Collective Edwards-Curve Digital Signature Algorithm

draft-ford-cfrg-cosi-00

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

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: $(K=g^k, k)$

<table>
<thead>
<tr>
<th>Signer</th>
<th>Challenger</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commitment</td>
<td>$V=g^v$</td>
</tr>
<tr>
<td>Challenge</td>
<td>$c$</td>
</tr>
<tr>
<td>Response</td>
<td>$r = (v - kc)$</td>
</tr>
</tbody>
</table>

Signature on $M$: $(c, r)$

Commitment recovery: $V' = g^r K^c = g^{v-kc} g^{kc} = g^v = V$

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: $(K=g^k, k)$

<table>
<thead>
<tr>
<th></th>
<th>Signer 1</th>
<th>Signer 2</th>
<th>Challenger</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commitment</td>
<td>$V_1=g^{v_1}$</td>
<td>$V_2=g^{v_2}$</td>
<td>$V = V_1 * V_2$</td>
</tr>
<tr>
<td>Challenge</td>
<td>$r_1 = (v_1 - k_1c)$</td>
<td></td>
<td>$c = H(M</td>
</tr>
<tr>
<td>Response</td>
<td>$r_2 = (v_2 - k_2c)$</td>
<td></td>
<td>$r = r_1 + r_2$</td>
</tr>
</tbody>
</table>

Classic Schnorr multisignature on $M$: $(c, r)$

- Commitment recovery
  
  $V' = g^rK^c = g^{v-kc}g^{kc} = g^v = V$

- 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]

“Amazon’s public key is X.”
“Bob's public key is Y.”
“The hash of latest iOS is Z.”

Veriﬁcation: signed by authority and ≥ T witnesses?
Results: Collective Signing Time

Comparing time for one collective signature

Seconds per round vs. Number of hosts

- JVSS
- Naive
- NTree
- CoSi
Results: Verification Cost

Collective versus individual signature verification

Time (ns)

1.00E+000
1.00E-003
1.00E-006

Number of cosigners

Collective Cached
Collective Worst-case
Individual
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

![Diagram of ByzCoin blockchain](attachment:image.png)
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
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…