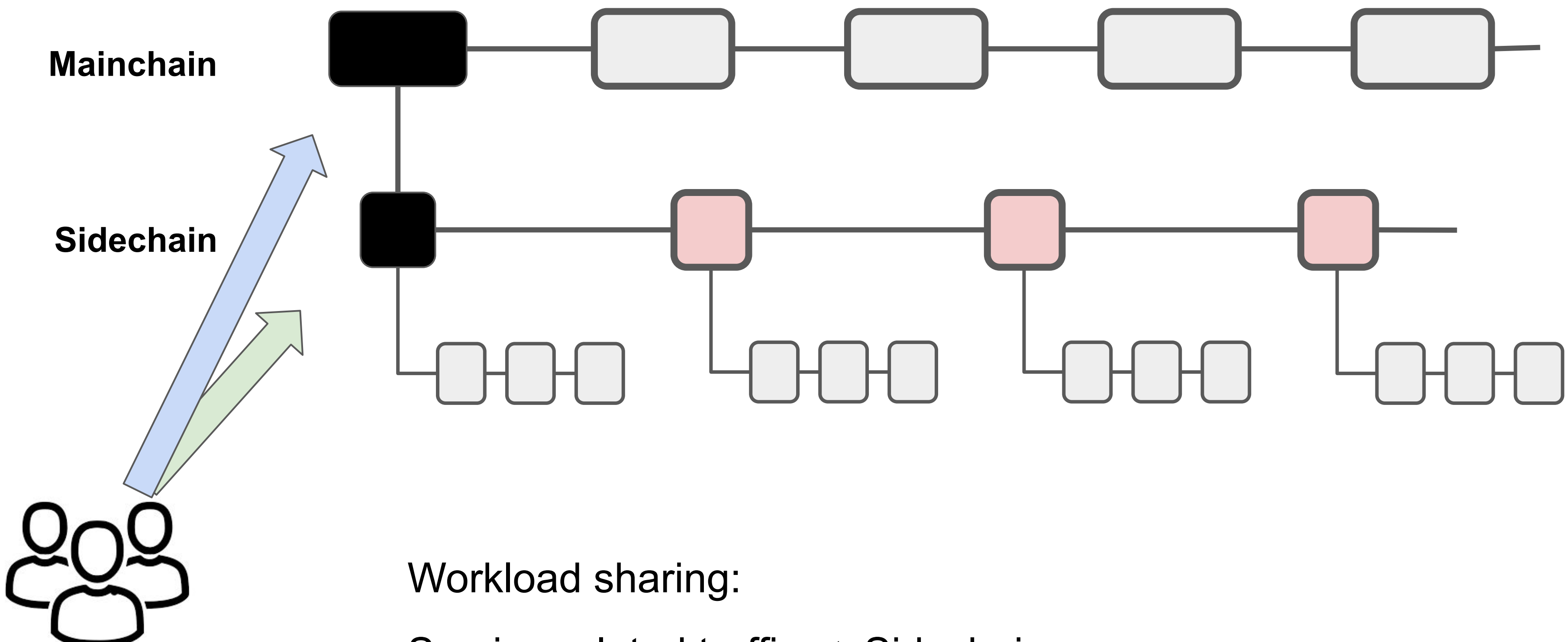
chainBoost Framework



Works in epochs and rounds

A new sidechain committee is elected for each epoch

chainBoost Framework

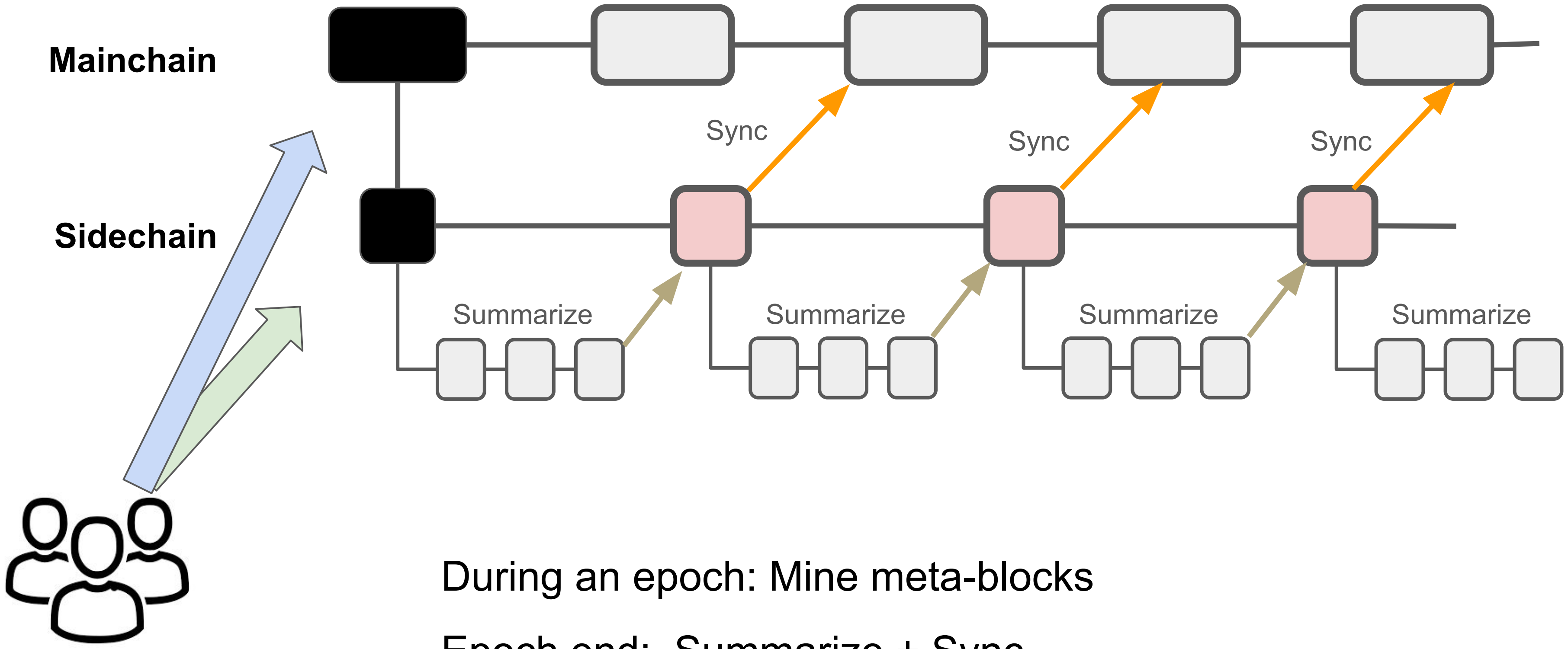


Workload sharing:

Service-related traffic ⇒ Sidechain

Rest of traffic ⇒ Mainchain

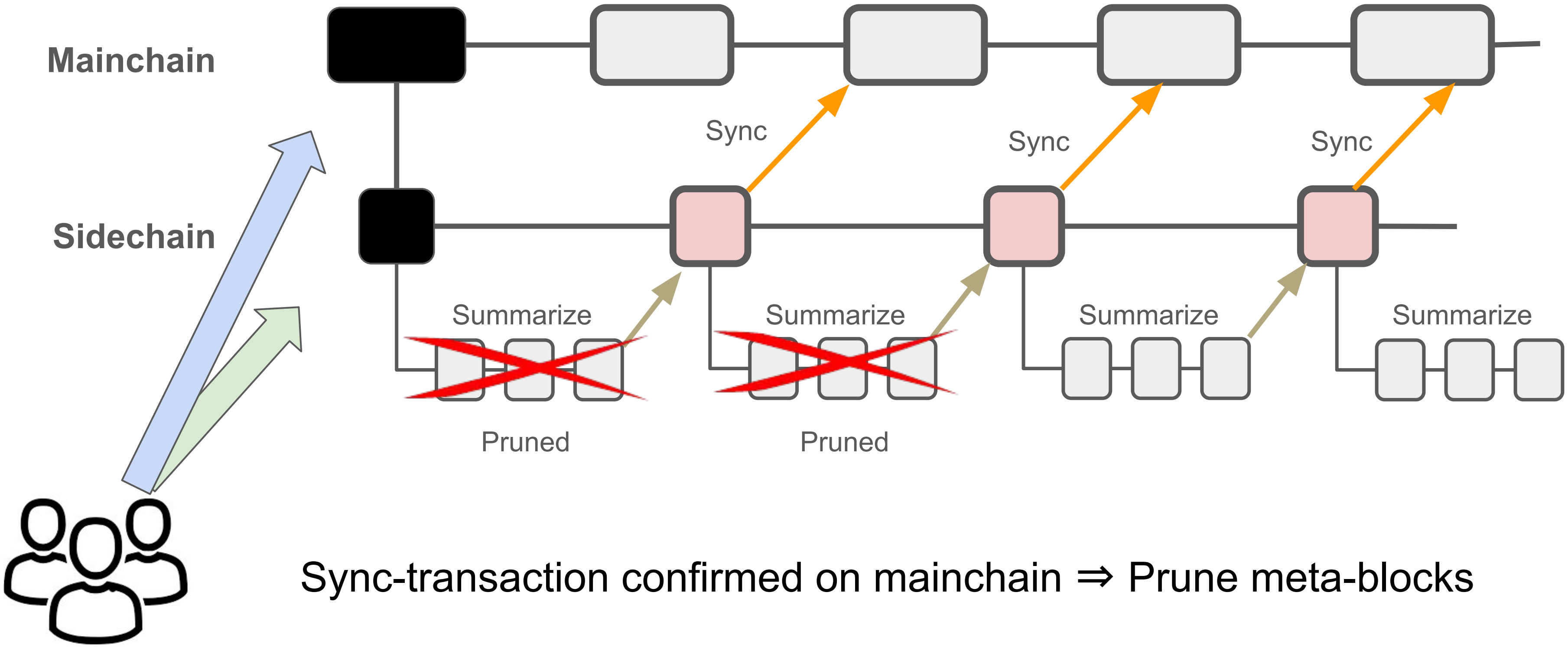
chainBoost Framework



During an epoch: Mine meta-blocks

Epoch end: Summarize + Sync

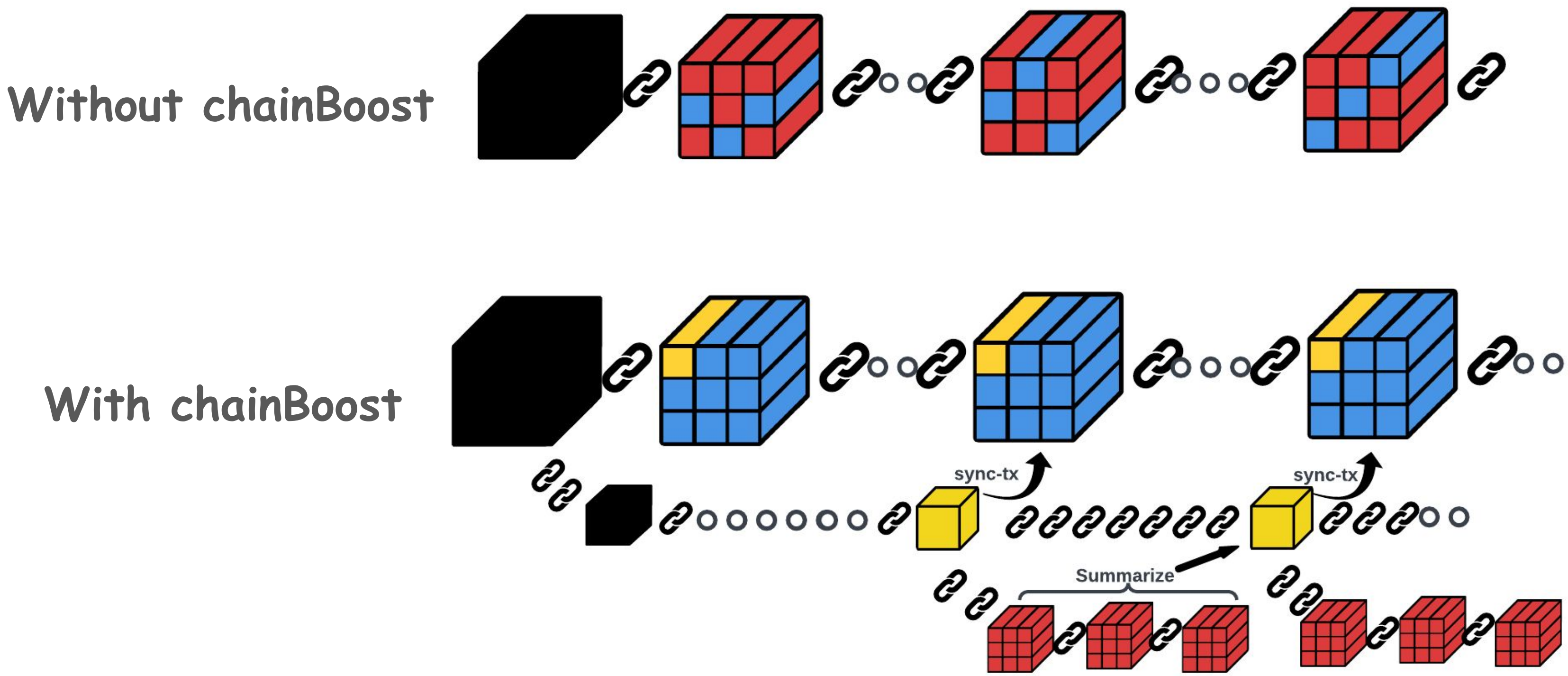
chainBoost Framework



Sync-transaction confirmed on mainchain ⇒ Prune meta-blocks

Performance Boosting

Blockchain size
Throughput
Latency



Service transactions are in red, others are in blue.

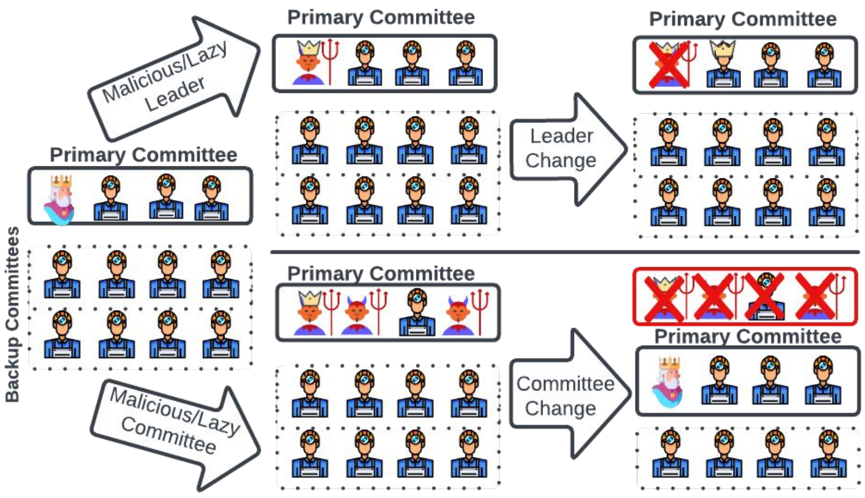
Summary-blocks and sync-transactions are in yellow.

Summary Rules

- Generic summary rules that can be customized based on the service type.
    - Service delivery proofs ⇒ their count per server
    - Market matching ⇒ finalized contracts
    - Disputes ⇒ incident summary + result/penalty

Robustness and Resilience

- Handling (mainchain) rollbacks:
  - Mass-syncing approach.
- Autorecovery protocol:
  - Leader change.
  - Backup committees.



Security and Performance

- **Security:**
  - We prove that chainBoost preserves safety and liveness of the underlying resource market.

- **Performance evaluation:**
  - A Filecoin-inspired use case.
  - Proof-of-concept implementation and extensive experiments.

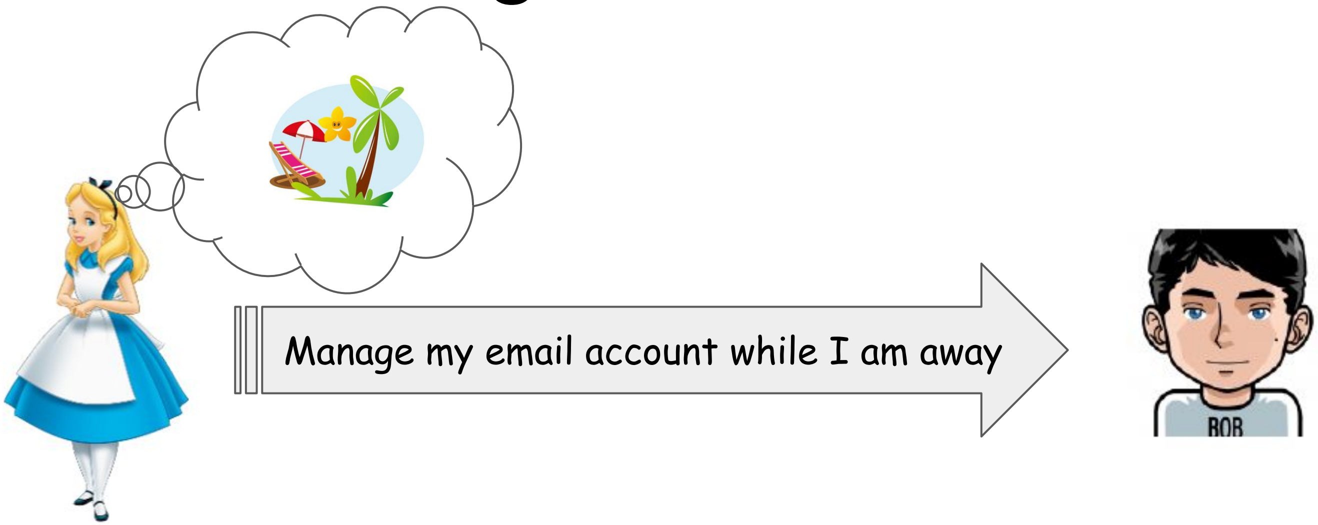
Results

- We report throughput, confirmation time, and blockchain size.
- Studied the impact of various parameters (file storage market with/without chainBoost):
  - **Network load (no. of storage contracts):** 4 - 11x throughput, ~60 - 90% reduction in latency, and up to ~90% blockchain size reduction.
  - **Block size and no. of sidechain rounds per epoch:** larger values are better.
  - **Traffic distribution:** chainBoost has utility for systems that have large workload of service-related transactions.
- Comparison with optimistic rollups:
  - Mainly it is about transaction finality (and the verifier issue).

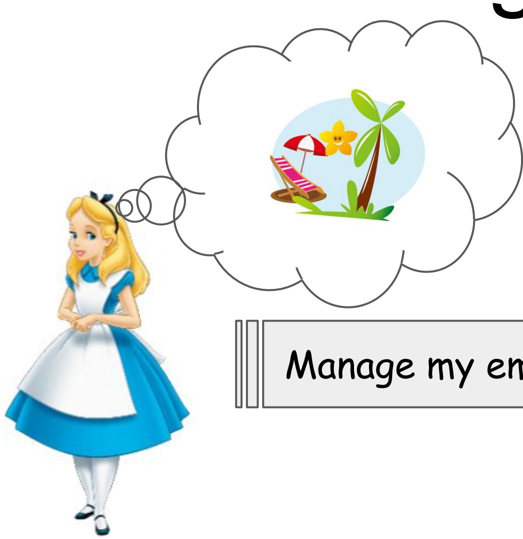
RelaySchnorr: Anonymous, Timed and Revocable Proxy Signatures

*G. Almashaqbeh and A. Nitulescu, *Anonymous, Timed and Revocable Proxy Signatures*, in ISC 2024 (https://eprint.iacr.org/2023/833).

Signature Delegation (Proxy Signatures)



Signature Delegation (Proxy Signatures)



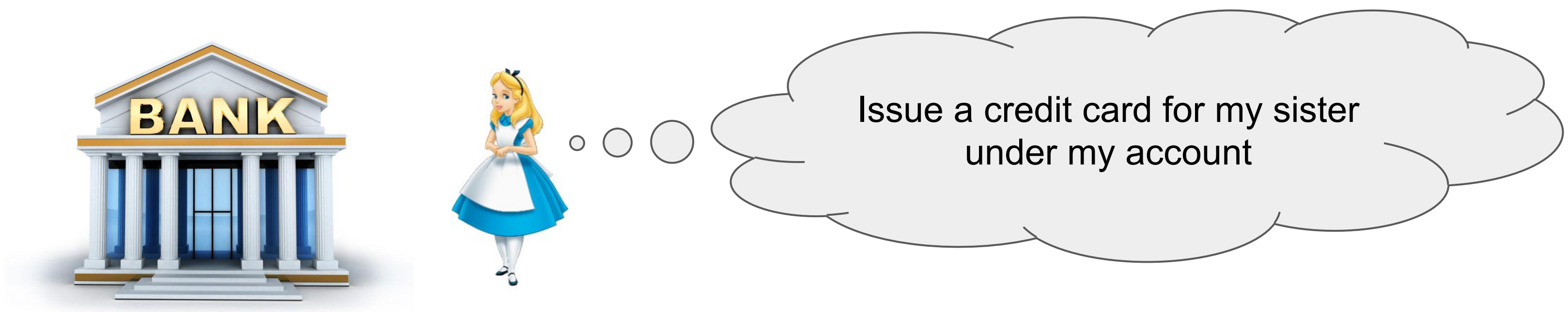
Manage my email account while I am away

Share the workload of handling emails

Produce signed messages on Alice's behalf

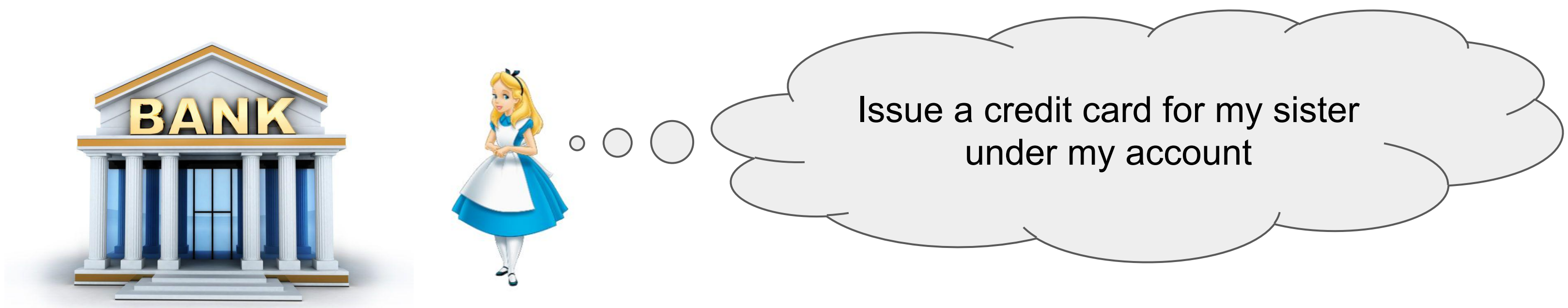
Motivating Applications

Can DeFi (decentralized finance) replace traditional banking services?

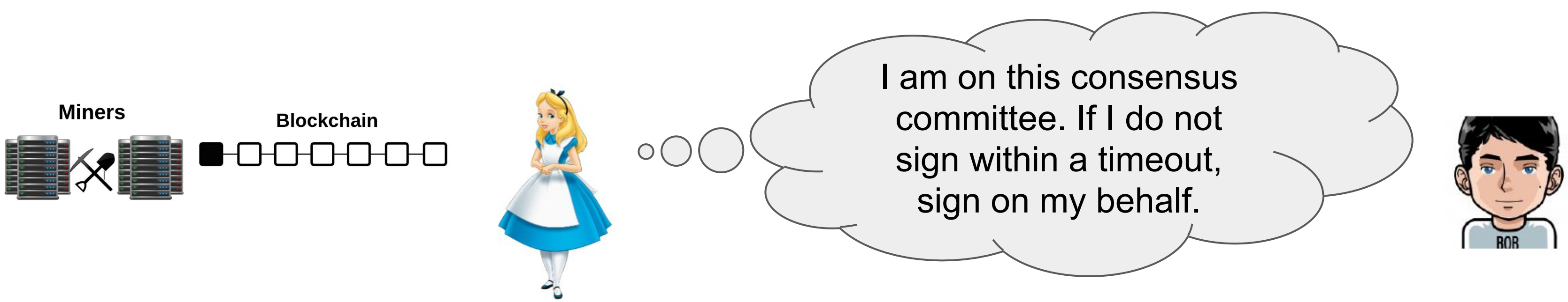


Motivating Applications

Can DeFi (decentralized finance) replace traditional banking services?



Mitigating Targeted Attacks



Desired Delegation Properties

- Anonymity of delegation.
- Timed delegation.
- Revocability.
- Policy enforcement.
- Decentralization.
- Non-interactivity.

Limitations of Prior Work

- No existing scheme achieved all these properties:
  - Many violate anonymity,
  - supported anonymity and policy enforcement without any revocation capability or timed notion,
  - or achieved revocability/timed notion at the expense of being interactive and/or involving a trusted third party.
- No formal security notion of proxy signatures encompassing all these properties.

This work

Can we do Better? … RelaySchnorr

- We define a security notion for anonymous, timed and revocable proxy signatures.
- We show a construction called **RelaySchnorr**
  - Combines Schnorr signatures, timelock encryption, and a public bulletin board.
  - Achieves all the desired properties listed before.
- We formally prove security of our scheme based on our notion.

Building Blocks - Schnorr Signatures

For a security parameter λ, let $\mathbb{G}$ be a cyclic group of a prime order q and a generator G, and $\mathsf{H} : \{0,1\}^* \times \mathbb{G}^2 \to \mathbb{Z}_q$ be a hash function. The Schnorr signature scheme is a tuple of three algorithms $\Sigma_{\mathsf{Schnorr}} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ defined as follows:

- $\mathsf{Schnorr.KeyGen}(1^\lambda)$: On input the security parameter λ, choose uniform $x \in \mathbb{Z}_q$ and compute $X = G^x$. Set the secret signing key $\mathsf{sk} = x$ and the public verification key $\mathsf{vk} = X$.
- $\mathsf{Schnorr.Sign}(\mathsf{sk}, m)$: On input the secret key $\mathsf{sk} = x$ and the message m, choose uniform $k \in \mathbb{Z}_q$. Compute $K = G^k$, $X = G^x$, $c = \mathsf{H}(m, X, K)$, and $s = k + cx \mod q$. Output the signature $\sigma = (c, s)$.
- $\mathsf{Schnorr.Verify}(\mathsf{vk}, m, \sigma)$: On input the public key $\mathsf{vk} = X$, the message m, and signature $\sigma = (c, s)$ over m, compute $K = G^s \cdot X^{-c}$ and $c' = \mathsf{H}(m, X, K)$, then output 1 if $c = c'$.

Building Blocks - TLE

A Timelock encryption scheme $\mathcal{E}$ is a tuple of five PPT algorithms defined as follows:

$\mathsf{TLE.Setup}(1^\lambda) \rightarrow (\mathrm{pp}, s)$: Takes as input the security parameter λ, and outputs public parameters pp and a private key s.

$\mathsf{TLE.RoundBroadcast}(s, \rho) \rightarrow \pi_\rho$: Takes as input the round number ρ and a private key s, and outputs the round-related decryption information π_ρ.

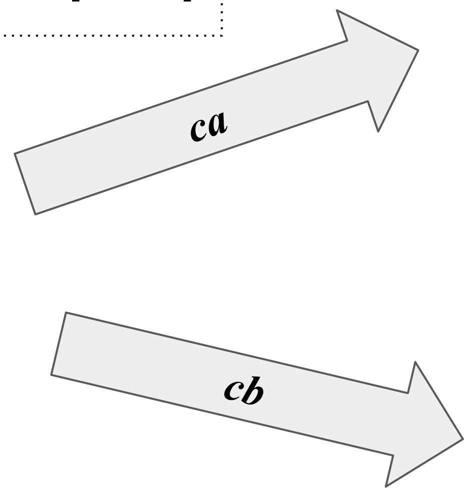
$\mathsf{TLE.Enc}(\rho, m) \rightarrow (\mathrm{ct}_\rho, \tau)$: Takes as input the round number ρ and a message m, and outputs a round-encrypted ciphertext ct_ρ, and trapdoor τ for pre-opening.

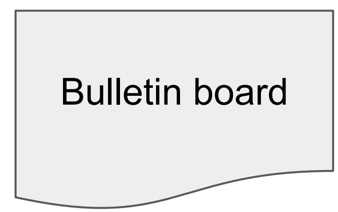
$\mathsf{TLE.Dec}(\pi_\rho, \mathrm{ct}_\rho) \rightarrow m'$: Takes as input the round-related decryption information π_ρ and a ciphertext ct_ρ, and outputs a message m'.

$\mathsf{TLE.PreOpen}(\mathrm{ct}_\rho, \tau) \rightarrow m'$: Takes as input a ciphertext ct_ρ and a trapdoor τ, and outputs a message m'.

RelaySchnorr Construction

Delegation period [Ta, Tb]

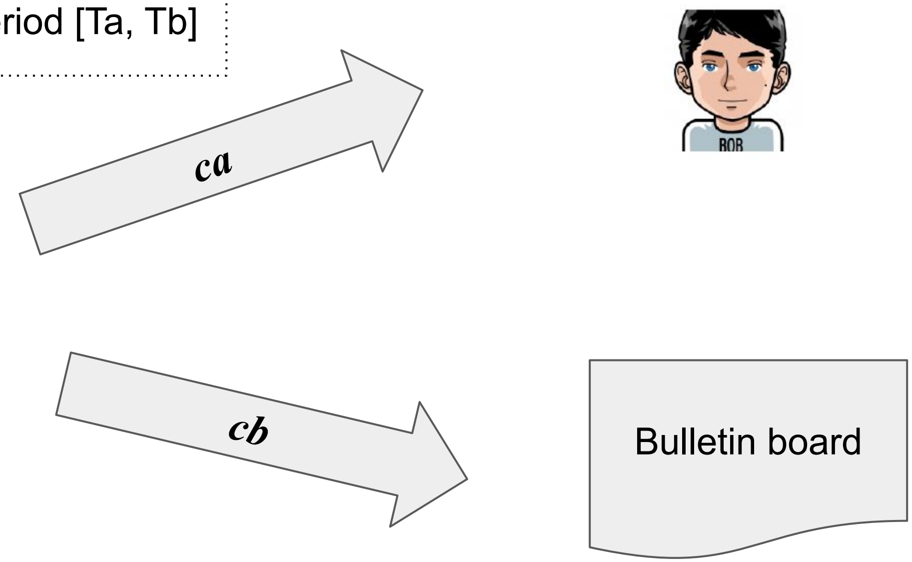




(1) Generate ***u*** random elements ***k1, …, ku***
(2) Generate ***u*** tokens: ***t1, …, tu*** (each token is a Schnorr signature over ***ki***)
(3) Encrypt the tokens to time ***Ta***, and the ***k*** values to time ***Tb***

RelaySchnorr Construction

Delegation period [Ta, Tb]



(1) Generate ***u*** random elements ***k1, …, ku***
(2) Generate ***u*** tokens: ***t1, …, tu*** (each token is a Schnorr signature over ***ki***)
(3) Encrypt the tokens to time ***Ta***, and the ***k*** values to time ***Tb***

At time ***Ta***:
(1) Decrypt the tokens.
(2) Use a token to sign message ***m*** (produce another Schnorr signature using the token).

RelaySchnorr Construction

Delegation period [Ta, Tb]

ca

(1) Generate u random elements $k1, \ldots, ku$
(2) Generate u tokens: $t1, \ldots, tu$ (each token is a Schnorr signature over ki)
(3) Encrypt the tokens to time Ta, and the k values to time Tb

cb

Bulletin board

ki

At time Ta:
(1) Decrypt the tokens.
(2) Use a token to sign message m (produce another Schnorr signature using the token).

Verify a signature

RelaySchnorr Construction

One-time tokenizable Schnorr

Delegation period [Ta, Tb]

ca

(1) Generate ***u*** random elements ***k1, …, ku***
(2) Generate ***u*** tokens: ***t1, …, tu*** (each token is a Schnorr signature over ***ki***)
(3) Encrypt the tokens to time ***Ta***, and the ***k*** values to time ***Tb***

cb

Bulletin board

ki

At time ***Ta***:
(1) Decrypt the tokens.
(2) Use a token to sign message ***m*** (produce another Schnorr signature using the token).

Verify a signature

RelaySchnorr Construction

Delegation period [Ta, Tb]

Board validators

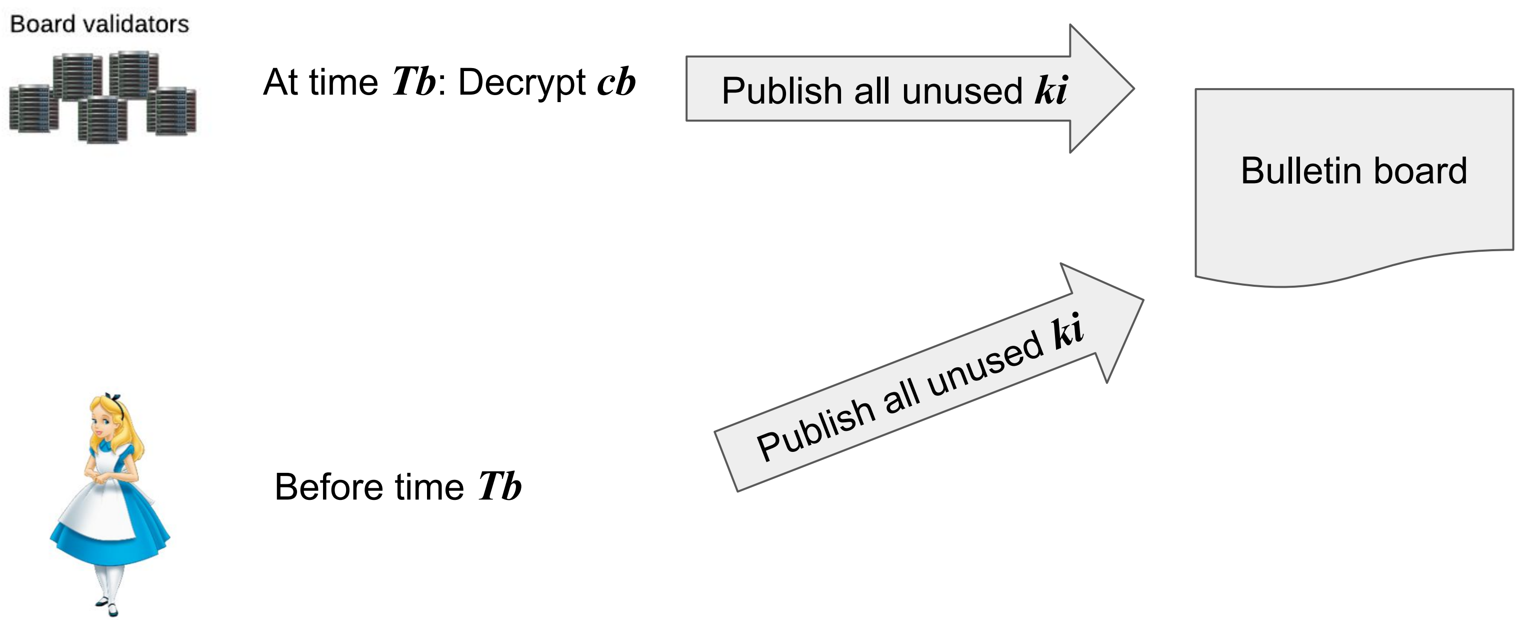
At time ***Tb***: Decrypt ***cb***

Publish all unused ***ki*** →

Bulletin board

RelaySchnorr Construction

Delegation period [Ta, Tb]



RelaySchnorr Construction

Delegation period [Ta, Tb]

Board validators

At time ***Tb***: Decrypt ***cb***

Publish all unused ***ki***

Bulletin board

Before time ***Tb***

Publish all unused ***ki***

Timed delegation

Automatic and on-demand revocation

Anonymity and Policy Enforcement

- **Anonymity is achieved by:**
  - Proxy signer identity is not included.
  - Delegation info is sent privately to the proxy signer.
  - The signature structure is the same for both the original or proxy signer, and verified using the same Verify algorithm.
  - Original signer mimics the behavior of having a delegation for her signatures.
- **Policy enforcement over messages:**
  - Conventional methods from the literature: public warrants and private ones (using NIZKs).

Issues in Practice

- Denial of service attacks against the signer.
- Bulletin board synchronization.
- Off-chain processing issues.
- Information lookup cost.
- Mass production of k values and delegation anonymity.

Security

Theorem 1. *Assuming EUF-CMA security of Schnorr signatures, the schnorr-koe assumption, a secure bulletin board, a CCA-secure TLE scheme, an EUF-CMA secure signature scheme, and a secure NIZK proof system, RelaySchnorr is an anonymous, timed and revocable proxy signature scheme (cf. Definition 2).*

- Unforgeability relies on the unforgeability of Schnorr signatures in the random oracle model, and the Schnorr knowledge of exponent assumption.
- Anonymity is achieved by having identical signature structure and behavior.
- Revocability relies on the security of timelock encryption and the bulletin board.
- Policy enforcement relies on the security of digital signatures (for public warrants) or NIZKs (for private policies), as well as security of timelock encryption and the bulletin board.

Last Stop!

Conclusion and Future Work

- The ‘give and take’ is an evolving relationship!
- Future work directions:
  - Adapt chainBoost for other blockchain system types, e.g. applications on top of Ethereum.
    - ammBoost for automated market makers.
  - Storage pricing/transaction fees in this multi-layer temporary/permanent storage.
  - Collateral and wallet management.
  - Explore delegation for other cryptographic primitives.
    - Zero knowledge proofs (aka delegation of private wallets).
    - Password-authenticated delegation.

Thank you!

Questions?

ghada@uconn.edu
https://ghadaalmashaqbeh.github.io/

Implementation

- Sidechain:
  - Implemented our architecture in Go.
  - A collective signature (CoSi)-based PBFT (the BLSCoSi one from Cothority).
  - Onet for communication between miners
  - The sliding window approach from Byzcoin for committee election.
- Underlying storage market:
  - Mimic Filecoin but with compact proof-of-retrievability as proof-of-storage.
  - Traffic generation follows the traffic distribution of Filecoin.
  - Mining power on the mainchain depends on the amount of service the miners (aka storage servers) provide.
- To compare with another layer-two solution, we implemented optimistic rollups (inspired by Optimism).

Concrete Construction

Setup phase

Let λ be a security parameter, S be the original signer, P be the proxy signer, and TLE be a timelock encryption scheme. Construct an anonymous, timed and revocable proxy signature scheme $\Sigma = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Delegate}, \mathsf{DegSign}, \mathsf{Revoke}, \mathsf{Verify})$ as follows:

$\mathsf{Setup}(1^\lambda)$: On input the security parameter λ, set $\mathbb{G}$ to be a cyclic group of a prime order q with a generator $G \in \mathbb{G}$ and $\mathsf{H} : \{0,1\}^* \times \mathbb{G}^2 \to \mathbb{Z}_q$ to be a hash function, initialize $\mathsf{state} = \{\}$, and invoke $\mathsf{TLE.Setup}(1^\lambda)$. Output $\mathsf{pp} = (\mathsf{TLE.pp}, \mathsf{H}, \mathbb{G}, G, q, \mathsf{state})$.

$\mathsf{KeyGen}(1^\lambda)$: On input the security parameter λ, choose uniform $x \in \mathbb{Z}_q$, then compute $X = G^x$. Output the signing key $\mathsf{sk} = x$ and the verification key $\mathsf{vk} = X$.

Concrete Construction

Sign—used by original signer S

Sign(sk, m): On input the signing key sk $= x$ and some message m, do:

- Choose uniform $k, r, e \in \mathbb{Z}_q$, compute $R = G^r$, $E = G^e$
- Compute $w = \mathsf{H}(k, X, R)$, $z = (r + wx) \mod q$, and $Z = G^z$
- Compute $c = \mathsf{H}(m, Z, E)$ and $s = (e + cz) \mod q$ (if $z = 0$ or $s = 0$ start again with fresh r and e)
- Output the signature $\sigma = (w, c, s, k, Z)$

Every now and then, S either (1) populates a set klist from the stored k values and fresh ones, encrypts it as $(\mathrm{ct}_b, \tau_b) = \mathsf{TLE.Enc}(\mathrm{klist}, \rho_b)$, where ρ_b is some future round number, and posts (ρ_b, ct_b) on the board (resulting in $\mathrm{state}'[\mathrm{vk}] = \mathrm{state}[\mathrm{vk}] \parallel (\rho_b, \mathrm{ct}_b)$), or (2) posts a fresh klist on the board (resulting in $\mathrm{state}'[\mathrm{vk}] = \mathrm{state}[\mathrm{vk}] \parallel \mathrm{klist}$).

Concrete Construction

Delegate—invoked by original signer S

Delegate(sk, vk, degspec): On input the keypair $(\mathsf{sk} = x, \mathsf{vk} = X)$ and delegation specifications $\mathsf{degspec} = (u, [\rho_a, \rho_b])$, where $u \in \mathbb{N}$ and $[\rho_a, \rho_b]$ is the delegation period, do the following:

- Set $\mathsf{klist} = \{\}$
- Do the following for $i \in \{1, \ldots, u\}$:
  - Choose uniform $k_i, r_i \in \mathbb{Z}_q$
  - Compute $R_i = G^{r_i}$ and $w_i = \mathsf{H}(k_i, X, R_i)$
  - Compute $z_i = (r_i + w_i x) \mod q$ (if $z_i = 0$ start again with fresh r_i)
  - Set $t_i = (z_i, w_i, k_i)$ and $\mathsf{klist} = \mathsf{klist} \cup \{k_i\}$
- Compute two ciphertexts: $(\mathsf{ct}_a, \tau_a) = \mathsf{TLE.Enc}(t_1 \parallel \cdots \parallel t_u, \rho_a)$ and $(\mathsf{ct}_b, \tau_b) = \mathsf{TLE.Enc}(\mathsf{klist}, \rho_b)$ (where τ_b is the revocation key rk).
- Set $\mathsf{degInfo} = (\rho_a, \rho_b, \mathsf{ct}_a)$
- Output $(\mathsf{degInfo}, \mathsf{ct}_b \parallel \tau_b)$

S stores ciphertext ct_b and trapdoor τ_b to be used for revocation if needed (τ_a is dropped as it is not needed), posts (ρ_b, ct_b) on the board (resulting in $\mathsf{state}'[\mathsf{vk}] = \mathsf{state}[\mathsf{vk}] \parallel (\rho_b, \mathsf{ct}_b)$), and sends degInfo to P.

Concrete Construction

Delegate Sign—used by proxy signer P

DegSign(m, degInfo): On input a message m and delegation information degInfo, P does the following (let ρ_{now} = state.round be the current round number):

- If $\rho_{now} < \rho_a$ or $\rho_{now} > \rho_b$, then do nothing
- If $\rho_a \leq \rho_{now} \leq \rho_b$, then:
  - If degInfo $= (\rho_a, \rho_b, \mathsf{ct}_a)$, then retrieve π_{ρ_a} from the board (π_{ρ_a} = state.roundInfo(ρ_a)) and set degInfo $= (\rho_a, \rho_b, \mathsf{TLE.Dec}(\pi_{\rho_a}, \mathsf{ct}_a))$
  - Pick an unused signing token $t = (z, w, k)$ from degInfo
  - Compute $Z = G^z$
  - Choose uniform $e \in \mathbb{Z}_q$ and compute $E = G^e$
  - Compute $c = \mathsf{H}(m, Z, E)$, and $s = e + cz \mod q$ (if $s = 0$ start again with a fresh e)
  - Output the signature $\sigma = (w, c, s, k, Z)$

Concrete Construction

Automatic/On demand revoke—invoked by validators or original signer S
Verify—Invoked by a verifier for any signature

Revoke(degInfo, rk, state[vk]): On input degInfo $= (\rho_b, \mathsf{ct}_b)$, revocation key rk, and revocation state state[vk], do (let $\rho_{now} = \mathsf{state.round}$ be the current round number):

- If $\rho_{now} \geq \rho_b$, then retrieve π_{ρ_b} from the board ($\pi_{\rho_b} = \mathsf{state.roundInfo}(\rho_b)$) and compute klist $= \mathsf{TLE.Dec}(\pi_{\rho_b}, \mathsf{ct}_b)$
- If $\rho_{now} < \rho_b$, then use $\mathsf{rk} = \tau_b$ to compute klist $= \mathsf{TLE.PreOpen}(\mathsf{ct}_b, \tau_b)$
- Add all k values such that $k \in \mathsf{klist} \wedge k \notin \mathsf{state[vk]}$ to the board state state[vk] associated with vk resulting in an updated state state[vk]$'$.

Verify(vk, $m, \sigma = (w, c, s, k, Z)$, revState = state[vk]): On input the verification key vk $= X$, the message m, signature $\sigma = (w, c, s, k, Z)$ over m, and the revocation state state[vk], if $k \in$ state[vk], then output 0. Else, add k to state[vk] (resulting in state$'$[vk] = state[vk] $\|$ k) and do the following:

- Compute $R = Z \cdot X^{-w}$ and $E = G^s \cdot Z^{-c}$
- Output 1 if and only if $w = \mathsf{H}(k, X, R) \wedge c = \mathsf{H}(m, Z, E)$.