The Multi-user Constrained PRF Security of Generalized GGM Trees for MPC and Hierarchical Wallets*

Chun Guo[1 2 3], Xiao Wang[4], Xiang Xie[5], and Yu Yu[6]

[1] School of Cyber Science and Technology, Shandong University, Qingdao, China
[2] Key Laboratory of Cryptologic Technology and Information Security of Ministry of Education, Shandong University, Qingdao, Shandong, 266237, China,
[3] Shandong Research Institute of Industrial Technology, Jinan, Shandong, 250102, China
[4] Northwestern University
[5] Shanghai Qizhi Institute & PADO Labs
[6] Shanghai Key Laboratory of Privacy-Preserving Computing and MatrixElements Technologies
chun.guo.sc@gmail.com,wangxiao@cs.northwestern.edu,xiexiangiscas@gmail.com,yuyu@cs.sjtu.edu.cn

Abstract. Multi-user (mu) security considers large-scale attackers that, given access to a number of cryptosystem instances, attempt to compromise at least one of them. We initiate the study of mu security of the so-called GGM tree that stems from the PRG-to-PRF transformation of Goldreich, Goldwasser, and Micali, with a goal to provide references for its recently popularized use in applied cryptography. We propose a generalized model for GGM trees and analyze its mu *prefix-constrained PRF* security in the random oracle model. Our model allows to derive concrete bounds and improvements for various protocols, and we showcase on the Bitcoin-Improvement-Proposal standard Bip32 hierarchical wallets and function secret sharing (FSS) protocols. In both scenarios, we propose improvements with better performance and concrete security bounds at the same time. Compared with the state-of-the-art designs, our SHACAL3- and KECCAK-p-based Bip32 variants reduce the communication cost of MPC-based implementations by 73.3%∼93.8%, while our AES-based FSS substantially improves mu security while reducing computations by 50%.

* Published at ACM Transactions on Privacy and Security (TOPS, previously known as TISSEC) in 04/2023: https://dl.acm.org/doi/abs/10.1145/3592608. Technical difference between this doc and the published version is minor.

Table of Contents

1 Introduction

GGM tree. Pseudorandom functions (PRFs) and Pseudorandom generators (PRGs) are fundamental building blocks of virtually all cryptosystems. The PRG-to-PRF transformation of Goldreich, Goldwasser, and Micali, dubbed GGM tree [GGM86], was originally proposed to show the theoretical feasibility of constructing PRFs from any one-way function. Interestingly, subsequent works exhibited various "beyond-PRF" security for the GGM tree. Importantly for us, Boneh et al. [BW13,BGI14] showed a *constrained PRF* based on GGM tree with prefix predicates, meaning that it allows to delegate "constrained" keys that can only be used to evaluate the GGM PRF on a subset of inputs with certain prefixes. This can also be used as a *puncturable PRF*, the "dual" of constrained PRFs. These features have motivated extensive practical uses of the GGM: as a constrained PRF in delegatable computations [KPTZ13] and functional signatures [BGI14], as a hierarchical access mechanism in the Bip32

hierarchical deterministic wallets (HDWs), as a puncturable PRF in Function-Secret Sharing (FSS) protocols [BGI15,BGI16]), and as part of the construction of a private puncturable PRF in Pseudorandom Correlation Generator (PCG) [BCGI18,BCG+19b] which supports more sophisticated applications including (RAM-based) secure multi-party computation [Ds17,SGRR19] and zero-knowledge proofs [BMRS21,DIO21].

Multi-user security. The classical (constrained/puncturable) PRF security of a function family F concerns with the behavior of $\mathsf{F}(K,\cdot)$ under a (fixed) secret key K, which is now known as the *single-user (*su*) setting*. Due to the pervasive uses, a large number of GGM tree instances using the same specification and independently chosen keys become available from different entities across the Internet. For example, the Exodus Platform is based on Bip32 and have millions of active users [Exo21]. This effectively creates millions of independently keyed "Bip32 key trees" or generalized GGM trees. As another example, to compute stable marriage (as in [DEs16]), a two-party computation (2PC) protocol (like GMW) needs to evaluate 2^{38} AND gates using 2^{39} OT executions. Using the state-of-the-art PCG-based OT protocol [YWL+20] which need 1295 GGM trees to generate 10^7 OTs, this requires executing FSS and instantiating GGM for $\approx 2^{26}$ times. By these, once different users have conducted such computations 2^{14} times, the total number of GGM instances approaches 2^{40}.

It could be sufficient to corrupt *just one* of the numerous instances: the breach of any of the Bip32 users already incurs severe loss of funds, and the breach of any of the trees in 2PC already breaks necessary (pseudo)randomness in the protocol. This challenges security *in the multi-user (*mu*) setting*, in which for a parameter u representing the maximal number of *users*, there are u independently chosen secret keys $K_1, ..., K_u$, and the adversary succeeds as long as it compromises at least one out of u instances $\mathsf{F}(K_1,\cdot), ..., \mathsf{F}(K_u,\cdot)$ of the target F.

The mu setting was first formalized in public-key cryptography [BBM00]. Asymptotically, it is equivalent to su security: a scheme with κ bits su security trivially ensures $\kappa - \log_2 u$ bits mu security by a standard hybrid argument. But, as discussed, u can be large in practice, and the $\log_2 u$ bits degradation has incurred serious concerns [BT16,GKW+20] and even practical Bitcoin cracking project [LBC16] (see Appendix A). Formally exploring the relation between concrete bounds and various parameters (including u) is technically challenging, as made evident in recent works regarding public-key [BBM00,Ber15,KMP16] and symmetric-key schemes [BBT16,BT16,BHT18,GKW+20], and results have explicitly influenced real world designs [BT16,Ber15].

Despite the raised mu challenge, formal mu constrained or puncturable PRF security treatments of GGM remain missing, and this undermines security of applications. As discussed, trivial bounds are seldom satisfactory. Recall the example of 2PC using 128-bit keys: the trivial bound indicates 88-bit security[7] and falls far below the expected 128 bits. For Bip32, recent work [DEF+21]

[7] This is tight: since each key guess falls in the 2^{40} keys with probability $2^{40}/2^{128}$, with 2^{88} computations or key guesses we can succeed to "hit" one of the 2^{40} keys with probability ≈ 1.

proved 111-bit su security, by which the trivial bound indicates 91-bit mu security as long as 2^{20} Bip32 users are available.

Facing the difficulty in providing dedicated mu security proofs for each application, we seek for unified treatments with full proofs of non-trivial mu bounds, in order to provide a systematic reference for their practical deployments (which is the ultimate goal of cryptography).

1.1 Our Contribution

We provide generalized mu security definitions, a unified model of the GGM tree, and concrete mu security proofs. Improved tree schemes with non-trivial, proven mu bounds can be easily derived from our model, and we showcase on Bip32 and recent FSS protocols. Below we elaborate in detail.

Generalized GGM tree and its mu security. To unify the models in various settings, we extend both the constrained PRF (CPRF) security definition and the GGM scheme. Regarding the former, we start with the fully adaptive constrained PRF security notion of Hofheinz et al. [HKKW19], and extend it along two axes to reach a mu *leakage constrained PRF security* definition in the simulation paradigm:

(i) *Multi-user*: Hofheinz et al.'s (su) CPRF security definition [HKKW19] allows the adversary to adaptively acquire constrained keys and function values, and requires the function value $\mathsf{F}(K, x)$ to be pseudorandom for any x that has never been constrained. Our natural mu extension considers u secret user keys $K_1, ..., K_u$, and requires the function values $\mathsf{F}(K_i, x), \mathsf{F}(K_j, x')$ to be pseudorandom and independent for any two involved user keys K_i, K_j, and for x unconstrained w.r.t. K_i and x' unconstrained w.r.t. K_j;

(ii) *Protocol-level Leakages*: In some applications such as the Bip32, the (secret) intermediate values of the F evaluations are used to derive *public* information and incur protocol-level leakages. To formalize these settings, we augment the real world with a *(context-dependent) leakage oracle* L outputting leakages of the corresponding intermediate values upon every query, and the ideal world with a *leakage simulator* S outputting faked leakages. The detailed leakages depend on the concrete context, and we refer to Sect. 4.1 for example.

Regarding the scheme, we propose a generalized model for GGM trees. Our model is built upon a *public* cryptographic primitive **Prim** (that can be accessed by the adversary), uses κ-bit internal secrets, and allows for multiple branches. To reflect influences of public parameters, every **Prim**-call in the tree has an additional input which we informally called "label". Modeling **Prim** as either a fixed-input-length random oracle (FIL RO) **H** (to justify instantiating **Prim** with cryptographic hash functions) or a Davies-Meyer construction $\mathsf{DM}^{\mathbf{E}}(L, x) = \mathbf{E}(L, x) \oplus x$ based on an ideal cipher **E** (to justify instantiating **Prim** with blockciphers), we analyze the generalized tree w.r.t. our mu leakage

CPRF security definition. We prove $\kappa - \log_2 C$ bits mu CPRF security, where C is a parameter depending on the probability among distinct "label" inputs of the internal **Prim** calls *across all the users*. When all "labels" are the same, security becomes (inferior) $\kappa - \log_2 D$ bits, where D, called *effective data complexity*, is the total number of **Prim**-calls internally made by the u tree instances and is related to the number of adversarial queries to the trees. When collisions among "labels" are unlikely, security becomes nearly optimal $\approx \kappa$ bits. Moreover, the leakage of certain intermediate values are indistinguishable from random "simulated leakages", indicating that protocols can extract (a limited amount of) pseudorandom bits from the intermediate values in the tree and use them in arbitrary.

The random oracle-based trees can be instantiated with truncated Keccak-p permutations of SHA3 [CLL19], while the results on Davies-Meyer-based trees enable instantiations using the compression function of SHA512 or the AES in Davies-Meyer mode. Note that even if $\mathbf{E}$ is ideal, the Davies-Meyer construction $\mathsf{DM}^{\mathbf{E}}$ cannot be modeled as a random oracle [DRST12], and we thus have to appeal for a dedicated analysis, and the proved bounds differ by a factor of 2.

While our security definition, model and provable bounds appear complicated, our analyses complete a large step of proofs for a wide range of GGM tree variants and shed lights on the influences of parameters. To derive bounds for concrete designs, designers just need to fill in the parameters and make some additional counting for the aforementioned C and D. We believe this could help characterize state-of-the-art designs and provide building blocks for the coming NIST standardization [NIS21]. We will showcase on two applications.

We remark that our bounds are proven in the ideal (function or cipher) model, and should be taken as a heuristic insurance for their practical instantiations. This theoretical caveat is shared by similar works [DFL19,ADE+20,DEF+21]. Though, the use of ideal model appears necessary to characterize how local computation (approximated by the number of ideal primitive queries) affects security in the mu setting [BT16,BHT18,GKW+20]. Meanwhile, as noticed in [ST16], standard model proofs fail to yield "realistic" mu security bounds for many symmetric schemes with rekeying [BHT18,GKW+20] (which *is* extensively used in Bip32).[8]

Puncturable PRFs We make a natural step further and consider generalized GGM as a *puncturable pseudorandom function (PPRF)* [KPTZ13,BW13,BGI16]. This is a PRF F such that given an input x and a key K, one can generate a punctured key, denoted $K\{x\}$, which allows evaluating F at every point *except for* x, and does not reveal any information about the function value $\mathsf{F}(K, x)$. The notion was subsequently extended to allow for puncturing multiple inputs [HKW15]. This functionality is the "dual" of CPRFs. In this respect, we extend PPRFs to the *multi-user and leakage* setting, and establish multi-user leakage PPRF security for generalized GGM using the aforementioned CPRF results.

[8] Though, we believe that assuming **Prim** is a weak PRF, inferior leakage security bounds can be proven. This is an interesting open question.

Mu security of Bip32, and improvements. Bip32 hierarchical deterministic wallet (HDW) specifies a GGM tree-based approach to derive a collection of digital signature keys organized under an access hierarchy [Med18]. We refer to Fig. 1 for an overview and Sect. 6.1 for more details. Prior works on such HDWs typically focused on enhancing functionalities or achieving new security notions (see Related Work below). Despite the importance for practical assurance, (a) the hierarchical security of Bip32 was never formally proved; (b) the concrete bounds were never characterized even in the su setting (modulo a concurrent work [DEF+21]: see Related Work below).

Along a parallel though related axis regarding efficiency, the HMAC-SHA512-based Bip32 standard [Med18] consumes a huge number of AND gates and is costly when implemented in the MPC setting (for distributed key management), and improvements were mentioned by Lindell as an open problem [Lin19]. While some ideas appear obvious (e.g., using more efficient hashing instead of HMAC-SHA512), the soundness is unclear due to the lack of formalism and justification.

To address the gap, we consider mu *hierarchical unforgeability* and mu *hierarchical unlinkability* of Bip32: the former guarantee that the collaboration of several "accounts" cannot forge transactions for the other "accounts" (and are thus unable to spend their money), while the latter ensure that in the view of several "accounts", the public signature keys derived by the other "accounts" are pseudorandom and independent. Then, using our leakage CPRF security of the RO-based trees, we prove that with u users, Bip32 achieves (roughly) $\min\{247, 256 - \log_2 u\}$ bits mu hierarchical unlinkability security and $\min\{247, 256 - \log_2 u, f(q_S)\}$ bits mu hierarchical unforgeability security, where $f(q_S)$ is the mu security of the underlying signature scheme for q_S the number of adversarial signing queries. By these, the concrete (even mu) unlinkability bounds of the whole Bip32 system is mostly close to the expectation of its designers (256 bits), and the degradation with u is limited. The concrete unforgeability bounds, however, depend on $f(q_S)$ the mu security of the signature, and this is unavoidable. This emphasizes on *carefully choosing the signature*. Besides, the relation with CPRFs provide cryptographic insights on HDWs including Bip32.

Our formal analysis opens the way to improving Bip32 tree using SHACAL3, the compression function underlying SHA512, and Keccak-$p[800, 11]$, a cryptographic permutation from the SHA3 family. Our proposals reduce the number of AND gates by 73.3%∼93.8% compared with the standard Bip32, and are promising in the context of threshold cryptography and side-channel protections. To demonstrate, we benchmark MPC implementations of our proposals and the standard Bip32, with results in Table 1 indicating expected improvements.

Mu security of trees in FSS, and improvements. FSS is a cryptographic primitive where the client can secretly share a function f to f_1 and f_2 such that each of the function does not reveal the parameters of f. Two servers holding f_i and x can locally evaluate $f_i(x)$ and the scheme ensures that $f(x) = f_1(x) \oplus f_2(x)$. When f is a point function, such construction can be easily used for private information retrieval. Furthermore, the splitting of function f can be

performed in a two-party computation protocol and thus eliminate the need of a client.

As mentioned before, most such applications require a private puncturable PRF, which is built upon a GGM tree. In detail, in PCG, a client (which could be simulated by MPC) samples a key of a puncturable PRF, which is essentially the root of a GGM tree, and then sends the key/root to one of the party. The other party chooses a leaf (in an oblivious manner) and obtains the punctured key enabling evaluating all but that leaf. A popular way to instantiate the GGM tree is to define the length-doubling PRG as $s \mapsto \mathsf{AES}_{fk}(s) \oplus s \| \mathsf{AES}_{fk}(s \oplus 1) \oplus s \oplus 1$ (denote FBTr), where AES_{fk} is the AES using a fixed, publicly known keys fk. However, security characterizations are lacked even in the su setting, leaving gaps in both concrete bounds and design references.

We bridge both gaps. First, regarding security of the aforementioned scheme FBTr, we formally prove (roughly tight) concrete mu puncturable PRF security of $128 - \log_2 d - \log_2 u$ bits, where d is the depth of the tree. These, as discussed, suffer from the (notable) $\log_2 u$ bit degradation.

We then show how to improve. 1) To overcome the $\log_2 u$ degradation, we propose to use *random* IV *as the fixed* AES *key in every FSS protocol instance*, and this improves the mu security to $128 - \log_2 d - 2$ bits which is a good bound since d is typically small in practice. 2) We propose multi-branch generalizations that offers a promising tradeoff with the same mu security, i.e., it saves 50% computations at the expense of 50% larger puncturable keys.

While our analysis does not cover the full FSS, it indicates the trees exhibit no practical security issues. We leave the full characterization for future work.

1.2 Related Work

Regarding CPRFs/PPRFs, a series of works overcame standard model challenges (which is complementary to our goal of providing practical reference) and greatly advance the theory [FKPR14,HKW15,BLW17,HKKW19].

Regarding HDW, Das et al. [DFL19,ADE+20] appeared the first to formalize security of *deterministic wallets* as *unforgeability* and *unlinkability*. Concurrently to ours, they extend their treatment to $\mathsf{Bip32}$ [DEF+21]. Das et al.'s hierarchical unforgeability definition is stronger than ours, as it requires unforgeability of signatures even if the parent chain codes are leaked. This models the settings of "hot wallet breach" and granting chain codes to auditors. Due to this, they have to resort to *rerandomizable signatures* and related-key properties of ECDSA. In contrast, our unforgeability definition assumes parent chain codes secret. Thus our bounds do not cover the "audit" use case [Med18, Use cases]. Moreover, our analysis does not distinguish "hardened" and "non-hardened" derivations (since parent chain codes are always secret). Our model mostly covers security against a subset of malicious offices in an enterprise. On the other hand, our treatments provided non-trivial multi-user security bounds. Thanks to the fine-grained analysis using multi-collisions, our mu unlinkability bound is much better than [DEF+21, Theorem A.1]. In all, our analysis of $\mathsf{Bip32}$ offers another complementary viewpoint. Perhaps more importantly, our main focus

in the generalized GGM tree construction and its results, with Bip32 a mere application.

Luzio et al. provided a systematic study of HDW and a new design Arcula [LFA20]. Their work emphasized more on offering a new proposal with richer functions than characterizing security of existing schemes (i.e., Bip32). In comparison, we emphasize more on schemes already used "in the wild". For a survey of earlier works [TVR16,AGKK19,MPs19], we refer to [DFL19, Sect. 1.3].

1.3 Organization

We establish notations and models in Sect. 2. Our extensions of constrained and puncturable PRF notions are also in Sect. 2. Then, we formally define our general GGM tree model in Sect. 3, and prove the mu leakage CPRF security in Sect. 4. The mu leakage PPRF security is established in Sect. 5. We finally demonstrate applications to Bip32 and FSS in Sect. 6 and Sect. 7 respectively.

2 Preliminaries

Denote by $[i]_m$ the m-bit binary encoding of the non-negative integer i, and by $\perp$ the empty string. Given an n-bit string x and $a \leq n$, denote by $\mathsf{left}_a(x)$ (resp., $\mathsf{right}_a(x)$) the a leftmost (resp., rightmost) bits of x.

To describe Bip32 wallets in Sect. 3 and 6, denote by $\mathbb{G}$ the group of the elliptic curve in use and G its primitive element. Denote by $\mathbb{Z}^+_{|\mathbb{G}|}$ the set $\{1, ..., |\mathbb{G}|-1\}$. The function $\mathsf{int}(X)$ interprets a 256-bit string X as a 256-bit number. The function $\mathsf{ser}_P(sk \cdot G) = \mathsf{ser}_P(pk)$ serializes the coordinate $pk = (e_x, e_y)$ as a bit string in the SEC1's compressed form, i.e., $\mathsf{ser}_P(e_x, e_y) = [2]_8 \| [e_x]_{256}$ when $e_y = 0 \mod 2$, while $\mathsf{ser}_P(e_x, e_y) = [3]_8 \| [e_x]_{256}$ when $e_y = 1 \mod 2$.

Signature schemes. Due to Bip32, we need the definition and security of digital signature schemes.

Definition 1 (Signature Scheme). *A signature scheme* $\mathsf{Sig} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrfy})$ *is a triple of algorithms. The randomized key generation algorithm* KGen *takes as input public parameters* pp *and returns a pair* $(pk, sk) \in \mathcal{K}_{\mathsf{sign}}$ *of public and secret keys. The randomized signing algorithm* Sign *takes as input a secret key* sk *and a message* m *and returns a signature* σ*. The deterministic verification algorithm* Vrfy *takes as input a public key* pk*, a signature* σ*, and a message* m*. It returns 1 (accept) or 0 (reject). We require correctness, i.e.,* $\forall (pk, sk) \leftarrow \mathsf{KGen}(\mathsf{pp})\ \forall m : \mathsf{Vrfy}\big(pk, \mathsf{Sign}(sk, m), m\big) = 1$.

We will reduce the security of the Bip32 wallet to the standard *existential unforgeability under chosen message attacks* (UFCMA) security of the signatures *in the multi-user setting*. For this, we adopt the formalism of Bernstein [Ber15].

Definition 2 (Multi-user UFCMA Security). *A (digital) signature scheme* $\mathsf{Sig} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrfy})$ *is* (u, q_S, t, ε)-muUFCMA *secure, if for any adversary*

$\mathfrak{A}$ *making* q_S *queries to* $\mathsf{Sign}(sk_1,\cdot),...,\mathsf{Sign}(sk_u,\cdot)$ *and running in time* t*, it holds* $\Pr\left[\mathfrak{A}^{\mathsf{Sign}(sk_1,\cdot),...,\mathsf{Sign}(sk_u,\cdot)}(pk_1,...,pk_u) \text{ forges}\right] \leq \varepsilon$*, where the event "forges" means* $\mathfrak{A}$ *outputs a pair* (m^*,σ^*) *such that* $\mathsf{Vrfy}(pk_i,m^*,\sigma^*)=1$ *for some* $i \in \{1,...,u\}$ *and* $\mathfrak{A}$ *never queried* $\mathsf{Sign}(sk_{i'},m^*)$ *for any* i' *with* $sk_{i'}=sk_i$*.*

A naive hybrid argument establishes $\kappa - \log_2 u$ muUFCMA security from κ bit UFCMA security. For the Schnorr and BLS schemes, this multi-user security loss can be overcame in the random oracle model [KMP16,Lac18]—for example, using 256-bit secret keys, the muUFCMA security of Schnorr is of 128 bits according to [KMP16],—or by using the key-prefixing technique [Ber15,Lac18].

Constrained PRF and its Multi-user security. Our formalism of constrained PRFs (CPRFs) basically follows [BW13,KPTZ13,BGI14], which is as follows.

Definition 3 (Constrained PRF). *With key space* $\mathcal{K}$*, domain* $\mathcal{X}$*, and range* $\mathcal{Y}$*, a constrained pseudorandom function for a set system* $\mathbb{S} \subseteq 2^{\mathcal{X}}$ *is a keyed function* F *with an additional constrained key space* $\mathcal{K}_c$ *and four probabilistic polynomial-time algorithms* $(\mathsf{F.KGen},\mathsf{F.Ev},\mathsf{F.Co},\mathsf{F.SubCo},\mathsf{F.CEv})$*:*

- *the key generation algorithm* $\mathsf{F.KGen}$ *returns* $K=(\mathsf{k},\mathsf{pp}) \in \mathcal{K}$*, where* k *is the ordinary secret master key and* pp *is the public parameter;*
- *the (ordinary) evaluation algorithm* $\mathsf{F.Ev}(K,x)$ *(always) outputs* $\mathsf{F}(K,x)$ *for the inputs* $K \in \mathcal{K}$ *and* $x \in \mathcal{X}$*;*
- *the constraining algorithm* $\mathsf{F.Co}(K,\mathcal{S})$ *outputs a constrained key* $K\{\mathcal{S}\} \in \mathcal{K}_c$ *on input a key* $K \in \mathcal{K}$ *and a set* $\mathcal{S} \in \mathbb{S}$*;*
- *the constrained evaluation algorithm* $\mathsf{F.CEv}(K\{\mathcal{S}\},x)$*: on input* $K\{\mathcal{S}\}$ *constraining all points in* $\mathcal{S}$ *and* x*, outputs* $\mathsf{F}(K,x)$ *if* $x \in \mathcal{S}$*, and* $\perp$ *otherwise.*

We will focus on CPRFs built upon a public ideal primitive **Prim**, and write $\mathsf{F}^{\mathbf{Prim}}$ to highlight. For a CPRF $\mathsf{F}^{\mathbf{Prim}}$, we follow [HKKW19] and formalize the adversarial goal as distinguishing the real world oracles $(\mathsf{F}^{\mathbf{Prim}}.\mathsf{Co},\mathsf{F}^{\mathbf{Prim}}.\mathsf{Ev},\mathbf{Prim})$ from the ideal world oracles $(\mathsf{F}^{\mathbf{Prim}}.\mathsf{Co},\mathsf{R},\mathbf{Prim})$ for a random function R. We extend it with *multiple users* and *leakages*, in the *concrete security paradigm*. Before the formal presentation, we first elaborate on the new ingredients.

Multi-user. Let u be the maximal number of users.[9] In the mu setting, the adversarial goal becomes distinguishing the real world $(\mathsf{F}^{\mathbf{Prim}}.\mathsf{Co}(K_1,\cdot),\mathsf{F}^{\mathbf{Prim}}.\mathsf{Ev}(K_1,\cdot),...,\mathsf{F}^{\mathbf{Prim}}.\mathsf{Co}(K_u,\cdot),\mathsf{F}^{\mathbf{Prim}}.\mathsf{E}$ and the ideal world $(\mathsf{F}^{\mathbf{Prim}}.\mathsf{Co}(K_1,\cdot),\mathsf{R}(1,\cdot),...,\mathsf{F}^{\mathbf{Prim}}.\mathsf{Co}(K_u,\cdot),\mathsf{R}(u,\cdot))$ for u independent random keys $\mathbf{K}=(K_1,...,K_u)=\big((\mathsf{k}_1,\mathsf{pp}_1),...,(\mathsf{k}_u,\mathsf{pp}_u)\big)$, where $\mathsf{R}(1,\cdot),...,\mathsf{R}(u,\cdot)$ instantiate u *independent* random functions. To simplify notations, we will use a single oracle $\mathsf{muEv}_{\mathbf{K}}(i,\cdot)$ for the functionality of $\mathsf{F}^{\mathbf{Prim}}.\mathsf{Ev}(K_i,\cdot)$, and a single $\mathsf{muCo}_{\mathbf{K}}(i,\cdot)$ for $\mathsf{F}^{\mathbf{Prim}}.\mathsf{Co}(K_i,\cdot)$.

[9] This is equivalent to the alternative definitions allowing adaptively adding new instances [BBT16,BT16].

Protocol-level leakages. As mentioned, to capture the cases where (secret) intermediate values become public information, we augment the real world oracles $\mathsf{muEv}_{\mathbf{K}}$ and $\mathsf{muCo}_{\mathbf{K}}$ with *a leakage oracle* L. Every time the distinguisher issues a query to either $\mathsf{muEv}_{\mathbf{K}}$ or $\mathsf{muCo}_{\mathbf{K}}$, besides the ordinary response, it obtains leakages due to the corresponding internal computations from L. To highlight, we use the notations $\mathsf{muCo}^{\mathsf{L}}_{\mathbf{K}}$ and $\mathsf{muEv}^{\mathsf{L}}_{\mathbf{K}}$ for these "leaky oracles".

For the ideal world oracles $\mathsf{muCo}_{\mathbf{K}}$ and R to produce consistent outputs, we augment them with a *leakage simulator* S and obtain the "leaky ideal oracles" $\mathsf{muCo}^{\mathsf{S}}_{\mathbf{K}}$ and R^{S}. Every time the distinguisher issues a query, it obtains both ordinary responses and leakages produced by S. This enables formalizing indistinguishability of the leaky real and ideal worlds, with L and S being parameters. The detailed definitions of L and S depend on the concrete applications and security requirements, and we refer to Sect. 4.1 for example.

While our simulation-based definition seems quite strong, we have provided positive results (Theorems 1 and 2) with *explicit simulators*, and our simulators simply output random leakages, indicating pseudorandomness of (leaked) intermediate values. This is in contrast with simulatability of *side-channel* leakages, the achievability of which remains open [LMO+14].

Effective data complexity. Concrete security of a cryptosystem is qualified by the attack advantage regarding adversaries with data and time complexities. In the ideal model, time complexity is typically captured by T the number of queries to the ideal function $\mathbf{Prim}$. Data complexity shall reflect the amount of information gained from the oracles $\mathsf{muCo}^{\mathsf{L}}_{\mathbf{K}}$ and $\mathsf{muEv}^{\mathsf{L}}_{\mathbf{K}}$, which is a complicated function of the true number of adversarial queries. To remedy, we follow [DRST12] and introduce *effective data complexity* D, which is the total number of queries to $\mathbf{Prim}$ internally made by $\mathsf{F}^{\mathbf{Prim}}$ during the interaction. As will be seen in Sect. 6 and 7, it is easy to count D for concrete applications. The output of every internal $\mathbf{Prim}$ query, including the ordinary output and the corresponding leakage, will be a part of the information collected by the distinguisher. Therefore, effective data complexity does measure the amount of information gained from $\mathsf{muCo}^{\mathsf{L}}_{\mathbf{K}}$ and $\mathsf{muEv}^{\mathsf{L}}_{\mathbf{K}}$.

Formal definition. With the above, our formal definition is as follows.

Definition 4 (Multi-user Leakage CPRF Security). *The keyed function* $\mathsf{F}^{\mathbf{Prim}}$ *is a* (u, T, D, ε)-(L, S)-constrained PRF*, if for any distinguisher* $\mathfrak{D}$ *making* T *queries to* $\mathbf{Prim}$ *and having effective data complexity* D*, we have*

$$\left|\Pr\left[\mathfrak{D}^{\mathsf{muCo}^{\mathsf{L}}_{\mathbf{K}},\mathsf{muEv}^{\mathsf{L}}_{\mathbf{K}},\mathbf{Prim}} = 1\right] - \Pr\left[\mathfrak{D}^{\mathsf{muCo}^{\mathsf{S}}_{\mathbf{K}},\mathsf{R}^{\mathsf{S}},\mathbf{Prim}} = 1\right]\right| \leq \varepsilon,$$

where the probability is taken over the u *user keys* $\mathbf{K} = (K_1, \ldots, K_u)$ *with* $K_i \leftarrow \mathsf{KGen}$*, over* $\mathfrak{D}$*'s random tape and the ideal primitive* $\mathbf{Prim}$*, and where:*

(i) $\mathsf{muCo}^{\mathsf{L}}_{\mathbf{K}}(i, \mathcal{S})$*: for* $1 \leq i \leq u$*, outputs the constrained key* $\mathsf{F}^{\mathbf{Prim}}.\mathsf{Co}(K_i, \mathcal{S})$ *and the corresponding information leakage* $\mathsf{L}(K_i, \mathcal{S})$ *for the* i*-th user;*

(ii) $\mathsf{muEv}^{\mathsf{L}}_{\mathbf{K}}(i,x)$*: for* $1 \le i \le u$*, outputs* $\mathsf{F}^{\mathbf{Prim}}.\mathsf{Ev}(K_i,x)$ *and the corresponding information leakage* $\mathsf{L}(K_i,x)$ *for the* i*-th user;*

(iii) $\mathsf{muCo}^{\mathsf{S}}_{\mathbf{K}}(i,\mathcal{S})$*: for* $1 \le i \le u$*, outputs the constrained key* $\mathsf{F}^{\mathbf{Prim}}.\mathsf{Co}(K_i,\mathcal{S})$ *and the corresponding simulated leakage* $\mathsf{S}(i,\mathcal{S})$ *for the* i*-th user;*[10]

(iv) $\mathsf{R}^{\mathsf{S}}(i,x)$*: for* $1 \le i \le u$*, outputs* $y \xleftarrow{\$} \mathcal{Y}$ *for every new pair of inputs* (i,x) *and the corresponding simulated leakage* $\mathsf{S}(i,x)$*.*

$\mathfrak{D}$ *is not allowed to makes a constraining query* $\mathsf{muCo}^{\mathsf{L}}_{\mathbf{K}}(i,\mathcal{S})$ *and an evaluation query* $\mathsf{muEv}^{\mathsf{L}}_{\mathbf{K}}(i,x)$ *with* $x \in \mathcal{S}$*,* in any order*, since this leads to trivial win.*

The classical CPRF security notion can be recovered by eliminating the leakages L and S.

Puncturable PRF and its Multi-user security. Depending on its structure, a PPRF scheme may impose certain relations among the multiple punctured inputs. To capture this, we follow Definition 3 and define PPRF for set systems.

Definition 5 (Puncturable PRF). *With key space* $\mathcal{K}$*, domain* $\mathcal{X}$*, and range* $\mathcal{Y}$*, a puncturable pseudorandom function for a set system* $\mathbb{S} \subseteq 2^{\mathcal{X}}$ *is a keyed function* F *with an additional punctured key space* $\mathcal{K}_p$ *and four probabilistic polynomial-time algorithms* $(\mathsf{F.KGen}, \mathsf{F.Ev}, \mathsf{F.Pu}, \mathsf{F.PEv})$*:*

- *the key generation algorithm* $\mathsf{F.KGen}$ *returns* $K = (\mathsf{k}, \mathsf{pp}) \in \mathcal{K}$*, where* k *is the ordinary secret master key and* pp *is the public parameter;*
- *the evaluation algorithm* $\mathsf{F.Ev}(K,x)$ *outputs* $\mathsf{F}(K,x)$ *for the inputs* $K \in \mathcal{K}$ *and* $x \in \mathcal{X}$*;*
- *the puncturing algorithm* $\mathsf{F.Pu}(K,\mathcal{S})$ *outputs a punctured key* $K\{\mathcal{S}\} \in \mathcal{K}_p$ *on input a key* $K \in \mathcal{K}$ *and a set* $\mathcal{S} \in \mathbb{S}$*;*
- *the punctured evaluation algorithm* $\mathsf{F.PEv}(K\{\mathcal{S}\},x)$*: on input* $K\{\mathcal{S}\}$ *punctured at all points in* $\mathcal{S}$ *and* x*, outputs* $\mathsf{F}(K,x)$ *if* $x \notin \mathcal{S}$*, and* $\perp$ *otherwise.*

For a secure PPRF $\mathsf{F}^{\mathbf{Prim}}$, $\mathsf{F}^{\mathbf{Prim}}.\mathsf{Ev}(K,x)$ shall be pseudorandom for any $x \in \mathcal{S}$, even if the punctured key $K\{\mathcal{S}\}$ has been given. This is just the dual of a restricted form of CPRF. For a formal definition in the mu setting, we borrow the ingredients from our Definition 4.

Definition 6 (Multi-user Leakage PPRF Security). $\mathsf{F}^{\mathbf{Prim}}$ *is a* (u,T,D,ε)*-*(L,S)*-*puncturable PRF*, if for any distinguisher* $\mathfrak{D}$ *making* T *queries to* **Prim** *and having effective data complexity* D*, we have*

$$\left|\Pr\left[\mathfrak{D}^{\mathsf{muPu}^{\mathsf{L}}_{\mathbf{K}},\mathsf{muEv}^{\mathsf{L}}_{\mathbf{K}},\mathbf{Prim}}=1\right]-\Pr\left[\mathfrak{D}^{\mathsf{muPu}^{\mathsf{S}}_{\mathbf{K}},\mathsf{R}^{\mathsf{S}},\mathbf{Prim}}=1\right]\right| \le \varepsilon,$$

where the probability is taken over the u *user keys* $\mathbf{K} = (K_1,\ldots,K_u)$*, with* $K_i \leftarrow \mathsf{KGen}$*, over* $\mathfrak{D}$*'s random tape and the ideal primitive* **Prim***, and where:*

[10] This paper focuses on the setting where S does not have access to the **Prim** oracle, which suffices for producing random leakages.

(i) $\mathsf{muPu}^{\mathsf{L}}_{\mathbf{K}}(i,\mathcal{S})$*: for* $1 \le i \le u$*, outputs the punctured key* $\mathsf{F}^{\mathbf{Prim}}.\mathsf{Pu}(K_i,\mathcal{S})$ *and the corresponding information leakage* $\mathsf{L}(K_i,\mathcal{S})$ *for the* i*-th user;*

(ii) $\mathsf{muEv}^{\mathsf{L}}_{\mathbf{K}}(i,x)$*: for* $1 \le i \le u$*, outputs* $\mathsf{F}^{\mathbf{Prim}}.\mathsf{Ev}(K_i,x)$ *and the leakage* $\mathsf{L}(K_i,x)$ *for the* i*-th user;*

(iii) $\mathsf{muPu}^{\mathsf{S}}_{\mathbf{K}}(i,\mathcal{S})$*: for* $1 \le i \le u$*, outputs the punctured key* $\mathsf{F}^{\mathbf{Prim}}.\mathsf{Pu}(K_i,\mathcal{S})$ *and the corresponding simulated leakage* $\mathsf{S}(i,\mathcal{S})$ *for the* i*-th user;*

(iv) $\mathsf{R}^{\mathsf{S}}(i,x)$*: for* $1 \le i \le u$*, outputs* $y \xleftarrow{\$} \mathcal{Y}$ *and simulated leakage* $\mathsf{S}(i,x)$ *for every new pair* (i,x)*.*

The following two restrictions are imposed on the distinguisher's queries:

(a) $\mathfrak{D}$ *is not allowed to make a puncture query* $\mathsf{muPu}^{\mathsf{L}}_{\mathbf{K}}(i,\mathcal{S})$ *if a previous evaluation query* $\mathsf{muEv}^{\mathsf{L}}_{\mathbf{K}}(i,x)$ *with* $x \notin \mathcal{S}$ *is made;*

(b) $\mathfrak{D}$ *is not allowed to make two distinct puncturing queries* $\mathsf{muPu}^{\mathsf{L}}_{\mathbf{K}}(i,\mathcal{S})$ *and* $\mathsf{muPu}^{\mathsf{L}}_{\mathbf{K}}(i',\mathcal{S}')$ *with* $i = i'$*. Namely, each user is punctured (at most) once.*

Restriction (a) is necessary to prevent trivial wins (i.e., query both $\mathsf{muPu}^{\mathsf{L}}_{\mathbf{K}}(i,\mathcal{S})$ and $\mathsf{muEv}^{\mathsf{L}}_{\mathbf{K}}(i,x)$ with $x \notin \mathcal{S}$ to check if the latter matches the result computed from the former). Restriction (b) stems from the nature of PPRFs: note that as the interaction proceeds, the number of $\mathsf{F}.\mathsf{Ev}(K,x)$ that can be computed by $\mathfrak{D}$ *can not decrease*. Thus, if $\mathfrak{D}$ makes two puncturing queries $\mathsf{muPu}^{\mathsf{L}}_{\mathbf{K}}(i,\mathcal{S})$ and $\mathsf{muPu}^{\mathsf{L}}_{\mathbf{K}}(i,\mathcal{S}')$ to the same i-th user/instance, then the later query $\mathcal{S}'$ has to be a subset of $\mathcal{S}$, otherwise the later puncturing request cannot succeed. But it's odd for $\mathfrak{D}$ to make such two puncturing queries. Therefore, we simply forbid $\mathfrak{D}$ puncturing the same user/instance twice.

H-coefficient method. We use Patarin's H-coefficient method [Pat09] to prove security of the trees, and provide a quick overview here. Our presentation borrows heavily from that of [CS14]. Fix a distinguisher $\mathfrak{D}$ that makes at most q queries to its oracles. As in the security definition presented above, $\mathfrak{D}$'s aim is to distinguish between two worlds: a "real world" and an "ideal world". Assume wlog that $\mathfrak{D}$ is deterministic. The execution of $\mathfrak{D}$ defines a *transcript* that includes the sequence of queries and answers received from its oracles; $\mathfrak{D}$'s output is a deterministic function of its transcript. Thus, if $T_{\mathrm{re}}, T_{\mathrm{id}}$ denote the probability distributions on transcripts induced by the real and ideal worlds, respectively, then $\mathfrak{D}$'s distinguishing advantage is upper bounded by the statistical distance $\mathsf{SD}(T_{\mathrm{re}}, T_{\mathrm{id}}) := \frac{1}{2}\sum_{\mathcal{Q}} \left|\Pr[T_{\mathrm{re}} = \mathcal{Q}] - \Pr[T_{\mathrm{id}} = \mathcal{Q}]\right|$, with sum taken over all possible transcripts $\mathcal{Q}$.

Let Θ denote the set of all transcripts that can be generated by $\mathfrak{D}$ in either world. We look for a partition of Θ into two sets Θ_{good} and Θ_{bad} of "good" and "bad" transcripts, respectively, along with a constant $\varepsilon_1 \in [0,1)$ such that

$$\mathcal{Q} \in \Theta_{\mathrm{good}} \Longrightarrow \frac{\Pr[T_{\mathrm{re}} = \mathcal{Q}]}{\Pr[T_{\mathrm{id}} = \mathcal{Q}]} \ge 1 - \varepsilon_1. \tag{1}$$

It is then possible to show (see [CS14]) that the statistical distance is upper bounded by

$$\mathsf{SD}(T_{\mathrm{re}}, T_{\mathrm{id}}) \le \varepsilon_1 + \Pr[T_{\mathrm{id}} \in \Theta_{\mathrm{bad}}]. \tag{2}$$

3 A Framework for Generalized GGM Trees

Before presenting our model in Sect. 3.2, we first serve intuitions in Sect. 3.1 to ease understanding.

3.1 Intuitions

Example 1. The classical GGM tree uses a length-doubling PRG $G : \{0,1\}^n \to \{0,1\}^{2n}$. Thus, every G invocation expands an n-bit intermediate value s into two children, and this constitutes a binary tree. An instantiation of [GKWY20] defines $G^{\mathsf{AES}_{fk}}(s) := \mathsf{DM}^{\mathsf{AES}}(fk, s) \| \mathsf{DM}^{\mathsf{AES}}(fk, s \oplus [1]_{128})$ with $n = 128$, where $\mathsf{DM}^{\mathsf{AES}}(x, y) := \mathsf{AES}_x(y) \oplus y$ and AES_{fk} is the AES using a fixed, publicly known keys fk (though, distinct high-level protocols may use distinct fk). The security of $G^{\mathsf{AES}_{fk}}$ is only justifiable by assuming AES is an ideal cipher. A natural extension is to increase "parallelization degree", i.e., mapping s to θn bits $\mathsf{DM}^{\mathsf{AES}}(fk, s) \| ... \| \mathsf{DM}^{\mathsf{AES}}(fk, s \oplus [\theta - 1]_{128})$ for some integer $\theta \geq 3$. If we view AES as a *public primitive*, then every internal secret s is involved in $\theta \geq 2$ distinct primitive-calls.

Example 2. The $\mathsf{Bip32}$ wallet defines a more sophisticated approach to generate a collection of keys organized in a *key tree* [Med18]. The tree has multiple branches. The 256-bit "chain codes" in [Med18] (the value ch_p in the dashed box in Fig. 1) essentially constitute the internal secret states of the tree. For each such state, the key tree makes $\theta \leq 2^{32}$ calls to $\mathsf{HMAC\text{-}SHA512}$ (henceforth abbreviated as HMAC) to derive θ children. Every HMAC-call has additional complicated inputs, including an index j and a signature key. A half of the HMAC-output (the I values in Fig. 1) will be used to derive the child signature (private and public) key, meaning that it may not be perfectly secret anymore. The default key tree recommended in [Med18] has depth 3, as shown in Fig. 1. If we replace HMAC by a general double-input function **Prim** that accept two inputs of ν and λ bits, then the obtained key tree is depicted in Fig. 1.

Summary. Inspired by the above examples, we would like to have a unified model compatible with flexible choices as follows.

First, we build our model $\mathsf{GGGM}^{\mathbf{Prim}}$ on a public primitive **Prim** functioning as the sub-tree derivation function. In many scenarios the output size of AES may be insufficient (since it may limit the security bound), and "larger" primitives such as $\mathsf{SHA512}$ and $\mathsf{SHA3}$ may be preferred. In this respect, our results should address both the case that **Prim** is a keyed random oracle, i.e., a function that maps each (key,message) pair to an independent and uniform point [DRST12], and the case that **Prim** is the the aforementioned Davies-Meyer mode of an ideal cipher.

Second, to increase design choices, we prefer that all parameters are flexible, including: the size n of the node in the tree, the size κ of the secret which can be viewed as security parameter and "parallelization degree" θ. Moreover, when the output size of **Prim** is long, a single **Prim** invocation may give rise to multiple child nodes, and we denote by w the number of such children.

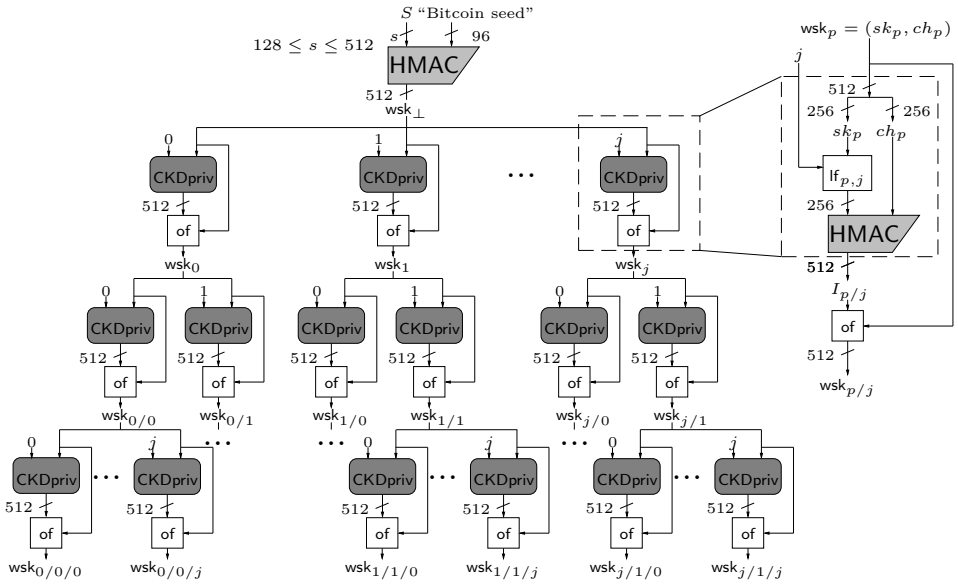

Fig. 1. The key tree in default configuration of Bip32. Due to space, we omit some nodes. S is the seed of the HDW instance, and "Bitcoin seed" is the 96-bit literal string. The values $\mathsf{wsk}_\perp, \mathsf{wsk}_0, \ldots$ are wallet secret keys of Bip32 and will be formalized in Sect. 6.2 and 6.3. The dashed box in the corner shows the internal computations of the function CKDpriv (which will be formally defined in Fig. 6). The functions lf and of will be defined in Eqs. (25) and (26).

Finally, to somewhat separate distinct **Prim** invocations, one may inject "labels" into the inputs to **Prim**; one may also "disturb" the node input by e.g., xoring constants, as the aforementioned use of s and $s \oplus [1]_{128}$ in the two parallel calls [GKWY20]. We model these ideas as two input mappings sf and lf. We will elaborate on the details in the next subsection.

3.2 Our General Tree Model

By the above intuitions, our general model $\mathsf{GGGM}^{\mathbf{Prim}}$ is built upon a public primitive $\mathbf{Prim} : \{0,1\}^{\nu} \times \{0,1\}^{\lambda} \mapsto \{0,1\}^{wn}$ functioning as the sub-tree derivation function, and is formally described in Fig. 2. Also see Fig. 3 for depiction.

In $\mathsf{GGGM}^{\mathbf{Prim}}$, every node represents a string of n bits. Every "internal" node is involved in θ calls to **Prim**, and every **Prim**-call gives rise to w new n-bit node values. Hence, the tree is $w\theta$-branched. Inspired by [Med18], we refer to positions in a tree using "paths" of the form $p = i_1/i_2/.../i_j$, referring to a node at depth j. A node value at "position" $p = i_1/i_2/.../i_j$ is denoted $\mathsf{Nd}(p)$, and $\mathsf{Nd}(\bot) = \mathsf{k}$ denotes the root. By this, when the depth of the tree is d, the domain of $\mathsf{GGGM}^{\mathbf{Prim}}.\mathsf{Ev}$ is $\mathcal{X} = \{i_1/i_2/.../i_d\}_{i_\ell \in \{0,...,w\theta-1\} \text{ for } \ell=1,...,d}$. We further define

- $\mathcal{P}^* := \{\bot\} \cup \{i_1/i_2/.../i_{d'}\}_{d' \in \{1,...,d-1\}, i_\ell \in \{0,...,w\theta-1\} \text{ for } \ell=1,...,d'}$ denoting the set of "incomplete" paths in such trees;
- $\mathcal{P} := \mathcal{X} \cup \mathcal{P}^*$ denoting the set of all valid paths in such trees;
- For any two paths $p, p' \in \mathcal{P}$, p *is prefix of* p', if $p = p'$, or if there exist $i'_1, ..., i'_\ell$ such that $p' = p/i'_1/.../i'_\ell$.

Correspondingly, the prefix set system is

$$\mathbb{S}_{\mathsf{pre},\mathsf{GGGM}} = \{\mathcal{S}_{p^*,\mathsf{GGGM}} : p^* \in \mathcal{P}\}, \text{ with } \mathcal{S}_{p^*,\mathsf{GGGM}} = \{p \in \mathcal{X} : p^* \text{ is prefix of } p\}. \quad (3)$$

Since every set $\mathcal{S}_{p^*,\mathsf{GGGM}} \in \mathbb{S}_{\mathsf{pre},\mathsf{GGGM}}$ has an associated path p^*, in Fig. 2 we adopt the formalism of $\mathsf{GGGM}^{\mathbf{Prim}}.\mathsf{CEv}$ taking a path p^* as the second input.

For a node value $z = \mathsf{Nd}(p)$, $p \in \mathcal{P}^*$, and an index $j \in \{0, 1, ..., \theta - 1\}$, the j-th derivation call of $\mathsf{Nd}(p)$ is $\mathbf{Prim}\big(\mathsf{lf}_{p,j}\big(\mathsf{left}_{n-\kappa}(z), \mathsf{pp}\big), \mathsf{sf}_{p,j}\big(\mathsf{right}_{\kappa}(z)\big)\big)$ (as shown in $\mathsf{StepDown}$, Fig. 2), where the two input map functions sf and lf are such that:

- The *labeling function* $\mathsf{lf}_{p,j}\big(\mathsf{left}_{n-\kappa}(z), \mathsf{pp}\big)$ maps the left most $n - \kappa$ bits of a node value z and the public parameter pp to the ν-bit 1st input for **Prim**. Subsequently, we call this ν-bit value the *label* of the derivation call.
- The *seeding function* $\mathsf{sf}_{p,j}\big(\mathsf{right}_{\kappa}(z)\big)$ is injective, and maps the right most κ bits of a node value z to the λ-bit 2nd input for **Prim**. In some sense, $\mathsf{sf}_{p,j}\big(\mathsf{right}_{\kappa}(z)\big)$ serves as the "secret seed" of the sub-tree, and κ is *security parameter* of the tree;

The output of the **Prim**-call $I = \mathbf{Prim}\big(\mathsf{lf}_{p,j}\big(\mathsf{left}_{n-\kappa}(z), \mathsf{pp}\big), \mathsf{sf}_{p,j}\big(\mathsf{right}_{\kappa}(z)\big)\big)$ does not immediately give rise to children nodes. Instead, an *outputting function* $\mathsf{of}\big(z, \mathsf{left}_n\big(\mathsf{right}_{bn}(I)\big)\big)$ maps the parent node $\mathsf{Nd}(p)$ and the b-th chunk $\mathsf{left}_n\big(\mathsf{right}_{bn}(I)\big)$ to the child node $\mathsf{Nd}(p/b)$.

Algorithm $\mathsf{Ev}(K, p)$
// Function value of $p \in \mathcal{X}$ under the key K
parse $(\mathsf{k}, \mathsf{pp}) \leftarrow K$
return $\mathsf{StepDown}(\mathsf{k}, \mathsf{pp}, p)$

Algorithm $\mathsf{Co}(K, p^*)$
// Constrain a (incomplete) path $p^* \in \mathcal{P}$
parse $(\mathsf{k}, \mathsf{pp}) \leftarrow K$
$K\{p^*\} \leftarrow \big(\mathsf{StepDown}(\mathsf{k}, \mathsf{pp}, p^*), \mathsf{pp}\big)$
return $K\{p^*\}$

Algorithm $\mathsf{CEv}(K\{p^*\}, p)$
// Evaluate $\mathsf{Ev}(K, p)$ using constrained $K\{p^*\}$
parse $(node, \mathsf{pp}) \leftarrow K\{p^*\}$
if $p^* = p$ **then return** $node$
if $\exists i_{d'+1}, ..., i_d$ s.t. $p = p^*/i_{d'+1}/.../i_d$ **then**
  return $\mathsf{StepDown}(node, \mathsf{pp}, i_{d'+1}/.../i_d)$
return $\perp$

Algorithm $\mathsf{StepDown}(node, \mathsf{pp}, p^*)$
// Starting from $node$ and stepping down along the path p^* in the tree, $|node| = n$
if $p^* = \perp$ **then return** $node$
parse $i_1/i_2/.../i_{d'} \leftarrow p^*$
$p \leftarrow \perp$, $\mathsf{Nd}(p) \leftarrow node$
for $j = 1, ..., d'$ **do**
  $b \leftarrow i_j \mod w$, $a \leftarrow \lfloor \frac{i_j}{w} \rfloor$
  // Thus $aw + b = i_j$, $a \in \{0, ..., \theta - 1\}$,
  // $b \in \{0, ..., w - 1\}$
  $I \leftarrow$
Prim$\Big(\mathsf{lf}_{p,a}\big(\mathsf{left}_{n-\kappa}(\mathsf{Nd}(p)), \mathsf{pp}\big), \mathsf{sf}_{p,a}\big(\mathsf{right}_{\kappa}(\mathsf{Nd}(p))\big)\Big)$
  $p \leftarrow p/i_j$
  $b \leftarrow b + 1$
  $\mathsf{Nd}(p) \leftarrow \mathsf{of}\big(\mathsf{Nd}(p), \mathsf{left}_n\big(\mathsf{right}_{bn}(I)\big)\big)$
end for
return $\mathsf{Nd}(p)$

Fig. 2. Constrained PRF based on the generalized GGM tree $\mathsf{GGGM}^{\mathbf{Prim}}[\kappa, n, \nu, \theta, w, d, \mathsf{sf}, \mathsf{lf}]$. For simplicity, below we omit the suffix $[\kappa, ..., \mathsf{lf}]$.

For the input functions, we require that $j \neq j' \Rightarrow \big(\mathsf{lf}_{p,j}\big(\mathsf{left}_{n-\kappa}(z), \mathsf{pp}\big), \mathsf{sf}_{p,j}\big(\mathsf{right}_{\kappa}(z)\big)\big) \neq \big(\mathsf{lf}_{p,j'}\big(\mathsf{left}_{n-\kappa}(z), \mathsf{pp}\big), \mathsf{sf}_{p,j'}\big(\mathsf{right}_{\kappa}(z)\big)\big)$ for any $p \in \mathcal{P}^*$, i.e., distinct **Prim**-calls using the same node value $z \in \{0,1\}^n$ are necessarily on distinct inputs. For the outputting function, we limit our discussion to *seeded bijections*, i.e., $\mathsf{of}(z, \cdot)$ is a bijection on $\{0,1\}^n$ for *any* $z \in \{0,1\}^n$. This means of has an inverse of^{-1} such that $\mathsf{of}^{-1}\big(z, \mathsf{of}(z, I)\big) = I$ for any $z, I \in \{0,1\}^n$.

The complexity of our model is worthy, and it has covered the intuitive examples. We refer to Sect. 6 and 7 for details.

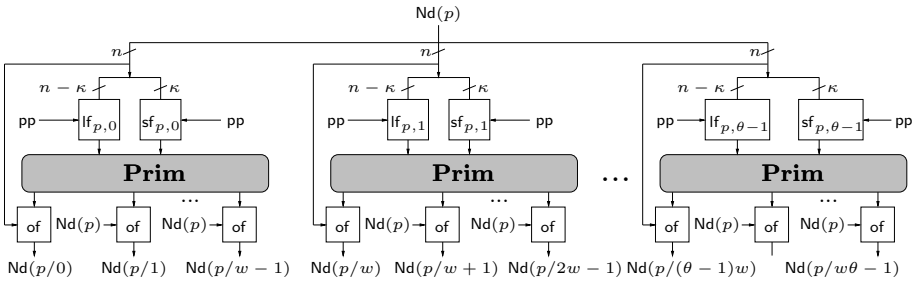

Fig. 3. Derivation calls relevant to a single node $\mathsf{Nd}(p)$ in the tree $\mathsf{GGGM}^{\mathbf{Prim}}$.

4 Multi-user Leakage CPRF Security of GGGM Trees

For clarity, below in Sect. 4.1 we first cast our general formalism in Sect. 2 into the concrete setting of $\mathsf{GGGM}^{\mathbf{Prim}}$. We also elaborate on the information leakages to-be-considered in this paper, as well as useful structural properties.

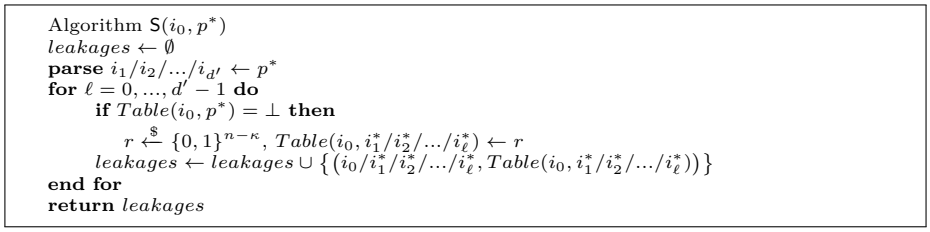

Fig. 4. Leakage simulator S for Theorems 1, 2, and 3.

With these preparations, we consider $\mathsf{GGGM}^{\mathbf{Prim}}$ with **Prim** being a FIL RO **H** in Sect. 4.2, and then the case of **Prim** being the Davies-Meyer construction $\mathsf{DM}^{\mathbf{E}}$ for an ideal cipher **E** in Sect. 4.3.

4.1 Concrete Settings

The oracles and leakage simulator. We assume *the leakage oracle* L *in* $\mathsf{muCo}^{\mathsf{L}}_{\mathbf{K}}$ *and* $\mathsf{muEv}^{\mathsf{L}}_{\mathbf{K}}$ *leaks the leftmost bits* $\mathsf{left}_{n-\kappa}(z)$ *for every intermediate node* $z \in \{0,1\}^n$ *appeared during the computations*. In detail, note that by our convention, a query to the evaluation oracle $\mathsf{muEv}^{\mathsf{L}}_{\mathbf{K}}$ is of the form (i_0, p), $p = i_1/i_2/.../i_d$. In the real world, $\mathsf{muEv}_{\mathbf{K}}$ will return $\mathsf{Nd}(i_0, p)$ the *n-bit node value at the end of the path p in the i_0-th tree*. As the corresponding leakages, we assume that $\mathsf{L}(K_{i_0}, p)$ will provide d additional values of $n-\kappa$ bits, i.e.,

$$\mathsf{left}_{n-\kappa}\big(\mathsf{Nd}(i_0, \bot)\big), \mathsf{left}_{n-\kappa}\big(\mathsf{Nd}(i_0, i_1)\big), ..., \mathsf{left}_{n-\kappa}\big(\mathsf{Nd}(i_0, i_1/.../i_{d-1})\big).$$

To wit, after issuing a query to $\mathsf{muEv}^{\mathsf{L}}_{\mathbf{K}}$, $\mathfrak{D}$ obtains $n + d(n-\kappa)$ bits information.

Similarly, a query to the constraining oracle $\mathsf{muCo}^{\mathsf{L}}_{\mathbf{K}}$ is of the form (i_0, p^*), $p^* = i_1^*/.../i_{d'}^*$ (see Fig. 2) which may be incomplete (i.e., $d' \leq d$), and $\mathsf{muCo}_{\mathbf{K}}$ will return the n-bit node value $\mathsf{Nd}(i_0, p^*)$. As the leakages, we assume that $\mathsf{L}(K_{i_0}, p^*)$ provides d' values $\mathsf{left}_{n-\kappa}\big(\mathsf{Nd}(i_0, \bot)\big), \mathsf{left}_{n-\kappa}\big(\mathsf{Nd}(i_0, i_1^*)\big), ..., \mathsf{left}_{n-\kappa}\big(\mathsf{Nd}(i_0, i_1^*/.../i_{d'-1}^*)\big)$, which resembles the evaluation queries. This means a single query to $\mathsf{muCo}^{\mathsf{L}}_{\mathbf{K}}(i_0, i_1^*/.../i_{d'}^*)$ gives rise to $n + d'(n-\kappa)$ bits.

As in Definition 4, the ideal oracle R returns an n-bit random value which is of the same size as $\mathsf{Nd}(i_0, p)$, and we consider a simulator S that simply outputs random simulated leakages. Formally, S is described in Fig. 4. Consequently, $\mathfrak{D}$ obtains the same amount of random bits (i.e., $n + d'(n-\kappa)$ bits) as in the real world. We stress that, while our security definition is simulation-based, our leakage simulator never "hijacks" adversarial random oracle queries. Therefore, our subsequent results only need *non-programmable random oracles*.

Query restrictions. Besides the query restriction imposed in Definition 4, we additionally assume that the distinguisher $\mathfrak{D}$ never makes redundant queries. Clearly, this cannot decrease attack advantage. For $\mathsf{GGGM}^{\mathbf{Prim}}$, these indicate:

a. $\mathfrak{D}$ never queries both $\mathsf{muEv}^{\mathsf{L}}_{\mathbf{K}}(i_0, p)$ and $\mathsf{muCo}^{\mathsf{L}}_{\mathbf{K}}(i_0, p^*)$ such that p^* is prefix of p (otherwise $p \in \mathcal{S}_{p^*, \mathsf{GGGM}}$).

b. $\mathfrak{D}$ never makes distinct queries $(i_0, p^*), (i_0, p^{**})$ to $\mathsf{muCo}^{\mathsf{L}}_{\mathbf{K}}$ such that p^* is prefix of p^{**} (otherwise (i_0, p^{**}) is redundant);

Brother paths. Given a path $p = i_1/.../i_{d'-1}/i_{d'} \in \mathcal{P}$, we define a set $\mathcal{B}\mathsf{r}(p) = \{p_1, p_2, ..., p_w\}$ of w paths, where for $1 \leq \ell \leq w$,

$$p_\ell = i_1/.../i_{d'-1}/i^{(\ell)}_{d'}, \quad i^{(\ell)}_{d'} = \lfloor \frac{i_{d'}}{w} \rfloor \cdot w + \ell - 1.$$

These w nodes $\mathsf{Nd}(i_0, p_1), ..., \mathsf{Nd}(i_0, p_w)$ are attributed to the same wn-bit output of a single call to **Prim** in the i_0-th tree. We thus call them *brother paths (of p)*. This notion will be used in subsequent analyses.

Concentration of labels. In the random oracle model, distinct labels separate corresponding **Prim**-calls. To formalize, consider the following sampling process. Given D distinct pairs $\mathcal{I} = \left\{(i^{(\ell)}_0, p^{(\ell)}) \in \{1, ..., u\} \times \mathcal{P}\right\}_{\ell=1,...,D}$,

1. We invoke KGen for u times to have u public parameters $\mathbf{P} = (\mathsf{pp}_1, ..., \mathsf{pp}_u)$;
2. We follow the strategy of S and sample D strings of $n - \kappa$ bits with replacement and define the list $\mathcal{L} = \{(i^{(1)}_0, p^{(1)}, r^{(1)}), ..., (i^{(D)}_0, p^{(D)}, r^{(D)})\}$.

Based on $\mathcal{L}$, we define a quantity for the maximal frequency of a certain label value, i.e.,

$$\mu(\mathcal{L}) := \max_{t \in \{0,1\}^\nu} \left\{ \left| \left\{ (i_0, p, r) \in \mathcal{L} : \mathsf{If}_{p,j}(r, \mathsf{pp}_{i_0}) = t \right\} \right| \right\}. \tag{4}$$

We denote this sampling process by $\mathbf{P} \leftarrow \mathsf{KGen}, \mathcal{L} \leftarrow \mathsf{S}$. Note that the quantity reflects *a property across multiple users in the ideal world*. A trivial upper bound is $\mu(\mathcal{L}) \leq D$, see Sect. 4.2 below. As will be seen in Theorems 1—3, the smaller $\mu(\mathcal{L})$, the better concrete security. A promising choice is to define $\mu(\mathcal{L})$ to be a (pseudo)random variable, as will be treated in Theorems 1-3. For example, one can choose a random initialization vector IV for every $\mathsf{GGGM}^{\mathbf{Prim}}$ instances/users and define $\mathsf{If}_{p,j}(r, \mathsf{pp}) := \cdot \| \mathsf{IV}$. This limits collisions between labels in distinct $\mathsf{GGGM}^{\mathbf{Prim}}$ instances and decreases $\mu(\mathcal{L})$. This approach will be used to improve FSS (see Sect. 7). As will be seen, a more sophisticated approach is to extract a part of the intermediate values as pseudorandom bits and "embed" them in the images of $\mathsf{If}_{p,j}(r, \mathsf{pp})$. This approach is used in $\mathsf{Bip32}$ key tree (see Sect. 6).

4.2 Random Oracle-based Trees

In detail, we consider defining $\mathbf{Prim}(x, y) := \mathbf{H}(x \| y)$ for an FIL RO $\mathbf{H} : \{0,1\}^{\nu+\omega} \mapsto \{0,1\}^{wn}$. Obviously, this is equivalent with $\mathbf{Prim}(x, y) := \mathbf{KH}(x, y)$ for a keyed FIL random oracle $\mathbf{KH} : \{0,1\}^\nu \times \{0,1\}^\lambda \mapsto \{0,1\}^{wn}$. Our main result is as follows.

Theorem 1. *Assume using the simulator S defined in Fig. 4, and:*

(i) $\mathbf{H} : \{0,1\}^{\nu+\omega} \mapsto \{0,1\}^{wn}$ *is modeled as a random oracle, and*
(ii) L *leaks* $\mathsf{left}_{n-\kappa}(\mathsf{Nd}(i_0, p^*))$ *for every intermediate node* $\mathsf{Nd}(i_0, p^*)$ *(see Sect. 4.1), and*
(iii) There exist quantities C *and* ε_μ *such that*

$$\Pr_{\mathbf{P}\leftarrow\mathsf{KGen},\mathcal{L}\leftarrow\mathsf{S}}\big[\mu(\mathcal{L}) > C\big] \leq \varepsilon_\mu. \tag{5}$$

Then, $\mathsf{GGGM}^{\mathbf{H}}$ *is a* (u, T, D, ε)*-*(L, S)*-constrained PRF for set system* $\mathbb{S}_{\mathsf{pre},\mathsf{GGGM}}$ *of Eq. (3), where*

$$\varepsilon = 2\varepsilon_\mu + \frac{2C \cdot (T + D)}{2^\kappa}. \tag{6}$$

Interpretation. The "leakages" and its simulator can be eliminated from Theorem 1 by setting $\kappa = n$. Therefore, the result implies the classical CPRF security of $\mathsf{GGGM}^{\mathbf{H}}$.

If $\mathsf{pp}_1, ..., \mathsf{pp}_u$ are picked according to some distribution, then $\mu(\mathcal{L})$ remains a random variable. An example is $\mathsf{FMTr}^{\mathsf{DM}^{\mathsf{AES}}}$ in Sect. 7. If $\mathsf{pp}_1, ..., \mathsf{pp}_u$ are not random, then $\mu(\mathcal{L})$ will be fixed by the design of the $\mathsf{GGGM}^{\mathbf{H}}$ instance. In particular, when lf is a constant function, we could (only) appeal for the trivial upper bound $C = D$. In this case, $\varepsilon_\mu = 0$, and the obtained bound $DT/2^\kappa + D^2/2^\kappa$ indicates the $\kappa - \log_2 D$ bits security mentioned in the introduction.

The leaked intermediate values are indistinguishable with the random simulated leakages. This means protocols can extract $n - \kappa$ pseudorandom bits from every n-bit intermediate value in $\mathsf{GGGM}^{\mathbf{H}}$ for arbitrary uses, as if these bits are "leaked" (to everyone including the adversaries). Indeed, Bip32 does use these $n - \kappa$ bits to derive signature keys: see Fig. 1 or Sect. 6.

Proof idea. Recall that the goal is to derive a bound for

$$\Big|\Pr\big[\mathfrak{D}^{\mathsf{muCo}^{\mathsf{L}}_{\mathbf{K}},\mathsf{muEv}^{\mathsf{L}}_{\mathbf{K}},\mathbf{H}} = 1\big] - \Pr\big[\mathfrak{D}^{\mathsf{muCo}^{\mathsf{S}}_{\mathbf{K}},\mathsf{R}^{\mathsf{S}},\mathbf{H}} = 1\big]\Big|.$$

In the ideal world $\mathfrak{D}^{\mathsf{muCo}^{\mathsf{S}}_{\mathbf{K}},\mathsf{R}^{\mathsf{S}},\mathbf{H}}$, the outputs of the oracle $\mathsf{muCo}^{\mathsf{S}}_{\mathbf{K}}$ (i.e., the constrained keys) depend on the secret keys $\mathsf{k}_1, ..., \mathsf{k}_u$. This complicates the H-coefficient based analysis. To remedy, we introduce a *random constraining oracle* $\$\mathsf{Cons}^{\mathsf{S}}$, which accepts queries of the same form (i_0, p^*) as $\mathsf{muCo}^{\mathsf{S}}_{\mathbf{K}}$ (and $\mathsf{muCo}^{\mathsf{L}}_{\mathbf{K}}$), but $\$\mathsf{Cons}$ returns true random n-bit strings as the constrained key $\mathsf{Nd}(i_0, p^*)$ (while S returns random leakages as before). This means for a query $(i_0, i_1^*/.../i_{d'}^*)$, the random constraining oracle $\$\mathsf{Cons}^{\mathsf{S}}$ returns in total $n+d'(n-\kappa)$ random bits. We take $(\$\mathsf{Cons}^{\mathsf{S}}, \mathsf{R}^{\mathsf{S}}, \mathbf{H})$ as an intermediate world. In this vein, when interacting with $(\$\mathsf{Cons}^{\mathsf{S}}, \mathsf{R}^{\mathsf{S}}, \mathbf{H})$, the information gained by $\mathfrak{D}$ is completely independent of any CPRF key, easing the analysis.

We then proceed with two steps: first, we prove indistinguishability of the real and the intermediate worlds; and then, we prove indistinguishability of the intermediate and the ideal worlds.

Indistinguishability of $(\mathsf{muCo}^{\mathsf{L}}_{\mathbf{K}}, \mathsf{muEv}^{\mathsf{L}}_{\mathbf{K}}, \mathbf{H})$ and $(\$\mathsf{Cons}^{\mathsf{S}}, \mathsf{R}^{\mathsf{S}}, \mathbf{H})$. For this step, we view the real $(\mathsf{muCo}^{\mathsf{L}}_{\mathbf{K}}, \mathsf{muEv}^{\mathsf{L}}_{\mathbf{K}}, \mathbf{H})$ as the real world, and the intermediate world $(\$\mathsf{Cons}^{\mathsf{S}}, \mathsf{R}^{\mathsf{S}}, \mathbf{H})$ the ideal world. We prove the following bound using the H-coefficient method.

$$\left|\Pr\left[\mathfrak{D}^{\mathsf{muCo}^{\mathsf{L}}_{\mathbf{K}}, \mathsf{muEv}^{\mathsf{L}}_{\mathbf{K}}, \mathbf{H}} = 1\right] - \Pr\left[\mathfrak{D}^{\$\mathsf{Cons}^{\mathsf{S}}, \mathsf{R}^{\mathsf{S}}, \mathbf{H}} = 1\right]\right| \leq \varepsilon_{\mu} + \frac{C \cdot (T + D)}{2^{\kappa}}. \quad (7)$$

Transcripts Since we are in the Random Oracle Model and aim at statistical indistinguishability, we could, without loss of generality [Pat09], consider a deterministic distinguisher $\mathfrak{D}$ interacting with the three oracles as reflected in Definition 4. To summarize the information gained by $\mathfrak{D}$ in a clear form, we introduce two list $\mathcal{Q}_{\mathbf{H}}$ and $\mathcal{Q}_{\mathsf{Nd}}$. The list $\mathcal{Q}_{\mathbf{H}} = \{(x^{(1)}, y^{(1)}), \ldots\}$ records $\mathfrak{D}$'s queries/answers to/from $\mathbf{H}$, with $(x, y) \in \mathcal{Q}_{\mathbf{H}}$ meaning $\mathbf{H}(x) = y$. The list $\mathcal{Q}_{\mathsf{Nd}} = \left\{(i_0^{(1)}, p^{(1)}, z^{(1)}, b^{(1)}), \ldots\right\}$ records the values in the tree obtained by $\mathfrak{D}$ via either the "standard" responses or the leakages. Every tuple $(i_0, p, z, b) \in \mathcal{Q}_{\mathsf{Nd}}$ is such that $p \in \mathcal{P}$, and:

- $z \in \{0,1\}^{n-\kappa}$ when $b = 0$, meaning that z is a leaked intermediate value;
- $z \in \{0,1\}^{n}$ when $b = 1$, meaning that z is either a constrained key or an output of $\mathsf{muEv}_{\mathbf{K}}$.

To simplify our proof language, we follow [GKWY20,GKW+20] and reveal a number of internal secrets to $\mathfrak{D}$ at the end of the interaction. In detail, we will reveal the κ-bit internal seeds to $\mathfrak{D}$, and add them to the transcript $\mathcal{Q}_{\mathsf{Nd}}$. In this respect, note that in the real world $(\mathsf{muCo}^{\mathsf{L}}_{\mathbf{K}}, \mathsf{muEv}^{\mathsf{L}}_{\mathbf{K}}, \mathbf{H})$, for every resulted tuple $(i_0, p, z, b) \in \mathcal{Q}_{\mathsf{Nd}}$, the corresponding node value $\mathsf{Nd}(i_0, p)$ necessarily appeared during processing the queries. Moreover, $z = \mathsf{Nd}(i_0, p)$ when $b = 1$, and $z = \mathsf{left}_{n-\kappa}\big(\mathsf{Nd}(i_0, p)\big)$ when $b = 0$. We will thus reveal the "missing" κ bits $z_R = \mathsf{right}_{\kappa}\big(\mathsf{Nd}(i_0, p)\big)$ to $\mathfrak{D}$ and add them to $\mathcal{Q}_{\mathsf{Nd}}$. Since $\mathsf{k}_{i_0} = \mathsf{Nd}(i_0, \perp)$, this means the u user keys $\mathsf{k}_1, ..., \mathsf{k}_u$ are also completely given. In the ideal world, we reveal and add random "dummy" bits to the transcript. We also append to the transcript u public parameters $\mathbf{P} = (\mathsf{pp}_1, ..., \mathsf{pp}_u)$ that are sampled according to the same distribution as the real world. By this, we obtain an extended list $\mathcal{Q}^{\mathsf{X}}_{\mathsf{Nd}} = \left\{(i_0^{(1)}, p^{(1)}, z^{(1)}, b^{(1)}), \ldots\right\}$, among which $(i_0^{(1)}, p^{(1)}), (i_0^{(2)}, p^{(2)}), ...$ are exactly the same as those in $\mathcal{Q}_{\mathsf{Nd}}$, while $z^{(1)}, z^{(2)}, ...$ are all "full" n-bit strings regardless of the values of $b^{(1)}, b^{(2)}, ...$ In all, we define

$$\mathcal{Q} = (\mathcal{Q}^{\mathsf{X}}_{\mathsf{Nd}}, \mathcal{Q}_{\mathbf{H}}, \mathbf{P})$$

as a transcript. It is without loss of generality to provide these additional values, since the distinguisher is free to ignore them. Below we will first define the set Θ_{bad} of bad transcripts, and then analyze good transcripts.

Internal evaluation list $\mathcal{Q}^{\mathsf{x}}_{\mathbf{H}}$ Regarding $\mathcal{Q}^{\mathsf{X}}_{\mathsf{Nd}}$, we make crucial observations as follows. For any $(i_0, p, z, b) \in \mathcal{Q}^{\mathsf{X}}_{\mathsf{Nd}}$, $p = i_1/.../i_{d'}$ with $d' \geq 1$,

– it holds $(i_0^\circ, p^\circ, \mathsf{Nd}(p^\circ), \star) \in \mathcal{Q}^{\mathsf{X}}_{\mathsf{Nd}}$ for every prefix p° of p, since $\mathsf{Nd}(i_0^\circ, p^\circ)$ indeed appeared during computing $\mathsf{Nd}(i_0, p) = z$;
– it holds $(i_0, p', \mathsf{Nd}(p'), \star) \in \mathcal{Q}^{\mathsf{X}}_{\mathsf{Nd}}$ for every brother path $p' \in \mathcal{B}\mathsf{r}(p)$ of p. To see this, note that for any path $p \in \mathcal{P}$, $(i_0, p, \mathsf{Nd}(p), \star) \notin \mathcal{Q}^{\mathsf{X}}_{\mathsf{Nd}}$ only if the corresponding value $\mathsf{Nd}(i_0, p)$ is never computed during the interaction. The only possibility is that there exists a constraining query $\mathsf{muCo}^{\mathsf{L}}_{\mathbf{K}}(i_0, p^*)$ such that p^* is a prefix of p, and this forbids querying (i_0, p) due to the query restrictions mentioned in Sect. 4.1. This essentially implies $\mathsf{Nd}(i_0, p')$ is never computed for any $p' \in \mathcal{B}\mathsf{r}(p)$. By this, $(i_0, p, \mathsf{Nd}(i_0, p), \star) \notin \mathcal{Q}^{\mathsf{X}}_{\mathsf{Nd}} \Leftrightarrow \forall p' \in \mathcal{B}\mathsf{r}(p) : (i_0, p', \mathsf{Nd}(i_0, p'), \star) \notin \mathcal{Q}^{\mathsf{X}}_{\mathsf{Nd}}$, and thus the claim.

By these, consider any $(i_0, p, z, \star) \in \mathcal{Q}^{\mathsf{X}}_{\mathsf{Nd}}$ with $p = i_1/.../i_{d'-1}/i_{d'}$, $1 \le d' \le d$. Let $p^* = i_1/.../i_{d'-1}$, $j^* = \lfloor \frac{i_{d'}}{w} \rfloor$, and let $p_\ell = i_1/.../i_{d'-1}/wj^*+\ell-1$ $(1 \le \ell \le w)$. Then $\mathcal{Q}^{\mathsf{x}}_{\mathbf{H}}$ includes the "internal evaluation tuple" $(i_0, p^*, j^*, x^*, y^*)$, where (recall from Sect. 3.2 that $\mathsf{of}(z, \cdot)$ is bijective)

$$\begin{aligned} x^* &= \mathsf{lf}_{p^*,j^*}\big(\mathsf{left}_{n-\kappa}\big(\mathsf{Nd}(i_0, p^*)\big), \mathsf{pp}_{i_0}\big) \parallel \mathsf{sf}_{p^*,j^*}\big(\mathsf{right}_\kappa\big(\mathsf{Nd}(i_0, p^*)\big)\big), \\ y^* &= \mathsf{of}^{-1}\big(\mathsf{Nd}(i_0, p^*), \mathsf{Nd}(i_0, p_1)\big) \parallel \mathsf{of}^{-1}\big(\mathsf{Nd}(i_0, p^*), \mathsf{Nd}(i_0, p_2)\big) \parallel \mathsf{of}^{-1}\big(\mathsf{Nd}(i_0, p^*), \mathsf{Nd}(i_0, p_w)\big). \end{aligned} \tag{8}$$

By the above two observations, all the mentioned node values can be found in $\mathcal{Q}^{\mathsf{X}}_{\mathsf{Nd}}$, and x^*, y^* are thus well-defined. The additional fields p^* and j^* will significantly simplify proof languages. It can be seen that the set $\mathcal{Q}^{\mathsf{x}}_{\mathbf{H}}$ records all the internal random oracle evaluations that appeared during the real world interaction, and $|\mathcal{Q}^{\mathsf{x}}_{\mathbf{H}}| = D$ is the aforementioned *effective data complexity* of $\mathfrak{D}$. Indeed, this constitutes its motivation.

Bad transcripts A transcript $\mathcal{Q} = (\mathcal{Q}^{\mathsf{X}}_{\mathsf{Nd}}, \mathcal{Q}_{\mathbf{H}})$ is *bad*, if any of the following conditions is fulfilled:

– (B-1) $\mu \ge C$.
– (B-2) There exist a tuple $(i_0^*, p^*, j^*, x^*, y^*) \in \mathcal{Q}^{\mathsf{x}}_{\mathbf{H}}$ and a pair $(x, y) \in \mathcal{Q}_{\mathbf{H}}$ such that $x^* = x$.
– (B-3) There are distinct tuples $(i_0^*, p^*, j^*, x^*, y^*), (i_0^{**}, p^{**}, j^{**}, x^{**}, y^{**}) \in \mathcal{Q}^{\mathsf{x}}_{\mathbf{H}}$ such that $x^* = x^{**}$.

The 1st condition captures that the "labels" of the internal calls are too concentrated. The 2nd condition addresses the case where an internal random oracle evaluation in a $\mathsf{muCo}^{\mathsf{L}}_{\mathbf{K}}/\mathsf{muEv}^{\mathsf{L}}_{\mathbf{K}}$ query collide with an adversarial offline random oracle query, while the 3rd condition addresses the case where distinct $\mathsf{muCo}^{\mathsf{L}}_{\mathbf{K}}/\mathsf{muEv}^{\mathsf{L}}_{\mathbf{K}}$ queries issue the same internal random oracle evaluation.

First, $\Pr[\text{(B-1)}] \le \varepsilon_\mu$ immediately follows from Eq. (5), since leakages in $\mathfrak{D}^{\$\mathsf{Cons}^{\mathsf{S}},\mathsf{R}^{\mathsf{S}},\mathbf{H}}$ are purely random. For (B-2), consider each choice of $((x, y), (i_0^*, p^*, j^*, x^*, y^*)) \in \mathcal{Q}_{\mathbf{H}} \times \mathcal{Q}^{\mathsf{x}}_{\mathbf{H}}$. By the definition of $\mathcal{Q}^{\mathsf{x}}_{\mathbf{H}}$, $x = x^*$ means $x = \mathsf{lf}_{p^*,j^*}\big(\mathsf{left}_{n-\kappa}\big(z^*\big), \mathsf{pp}_{i_0^*}\big) \| \mathsf{sf}_{p^*,j^*}\big(\mathsf{right}_\kappa\big(z^*\big)\big)$ for $(i_0^*, p^*, z^*, \star) \in \mathcal{Q}^{\mathsf{X}}_{\mathsf{Nd}}$. Since the seed $\mathsf{right}_\kappa(z^*)$ is uniform in $\{0,1\}^\kappa$ in the ideal world (it is the random "dummy" value appended to $\mathcal{Q}$), and since $\mathsf{sf}_{p^*,j^*}(\cdot)$ is injective, the probability to have $\mathsf{sf}_{p^*,j^*}\big(\mathsf{right}_\kappa\big(z^*\big)\big) = \mathsf{right}_\lambda(x)$ is $1/2^\kappa$.

Now, the condition $x = x^*$ is fulfilled only if $\mathsf{left}_\nu(x) = \mathsf{lf}_{p^*,j^*}\big(\mathsf{left}_{n-\kappa}\big(z^*\big), \mathsf{pp}_{i_0^*}\big)$. By this, for any $t \in \{0,1\}^\nu$, we define $\mathcal{Q}_{\mathbf{H}}[t] := \big\{s \in \{0,1\}^\lambda : (t\|s, \star) \in \mathcal{Q}_{\mathbf{H}}\big\}$. Then, the probability that (B-2) is fulfilled w.r.t. the above tuple $(i_0^*, p^*, j^*, x^*, y^*)$ is $\big|\mathcal{Q}_{\mathbf{H}}\big[\mathsf{lf}_{p^*,j^*}\big(\mathsf{left}_{n-\kappa}\big(z^*\big), \mathsf{pp}_{i_0^*}\big)\big]\big|/2^\kappa$. By the above and our definition Eq. (4),

$$\begin{aligned}
\Pr\big[\text{(B-2)} \mid \neg\text{(B-1)}\big] &= \sum_{(i_0^*,p^*,j^*,x^*,y^*)\in\mathcal{Q}^{\mathsf{x}}_{\mathbf{H}}} \frac{\big|\mathcal{Q}_{\mathbf{H}}\big[\mathsf{lf}_{p^*,j^*}\big(\mathsf{left}_{n-\kappa}(\mathsf{Nd}(i_0^*,p^*)), \mathsf{pp}_{i_0^*}\big)\big]\big|}{2^\kappa} \\
&= \sum_{t\in\{0,1\}^\nu} \sum_{(i_0^*,p^*,j^*,x^*,y^*)\in\mathcal{Q}^{\mathsf{x}}_{\mathbf{H}}:\mathsf{lf}_{p^*,j^*}\left(\mathsf{left}_{n-\kappa}(\mathsf{Nd}(i_0^*,p^*)),\mathsf{pp}_{i_0^*}\right)=t} \frac{\big|\mathcal{Q}_{\mathbf{H}}[t]\big|}{2^\kappa} \\
&\le \sum_{t\in\{0,1\}^\nu} \frac{C\cdot\big|\mathcal{Q}_{\mathbf{H}}[t]\big|}{2^\kappa} \qquad \text{(By } \neg\text{(B-1))} \\
&\le \frac{C\cdot T}{2^\kappa} \qquad \Big(\text{Since } \sum_{t\in\{0,1\}^\nu} \big|\mathcal{Q}_{\mathbf{H}}[t]\big| = \big|\mathcal{Q}_{\mathbf{H}}\big| = T\Big).
\end{aligned}$$

For (B-3), consider each pair $(i_0^*, p^*, j^*, x^*, y^*), (i_0^{**}, p^{**}, j^{**}, x^{**}, y^{**}) \in \mathcal{Q}^{\mathsf{x}}_{\mathbf{H}}$. By the definition of $\mathcal{Q}^{\mathsf{x}}_{\mathbf{H}}$, it holds $x^* = \mathsf{lf}_{p^*,j^*}\big(\mathsf{left}_{n-\kappa}\big(z^*\big), \mathsf{pp}_{i_0^*}\big)\|\mathsf{sf}_{p^*,j^*}\big(\mathsf{right}_\kappa\big(z^*\big)\big)$ and $x^{**} = \mathsf{lf}_{p^{**},j^{**}}\big(\mathsf{left}_{n-\kappa}\big(z^{**}\big), \mathsf{pp}_{i_0^{**}}\big)\|$ $\mathsf{sf}_{p^{**},j^{**}}\big(\mathsf{right}_\kappa\big(z^{**}\big)\big)$ for $(i_0^*, p^*, z^*, \star), (i_0^{**}, p^{**}, z^{**}, \star) \in \mathcal{Q}^{\mathsf{X}}_{\mathsf{Nd}}$ respectively. Then the condition $x^* = x^{**}$ is fulfilled only if $(i_0^*, p^*) \neq (i_0^{**}, p^{**})$, as per our restriction on sf and lf. By this, $\mathsf{right}_\kappa(z^*)$ and $\mathsf{right}_\kappa(z^{**})$ are uniform and independent. Moreover, $\mathsf{sf}_{p^*,j^*}\big(\cdot\big)$ and $\mathsf{sf}_{p^{**},j^{**}}\big(\cdot\big)$ are injective. Hence, the probability to have $\mathsf{sf}_{p^*,j^*}\big(\mathsf{right}_\kappa\big(z^*\big)\big) = \mathsf{sf}_{p^{**},j^{**}}\big(\mathsf{right}_\kappa\big(z^{**}\big)\big)$ is $1/2^\kappa$. Further, $x^* = x^{**}$ requires $\mathsf{lf}_{p^*,j^*}\big(\mathsf{left}_{n-\kappa}\big(z^*\big), \mathsf{pp}_{i_0^*}\big) = \mathsf{lf}_{p^{**},j^{**}}\big(\mathsf{left}_{n-\kappa}\big(z^{**}\big), \mathsf{pp}_{i_0^{**}}\big)$. Therefore,

$$\begin{aligned}
&\Pr\big[\text{(B-3)} \mid \neg\text{(B-1)}\big] \\
&= \sum_{(i_0^*,p^*,j^*,x^*,y^*)\in\mathcal{Q}^{\mathsf{x}}_{\mathbf{H}}} \Bigg(\sum_{\substack{(i_0^{**},p^{**},j^{**},x^{**},y^{**}) \neq (i_0^*,p^*,j^*,x^*,y^*) : \\ \mathsf{lf}_{p^*,j^*}\left(\mathsf{left}_{n-\kappa}(\mathsf{Nd}(i_0^*,p^*)),\mathsf{pp}_{i_0^*}\right) = \mathsf{lf}_{p^{**},j^{**}}\left(\mathsf{left}_{n-\kappa}(\mathsf{Nd}(i_0^{**},p^{**})),\mathsf{pp}_{i_0^{**}}\right)}} \frac{1}{2^\kappa} \Bigg) \\
&\le \sum_{(i_0^*,p^*,j^*,x^*,y^*)\in\mathcal{Q}^{\mathsf{x}}_{\mathbf{H}}} \frac{C}{2^\kappa} \qquad \text{(By } \neg\text{(B-1))} \\
&\le \frac{C\cdot D}{2^\kappa}.
\end{aligned}$$

In all, a union bound yields

$$\Pr\big[T_{\mathrm{id}} \in \Theta_{\mathrm{bad}}\big] \le \varepsilon_\mu + \frac{C\cdot(T+D)}{2^\kappa}. \tag{9}$$

Ratio of probabilities of good transcripts. One key insight of the H-coefficient method is that, for any $\mathcal{Q} \in \Theta_{\mathrm{good}}$, the probability ratio $\Pr[T_{\mathrm{re}} = \mathcal{Q}]/\Pr[T_{\mathrm{id}} = \mathcal{Q}]$

is equal to the ratio between the probability that the real-world oracles are consistent with $\mathcal{Q}$ and the probability that the ideal-world oracles are consistent with $\mathcal{Q}$. Now, for any attainable transcript $\mathcal{Q} = (\mathcal{Q}^{\mathsf{X}}_{\mathsf{Nd}}, \mathcal{Q}_{\mathbf{H}}, \mathbf{P})$, the probability that the ideal world transcript is consistent with $\mathcal{Q}$ is always exactly

$$\frac{1}{2^{Twn}} \times \frac{1}{2^{|\mathcal{Q}^{\mathsf{X}}_{\mathsf{Nd}}|n}} \times \Pr\big[\mathbf{P} \leftarrow \mathsf{KGen}\big] \tag{10}$$

This is so since in $\mathfrak{D}^{\$\mathsf{Cons}^{\mathsf{S}},\mathsf{R}^{\mathsf{S}},\mathbf{H}}$,

(i) the probability that a random oracle with wn-bit outputs is consistent with the T queries in $\mathcal{Q}_{\mathbf{H}}$ is exactly $1/2^{Twn}$;
(ii) the probability that the random node values equal those in $\mathcal{Q}^{\mathsf{X}}_{\mathsf{Nd}}$ is $1/2^{|\mathcal{Q}^{\mathsf{X}}_{\mathsf{Nd}}|n}$. Note that these nodes include the u secret keys $\mathsf{k}_1, ..., \mathsf{k}_u$ since $\mathsf{Nd}(i_0, \perp) = \mathsf{k}_{i_0}$;
(iii) the probability that the public parameter vector $\mathbf{P}$ is produced by KGen is $\Pr\big[\mathbf{P} \leftarrow \mathsf{KGen}\big]$.

Bounding the distinguishing advantage of $\mathfrak{D}$ thus reduces to bounding the probability that the real world is consistent with transcripts $\mathcal{Q} \in \Theta_{\text{good}}$.

Let $\mathbf{H} \vdash \mathcal{Q}_{\mathbf{H}}$ denote the event that a function $\mathbf{H}$ is consistent with the queries/answers in $\mathcal{Q}_{\mathbf{H}}$, i.e., that $\mathbf{H}(x) = y$ for all $(x, y) \in \mathcal{Q}_{\mathbf{H}}$. Since, in the real world, the information in $\mathcal{Q}^{\mathsf{X}}_{\mathsf{Nd}}$ is completely determined by $\mathbf{H}$ and the user keys $\mathbf{K} = (K_1, ..., K_u)$, we can also write $(\mathbf{H}, \mathbf{K}) \vdash \mathcal{Q}^{\mathsf{X}}_{\mathsf{Nd}}$ to denote the event that the function $\mathbf{H}$ and keys $\mathbf{K}$ are consistent with the queries/answers in $\mathcal{Q}^{\mathsf{X}}_{\mathsf{Nd}}$. For a (good) transcript $\mathcal{Q} = (\mathcal{Q}^{\mathsf{X}}_{\mathsf{Nd}}, \mathcal{Q}_{\mathbf{H}}, \mathbf{P})$, the probability that the real world is consistent with $\mathcal{Q}$ is exactly

$$\begin{aligned} &\Pr\big[(\mathbf{H}, \mathbf{K}) \vdash \mathcal{Q}^{\mathsf{X}}_{\mathsf{Nd}} \mid \mathbf{H} \vdash \mathcal{Q}_{\mathbf{H}}\big] \times \Pr\big[\mathbf{H} \vdash \mathcal{Q}_{\mathbf{H}}\big] \times \Pr\big[\mathbf{K} \leftarrow \mathsf{KGen}\big] \\ = &\Pr\big[(\mathbf{H}, \mathbf{K}) \vdash \mathcal{Q}^{\mathsf{X}}_{\mathsf{Nd}} \mid \mathbf{H} \vdash \mathcal{Q}_{\mathbf{H}}\big] \times \Pr\big[\mathbf{H} \vdash \mathcal{Q}_{\mathbf{H}}\big] \times \frac{1}{2^{un}} \times \Pr\big[\mathbf{P} \leftarrow \mathsf{KGen}\big] \end{aligned} \tag{11}$$

(using independence of $\mathbf{K}$ and $\mathbf{H}$). We have $\Pr[\mathbf{H} \vdash \mathcal{Q}_{\mathbf{H}}] = 1/2^{Twn}$ exactly as before. The crux of the proof thus reduces to bounding $\Pr[(\mathbf{H}, \mathbf{K}) \vdash \mathcal{Q}^{\mathsf{X}}_{\mathsf{Nd}} \mid \mathbf{H} \vdash \mathcal{Q}_{\mathbf{H}}]$. For this, note that in the real world, the list $\mathcal{Q}^{\mathsf{x}}_{\mathbf{H}}$ essentially summarizes *all* the random oracle queries internally issued by $\mathsf{GGGM}^{\mathbf{H}}$ for producing the transcript $\mathcal{Q}^{\mathsf{X}}_{\mathsf{Nd}}$. Therefore,

$$\Pr\big[(\mathbf{H}, \mathbf{K}) \vdash \mathcal{Q}^{\mathsf{X}}_{\mathsf{Nd}} \mid \mathbf{H} \vdash \mathcal{Q}_{\mathbf{H}}\big] = \Pr\big[\forall (i_0^*, p^*, j^*, x^*, y^*) \in \mathcal{Q}^{\mathsf{x}}_{\mathbf{H}} : \mathbf{H}(x^*) = y^* \mid \mathbf{H} \vdash \mathcal{Q}_{\mathbf{H}}\big].$$

We further show that the latter probability concerns with $\mathbf{H}$ satisfying "new" and distinct equations. Let $\mathcal{Q}^{\mathsf{x}}_{\mathbf{H}} = \big((i_0^{(1)}, p^{(1)}, j^{(1)}, x^{(1)}, y^{(1)}), ..., (i_0^{(D)}, p^{(D)}, j^{(D)}, x^{(D)}, y^{(D)})\big)$ in arbitrary order. The probability can be expressed as

$$\prod_{\ell=1}^{D} \Pr\big[\mathbf{H}(x^{(\ell)}) = y^{(\ell)} \mid \mathbf{H} \vdash \mathcal{Q}_{\mathbf{H}} \wedge \forall \ell' < \ell : \mathbf{H}(x^{(\ell')}) = y^{(\ell')}\big].$$

Fix some ℓ. Since the transcript is good, there is no query of the form $(x^{(\ell)}, \star)$ in $\mathcal{Q}_{\mathbf{H}}$ (since (B-2) does not occur), nor is $\mathbf{H}(x^{(\ell)})$ determined by the fact that $\mathbf{H}(x^{(\ell')}) = y^{(\ell')}$ for all $\ell' < \ell$ (since (B-3) does not occur). Thus, we have

$$\Pr\big[\mathbf{H}(x^{(\ell)}) = y^{(\ell)} \mid \mathbf{H} \vdash \mathcal{Q}_{\mathbf{H}} \wedge \forall \ell' < \ell : \mathbf{H}(x^{(\ell')}) = y^{(\ell')}\big] = 1/2^{wn}.$$

for all i. The term $1/2^{Dwn}$ thus follows.

It remains to determine the size of $\mathcal{Q}^{\mathsf{x}}_{\mathbf{H}}$. Recall that, for every pair

$$\Big(i_0, p, j, \mathsf{lf}_{p,j}\big(\mathsf{left}_{n-\kappa}\big(\mathsf{Nd}(i_0,p)\big), \mathsf{pp}_{i_0}\big) \big\| \mathsf{sf}_{p,j}\big(\mathsf{right}_{\kappa}\big(\mathsf{Nd}(i_0,p)\big)\big), \quad \overline{z_1} \| ... \| \overline{z_w}\Big) \in \mathcal{Q}^{\mathsf{x}}_{\mathbf{H}},$$

it holds $\big(i_0, p/jw, \mathsf{of}(\mathsf{Nd}(i_0,p), \overline{z_1}), \star\big), ..., \big(i_0, p/jw+w-1, \mathsf{of}(\mathsf{Nd}(i_0,p), \overline{z_w}), \star\big) \in \mathcal{Q}^{\mathsf{X}}_{\mathsf{Nd}}$. On the other hand, it can be seen that, for every set of w triples $(i_0, p_1, z_1, \star), ..., (i_0, p_w, z_w, \star) \in \mathcal{Q}^{\mathsf{X}}_{\mathsf{Nd}}$ such that $p_1 = p^*/j^*w$, $p_1, ..., p_w \in \mathcal{B}\mathsf{r}(p_1)$, the n-bit values $\mathsf{Nd}(i_0, p^*) = z^*$ and $z_1, ..., z_w$ correspond to a unique tuple $(i_0, p^*, j^*, x^*, y^*) \in \mathcal{Q}^{\mathsf{x}}_{\mathbf{H}}$ with $y^* = \mathsf{of}^{-1}(z^*, z_1) \| ... \| \mathsf{of}^{-1}(z^*, z_w)$. By this, there is a one-to-one correspondence between *sets of w tuples in $\mathcal{Q}^{\mathsf{X}}_{\mathsf{Nd}}$ with associated paths being brothers* and *tuples in $\mathcal{Q}^{\mathsf{x}}_{\mathbf{H}}$*, and thus

$$\big|\mathcal{Q}^{\mathsf{x}}_{\mathbf{H}}\big| = \frac{\big|\{(i_0, p, z, b) \in \mathcal{Q}^{\mathsf{X}}_{\mathsf{Nd}} : p \neq \bot\}\big|}{w}.$$

Note that $|\mathcal{Q}^{\mathsf{X}}_{\mathsf{Nd}}| - \big|\{(i_0, p, z, b) \in \mathcal{Q}^{\mathsf{X}}_{\mathsf{Nd}} : p \neq \bot\}\big| = u$, and the u entries are $(1, \bot, \mathsf{k}_1, \star), ..., (u, \bot, \mathsf{k}_u, \star)$. By this,

$$\begin{aligned}
\text{Eq. (11)} &= \Big(\frac{1}{2^{wn}}\Big)^{D} \times \frac{1}{2^{Twn}} \times \frac{1}{2^{un}} \times \Pr\big[\mathbf{P} \leftarrow \mathsf{KGen}\big] \\
&= \Big(\frac{1}{2^{n}}\Big)^{\left|\{(i_0,p,z,b) \in \mathcal{Q}^{\mathsf{X}}_{\mathsf{Nd}} : p \neq \bot\}\right|} \times \frac{1}{2^{Twn}} \times \frac{1}{2^{un}} \times \Pr\big[\mathbf{P} \leftarrow \mathsf{KGen}\big] \\
&= \frac{1}{2^{Twn}} \times \Big(\frac{1}{2^{n}}\Big)^{|\mathcal{Q}^{\mathsf{X}}_{\mathsf{Nd}}|} \times \Pr\big[\mathbf{P} \leftarrow \mathsf{KGen}\big].
\end{aligned}$$

So the probability that the real world is consistent with the transcript is the same as Eq. (10). This means Eq. (9), the probability of obtaining bad transcripts, constitutes the gap between the real world $(\mathsf{muCo}^{\mathsf{L}}_{\mathbf{K}}, \mathsf{muEv}^{\mathsf{L}}_{\mathbf{K}}, \mathbf{H})$ and the intermediate world $(\$\mathsf{Cons}^{\mathsf{S}}, \mathsf{R}^{\mathsf{S}}, \mathbf{H})$.

Indistinguishability of $(\$\mathsf{Cons}^{\mathsf{S}}, \mathsf{R}^{\mathsf{S}}, \mathbf{H})$ and $(\mathsf{muCo}^{\mathsf{S}}_{\mathbf{K}}, \mathsf{R}^{\mathsf{S}}, \mathbf{H})$. For this, we view $(\mathsf{muCo}^{\mathsf{S}}_{\mathbf{K}}, \mathsf{R}^{\mathsf{S}}, \mathbf{H})$ as the real world and $(\$\mathsf{Cons}^{\mathsf{S}}, \mathsf{R}^{\mathsf{S}}, \mathbf{H})$ as the ideal, and prove the following bound:

$$\Big|\Pr\big[\mathfrak{D}^{\$\mathsf{Cons}^{\mathsf{S}}, \mathsf{R}^{\mathsf{S}}, \mathbf{H}} = 1\big] - \Pr\big[\mathfrak{D}^{\mathsf{muCo}^{\mathsf{S}}_{\mathbf{K}}, \mathsf{R}^{\mathsf{S}}, \mathbf{H}} = 1\big]\Big| \leq \varepsilon_{\mu} + \frac{C \cdot (T+D)}{2^{\kappa}}. \tag{12}$$

Essentially, this step requires establishing pseudorandomness of the constrained keys, which follows the same idea as the first step with a number of changes. In detail, we employ four list $\mathcal{Q}_{\mathbf{H}}, \mathcal{Q}_{\mathsf{S}}, \mathcal{Q}_{\mathsf{Nd}}$, and $\mathcal{Q}_R$ to keep the information

gained by $\mathfrak{D}$. The RO-query list $\mathcal{Q}_{\mathbf{H}}$ is just the same as before. The list $\mathcal{Q}_{\mathsf{S}} = \{(i_0^{(1)}, p^{(1)}, r^{(1)}), \dots\}$ records all the leakages returned by S, where $i_0^{(\ell)} \in \{1, ...u\}$, $p^{(\ell)} \in \mathcal{P}^*$, and $r^{(\ell)} \in \{0,1\}^{n-\kappa}$ indicates that at the position $p^{(\ell)}$ in the $i_0^{(1)}$-th tree, the leakage block returned by S is $r^{(\ell)}$. The list $\mathcal{Q}_{\mathsf{Nd}}$ is modified such that $\mathcal{Q}_{\mathsf{Nd}} = \{(i_0^{(1)}, p^{(1)}, z^{(1)}), \dots\}$ records the queries and responses of $\$\mathsf{Cons}/\mathsf{muCo}_{\mathbf{K}}$. Namely, the ℓ-th tuple $(i_0^{(\ell)}, p^{(\ell)}, z^{(\ell)}) \in \mathcal{Q}_{\mathsf{Nd}}$ indicates that $\mathfrak{D}$ made a query $(i_0^{(\ell)}, p^{(\ell)})$ to $\$\mathsf{Cons}/\mathsf{muCo}_{\mathbf{K}}$ and the n-bit "standard" response is $z^{(\ell)}$. Finally, $\mathcal{Q}_R = \{(i_0^{(1)}, p^{(1)}, z^{(1)}), \dots\}$ records the queries and responses of R, where $(i_0^{(\ell)}, p^{(\ell)}, z^{(\ell)}) \in \mathcal{Q}_{\mathsf{Nd}}$ indicates that $\mathfrak{D}$ made a query $(i_0^{(\ell)}, p^{(\ell)})$ to R^{S} and the n-bit (random) response of R is $z^{(\ell)}$.

We also reveal the internal values corresponding to $\mathsf{muCo}_{\mathbf{K}}^{\mathsf{S}}$'s evaluations to $\mathfrak{D}$ at the end of the interaction to extend $\mathcal{Q}_{\mathsf{Nd}}$, but the strategy differs from the previous step (Sect. 4.2). In detail, note that every tuple $(i_0, p, z) \in \mathcal{Q}_{\mathsf{Nd}}$ indicates $\mathfrak{D}$ making a query (i_0, p) to $\mathsf{muCo}_{\mathbf{K}}^{\mathsf{S}}$, and for every p' that is prefix of p, the corresponding node value $z' = \mathsf{Nd}(i_0, p')$ necessarily appeared during processing the query. We thus reveal all such node values $z' = \mathsf{Nd}(i_0, p')$ to $\mathfrak{D}$ and add the corresponding triple (i_0, p', z') to $\mathcal{Q}_{\mathsf{Nd}}$. We also reveal all the u secret keys $\mathsf{k}_1, ..., \mathsf{k}_u$ and add the corresponding triple $(i_0, \perp, \mathsf{k}_{i_0})$ to $\mathcal{Q}_{\mathsf{Nd}}$. In the ideal world, we reveal random "dummy" n-bit blocks $z' \xleftarrow{\$} \{0,1\}^n$ to $\mathfrak{D}$ and add (i_0, p', z') to $\mathcal{Q}_{\mathsf{Nd}}$ correspondingly. By this, we obtain an extended list $\mathcal{Q}_{\mathsf{Nd}}^{\mathsf{X}} = \{(i_0^{(1)}, p^{(1)}, z^{(1)}), \dots\}$, among which $(i_0^{(1)}, p^{(1)}), (i_0^{(2)}, p^{(2)}), ...$ represents the positions to the nodes in $\mathcal{Q}_{\mathsf{Nd}}$, while $z^{(1)}, z^{(2)}, ...$ are all "full" n-bit strings that are either "real" intermediate values (in the real world) or random "dummy" blocks (in the ideal world). Finally, we also sample u public parameters $\mathbf{P}$ in the ideal world. In all, we define

$$\mathcal{Q} = (\mathcal{Q}_{\mathsf{Nd}}^{\mathsf{X}}, \mathcal{Q}_R, \mathcal{Q}_{\mathsf{S}}, \mathcal{Q}_{\mathbf{H}}, \mathbf{P})$$

as a transcript. Clearly, $|\mathcal{Q}_{\mathsf{Nd}}^{\mathsf{X}}| \le D$. The real world probability $\Pr[T_{\mathrm{re}} = \mathcal{Q}]$ is then written as

$$\Pr[\mathbf{H} \vdash \mathcal{Q}_{\mathbf{H}}] \times \Pr[\mathsf{S} \vdash \mathcal{Q}_{\mathsf{S}}] \times \Pr[\mathsf{R} \vdash \mathcal{Q}_R] \times \Pr[(\mathbf{H}, \mathbf{K}) \vdash \mathcal{Q}_{\mathsf{Nd}}^{\mathsf{X}} \mid \mathbf{H} \vdash \mathcal{Q}_{\mathbf{H}}] \times \Pr[\mathbf{P} \leftarrow \mathsf{KGen}], \tag{13}$$

where $\mathsf{S} \vdash \mathcal{Q}_{\mathsf{S}}$ denotes the event that the random leakages returned by S are consistent with those in $\mathcal{Q}_{\mathsf{S}}$, $\mathsf{R} \vdash \mathcal{Q}_R$ denotes $\mathsf{R}(i_0, p) = z$ for every $(i_0, p, z) \in \mathcal{Q}_R$, and $(\mathbf{H}, \mathbf{K}) \vdash \mathcal{Q}_{\mathsf{Nd}}^{\mathsf{X}}$ denotes the event that values generated by the oracle $\mathsf{muCo}_{\mathbf{K}}$ (using $\mathbf{H}$) are consistent with the records in $\mathcal{Q}_{\mathsf{Nd}}^{\mathsf{X}}$. The above expansion is possible since S, R, and $(\mathbf{H}, \mathbf{K})$ are independent. Similarly,

$$\begin{aligned}\Pr[T_{\mathrm{id}} = \mathcal{Q}] = \Pr[\mathbf{H} \vdash \mathcal{Q}_{\mathbf{H}}] \times \Pr[\mathsf{S} \vdash \mathcal{Q}_{\mathsf{S}}] \times \Pr[\mathsf{R} \vdash \mathcal{Q}_R] \\ \times \Pr[(\$\mathsf{Cons}, \mathbf{K}) \vdash \mathcal{Q}_{\mathsf{Nd}}^{\mathsf{X}}] \times \Pr[\mathbf{P} \leftarrow \mathsf{KGen}],\end{aligned} \tag{14}$$

since $(\$\mathsf{Cons}, \mathbf{K})$ is independent from $\mathbf{H}$. Gathering Eqs. (13) and (14) yields

$$\frac{\Pr[T_{\mathrm{re}} = \mathcal{Q}]}{\Pr[T_{\mathrm{id}} = \mathcal{Q}]} = \frac{\Pr[(\mathbf{H}, \mathbf{K}) \vdash \mathcal{Q}_{\mathsf{Nd}}^{\mathsf{X}} \mid \mathbf{H} \vdash \mathcal{Q}_{\mathbf{H}}]}{\Pr[(\$\mathsf{Cons}, \mathbf{K}) \vdash \mathcal{Q}_{\mathsf{Nd}}^{\mathsf{X}}]}, \tag{15}$$

and the problem thus reduces to bounding $\Pr[(\mathbf{H},\mathbf{K}) \vdash \mathcal{Q}^{\mathsf{X}}_{\mathsf{Nd}} \mid \mathbf{H} \vdash \mathcal{Q}_{\mathbf{H}}]$. Subsequent analyses thus simply follow the same line as the 1st step. The definition of bad transcripts is the same as the first step, resulting in $\Pr[T_{\mathrm{id}} \in \Theta_{\mathrm{bad}}] \leq \varepsilon_\mu + \frac{C\cdot(T+D)}{2^\kappa}$. For a good transcript $\mathcal{Q} = (\mathcal{Q}^{\mathsf{X}}_{\mathsf{Nd}}, \mathcal{Q}_R, \mathcal{Q}_{\mathsf{S}}, \mathcal{Q}_{\mathbf{H}}, \mathbf{P})$, following the same line as the previous step, it can be shown that

$$\Pr\big[(\mathbf{H},\mathbf{K}) \vdash \mathcal{Q}^{\mathsf{X}}_{\mathsf{Nd}} \mid \mathbf{H} \vdash \mathcal{Q}_{\mathbf{H}}\big] = \left(\frac{1}{2^n}\right)^{\left|\{(i_0,p,z,b)\in\mathcal{Q}^{\mathsf{X}}_{\mathsf{Nd}} : p\neq\bot\}\right|}.$$

The above established Eq. (12). Gathering Eqs. (7) and (12) yields Eq. (6) and completes the proof.

4.3 Davies-Meyer-based Trees

Using a block cipher $\mathbf{E} : \{0,1\}^\nu \times \{0,1\}^{wn} \mapsto \{0,1\}^{wn}$, the Davies-Meyer-based derivation function is defined by $\mathbf{Prim}(x,y) := \mathsf{DM}^{\mathbf{E}}(x,y) = \mathbf{E}(x,y) \oplus y$ (therefore, $\lambda = wn$). This setting is interesting when we are to instantiate the tree using (fixed-key) block ciphers in their Davies-Meyer modes.

Security of this model crucially relies on the feeding forward in Davies-Meyer, i.e., feeding the κ-bit secret $\mathsf{right}_\kappa(z)$ forward. However, a bad outputting function of may cancel these bits. In this respect, we additionally require that *the outputting function* of *only relies on the leftmost* $n-\kappa$ *bits of its seed*, i.e., $\mathsf{of}\big(z,I\big) = \mathsf{of}\big(\mathsf{left}_{n-\kappa}\big(z\big),I\big)$ for any inputs $z, I \in \{0,1\}^n$. Our formal results regarding $\mathsf{GGGM}^{\mathsf{DM}^{\mathbf{E}}}$ will emphasize on this additional restriction.

Theorem 2. *Assume using the simulator* S *defined in Fig. 4, and:*

(i) $\mathbf{E} : \{0,1\}^\nu \times \{0,1\}^{wn} \mapsto \{0,1\}^{wn}$ *is modeled as an ideal cipher;*
(ii) $\mathsf{of} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ *is such that* $\mathsf{of}(z,I) = \mathsf{of}(\mathsf{left}_{n-\kappa}(z),I)$ *for any* $z, I \in \{0,1\}^n$*;*
(iii) L *leaks* $\mathsf{left}_{n-\kappa}(\mathsf{Nd}(i_0,p^*))$ *for every intermediate node* $\mathsf{Nd}(i_0,p^*)$*;*
(iv) There exist quantities C *and* ε_μ *such that*

$$\Pr_{\mathbf{P}\leftarrow\mathsf{KGen},\mathcal{L}\leftarrow\mathsf{S}}\big[\mu(\mathcal{L}) > C\big] \leq \varepsilon_\mu. \tag{16}$$

Then, $\mathsf{GGGM}^{\mathsf{DM}^{\mathbf{E}}}$ *is a* (u,T,D,ε)*-*(L,S)*-constrained PRF for the set system* $\mathbb{S}_{\mathsf{pre},\mathsf{GGGM}}$ *of Eq. (3), where*

$$\varepsilon = 2\varepsilon_\mu + \frac{4C\cdot\big(T+D\big)}{2^\kappa}. \tag{17}$$

We also rely on the random intermediate system $(\$\mathsf{Cons}^{\mathsf{S}}, \mathsf{R}^{\mathsf{S}}, \mathbf{E})$ for "relay".

Indistinguishability of $(\mathsf{muCo}^{\mathsf{L}}_{\mathbf{K}}, \mathsf{muEv}^{\mathsf{L}}_{\mathbf{K}}, \mathbf{E})$ and $(\$\mathsf{Cons}^{\mathsf{S}}, \mathsf{R}^{\mathsf{S}}, \mathbf{E})$ View $(\mathsf{muCo}^{\mathsf{L}}_{\mathbf{K}}, \mathsf{muEv}^{\mathsf{L}}_{\mathbf{K}}, \mathbf{E})$ as the real world, and $(\$\mathsf{Cons}^{\mathsf{S}}, \mathsf{R}^{\mathsf{S}}, \mathbf{E})$ the ideal world. We prove the following bound.

$$\left|\Pr\left[\mathfrak{D}^{\mathsf{muCo}^{\mathsf{L}}_{\mathbf{K}},\mathsf{muEv}^{\mathsf{L}}_{\mathbf{K}},\mathbf{E}}=1\right]-\Pr\left[\mathfrak{D}^{\$\mathsf{Cons}^{\mathsf{S}},\mathsf{R}^{\mathsf{S}},\mathbf{E}}=1\right]\right|\leq \varepsilon_{\mu}+\frac{2C\cdot\left(T+D\right)}{2^{\kappa}}. \quad (18)$$

The setting is similar to that studied in Sect. 4.2, except that the random oracle $\mathbf{H}$ is replaced with an ideal cipher $\mathbf{E}:\{0,1\}^{\nu}\times\{0,1\}^{wn}\mapsto\{0,1\}^{wn}$ that can be queried in both forward and backward directions, with $wn=\lambda$. By this, the transcript of $\mathfrak{D}$'s interaction consists of $\mathcal{Q}_{\mathsf{Nd}}$ and $\mathcal{Q}_{\mathbf{E}}$, where $\mathcal{Q}_{\mathbf{E}}=\{(L_1,x_1,y_1),\ldots\}$ records $\mathfrak{D}$'s queries/answers to/from $\mathbf{E}$ (with $(L,x,y)\in\mathcal{Q}_{\mathbf{E}}$ meaning $\mathbf{E}(L,x)=y$), while the transcript $\mathcal{Q}_{\mathsf{Nd}}=\left\{(i_0^{(1)},p^{(1)},z^{(1)},b^{(1)}),\ldots\right\}$, $i_0^{(\ell)}\in\{1,...,u\}$, $p^{(\ell)}\in\mathcal{P}$, $z^{(\ell)}\in\{0,1\}^n\cup\{0,1\}^{n-\kappa}$, $b^{(\ell)}\in\{0,1\}$, is just similar to Sect. 4.2. We also append the κ bit internal secrets to $\mathcal{Q}_{\mathsf{Nd}}$ to have the extended list $\mathcal{Q}^{\mathsf{X}}_{\mathsf{Nd}}$ as in Sect. 4.2, and concentrate on $\mathcal{Q}=(\mathcal{Q}^{\mathsf{X}}_{\mathsf{Nd}},\mathcal{Q}_{\mathbf{E}},\mathbf{P})$ with $\mathcal{Q}^{\mathsf{X}}_{\mathsf{Nd}}=\left((i_0^{(1)},p^{(1)},z^{(1)},b^{(1)}),...\right)$, $i_0^{(\ell)}\in\{1,...,u\}$, $p^{(\ell)}\in\mathcal{P}$, $z^{(\ell)}\in\{0,1\}^n$, $b^{(\ell)}\in\{0,1\}$. Denote by $\mathbf{E}\vdash\mathcal{Q}_{\mathbf{E}}$ the event that a block cipher $\mathbf{E}$ is consistent with the queries/answers in $\mathcal{Q}_{\mathbf{E}}$, i.e., that $\mathbf{E}(L,x)=y$ for all $(L,x,y)\in\mathcal{Q}_{\mathbf{E}}$. Then the probability of $\mathbf{E}\vdash\mathcal{Q}_{\mathbf{E}}$ for an ideal cipher $\mathbf{E}$ (with wn-bit blocks and ν-bit keys) is exactly $\left(\prod_{L\in\{0,1\}^{\nu}}(2^{wn})_{T_L}\right)^{-1}$. where for integers $1\leq b\leq a$, we set $(a)_b=a\cdot(a-1)\cdots(a-b+1)$ with $(a)_0=1$ by convention, and T_L is the number of tuples of the form $(L,\star,\star)$ in $\mathcal{Q}_{\mathbf{E}}$ (thus $\sum_{L\in\{0,1\}^{\nu}}T_L=T$). Therefore, for any attainable transcript $\mathcal{Q}=(\mathcal{Q}^{\mathsf{X}}_{\mathsf{Nd}},\mathcal{Q}_{\mathbf{E}},\mathbf{P})$, the probability that the ideal world is consistent with $\mathcal{Q}$ is

$$\frac{1}{\prod_{L\in\{0,1\}^{\nu}}(2^{wn})_{T_L}}\times\frac{1}{2^{|\mathcal{Q}^{\mathsf{X}}_{\mathsf{Nd}}|n}}\times\Pr\left[\mathbf{P}\leftarrow\mathsf{KGen}\right]. \quad (19)$$

For the real world, we also write $(\mathbf{E},\mathbf{K})\vdash\mathcal{Q}^{\mathsf{X}}_{\mathsf{Nd}}$ to denote the event that the cipher $\mathbf{E}$ and keys $\mathbf{K}$ are consistent with the values in $\mathcal{Q}^{\mathsf{X}}_{\mathsf{Nd}}$. Similarly to Sect. 4.2, the real world probability is exactly

$$\Pr\left[(\mathbf{E},\mathbf{K})\vdash\mathcal{Q}^{\mathsf{X}}_{\mathsf{Nd}}\mid\mathbf{E}\vdash\mathcal{Q}_{\mathbf{E}}\right]\times\frac{1}{\prod_{L\in\{0,1\}^{\nu}}(2^{wn})_{T_L}}\times\frac{1}{2^{un}}\times\Pr\left[\mathbf{P}\leftarrow\mathsf{KGen}\right], \quad (20)$$

and the problem also reduces to bounding $\Pr[(\mathbf{E},\mathbf{K})\vdash\mathcal{Q}^{\mathsf{X}}_{\mathsf{Nd}}\mid\mathbf{E}\vdash\mathcal{Q}_{\mathbf{E}}]$.

Internal evaluation list $\mathcal{Q}^{\mathsf{x}}_{\mathbf{E}}$. As in Sect. 4.2, we will show that, conditioned on $\mathbf{E}\vdash\mathcal{Q}_{\mathbf{E}}$, the event $(\mathbf{E},\mathbf{K})\vdash\mathcal{Q}^{\mathsf{X}}_{\mathsf{Nd}}$ is equivalent to $\mathbf{E}$ satisfying a series of new and distinct equations. The conditions for bad transcripts are essentially defined to ensure these equations. We start by explicitly constructing the list $\mathcal{Q}^{\mathsf{x}}_{\mathbf{E}}$ of such equations. For this, consider any $(i_0,p,z,\star)\in\mathcal{Q}^{\mathsf{X}}_{\mathsf{Nd}}$ with $p=p^*/i_{d'}\neq\perp$. Let $j^*=\lfloor\frac{i_{d'}}{w}\rfloor$ and $p_\ell=p^*/(wj^*+\ell-1)$ $(1\leq\ell\leq w)$. Then the extended list $\mathcal{Q}^{\mathsf{x}}_{\mathbf{E}}$ includes

an "internal evaluation tuple" $(i_0, p^*, j^*, L^*, x^*, y^*)$, where $(z^* = \mathsf{Nd}(i_0, p^*))$

$$\begin{aligned} L^* &:= \mathsf{lf}_{p^*,j^*}\big(\mathsf{left}_{n-\kappa}(z^*), \mathsf{pp}_{i_0}\big), \qquad\qquad x^* := \mathsf{sf}_{p^*,j^*}\big(\mathsf{right}_\kappa(z^*)\big) \\ y^* &:= \big(\mathsf{of}^{-1}\big(z^*, \mathsf{Nd}(i_0, p_1)\big) \parallel \ldots \parallel \mathsf{of}^{-1}\big(z^*, \mathsf{Nd}(i_0, p_w)\big)\big) \ \oplus\ \mathsf{sf}_{p^*,j^*}\big(\mathsf{right}_\kappa(z^*)\big) \end{aligned} \tag{21}$$

Bad transcripts. An extended transcript $\mathcal{Q} = (\mathcal{Q}^{\mathsf{X}}_{\mathsf{Nd}}, \mathcal{Q}_{\mathbf{E}}, \mathbf{P})$ is *bad*, if any of the following conditions is fulfilled:

- (B-1) $\mu \geq C$.
- (B-2) There exist a pair of tuples $((L, x, y), (i_0^*, p^*, j^*, L^*, x^*, y^*)) \in \mathcal{Q}_{\mathbf{E}} \times \mathcal{Q}^{\mathrm{x}}_{\mathbf{E}}$ such that $(L, x) = (L^*, x^*)$, or $(L, y) = (L^*, y^*)$;
- (B-3) There exist distinct $(i_0^*, p^*, j^*, L^*, x^*, y^*), (i_0^{**}, p^{**}, j^{**}, L^{**}, x^{**}, y^{**}) \in \mathcal{Q}^{\mathrm{x}}_{\mathbf{E}}$ with $(L^*, x^*) = (L^{**}, x^{**})$ or $(L^*, y^*) = (L^{**}, y^{**})$.

The bound $\Pr[\text{(B-1)}] \leq \varepsilon_\mu$ also follows from Eq. (16) straightforwardly. For (B-2), consider each choice of $((L, x, y), (i_0^*, p^*, j^*, L^*, x^*, y^*)) \in \mathcal{Q}_{\mathbf{E}} \times \mathcal{Q}^{\mathrm{x}}_{\mathbf{E}}$. By the fact that $\mathsf{right}_\kappa(z^*)$ is uniform in $\{0,1\}^\kappa$ for $(i_0^*, p^*, z^*, \star) \in \mathcal{Q}^{\mathsf{X}}_{\mathsf{Nd}}$, and that $\mathsf{sf}_{p^*,j^*}(\star)$ is injective, the probability to have $x = x^*$ is $1/2^\kappa$.

The other condition $y = y^*$ translates into

$$y = \big(\mathsf{of}^{-1}\big(r^*, \mathsf{Nd}(i_0^*, p_1^*)\big) \ \big\|\ldots\big\|\ \mathsf{of}^{-1}\big(r^*, \mathsf{Nd}(i_0^*, p_w^*)\big)\big) \oplus \mathsf{sf}_{p^*,j^*}\big(\mathsf{right}_\kappa\big(\mathsf{Nd}(i_0^*, p^*)\big)\big),$$

where $p_\ell^* = p^*/(wj^*+\ell-1)$ $(1 \leq \ell \leq w)$ and $r^* = \mathsf{left}_{n-\kappa}\big(\mathsf{Nd}(i_0^*, p^*)\big)$. In the ideal world, $\mathsf{right}_\kappa\big(\mathsf{Nd}(i_0^*, p^*)\big)$ is uniform, and is independent from $\mathsf{left}_{n-\kappa}(\mathsf{Nd}(i_0^*, p^*)), \mathsf{Nd}(i_0^*, p_1), \ldots, \mathsf{Nd}(i_0^*, p^*)$ (since $\mathsf{right}_\kappa\big(\mathsf{Nd}(i_0^*, p^*)\big)$ is the random "dummy" value sampled at the end of the ideal world interaction). By this and by the injectivity of $\mathsf{sf}_{p^*,j^*}(\star)$, the probability to have $y = y^*$ is $1/2^\kappa$.

Finally, note that $L = \mathsf{lf}_{p^*,j^*}\big(\mathsf{left}_{n-\kappa}\big(\mathsf{Nd}(i_0^*, p^*)\big), \mathsf{pp}_{i_0^*}\big)$ is necessary for both $(L, x) = (L^*, x^*)$ and $(L, y) = (L^*, y^*)$. By this, for any $L \in \{0,1\}^\nu$, we define

$$\mathcal{Q}^+_{\mathbf{E}}[L] := \big\{x \in \{0,1\}^{wn} : (L, x, \star) \in \mathcal{Q}_{\mathbf{E}}\big\}, \qquad\qquad \mathcal{Q}^-_{\mathbf{E}}[L] := \big\{y \in \{0,1\}^{wn} : (L, \star, y) \in \mathcal{Q}_{\mathbf{E}}\big\}.$$

Then, the condition (B-2) is equivalent with $\exists (i_0^*, p^*, j^*, L^*, x^*, y^*) \in \mathcal{Q}^{\mathrm{x}}_{\mathbf{E}} : x^* \in \mathcal{Q}^+_{\mathbf{E}}[L^*] \ \vee\ y^* \in \mathcal{Q}^-_{\mathbf{E}}[L^*]$, the probability of which is

$$\begin{aligned} \Pr\big[\text{(B-2)} \mid \neg\text{(B-1)}\big] &\leq \sum_{(i_0^*, p^*, j^*, L^*, x^*, y^*) \in \mathcal{Q}^{\mathrm{x}}_{\mathbf{E}}} \frac{2\big|\mathcal{Q}^+_{\mathbf{E}}[L^*]\big|}{2^\kappa} \\ &\leq \sum_{L \in \{0,1\}^\nu} \ \sum_{(i_0^*, p^*, j^*, L^*, x^*, y^*) \in \mathcal{Q}^{\mathrm{x}}_{\mathbf{E}} : \mathsf{lf}_{p^*,j^*}\left(\mathsf{left}_{n-\kappa}(\mathsf{Nd}(i_0^*, p^*)), \mathsf{pp}_{i_0^*}\right) = L} \frac{2\big|\mathcal{Q}^+_{\mathbf{E}}[L]\big|}{2^\kappa} \\ &\leq \frac{2C \cdot T}{2^\kappa}. \end{aligned}$$

The last inequality follows from $\sum_{L \in \{0,1\}^\nu} \big|\mathcal{Q}^+_{\mathbf{E}}[L^*]\big| = \sum_{L \in \{0,1\}^\nu} \big|\mathcal{Q}^-_{\mathbf{E}}[L^*]\big| = T$.

For (B-3), consider each pair $(i_0^*, p^*, j^*, L^*, x^*, y^*), (i_0^{**}, p^{**}, j^{**}, L^{**}, x^{**}, y^{**}) \in \mathcal{Q}_{\mathbf{E}}^{\mathsf{x}}$. Due to our restriction on sf and lf, if $(i_0^*, p^*) = (i_0^{**}, p^{**})$ then ($j^* \neq j^{**}$ and) $x^* \neq x^{**}$. Thus we could focus on the case $p^* \neq p^{**}$, meaning that the probability to have $\mathsf{sf}_{p^*,j^*}\big(\mathsf{right}_\kappa\big(\mathsf{Nd}(i_0^*, p^*)\big)\big) = \mathsf{sf}_{p^{**},j^{**}}\big(\mathsf{right}_\kappa\big(\mathsf{Nd}(i_0^{**}, p^{**})\big)\big)$ is $1/2^\kappa$ since both $\mathsf{Nd}(i_0^*, p^*)$ and $\mathsf{Nd}(i_0^{**}, p^{**})$ are independent and uniform. On the other side, the equality $y^* = y^{**}$ translates into

$$\begin{aligned}&\big(\mathsf{of}^{-1}\big(r^*, \mathsf{Nd}(i_0^*, p_1^*)\big)\|...\|\mathsf{of}^{-1}\big(r^*, \mathsf{Nd}(i_0^*, p_w^*)\big)\big) \oplus \mathsf{sf}_{p^*,j^*}\big(\mathsf{right}_\kappa\big(\mathsf{Nd}(i_0^*, p^*)\big)\big),\\ =&\big(\mathsf{of}^{-1}\big(r^{**}, \mathsf{Nd}(i_0^{**}, p_1^{**})\big)\|...\|\mathsf{of}^{-1}\big(r^{**}, \mathsf{Nd}(i_0^{**}, p_w^{**})\big)\big) \oplus \mathsf{sf}_{p^{**},j^{**}}\big(\mathsf{right}_\kappa\big(\mathsf{Nd}(i_0^{**}, p^{**})\big)\big),\end{aligned}$$

where $p_\ell^* = p^*/(wj^* + \ell - 1)$ $(1 \leq \ell \leq w)$, $r^* = \mathsf{left}_{n-\kappa}\big(\mathsf{Nd}(i_0^*, p^*)\big)$, $p_\ell^{**} = p^{**}/(wj^{**} + \ell - 1)$ $(1 \leq \ell \leq w)$ and $r^{**} = \mathsf{left}_{n-\kappa}\big(\mathsf{Nd}(i_0^{**}, p^{**})\big)$. In the ideal world, $\mathsf{right}_\kappa\big(\mathsf{Nd}(i_0^*, p^*)\big)$ and $\mathsf{right}_\kappa\big(\mathsf{Nd}(i_0^{**}, p^{**})\big)$ are uniform and independent. By the injectivity of sf, the probability to have $y^* = y^{**}$ is $1/2^\kappa$.

Last, $(L^*, x^*) = (L^{**}, x^{**})$ and $(L^*, y^*) = (L^{**}, y^{**})$ hold only if $\mathsf{lf}_{p^*,j^*}\big(\mathsf{left}_{n-\kappa}(\mathsf{Nd}(i_0^*, p^*)), \mathsf{pp}_{i_0^*}\big) = \mathsf{lf}_{p^{**},j^{**}}\big(\mathsf{left}_{n-\kappa}(\mathsf{Nd}(i_0^{**}, p^{**})), \mathsf{pp}_{i_0^{**}}\big)$. Meanwhile, by our definition Eq. (4) and by $\neg$(B-1), for each pair (p^*, j^*), the number of pairs (p^{**}, j^{**}) satisfying $\mathsf{left}_{n-\kappa}(\mathsf{Nd}(i_0^*, p^*)) = \mathsf{left}_{n-\kappa}(\mathsf{Nd}(i_0^{**}, p^{**}))$ cannot exceed C. By these, the probability $\Pr\big[\text{(B-3)} \mid \neg\text{(B-1)}\big]$ is bounded by

$$\sum_{(i_0^*, p^*, j^*, L^*, x^*, y^*) \in \mathcal{Q}_{\mathbf{E}}^{\mathsf{x}}} \Bigg(\sum_{\substack{(i_0^{**}, p^{**}, j^{**}, L^{**}, x^{**}, y^{**}) \neq (i_0^*, p^*, j^*, L^*, x^*, y^*) : \\ \mathsf{lf}_{p^*,j^*}\left(\mathsf{left}_{n-\kappa}(\mathsf{Nd}(i_0^*, p^*)), \mathsf{pp}_{i_0^*}\right) = \mathsf{lf}_{p^{**},j^{**}}\left(\mathsf{left}_{n-\kappa}(\mathsf{Nd}(i_0^{**}, p^{**})), \mathsf{pp}_{i_0^{**}}\right)}} \frac{2}{2^\kappa} \Bigg)$$

$$\leq \frac{2C \cdot |\mathcal{Q}_{\mathsf{Nd}}^{\mathsf{x}}|}{2^\kappa}.$$

In all, a union bound yields

$$\Pr\big[T_{\mathrm{id}} \in \Theta_{\mathrm{bad}}\big] \leq \varepsilon_\mu + \frac{2C \cdot \big(T + D\big)}{2^\kappa}. \tag{22}$$

Ratio of Probabilities of Good Transcripts. Fix a good transcript $\mathcal{Q}$. The idea resembles Sect. 4.2, concentrating on analyzing $\Pr[(\mathbf{E}, \mathbf{K}) \vdash \mathcal{Q}_{\mathsf{Nd}}^{\mathsf{x}} \mid \mathbf{E} \vdash \mathcal{Q}_{\mathbf{E}}] = \Pr[\mathbf{E} \vdash \mathcal{Q}_{\mathbf{E}}^{\mathsf{x}} \mid \mathbf{E} \vdash \mathcal{Q}_{\mathbf{E}}]$. Let

$$\mathcal{Q}_{\mathbf{E}}^{\mathsf{x}} = \big((i_0^{(1)}, p^{(1)}, j^{(1)}, L^{(1)}, x^{(1)}, y^{(1)}), ..., (i_0^{(D)}, p^{(D)}, j^{(D)}, L^{(D)}, x^{(D)}, y^{(D)})\big)$$

in arbitrary order, then the latter probability can be expressed as

$$\prod_{\ell=1}^{D} \Pr\big[\mathbf{E}(L^{(\ell)}, x^{(\ell)}) = y^{(\ell)} \mid \mathbf{E} \vdash \mathcal{Q}_{\mathbf{E}} \wedge \forall \ell' < \ell : \mathbf{E}(L^{(\ell')}, x^{(\ell')}) = y^{(\ell')}\big].$$

Fix some ℓ. Since the transcript is good, there is no query of the form $(L^{(\ell)}, x^{(\ell)}, \star)$ in $\mathcal{Q}_{\mathbf{E}}$ (since (B-2) does not occur), nor is $\mathbf{E}(L^{(\ell)}, x^{(\ell)})$ determined by the fact that $\mathbf{E}(L^{(\ell')}, x^{(\ell')}) = y^{(\ell')}$ for all $\ell' < \ell$ (since (B-3) does not occur). Similarly by symmetry, there is no query of the form $(L^{(\ell)}, \star, y^{(\ell)})$ in $\mathcal{Q}_{\mathbf{E}}$ (since (B-2) does not occur), nor is

Algorithm $\mathsf{Pu}(K, \mathcal{S})$ // Puncture a set $\mathcal{S} \subseteq \mathcal{X}$
parse $(\mathsf{k}, \mathsf{pp}) \leftarrow K$
$list \leftarrow \emptyset$
Let $\mathcal{P}_c \subseteq \mathcal{P}$ be such that $p \notin \mathcal{S}$ if and only if there exists $p^* \in \mathcal{P}_c$ and p^* is prefix of p
for $p^* \in \mathcal{P}_c$ **do**
  $ckey_{p^*} \leftarrow \mathsf{StepDown}(\mathsf{k}, \mathsf{pp}, p^*)$, $list \leftarrow list \cup (p^*, ckey_{p^*})$
end for
return $K\{\mathcal{S}\} \leftarrow (\mathsf{pp}, list)$

Fig. 5. The Pu algorithm corresponding to $\mathsf{GGGM}^{\mathbf{Prim}}[\kappa, n, \nu, \theta, w, d, \mathsf{sf}, \mathsf{lf}]$.

$\mathbf{E}^{-1}(L^{(\ell)}, y^{(\ell)})$ determined by the fact that $\mathbf{E}(L^{(\ell')}, y^{(\ell')}) = x^{(\ell')}$ for all $\ell' < \ell$ (since (B-3) does not occur). Thus, we have

$$\Pr\big[\mathbf{E}(L^{(\ell)}, x^{(\ell)}) = y^{(\ell)} \mid \mathbf{E} \vdash \mathcal{Q}_{\mathbf{E}} \wedge \forall \ell' < \ell : \mathbf{E}(L^{(\ell')}, x^{(\ell')}) = y^{(\ell')}\big] \geq 1/2^{wn}.$$

for all ℓ. Finally, it also holds $|\mathcal{Q}^{\mathsf{x}}_{\mathbf{E}}| = |\{(i_0, p, z, b) \in \mathcal{Q}^{\mathsf{X}}_{\mathsf{Nd}} : p \neq \bot\}|/w$, which resembles Sect. 4.2. By the above,

$$\begin{aligned}\text{Eq. (20)} &\geq \left(\frac{1}{2^{wn}}\right)^{D} \times \frac{1}{\prod_{L \in \{0,1\}^{\nu}} (2^{wn})_{T_L}} \times \frac{1}{2^{un}} \times \Pr\big[\mathbf{P} \leftarrow \mathsf{KGen}\big] \\ &= \left(\frac{1}{2^{n}}\right)^{|\mathcal{Q}^{\mathsf{X}}_{\mathsf{Nd}}|} \times \frac{1}{\prod_{L \in \{0,1\}^{\nu}} (2^{wn})_{T_L}} \times \Pr\big[\mathbf{P} \leftarrow \mathsf{KGen}\big],\end{aligned}$$

meaning that the probability that the real world is consistent with the transcript is at least Eq. (19). This completes the proof.

Indistinguishability of $(\$\mathsf{Cons}^{\mathsf{S}}, \mathsf{R}^{\mathsf{S}}, \mathbf{E})$ and $(\mathsf{muCo}^{\mathsf{S}}_{\mathbf{K}}, \mathsf{R}^{\mathsf{S}}, \mathbf{E})$ For this step, we view $(\mathsf{muCo}^{\mathsf{S}}_{\mathbf{K}}, \mathsf{R}^{\mathsf{S}}, \mathbf{E})$ as the real world and $(\$\mathsf{Cons}^{\mathsf{S}}, \mathsf{R}^{\mathsf{S}}, \mathbf{E})$ as the ideal. The core step is to establish pseudorandomness of the constrained keys, which combines the ideas of Sect. 4.2 and 4.3. In all, the bound remains.

$$\left|\Pr\big[\mathfrak{D}^{\$\mathsf{Cons}^{\mathsf{S}}, \mathsf{R}^{\mathsf{S}}, \mathbf{E}} = 1\big] - \Pr\big[\mathfrak{D}^{\mathsf{muCo}^{\mathsf{S}}_{\mathbf{K}}, \mathsf{R}^{\mathsf{S}}, \mathbf{E}} = 1\big]\right| \leq \varepsilon_{\mu} + \frac{2C \cdot (T + D)}{2^{\kappa}}. \tag{23}$$

Gathering Eqs. (18) and (23) yields Eq. (17) and completes the proof.

5 Multi-user Leakage PPRF Security of GGGM Trees

While it was believed obvious, we first formally describe the puncturing algorithm Pu of $\mathsf{GGGM}^{\mathbf{Prim}}$ in Fig. 5 for completeness. This specification would also be used in subsequent proof of Theorem 3.

The leakage assumption and simulator S are roughly the same as Sect. 4.1 and Fig. 4. With these, the leakage PPRF security of GGGM is as follows.

Theorem 3. *Assume using the simulator S defined in Fig. 4, and that there exist quantities C and ε_{μ} such that $\Pr_{\mathbf{P} \leftarrow \mathsf{KGen}, \mathcal{L} \leftarrow \mathsf{S}}\big[\mu(\mathcal{L}) > C\big] \leq \varepsilon_{\mu}$. Then:*

– *The tree $\mathsf{GGGM}^{\mathbf{H}}$ built upon a random oracle $\mathbf{H} : \{0,1\}^{\nu+\omega} \mapsto \{0,1\}^{wn}$ is a (u, T, D, ε)-(L, S)-puncturable PRF, where $\varepsilon = 2\varepsilon_{\mu} + \frac{2C \cdot (T+D)}{2^{\kappa}}$;*

– *With the additional restriction that* $\mathsf{of} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ *is such that* $\mathsf{of}(z, I) = \mathsf{of}(\mathsf{left}_{n-\kappa}(z), I)$ *for any* $z, I \in \{0,1\}^n$*, the tree* $\mathsf{GGGM}^{\mathsf{DM}^{\mathbf{E}}}$ *built upon an ideal cipher* $\mathbf{E} : \{0,1\}^\nu \times \{0,1\}^{wn} \mapsto \{0,1\}^{wn}$ *is a* (u, T, D, ε)*-*(L, S)*-puncturable PRF, where* $\varepsilon = 2\varepsilon_\mu + \frac{4C \cdot \left(T+D\right)}{2^\kappa}$.

Proof. A distinguisher $\mathfrak{D}_1$ against the PPRF security of $\mathsf{GGGM}^{\mathbf{Prim}}$ can be transformed to a distinguisher $\mathfrak{D}_2$ against the CPRF security of $\mathsf{GGGM}^{\mathbf{Prim}}$. Indeed, $\mathfrak{D}_2$ runs $\mathfrak{D}_1$ and simulates the $\mathsf{muPu}^{\mathsf{L}}_{\mathbf{K}}$ oracle for $\mathfrak{D}_1$ using its $\mathsf{muCo}^{\mathsf{L}}_{\mathbf{K}}$ oracle, as shown in Fig. 5 (the algorithm $\mathsf{Pu}(K, \mathcal{S})$ calls $\mathsf{StepDown}(K, p^*)$, but as shown in Fig. 2, this internal procedure functions the same as $\mathsf{Co}(K, p^*)$). It is easy to see $\mathfrak{D}_2$'s effective data complexity D and query complexity T are the same as $\mathfrak{D}_1$. The bounds thus follow Theorems 1 and 2.

6 Improving Hierarchical Deterministic Wallets

We first review hierarchical deterministic wallet: its basic ideas in Sect. 6.1, its formalism in Sect. 6.2, and specification of Bip32 in Sect. 6.3. Then, in Sect. 6.4, we introduce mu security definitions for HDWs; in Sect. 6.5, we establish mu security for Bip32 using Theorem 1. Finally, we present improvements in Sect. 6.6.

6.1 Hierarchical Deterministic Wallets

Briefly, a wallet in blockchain consists of a pair of secret and public keys for a digital signature scheme. To transfer assets, the user signs *transactions* (i.e., messages) with its secret key, with the digital identity named "address" of the receiver (e.g., the receiver's public key) embedded in the transactions. A deterministic wallet derives a sequence of *session key pairs* from a single master key and uses distinct public session keys as multiple identities for receiving, in order to achieve anonymity and limit the damage of (session) key exposure.

A hierarchical deterministic wallet (HDW) makes one step further, derives a collection of signing keys from a master and organizes them under an access hierarchy, where each element represents a group of users and each user has its associated keys. Users staying higher in the hierarchy must be able to derive the keys of users on lower levels and to further sign transactions on their behalf. Users on lower levels, however, should not be able to escalate their privileges along the hierarchy, not even when colluding with others. This *hierarchical access control* exactly fits into the manager(s)-departments architecture of large-scale enterprises. It also eases *wallet delegation* and *auditing*, which turns out to be crucial for e-commerce and Decentralized Finance.

Clearly, secret signing keys are central in wallets. To achieve decentralized key managements, a promising approach is to use *threshold signatures* [Des88,Lin19]. In a threshold signature, the secret signing key is divided into shares held by multiple parties. A threshold number (i.e., a subset) of these parties can follow the protocol (typically the TSS protocol [AHS20]) to collaboratively sign cryptocurrency transaction. The protocol is fully decentralized, and the collapse of fewer parties/shares will not incur loss of funding. See [AHS20] for a survey.

Decentralized key managements for HDW, however, require *dividing the master key into shares* and *computing the signing key shares from the master key shares without combining any one*. Thus, multiple parties holding the master key shares have to

evaluate the hierarchical key derivations in MPC. Probably, the best approach is to garble the key derivation functions. The bottleneck is communication overhead, which is mainly determined by the number of AND gates in key derivations. The default Bip32 configuration consumes 4 HMAC-SHA512 executions with nearly 1 million AND gates to derive a concrete signing key from the wallet seed (see Fig. 1). This incurs a heavy communication cost and prohibits the use, especially for mobile users. While some ideas to remedy appear obvious, the soundness is unclear due to the lack of formalism and justification.

6.2 Formalism of Public-underivable HDW

Luzio et al. introduced a model for HDW [LFA20]. Their formalism particularly emphasizes *public-derivability*, i.e., the ability of *generating all public keys of the wallet without relying on any secret information*. As they discussed [LFA20, Sect. 7], public-derivability somewhat contradicts the privacy notion *(transaction) unlinkability*. However, Bip32 [Med18, Security] indeed *insisted on unlinkability rather than public-derivability* (see Sect. 6.4 for elaboration). Bip32 tries to achieve unlinkability-like notion by using *secret* "chain codes", which essentially disables public-derivability. Auditing was intended to be carried out by granting certain secret "chain codes" to the auditor. With these considerations, we henceforth refer to Luzio et al.'s formalism [LFA20, Sect. 5] as *public-derivable HDW*, and resort to a natural relaxation *(pub-underivable) HDW*.[11]

Despite disabling public-derivability, our formalism of HDW still follows [LFA20]. Concretely, let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a directed acyclic graph (DAG) representing the access hierarchy of the HDW, where:

(i) $\mathcal{V} = \{v_\perp, v_{p_1}, v_{p_2}, v_{p_3}...\}$ is the set of vertexes, where $v_\perp$ has indegree 0 (meaning that $v_\perp$ has the highest privilege in the hierarchy defined by $\mathcal{G}$).

(ii) $\mathcal{E}$ is the set of edges, and elements in $\mathcal{E}$ are of the form $(v_p \xrightarrow{i_j} v_{p/i_j})$ indicating an edge from v_p to v_{p/i_j} (of lower privilege) with label i_j. To ease understanding and highlight the concrete hierarchy in $\mathcal{G}$, we use the path notation system of Sect. 3.2 for subscripts. This means a vertex $v_{i_1/i_2/.../i_d}$ is in $\mathcal{V}$ *if and only if* the d edges $(v_\perp \xrightarrow{i_1} v_{i_1}), (v_{i_1} \xrightarrow{i_2} v_{i_1/i_2})..., (v_{i_1/i_2/.../i_{d-1}} \xrightarrow{i_d} v_{i_1/i_2/.../i_d})$ are all in $\mathcal{E}$.

We define the set of descendants $\mathcal{D}\mathsf{esc}(\mathcal{G}, v_p)$ of node v_p to be the set of nodes $v_{p'}$ such that there exists a direct path from v_p to $v_{p'}$ in $\mathcal{G}$. By our notations, this means there exists $p^* \in \mathcal{P}^*$ such that $p' = p/p^*$.

With these, a *(public-underivable) hierarchical deterministic wallet* $\mathsf{Wal} = (\mathsf{Setup}, \mathsf{DPub}, \mathsf{DPriv}, \mathsf{Sign}, \mathsf{Vrfy})$, defined over seed space $\mathcal{S}$ and message space $\mathcal{M}$, is defined in the following way:

- $\mathsf{Setup}(\mathcal{G}, S)$: The deterministic setup algorithm (a.k.a. *master key generation* due to [DFL19]) takes as input an initial seed $S \in \mathcal{S}$ and a DAG $\mathcal{G}$ that has a unique node with indegree 0, and returns the keys $(\mathsf{wsk}_\perp, \mathsf{wpk}_\perp)$ of the node $v_\perp \in \mathcal{V}$ with indegree 0 (with the highest privileges). $(\mathsf{wsk}_\perp, \mathsf{wpk}_\perp)$ are also viewed as the master wallet secret and public keys of the HDW.
- $\mathsf{DPub}(\mathcal{G}, \mathsf{wpk}_p, v_p, v_{p'})$: The *delegated deterministic public derivation* algorithm takes as input the wallet public key wpk_p associated to node $v_p \in \mathcal{V}$ and a target node $v_{p'} \in \mathcal{D}\mathsf{esc}(\mathcal{G}, v_p)$, and outputs the wallet public key $\mathsf{wpk}_{p'}$ of v_p. This functionality fits into the "audit" use case of [Med18, Use cases].

[11] Concurrently to us, Das et al. [DEF+21] also proposed a model for HDW.

– $\mathsf{DPriv}(\mathcal{G}, \mathsf{wsk}_p, v_p, v_{p'})$: The deterministic private derivation algorithm takes as input the wallet secret key wsk_p associated to node $v_p \in \mathcal{V}$ and a target node $v_{p'} \in \mathcal{D}\mathsf{esc}(\mathcal{G}, v_p)$, and outputs the wallet secret key $\mathsf{wsk}_{p'}$ of $v_{p'}$.
– $\mathsf{WSign}(\mathsf{wsk}_p, m)$: The randomized signing algorithm takes as input a message $m \in \mathcal{M}$ and a wallet secret key wsk_p, and outputs a signature σ.
– $\mathsf{WVrfy}(pk, m, \sigma)$: The deterministic verification algorithm takes as input a signature public key pk, a message m, and a signature σ. It outputs 1 (accept) or 0 (reject).

A hierarchical deterministic wallet is *correct*, if any user v_p (that holds the wallet secret key wsk_p) can derive the private and public keys $\mathsf{wsk}_{p'}$ and $\mathsf{wpk}_{p'}$ of any of its descendants $v_{p'} \in \mathcal{D}\mathsf{esc}(\mathcal{G}, v_p)$ and create a valid signature on behalf of $v_{p'}$ (i.e., that passes the verification process against the public key $pk_{p'}$ obtained through public key derivation). We omit the formalism.

6.3 Bip32 HDW and the Underlying GGGM Instance

The Bitcoin Improvement Proposal Bip32 [Med18] uses a double-input function $\mathbf{Prim} : \{0,1\}^{512} \times \{0,1\}^{256} \mapsto \{0,1\}^{512}$. In Fig. 6, we provide a description of Bip32 following the formalism in Sect. 6.2. Briefly speaking,

(i) During $\mathsf{Setup}(\mathcal{G}, S)$, the wallet first invokes **Prim** with the seed S to derive master keys $(ch_\perp, sk_\perp, pk_\perp)$ that are intended to be held by the enterprise, where $(sk_\perp, pk_\perp)$ are the signature secret and public key and mch is an additional secret called *chain code*. Essentially, the chain code functions as secret seeds of pseudorandom primitives. As shown in Fig. 1, $\mathsf{wsk}_\perp = (sk_\perp, ch_\perp)$ and $\mathsf{wpk}_\perp = (pk_\perp, ch_\perp)$ are viewed as the extended private and public keys of the enterprise.
(ii) Given the extended secret key $\mathsf{wsk}_p = (sk_p, ch_p)$ of a node v_p, the CKDpriv algorithm derives the extended secret key $\mathsf{wsk}_{p/i} = (sk_{p/i}, ch_{p/i})$ of a child node $v_{p/i}$ via invoking **Prim** once and calculating a modular addition.
(iii) Given the extended public key $\mathsf{wpk}_p = (pk_p, ch_p)$ of a node v_p, the CKDpub algorithm derives the extended public key $\mathsf{wpk}_{p/i} = (pk_{p/i}, ch_{p/i})$ of a child node $v_{p/i}$, thanks to the homomorphism property between the secret key and public key space.

Note that by the specification, a node v_p specified by $\mathcal{G}$ may still have $\mathsf{wsk}_p = \perp$, although the probability is extremely low. A default configuration for an enterprise with several offices is recommended in [Med18]. The default key tree is essentially a GGGM tree with depth $d = 3$, as illustrated in Fig. 1.

As discussed in Sect. 3.1, the key tree of Bip32 is actually a major motivation of our model. Actually, the definition and analysis of Bip32 can be based on an instance of GGGM that (roughly) takes the 512-bit strings $z_p = [sk_p]_{256} \| ch_p$ as nodes (with chain code ch_p functioning as the secrets). In detail, consider the GGGM instance with parameters as follows.

– Node size $n = 512$, security parameter $\kappa = 256$;
– $\theta \leq 2^{32}$, and $w = 1$, i.e., output size $wn = n = 512$;
– Depth d equals the length of the longest (directed) path in $\mathbf{G}$. In this vein, a (directed) path $v_\perp \xrightarrow{i_1} v_{i_1} \xrightarrow{i_2} \ldots \xrightarrow{i_{d'}} v_{i_1/\ldots/i_{d'}}$ in $\mathcal{G}$ identifies a path $p^* = i_1/\ldots/i_{d'}$ in this tree;
– The root node $\mathsf{Nd}(\perp) = \mathbf{Prim}\big(S, \text{“Bitcoin seed”}\big)$;

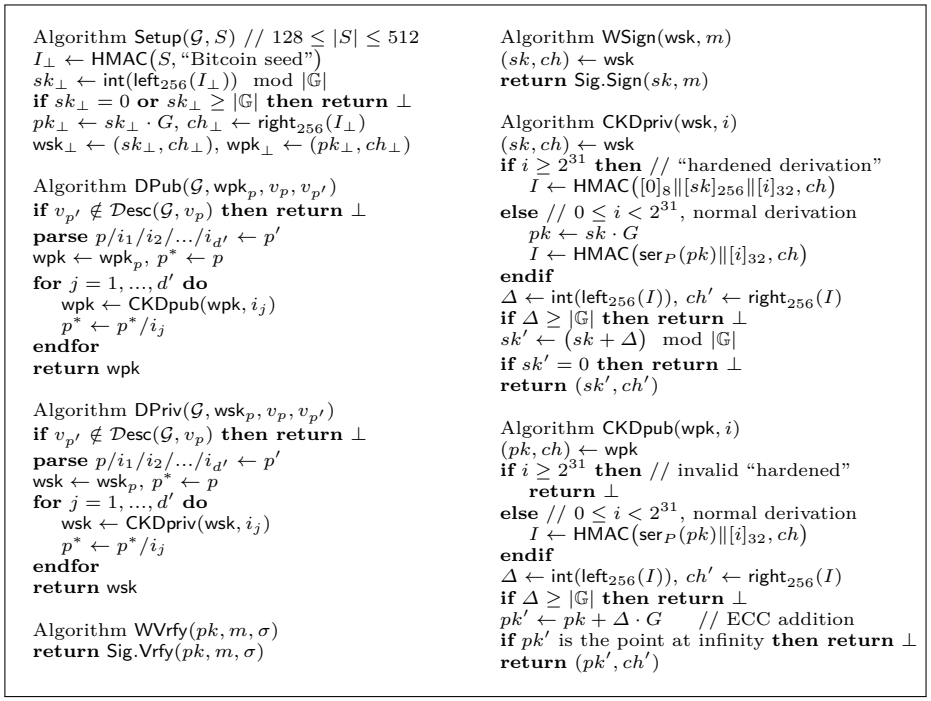

Fig. 6. Specification of Bip32 HDW following the formalism of Sect. 6.2.

– For a node value $z \in \{0,1\}^{512}$ and $(p, j) \in \big(\{\perp\} \cup \{i_1/i_2/.../i_{d'}\}_{d' \in \{1,...,d-1\}}\big) \times \{0, ..., 2^{32}-1\}$, let $r = \mathsf{left}_{256}(z)$, then the seeding, labeling and outputting functions are as follows.

$$\mathsf{sf}_{p,j}\big(\mathsf{right}_{256}(z)\big) = \mathsf{right}_{256}(z) \tag{24}$$

$$\mathsf{lf}_{p,j}(r, \mathsf{pp}) = \begin{cases} [0]_8 \parallel r \parallel [j]_{32} & \text{if } 2^{31} \le j < 2^{32} \\ \mathsf{ser}_P(\mathsf{int}(r) \cdot G) \parallel [j]_{32} & \text{if } 0 \le j < 2^{31} \text{ and } \mathsf{int}(r) \in \mathbb{Z}^+_{|\mathbb{G}|} \\ [0]_8 \parallel r \parallel [j]_{32} & \text{if } 0 \le j < 2^{31} \text{ and } \mathsf{int}(r) \notin \mathbb{Z}^+_{|\mathbb{G}|} \end{cases} \tag{25}$$

$$\mathsf{of}(z, I) = \begin{cases} I & \text{if } \mathsf{int}\big(\mathsf{left}_{256}(I)\big) \ge |\mathbb{G}| \\ \big[\mathsf{int}(r) + \mathsf{int}(\mathsf{left}_{256}(I)) \mod |\mathbb{G}|\big]_{256} \parallel \mathsf{right}_{256}(I) & \text{if } 0 \le \mathsf{int}\big(\mathsf{left}_{256}(I)\big) < |\mathbb{G}| \end{cases} \tag{26}$$

The above GGGM instance is *not exactly* the same as a Bip32 key tree: it may contain nodes $z \in \{0,1\}^{512}$ such that $\mathsf{int}\big(\mathsf{left}_{256}(z)\big) \notin \mathbb{Z}^+_{|\mathbb{G}|}$ cannot be interpreted as a “correct” signature private key (and lf computes differently from Bip32 key tree), while the Bip32 key tree discards such “unparsable” values. But the gap is limited, as will be reflected in the subsequent reduction in Sect. 6.5. In all, the algorithm $\mathsf{DPriv}(\mathcal{G}, \mathsf{wsk}_p, v_p, v_{p'})$ in Fig. 6 can be redefined based on the Co algorithm of the above GGGM instance. In particular, when the “staring” node $v_\perp$ is the root node, the key can be derived via querying the constraining oracle of F. This will be the core idea of our reduction in Sect. 6.5.

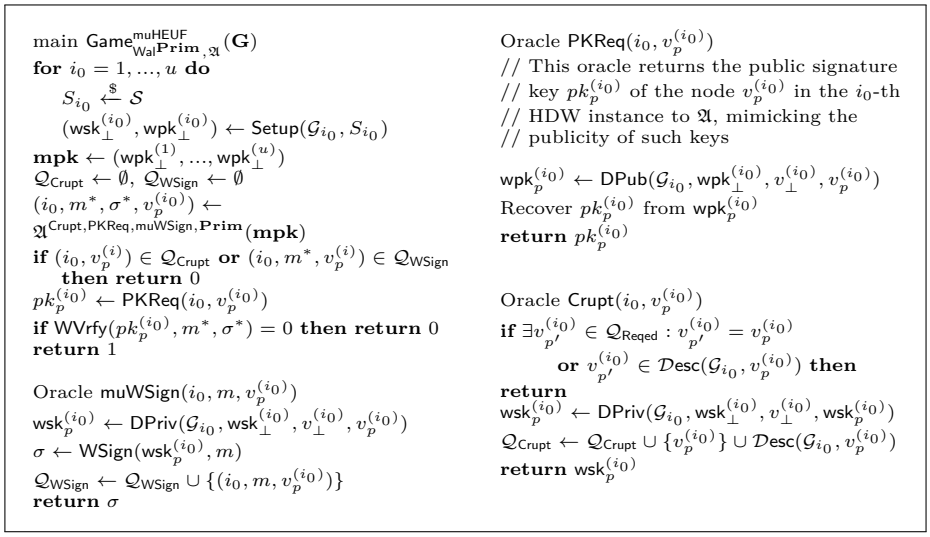

Fig. 7. Multi-user HEUF security game $\mathsf{Game}^{\mathsf{muHEUF}}_{\mathsf{Wal}^{\mathbf{Prim}},\mathfrak{A}}(\mathbf{G})$. The invoked Setup, DPub, DPriv, WSign, and WVrfy belongs to the wallet $\mathsf{Wal}^{\mathbf{Prim}}$.

6.4 mu Security Definitions for Hierarchical Deterministic Wallet

We follow Luzio et al.'s *hierarchical unforgeability* notion [LFA20], which allows an attacker to corrupt an arbitrary number of users (and their descendants) in the hierarchy, and challenges the attacker to forge a signature on behalf of an uncorrupted node. The ability of corruption is formalized by the oracle Crupt in Fig. 7, which also models multiple nodes colluding and sharing their secrets. Below we extend it into the mu setting. Also we consider the case wallet is built upon a public ideal function **Prim**, and write $\mathsf{Wal}^{\mathbf{Prim}}$ to highlight. Another parameter that helps characterizing adversarial power is *the maximal number of allowed sessions*. Concretely, let $\mathbf{G} = (\mathcal{G}_1, ..., \mathcal{G}_u)$ be a sequence of DAGs that defines the configurations for the u wallet users, and let $\mathcal{G}_{i_0} = (\mathcal{V}_{i_0}, \mathcal{E}_{i_0})$ for $i_0 = 1, ..., u$. Then, the maximal number of allowed sessions is defined as $V(\mathbf{G}) := \sum_{i_0=1}^{u} |\mathcal{V}_{i_0}|$. I.e., the DAGs limit the total number of derived keys in the system.

Definition 7 (Multi-user HEUF Security). *An HDW scheme* $\mathsf{Wal}^{\mathbf{Prim}}$ *is* $(u, q_C, q_D, q_S, T, D, t, \varepsilon)$*-multi-user hierarchically existentially unforgeable, if for every sequence of* u *DAGs* $\mathbf{G} = (\mathcal{G}_1, ..., \mathcal{G}_u)$ *such that* $V(\mathbf{G}) \leq D$ *and any adversary* $\mathfrak{A}$ *making* q_C *queries to the* Crupt *oracle,* q_D *queries to the* PKReq *oracle,* q_S *queries to the signing oracle* muWSign*, and* T *queries to the ideal primitive* **Prim***, it holds*

$$\Pr\left[\mathsf{Game}^{\mathsf{muHEUF}}_{\mathsf{Wal}^{\mathbf{Prim}},\mathfrak{A}}(\mathbf{G}) = 1\right] \leq \varepsilon,$$

where the experiment $\mathsf{Game}^{\mathsf{muHEUF}}_{\mathsf{Wal}^{\mathbf{Prim}},\mathfrak{A}}(\mathbf{G})$ *is defined in Fig. 7.*

Luzio et al. mentioned *hierarchical unlinkability* without a detailed formalism [LFA20]. This notion intends to capture that there is no "non-trivial" relation between distinct child public keys of the same node. We formalize this idea in the mu setting, allowing an attacker to corrupt an arbitrary number of users/sessions in the hierarchy, and

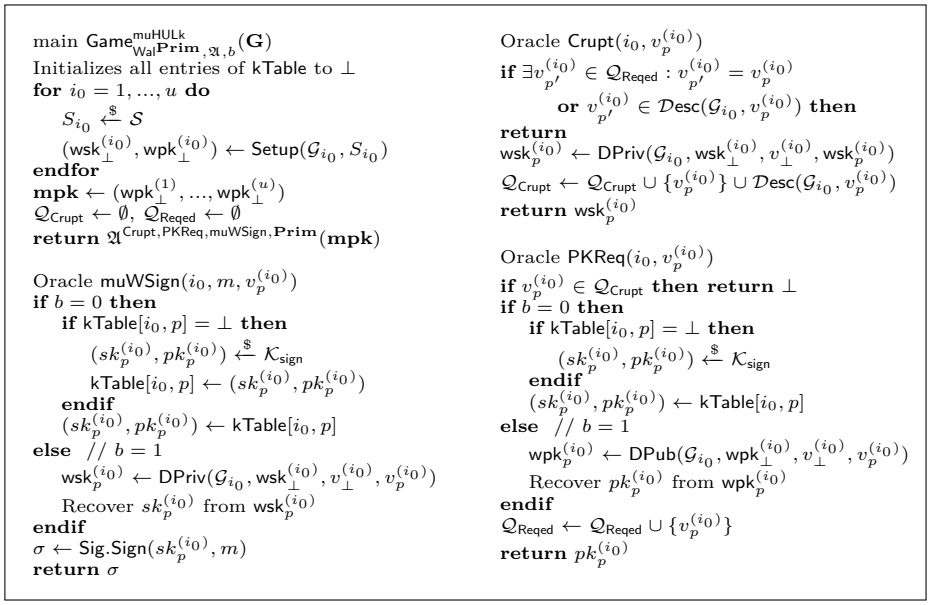

Fig. 8. Multi-user muHULk security game $\mathsf{Game}^{\mathsf{muHULk}}_{\mathsf{Wal}^{\mathbf{Prim}},\mathfrak{A},b}(\mathbf{G})$.

challenging it to distinguish public signature keys of uncorrupted nodes from random points. This is formalized by the oracle PKReq in Fig. 8, which returns the true requested public signature key when $b = 1$, and random public key when $b = 0$. These two definitions essentially capture desired security in the intended use case "Per-office balances" in [Med18].

Definition 8 (Multi-user HULk Security). *An HDW scheme* $\mathsf{Wal}^{\mathbf{Prim}}$ *is* $(u, q_C, q_D, q_S, T, D, t, \varepsilon)$*-multi-user hierarchically unlinkable, if for every sequence of* u *DAGs* $\mathbf{G} = (\mathcal{G}_1, ..., \mathcal{G}_u)$ *such that* $V(\mathbf{G}) \leq D$ *and any adversary* $\mathfrak{A}$ *making* q_C *queries to the* Crupt *oracle,* q_D *queries to the oracle* PKReq*,* q_S *queries to the signing oracle* muWSign*, and* T *queries to the ideal primitive* **Prim***, it holds*

$$\left|\Pr\left[\mathsf{Game}^{\mathsf{muHULk}}_{\mathsf{Wal}^{\mathbf{Prim}},\mathfrak{A},0}(\mathbf{G}) = 1\right] - \Pr\left[\mathsf{Game}^{\mathsf{muHULk}}_{\mathsf{Wal}^{\mathbf{Prim}},\mathfrak{A},1}(\mathbf{G}) = 1\right]\right| \leq \varepsilon,$$

where the experiment $\mathsf{Game}^{\mathsf{muHULk}}_{\mathsf{Wal}^{\mathbf{Prim}},\mathfrak{A},b}(\mathbf{G})$ *is defined in Fig. 8.*

6.5 Multi-user Security of Bip32

For simplicity and clearness, we assume using parameters close to the 256-bit elliptic curve secp256k1 recommended in [Med18]. In the standard Bip32, the double-input function **Prim** is instantiated by HMAC, i.e., $\mathbf{Prim}(x, y) := \mathsf{HMAC}(y, x)$ (using y as the key of HMAC). We note that the first input x is not always secret in Bip32. By Dodis et al. [DRST12, Theorem 4.4], with such parameters, HMAC instantiates a keyed FIL random oracle. Therefore, this subsection considers $\mathbf{Prim}(x, y) = \mathbf{KH}(x, y)$ for a keyed RO $\mathbf{KH} : \{0,1\}^{512} \times \{0,1\}^{256} \mapsto \{0,1\}^{512}$, and our results are formally stated as follows.

Theorem 4. *The* $\mathsf{Bip32}^{\mathbf{KH}}$ *HDW built upon a keyed random oracle* $\mathbf{KH}: \{0,1\}^{512} \times \{0,1\}^{256} \mapsto \{0,1\}^{512}$ *is* $(u, q_C, q_D, q_S, T, D, t, \varepsilon)$*-multi-user hierarchically existentially unforgeable, where (assuming* $D \leq |\mathbb{G}|/2$*)*

$$\varepsilon \leq \frac{2u(T+D)}{2^s} + \frac{u^2}{2^s} + \frac{D}{|\mathbb{G}|} + \frac{2^9 \cdot (T+D)}{2^{256}} + D \times \left(1 - \frac{|\mathbb{G}|-1}{2^{256}}\right). \tag{27}$$

The $\mathsf{Bip32}^{\mathbf{KH}}$ *HDW built upon a keyed random oracle* $\mathbf{KH}: \{0,1\}^{512} \times \{0,1\}^{256} \mapsto \{0,1\}^{512}$ *and a* $(q_S, q_S, O(t+D), \varepsilon_{\mathsf{muUFCMA}})$-mu *unforgeable digital signature scheme is* $(u, q_C, q_D, q_S, T, D, t, \varepsilon)$*-multi-user hierarchically existentially unforgeable, where (assuming* $D \leq |\mathbb{G}|/2$*)*

$$\varepsilon \leq \frac{u(T+D)}{2^s} + \frac{u^2}{2^s} + \frac{D}{|\mathbb{G}|} + \frac{2^9 \cdot (T+D)}{2^{256}} + D \times \left(1 - \frac{|\mathbb{G}|-1}{2^{256}}\right) + \varepsilon_{\mathsf{muUFCMA}}. \tag{28}$$

When the 256-bit elliptic curse domain parameters recommended in [Res10, Sect. 2.4] is used, it can be checked that $D \times \left(1 - \frac{|\mathbb{G}|-1}{2^{256}}\right) \leq D/2^{127}$ (as claimed in [Med18]) and $|\mathbb{G}|/2 \approx 2^{255}$.

As mentioned, with the parameters in Bip32, HMAC instantiates a keyed FIL random oracle $\mathbf{KH}: \{0,1\}^{512} \times \{0,1\}^{256} \mapsto \{0,1\}^{512}$ with 256-bit security [DRST12, Theorem 4.4]. Thus, this is not the bottleneck. Otherwise, Eq. (27) indicates computational security of $\min\{247, s - \log_2 u\}$ bits, as long as the total number D of active Bip32 sessions across the world does not exceed $\min\{2^s/u, |\mathbb{G}|/2, 2^{127}\}$. Whereas Eq. (28) indicate computational security of $\min\{247, s - \log_2 u, f(q_S)\}$ bits, as long as the signature scheme delivers $f(q_S)$ bit muUFCMA security (Definition 2). Importantly, this means the amount of computations needed for attacks does *not* decrease as the number D of "active sessions" increases.

Consider the case of $s \geq 256$. In the su setting, i.e., $u = 1$, the amount of computations needed to break unlinkability with a notable success probability is 2^{247}. The $\log_2 u$ bit loss in the multi-user setting is inevitable, as it matches the effort for guessing one out of s-bit seeds $S_1, ..., S_u$. Fortunately, u reflects the number of "enterprises" using Bip32 excluding their sub-accounts, and would not be too large.

On the other hand, the mu security loss due to the signature scheme has to be seriously addressed, as it straightforwardly determines the concrete unforgeability security *even in the* su *setting*. Any signature with sufficient mu security could be employed: see Sect. 2. For example, using 256-bit secret keys, the mu security of Schnorr is of 128 bits [KMP16], giving rise to $\min\{247, s - \log_2 u, 128\}$ bits computational security. We are not aware of ECDSA variants with good mu bounds, which is a natural open problem.

In the remaining of this subsection, we devote to prove the unlinkability and unforgeability claims in turn.

Unlinkability. Consider any adversary $\mathfrak{A}_1$ against the muHULk security of $\mathsf{Bip32}^{\mathbf{KH}}$. By construction, the security game $\mathsf{G}_1 = \mathsf{Game}^{\mathsf{muHULk}}_{\mathsf{Wal}^{\mathbf{KH}}, \mathfrak{A}_1, 0}(\mathbf{G})$ first invokes $\mathsf{Setup}(\mathcal{G}_{i_0}, S_{i_0})$ for $i_0 = 1, ..., u$, which derives the u master keys or root nodes $\mathsf{Nd}(1, \perp) = \mathsf{wsk}^{(1)}_{\perp} = \mathbf{KH}\big(S_1, \text{"Bitcoin seed"}\big)$, ..., $\mathsf{Nd}(u, \perp) = \mathsf{wsk}^{(u)}_{\perp} = \mathbf{KH}\big(S_u, \text{"Bitcoin seed"}\big)$ for the u instances. Similarly for the game $\mathsf{G}^*_1 = \mathsf{Game}^{\mathsf{muHULk}}_{\mathsf{Wal}^{\mathbf{KH}}, \mathfrak{A}_1, 1}(\mathbf{G})$. We first replace these nodes by uniform and independent strings. After this, we construct an adversary $\mathfrak{A}_2$ that simulates the muHULk security game using the CPRF oracles in front of $\mathfrak{A}_1$. The advantage of $\mathfrak{A}_1$ is then bounded by Theorem 1. We describe the two steps in the subsequent two paragraphs in turn.

The initial derivation We start by replacing the u 512-bit root nodes $\mathsf{Nd}(1,\bot),...,\mathsf{Nd}(u,\bot)$ by u 512-bit independent and uniform strings, and obtain an intermediate game G_2. The gap between G_1 and G_2 is at most $u(T+D)/2^s+u^2/2^{s+1}$, i.e., the mu PRF security of $\mathbf{KH}\big(S,\text{“Bitcoin seed”}\big)$ (viewing S as the secret key):

(i) The u master key derivation calls $\mathsf{Nd}(i_0,\bot)=\mathbf{KH}\big(S_{i_0},\text{“Bitcoin seed”}\big)$, $i_0=1,...,u$, are fresh and produce u random strings $\mathsf{Nd}(1,\bot),...,\mathsf{Nd}(u,\bot)$, as long as none of the u arguments $\big(S_1,\text{“Bitcoin seed”}\big),...,$ $\big(S_u,\text{“Bitcoin seed”}\big)$ collides with the other queries to $\mathbf{KH}$ in this system. Here “other queries” include queries made by both $\mathfrak{A}_1$ and the subsequent derivations in the $\mathsf{Bip32}$ instances. The number of the former type is at most T, while the number of the latter type does not exceed $V(\mathbf{G})\le D$. Therefore, the probability of this type of collision event is at most $u(T+D)/2^s$;

(ii) The u outputs $\mathsf{Nd}(1,\bot),...,\mathsf{Nd}(u,\bot)$ are independent if and only if the u seeds $S_1,...,S_u$ are collision free. This probability is at most $\binom{u}{2}\times\frac{1}{2^s}\le u^2/2^{s+1}$.

Similarly, we obtain an intermediate game G_2^* from $\mathsf{G}_1^*=\mathsf{Game}^{\mathsf{muHULk}}_{\mathsf{Wal}[\mathbf{KH}],\mathfrak{A}_1,1}(\mathbf{G})$ via replacing the u 512-bit root nodes by random, with a gap at most $u(T+D)/2^s+u^2/2^{s+1}$.

G_2 *versus* G_2^* *using Theorem 1* For the remaining argument, $\mathfrak{A}_2$ runs $\mathfrak{A}_1$ and simulates the oracles as follows, and outputs $\mathfrak{A}_1$'s decision bit at the end.

- Upon $\mathfrak{A}_1$ querying $\mathsf{Crupt}(i_0,v^{(i_0)}_{i_1/.../i_{d'}})$, $\mathfrak{A}_2$ queries $\mathsf{muCo}^{\mathsf{L}}_{\mathbf{K}}(i_0,i_1/.../i_{d'})$ to have the constrained key $\mathsf{Nd}(i_0,i_1/.../i_{d'})$ and the leakages $\mathsf{left}_{256}(\mathsf{Nd}(i_0,\bot)),...,\mathsf{left}_{256}(\mathsf{Nd}(i_0,i_1/.../i_{d'}))$. $\mathfrak{A}_2$ returns $\bot$ to $\mathfrak{A}_1$, if any of the $d'+1$ 256-bit integers $\mathsf{int}\big(\mathsf{left}_{256}(\mathsf{Nd}(i_0,\bot))\big),...,\mathsf{int}\big(\mathsf{left}_{256}(\mathsf{Nd}(i_0,i_1/.../i_{d'}))\big)$ is not in $\mathbb{Z}^+_{|\mathbb{G}|}$ (recall from Sect. 6.3 that this is consistent with the actual $\mathsf{Bip32}$ specification). Otherwise, $\mathfrak{A}_2$ sets $sk^{(i_0)}_{i_1/.../i_{d'}}\leftarrow\mathsf{int}\big(\mathsf{left}_{256}\big(\mathsf{Nd}(i_0,i_1/.../i_{d'})\big)\big)$, $ch^{(i_0)}_{i_1/.../i_{d'}}\leftarrow\mathsf{right}_{256}\big(\mathsf{Nd}(i_0,i_1/.../i_{d'})\big)$, and passes $\mathsf{wsk}^{(i_0)}_{i_1/.../i_{d'}}=(sk^{(i_0)}_{i_1/.../i_{d'}},ch^{(i_0)}_{i_1/.../i_{d'}})$ to $\mathfrak{A}_1$.
- Upon $\mathfrak{A}_1$ querying $\mathsf{muWSign}(i_0,m,v^{(i_0)}_{i_1/.../i_{d'}})$, $\mathfrak{A}_2$ queries $\mathsf{muCo}^{\mathsf{L}}_{\mathbf{K}}(i_0,i_1/.../i_{d'})$ for the constrained key $\mathsf{Nd}(i_0,i_1/.../i_{d'})$ and the leakages $\mathsf{left}_{256}(\mathsf{Nd}(i_0,\bot)),...,\mathsf{left}_{256}(\mathsf{Nd}(i_0,i_1/.../i_{d'}))$. If any of these $d'+1$ integers $\mathsf{int}\big(\mathsf{left}_{256}(\mathsf{Nd}(i_0,\bot))\big),...,\mathsf{int}\big(\mathsf{left}_{256}(\mathsf{Nd}(i_0,i_1/.../i_{d'}))\big)$ is not in $\mathbb{Z}^+_{|\mathbb{G}|}$ then $\mathfrak{A}_2$ returns $\bot$ to $\mathfrak{A}_1$. Otherwise, $\mathfrak{A}_2$ computes the chain code $ch^{(i_0)}_{i_1/.../i_{d'}}\leftarrow\mathsf{right}_{256}\big(\mathsf{Nd}(i_0,i_1/.../i_{d'})\big)$ and the secret signing key $sk^{(i_0)}_{i_1/.../i_{d'}}\leftarrow\mathsf{int}\big(\mathsf{left}_{256}(\mathsf{Nd}(i_0,i_1/.../i_{d'}))\big)$ and returns $\mathsf{WSign}\big((sk^{(i_0)}_{i_1/.../i_{d'}},ch^{(i_0)}_{i_1/.../i_{d'}}),m\big)$ to $\mathfrak{A}_1$.
- Upon $\mathfrak{A}_1$ querying $\mathsf{PKReq}(i_0,v^{(i_0)}_{i_1/.../i_{d'}})$, $\mathfrak{A}_2$ pinpoints the longest path in $\mathcal{G}_{i_0}$ that contains the node $v^{(i_0)}_j$. Formally, $\mathfrak{A}_2$ pinpoints $v^{(i_0)}_0\xrightarrow{i_1}v^{(i_0)}_{i_1}\xrightarrow{i_2}...\xrightarrow{i_{d'}}v^{(i_0)}_{i_1/.../i_{d'}}\xrightarrow{i_{d'+1}}...\xrightarrow{i_d}v^{(i_0)}_{i_1/.../i_d}$ such that the outdegree of $v^{(i_0)}_{i_1/.../i_d}$ is zero. $\mathfrak{A}_1$ then queries $\mathsf{muEv}^{\mathsf{L}}_{\mathbf{K}}(i_0,i_1/.../i_d)$[12] to have $\mathsf{left}_{256}(\mathsf{Nd}(i_0,i_1/.../i_d))$ and the leakages $\mathsf{left}_{256}(\mathsf{Nd}(i_0,\bot)),...,\mathsf{left}_{256}(\mathsf{Nd}(i_0,i_1/.../i_{d'})),...$ Again, if any of the $d'+1$ integers $\mathsf{int}\big(\mathsf{left}_{256}(\mathsf{Nd}(i_0,\bot))\big),...,\mathsf{int}\big(\mathsf{left}_{256}(\mathsf{Nd}(i_0,i_1/.../i_{d'}))\big)$ is not in $\mathbb{Z}^+_{|\mathbb{G}|}$, then $\mathfrak{A}_2$ returns $\bot$ to $\mathfrak{A}_1$. Otherwise, $\mathfrak{A}_2$ computes $sk^{(i_0)}_{i_1/.../i_{d'}}\leftarrow\mathsf{int}\big(\mathsf{left}_{256}(\mathsf{Nd}(i_0,i_1/.../i_{d'}))\big)$ and $pk^{(i_0)}_{i_1/.../i_{d'}}\leftarrow sk^{(i_0)}_{i_1/.../i_{d'}}\cdot G$ and returns $pk^{(i_0)}_{i_1/.../i_{d'}}$ to $\mathfrak{A}_1$.
- Upon $\mathfrak{A}_1$ querying $\mathbf{KH}$, $\mathfrak{A}_2$ simply relays the query and response.

[12] $\mathfrak{A}_2$ cannot simply query $\mathsf{muCo}^{\mathsf{L}}_{\mathbf{K}}(i_0,i_1/.../i_{d'})$, otherwise it cannot react to $\mathfrak{A}_1$ querying $\mathsf{PKReq}(i_0,v^{(i_0)}_{i_1/.../i_{d'}/.../i_d})$ later.

When $\mathfrak{A}_2$ is interacting with the real world $(\mathsf{muCo}^{\mathsf{L}}_{\mathbf{K}}, \mathsf{muEv}^{\mathsf{L}}_{\mathbf{K}}, \mathbf{KH})$, $\mathfrak{A}_1$ is playing with the game G_2: the actions are exactly the same, despite the gap between the GGGM instance and the Bip32 key tree.

On the other hand, when $\mathfrak{A}_2$ is interacting with the ideal $(\mathsf{muCo}^{\mathsf{S}}_{\mathbf{K}}, \mathsf{R}^{\mathsf{S}}, \mathbf{KH})$, a public key $pk^{(i_0)}_{i_1/\dots/i_{d'}}$ returned by $\mathfrak{A}_2$ due to $\mathfrak{A}_1$ querying $\mathsf{PKReq}(i_0, v^{(i_0)}_{i_1/\dots/i_{d'}})$ query is computed via $pk^{(i_0)}_{i_1/\dots/i_{d'}} \leftarrow \mathsf{int}(r^{(i_0)}_{i_1/\dots/i_{d'}}) \cdot G$, where $r^{(i_0)}_{i_1/\dots/i_{d'}} \xleftarrow{\$} \{0,1\}^{256}$ is a block of simulated leakage (it corresponds to $\mathsf{left}_{256}(\mathsf{Nd}(i_0, i_1/\dots/i_{d'}))$ in the real world). This resembles the game G_2^*, except that $\mathfrak{A}_2$ may respond $\mathfrak{A}_1$ with $\perp$. Denote this event by "$\mathfrak{A}_2^{\mathsf{muCo}^{\mathsf{S}}_{\mathbf{K}}, \mathsf{R}^{\mathsf{S}}, \mathbf{KH}}$ returns $\perp$". Then, as long as this event does not occur, the computed key $pk^{(i_0)}_{i_1/\dots/i_{d'}}$ is uniformly distributed in $\mathbb{G}$. Moreover, the key computed due to any other query $\mathsf{PKReq}(i'_0, v^{(i'_0)}_{i'_1/\dots/i'_{d''}})$ has $pk^{(i'_0)}_{i'_1/\dots/i'_{d''}} \leftarrow \mathsf{int}(r^{(i'_0)}_{i'_1/\dots/i'_{d''}}) \cdot G$ and $r^{(i'_0)}_{i'_1/\dots/i'_{d''}}$ is independent from $r^{(i_0)}_{i_1/\dots/i_{d'}}$. Therefore, as long as the event "$\mathfrak{A}_2^{\mathsf{muCo}^{\mathsf{S}}_{\mathbf{K}}, \mathsf{R}^{\mathsf{S}}, \mathbf{KH}}$ returns $\perp$" does not occur, $\mathfrak{A}_2$ perfectly emulates the game G_2^* in front of $\mathfrak{A}_1$, i.e.,

$$\left|\Pr\left[\mathsf{G}_2^* = 1\right] - \Pr\left[\mathfrak{A}_2^{\mathsf{muCo}^{\mathsf{S}}_{\mathbf{K}}, \mathsf{R}^{\mathsf{S}}, \mathbf{KH}} = 1\right]\right| \leq \Pr\left[\mathfrak{A}_2^{\mathsf{muCo}^{\mathsf{S}}_{\mathbf{K}}, \mathsf{R}^{\mathsf{S}}, \mathbf{KH}} \text{ returns } \perp\right] \leq D \times \left(1 - \frac{|\mathbb{G}| - 1}{2^{256}}\right),$$

and thus

$$\left|\Pr\left[\mathsf{G}_2^* = 1\right] - \Pr\left[\mathsf{G}_2 = 1\right]\right| \leq \left|\Pr\left[\mathfrak{A}_2^{\mathsf{muCo}^{\mathsf{L}}_{\mathbf{K}}, \mathsf{muEv}^{\mathsf{L}}_{\mathbf{K}}, \mathbf{KH}} = 1\right] - \Pr\left[\mathfrak{A}_2^{\mathsf{muCo}^{\mathsf{S}}_{\mathbf{K}}, \mathsf{R}^{\mathsf{S}}, \mathbf{KH}} = 1\right]\right| + D \times \left(1 - \frac{|\mathbb{G}| - 1}{2^{256}}\right).$$

Concrete bounds It remains to calculate the concrete bounds. Regardless of the adversarial strategy, it can be seen the effective data complexity cannot exceed $V(\mathbf{G}) \leq D$. For the quantity C, using the uniformness of the private/public keys (in the ideal world) and a multi-collision argument (we defer the details to the next paragraph), when $C = 256 = 2^8$ and $D \leq |\mathbb{G}|/2$ (so that $2D/|\mathbb{G}| \leq 1$), it can be shown that $\Pr\left[\mu(\mathcal{I}) \geq C\right] \leq D/2|\mathbb{G}|$. Then, using Theorem 1, we obtain

$$\left|\Pr\left[\mathsf{G}_2^* = 1\right] - \Pr\left[\mathsf{G}_2 = 1\right]\right| \leq \frac{D}{|\mathbb{G}|} + \frac{2^9 \cdot (T + D)}{2^{256}} + D \times \left(1 - \frac{|\mathbb{G}| - 1}{2^{256}}\right). \tag{29}$$

Gathering this with the gaps between G_1, G_2, G_1^* and G_2^* (which is $2 \times \left(u(T+D)/2^s + u^2/2^{s+1}\right)$) yields Eq. (27).

Proof of $\Pr\left[\mu(\mathcal{I}) \geq C\right] \leq D/2|\mathbb{G}|$ For the quantity $\mu(\mathcal{L})$, we will rely on the uniformness of the private/public keys. In detail, consider any $(i_0, p, j) \neq (i'_0, p', j')$. Then,

- Case 1: $0 \leq j < 2^{31}$, whereas $2^{31} \leq j' < 2^{32}$. Then it is impossible to have $\mathsf{If}_{p,j}\left(\mathsf{left}_{256}(\mathsf{Nd}(i_0, p)), \mathsf{pp}_{i_0}\right) = \mathsf{If}_{p',j'}\left(\mathsf{left}_{256}(\mathsf{Nd}(i'_0, p')), \mathsf{pp}_{i'_0}\right)$, since the former has leftmost byte $[0]_8$ while the latter has $[2]_8$ or $[3]_8$;
- Case 2: $2^{31} \leq j < 2^{32}$, $0 \leq j' < 2^{31}$. Then $\mathsf{If}_{p,j}\left(\mathsf{left}_{256}(\mathsf{Nd}(i_0, p)), \mathsf{pp}_{i_0}\right) \neq \mathsf{If}_{p',j'}\left(\mathsf{left}_{256}(\mathsf{Nd}(i'_0, p')), \mathsf{pp}_{i'_0}\right)$ which resembles Case 1;
- Case 3: $0 \leq j, j' < 2^{31}$. Then $\mathsf{If}_{p,j}\left(\mathsf{left}_{256}(\mathsf{Nd}(i_0, p)), \mathsf{pp}_{i_0}\right) \neq \mathsf{If}_{p',j'}\left(\mathsf{left}_{256}(\mathsf{Nd}(i'_0, p')), \mathsf{pp}_{i'_0}\right)$ holds with probability at most $\Pr\left[sk, sk' \xleftarrow{\$} \mathbb{Z}_{|\mathbb{G}|} : sk = sk'\right] = 1/|\mathbb{G}|$;
- Case 4: $2^{31} \leq j, j' < 2^{32}$. Then $\mathsf{If}_{p,j}\left(\mathsf{left}_{256}(\mathsf{Nd}(i_0, p)), \mathsf{pp}_{i_0}\right) \neq \mathsf{If}_{p',j'}\left(\mathsf{left}_{256}(\mathsf{Nd}(i'_0, p')), \mathsf{pp}_{i'_0}\right)$ holds with probability at most $\Pr\left[pk, pk' \xleftarrow{\$} \mathbb{G} : \mathsf{ser}_P(pk) = \mathsf{ser}_P(pk')\right]$. The map $\mathsf{ser}_P : \mathbb{G} \mapsto \{0,1\}^{264}$ is bijective, and thus the probability is $1/|\mathbb{G}|$.

By the above, in any case, the probability to have $\mathsf{If}_{p,j}\big(\mathsf{left}_{256}(\mathsf{Nd}(i_0,p)),\mathsf{pp}_{i_0}\big) = \mathsf{If}_{p',j'}\big(\mathsf{left}_{256}(\mathsf{Nd}(i'_0,p')),\mathsf{pp}_{i'_0}\big)$ does not exceed $1/|\mathbb{G}|$. Thus, for any integer $C \geq 2$,

$$\begin{aligned}\Pr\big[\mu(\mathcal{L}) \geq C\big] = \Pr\Big[\exists (i_0^{(1)},p^{(1)},j^{(1)}),\ldots,(i_0^{(C)},p^{(C)},j^{(C)}) : \mathsf{If}_{p^{(1)},j^{(1)}}\big(\mathsf{left}_{256}(\mathsf{Nd}(i_0^{(1)},p^{(1)})),\mathsf{pp}_{i_0^{(1)}}\big) \\ = \ldots = \mathsf{If}_{p^{(C)},j^{(C)}}\big(\mathsf{left}_{256}(\mathsf{Nd}(i_0^{(C)},p^{(C)})),\mathsf{pp}_{i_0^{(C)}}\big)\Big] \\ \leq \binom{D}{C}\cdot\Big(\frac{1}{|\mathbb{G}|}\Big)^{C-1}.\end{aligned}$$

For $C = 256$, using $|\mathbb{G}| < 2^{256}$ in our context and as long as $D \leq |\mathbb{G}|/2$ (so that $2D/|\mathbb{G}| \leq 1$), it further holds

$$\Pr\big[\mu(\mathcal{L}) \geq C\big] \leq \frac{|\mathbb{G}|}{C!}\Big(\frac{D}{|\mathbb{G}|}\Big)^{C} \leq \frac{1}{C!}\Big(\frac{2D}{|\mathbb{G}|}\Big)^{256} \leq \frac{1}{256!}\frac{2D}{|\mathbb{G}|} \leq \frac{D}{2|\mathbb{G}|}.$$

Unforgeability. Let $\mathsf{G}_1 = \mathsf{Game}^{\mathsf{muHEUF}}_{\mathsf{Wal}[\mathbf{KH}],\mathfrak{A}_1}(\mathbf{G})$ be the real hierarchical unforgeability game. We modify G_1 by replacing the oracles PKReq and $\mathsf{muWSign}$ with the oracles PKReq and $\mathsf{muWSign}$ with $b = 0$ in Fig. 8. Denote by G_2 the obtained modified game. It is easy to see that, the gap between G_1 and G_2 is the already established unlinkability bound, i.e.,

$$\Big|\Pr\big[\mathsf{G}_2 = 1\big] - \Pr\big[\mathsf{G}_1 = 1\big]\Big| \leq \frac{2u(T+D)}{2^s} + \frac{u^2}{2^s} + \frac{D}{|\mathbb{G}|} + \frac{2^9\cdot(T+D)}{2^{256}} + D \times \Big(1 - \frac{|\mathbb{G}|-1}{2^{256}}\Big).$$

Signing keys generated in the game G_2 are independent and uniformly distributed. Thus, the forgery probability $\Pr\big[\mathsf{G}_2 = 1\big]$ is bounded by the *multi-user existential unforgeability security* of the signature in use (see Definition 2). Informally, consider an adversary $\mathfrak{A}_2$ having access to q_S signing oracles of the signature instantiated with q_S independent secret keys, and it makes q_S signing queries and runs in time $O(t_{\mathfrak{A}_1})$ and succeeds as long as it forges for *any* of the q_S keys. If the signature is $(q_S, q_S, O(t+D), \varepsilon_{\mathsf{muUFCMA}})$-mu unforgeable, then the success probability of $\mathfrak{A}_2$ is bounded by $\varepsilon_{\mathsf{muUFCMA}}$. These establish the bound in Eq. (28).

6.6 Improving Prim for Bip32, and Performance of MPC Implementations

Since the security of Bip32 (variants) simply follow from our results on GGGM, sound improvements addressing Lindell's question [Lin19] become clear. In detail, the initial seed S is of $128 \leq s \leq 512$ bits. The string "Bitcoin seed" is of 96 bits. However, as the security of the entire system cannot exceed 256 bits due to the limitation $\kappa = 256$—and in fact, practical uses typically adopt $s = 256$,—we focus on the case $128 \leq s \leq 256$, and pad the seed S to $S\|[0]_{296-s}$, i.e., of 296 bits. With these considerations, we describe our Bip32 variants using a general function $\mathbf{Prim} : \{0,1\}^{296} \times \{0,1\}^{256} \to \{0,1\}^{256}$.

(i) The HMAC-call in $\mathsf{Setup}(\mathcal{G},S)$ is replaced by $I_\perp \leftarrow \mathbf{Prim}\big(S,\text{"Bitcoin seed"}\big)$;
(ii) The HMAC-calls in $\mathsf{CKDpriv}(\mathsf{wsk},i)$ are replaced by $I \leftarrow \mathbf{Prim}\big([0]_8\|[sk]_{256}\|[i]_{32},ch\big)$ and $I \leftarrow \mathbf{Prim}\big(\mathsf{ser}_P(pk)\|[i]_{32},ch\big)$ correspondingly;
(iii) The HMAC-call in $\mathsf{CKDpub}(\mathsf{wpk},i)$ is replaced by $I \leftarrow \mathbf{Prim}\big(\mathsf{ser}_P(pk)\|[i]_{32},ch\big)$.

We then propose two instantiations of **Prim** with less AND gates as follows.

- **SHACAL instantiation** $\mathsf{Bip32}^{\mathsf{SHACAL3}}$: define the function $\mathbf{Prim}(x, y) := \mathsf{SHACAL3}(x, y\|[0]_{256}) \oplus (y\|[0]_{256})$, i.e., using a single call to $\mathsf{SHACAL3} : \{0,1\}^{1024} \times \{0,1\}^{512} \mapsto \{0,1\}^{512}$, the block cipher underlying SHA512. The reliability is essentially the same as $\mathsf{Bip32}^{\mathsf{HMAC}}$, since the two wallets rely on the same assumption (i.e., the security of SHACAL3).
- **Fast instantiation** $\mathsf{Bip32}^{\mathsf{kp800}}$: define $\mathbf{Prim}(x, y) := \mathsf{Trunc}_{288}\big(\textsc{Keccak}\text{-}p[800, 11](x\|y\|[0]_{248})\big)$, i.e., using a call to the permutation $\textsc{Keccak}\text{-}p[800, 11]$, which is a member of the $\textsc{Keccak}\text{-}p$ permutation family [BDP+17] with number of rounds halved (this follows the SHA3 designers, and its still has security margin of 6 rounds [BDP+17]). To shorten notations, we abbreviate $\textsc{Keccak}\text{-}p[800, 11]$ as kp800.

The security analysis of $\mathsf{Bip32}^{\mathsf{kp800}}$ just follows $\mathsf{Bip32}^{\mathbf{KH}}$ in Sect. 6.5, since truncating $\textsc{Keccak}\text{-}p[800, 11]$ also yields an FIL RO with 248-bit security [CLL19, Theorem 1]. By these, $\mathsf{Bip32}^{\mathsf{kp800}}$ achieves $\min\{247, s - \log_2 u\}$ bits unlinkability and $\min\{247, s - \log_2 u, f(q_S)\}$ unforgeability (as long as we model the permutation kp800 as a *public random 800-bit permutation*). On the other hand, analysis of $\mathsf{Bip32}^{\mathsf{SHACAL3}}$ follows $\mathsf{Bip32}^{\mathbf{KH}}$, except that the terms in Eq. (29) are replaced with terms from Theorem 2. According to Theorems 1 and 2, the concrete bounds differ by only a factor of 2. Thus, $\mathsf{Bip32}^{\mathsf{SHACAL3}}$ achieves $\min\{246, s - \log_2 u\}$ bits unlinkability and $\min\{246, s - \log_2 u, f(q_S)\}$ unforgeability. Concrete mu security of the two improved instantiations are thus comparable with the original Bip32 standard.

We benchmark the two-party protocols based on garbled circuit which securely compute various instantiations of Bip32. Our implementations focus on the (most widely deployed) *default configuration* of the *key tree* [Med18] (see Fig. 1). In particular, we consider the process of deriving the shares of a session key $sk_{i_1/i_2/i_3}$ from the shares of the seed S. As discussed and as shown in Fig. 1, this consists of 4 **Prim** executions (in the MPC manner). We use EMP-Toolkit [WMK16] as the backend of our implementations, and choose the state-of-the-art garbling schemes with semi-honest security. Our experiments are performed in a virtual machine with an Intel(R) Core(TM) i5-1038NG7 CPU at 2.0GHZ with localhost communication, and the performance is reported in Table 1. By the results, $\mathsf{Bip32}^{\mathsf{SHACAL3}}$ achieves much better performance while retaining the same reliability as $\mathsf{Bip32}^{\mathsf{HMAC}}$, while $\mathsf{Bip32}^{\mathsf{kp800}}$ achieves the best performance with a moderate security margin.

Table 1. Performance of Bip32 instantiations. # AND presents the number of involved AND gates, and roughly match theoretical results on 4 **Prim** executions. They deliver similar mu security: see Sect. 6.5 and 6.6.

Scheme	# AND	Time (ms)	Comm. (KB)	Unlink. Sec. (bit)	Note
$\mathsf{Bip32}^{\mathsf{HMAC}}$	944628	87	29790	$\min\{247, s - \log_2 u\}$	Standard
$\mathsf{Bip32}^{\mathsf{SHACAL3}}$	245172	33	7937	$\min\{246, s - \log_2 u\}$	Reliability
$\mathsf{Bip32}^{\mathsf{kp800}}$	49908	19	1835	$\min\{247, s - \log_2 u\}$	High perform.

7 Improving Function Secret Sharing

The state-of-art. Function secret sharing of point functions [GI14] crucially relies on GGM trees functioning as PPRFs. The closest provably secure construction [GKWY20]

instantiates the classical GGM with the length-doubling PRG $G^{\mathsf{AES}_{fk}}(s) := \mathsf{AES}_{fk}(s) \oplus s \| \mathsf{AES}_{fk}(s \oplus [1]_{128}) \oplus s \oplus [1]_{128}$, where AES_{fk} is *the* AES *using a fixed, publicly-known key* fk (as discussed in Sect. 3.1). To characterize its concrete mu security, we model AES as an ideal cipher $\mathbf{E} : \{0,1\}^{wn} \times \{0,1\}^{wn} \mapsto \{0,1\}^{wn}$. Then, this state-of-the-art $\mathsf{FBTr}^{\mathsf{DM}^{\mathsf{AES}}}$ (i.e., *FSS's Binary Tree*), is an instance of $\mathsf{GGGM}^{\mathsf{DM}^{\mathbf{E}}}$ given in Sect. 4.3, with $\mathbf{Prim}(x, y) := \mathsf{AES}_x(y) \oplus y$ and parameters as follows.

- Node size equals the security parameter $n = \kappa = 128$. Note that in our model, this means *there is no "leakage" at all*, and we are essentially in the *classical puncturable PRF setting*;
- $\theta = 2$, and $w = 1$, i.e., branch number $w\theta = 2$, and output size $wn = n = 128$;
- $\mathsf{sf}_{p,j}\big(\mathsf{right}_\kappa(z)\big) = \mathsf{sf}_{p,j}\big(z\big) = z \oplus [j]_{128}$ (which is indeed injective);
- $\mathsf{pp} = fk$ which is a fixed constant, and $\mathsf{lf}_{p,j}\big(\bot, \mathsf{pp}\big) = fk$, i.e., labels don't depend on p, j at all. The restriction $j \neq j' \Rightarrow (\mathsf{lf}_{p,j}\big(\bot, \mathsf{pp}\big), \mathsf{sf}_{p,j}\big(z\big)) \neq (\mathsf{lf}_{p,j'}\big(\bot, \mathsf{pp}\big), \mathsf{sf}_{p,j'}\big(z\big))$ holds since $\mathsf{sf}_{p,j}\big(z\big) = z \oplus [j]_{128} \neq z \oplus [j']_{128} = \mathsf{sf}_{p,j'}\big(z\big)$;
- For any $z, I \in \{0,1\}^{128}$, $\mathsf{of}(z, I) = I$. This satisfies restriction (ii), i.e., $\mathsf{of}(z, I) = \mathsf{of}\big(\mathsf{left}_{128-128}(z), I\big) = I$.
- Depth $d = \log_2 N_{\text{leaf}}$, where N_{leaf} is the desired number of leaves.

FSS uses a $\mathsf{FBTr}^{\mathsf{DM}^{\mathsf{AES}}}$ instance as a *puncturable PRF (PPRF)*: see Definition 5. In detail, assume that the MPC system executes u FSS instances using independent keys $K_1, ..., K_u$. For $i_0 = 1, ..., u$, the i_0-th FSS instance generates a *punctuated key* $K_{i_0}\{p_{i_0}\}$ for a single punctured point $p_{i_0} \in \{0,1\}^d$, i.e., an entity holding the punctured key $K_{i_0}\{p_{i_0}\}$ is able to evaluate exactly $|\mathcal{X}| - 1$ functions values, *except for* $\mathsf{FBTr}^{\mathsf{DM}^{\mathsf{AES}}}.\mathsf{Ev}(K_{i_0}, p_{i_0})$. The i_0-th punctured key $K_{i_0}\{p_{i_0}\}$ can be written in $128d$ bits. The effective data complexity is $2ud$, since (it is easy to count) the number of AES calls in each tree is exactly $2d$.

It remains to determine the quantity C. Unfortunately, the label $\mathsf{lf}_{p,j}\big(\bot, \mathsf{pp}\big) = fk$ is a constant, and we can only expect $C = 2ud$ and $\varepsilon_\mu = \Pr[\mu(\mathcal{L}) > C] = 0$. Injecting all the above parameters into Theorem 2, we conclude that $\mathsf{FBTr}^{\mathsf{DM}^{\mathsf{AES}}}$ is a (u, T, D, ε)-puncturable PRF with effective data complexity $D = 2ud$ and

$$\varepsilon \leq \frac{8udT + 16(ud)^2}{2^{128}}. \tag{30}$$

As a concrete example, consider the 2PC discussed in the introduction: assuming $N_{\text{2pc}} = 2^{14}$ 2PC protocol instances and each protocol instantiating $N_{\text{tr}} = 2^{26}$ FSS or $\mathsf{FBTr}^{\mathsf{DM}^{\mathsf{AES}}}$ tree instances with depth $d \leq 16$. Then, $u = N_{\text{2pc}} \times N_{\text{tr}} = 2^{40}$ and $8ud = 2^{47}$, indicating security up to $T \approx 2^{128}/2^{47} = 2^{81}$ AES "queries" or computations (i.e., a theoretical degradation up to 36.7%). Such computations are hardly infeasible.

Improving mu **security and flexibility.** We first address the mu security degradation, which mainly requires to overcome the factor $u = N_{\text{2pc}} \cdot N_{\text{tr}}$ due to the numerous instances. Following Sect. 6.6, we define the public parameter $\mathsf{pp} := \mathsf{IV}$ for $\mathsf{IV} \xleftarrow{\$} \{0,1\}^{128}$ that is *picked at uniform during the setup of every FSS instance*, and propose to use $\mathsf{AES}_{\mathsf{IV}}$ instead of AES_{fk}. Then, labels $\mathsf{lf}_{\star,\star}\big(\bot, \mathsf{pp}_{i_0}\big) = \mathsf{IV}_{i_0}$ and $\mathsf{lf}_{\star,\star}\big(\bot, \mathsf{pp}_{i'_0}\big) = \mathsf{IV}_{i'_0}$ in distinct trees collide only if $\mathsf{IV}_{i_0} = \mathsf{IV}_{i'_0}$. Thus, for any threshold $t \geq 2$, if there is no t-collision $\mathsf{IV}_{i_0^{(1)}} = ... = \mathsf{IV}_{i_0^{(t)}}$ among the u trees, then the number of tree instances with their IVs equaling a certain value is at most $t-1$. As all IVs are uniform and independent, the probability to have a t-collision is $\binom{u}{t}/2^{128(t-1)} \leq \frac{u^t}{t! \cdot 2^{128(t-1)}}$.

Then, observing that increasing branches trades punctured key size for computations and improves flexibility, we propose to "naturally" increase the "parallelization degree" of $\mathsf{FBTr}^{\mathsf{DM}^{\mathsf{AES}}}$ to $\theta \geq 3$, i.e., using large-expansion PRG $G(s) = \mathsf{AES}_{\mathsf{IV}}(s) \oplus s \| \mathsf{AES}_{\mathsf{IV}}(s \oplus [1]_{128}) \oplus s \oplus [1]_{128} \| ... \| \mathsf{AES}_{\mathsf{IV}}(s \oplus [\theta-1]_{128}) \oplus s \oplus [\theta-1]_{128}$. We denote this by $\mathsf{FMTr}^{\mathsf{DM}^{\mathsf{AES}}}$, meaning *FSS's Multi-branch Tree*.

- The depth decreases to $d = \log_\theta N_{\text{leaf}}$;
- The punctured key for a single tree has $128(\theta-1)\log_\theta N_{\text{leaf}} + 128$ bits (including the 128-bit $\mathsf{pp} = \mathsf{IV}$);
- The effective data complexity $D = u\theta \log_\theta N_{\text{leaf}}$.

By the above, for $C = (t-1)\theta \log_\theta N_{\text{leaf}}$, $\Pr[\mu(\mathcal{L}) > C] \leq \frac{u^t}{t! \cdot 2^{128(t-1)}}$, meaning that $\mathsf{FMTr}^{\mathsf{DM}^{\mathsf{AES}}}$ is a (u, T, D, ε)-puncturable PRF, with

$$\varepsilon \leq \frac{2u^t}{t! \cdot 2^{128(t-1)}} + \frac{4(t-1)\theta T \cdot \log_\theta N_{\text{leaf}} + 4(t-1)u(\theta \log_\theta N_{\text{leaf}})^2}{2^{128}}. \tag{31}$$

Eq. (31) holds for *any* threshold t. Since the parameters u, T, N_{leaf} are incomparable, it is difficult to conclude on optimizing t. But certain choices already yield satisfactory bounds. E.g., with $t = 5$, Eq. (31) becomes

$$\varepsilon \leq \frac{u^5}{2^{517}} + \frac{16\theta T \cdot \log_\theta N_{\text{leaf}} + 16u(\theta \log_\theta N_{\text{leaf}})^2}{2^{128}}, \tag{32}$$

indicating security up to $T \approx 2^{112}$ computations and $u \approx 2^{103}$ users.

More concretely, when $\theta = 2$, Eq. (32) indicates security up to $\approx 2^{128}/2^9 = 2^{119}$ computations (which is nearly optimal for $\kappa = 128$) and running the FSS protocol $u \leq 2^{103}$ times. The additional computation compared to the state-of-art FSS protocol is the mere generation of a 128-bit (pseudo)random IV, which is negligible. The punctured key is $128 \cdot \log_2 N_{\text{leaf}} + 128$ bits. Setting θ to 4, Eq. (32) indicates the same security, whereas the computation cost for each input is reduced by 50% due to the halved depth at the expense of a 50% blow-up in the punctured key size. We list the parameters in Table 2 for clarity.

Table 2. Comparison with prior works. The execution time is benchmarked on an AWS machine of type `m5.large`. N_{leaf} represents the number of leaves in the generated FSS trees. For clearness, the mu security column demonstrates numerical results for the specific case $N_{\text{leaf}} = 2^{16}$ and $u = N_{2\text{pc}} \times N_{\text{tr}} = 2^{40}$ trees.

Scheme	# **Prim**	Cost of **Prim** (ns)	Total time (ns)	Seed size (Bytes)	mu sec. (bits)
State-of-art $\mathsf{FBTr}^{\mathsf{DM}^{\mathsf{AES}}}$	N_{leaf}	3.5	$3.5N_{\text{leaf}}$	$16 \log_2 N_{\text{leaf}}$	81
$\mathsf{FMTr}^{\mathsf{DM}^{\mathsf{AES}}}$ with $\theta = 2$	N_{leaf}	3.5	$3.5N_{\text{leaf}}$	$16 \log_2 N_{\text{leaf}}$	119
$\mathsf{FMTr}^{\mathsf{DM}^{\mathsf{AES}}}$ with $\theta = 4$	$N_{\text{leaf}}/3$	7	$2.3N_{\text{leaf}}$	$24 \log_2 N_{\text{leaf}}$	119

In Table 2, we compare the performance of our protocol against the state-of-the-art. $\mathsf{FMTr}^{\mathsf{DM}^{\mathsf{AES}}}$ with $\theta = 2$ has the same overhead as $\mathsf{FBTr}^{\mathsf{DM}^{\mathsf{AES}}}$, while $\mathsf{FMTr}^{\mathsf{DM}^{\mathsf{AES}}}$ with $\theta = 4$ incurs a 50% increase in seed size (and communication) but 1.5× faster computation. In major applications such as MPC based on preprocessed correlation [BCG+19a], the communication overhead due to the GGM punctuated key is less than 10%. Thus, using $\mathsf{FMTr}^{\mathsf{DM}^{\mathsf{AES}}}$ with $\theta = 4$ accelerates computations much with only insignificant communication overheads.

Acknowledgments

We thank the anonymous reviewers for their insightful suggestions. Chun Guo was supported by the National Key Research and Development Program of China under Grant No. 2022YFA1004900, the National Natural Science Foundation of China (Grant No. 62002202) and the Taishan Scholars Program (for Young Scientists) of Shandong. Xiao Wang was supported by NSF awards #2016240 and #2236819. Yu Yu was supported by the National Natural Science Foundation of China (Grant Nos. 62125204 and 92270201), the National Key Research and Development Program of China (Grant No. 2018YFA0704701), and the Major Program of Guangdong Basic and Applied Research (Grant No. 2019B030302008). Yu Yu also acknowledges the support from the XPLORER PRIZE.

References

ADE+20. Nabil Alkeilani Alkadri, Poulami Das, Andreas Erwig, Sebastian Faust, Juliane Krämer, Siavash Riahi, and Patrick Struck. Deterministic wallets in a quantum world. In *ACM Conf. on Computer and Communications Security (CCS) 2020*, pages 1017–1031. ACM Press, 2020.

AGKK19. Myrto Arapinis, Andriana Gkaniatsou, Dimitris Karakostas, and Aggelos Kiayias. A formal treatment of hardware wallets. In *Financial Cryptography and Data Security (FC)*, LNCS, pages 426–445. Springer, 2019.

AHS20. Jean-Philippe Aumasson, Adrian Hamelink, and Omer Shlomovits. A survey of ECDSA threshold signing. Cryptology ePrint Archive, Report 2020/1390, 2020. https://eprint.iacr.org/2020/1390.

BBM00. Mihir Bellare, Alexandra Boldyreva, and Silvio Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In *Advances in Cryptology—Eurocrypt 2000*, volume 1807 of *LNCS*, pages 259–274. Springer, 2000.

BBT16. Mihir Bellare, Daniel J. Bernstein, and Stefano Tessaro. Hash-function based PRFs: AMAC and its multi-user security. In *Advances in Cryptology—Eurocrypt 2016, Part I*, volume 9665 of *LNCS*, pages 566–595. Springer, 2016.

BCG+19a. Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Peter Rindal, and Peter Scholl. Efficient two-round OT extension and silent non-interactive secure computation. In *ACM Conf. on Computer and Communications Security (CCS) 2019*, pages 291–308. ACM Press, 2019.

BCG+19b. Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators: Silent OT extension and more. In *Advances in Cryptology—Crypto 2019, Part III*, volume 11694 of *LNCS*, pages 489–518. Springer, 2019.

BCGI18. Elette Boyle, Geoffroy Couteau, Niv Gilboa, and Yuval Ishai. Compressing vector OLE. In *ACM Conf. on Computer and Communications Security (CCS) 2018*, pages 896–912. ACM Press, 2018.

BDP+17. Guido Bertoni, Joan Daemen, Michaël Peeters, Gilles Van Assche, and Ronny Van Keer. Team Keccak, 2017.

Ber15. Daniel J. Bernstein. Multi-user Schnorr security, revisited. Cryptology ePrint Archive, Report 2015/996, 2015. https://eprint.iacr.org/2015/996.

BGI14. Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In *Intl. Conference on Theory and Practice of Public Key Cryptography*, LNCS, pages 501–519. Springer, 2014.

BGI15. Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing. In *Advances in Cryptology—Eurocrypt 2015, Part II*, volume 9057 of *LNCS*, pages 337–367. Springer, 2015.

BGI16. Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing: Improvements and extensions. In *ACM Conf. on Computer and Communications Security (CCS) 2016*, pages 1292–1303. ACM Press, 2016.

BHT18. Priyanka Bose, Viet Tung Hoang, and Stefano Tessaro. Revisiting AES-GCM-SIV: Multi-user security, faster key derivation, and better bounds. In *Advances in Cryptology—Eurocrypt 2018, Part I*, volume 10820 of *LNCS*, pages 468–499. Springer, 2018.

BLW17. Dan Boneh, Kevin Lewi, and David J. Wu. Constraining pseudorandom functions privately. In *Intl. Conference on Theory and Practice of Public Key Cryptography 2017, Part II*, LNCS, pages 494–524. Springer, 2017.

BMRS21. Carsten Baum, Alex J. Malozemoff, Marc B. Rosen, and Peter Scholl. Mac'n'cheese: Zero-knowledge proofs for boolean and arithmetic circuits with nested disjunctions. In *Advances in Cryptology—Crypto 2021, Part IV*, LNCS, pages 92–122. Springer, 2021.

BT16. Mihir Bellare and Björn Tackmann. The multi-user security of authenticated encryption: AES-GCM in TLS 1.3. In *Advances in Cryptology—Crypto 2016, Part I*, volume 9814 of *LNCS*, pages 247–276. Springer, 2016.

BW13. Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In *Advances in Cryptology—Asiacrypt 2013, Part II*, volume 8270 of *LNCS*, pages 280–300. Springer, 2013.

CBM15. Henry Corrigan-Gibbs, Dan Boneh, and David Mazières. Riposte: An anonymous messaging system handling millions of users. In *IEEE Symp. Security and Privacy 2015*, pages 321–338. IEEE, 2015.

CDMP05. Jean-Sébastien Coron, Yevgeniy Dodis, Cécile Malinaud, and Prashant Puniya. Merkle-Damgård revisited: How to construct a hash function. In *Advances in Cryptology—Crypto 2005*, volume 3621 of *LNCS*, pages 430–448. Springer, 2005.

CK16. Aloni Cohen and Saleet Klein. The GGM function family is a weakly one-way family of functions. In *9th Theory of Cryptography Conference—TCC 2016*, LNCS, pages 84–107. Springer, 2016.

CLL19. Wonseok Choi, ByeongHak Lee, and Jooyoung Lee. Indifferentiability of truncated random permutations. In *Advances in Cryptology—Asiacrypt 2019, Part I*, LNCS, pages 175–195. Springer, 2019.

CS14. Shan Chen and John P. Steinberger. Tight security bounds for key-alternating ciphers. In *Advances in Cryptology—Eurocrypt 2014*, volume 8441 of *LNCS*, pages 327–350. Springer, 2014.

CT21. Yu Long Chen and Stefano Tessaro. Better security-efficiency trade-offs in permutation-based two-party computation. In *Advances in Cryptology - ASIACRYPT 2021*, pages 275–304, 2021.

DEF+21. Poulami Das, Andreas Erwig, Sebastian Faust, Julian Loss, and Siavash Riahi. The Exact Security of BIP32 Wallets. In *ACM Conf. on Computer and Communications Security (CCS) 2021*. ACM Press, 2021.

Des88. Yvo Desmedt. Society and group oriented cryptography: A new concept. In *Advances in Cryptology—Crypto 1987*, LNCS, pages 120–127. Springer, 1988.

DEs16. Jack Doerner, David Evans, and abhi shelat. Secure stable matching at scale. In *ACM Conf. on Computer and Communications Security (CCS) 2016*, pages 1602–1613. ACM Press, 2016.

DFL19. Poulami Das, Sebastian Faust, and Julian Loss. A formal treatment of deterministic wallets. In *ACM Conf. on Computer and Communications Security (CCS) 2019*, pages 651–668. ACM Press, 2019.

DIO21. Samuel Dittmer, Yuval Ishai, and Rafail Ostrovsky. Line-Point Zero Knowledge and Its Applications. In *2nd Conference on Information-Theoretic Cryptography (ITC 2021)*. Schloss Dagstuhl, 2021.

DRST12. Yevgeniy Dodis, Thomas Ristenpart, John P. Steinberger, and Stefano Tessaro. To hash or not to hash again? (In)differentiability results for H^2 and HMAC. In *Advances in Cryptology—Crypto 2012*, volume 7417 of *LNCS*, pages 348–366. Springer, 2012.

Ds17. Jack Doerner and abhi shelat. Scaling ORAM for secure computation. In *ACM Conf. on Computer and Communications Security (CCS) 2017*, pages 523–535. ACM Press, 2017.

Exo21. Exodus. Preliminary Offering Circular dated February 26, 2021. https://www.sec.gov/Archives/edgar/data/1821534/000114036121006439/nt10013846x8_1a.htm, 2021.

FKPR14. Georg Fuchsbauer, Momchil Konstantinov, Krzysztof Pietrzak, and Vanishree Rao. Adaptive security of constrained PRFs. In *Advances in Cryptology—Asiacrypt 2014, Part II*, volume 8874 of *LNCS*, pages 82–101. Springer, 2014.

GGM86. Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.

GI14. Niv Gilboa and Yuval Ishai. Distributed point functions and their applications. In *Advances in Cryptology—Eurocrypt 2014*, volume 8441 of *LNCS*, pages 640–658. Springer, 2014.

GKW+20. Chun Guo, Jonathan Katz, Xiao Wang, Chenkai Weng, and Yu Yu. Better concrete security for half-gates garbling (in the multi-instance setting). In *Advances in Cryptology—Crypto 2020, Part II*, LNCS, pages 793–822. Springer, 2020.

GKWY20. Chun Guo, Jonathan Katz, Xiao Wang, and Yu Yu. Efficient and secure multiparty computation from fixed-key block ciphers. In *IEEE Symp. Security and Privacy 2020*, pages 825–841. IEEE, 2020.

HKKW19. Dennis Hofheinz, Akshay Kamath, Venkata Koppula, and Brent Waters. Adaptively secure constrained pseudorandom functions. In *Financial Cryptography and Data Security (FC)*, LNCS, pages 357–376. Springer, 2019.

HKW15. Susan Hohenberger, Venkata Koppula, and Brent Waters. Adaptively secure puncturable pseudorandom functions in the standard model. In *Advances in Cryptology—Asiacrypt 2015, Part I*, volume 9452 of *LNCS*, pages 79–102. Springer, 2015.

KMP16. Eike Kiltz, Daniel Masny, and Jiaxin Pan. Optimal security proofs for signatures from identification schemes. In *Advances in Cryptology—Crypto 2016, Part II*, volume 9815 of *LNCS*, pages 33–61. Springer, 2016.

KPTZ13. Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In *ACM Conf. on Computer and Communications Security (CCS) 2013*, pages 669–684. ACM Press, 2013.

Lac18. Marie-Sarah Lacharité. Security of BLS and BGLS signatures in a multi-user setting. *Cryptogr. Commun.*, 10(1):41–58, 2018.

LBC16. LBC. The LBC Server. https://lbc.cryptoguru.org/, 2016.

LFA20. Adriano Di Luzio, Danilo Francati, and Giuseppe Ateniese. Arcula: A secure hierarchical deterministic wallet for multi-asset blockchains. In *CANS 20 International Conference on Cryptology and Network Security*, LNCS, pages 323–343. Springer, 2020.

Lin19. Yehuda Lindell. A Full CryptoCurrency Custody Solution Based on MPC and Threshold ECDSA. Real World Crypto 2019, 2019. https://rwc.iacr.org/2019/slides/Multiparty-ECDSA-RWC2019.pdf.

LMO+14. Jake Longo, Daniel P. Martin, Elisabeth Oswald, Daniel Page, Martijn Stam, and Michael Tunstall. Simulatable leakage: Analysis, pitfalls, and new constructions. In *Advances in Cryptology—Asiacrypt 2014, Part I*, volume 8873 of *LNCS*, pages 223–242. Springer, 2014.

Med18. Mediawiki. BIP32 Specification. https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki, 2018.

Men17. Bart Mennink. Insuperability of the standard versus ideal model gap for tweakable blockcipher security. In *Advances in Cryptology—Crypto 2017, Part II*, volume 10402 of *LNCS*, pages 708–732. Springer, 2017.

MPs19. Antonio Marcedone, Rafael Pass, and abhi shelat. Minimizing trust in hardware wallets with two factor signatures. In *Financial Cryptography and Data Security (FC)*, LNCS, pages 407–425. Springer, 2019.

NIS21. NIST. Multi-Party Threshold Cryptography | CSRC. https://csrc.nist.gov/projects/threshold-cryptography, 2021.

Pat09. Jacques Patarin. The "coefficients H" technique (invited talk). In *Annual International Workshop on Selected Areas in Cryptography (SAC) 2008*, volume 5381 of *LNCS*, pages 328–345. Springer, 2009.

Res10. Certicom Research. Sec 2: Recommended elliptic curve domain parameters. Standards for Efficient Cryptography, 2010. https://www.secg.org/sec2-v2.pdf.

SGRR19. Phillipp Schoppmann, Adrià Gascón, Leonie Reichert, and Mariana Raykova. Distributed vector-OLE: Improved constructions and implementation. In *ACM Conf. on Computer and Communications Security (CCS) 2019*, pages 1055–1072. ACM Press, 2019.

SPY13. François-Xavier Standaert, Olivier Pereira, and Yu Yu. Leakage-resilient symmetric cryptography under empirically verifiable assumptions. In *Advances in Cryptology—Crypto 2013, Part I*, volume 8042 of *LNCS*, pages 335–352. Springer, 2013.

ST16. Thomas Shrimpton and R. Seth Terashima. Salvaging weak security bounds for blockcipher-based constructions. In *Advances in Cryptology—Asiacrypt 2016, Part I*, LNCS, pages 429–454. Springer, 2016.

TVR16. Mathieu Turuani, Thomas Voegtlin, and Michaël Rusinowitch. Automated verification of electrum wallet. In *FC 2016 Workshops*, volume 9604 of *LNCS*, pages 27–42, February 2016.

WMK16. Xiao Wang, Alex J. Malozemoff, and Jonathan Katz. EMP-toolkit: Efficient MultiParty computation toolkit. https://github.com/emp-toolkit, 2016.

WYG+17. Frank Wang, Catherine Yun, Shafi Goldwasser, Vinod Vaikuntanathan, and Matei Zaharia. Splinter: Practical Private Queries on Public Data. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pages 299–313. USENIX Association, 2017.

WYKW21. Chenkai Weng, Kang Yang, Jonathan Katz, and Xiao Wang. Wolverine: Fast, scalable, and communication-efficient zero-knowledge proofs for boolean and arithmetic circuits. In *IEEE Symp. Security and Privacy 2021*. IEEE, 2021.

YWL+20. Kang Yang, Chenkai Weng, Xiao Lan, Jiang Zhang, and Xiao Wang. Ferret: Fast extension for correlated OT with small communication. In *ACM Conf. on Computer and Communications Security (CCS) 2020*, pages 1607–1626. ACM Press, 2020.

A The LBC Bitcoin Cracking Project

Briefly speaking, given a pair of signature keys (sk_1, pk_1), a Bitcoin user could sign with sk_1 to spend funds associated with the public "address" $addr_1 = \mathsf{RIPEMD160}\big(\mathsf{SHA256}(pk_1)\big)$, i.e., a 160-bit hash digest of the public key pk_1.[13] But this means the map from signature keys to "addresses" is not bijective, and an adversary holding (sk', pk') with $\mathsf{RIPEMD160}\big(\mathsf{SHA256}(pk')\big) = addr_1$ can also spend funding associated with the address $addr_1$. Since $|addr_1| = 160$, the success probability to guess such a key pair (sk', pk') with T computations is nearly $T/2^{160}$ (this is much higher than $T/2^{256}$ of straightforwardly guessing the "original" secret key sk_1). While $T/2^{160}$ remains small, with u targeted Bitcoin users $(sk_1, pk_1), ..., (sk_u, pk_u)$, the probability to successfully guess (sk', pk') such that $\mathsf{RIPEMD160}\big(\mathsf{SHA256}(pk')\big) = addr_i = \mathsf{RIPEMD160}\big(\mathsf{SHA256}(pk_i)\big)$ for some i increases to $\frac{uT}{2^{160}}$. It can be seen this is a variant of multi-user secret (signing) key recovery attack, and the concrete security is of $160 - \log_2 u$ bits which degrades significantly with the number of targeted Bitcoin users. The LBC Bitcoin cracking project instantiates this idea in a distributed manner to gather computation power from multiple participants, and it recovered more than a dozen secret (signing) keys in 2016 and 2017 [LBC16].

It is tempting to ask how our treatments capture this attack. To clarity, note that our security definitions in Sect. 6.2 implicitly assume that *the cryptocurrency address of an account is its public signature key, rather than the hash digest of the key*. This ensures a bijective mapping between cryptocurrency addresses and signature key pairs (sk, pk), and the success probability to guess a key pair (sk, pk) corresponding to a certain address with T computations is $T/2^{256}$. In this case, the success probability of the above multi-user attack decreases to (much smaller) $uT/2^{256}$.

[13] This slightly deviates from the choice mentioned in Sect. 6.1, i.e., simply using pk_1 as the address.