Collective Edwards-Curve Digital Signature Algorithm

draft-ford-cfrg-cosi-00

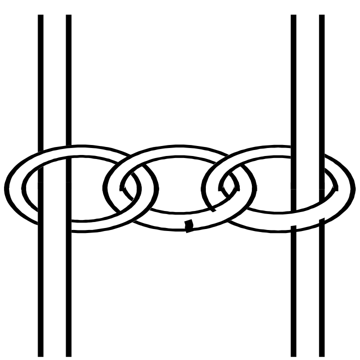
Nicolas Gailly, Phillip Jovanovic,
Linus Gasser, Bryan Ford
Decentralized/Distributed Systems (DEDIS)

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

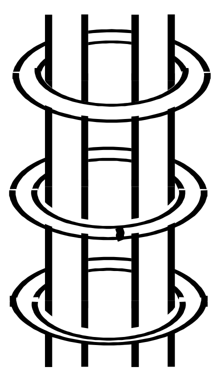
IETF 99 – Prague – July 18, 2017

Goal: Efficient, Scalable Trust Splitting

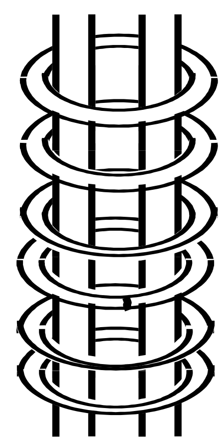
Weakest-link security:
T = 1



Strongest-link security:
T = 2-10



Collective security:
T = 100s,1000s



A Basic Tool: Collective Signing

Multiple independent parties collaborate to validate and sign a single message or statement

- Often a threshold of a predefined group (t-of-n)
- Ensure transparency (many witnesses)
- Eliminate single points of failure

Schnorr signatures easy & efficient to aggregate

- Many signatures compressed into space of ~1
- Verifier incurs CPU cost of only ~1 verification

Basic Schnorr Signature (e.g., Ed25519)

- Generator *g* of prime order *q* group
- Public/private key pair: $(K=g^k, k)$

	Signer		Challenger
Commitment	$V=g^v$	⟶	V
Challenge	c	⟵	$c = H(M\|V)$
Response	$r = (v - kc)$	⟶	r

Signature on M: (c, r)

Commitment recovery	$V' = g^r K^c \quad = g^{v-kc} g^{kc} = g^v = V$
Challenge recovery	$c' = H(M\|V')$
Decision	$c' = c$? ✓

Note: Ed25519/Ed448 use slightly different but equivalent (R,S) signature format

Schnorr Multisignature

- Generator g of prime order q group
- Public/private key pair: $(K=g^k, k)$

	Signer 1	Signer 2		Challenger
Commitment	$V_1=g^{v1}$	$V_2=g^{v2}$	→	$V = V_1 * V_2$
Challenge			←	$c = H(M\|V)$
Response	$r_1 = (v_1 - k_1 c)$	$r_2 = (v_2 - k_2 c)$	→	$r = r1 + r2$

Classic Schnorr multisignature on M: (c, r)

Commitment recovery	$V' = g^r K^c \quad = g^{v-kc} g^{kc} = g^v = V$
Challenge recovery	$c' = H(M\|V')$
Decision	$c' = c$? ✓

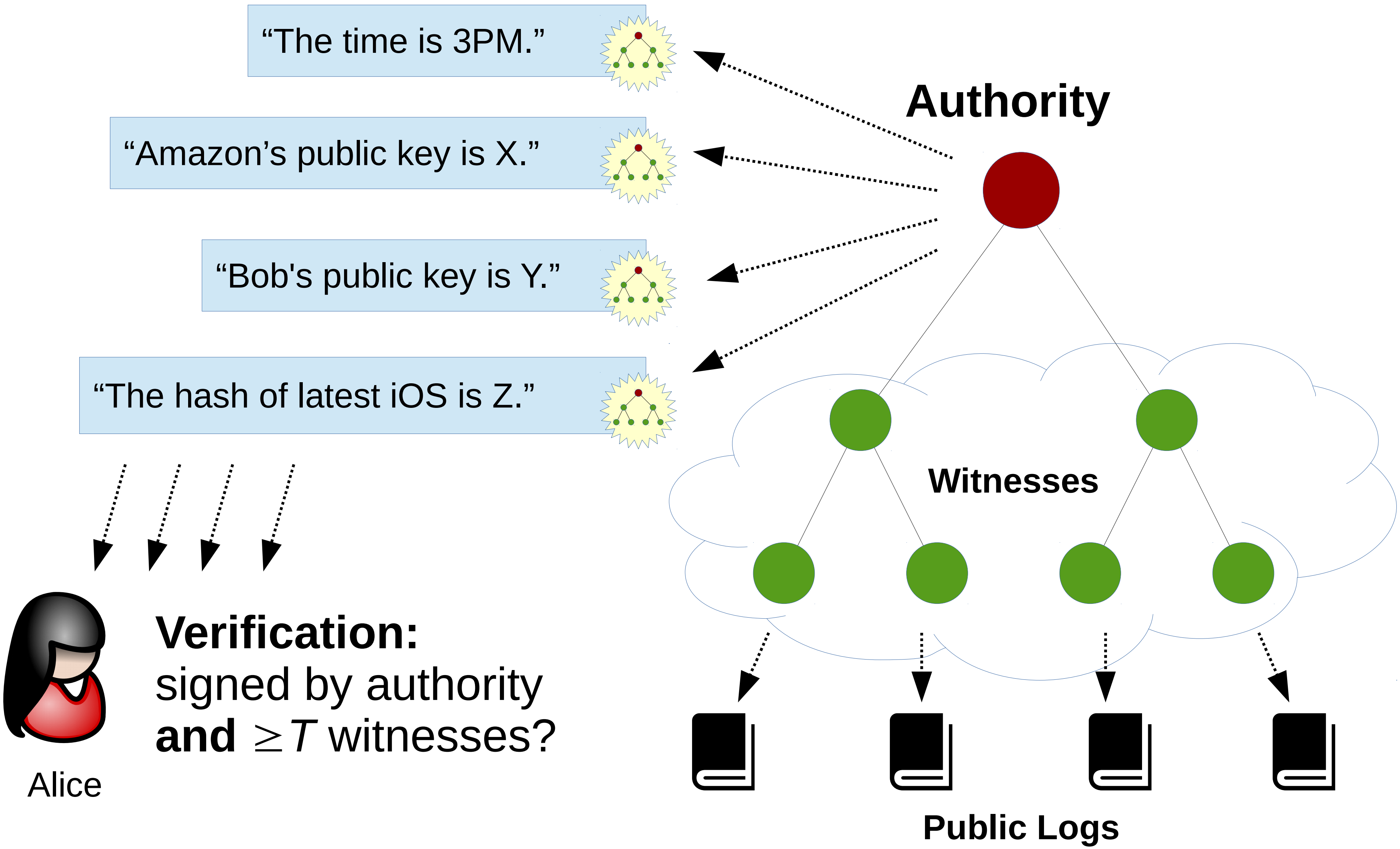
Note: draft-ford-cfrg-cosi-00 uses (R,S) format for consistency with Ed25519/Ed448 specs

Formal Background, Analysis, Proofs

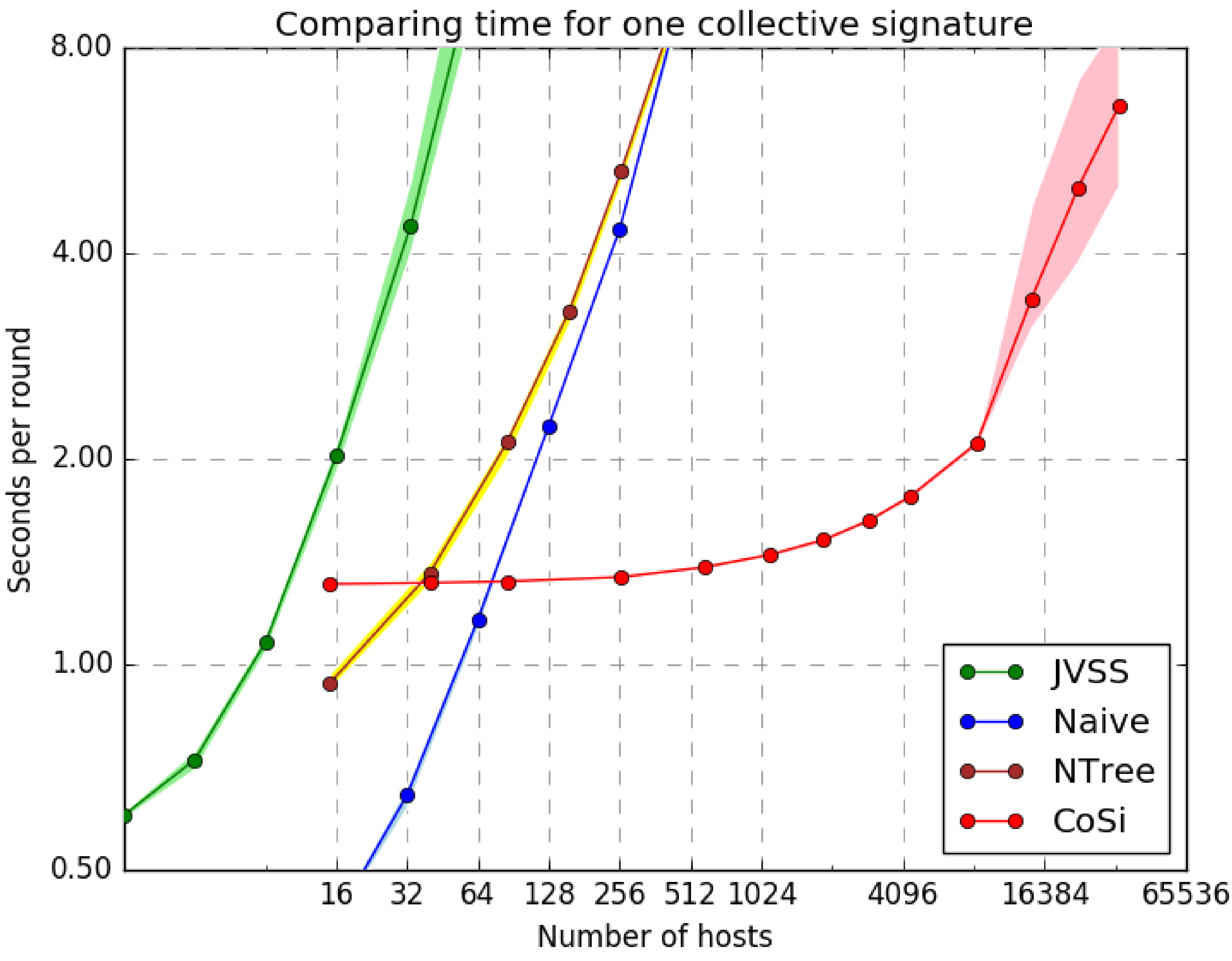
Based on well-established body of formal work

- Horster et al, "Meta-Multisignature schemes based on the discrete logarithm problem"
- Michels et al, "On the risk of disruption in several multiparty signature schemes"
- Ohta/Okamoto, "Multi-Signature Schemes Secure against Active Insider Attacks"
- Micali et al, "Accountable-Subgroup Multisignatures"
- Bellare/Neven, "Multi-signatures in the plain public-key model and a general forking lemma" (delinearization approach)
- …

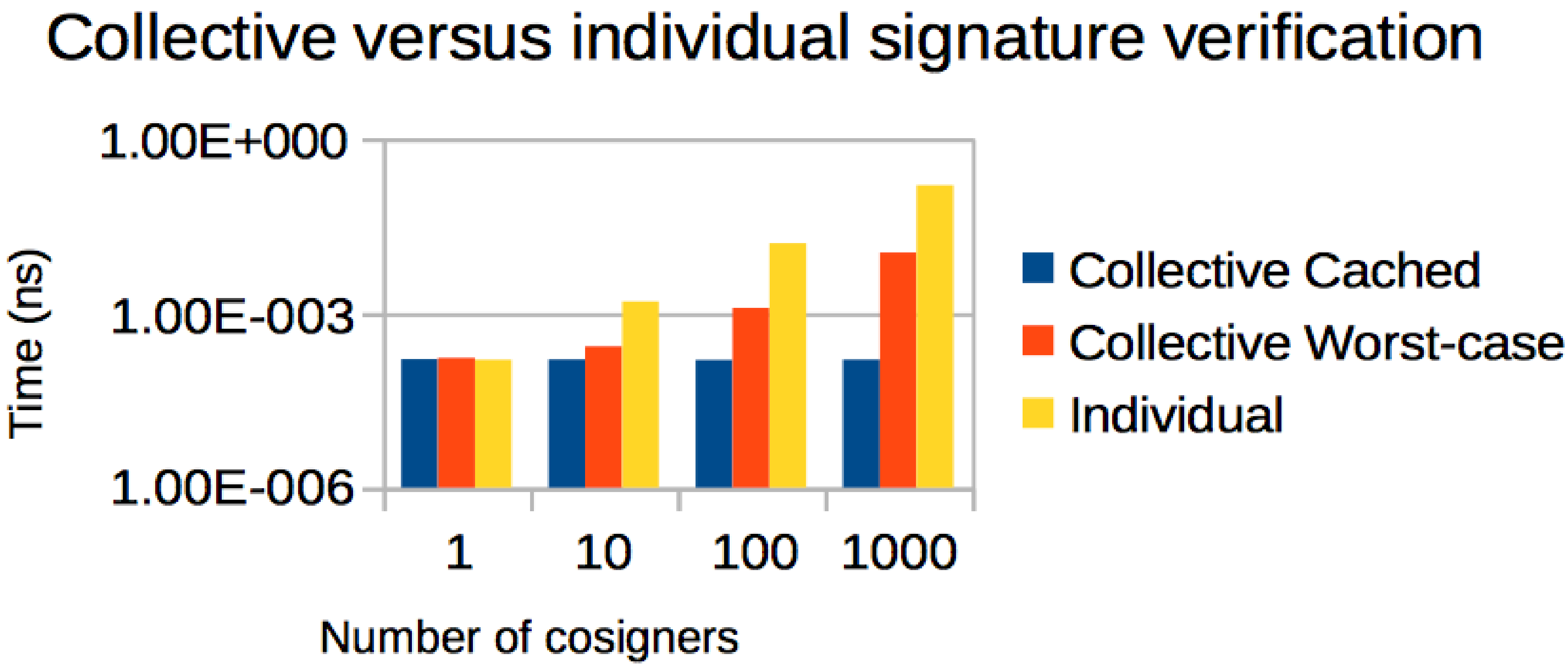
Use-Case: Scalable Collective Signing [Syta et al, IEEE Security&Privacy '16]



Results: Collective Signing Time

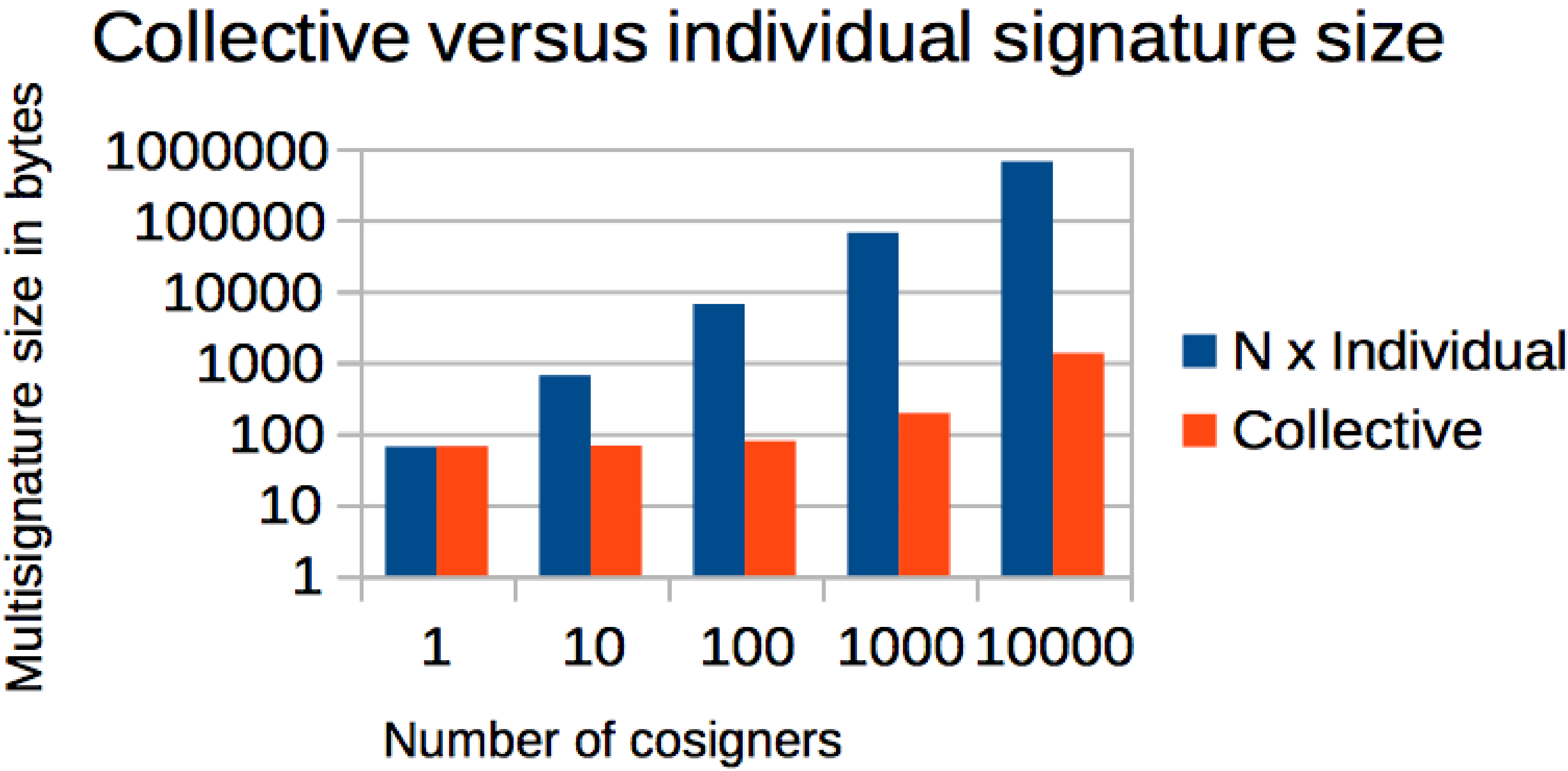


Results: Verification Cost



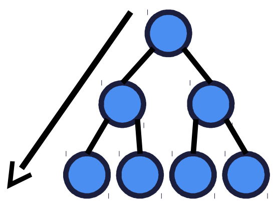
Results: Collective Signature Size

Ed25519: up to 512x smaller than N signatures

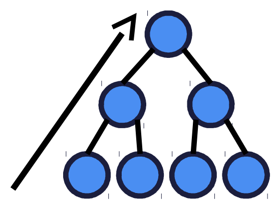


Optional: Higher Scalability via Trees

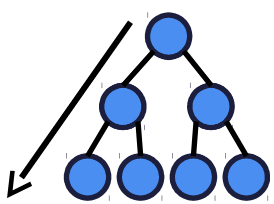
1. Announcement Phase



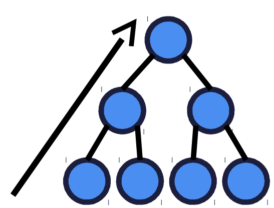
2. Commitment Phase



3. Challenge Phase



4. Response Phase



Use-Case: Bitcoin Transactions

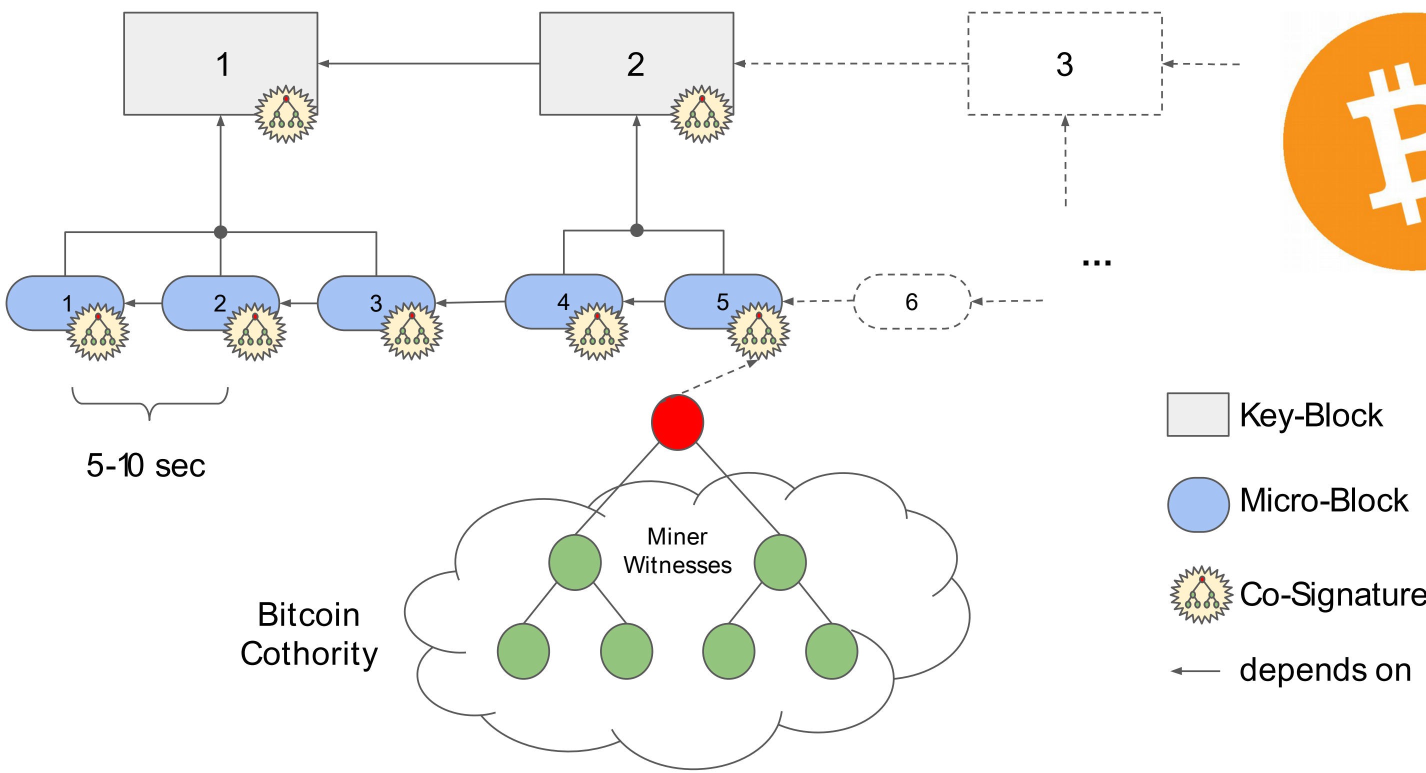
"Schnorr Multisignatures for Bitcoin" [Wuille]

- Compress many signatures on a transaction (or many transactions) into space of one
- Yield ~25% transaction space savings

Use-Case: Fast, Scalable Blockchains

ByzCoin blockchain presented in [USENIX Security '16]

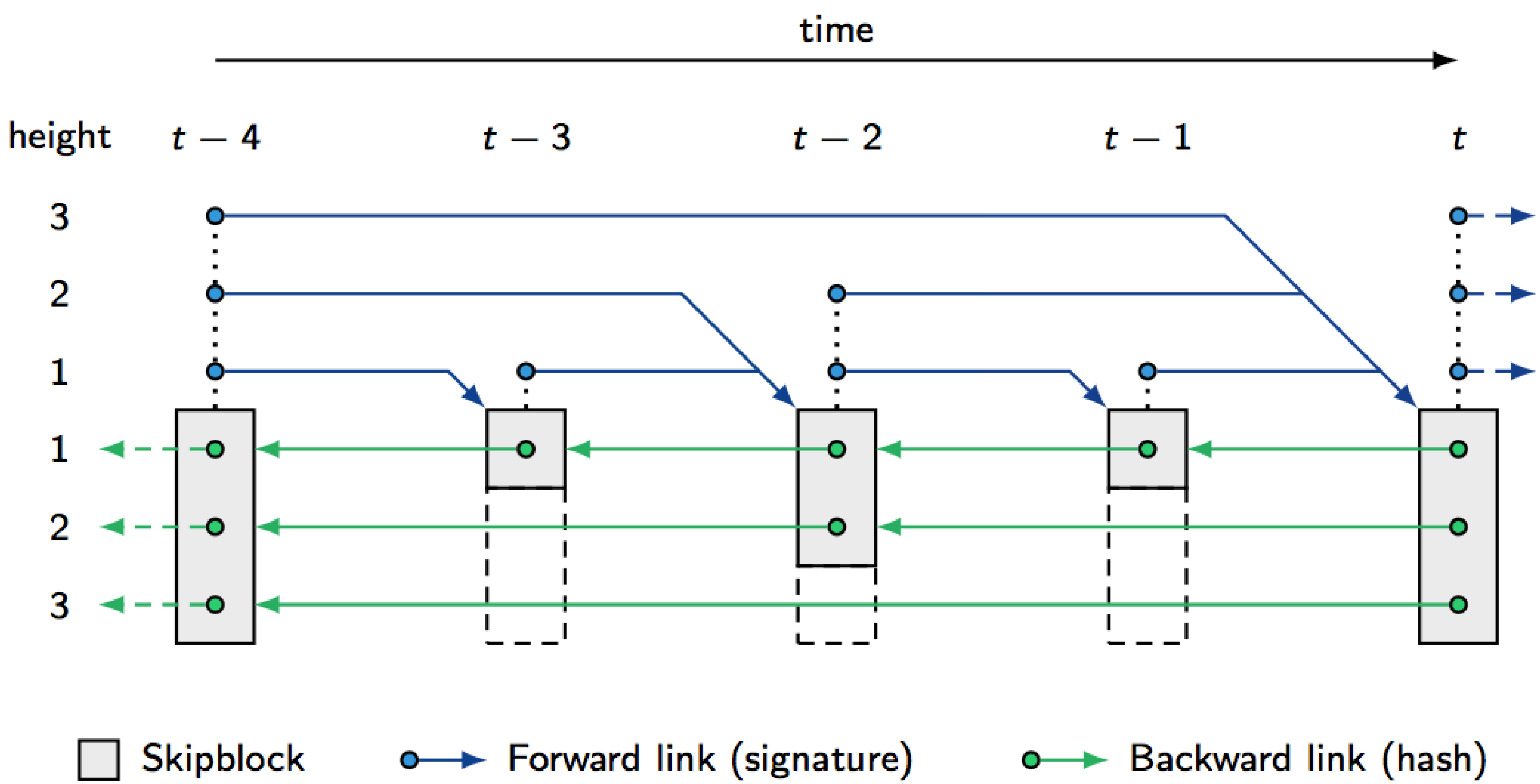
- **Permanent** transaction commitment in seconds
- **700+ TPS** demonstrated (100x Bitcoin, ~PayPal)
- **Low-power** verification on light mobile devices



Use-Case: Offline-Verifiable Histories

SkipChains: part of Chainiac [USENIX Sec '17]

- Efficiently prove existence+correctness of a transaction anywhere *forward* or *back* in time



draft-ford-cfrg-cosi-00

Starting-point draft (only)

- Intended to be consistent with & extension to Ed25519/Ed448 specified by RFC 8032

Structure summary:

- Basic signing/verification algorithms
- Simple 4-phase protocol for collective signing
- Optional more scalable tree-based protocol
- Message formats (currently protobuf-based)
- Security considerations, issues for discussion

Issues to be Discussed (if Adopted)

Many design/implementation tradeoffs, e.g.:

- Strict Non-Malleability versus DoS Resistance
  - Include participant mask in hash for non-malleability
  - Precludes defense against O(N)-time restart attacks
- Best defense(s) against related-key attacks
  - At group formation via self-signed keys (e.g., PGP)
  - In signing/verification via delinearization (Bitcoin)
- Minimizing verifier state (Merkle pubkey trees)
- Probably many more…