Research Spike: Evaluation of Using a Single Mnemonic both for Cryptocurrency and Identity Wallets

June 2, 2022

Abstract. # **DISCLAIMER**: Please, keep in mind that this is a working draft. Its main use in its current state is to foster debate on the topics it deals with. Do not take anything written here as ground truth.

1 Introduction

Hierarchical Deterministic Wallets (HD wallets, for short) in the cryptocurrency domain derive from the BIP32 specification[1] The motivation behind creating this type of wallets is clear if one thinks about how typical cryptocurrency systems work: each address is associated with a key pair and, in order to achieve high security and privacy levels, it is best to limit the reusage of addresses (thus, key pairs). Therefore, a too high number of needed key pairs is quickly reached. If each key is generated independently at random, management becomes too complex – especially, if several devices are expected to be synchronised. As a solution to this challenge, a hierachical tree-based data structure was proposed, in which each parent node can be used to derive multiple child nodes, deterministically. This approach seems also useful to foster some sort of separation – i.e., create sub-trees that are (computationally) independent from one another, in the sense that a compromise in one does not lead to a compromise in the other. Typically, the root node of the tree is derived from a seed encoded as a mnemonic. BIP39[2] is the main specification for this latter purpose.

1.1 Preliminaries

In the sequel, the processes that compose an (BIP44-compliant) HD wallet are described, including high-level cryptographic algorithms. For the sake of readability of non-cryptography savvy audience, we give informal descriptions of the cryptographic concepts and schemes mentioned next.

[1] `https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki`. Last access, December 13th, 2021.

[2] `https://github.com/bitcoin/bips/blob/master/bip-0039.mediawiki`. Last access, December 13th, 2021.

Signature schemes. A signature scheme uses an asymmetric key pair, composed of a public key and a private key (also frequently referred to as verification key and signing key, respectively). Intuitively, the owner of the signing key can produce a digital signature over arbitrary messages, which can be verified by anyone with the verification key, which is frequently somehow made publicly accessible. Secure signature schemes ensure that only the owner of the signing key can produce signatures verifiable with the corresponding verifying key. In the HD wallets studied next, typical signing algorithms are ECDSA and EdDSA.

HMACs and HKDFs. A Hash-based Message Authentication Code scheme is a concrete approach to build MAC functions, using cryptographic hashes as main building block. A MAC function is a kind of symmetric equivalent to digital signature – i.e., it relies on a shared secret rather than on an asymmetric key pair. HMACs have been proved to be secure pseudo random number generators [1], which is a useful building block to build secure HKDFs [7], or Hash-based Key Derivation Functions. Specifically, an HKDF, as defined in [7], can be applied to produce secure cryptographic keys from low entropy sources, by applying a two-phase process of randomness extraction and expansion (the extract-expand approach). In summary, given a (possibly low-entropy) bitstring, an HKDF produces an arbitrarily long pseudo random bitstring that is fit for cryptographic purposes.

2 An Overview of BIP32-based HD Wallets

In this section, we give a brief overview of the BIP32 specification, and provide a "full path" from mnemonic to key pair. To avoid repeating already well documented processes, the overview will refer to the specifications when appropriate. In a final subsection, we will also overview how does all this apply to the Cardano Lightwallet and Atala Prism cases.

2.1 Main Specifications: A Brief History of BIPs for Wallets

BIPs, or Bitcoin Improvement Proposals, are "standards" in the Bitcoin domain that specify how concrete components of the ecosystem work and/or should be implemented. Being Bitcoin the first blockchain, many others import these standards.

In 2013, BIP32 was published. It defines an approach to HD wallets using a tree structure in which each node can be associated to a key pair. It also defines two ways to derive child nodes from parent nodes: one that allows, given a public key of a node, to derive the public and private keys of all its descendants; and another one that requires the private key for derivation of its child nodes. Based on this child derivation processes, it also specifies a basic strategy for building HD structures for payments.

However, although BIP32 does specify a basic structure, it does not mandate that it has to be followed. According to BIP43[3], published in 2014, this led to a set of self-claimed BIP32-compliant wallets not to be compliant between themselves. BIP43 requires that the first level of child derivation (directly from the root) be dedicated to define a "purpose". BIP44[4], published simultaneously to BIP43, further refines this, by establishing 6 different levels (which we introduce in subsequent sections). Thus, BIP44 is an specialization of BIP43, which is itself an specialization of BIP32.

Somehow orthoghonally to these, BIP39 defines how to encode random binary seeds into mnemonic sentences, and back. In this manner, the root node of a BIP44 wallet is decoded into a bitstring, from a mnemonic, using BIP39. Then, this root node and its child nodes are computed as defined in BIP44. Table 1 summarizes the mentioned standards.

BIP	**Year**	**Description**
BIP32	2013	Basic structure and derivation rules for HD wallets.
BIP43	2014	Definition of "purpose" level.
BIP44	2014	Definition of 6-level structure (root, purpose, coin type, account, change, address.)
BIP32	2013	Encoding/decoding of bitstrings to/from mnemonic sentences.

Table 1. Summary of BIPs for HD wallets.

HD wallets that follow the BIP44 specification have to define a value of 44 (with hardened derivation; see next) at the "purpose" layer. Wallets following a different structure, but still compatible with BIP32, can (should?) define their own code, for the sake of ensuring compatibility across implementations. For instance, Cardano defines the code 1852 (again, hardened derivation), as it has some minor variations with respect to BIP44. There are several well-known wallets that follow this specification. For instance, Trezor[5], or Ledger[6].

Additionally, Satoshi Labs SLIP44 maintains a registry of the different cryptocurrencies that have "registered" codes for wallet specifications that follow BIP44. That is, this registry contains codes for the "coin type" level of BIP44. Examples of registered coins include Bitcoin mainnet and testnet, Litecoin, Cardano, Ether, and many others (currently, there are several hundreds).

[3] `https://github.com/bitcoin/bips/blob/master/bip-0043.mediawiki`. Last access, December 13th, 2021.

[4] `https://github.com/bitcoin/bips/blob/master/bip-0044.mediawiki`. Last access, December 13th, 2021.

[5] `https://docs.trezor.io/trezor-firmware/misc/coins-bip44-paths.html`. Last access, January 10th, 2022.

[6] `https://support.ledger.com/hc/en-us/articles/4404388633489-Export-your-accounts?docs=true`. Last access, January 10th, 2022.

2.2 Main Concepts

HD wallet tree structure. HD wallets are organised as trees, where the root node is directly derived from the main seed (typically, a mnemonic-encoded random string). We denote the root level with level 0, and all descendants with increasing numbers. A level other than the root level can have an arbitrary number of nodes.

(Child) Index. The child index just identifies how many childs can be (or have been) created before the current child, at the current depth of the tree. We assume 0 to be the first index.

Chaincode. To introduce extra "non-determinism" in the way child keys are derived from parent keys, part of the pseudo-randomly derived data is used as key for pseudo-random derivations of lower levels. This part of the derived data is referred to as chaincode. We will denote the chaincode of the j-th node at level i of the tree with $c_{i,j}$. This data should be kept secret; otherwise, entire subtrees can be compromised.

Extended keys. An extended public (resp. private) key is composed by the public (resp. private) key and the chaincode.

Hardened child keys. A hardened child key can only be derived from the parent private key, and the parent key chaincode.

Non-hardened child keys. A non-hardened child key can be derived both from the parent private and parent public key, and the parent key chaincode.

Hardened vs non-hardened child derivation. Basically, child nodes are derived as rerandomizations of their parent node private key. The pseudo-random data used for rerandomization can be derived from the parent private extended key, or from the parent public extended key. The former case is referred to as hardened derivation, and the latter as non-hardened derivation. In a nutshell, this means that hardened child keys (private or public) can only be derived from their parent private key, but not from their parent public key. Naturally, child private keys can only be derived from parent private keys.

For self-containedness, Fig. 1 shows the different algorithms to derive child keys from a parent keys, excluding some details. For the full algorithms, we refer to BIP32.

Isolation of HD wallet subtrees. In the following section, we use the concept of isolated and non-isolated subtrees. This is something that is only indirectly mentioned in the BIPs and other related works, but which is of crucial importance in order to limit the impact of compromised keys. Namely, assume that a child key is derived in non-hardened mode. Then, if the extended parent public key is leaked, as well as the private (non-extended) non-hardened child key, it is possible to extract the parent private key – and, therefore, all the keys (private and public; hardened or not) that derive from it. If child keys are derived in hardened mode, this does not apply. For the sake of clarity, we illustrate the difference through the diagram in Fig. 2.

For instance, assume that the extended public key $(K_{i,j}, c_{i,j})$ is compromised, along with its child private non-hardened key $k^{nh}_{(i+1),t}$. Then, it is straightforward

$prv2HardChild$ // Parent private key $\rightarrow$ i-th Child hardened key pair

$I \leftarrow HMAC(c_{par}, k_{par}, i)$

$k^h \leftarrow (k_{par} + Left(I)) \bmod n; K^h \leftarrow k^h G; c^h \leftarrow Right(I)$

return (k^h, K^h, c^h)

$prv2NonHardChild$ // Parent private key $\rightarrow$ i-th Child non-hardened key pair

$I \leftarrow HMAC(c_{par}, K_{par}, i)$

$k^{nh} \leftarrow (k_{par} + Left(I)) \bmod n; K^{nh} \leftarrow k^{nh} G; c^{nh} \leftarrow Right(I)$

return (k^{nh}, K^{nh}, c^{nh})

$pub2NonhardPubChild$ // Parent public key $\rightarrow$ i-th Child non-hardened public key

$I \leftarrow HMAC(c_{par}, K_{par}, i)$

$K^{nh} \leftarrow Left(I)G + K_{par}; c^{nh} \leftarrow Right(I)$

return (K^{nh}, c^{nh})

Fig. 1. Algorithms for child derivation. $k_{par}, K_{par}, c_{par}$ denote the parent's public key, private key, and chaincode, respectively. k, K, c denote the resulting child's private key, public key, and chaincode. h or nh superscripts denote hardened or non-hardened derivation. $Left$ (resp. $Right$) divides a bitstring in two equal parts, and takes the left (resp. right) half. The result of both $Left$ and $Right$ can be interpreted as a scalar, and thus multipled by points in the elliptic curve, such as G, which is the base point, or public keys, which also are points in the curve.

to recompute the *parent* private key $k_{i,j}$ (independently on whether it was derived in hardened mode or not) as: $k_{i,j} \leftarrow (K^{nh}_{(i+1),t} - Left(I^{nh}_{(i+1),t})) \bmod n$. Obviously, given the parent private key $k_{i,j}$, it is straightforward to derive all its descendants (hardened or not), as well as possibly and recursively apply the same strategy, if the just recovered parent key was derived in non-hardened mode.

2.3 From Mnemonic to Key Pairs: High-level Overview

The following description is based on BIP39 for the processing of mnemonics, and BIP44 (which is a restriction of BIP43 and BIP32) for general structure of an HD wallet, which we depict in Fig. 3.

Path notation. As can be seen in Fig. 3, the hierarchical structure of an HD wallet is defined in layers (or levels), where different indexes are used to derive the cryptographic values associated to different nodes in a same layer, with the exception of the root node (at layer 0), which is frequently just denoted with an `m`, after master seed. Thus, for instance, paths `m/1` and `m/2` denote a tree with two layer 1 nodes computed with index `1` and index `2`, respectively. If a child node index is *primed*, it means that hardened derivation is used (whereas

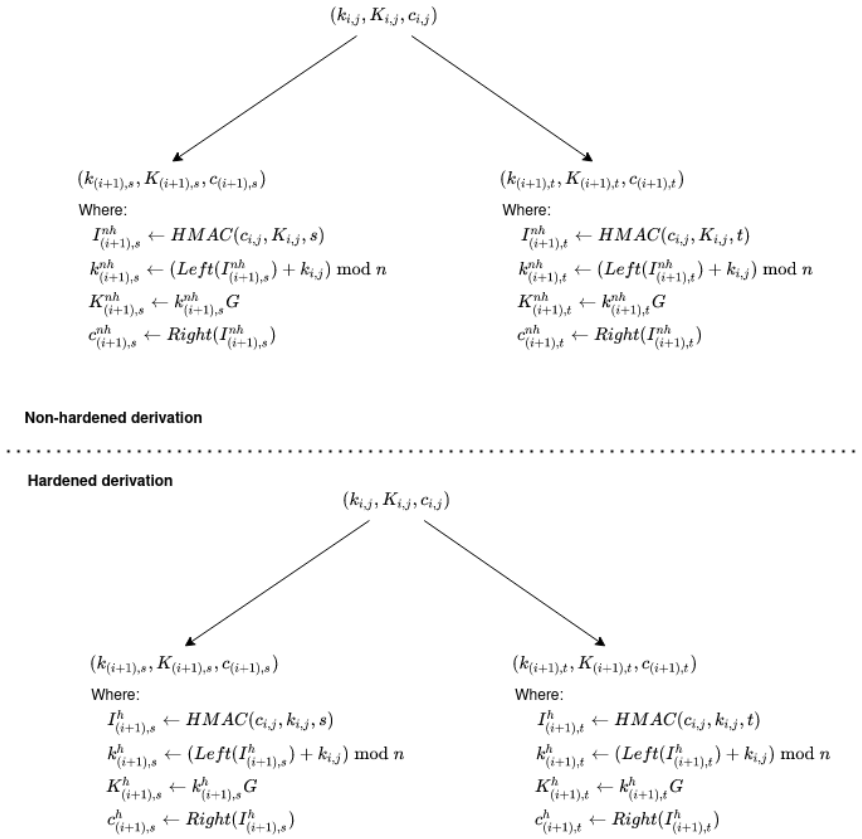

Fig. 2. Sub-tree isolation through hardened child derivation. G is the base point of the underlying elliptic curve, and n is the order of the associated group.

the absence of a prime means non-hardened derivation). More specifically, since BIP44 wallets have up to layer 6, paths look like `m/a'/b'/c'/d'/e/f`. Note that layers 1 to 3 are derived in hardened mode, and layers 4 and 5 are derived in non-hardened mode.

From (pseudo-)random bitstrings to mnemonics, and back. Mnemonics encode randomness in human-readable form. BIP39 is the main specification for this. Roughly, between 128 and 256 random bits (in multiples of 32) are generated, a checksum (the first few bits of a hash of these random bits) is concatenated, and the result is divided in chunks of 11 bits. Each of these chunks is used as index to a predefined dictionary of words. The result is a mnemonic of 12 to 24 words. To convert from mnemonic to random bits, just the inverse process has to be followed (verifying the checksum).

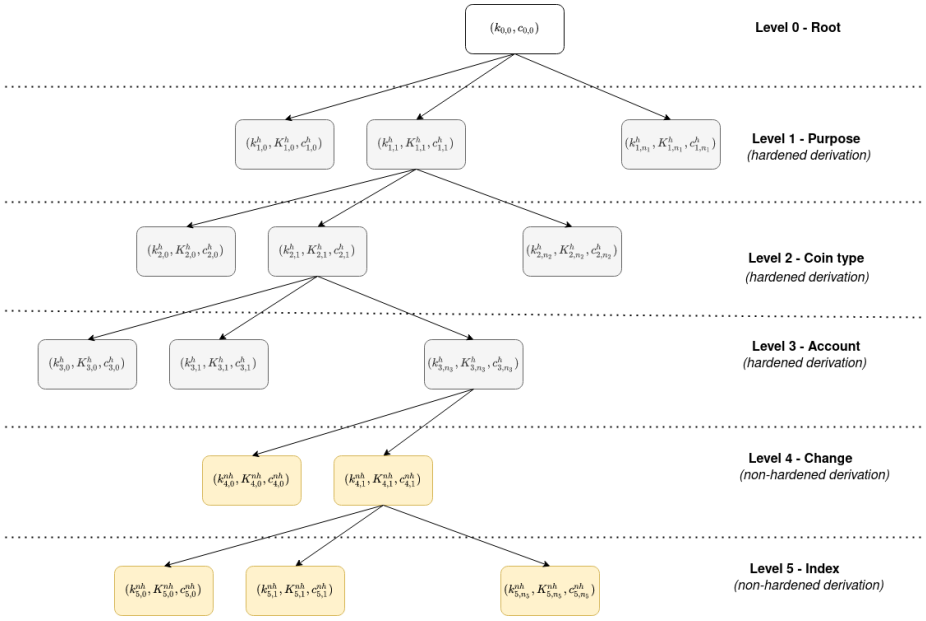

Fig. 3. Basic tree structure of a BIP44-based HD wallet.

Given the mnemonic, it is converted back into a bitstring as just described. Then, this bitstring is processed with an HMAC function, using the string "Bitcoin seed" as key based on SHA512 – hence producing 512 bits of output. The first 256 bits are set as master secret key, and the last 256 bits as master chain-code (i.e., $k_{0,0}$ and $c_{0,0}$, in our previous notation).

Layer 1: isolated *subtrees per* `purpose` The first layer of a BIP44-compatible HD wallet is composed of up to 2^{31} nodes of hardened childs: i.e., layer 1 is composed of childs $(1, 2^{31})$ to $(1, 2^{62} - 1)$, where the second element in the pair is the index i, whose value is used in the child derivation processes, and denotes for what "purpose" the descendant keys will be used – this roughly translates to the system (e.g., Bitcoin, Cardano... or even non-blockchain systems, in theory).

Note that, being layer 1 computed in hardened mode, a private key compromise of one node in layer 1 does not affect the master node, nor other layer 1 siblings. Hence, this is an "isolated" subtree, according to our terminology.

For BIP44-compatible wallets, the expected value for the i index is 44, so all paths begin with `m/44'`. Cardano, since the Shelley era, uses index 1852 (and thus, Cardano paths begin with `m/1852'`[7]. Note that, since these are hardened indexes, the encoding rules translate this into $1852 + 2^{31}$ (resp. $44 + 2^{31}$ for BIP44); see BIP44 for more details.

[7] `https://cips.cardano.org/cips/cip1852/`. Last access, December 14th, 2021.

Layer 2: isolated *subtrees per* ***coin type*** The second layer is again composed of up to 2^{31} nodes of hardened childs: i.e., layer 2 is composed of childs $(2, 2^{31})$ to $(2, 2^{62} - 1)$ per each child node of layer 1. In theory, up to $2^{31} * 2^{31} = 2^{62}$ second-level nodes can coexist in the same HD wallet. The aim of the second level is for any HD wallet to be able to support multiple types of coins. E.g., so that the same mnemonic can be used to derive key pairs for Bitcoin, Cardano, etc.

Note that, being layer 2 computed in hardened mode, a private key compromise of one node in layer 2 does not affect its parent in layer 1, nor other layer 2 siblings. Hence, this is an "isolated" subtree, according to our terminology.

Although not mandatory, each coin is expected to have an assigned coin type index. The complete list of reserved indexes is available in SLIP44[8]. For instance, Cardano's ADA has an index of 1815 in hardened mode – thus, its path prefix is `m/1852'/1815'`.

Layer 3: isolated *subtrees per* ***account*** The third layer is again composed of up to 2^{31} nodes of hardened childs: i.e., layer 3 is composed of childs $(3, 2^{31})$ to $(3, 2^{62} - 1)$ per each child node of layer 2. In theory, up to $2^{3*31} = 2^{93}$ third-level nodes can coexist in the same HD wallet. The aim of the third level is to allow the HD wallet user to maintain many accounts per coin type. E.g., the same mnemonic seed could be used for up to 2^{31} key pairs for Cardano addresses.

Again, note that being layer 3 computed in hardened mode, a private key compromise of one node in layer 3 does not affect its parent in layer 2, nor other layer 3 siblings. Hence, this is an "isolated" subtree, according to our terminology.

Layer 4: non-isolated *subtrees per* ***change*** According to BIP32, the fourth layer can be composed of up to 2^{31} nodes of non-hardened childs: i.e., layer 4 can be composed of childs $(4, 0)$ to $(4, 2^{31} - 1)$ per each child node of layer 3. However, in BIP44, only non-hardened childs 0 and 1 are used: i.e., each node of layer 3 will only have non-hardened childs $(4, 0)$ and $(4, 1)$. This layer is used to differentiate whether the descendant nodes will be used as change addresses, or not. I.e., child $(4, 0)$ will *not* be used for receiving change in payments, and therefore is expected to be associated to an address that may be published outside of the wallet; while child $(4, 1)$ is expected to be used for receiving change in payments, and will typically not be published outside of the transaction.

Note that, since nodes in layer 4 are not derived in hardened mode, if the private key of a layer 4 node is compromised, along with the extended public key of its layer 3 parent, then the entire subtree of that layer 3 node will be compromised (but not the subtrees of other layer 3 nodes, as layer 3 is derived in hardened mode). For example, take the derivation path of a layer 4 node to be `m/x'/y'/z'/0`. If the private key of `m/x'/y'/z'/0` is leaked to an attacker, who also gains access to the *extended public* key (i.e., the public key and chaincode)

[8] `https://github.com/satoshilabs/slips/blob/master/slip-0044.md`. Last access, December 14th, 2021.

of `m/x'/y'/z'`, then the attacker will be able to compute the extended private (and public) keys of the sibling node `m/x'/y'/z'/1`. This is precisely the attack explained in the last paragraph of Section 2.2.

Layer 5: non-isolated *final key pairs (`index`)* The fifth and last layer is used to derive final key pairs. It can be composed of up to 2^{31} non-hardened nodes: i.e., every change (and non-change) node of level 4 may have level 5 childs from $(5, 0)$ to $(5, 2^{31} - 1)$.

Note that, since layer 4 nodes are derived in non-hardened mode, if a layer 5 private key is compromised, as well as its parent extended public key of layer 4, then the whole layer 4 subtree will be compromised. Furthermore, if the layer 3 extended public key from which the layer 4 parent derives is also compromised, the entire account will be compromised too, as layer 4 is also derived in non-hardened mode.

2.4 Comments on a Recent Security Evaluation of BIP32 Wallets

Very recently, [2] was published in CCS'21, a major applied cryptography and security conference. The paper builds on prior work from 2019 [3], and analyses the security of BIP32-like wallets. For this purpose, the authors first model and analyse (additively) re-randomizable signatures built on ECDSA. They prove security, as long as every message is signed at most once, at the cost of a security loss proportional to the number of signature re-randomizations performed by the adversary[9]. Then, they build a model for HD wallets, and provide a generic construction using additively re-randomizable signatures. The properties considered in this HD wallet model are unlinkability and unforgeability (restricted to only one signature per message). The unlinkability property expects that an adversary cannot distinguish (uncompromised) keys derived from a non-hardened node, from keys derived from keys derived from an independent master key. The unforgeability property is the usual one, excluding the one message restriction (which seems reasonable, as long as new randomness is included in every signed message); namely, the adversary should not be able to create a valid signature under a hardened key that has not been compromised, or a non-hardened key.

An interesting extension is mentioned, to achieve forward unlinkability in case of compromise of non-hardened node public key and chaincode. Namely, if that happens, the adversary will be able to derive the public extended keys for all the subtree, which breaks unlinkability for the subtree. The proposal is to keep a state along with every node. On every child derivation in the tree, the state of the existing nodes is refreshed. Then, in case of compromise of a node, the keys previously derived in subtrees of it are not affected (assuming that previous states are securely erased).

Building on the proven security of the additively re-randomizable ECDSA, they prove security of their generic construction, which now also depends on

[9] They also prove this security loss to be unavoidable with additive re-randomization, although I have not checked that proof.

the number of leaked hardened keys. Assuming that roughly 1% of the total number of (additively re-randomized) hardened keys are compromised, the estimated security level for ECDSA of 256-bit keys (hence, 128-bit security) is 91-bit security.

Discussion. The model does seem to follow BIP32, but **assuming a hot/cold approach**. Concretely, they assume that non-hardened secret keys are always kept in a cold wallet, and therefore cannot be compromised. On the other hand, hardened secret keys are assumed *not* to be stored in cold wallets, and therefore can be compromised (this is reflected in the model with corresponding oracles or lack thereof). The underlying reasoning being that hardened derivation is used to share keys with not fully trusted entities (e.g., employees in a company). Also, hardened keys are assumed to be leafs (i.e., keys used directly for signing); or, alternatively, roots of independent subtrees. Under this modelling, if the partially trusted entity leaks the hardened key, this would only affect the security of its subtree (if it exists), but certainly not siblings or ancestors. However, this may not be fit for BIP44 wallets, where hardened derivation is expected to be used directly for the first 3 layers, and the leaves are actually computed with non-hardened derivation.

The property of forward unlinkability is interesting for identity-related wallets.

3 IO Ecosystem Specifics

Lightwallet. HD wallets in Cardano are mostly BIP44-compatible wallets, with the exception that the used elliptic curve is Ed25519, instead of Secp256k1, like Bitcoin's BIP32 (and thus, BIP44). This difference is marked by using index 1852 for hardened child derivation at layer 1 (purpose level), as specified in CIP1852[10]. Mnemonics in Cardano are composed by 24 words, and hence encode 256-bit random bitstrings.

Atala. HD wallets in Atala follow the same overall rules as in BIP32. Specifically, the same distinction between hardened and non-hardened child derivation rules are applied, as well as Secp256k1 as the elliptic curve, wich ECDSA as digital signature scheme. However, Atala defines a different tree structure, with the following layers, depicted also in Fig. 4:

Layer 0: Root. Encoded as/derived from a 12-word mnemonic.
Layer 1: DID Number. Obtained by hardened child derivation from the root. Each node in this level corresponds to a different DID.
Layer 2: Key Type. Obtained by hardened child derivation from a layer 1 node. Currently, the Atala specification defines four types of keys: master keys, issuing keys, communication keys, and authentication keys.

[10] `https://cips.cardano.org/cips/cip1852/`. Last access, December 15th, 2021.

Layer 3: Index. Obtained by hardened child derivation from a layer 2 node. Each layer 2 node may have up to 2^{31} child nodes, which are the leaves of the tree.

See the key derivation document for details[11]. Mnemonics in Atala are composed by 12 words, and hence encode 256-bit random bitstrings.

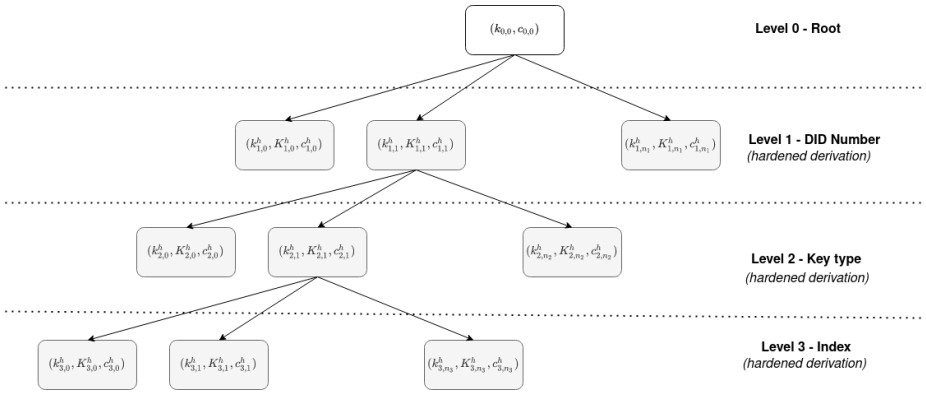

Fig. 4. Atala tree derivation structure.

Note on the different cryptographic choices. As stated, Cardano's Lightwallet CIP1852 follows a variation of BIP44 where the chosen singing algorithm is EdDSA, which uses Ed25519 as elliptic curve; whereas Atala builds from BIP32, thus employs ECDSA with a secp256k1 curve. The reason behind this difference seems just circumstancial, and due to the fact that Atala started to build from a different specification. From a cryptographic point of view, it is possible to unify under the same curve, or to stick to different ones. In the latter case, however, we must make sure that domain separation is correctly done: i.e., even if having a sort of unified wallet derived from the same mnemonic, it should not be possible to generate the same key to be used for payment purposes and identity purposes. More detail on this is given in Section 3.1.

Other than the previous, either choice seems to have its pros and cons. On the one hand, having both Cardano and Atala use the same curve would make it possible to benefit from joint development efforts. Also, it would avoid pitfalls such as using the same key in different curves, which can lead to security issues as described next (however, this can also be avoided with proper domain separation). Also, Edwards curves (such as Ed25519) are, at least in theory, more efficient. On the other hand, Secp256k1 is an standardized curve, which has

[11] `https://github.com/input-output-hk/atala-prism/blob/master/prism-backend/docs/protocol/key-derivation.md`. Last access December 15th, 2021. **WARNING! Internal link!**

seen more implementation and optimization effort. This means that the better theoretical efficiency of Ed25519 may somehow "mitigated" by more efficient implementations in Secp256k1. Also, Secp256k1 probably has more availability of implementations in different programming languages. In any case, this is something that needs to be considered by the development/engineering teams.

Related effots. There seems to be an effort to analyse the security of wallets that manage addresses combining payment and staking keys [6]. These wallets are referred to as PoS wallets. The paper identifies a series of malleability attacks, depending on how the addresses are derived from the associated staking and payment keys. A core wallet functionality is proposed, and a base protocol realizing that functionality is given. While the proposal seems to be somehow compatible with BIP44 wallets, the proposal seems to be somehow orthogonal to the way keys are derived – instead, they focus on how to generate addresses, and check that produced/received addresses are correct.

3.1 Aspects to consider before unification

Next, we emphasize some concrete topics that would need to be considered from an implementation/software-design point of view, prior to making Atala and Cardano keys to be derived from a single mnemonic.

Domain separation when different cryptosystems is necessary. As mentioned, Cardano uses EdDSA (which is based on Ed25519 curve), while Atala uses ECDSA with Secp256k1. It is *not* a good idea[12] to use the same key in different cryptosystems, nor for different curves, even though "structurally" it may be possible (e.g., in EdDSA and ECDSA, private keys are 32-byte random numbers). However, this is easily avoidable by ensuring domain separation in the key derivation functions that are applied on the common seed. This is an important concern on its own; yet, it should be addressed natively through the considerations in the next paragraph.

Ensure a correct hierarchical key derivation strategy. We need to take into account what specific usage we expect from the wallets and, from there, define hierarchical derivation rules that ensure security without disabling current utility. Related to the previous paragraph, we should make sure that keys that are aimed to be used in different cryptosystems, are derived in separate tree branches that ensure domain separation. Or, since the Atala derivation path is shorter, than no Atala derivation path can be a prefix of a Cardano derivation path. For instance, according to the current specification, the derivation path `m/1852'/1815'/0'`

[12] I have not been able to find a paper that describes some concrete related attack or gives some impossibility result for proving security under this circumstance but, at the least, it seems to be folklore knowledge. See Lindell's answer at `https://crypto.stackexchange.com/a/54666/52362` for instance. [4,9] seem good references to study this topic further, if needed.

corresponds to account `0'` for Ada coins in CIP1852-compliant Cardano wallets. In Atala, `1852'` could be a valid DID number and, while `1815'` is currently not a valid key type, if, in the future it becomes so, then `m/1852'/1815'/0'` would be a valid derivation path in both Atala and Cardano – which could open an attack vector. An easy fix would be to concatenate the master seed with some value `C` in Cardano, and some distinct value `A` in Atala. The derivation paths would become `Cm'/1852'/1815'/...` in Cardano, and `Am'/...` in Atala – hence, domain separation would be ensured. Probably, many other alternatives exist, more suitable from an engineering perspective (e.g., assigning Atala its own `purpose` level in BIP44 wallets). But, whatever option is followed, domain separation should be ensured.

On the mnemonic length. Atala uses 12-word mnemonics to encode the master seed from which all keys are derived, while Cardano uses 24 words. According to BIP39, 12 words encode 128-bits seeds, and 24 words encode 256-bit seeds. These entropy bits are then subject to the extract-and-expand approach of the HKDF used in BIP32 to produce pseudoranom bits that are computationally indistinguishable from true random numbers[13]. The keys used by both Atala (for ECDSA), and Cardano (for EdDSA), are thus securely derived pseudorandom keys of 256 bits. For keys of 256 bits, both ECDSA and EdDSA are expected to provide 128 bits of security against unforgeability attacks. Given this, and assuming (like [2] does) that some keys will be compromised, there are two ways to attack a wallet: by breaking security of the HKDF used to derive the keys; or by breaking security of the wallet construction itself. Given this, on the one hand, in [2][14], security of BIP32 wallets against unforgeability is estimated to have a security loss of 37 bits over the underlying digital signature scheme (under somewhat arbitrary estimations for a typical setting, like leaking about 1% of roughly 2^{20} produced keys). For 256-bit ECDSA keys, this translates to 91 bits of security. On the other hand, even for 128-bit seeds (thus, with entropy at most 128), breaking the security of the HMAC-based HKDF construction (that would seem to affect the unlinkability property of BIP32 wallets, rather than their unforgeability) would be harder than breaking the unforgeability property of the BIP32 wallet, as per the results in [7][15]. Given this, and lacking a more detailed analysis, seeds of 12 words would appear to be enough for security, strictly speaking. However, taking into account that we are aiming to combine two wallets into one, it does not seem logical to reduce the overall security of the wallet that uses 24-word mnemonics, as the unified wallet will use the

[13] Note: only 256 of these bits are used to produce the final keys.

[14] Important! Note that [2] is defined for ECDSA-based BIP32 wallets with a structure slightly different from BIP44 wallets. Hence, its results may not translate to our setting – and of course, neither to EdDSA-based BIP44 wallets.

[15] According to the analysis in [7], the outputs of HMAC-based HKDFs are roughly $q2^{-m}$-close to uniform, where q in our setting would be 2^{20} as per [2], and m is the min-entropy of the randomness source (our mnemonic). Hence, for seeds (mnemonics) of 128 bits of entropy, we can still expect the outputs to be $\sim 2^{-100}$-close to uniform.

same mnemonic to produce even more keys than what it was producing before. Therefore, it seems advisable to increase the mnemonic length of Atala to 24 words, rather than reducing the length of Cardano to 12 words.

4 Discussion

From a theoretical point of view, it seems very reasonable to derive all keys, even for different purposes (payments, staking, or identities) from the same mnemonic. Section 3.1, listed a series of topics that should be taken into account prior to unifying Cardano and Atala wallets under the same mnemonic, to avoid security pitfalls. In addition to those, there are also some aspects that may need careful consideration, as they may have impact in the overall security and privacy properties of the wallet and related systems.

Malleability considerations when combining multiple attributes into one address. Are the keys for the different purposes going to be used *only* in a standalone manner? Or, rather, it can be expected that they are somehow encoded and distributed jointly? The latter seems to be the case for PoS wallets, which are expected to be used in the Cardano ecosystem. As pointed out by [6], naive encoding of keys with different purposes, into a single address, may enable certain (malleability) attacks.

Concrete differences in prior security analysis. The security analysis in [2] (which is one of the main references used in this research spike) is on BIP32, not BIP44. While BIP44 is actually a subset of BIP32, it further specifies it in concrete ways that may alter the result of the analysis. For instance, [2] assumes that non-hardened nodes are leaves of the tree (which, alternatively, can be seen as roots in new trees). However, in BIP44 the leaves are non-hardened nodes, and intermediate nodes are mostly hardened. Yet another possible source of discrepancy is that the analysis assumes that non-hardened nodes are maintained in a hot/cold wallet setting and, thus, the private keys cannot be compromised. This assumption may not hold for all use cases, and that would change completely the security reasoning. If we want to determine the concrete security level we want/need in some specific use cases that differ from the analysis of [2] as mentioned above, a new model might be necessary.

Careful consideration of extra security properties. The main related work analysed so far[16] study the security of either BIP32 wallets for payments, or PoS wallets. The introduction of identity-related (frequently associated to actual people) data may require additional properties, only mentioned in these works (or not considered, as in the BIPs). For instance, some sort of forward security will be very desirable. While, for the case of payments or staking, the impact of a compromise (either of a leaf node – i.e., a key – or a complete subtree) can be mitigated by prompt detection and transfer of funds to an uncompromised

[16] BIP32, BIP44, [2] and, partially, [6].

address, this may be harder for keys representing in some way real world identities. Most probably, the latter will frequently be long-lived keys, that also can, in turn, be used to issue further identities (e.g., as in the case of identity providers). Thus, the consequences of a compromise may be more cumbersome to address (re-issuing credentials, distributing revocation lists, etc.) and, consequently, it seems desirable to look for constructions that give some sort of guarantee about the security of keys produced at time $t' < t$, if a compromise happens at time t.

Also, it cannot be discarded that further desirable privacy or security properties arise, as we get more familiar with the targetted use cases.

Secure wallet storage. Currently, Atala has some basic support for encrypted storage[17], that basically allows storing and fetching data encrypted with a public key owned by a user. Some secure storage proposals for cryptocurrency wallets already exist, like Aries's RFC0050[18] or, in a more generic sense, Identity Foundation's Confidential Storage[19]. However, a careful analysis would be needed depending on the features we want to support (e.g., some sort of generic search on encrypted data vs just specific types of search), and also the expected infrastructure (e.g., an oblivious centralized secure storage service vs multiple user-owned devices that synchronize "automagically"). Although initially, these may be highly orthogonal topics to the hierarchical derivation of cryptographic keys, decissions in one domain may affect the other – e.g., if we are to implement some sort of forward security mechanism in an identity wallet, how can we ensure that proper re-encryption is done in a centralized oblivious secure storage system?

Trust registries. A core component of a decentralized identity system will be one that lets parties, a priori unknown to each other, derive some sort of trust based on the system itself. It is reasonable to expect that cryptographic keys (in the shape of DIDs, or related notions) are used to identify parties in the system, and assign them some sort of trust metric. As in the topic of secure storage, this would initially seem an orthogonal issue. But some properties (like, again, some notion of forward security), may require ad hoc updates of trust registries in a coordinated manner with an identity wallet – which might have some implications into the security analysis.

References

1. Bellare, M., Canetti, R., Krawczyk, H.: Keying hash functions for message authentication. In: Koblitz, N. (ed.) Advances in Cryptology - CRYPTO '96, 16th

[17] `https://github.com/input-output-hk/atala-prism/tree/53f3414a34c88a4a8e54b6d2128d4b14ce2ee8b9/prism-backend/docs/data-vault` **WARNING! Internal link.**

[18] `https://github.com/hyperledger/aries-rfcs/blob/main/concepts/0050-wallets/README.md` (last access January 18th, 2022).

[19] `https://identity.foundation/confidential-storage` (last access January 18th, 2022).

Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings. Lecture Notes in Computer Science, vol. 1109, pp. 1–15. Springer (1996). https://doi.org/10.1007/3-540-68697-5_1, https://doi.org/10.1007/3-540-68697-5_1

2. Das, P., Erwig, A., Faust, S., Loss, J., Riahi, S.: The exact security of BIP32 wallets. In: Kim, Y., Kim, J., Vigna, G., Shi, E. (eds.) CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021. pp. 1020–1042. ACM (2021). https://doi.org/10.1145/3460120.3484807, https://doi.org/10.1145/3460120.3484807
3. Das, P., Faust, S., Loss, J.: A formal treatment of deterministic wallets. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019. pp. 651–668. ACM (2019). https://doi.org/10.1145/3319535.3354236, https://doi.org/10.1145/3319535.3354236
4. Degabriele, J.P., Lehmann, A., Paterson, K.G., Smart, N.P., Strefler, M.: On the joint security of encryption and signature in EMV. In: Dunkelman, O. (ed.) Topics in Cryptology - CT-RSA 2012 - The Cryptographers' Track at the RSA Conference 2012, San Francisco, CA, USA, February 27 - March 2, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7178, pp. 116–135. Springer (2012). https://doi.org/10.1007/978-3-642-27954-6_8, https://doi.org/10.1007/978-3-642-27954-6_8
5. Freire, E.S.V., Paterson, K.G., Poettering, B.: Simple, efficient and strongly ki-secure hierarchical key assignment schemes. In: Dawson, E. (ed.) Topics in Cryptology - CT-RSA 2013 - The Cryptographers' Track at the RSA Conference 2013, San Francisco,CA, USA, February 25-March 1, 2013. Proceedings. Lecture Notes in Computer Science, vol. 7779, pp. 101–114. Springer (2013). https://doi.org/10.1007/978-3-642-36095-4_7, https://doi.org/10.1007/978-3-642-36095-4_7
6. Karakostas, D., Kiayias, A., Larangeira, M.: Account management in proof of stake ledgers. In: Galdi, C., Kolesnikov, V. (eds.) Security and Cryptography for Networks - 12th International Conference, SCN 2020, Amalfi, Italy, September 14-16, 2020, Proceedings. Lecture Notes in Computer Science, vol. 12238, pp. 3–23. Springer (2020). https://doi.org/10.1007/978-3-030-57990-6_1, https://doi.org/10.1007/978-3-030-57990-6_1
7. Krawczyk, H.: Cryptographic extraction and key derivation: The HKDF scheme. In: Rabin, T. (ed.) Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings. Lecture Notes in Computer Science, vol. 6223, pp. 631–648. Springer (2010). https://doi.org/10.1007/978-3-642-14623-7_34, https://doi.org/10.1007/978-3-642-14623-7_34
8. Luzio, A.D., Francati, D., Ateniese, G.: Arcula: A secure hierarchical deterministic wallet for multi-asset blockchains. In: Krenn, S., Shulman, H., Vaudenay, S. (eds.) Cryptology and Network Security - 19th International Conference, CANS 2020, Vienna, Austria, December 14-16, 2020, Proceedings. Lecture Notes in Computer Science, vol. 12579, pp. 323–343. Springer (2020). https://doi.org/10.1007/978-3-030-65411-5_16, https://doi.org/10.1007/978-3-030-65411-5_16
9. Thormarker, E.: On using the same key pair for ed25519 and an X25519 based KEM. IACR Cryptol. ePrint Arch. p. 509 (2021), https://eprint.iacr.org/2021/509

A Appendix

A.1 Feedback Welcome

Are you aware of further challenges that may be worth looking into? Have some feedback? Please, reach out to us:

- Jesus Diaz Vico (Atala Semantics team).
- Ezequiel Postan (Atala Semantics team).
- Tony Rose (Atala Head of Product).
- Bart Suichies (Atala Technical Director).

A.2 Further related work

There seems to be a growing body of related research, which is specially growing lately (probably, due to the rise of cryptocurrencies). The following just includes some of the main references for self-bookkeeping (besides [6] and [2], already mentioned in earlier sections). Note that they, in turn, include references to further related work.

- "Arcula: A Secure Hierarchical Deterministic Wallet for Multi-asset Blockchains", [8].
- "Simple, Efficient and Strongly KI-Secure Hierarchical Key Assignment Schemes", [5].
- "A Formal Treatment of Deterministic Wallets", [3].