A Two-Party Hierarchical Deterministic Wallets in Practice

ChihYun Chuang[1], IHung Hsu[1] and TingFang Lee[2]
[1]*AMIS, Taipei, Taiwan*
[2]*Department of Population Health, NYU Grossman School of Medicine, New York, U.S.A.*

Keywords: Secure Two-Party Computation, Cryptocurrencies, Key Derivation, Key Generation, Wallets.

Abstract: The applications of Hierarchical Deterministic Wallet are rapidly growing in various areas such as cryptocurrency exchanges and hardware wallets. Improving privacy and security is more important than ever. In this study, we proposed a protocol that fully support a two-party computation of BIP32. Our protocol, similar to the distributed key generation, can generate each party's secret share, the common chain-code, and the public key without revealing a seed and any descendant private keys. We also provided a simulation-based proof of our protocol assuming a rushing, static, and malicious adversary in the hybrid model. Our master key generation protocol produces up to total of two bit leakages from a honest party given the feature that the seeds will be re-selected after each execution. The proposed hardened child key derivation protocol leads up to a one bit leakage in the worst situation of simulation from a honest party and will be accumulated with each execution. Fortunately, in reality, this issue can be largely mitigated by adding some validation criteria of boolean circuits and masking the input shares before each execution. We then implemented the proposed protocol and ran in a single thread on a laptop which turned out with practically acceptable execution time. Lastly, the outputs of our protocol can be easily integrated with many threshold sign protocols.

1 INTRODUCTION

Blockchains are well-known for their role in cryptocurrency systems, where they provide a secure and decentralized record of transactions without requiring a trusted third party. Digital signatures are used to authorize and validate transactions in which users must protect their private keys. If a private key is lost, the user loses all assets that are associated with it. This makes key management crucial. However, managing multiple private keys is challenging with the growing demand of blockchains. Hierarchical Deterministic (abbrev. HD) Wallet (Wuille, 2013) and $t-n$ threshold signature scheme (abbrev. TSS) are current two methods of managing private keys in blockchains. HD wallet offers convenience, and TSS provides greater security than a single party scheme. The goal of this paper is to employ the advantages of these methods, as previously commented in (Gennaro and Goldfeder, 2018, Section 6.4).

HD wallets were introduced in the blockchain world to simplify private key management. With HD wallets, users utilize deterministic algorithms described in BIP32 (Wuille, 2013), a Bitcoin Improvement Proposal, to generate multiple private keys. Ideally, they generate a set of random words called "seed" S, and then use HMAC-SHA512("Bitcoin seed", S) to derive the master private key k. Next, users derive the new private key, which is $k+$ HMAC-SHA512(chaincode,k,keyID). By iterating this process, BIP32 provides a tree structure of private keys generating paths. BIP32 allows users to securely manage and generate a large number of addresses by securing only one random seed.

There are some potential security issues of HD wallets. First, a compromised private key can lead to the loss of all descendant private keys, which becomes a risk with improper ECDSA implementations. For instance, (Breitner and Heninger, 2019) used lattice attacks against ECDSA signatures to demonstrate that repeated or non-uniform generation of the nonce can potentially enable an attacker to compute the long-term signing key. Second, if given a parent public key and any non-hardened child private keys, it is possible to easily recover the parent private key. To address this issue, (Gutoski and Stebila, 2015) proposed a new HD wallet (i.e. different from the formula defined by BIP32) that can tolerate the leakage of up to m private keys, with a parent public key size of $O(m)$.

TSS enables t out of n parties to generate sig-

Chuang, C., Hsu, I. and Lee, T.
A Two-Party Hierarchical Deterministic Wallets in Practice.
DOI: 10.5220/0012138700003555
In *Proceedings of the 20th International Conference on Security and Cryptography (SECRYPT 2023)*, pages 850-856
ISBN: 978-989-758-666-8; ISSN: 2184-7711

natures on behalf of a group. In most TSS implementations, each party holds the same public key and their own secret, also known as a "share". The private key can only be compromised when all t shares are stolen simultaneously. However, TSS also features "proactive refresh" (Ostrovsky and Yung, 1991), which voids shares produced from an old epoch. TSS is secure, robust, and flexible key management schemes that allow for signature generation without needing for private keys (Luis et al., 2019). In practice, users often use multiple addresses to manage their assets, causing management burden for service providers who must manage numerous shares for different addresses. Another popular solution is multi-signature, in which a valid transaction requires multiple keys instead of a single signature from one key. Multi-signature also avoids the risk of a single point of failure and is easier to implement than TSS. However, the main drawback of multi-signature is its significantly higher cost in blockchains.

Current TSS protocols cannot fully support BIP32's key generation protocols. To address this, we employ garbled circuit protocols (Yao, 1986) that allow parties to compute a predetermined function using their inputs while revealing only the output. Various garbled circuits have been proposed to ensure security against malicious adversaries. For efficiency, we employ the DualEx protocol (Huang et al., 2012), an enhanced semi-honest protocol with dual execution against malicious adversaries. It involves conducting two separate runs of a garbled-circuit protocol against a semi-honest adversary with the parties swapping roles, followed by a secure equality test that leaks no more than one bit against a malicious adversary. The challenge of this paper is to design a "TSS-type BIP32 protocol".

(Das et al., 2023) proposed a new and effective derivation mechanism for BIP32's type wallet achieving the threshold setting easily. However, their constructions are not fully supporting the specification of hardened key derivation described in the original BIP32. Unbound Security[1], a pioneer in multi-party computation technology, has developed a two-party HD wallet protocol that utilizes the garbled circuit scheme. Unbound's idea of the master key generation protocol and child key derivations is to produce the results through DualEx that conform to BIP32 standard. To prevent attackers from using incorrect inputs, they added some randomized information such that shares of a key are outputs. Eventually, both parties receive the correct public key and validate their own shares.

In our study, we also apply the DualEx protocol (Huang et al., 2012) to BIP32. Our protocol uses two layers of twist masking r_i and n_i of i-th party to prevent cheating, with the DualEx output $s' := \text{secret} + r_i * n_{1-i} + r_{1-i} * n_i$. Each party then announces $r_i \cdot G$ and n_i sequentially to ensure the correctness of s'. Our protocol's efficiencies for child key derivation are comparable to Unbound's, while Unbound's master key generation is more efficient. However, it is unclear what security model Unbound considered, whereas our protocols were proven to be secure in the hybrid-world.

As far as we know, there have been no formal papers that study two-party BIP32. Our contributions in this work are as follows: 1) We present a protocol that employs two-party computation, including generating a master private key, hardened private keys, and non-hardened private keys, that complies with all BIP32 regulations (Figures 1 and 2); 2) We provide a security proof for our protocol in the hybrid model using simulation-based methods. We utilize the concepts of real/ideal worlds, as described in (Huang et al., 2012), to demonstrate that the outputs of the two worlds are indistinguishable. Our master key generation protocol allows attackers to use up to two arbitrary boolean circuit functions, costing up to a two-bit leakage from the honest party in the ideal world. However, this is not a significant security concern because the seeds are re-selected after each execution, preventing the leakage from accumulating. In contrast, our hardened child key derivation protocol permits attackers to use only one arbitrary circuit function, resulting in up to a one-bit leakage from the honest party that accumulates with each execution. Nevertheless, this issue can be mitigated considerably by adding some circuit function validation criteria and masking the input shares before each execution. The leakage of bits in the ideal world is the worst-case scenario, and can be prevented in the real world, as discussed in (Huang et al., 2012); 3) We implement the proposed protocol running in a single thread on a laptop, without the need of downloading boolean circuits, as each execution uses the same boolean circuits. The average time spent for the master key generation is 7.37s, and for the hardened child key derivation it is 2.74s. The efficiency can be achieved if network bandwidth allows. However, the number of executions required to generate keys or shares for individual users is usually less than number of signings. The master key generation and the translation of the child key only need to be performed once in most cases. Therefore, a slightly longer execution time is acceptable and practical. More importantly, the outputs of our protocol can be easily integrated with many sign protocols (Canetti et al., 2020a; Castagnos et al., 2020;

[1]https://github.com/unboundsecurity/blockchain-crypto-mpc

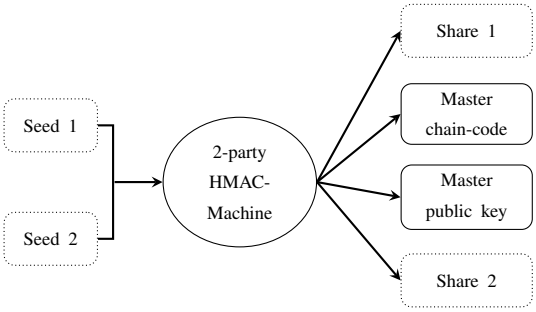

Figure 1: BIP32: 2-Party Master Key Generation.

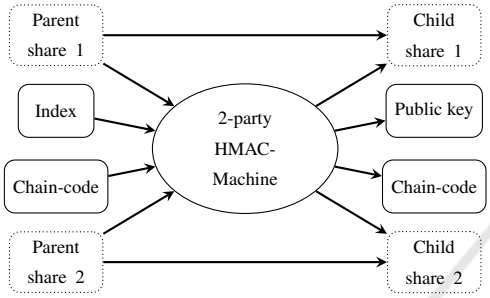

Figure 2: BIP32: 2-Party Child Key Derivation.

Doerner et al., 2018; Gennaro and Goldfeder, 2018; Gennaro and Goldfeder, 2020; Lindell, 2021; Pettit, 2021; Xue et al., 2021) to generate signatures.

2 PRELIMINARIES

Notation. The sets of natural numbers, integers are denoted by $\mathbb{N}$, $\mathbb{Z}$ respectively. Given a prime integer $p \in \mathbb{N}$, we define $\mathbb{Z}_p$ to be the quotient field of $\mathbb{Z}/p\mathbb{Z}$, and $\mathbb{Z}_p^\times$ to be the multiplicative group of $\mathbb{Z}_p$. Let E be an elliptic curve over $\mathbb{Z}_p$, and G be a point of order q on the curve E. The point $b \cdot G := \overbrace{G + \cdots + G}^{b-\text{times}}$ is the scalar multiplication of E by a non-negative integer b. In the rest of this paper, we will use the commonly used elliptic curve defined by secp256k1 [2] in which q is a prime. Given m, $n \in \mathbb{N}$, we let $\text{ser}_m(n)$ serialize a m-bit unsigned integer n as a $\left\lceil \frac{m}{8} \right\rceil$-byte sequence, most significant byte first, $\text{parse}_m(n)$ interpret a $\left\lceil \frac{m}{8} \right\rceil$-byte sequence n as a m-bit number, most significant byte first, and $\text{ser}_P(Q)$ serialize the coordinate pair $Q = (x, y)$ as a byte sequence using SEC1's compressed form. Denote H to be a hash function and HMAC-SHA512 to be the function specified in RFC 4231(M., 2005). Lastly, the notation $\oplus$ is the XOR operation and $\|$ is the concatenation.

[2] https://csrc.nist.gov/publications/detail/sp/800-57-part-1/rev-5/final

2.1 Circuit Garbling

A boolean circuit f can be parameterized by $(n, m, \ell, \text{Gates})$ with a chosen topological order, where n is the number of input, m the number of output, ℓ the number of all wires, and Gates the set of all gates. Circuit garbling (Yao, 1986) allows that two semi-honest parties use their own secrets to evaluate a prior agreed boolean circuit function without leaking any information from their inputs beyond what is revealed by the function output itself. A standard definition of garbling scheme is proposed by (Bellare et al., 2012) as follows:

Definition 1. *A garbling scheme consists of four algorithms* (Zahur et al., 2015, Section 2)*:*

GB*: On input* $\mathbf{1}^k$ *and a boolean circuit* f*, outputs* (F, e, d)*. Here* F *is a garbled circuit,* e *is encoding information, and* d *is decoding information.*

En*: On input* (e, x)*, where* e *is as above and* x *is an input suitable for* f*, outputs a garbled input* X*.*

Ev*: On input* (F, X) *stated as above, outputs a garbled output* Y*.*

De*: On input* (d, Y) *stated as above, outputs a plain output* y*.*

The algorithms have the following conditions:

Correctness*: For any circuit* f *and input* x*, and sampling* $(F, e, d) \leftarrow \mathbf{GB}(\mathbf{1}^k, f)$*, we have*

$$\mathbf{De}\left(d, \mathbf{Ev}\left(F, \mathbf{En}(e, x)\right)\right) = f(x).$$

Privacy*: There exists a simulator* S *that takes input* $(\mathbf{1}^k, f, f(x))$ *and whose output is indistinguishable from* (F, X, d) *generated the usual way.*

Obliviousness*: There must exist a simulator* S *that takes input* $(\mathbf{1}^k, f)$ *and whose output is indistinguishable from* (F, X) *generated the usual way.*

Authenticity*: Given input* (F, X) *alone, no adversary should be able to produce* $\tilde{Y} \neq \mathbf{Ev}(F, X)$ *such that* $\mathbf{De}(d, \tilde{Y}) \neq \perp$*, except with negligible probability.*

2.2 Oblivious Transfer

Oblivious transfer (abbrev. OT) is a fundamental cryptographic primitive in the multi-party computation. To be more specific, 1-*n* OT protocol has two characters called *sender* and *receiver* such that the receiver asks for one of the sender's n messages, which satisfies two conditions: the receiver does not know the other messages except for the selected message, and the sender does not know the receiver's choice.

So far, OT has become the most widely used building block in the two-party setting, such as garbling scheme.

3 OUR SCHEMES

We describe the one proposed protocol, 2-party master key generation(abbrev. 2P-MKG). Another protocol harden key derivation appears in the extended version. The 2P-MKG allows that while both parties do not know each other's seed s_i, they can receive their own secret share $x_{\text{mas},i}$, the chain-code I_R, and the corresponding public key Q such that

- HMAC-SHA512("Bitcoin seed", $s_0 \oplus s_1$) $= \mathrm{I}_L \| \mathrm{I}_R$,
- $x_{\text{mas},0} + x_{\text{mas},1} = \text{parse}_{256}(\mathrm{I}_L) \mod q$,
- $\text{parse}_{256}(\mathrm{I}_L) \in (0,q)$, and $Q = (x_{\text{mas},0} + x_{\text{mas},1}) \cdot G$.

3.1 A Protocol of Two-Party Master Key Generation

Let k and k' be positive integers and $\mathbb{Z}_{q,k}^{\times}$ be the subset of the finite field $\mathbb{Z}_q^{\times}$ with the bit-length of all elements at most k. Consider the following boolean circuits with security parameters $k < 255$ and k'.

$f_{\text{mkg}} \colon \{0,1\}^{k'} \times \mathbb{Z}_q^{\times} \times \mathbb{Z}_{q,k}^{\times} \times \{0,1\}^{k'} \times \mathbb{Z}_q^{\times} \times \mathbb{Z}_{q,k}^{\times} \to \{0,1\}^{513+k}$

$$s_0, r_0, n_0, s_1, r_1, n_1 \mapsto \text{ser}_{256}(\text{parse}_{256}(\mathrm{I}_L) + r_0 n_1 + r_1 n_0 \bmod q) \| \mathrm{I}_R \| \text{ser}_{256}(n_0 + n_1) \quad (1)$$

and $f_{\text{aux}} \colon \{0,1\}^{k'} \times \mathbb{Z}_q^{\times} \times \{0,1\}^{k'} \times \mathbb{Z}_{q,k}^{\times} \to \{0,1\}^{257}$

$$s_0, r_0, s_1, n_1 \mapsto \mathbf{1}_{[0,q)}(\text{parse}_{256}(\mathrm{I}_L)) \| \text{ser}_{256}(\text{parse}_{256}(\mathrm{I}_L) + r_0 n_1 \bmod q). \quad (2)$$

Here $\mathbf{1}_{[0,q)}(x)$ is the indicator function of the interval $[0,q)$. Since k and k' are fixed in the beginning, we omit the notation k and k' in functions f_{mkg} and f_{aux} for simplicity. Before introducing our protocol and theorem, we explain our idea briefly.

Idea. 2P-MKG makes sure that the summation of obtained secret shares, $x_{\text{mas},0}$ and $x_{\text{mas},1}$, is equal to the master private key $s := \text{parse}_{256}(\mathrm{I}_L)$ in which the public key is $s \cdot G$, under the circumstance that neither of them know the master private key and each other's private share. Both parties can access $f_{\text{aux}}(s_0, r_0, s_1, n_1)$ through a garbled circuit to reach an agreement of the public key. During this process, a malicious adversary might attempt to input $n_1 = 0$ in (2) for getting the private key. This issue is resolved by directly sending one non-zero input-wire label of n_1 (ref. step 1 in Protocol 1). Next, both parties use DualEx protocol to verify their outputs of the garbled circuit f_{mkg} are the same. At this stage, it is very hard for malicious adversaries pass a series of the checking lists through non-consistent inputs without knowing the another party's s_i, r_i, and n_i. These processes can ensure the consistency of garbled circuit inputs and the correctness of the secret shares.

Let $f = (n, m, \ell, \text{Gates})$ be a boolean circuit. A participant $\mathcal{P}_i$ applies the algorithm $\mathbf{GB}(\mathbf{1}^k, f)$ to output (F_i, e_i, d_i) (resp. $\mathbf{GB}(\mathbf{1}^k, f_*)$ to $(F_{*,i}, e_{*,i}, d_{*,i})$), where $i \in \{0,1\}$. We denote the input-wire matrix

$$\mathbb{X}_{F_i} = \begin{bmatrix} X_{F_i}^0[0] & X_{F_i}^0[1] & \dots & X_{F_i}^0[n-1] \\ X_{F_i}^1[0] & X_{F_i}^1[1] & \dots & X_{F_i}^1[n-1] \end{bmatrix}.$$

Here $X_{F_i}^b[k]$ is the k-th input-wire label representing the bit value b. If $v \in \{0,1\}^n$ is a bit-string (i.e. $v = v_0 \| v_1 \| \dots \| v_{n-1}$), then we set $\mathbb{X}_{F_i}^v := (X_{F_i}^{v_1}[0], \dots, X_{F_i}^{v_{n-1}}[n-1])$ which also writes this vector to be $X_{F_i}^v$. Moreover, if v' is a sub-string of v, which means $v' = v_t \| v_{t+1} \| \dots \| v_{t+k-1}$ with k the bit-Length of v', then we set $\mathbb{X}_{F_i}^{v'} := (X_{F_i}^{v_t}[t], \dots, X_{F_i}^{v_{t+k-1}}[t+k-1]) = X_{F_i}^{v'}$. Set $\mathbb{X}_{F_i}(0)$ (resp. $\mathbb{X}_{F_i}(1)$) to be the participant $\mathcal{P}_0$'s (resp. $\mathcal{P}_1$'s) input-wire matrix. Given two $d_1 \times d_2$ matrices $A = [a_{ij}]$, $B = [b_{ij}]$, we define

$$A \| B = \begin{bmatrix} a_{11} & \dots & a_{1d_2} & b_{11} & \dots & b_{1d_2} \\ a_{21} & \dots & a_{2d_2} & b_{21} & \dots & b_{2d_2} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{d_1 1} & \dots & a_{d_1 d_2} & b_{d_1 1} & \dots & b_{d_1 d_2} \end{bmatrix}_{d_1 \times 2d_2}.$$

Therefore, given an input $x = x_0 \| x_1$ with x_j being $\mathcal{P}_i$'s input, one has

$$\mathbf{En}(e_i, x) = \mathbb{X}_{F_i}^x = \mathbb{X}_{F_i}^{x_0} \| \mathbb{X}_{F_i}^{x_1} = \mathbb{X}_{F_i}^{x_0}(0) \| \mathbb{X}_{F_i}^{x_1}(1).$$

For the output-wire matrix

$$\mathbb{Y}_{F_i} := \begin{bmatrix} Y_{F_i}^0[0] & Y_{F_i}^0[1] & \dots & Y_{F_i}^0[m-1] \\ Y_{F_i}^1[0] & Y_{F_i}^1[1] & \dots & Y_{F_i}^1[m-1] \end{bmatrix},$$

we define the hash matrix of $\mathbb{Y}_{F_i}$,

$$\mathrm{H}\mathbb{Y}_{F_i} := \begin{bmatrix} \mathrm{H}(Y_{F_i}^0[0]) & \mathrm{H}(Y_{F_i}^0[1]) & \dots & \mathrm{H}(Y_{F_i}^0[m-1]) \\ \mathrm{H}(Y_{F_i}^1[0]) & \mathrm{H}(Y_{F_i}^1[1]) & \dots & \mathrm{H}(Y_{F_i}^1[m-1]) \end{bmatrix}.$$

- The input of $\mathcal{P}_0$: x_0.
- The input of $\mathcal{P}_1$: x_1.
- The output of $\mathcal{P}_0$ and $\mathcal{P}_1$: if x_0 equals x_1, return 1; otherwise, return 0.

Figure 3: The ideal functionality f_{VT} for Validation Test.

We consider Protocol 1 runs in a hybrid world where the parties are given access to the trusted party

Protocol 1: π_{MKG}: 2-party master key generation 1.

Input: two security parameters k, k' and a seed $s_i \in \{0,1\}^{k'}$.

output: a share $x_{\text{mas},i}$, the corresponding public key Q, and the chain code.

1: Each participant $\mathcal{P}_i$:

- randomly chooses $r_i \in \mathbb{Z}_q^{\times}$ and an odd integer $n_i \in [1, 2^k)$.
- performs **GB** in Definition 1 with f_{mkg} and f_{aux} to obtain $(F_{\text{mkg},i}, e_{\text{mkg},i}, d_{\text{mkg},i})$, and $(F_{\text{aux},i}, e_{\text{aux},i}, d_{\text{aux},i})$.
- computes $\mathbb{X}_{F_{\text{mkg},i}}^{s_i \| \text{ser}_{256}(r_i) \| \text{ser}_k(n_i)} = \mathbf{En}\big(e_{\text{mkg},i}, s_i \| \text{ser}_{256}(r_i) \| \text{ser}_k(n_i)\big)$, $\mathbb{X}_{F_{\text{aux},i}}^{s_i \| \text{ser}_{256}(r_i)} = \mathbf{En}\big(e_{\text{aux},i}, (s_i \| \text{ser}_{256}(r_i))\big)$.
- performs the OT protocol with $\mathcal{P}_{1-i}$ to obtain garbled input-wire labels $\mathbb{X}_{F_{\text{mkg},1-i}}^{s_i \| \text{ser}_{256}(r_i) \| \text{ser}_k(n_i)}$ (resp. $\mathbb{X}_{F_{\text{aux},1-i}}^{s_i \| \text{ser}_k(n_i)}$) except for $X_{F_{\text{mkg},1-i}}^1[\dot{n}_{\text{mkg},i}]$ (resp. $X_{F_{\text{aux},1-i}}^1[\dot{n}_{\text{aux},i}]$), where $\dot{n}_{\text{mkg},i}$ (resp. $\dot{n}_{\text{aux},i}$) is the index of all input-wire labels corresponding to the least significant bit of n_i. Sends $X_{F_{\text{mkg},i}}^1[\dot{n}_{\text{mkg},1-i}]$, $X_{F_{\text{aux},i}}^1[\dot{n}_{\text{aux},1-i}]$, and the first message of OT.

2: Each participant $\mathcal{P}_i$:

- After OT completed, sends the garbled message $(F_{\text{aux},i}, \mathbb{X}_{F_{\text{aux},i}}^{s_i \| \text{ser}_{256}(r_i)}, d_{\text{aux},i})$, and the point $r_i \cdot G$ to the participant $\mathcal{P}_{1-i}$.

3: Each participant $\mathcal{P}_i$:

- interprets the coming point to be R_{1-i}.
- uses **Ev** to get $Y_{F_{\text{aux},1-i}}$ and then applies **De** to obtain the binary string $v_{\text{aux},i}$, interpret $v_{\text{aux},i}$ to be the result of the indicator function $\mathbf{1}_{[0,q)}(\text{I}_L)$ and $w_{\text{aux},i}$.
- denotes $Q := w_{\text{aux},i} \cdot G - n_i \cdot R_{1-i}$ which is the shared public key. If $Q = 0 \cdot G$, then sets Q be an arbitrary point of the elliptic curve E.
- if the indicator function $\mathbf{1}_{[0,q)}(\text{I}_L) = 0$, then assign $(F_{\text{mkg},i}, \mathbb{X}_{F_{\text{mkg},i}}^{s_i \| \text{ser}_{256}(r_i) \| \text{ser}_k(n_i)}, \text{H}\mathbb{Y}_{F_{\text{mkg},i}})$ to be an arbitrary chosen message with the valid length.
- performs the validation test in (Huang et al., 2012, Figure 5) with $\mathcal{P}_{1-i}$ to compare the shared public key Q.

4: Each participant $\mathcal{P}_i$:

- sends $(F_{\text{mkg},i}, \mathbb{X}_{F_{\text{mkg},i}}^{s_i \| \text{ser}_{256}(r_i) \| \text{ser}_k(n_i)}, \text{H}\mathbb{Y}_{F_{\text{mkg},i}})$ to the participants $\mathcal{P}_{1-i}$.

Protocol 1: π_{MKG}: 2-party master key generation 2.

5: Each participant $\mathcal{P}_i$:

- uses **Ev** to get $Y_{F_{\text{mkg},1-i}}$ and the incoming hash table $\text{H}\mathbb{Y}_{F_{\text{mkg},1-i}}$ to obtain the binary string $v_{\text{mkg},i}$. Interpret $v_{\text{mkg},i}$ to be $w_{\text{mkg},i}$, chain-code, and n.
- verifies $w_{\text{mkg},i} = v_{\text{aux},i} + (n - n_i) \cdot r_i \mod q$.
- computes $x_{\text{mas},i} := \dfrac{w_{\text{mkg},i}}{2} - (n - n_i) \cdot r_i \mod q$
- uses $\text{H}W_{f_0,1-i}$ to learn the bit-string v_i.
- computes

$$h_{\text{mkg},i} := \begin{cases} \text{H}\left(\mathbb{Y}_{F_{\text{mkg},i}}^{v_{\text{mkg},i}} \| Y_{F_{\text{mkg},1-i}}\right), & \text{if } i = 0, \\ \text{H}\left(Y_{F_{\text{mkg},1-i}} \| \mathbb{Y}_{F_{\text{mkg},i}}^{v_{\text{mkg},i}}\right), & \text{if } i = 1. \end{cases}$$

- performs the validation test with $\mathcal{P}_{1-i}$ to compare $h_{\text{mkg},i}$, and $h_{\text{mkg},1-i}$. If they are equal, then the protocol is complete. Otherwise, abort.

computing two functionalities: oblivious transfer f_{OT} (Chuang et al., 2023, Figure 5) and validation testing f_{VT} (ref. Figure 3). Then one has the following theorem:

Theorem 1. *Assume that a garbling scheme satisfies correctness, privacy, and obliviousness, a hash function H is modeled as an random oracle, and the discrete logarithm problem is hard, then π_{MKG} is securely computed $f_{\pi_{\text{MKG}}}$ with 2-bit leakage in the hybrid world described above. Here for input $s_0, s_1 \in \{0,1\}^{k'}$, the function $f_{\pi_{\text{MKG}}} = (f_0, f_1)$ with*

$$f_i(s_0, s_1) := \Bigg(\frac{\big(\text{parse}_{256}(\text{I}_L) + (-1)^i \cdot r\big)}{2} \mod q, \\ \text{I}_R, \text{parse}_{256}(\text{I}_L) \cdot G \Bigg) \in \mathbb{Z}_q \times \{0,1\}^{256} \times E,$$

for r uniformly random in $\mathbb{Z}_q$ and $\text{HMAC-SHA512}(\text{“Bitcoin seed”}, s_0 \oplus s_1) = \text{I}_L \| \text{I}_R$.

The correctness and the security proof of our theorem can be found in (Chuang et al., 2023).

4 IMPLEMENTATION, BENCHMARKS, AND EVALUATION

The garbled scheme were generated using Two Halves Make a Whole (Zahur et al., 2015) to enhance the efficacy. We employ OT that proposed by (Canetti et al., 2020b) and optimize the security according to the suggestions in (McQuoid et al., 2021, Section 3.3). In the validation protocol, we let both parties use one-side validation protocol in (Huang

et al., 2012, Figure 5) to verify that both outputs from garbled circuit are the same. All benchmarks without downloading boolean circuits were ran in a single-threaded on an Intel i5 CPU 2.3GHz and 16GB 2133MHz LPDDR3 of RAM in the 13-inch (2018) macbook pro.

Table 1: Basic information of circuits: f_{mkg}, f_{aux}, and f_{ckd}.

	f_{mkg}	f_{aux}	f_{ckd}
Total number of gates	847,420	796,827	507,581
Total number of wires	850,558	799,676	510,463

Table 2: Time consuming for running 20 samples.

	2P-MKG	2P-CKD
Fast time	7.181s	2.688s
Slow time	7.748s	2.839s
Avg time	7.374s $\pm$ 0.14s	2.739s $\pm$ 0.04s

The bit-length of seed = 512, n = 33, and Paillier public key = 2048; The number of base OT = 128; The half-gates scheme uses the Hash function $\widehat{\text{MMO}}^{E}$ (ref. (Guo et al., 2020, Section 7.3)) with the ideal cipher AES-128.

5 CONCLUSION AND DISCUSSION

The proposed protocols comply with all BIP32 regulations. We also provided a security proof in the hybrid model using simulation-based methods and experimental data. Theoretically, there will be risk in leaking bits while using DualEx protocol since we considered the worst scenario without any restrictions on g in the security proof. The leaking risk can be potentially overcame by utilizing garbled circuit schemes against malicious adversaries such as authenticated Garbling method.

REFERENCES

Bellare, M., Hoang, T., and Rogaway, P. (2012). Foundations of garbled circuits. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 784–796. Association for Computing Machinery.

Breitner, J. and Heninger, N. (2019). *Biased Nonce Sense: Lattice Attacks Against Weak ECDSA Signatures in Cryptocurrencies*, pages 3–20. Springer-Verlag.

Canetti, R., Gennaro, R., Goldfeder, S., Makriyannis, N., and Peled, U. (2020a). Uc non-interactive, proactive, threshold ecdsa with identifiable aborts. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1769–1787. Association for Computing Machinery.

Canetti, R., Sarkar, P., and Wang, X. (2020b). *Blazing Fast OT for Three-Round UC OT Extension*, pages 299–327. Springer International Publishing.

Castagnos, G., Catalano, D., Laguillaumie, F., Savasta, F., and Tucker, I. (2020). *Bandwidth-Efficient Threshold EC-DSA*, pages 266–296. Springer International Publishing.

Chuang, C., Hsu, I., and Lee, T. (2023). A two-party hierarchical deterministic wallets in practice (extended version). https://eprint.iacr.org/2023/714.

Das, P., Erwig, A., Faust, S., Loss, J., and Riahi, S. (2023). Bip32-compatible threshold wallets. *Cryptology ePrint Archive*.

Doerner, J., Kondi, Y., Lee, E., and Shelat, A. (2018). Secure two-party threshold ecdsa from ecdsa assumptions. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 980–997.

Gennaro, R. and Goldfeder, S. (2018). Fast multiparty threshold ecdsa with fast trustless setup. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1179–1194. Association for Computing Machinery.

Gennaro, R. and Goldfeder, S. (2020). One round threshold ecdsa with identifiable abort. *Cryptology ePrint Archive*.

Guo, C., Katz, J., Wang, X., and Yu, Y. (2020). Efficient and secure multiparty computation from fixed-key block ciphers. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 825–841.

Gutoski, G. and Stebila, D. (2015). Hierarchical deterministic bitcoin wallets that tolerate key leakage. In *Financial Cryptography*, pages 497–504.

Huang, Y., Katz, J., and Evans, D. (2012). Quid-pro-quotocols: Strengthening semi-honest protocols with dual execution. *Proceedings - IEEE Symposium on Security and Privacy*, pages 272–284.

Lindell, Y. (2021). Fast secure two-party ecdsa signing. *Journal of Cryptology*, 34.

Luis, A., Nicky, M., and Apostol, V. (2019). Threshold schemes for cryptographic primitives.

M., N. (2005). Identifiers and test vectors for hmac-sha-224, hmac-sha-256, hmac-sha-384, and hmac-sha-512. https://datatracker.ietf.org/doc/html/rfc4231.

McQuoid, I., Rosulek, M., and Roy, L. (2021). *Batching Base Oblivious Transfers*, pages 281–310. Springer International Publishing.

Ostrovsky, R. and Yung, M. (1991). How to withstand mobile virus attacks (extended abstract). In *Proceedings of the Tenth Annual ACM Symposium on Principles of Distributed Computing*, PODC '91, pages 51–59. Association for Computing Machinery.

Pettit, M. (2021). *Efficient Threshold-Optimal ECDSA*, pages 116–135. Springer-Verlag.

Wuille, P. (2013). Hierarchical deterministic wallets. https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki.

Xue, H., Au, M. H., Xie, X., Yuen, T., and Cui, H. (2021). Efficient online-friendly two-party ecdsa signature. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 558–573. Association for Computing Machinery.

Yao, A. (1986). How to generate and exchange secrets. In *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*, volume 10, pages 162 – 167.

Zahur, S., Rosulek, M., and Evans, D. (2015). Two halves make a whole. In *Advances in Cryptology - EUROCRYPT 2015*, volume 9057, pages 220–250. Springer Berlin Heidelberg.