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.

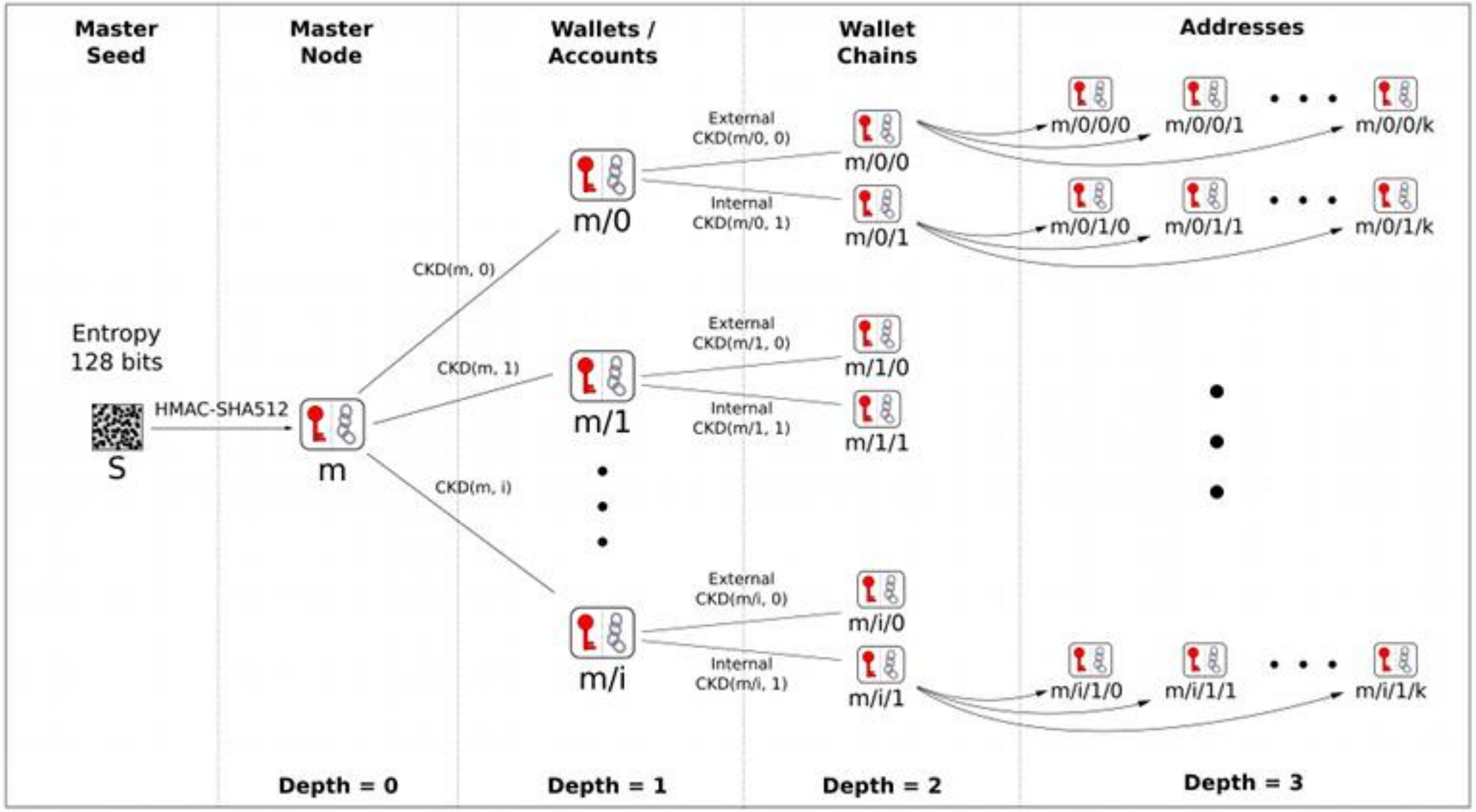
$k + x$ = new private key.

$[k]B + [x]B$ = 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 ~ CKD(x,n) = HMAC-SHA512(x_{Chain} , x_{PubKey} || n)

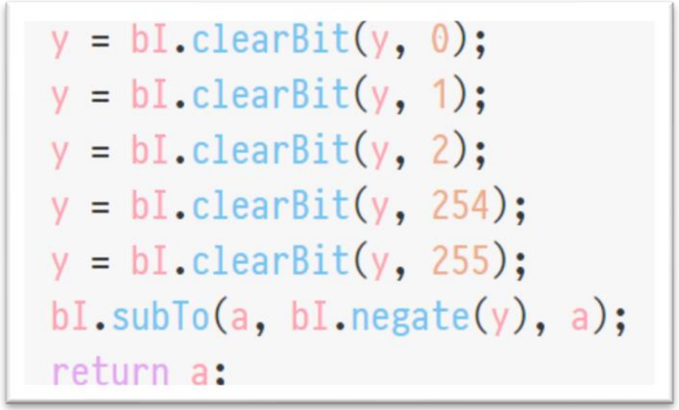
But Ed25519 is not just scalar multiplication…

- 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



```
y = bI.clearBit(y, 0);
y = bI.clearBit(y, 1);
y = bI.clearBit(y, 2);
y = bI.clearBit(y, 254);
y = bI.clearBit(y, 255);
bI.subTo(a, bI.negate(y), a);
return a;
```

HDK Trees (simplified)

`(k, x) ← HKDF(w, sid)`

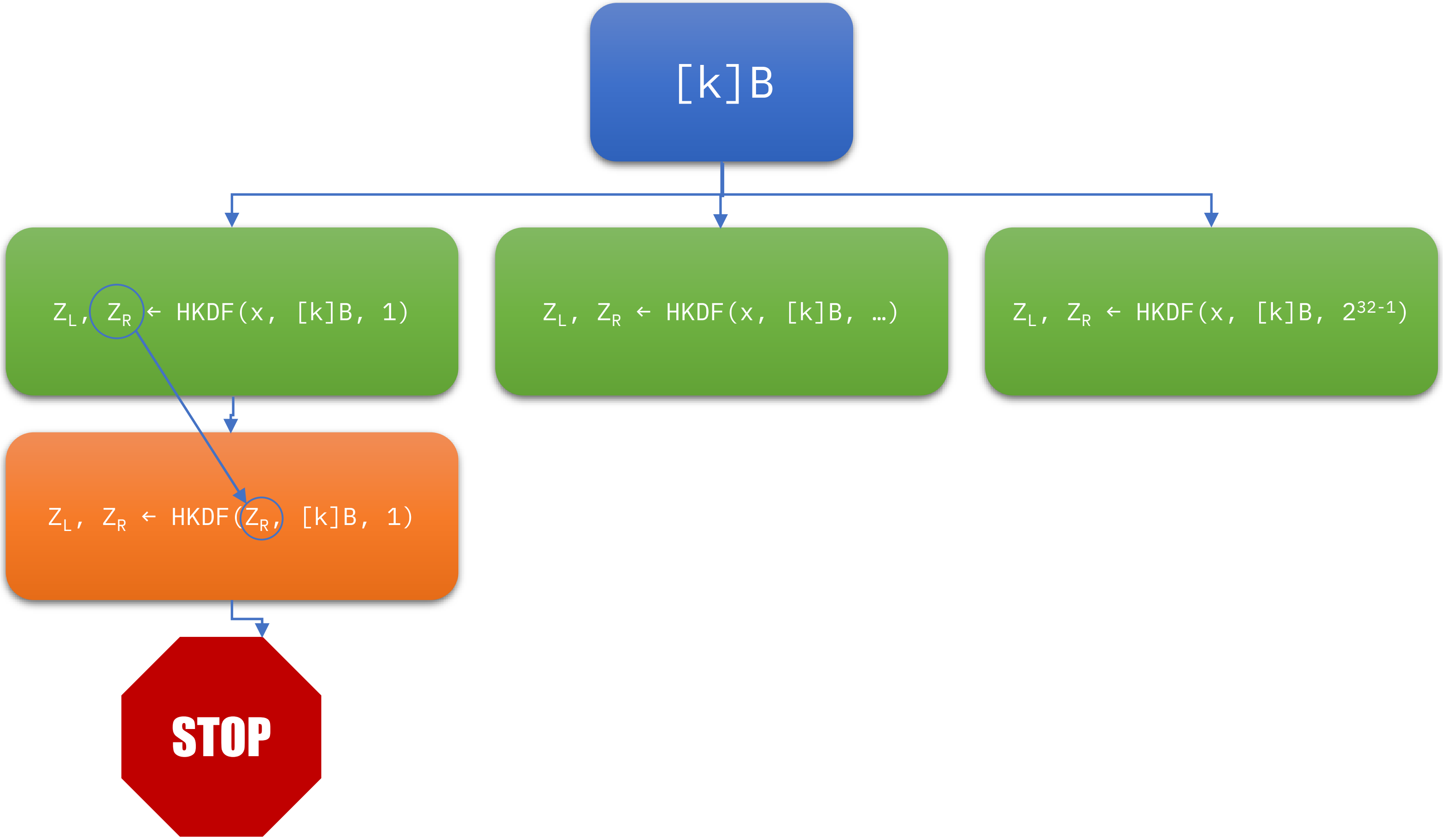
Private key: $k + Z_L$

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

🟦 Root key

🟩 Child key

🟧 Hardened child key



[k]B

$Z_L, Z_R \leftarrow \mathrm{HKDF}(x, [k]B, 1)$

$Z_L, Z_R \leftarrow \mathrm{HKDF}(x, [k]B, \ldots)$

$Z_L, Z_R \leftarrow \mathrm{HKDF}(x, [k]B, 2^{32-1})$

$Z_L, Z_R \leftarrow \mathrm{HKDF}(Z_R, [k]B, 1)$

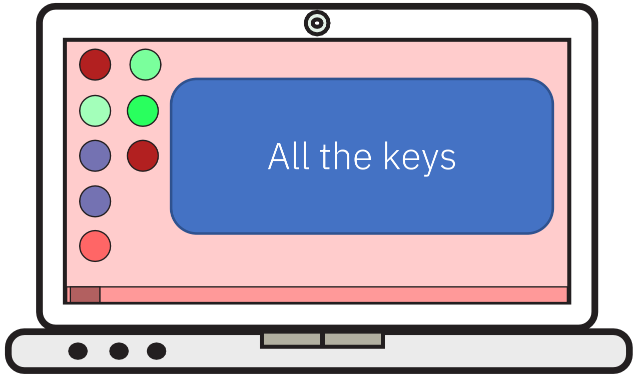
Potential applications to MLS

- 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.

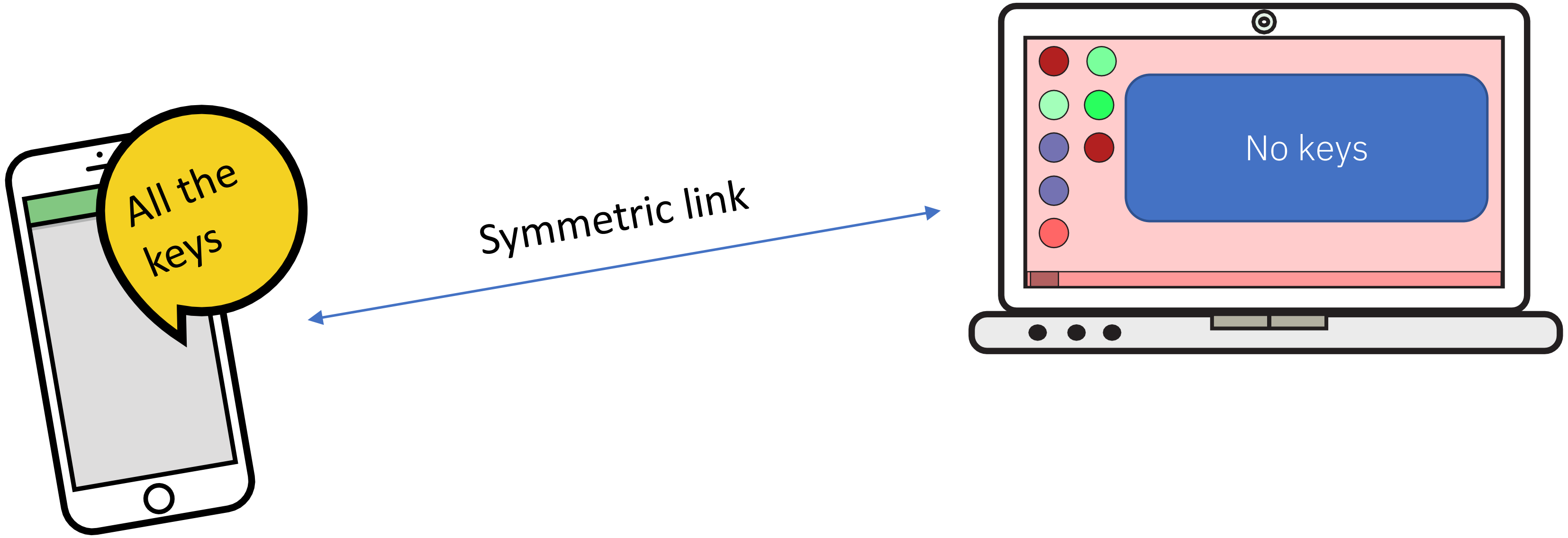
- Improvements are clear for partial state compromise.
- *But what are the improvements in the case of full state compromise?*

Signal Desktop key management

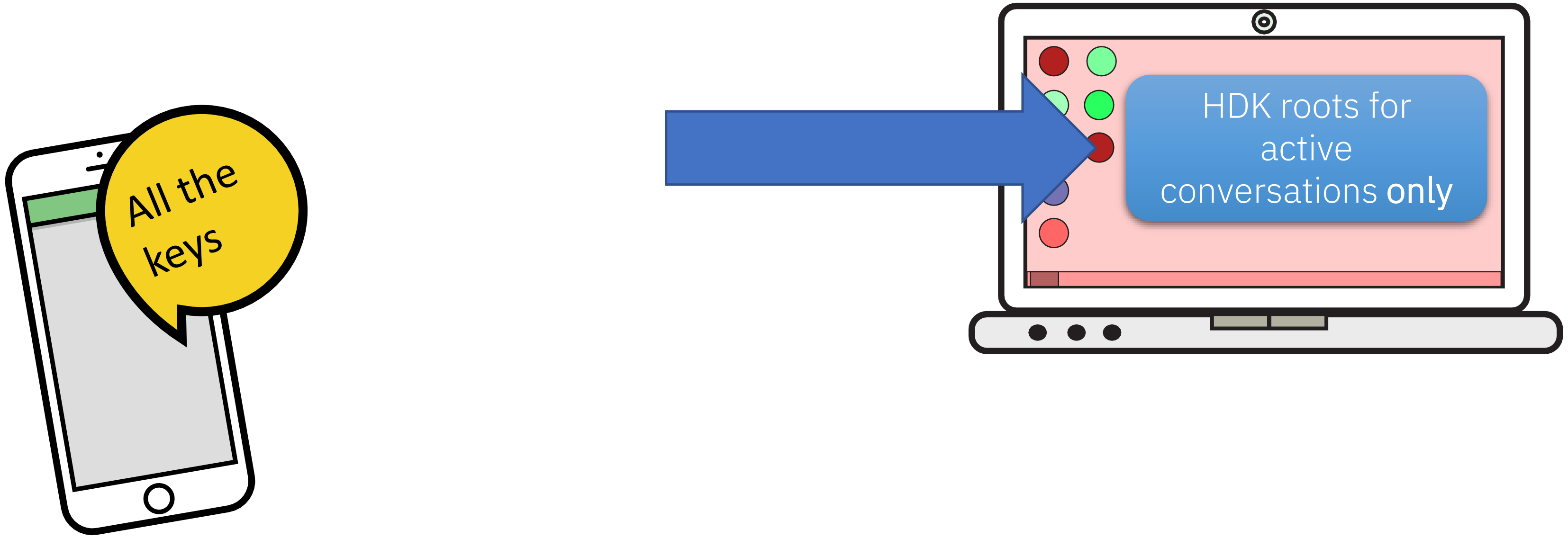
All the keys



WhatsApp Desktop key management



MLS Desktop key management



To what demarcation of state compromise can we generalize these improvements?