Adapting Hierarchical Key Derivation for Ephemeral Signatures in MLS?

Nadim Kobeissi
INRIA Paris, NYU Paris

IETF 103 MLS Hackathon
November 5, 2018
HDK: General Idea

- $B$ is a base point.
- $k$ is a secret key.
- $[k]B$ is a public key.
- $x$ is a scalar.

$k + x = \text{new private key.}$

$[k]B + [x]B = \text{new public key.}$

$[k+x]$ corresponds to $[k+x]B!$
HDKs are already used in Bitcoin...

**BIP 32 - Hierarchical Deterministic Wallets**

**Child Key Derivation Function**

\[ \text{CKD}(x, n) = \text{HMAC-SHA512}(x_{\text{Chain}}, x_{\text{PubKey}} \parallel n) \]
But Ed25519 is not just scalar multiplication...

- Unlike secp256k1, Ed25519 does a bunch of hashing.
- A bunch of “bit clearing”, “clamping”,

Khovratovich and Law show ways around that in their paper:

**BIP32-Ed25519: Hierarchical Deterministic Keys over a Non-linear Keyspace**

*Dmitry Khovratovich, Jason Law*
HDK Trees (simplified)

(k, x) ← HKDF(w, sid)

Private key: k + Z_L
Public key: [k]B + [Z_L]B

Root key
Child key
Hardened child key
Potential applications to MLS

• Currently in MLS, there is one signature key (identity key) per user for all of their conversations, always.

• HDK allows us to compartmentalize signature keys per conversation/epoch etc. without additional key exchange.

• Improvements are clear for partial state compromise.

• But what are the improvements in the case of full state compromise?
Signal Desktop key management
WhatsApp Desktop key management

All the keys

Symmetric link

No keys
MLS Desktop key management

All the keys

HDK roots for active conversations only
To what demarcation of state compromise can we generalize these improvements?