BIP32-Ed25519

Dmitry Khovratovich   Jason Law

University of Luxembourg

Evernym, Inc.

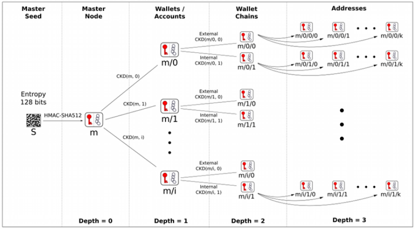
April 30th, 2017

Story of BIP32

Bitcoin wallet:

- Need of multiple addresses to minimize privacy leakage, every address is used just once;
- Single secret to derive all private keys;
- Public key generation can be delegated to third parties (example: web-shops);
- Hierarchy or its part can be proven without disclosing the seed.

BIP 32 - Hierarchical Deterministic Wallets

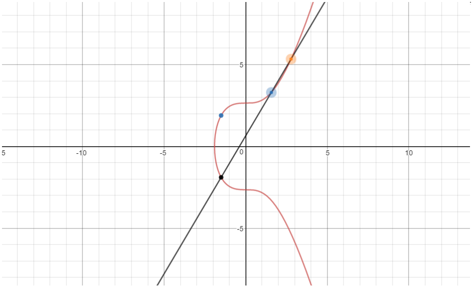


Child Key Derivation Function ~ CKD(x,n) = HMAC-SHA512(x_{Chain}, x_{PubKey} || n)

Bitcoin curve linearity

Bitcoin uses the secp256k1 curve:

- Base point B;
- Private-public key pair: $(k, [k] \cdot B)$;
- ECDSA signature needs a fresh randomness for each message;
- Linearity property: for any a the pair $(k + a, [k]B + [a] \cdot B)$ is a valid keypair.



BIP32 standard

Hierarchy of keys:

- Exploit the linearity of secp256k1 to generate arbitrarily many keypairs:

$$(k_{priv}, k_{pub}) = (k + a, [k]B + [a] \cdot B).$$

- Generate $a'_i \leftarrow F(k_{pub}, i, c)$ to get hierarchy indexed by i and controlled by the chain code c (can be disclosed if needed);
- Branches for $i \geq 2^{31}$ are called hardened as they need the private key to be produced.

Drawbacks:

- No design rationale;
- Too heavy primitives (HMAC instead of hash etc.)

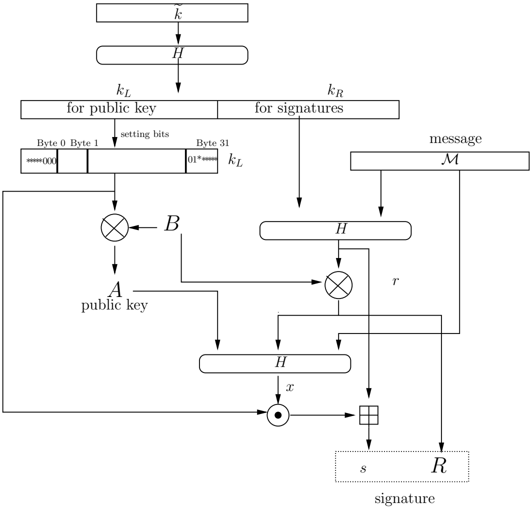
Ed25519/EdDSA

EdDSA:

- Based on the curve used for the DH handshake;
- Public design rationale;
- Security by design;
- Schnorr signatures: deterministic generation;
- Better performance;
- Non-trivial private key derivation from a master secret;
- Non-linear key space.

The EdDSA standard allows reusing the public keys as encryption keys derived from the DH handshake on Curve25519. This is (to some extent) used in SC4 and X3DH protocols.

Ed25519 signature generation



Need in BIP32-Ed25519

Many systems need both hierarchical keys and fast/secure signatures:

- Tor key blinding (in response to hidden services deanonymization attacks).
- Zcash decided to use the Bitcoin curve partly because no derivation scheme was proposed for Ed25519.
- Sovrin, the identity ledger, uses Ed25519 signatures and needs key derivation for users.

Possible to combine?

BIP32-Ed25519, minimum requirements:

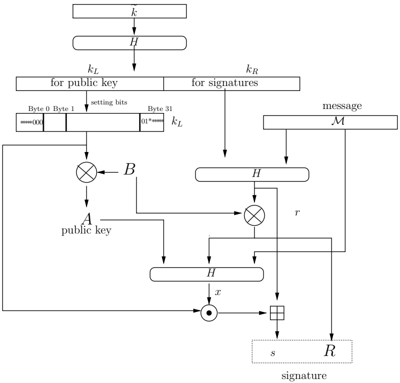
- Hierarchical key derivation;
- Signatures verification compatible with existing API of Ed25519-supporting libraries.

Additional requirements:

- Maximum compatibility with existing BIP32 code;
- Signature generation compatible with the existing API;
- Keypairs compatible with Curve25519;
- No potential vulnerabilities introduced.

Main problems

- Master secret undergoes hashing in Ed25519;
- The private key space is nonlinear;
- The BIP32 code generates 256-bit scalars for private keys, whereas we might need more.



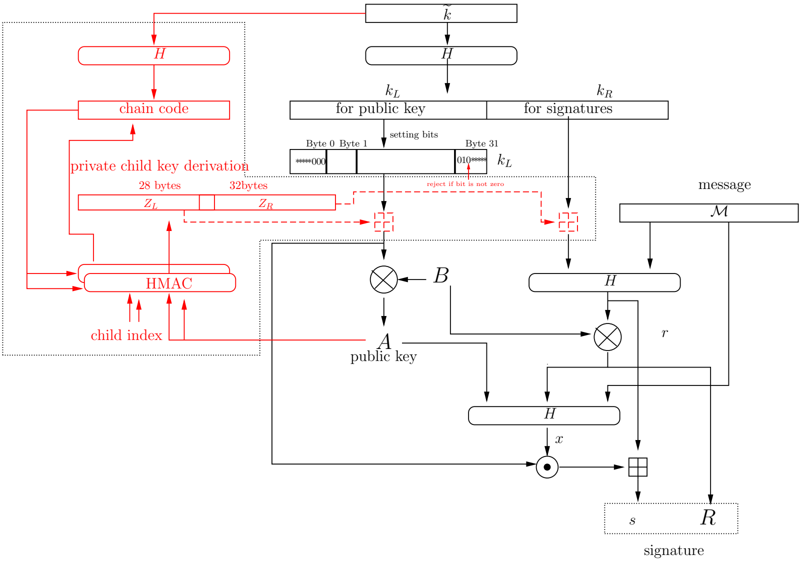
Main problems

- Master secret undergoes hashing in Ed25519;
- The private key space is nonlinear;
- The BIP32 code generates 256-bit scalars for private keys, whereas we might need more.

Mitigations:

- Work on extended private keys rather than on master secrets;
- Filtering master secrets to ensure key space linearity;
- Cloning HMAC calls.

Our solution



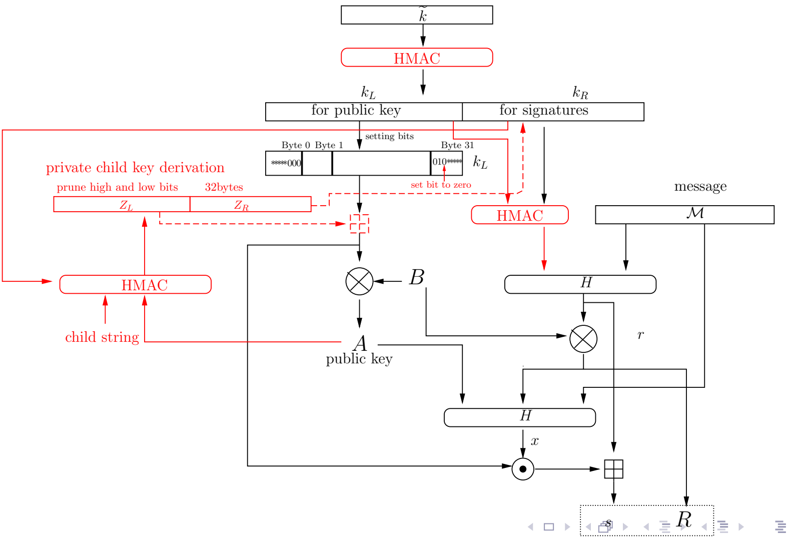
Chain v1

Independently of us, the Chain team released their own version of BIP32-Ed25519 in the fall of 2016. Main highlights:

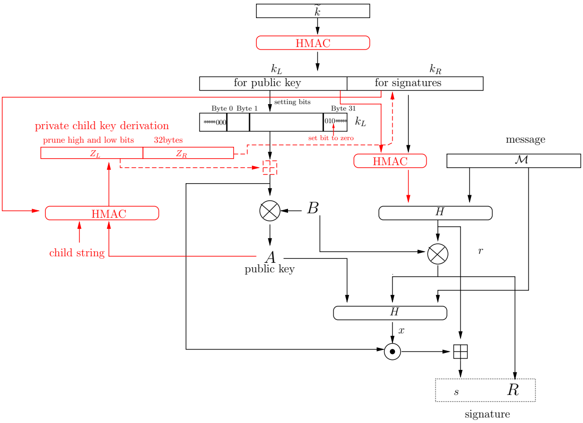
- No extra bit zeroed/checked in the private key, so easy overflows.
- As a result, side-channel timing leakage, thus timing attacks on the key generation are possible.
- The signature procedure incompatible with the EdDSA procedure (an extra HMAC call in the beginning).
- Some key parts are exposed to the public for the unknown reasons.

Chain v2

After a conversation with us Chain released an updated version in March 2017.



Chain v2

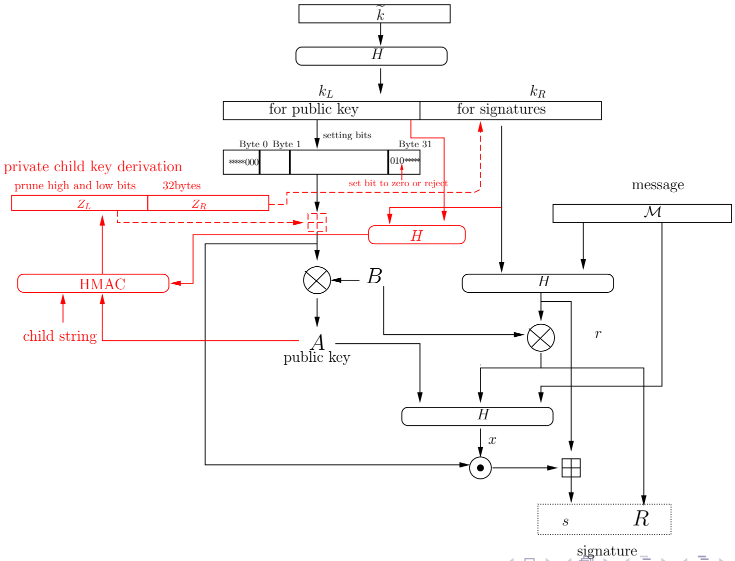


Crucial points: no more timing leakage, replacing the chain code with a key-derived value.

However, more intrusive: an extra HMAC call for the master secret, still incompatibility in the signature process.

Possible workaround

Both worlds combined:



Need for feedback

1. Other applications that need hierarchical keys;
2. Should the new private keys be compatible with Ed25519 private keys?
3. Will the new private keys be used for the DH handshake?
4. Should the signature generation process be the same as in Ed25519?
5. How much can we deviate from BIP32? Drop HMACs?