Hierarchical deterministic Bitcoin wallets that tolerate key leakage
(Short paper)

Gus Gutoski[1] and Douglas Stebila[2]

[1] Perimeter Institute for Theoretical Physics, Waterloo, Canada
ggutoski@perimeterinstitute.ca
[2] School of Electrical Engineering and Computer Science
and School of Mathematical Sciences
Queensland University of Technology, Brisbane, Australia
stebila@qut.edu.au

Abstract. A Bitcoin wallet is a set of private keys known to a user and which allow that user to spend any Bitcoin associated with those keys. In a *hierarchical deterministic (HD)* wallet, child private keys are generated pseudorandomly from a master private key, and the corresponding child public keys can be generated by anyone with knowledge of the master public key. These wallets have several interesting applications including Internet retail, trustless audit, and a treasurer allocating funds among departments. A specification of HD wallets has even been accepted as Bitcoin standard BIP32.
Unfortunately, in all existing HD wallets—including BIP32 wallets—an attacker can easily recover the master private key given the master public key and any child private key. This vulnerability precludes use cases such as a combined treasurer-auditor, and some in the Bitcoin community have suspected that this vulnerability cannot be avoided.
We propose a new HD wallet that is not subject to this vulnerability. Our HD wallet can tolerate the leakage of up to m private keys with a master public key size of $O(m)$. We prove that breaking our HD wallet is at least as hard as the so-called "one more" discrete logarithm problem.

1 Introduction

Bitcoin [11] is a popular, decentralized cryptocurrency with monetary base worth approximately USD 5 billion as of December 2014. Each stash of Bitcoin (technically, an *unspent transaction output*) is associated with a public key Q for the Elliptic Curve Digital Signature Algorithm (ECDSA) [12]. A stash is spent by presenting a new *transaction* with a valid digital signature under Q. Under normal use, signatures are generated via knowledge of the private key d corresponding to Q. Ownership of a stash of Bitcoin is equated with knowledge of the associated private key.

A Bitcoin *wallet* is a set of private keys known to a user. A single wallet may contain hundreds or even thousands of distinct private keys. Wallets are

often stored in a database on a user's computer with appropriate backups to guard against accidental loss. A typical Bitcoin user is constantly generating new, random private keys and so frequent (and burdensome) backups are essential.

Deterministic wallets. *Deterministic wallets* alleviate much of the burden of wallet maintenance by generating a pseudorandom sequence of *child private keys* $d_1, d_2, \ldots$ from a *master private key* $\hat{d}$ according to a formula such as

$$d_i = \text{hash}(i, \hat{d})$$

where $\text{hash}(\cdot)$ is a cryptographically secure hash function that is indistinguishable from a random function and which may or may not be publicly known.

These wallets are *hierarchical* in that each child key d_i could be viewed as a new master private key in its own right, from which a new sequence of child private keys $d_{i,1}, d_{i,2}, \ldots$ could be generated and so on *ad infinitum*.

It is worth emphasizing that the entire hierarchy of private keys in the wallet can be recovered from knowledge of $\hat{d}$, making the wallet highly portable and easy to maintain. (Some Bitcoin users derive this master private key from the hash of a memorized password; the resulting wallet is called a *brain wallet*.)

The master public key property. Interestingly, the mathematics of any discrete logarithm system—including the ECDSA scheme used in Bitcoin—allow for deterministic wallets with the additional property that a user could create and publish a *master public key* $\hat{Q}$, from which anyone could compute the sequence $Q_1, Q_2, \ldots$ of child public keys corresponding to the child private keys $d_1, d_2, \ldots$ derived from $\hat{d}$, and yet knowledge of $\hat{Q}$ alone is insufficient to recover any of the private keys $\hat{d}, d_1, d_2, \ldots$. (See Section3 for details.) We refer to this property as the *master public key property*.

Deterministic wallets with the master public key property are confusingly called *hierarchical deterministic (HD) wallets* in the Bitcoin community. This label is something of a misnomer as the salient feature of HD wallets is not the hierarchy but rather the master public key property.

Credit for this concept is typically attributed to Maxwell [10]. The first widely available HD wallet software was the *Electrum* wallet, which appeared in November 2011 [1]. A specification of HD wallets was proposed in 2012 and subsequently accepted as Bitcoin standard BIP32 [13].

A vulnerability in existing HD wallets. Unfortunately, all existing HD wallets—including BIP32-compliant wallets—admit an exploit whereby an attacker could easily recover the master private key $\hat{d}$ given the master public key $\hat{Q}$ and any child private key d_i. (Again, see Section 3 for details.)

This vulnerability was known to the author of the BIP32 standard [13]. Indeed, BIP32 compensates for this vulnerability by allowing for "hardened" child private keys that can be compromised without also compromising the master private key. Unfortunately, those hardened keys lack the master public key property: their public keys cannot be generated from the master public key.

Buterin calls attention to this vulnerability in his informative article, in which he announces open-source software that cracks BIP32 and Electrum wallets [5]. His pessimism is a challenge to the cryptography community:

> [T]he obvious question is: can this [vulnerability] be fixed? The answer seems to be no; ... If this is indeed true, then raising awareness is the only solution, together with a change in BIP32 representation and in clients to make it clear that master public keys and hierarchical wallets do not mix.

Our contribution. We present a new HD wallet that eliminates this vulnerability while retaining the master public key property (Section 4). For a chosen parameter m, our HD wallet can tolerate the leakage of up to m private keys with a master public key of size $O(m)$ and no blow-up whatsoever in the size of the master private key. We prove in Section 5 that breaking our HD wallet is at least as hard as the so-called "one-more" discrete logarithm problem.

We begin in Section 2 with a survey of several previously known use cases for HD wallets, including one—the combined treasurer-auditor—that is precluded by the vulnerability of existing HD wallets but not by our new HD wallet. Section 3 reviews the BIP32 vulnerability in detail.

Addendum, 2015-Aug-28. After publication of the present manuscript, related work by Courtois, Emirdag, and Valsorda [7] was brought to our attention. Those authors study combination attacks on BIP32 wallets in the presence of bad randomness.

2 Use cases for HD wallets

1. *Low-maintenance wallets, brain wallets.* As mentioned in Section 1, a rudimentary use of deterministic wallets is to allow the complete reconstruction of any Bitcoin wallet from a single master private key. These wallets are easier to maintain, more portable, and make brain wallets possible.
2. *A web merchant receiving payment from customers.* A motivating use case suggested by Maxwell [10] and described in the BIP32 standard [13] is that of a web merchant who generates fresh public keys for each sale. A deterministic wallet allows the merchant to easily generate and store only the *public* keys on a vulnerable online server while all the corresponding *private* keys are kept safe in offline storage.
   Moreover, the merchant can employ the hierarchical property of HD wallets to store only those public keys that are needed for receiving customer payments. This practice could enhance the merchant's privacy by eliminating the need to store *every* public key in his entire wallet on the vulnerable server.
3. *Detailed, trustless audits.* A user could reveal her master public key $\hat{Q}$ to third-party auditors, who could then use that key to view the full details of every subsequent transaction using Bitcoins from the stash associated with

$\hat{Q}$. Such a user is assured that her funds are safe from theft by the auditor because the private keys associated with those funds are never revealed.
One frequently suggested use case is a large company that reveals its master public key to regulators, thereby allowing an extremely detailed degree of oversight with near-negligible overhead costs. Another use case is that of a bank or online wallet service; revealing master public keys to depositors allows the bank to prove to those depositors that their funds are safe and that the bank is not operating a fractional reserve. *Coinkite* is a recent commercial example of such an online HD wallet [3].

4. *A treasurer allocating funds to departments.* The treasurer of a large company could create child key pairs for each department within the company. Department managers are given only the child private key for their department, so they cannot spend the funds allocated to other departments. Managers of large departments can employ the hierarchical property of HD wallets to create their own sub-tree of child keys and allocate funds among middle-managers in a similar fashion, and so on down the corporate hierarchy. Meanwhile, the treasurer, who knows the master private key, retains the full ability to move funds into and out of the different departments. This use case is suggested by, for example, the developers of the open-source *MoneyTree* HD wallet [2].

Treasurers and auditors don't mix. An organization that simultaneously implements the treasurer (4) and auditor (3) use cases using current HD wallets leaves itself open a collusion between the auditor and a department manager, allowing them to run off with all the company's funds via the exploit mentioned in Section 1 and detailed in Section 3.

Enabling the combined treasurer-auditor. Our HD wallet, presented in Section 4, is the first to facilitate the combined treasurer-auditor use case. Specifically, an organization with t departments is safe from a collusion between the auditor and all department managers if it uses our HD wallet with parameter $m > t$.

3 A vulnerability in BIP32 wallets

We now review the exploit in BIP32 Bitcoin HD wallets that allows an adversary to recover the master private key given the master public key and any child private key, thus precluding the combined treasurer-auditor use case discussed above. (This exploit appears to be folklore knowledge in the Bitcoin community.)

The key-generation formula employed by current HD wallets can be applied to any discrete logarithm system, such as the ECDSA used by Bitcoin. It is convenient to present the formula in simplified form using the familiar language of a generic additive group $\mathbb{G}$ of prime order p with generator $P \in \mathbb{G}$.

Recall that a private key d in a discrete logarithm system is an element of $\mathbb{Z}_p$ and the public key $Q \in \mathbb{G}$ corresponding to d is easily computed via $Q = dP$. Recall also that the task of recovering a private key d given only the public key Q and the generator P is the familiar *discrete logarithm problem (DLP)* for $\mathbb{G}$.

BIP32 child key generation. Given a master key pair $(\hat{d}, \hat{Q})$ for a BIP32-compliant HD wallet, compute child private keys $d_1, d_2, \ldots$ and corresponding public keys $Q_1, Q_2, \ldots$ as

$$d_i = \hat{d} + \text{hash}(i, \hat{Q}) \pmod{p}$$
$$Q_i = \hat{Q} + \text{hash}(i, \hat{Q})P$$

for a strong, publicly known hash function $\text{hash} : \mathbb{Z} \times \mathbb{G} \to \mathbb{Z}_p$. It is easily seen that Q_i is indeed equal to $d_i P$ and thus the public key corresponding to d_i.

By contrast with the rudimentary deterministic wallet described in Section 1, child public keys Q_i can be computed using only knowledge of i, the master public key $\hat{Q}$, and the function $\text{hash}(\cdot)$. It is this fact that gives rise to the master public key feature of BIP32 HD wallets.

Exploit. Recovery of the master private key $\hat{d}$ given the master public key $\hat{Q}$, any child private key d_i, and corresponding index i is given by the formula

$$\hat{d} = d_i - \text{hash}(i, \hat{Q}) \pmod{p}.$$

4 A new HD wallet that tolerates key leakage

Master key generation. Instead of one master private key, our HD wallet uses m master private keys $\hat{d}_1, \ldots, \hat{d}_m$ for some reasonably-sized m to be determined by the requirements of the wallet. (For example, in the combined treasurer-auditor use case of Section 2, m must exceed the number t of departments in the organization.) To keep the master private key size down, these master private keys could be generated pseudorandomly with no loss of security using, say, the rudimentary deterministic wallet described in Section 1.

The master public keys $\hat{Q}_1, \ldots, \hat{Q}_m$ corresponding to master private keys $\hat{d}_1, \ldots, \hat{d}_m$ are given, as usual, by $\hat{Q}_i = \hat{d}_i P$. Whereas the master private keys could be generated pseudorandomly to save storage, one cannot simply publish a single "grand master" public key from which users could deduce the master public keys $\hat{Q}_1, \ldots, \hat{Q}_m$ as otherwise one would succeed only in pushing the original vulnerability up from child keys to master keys. Thus, these master public keys must be stored explicitly, incurring a $O(m)$ blow-up in public key size.

Child key generation. We now describe how child keys are derived from these m master keys. To this end let s be some publicly known master seed. This seed could be a universal constant such as 42—the Answer to the Ultimate Question of Life, The Universe, and Everything. Alternately, s might depend upon the wallet—say, a concatenation of the master public keys $\hat{Q}_1, \ldots, \hat{Q}_m$.

In what follows the hash function $\text{hash}(i, s) \mapsto (\alpha_1, \ldots, \alpha_m)$ produces an m-tuple of integers modulo p. As usual, we assume that $\text{hash}(\cdot)$ is publicly known and behaves as a random function. The ith child private key d_i and public key

Q_i in our HD wallet are given by

$$d_i = \sum_{j=1}^{m} \alpha_j \hat{d}_j \qquad \text{and} \qquad Q_i = \sum_{j=1}^{m} \alpha_j \hat{Q}_j.$$

It is clear that child public keys can be computed from the master public keys $\hat{Q}_1, \ldots, \hat{Q}_m$, so that this HD wallet has the desired master public key property.

5 Security of our HD wallet scheme

5.1 Security definitions

If a child private key d_i is compromised then the adversary has learned only a random linear combination of the master private keys. Indeed, even if a *master* private key $\hat{d}_j$ is compromised then the adversary has learned only one out of the m keys needed to generate child keys. Intuitively speaking, in either case the breach is a linear constraint that reduces by at most one the dimension of the space of all possible master private key combinations; it seems that the adversary gains no useful information about any other master or child private key.

However, if m private keys are leaked then with overwhelming probability the adversary could recover all the master private keys by solving the corresponding linear system, in which case our HD wallet is completely broken. At best, then, our HD wallet is secure only if fewer than m private keys are leaked.

Within this context there is a wide spectrum of possible security definitions for an HD wallet. For example, a very strong definition of security might require that an adversary who obtains some combination of fewer than m master and child private keys cannot forge a signature for any uncompromised master or child key, even under an adaptive chosen-message attack. Since there is no known proof that the ECDSA (or finite field DSA) scheme is existentially unforgeable under chosen message attack, it is reasonable to consider somewhat weaker definitions.

Another security definition might require that an adversary who obtains fewer than m master and child private keys cannot recover any uncompromised master or child private key. We suspect that a security proof for our HD wallet could be obtained by reducing some variant of DLP to the task of breaking our HD wallet according to this security definition. However, it is likely that the variant of DLP in such a reduction would be new and contrived. (For a discussion of the dangers of basing the security of a cryptosystem upon the presumed intractability of contrived problems the reader is referred to Koblitz and Menezes [9] and references therein.)

For the purpose of this preliminary short paper, then, we content ourselves with a proof of security of our HD wallet against a *complete break*, in which an adversary who obtains fewer than m master and child private keys is able to recover *all* of the master private keys. Specifically, the problem of completely breaking our HD wallet is formalized as follows in Problem 1.

Problem 1 (Complete break of our HD wallet).

Input:	(i) Master public keys $\hat{Q}_1, \ldots, \hat{Q}_m$, and (ii) an oracle that on input $\alpha_1, \ldots, \alpha_m \in \mathbb{Z}_p$ returns $k = \sum_{j=1}^{m} \alpha_j \hat{d}_j \pmod{p}$.
Restriction:	The number of calls made to the oracle must be less than m.
Output:	The master private keys $\hat{d}_1, \ldots, \hat{d}_m$.

5.2 Cryptographic assumptions

We will prove that a complete break of our HD wallet is at least as hard as the so-called *one-more discrete logarithm problem (1MDLP)* defined as follows.

Problem 2 (One-more discrete logarithm (1MDLP) for generator $P \in \mathbb{G}$).

Input:	(i) A *challenge* oracle that produces a random $Q_i \in \mathbb{G}$ when queried, and (ii) an oracle for DLP.
Restriction:	Let m be the number of calls made to the challenge oracle. The number of calls made to the DLP oracle must be less than m.
Output:	The discrete logarithms of all elements $Q_1, \ldots, Q_m$. That is, elements $d_1, \ldots, d_m$ of $\mathbb{Z}_p$ with $Q_i = d_i P$.

Although not as "natural" a problem as DLP, 1MDLP is arguably still a natural and clean mathematics problem. 1MDLP has appeared in prior literature—it was used, for example, by Bellare and Palacio to argue the security of the well known GQ and Schnorr identification schemes [4]. Indeed, 1MDLP has even been the subject of at least a bare minimum of scrutiny by the cryptographic community—again, see Koblitz and Menezes [9].

1MDLP is obviously no harder than DLP and there is some evidence suggesting that it is strictly easier, at least in some cases [8]. Nonetheless, it seems reasonable to assume that if DLP is intractable then so too is 1MDLP.

A word of caution. Although it may be reasonable to assume that 1MDLP is intractable in an asymptotic sense, it does not necessarily follow that an attacker could not efficiently solve 1MDLP for the specific choice of parameters used in real-world cryptosystems.

For example, the elliptic curve parameters in the secp256k1 standard used by Bitcoin are chosen so that the best known algorithms for DLP on the elliptic curve group take approximately 2^{128} steps [6]. However, it is conceivable that 1MDLP with these parameters could be solved in, say, only 2^{64} steps.

Such a solution to 1MDLP would not necessarily imply a complete break of our HD wallet because our security reduction is only unidirectional. Nevertheless, it would seriously call into question the security of our HD wallet with the secp256k1 parameters used by Bitcoin; a new security proof would be required.

5.3 Security proof

Theorem 1. *A complete break of our HD wallet (Problem 1) is at least as hard as 1MDLP (Problem 2).*

Proof. Suppose we have an oracle that completely breaks our HD wallet for some number m of master private keys. This oracle can be used to solve 1MDLP with m queries to the challenge oracle as follows. First, query the challenge oracle m times to get m random group elements $\hat{Q}_1, \ldots, \hat{Q}_m$, which are passed as input to the oracle that completely breaks our HD wallet (Problem 1).

Calls by the oracle for Problem 1 to its input oracle on input $(\alpha_1, \ldots, \alpha_m)$ are simulated by querying the DLP oracle on $\alpha_1\hat{Q}_1 + \cdots + \alpha_m\hat{Q}_m$ to obtain the required k. By assumption, the oracle for Problem 1 makes fewer than m such calls, so our reduction obeys the restriction of Problem 2. Also by assumption, the oracle for Problem 1 returns the master private keys $\hat{d}_1, \ldots, \hat{d}_m$, which is a correct solution to 1MDLP. □

Remark 1. In Problem 1 the adversary is granted the luxury to *choose* the linear combination of master private keys revealed to him. In contrast, by compromising a child key in our HD wallet the adversary learns only a *random* linear combination of master private keys. Thus, our security proof holds even against adversaries who can somehow control the randomness used in deriving child private keys.

Acknowledgements

Research at the Perimeter Institute is supported by the Government of Canada through Industry Canada and by the Province of Ontario through the Ministry of Research and Innovation. GG also acknowledges support from CryptoWorks21. DS is supported by Australian Research Council (ARC) Discovery Project DP130104304.

References

1. Electrum lightweight Bitcoin wallet. `https://electrum.org/` (November 2011)
2. Moneytree. `https://github.com/BitVault/money-tree`, `https://bitcointalk.org/index.php?topic=296139` (2013)
3. Coinkite. `https://coinkite.com` (2014)
4. Bellare, M., Palacio, A.: GQ and Schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks. In: Advances in Cryptology – CRYPTO 2002. LNCS, vol. 2442, pp. 162–177 (2002)
5. Buterin, V.: Deterministic wallets, their advantages and their understated flaws. Bitcoin Magazine (November 2013), `http://bitcoinmagazine.com/8396/deterministic-wallets-advantages-flaw/`
6. Certicom Research: SEC 2: Recommended Elliptic Curve Domain Parameters, v2.0 (2000), `http://www.secg.org/`
7. Courtois, N.T., Emirdag, P., Valsorda, F.: Private key recovery combination attacks: On extreme fragility of popular bitcoin key management, wallet and cold storage solutions in presence of poor rng events. Cryptology ePrint Archive, Report 2014/848 (2014), `http://eprint.iacr.org/`
8. Koblitz, N., Menezes, A.: Another look at non-standard discrete log and Diffie–Hellman problems. Journal of Mathematical Cryptology 2(4), 311–326 (2008)

9. Koblitz, N., Menezes, A.: Intractable problems in cryptography. In: Proceedings of the 9th Conference on Finite Fields and Their Applications. Contemporary Mathematics, vol. 518, pp. 279–300 (2010)
10. Maxwell, G.: Deterministic wallets. https://bitcointalk.org/index.php?topic=19137 (June 2011)
11. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008), https://bitcoin.org/bitcoin.pdf
12. National Institute of Standards and Technology: FIPS-186-4: Digital Signature Standard (DSS) (July 2013), http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf
13. Wuille, P.: BIP32: Hierarchical Deterministic Wallets (February 2012), https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki