Traitor Tracing without Trusted Authority from Registered Functional Encryption

Pedro Branco [1], Russell W. F. Lai [2], Monosij Maitra [1,3],
Giulio Malavolta [1,4], Ahmadreza Rahimi [1], and Ivy K. Y. Woo[2]

[1]Max-Planck Institute for Security and Privacy
pedrodemelobranco@gmail.com
ahmadrezar@pm.me
[2]Aalto University
russell.lai@aalto.fi
ivy.woo@aalto.fi
[3]Ruhr-Universität Bochum
monosij.maitra@rub.de
[4]Bocconi University
giulio.malavolta@hotmail.it

Abstract

Traitor-tracing systems allow identifying the users who contributed to building a rogue decoder in a broadcast environment. In a traditional traitor-tracing system, a key authority is responsible for generating the global public parameters and issuing secret keys to users. All security is lost if the *key authority itself* is corrupt. This raises the question: Can we construct a traitor-tracing scheme, without a trusted authority?

In this work, we propose a new model for traitor-tracing systems where, instead of having a key authority, users could generate and register their own public keys. The public parameters are computed by aggregating all user public keys. Crucially, the aggregation process is *public*, thus eliminating the need of any trusted authority. We present two new traitor-tracing systems in this model based on bilinear pairings. Our first scheme is proven adaptively secure in the generic group model. This scheme features a transparent setup, ciphertexts consisting of $6\sqrt{L}+4$ group elements, and a public tracing algorithm. Our second scheme supports a bounded collusion of traitors and is proven selectively secure in the standard model. Our main technical ingredients are new registered functional encryption (RFE) schemes for quadratic and linear functions which, prior to this work, were known only from indistinguishability obfuscation.

To substantiate the practicality of our approach, we evaluate the performance a proof of concept implementation. For a group of $L = 1024$ users, encryption and decryption take roughly 50ms and 4ms, respectively, whereas a ciphertext is of size 6.7KB.

1 Introduction

Traitor-tracing systems [CFN94] allow identifying the users who contributed to building a rogue decoder in a broadcast environment. In a traditional traitor-tracing system, a key authority is responsible for generating the global public parameters and issuing user secret keys. Given the public parameters, it is possible to encrypt a message so that any user in possession of a secret key can decrypt it. As in standard broadcast encryption, the encrypted message is hidden from any unauthorized user, i.e. those who do not have access to any secret key. The most important property of a traitor-tracing system, however, is the presence of a *tracing algorithm* which identifies corrupt users. More specifically, if an attacker produces a device that can decrypt ciphertexts with some non-negligible probability, then the tracing algorithm, given black-box access to the device, is guaranteed to identify at least one corrupt user, i.e. a member of those who contributed to the creation of the decryption device.

Traditional traitor-tracing systems [BSW06, BW06, BZ14, NWZ16, GKW18, GKW19, KW20, Zha20, GLW23, AKYY23] focus on the settings where an arbitrary set of users can be corrupt, assuming that the key authority is honest. Notably, all guarantees are lost if *the key authority itself* is corrupt. This limits the use cases of traditional traitor-tracing systems in the sense that the key authority must either be the same party as the encryptor or trusted by the latter. Even in the latter case, this limitation is clearly undesirable from a security perspective, as it introduces a single point of failure. In fact, we speculate that this limitation has played a significant role in preventing the adoption of traitor-tracing systems in practice, as it is identical in spirit to the key-escrow problem of identity-based encryption [Rog15].

In light of the above limitation of traditional traitor-tracing systems, a natural question is whether we can remove the trust assumption on the key authority.

Can we construct *efficient* traitor-tracing without a trusted authority?

Specifically, we are interested in constructions that achieve non-trivial efficiency (i.e. ciphertext size sublinear in the number of users L) and are based on simple and well-understood cryptographic structure, such as bilinear groups.

A New Model For Traitor Tracing. We envision a new model for traitor tracing without any trusted authority, that we refer to as *registered traitor-tracing*. In our model, each user samples its own pair of public and secret keys locally, without needing any interaction with any other users. Upon collecting all the public keys $(\mathsf{pk}_1, \ldots, \mathsf{pk}_L)$, for instance in a public directory or bulletin-board, it is possible to aggregate them into a *short* master public key mpk. Given mpk, anyone can encrypt a message m in such a way that only the registered users are able to recover it. Crucially, the aggregation of the public keys is a completely transparent and deterministic process, and therefore no trusted party is needed to perform this operation.[1] This model is directly inspired by recent works on registration-based encryption [GHMR18, GHM+19, GKMR22, FKdP23, DKL+23, HLWW23, ZZGQ23, FFM+23, DP23] and distributed broadcast [WQZDF10, BZ14, FWW23, KMW23] that adopt a similar paradigm to solve the key escrow problem in related settings.

The distinguishing property of traitor tracing system is (*public*) traceability: If a malicious user i^* builds a decryption box D, it should be possible (for *anyone*) to track i^* given only black-box access to D. To substantiate the usefulness of this primitive, we discuss how it can be useful in some recurrent scenarios below.

[1] Anyone can re-evaluate the aggregation process and check if the output is identical to what is claimed.

Application: Traceable Group Messaging. In a group messaging system, a group of L users wants to broadcast messages to each other privately. Given that messages are constantly exchanged, it is important that the size of the ciphertext should be sublinear in L, especially for large groups. As the simplest notion of security, we want that an external observer learns no information about the messages exchanged within the group. Furthermore, we want to protect against users leaking their secret key, for instance by having their device compromised: In order to do this, one needs to be able to trace the users corresponding to the leaked key, in order to exclude them from the group.

Superficially, it may appear that group messaging is the killer application for traitor-tracing systems. However, at present, no existing system uses traitor-tracing techniques to build their protocols. We speculate that this is due to the presence of a trusted authority: The cost of adding the tracing property to the system is to insert a trusted authority that can potentially decrypt all messages of all groups! On the other hand, in *registered traitor-tracing* no such tradeoff is present, and we can add traceability to groups without introducing any backdoor. We envision that register-traitor tracing can be used as a cryptographic building block in messaging schemes to add a traceability guarantee, which is not present in current systems. At the time of writing, the maximum size of a Whatsapp group is $L = 1024$, which is well within the range of practicality of our scheme.

1.1 Our Contributions

We construct traitor-tracing systems without a trusted authority, where users can sample their own keys locally without interaction. Formally, we introduce a new primitive called *registered traitor-tracing* (RTT) supporting an unbounded collusion of traitors. We then present two constructions in the bounded and unbounded collusion settings.

Our main technical ingredients are new constructions of *registered functional encryption* (RFE) for quadratic and linear functions from bilinear groups. Prior works either built general-purpose RFE based on indistinguishability obfuscation (iO) [FFM+23, DP23], or specialised RFE for *inner-product predicates* from bilinear groups [FFM+23]. In more detail, our contributions are summarised as follows:

(1) A New Model for Traitor-Tracing. We introduce the notion of registered traitor-tracing (RTT) as a new model to build traitor-tracing systems without a trusted authority. We propose appropriate security definitions for RTT and show compilers (inspired by the literature in non-registered setting) that allows us to reduce the problem of constructing RTT to building a weak form of RFE for quadratic functions and an RFE for linear functions. We also discuss efficient strategies to revoke traitors.

(2) Unbounded-Collusion RTT. We propose a new RFE for *quadratic* functions (RQFE), in a weaker setting where all functions to be registered are known during setup. This weaker form of RQFE is nevertheless sufficient for RTT supporting an *unbounded* collusion of traitors. The resulting RTT scheme has a *transparent setup*, ciphertext size $6\sqrt{L}+4$ in number of group elements, and a *public tracing* algorithm. The scheme is based on prime-order groups and is *adaptively* secure in the generic (bilinear) group model (GGM).

(3) Bounded-Collusion RTT. We present an RFE for *linear* functions (RLFE), in the ordinary setting where functions to be registered can be adaptively chosen after setup. This scheme is sufficient for RTT supporting a *bounded* collusion, where the maximum number of traitors is fixed at setup. We prove that our RLFE is secure, against an arbitrary subset of *selectively* corrupted users, in the standard model. The security of this scheme rests upon a static q-type assumption, which we show to hold in the GGM. As a bonus, we further show how our RLFE enables other new applications, such as registered threshold encryption (RTE) for t-out-of-L thresholds, where ciphertexts are of size $O(t)$ in number of group elements. RTE generalises the notion of distributed broadcast [WQZDF10, BZ14, FWW23, KMW23] to t-out-of-L thresholds.

(4) Prototype Implementation. We provide an open-source prototype implementation of our RTT scheme with unbounded collusion. For a group of $L = 1024$ users (which is currently the maximum size of a Whatsapp group chat), our benchmarks demonstrate that our scheme is quite practical: The key generation, encryption, and decryption algorithms take 553ms, 51ms, and 4ms, respectively, whereas a ciphertext is of size 6.7KB.

1.2 Related Work

Traitor Tracing. Traitor tracing was first introduced in [CFN94], and ever since it has become one of the most studied topics in cryptography, with a large body of literature improving on the original proposal. Works on traitor tracing, e.g. [BSW06, BW06, BZ14, NWZ16, GKW18, GKW19, KW20, Zha20, GLW23, AKYY23], focus on constructing schemes with sublinear[2] efficiency, in terms of size of public parameters and/or ciphertexts. This is possible by leveraging computational assumptions in bilinear groups or lattices.

To the best of our knowledge, most prior traitor tracing schemes require a trusted setup to generate the public parameters and secret keys. An exception is a very recent work of Luo [Luo22] that constructs a *Broadcast, Trace and Revoke* (or simply *Trace and Revoke*) system in the setting without a trusted authority, where each party samples its own keys. This work achieves essentially asymptotically optimal parameters, but relies on indistinguishability obfuscation [BGI+01, JLS21], and thus it is not concretely efficient. Trace and Revoke schemes integrate the notion of revocation in traitor-tracing, where the functionality allows one to revoke decryption rights for any subset of users whose secret keys have been compromised. This notion has been extensively studied [NNL01, NP01, DF03, KHL03, BW06] in the setting with a centrally trusted authority.

Furthermore, a series of work has explored the related notion of *distributed broadcast encryption* [WQZDF10, BZ14, FWW23, KMW23], which does not concern traceability.

Registered Cryptography. Registered cryptography is a paradigm introduced by Garg et al. [GHMR18, GHM+19] to remove the key-escrow from advanced forms of encryption that require a trusted setup. The paradigm has recently gained attention and a series of works have improved its functionality [GV20] and efficiency [CES21], ultimately leading to practical constructions from pairings [GKMR22, FKdP23] and lattices [DKL+23]. The notion of registration-based encryption (RBE) was recently extended to the settings of attribute-based [HLWW23, ZZGQ23] and functional encryption [FFM+23, DP23].

[2] There exists a "trivial" traitor-tracing scheme where each party samples a public-key encryption individually, the master public key consists of the concatenation of the L public keys, and the ciphertext is simply the concatenation of encryptions under each individual key.

Closest to our work is the scheme introduced by [FFM+23], which builds RFE for inner product *predicates*: Here keys are associated with vectors $\mathbf{y}$ and a ciphertext encrypts vector $\mathbf{x}$ and a message m, and security requires that the message m (but nothing else) is revealed if and only if $\mathbf{x}^T\mathbf{y} = 0$.[3] We emphasise that this is different from the setting of inner product *functions* that we consider in this work (also known as inner product FE), where (for the case of linear functions) the ciphertext reveals the inner product $\mathbf{x}^T\mathbf{y}$ itself, regardless on whether it is 0 or not. The two functionalities are incomparable, in the sense that there is no obvious reduction in either direction. In the non-registered settings, this difference is analogous to the distinction between [KSW08] and [ABDP15].

1.3 Discussion

Interactive vs Non-Interactive Solution. An alternative (generic) solution to remove trusted authorities in traitor-tracing systems, or in general any cryptographic systems, is to let participants *simulate* the trusted authority with a secure multi-party computation (MPC) protocol: All users run an MPC where they jointly sample the master secret key, and the output of each user consist of its tracing key, as well as the master public parameters. While this solution effectively bypasses the need for a trusted authority, it is undesirable for several reasons: (i) All users must be simultaneously online to run the MPC. Even a single user failing would stall the entire process. As the number of users in the system grows, this solution scales poorly. (ii) It is harder to support dynamic joins of new users, since for every new user that joins a new MPC protocol must be jointly run by all participants. (iii) MPC protocols require interaction, which adds latency to the key registration process. (iv) Running the key generation of a traitor-tracing scheme as an MPC is computationally intense, making this solution computationally very expensive.

In this work we focus on the *non-interactive* settings, where users sample their own keys locally, and simply upload it to a public bulletin board once they are done. Different users do not even have to be aware of each other's existence, and it is much easier to support a dynamic set of participants (more discussion on this later). For these reasons, we believe that the non-interactive settings is both theoretically more elegant and preferable from a practical standpoint.

Common Reference String vs Trusted Authority. We acknowledge that, in line with the literature on registered/distributed cryptography, our schemes are in the common reference string model, where all parties are assumed to receive a common reference string that was sampled in a trusted manner. However, we also highlight the fact that our RQFE scheme has a *transparent setup*, meaning that the common reference string is just a collection of random bits. In practice, this is desirable since there are very simple protocols to sample such strings (e.g. hash some fixed bitstring). On the other hand, our RLFE has a structured setup. We claim that, even for the case of a non-transparent setup, this model is substantially better than having a trusted authority, since there is no long-term secret that needs to be stored. It also resolves questions about the availability of the trusted authority, and how parties can establish secure communication channels for receiving their keys.

Static Joins vs Dynamic Joins. Throughout this work, we will always assume that the set of users that register their keys is fixed ahead of time, and the public parameters are aggregated only

[3] Although we note that a revised work [DPY23], developed concurrently with our work, presents an RLFE scheme and proves it secure in the GGM.

after all users have registered their keys. That is, we assume that the set of users participating in the protocol is *static*, and if a new user joins the system, the master public key needs to be recomputed and all users have to be notified of this change, and (possibly) must update their information. [HLWW23] refers to this as the *slotted* setting.

In practice, it is desirable to allow users to join the system dynamically, and one does not want to re-initialise the public parameters and/or to notify all existing users. Fortunately, it is possible to generically move from the slotted/static settings to support dynamic joins, while minimising the number of updates. Informally, the transformation works by partitioning the users in sets of exponentially increasing cardinality, e.g. $\{S_i : |S_i| = 2^i\}_{i\in[\log(L)]}$, and filling those sets as users join, starting from the smaller ones. Updates then only need to be issued when a set is filled up and needs to be transferred to the next empty set. It is easy to see that each user receives at most $\log(L)$ updates throughout its lifetime. Variants of this transformation have been described many times in the literature [GHMR18, GHM+19, GKMR22, HLWW23, FFM+23, KMW23] and we refer the reader to these works for more details. In what follows, we will only describe schemes in the slotted/static settings, with the understanding that dynamic joins can be supported with this transformation.

2 Technical Highlights

We highlight the technical innovations of our work. We begin by showing how constructing RTT boils down to building the right notion of RQFE, then we present our RFE schemes. We conclude by outlining registered threshold encryption as another new application of RFEs.

2.1 Registered Traitor-Tracing

To set some context, let us make more concrete the desiderata for an RTT scheme. In an RTT scheme, the setup outputs a (preferably *unstructured*) common reference string crs. Each party i starts by generating its own pair of public and secret keys $(\mathsf{pk}_i, \mathsf{sk}_i)$ relative to crs. Upon collecting all the public keys $(\mathsf{pk}_1, \ldots, \mathsf{pk}_L)$, anyone can use the crs to compute a short master public key mpk and the helper decryption key hsk_i for each user i. Given mpk, anyone can encrypt m in such a way that only a registered user i is able to obtain the message, using its secret key sk_i and helper key hsk_i. Additionally, RTT should fulfill a strong traceability property: If a malicious user i^* builds a decryption box D (which receives ciphertexts and outputs the corresponding message with non-negligible probability), then the user i^* can be caught given only black-box access to D. The RTT scheme is said to be bounded-collusion secure if the setup additionally inputs the maximum number of traitors, else it supports an unbounded collusion.

TT via Functional Encryption. To better understand the challenge of constructing (R)TT, it is useful to recall how to construct traditional traitor-tracing schemes (with a trusted authority). The work of Boneh, Sahai, and Waters [BSW06] reduces this problem to a simpler cryptographic primitive called private linear broadcast encryption (PLBE) and shows how to generically turn a PLBE scheme into a traitor-tracing scheme. In a nutshell, a PLBE is a broadcast encryption scheme with an additional trace-encrypt algorithm. This algorithm takes as input an index $i \in [L]$ and a message, and generates an ordinary-looking ciphertext of the message which can only be decrypted by user $\ell \geq i$. Importantly, this ciphertext must keep the index i hidden (except to users

i and $i+1$, who can trivially test the position of the index). The trace-encrypt algorithm can be used in a linear scan to identify the user with the smallest index who contributed to creating a rogue decryption device.

Abstracting even further, it turns out that PLBE is nothing but a special case of functional encryption (FE) [Gay16], where keys are associated with an index ℓ and a predicate $F_\ell(i, m)$ such that

$$F_\ell(i, m) := \begin{cases} m & \text{if } i \leq \ell \\ 0 & \text{otherwise} \end{cases}$$

whereas ciphertexts contain information about the message m and the index i. This connection is made explicit in [Gay16] where they show that an FE for quadratic functions (QFE) is sufficient to implement the above comparison predicate, and consequently PLBE, with ciphertext size $O(\sqrt{L})$. Thus, the problem of traitor tracing is nothing but QFE in disguise.[4]

For the (weaker) bounded-collusion setting, Agrawal et al. [ABP+17] show how to reduce the problem of traitor-tracing (with revocation) to that of bounded-collusion FE for linear functions (LFE).

Registered FE: Removing the Authority. Via the aforementioned series of transformations, we have reduced the task of constructing traitor-tracing without authority to that of constructing QFE/LFE without authority. This notion was recently introduced under the name of *registered functional encryption* (RFE) [FFM+23, DP23] as a natural generalisation of registration-based encryption [GHMR18]. In short, RFE provides a mechanism to publicly aggregate L independent key-function tuples $(\mathsf{pk}_1, f_1), \ldots, (\mathsf{pk}_L, f_L)$ into a digest, so that a ciphertext of m generated with respect to the digest can be decrypted by sk_j to recover $f_j(m)$.

For the remainder of this overview, we will focus on describing our RFE schemes (for quadratic and linear functions), along with other applications. Extending the transformation from QFE/LFE to traitor-tracing in the registered settings require some care, but the main ideas are analogous to the traditional settings. Therefore we omit them here, and refer the reader to Section 5 for more details.

2.2 RQFE in the GGM

Our first observation that facilitates our task is that one does not need the full power of (R)QFE to build traitor tracing. Since the functions $f_1, \ldots, f_L$ depend only on the identity of each user, it suffices to build a scheme where all functions associated with secret keys are known ahead of time. In other words, we can assume that each user knows all the other functions during key generation. With this observation in mind, we describe our RQFE below.

Conceptually, we build our RQFE by compiling a traditional QFE into a registered one, provided that it satisfies a *master secret key homomorphism*. In other words, we want the master public key of the scheme to be some encoding of the master secret key, that satisfies the following homomorphic relation:

$$\underbrace{\mathsf{Encode}(\mathsf{msk}_0)}_{\mathsf{mpk}_0} * \underbrace{\mathsf{Encode}(\mathsf{msk}_1)}_{\mathsf{mpk}_1} = \mathsf{Encode}(\mathsf{msk}_0 + \mathsf{msk}_1)$$

[4] Note that linearising a quadratic polynomial achieves the desired functionality, but nullifies the efficiency of the transformation. In particular, the resulting PLBE scheme would have ciphertexts linear in L, which does not improve over trivial constructions.

and furthermore for all functions f:

$$\underbrace{\mathsf{KGen}(\mathsf{msk}_0, f)}_{\mathsf{sk}_f^{(0)}} * \underbrace{\mathsf{KGen}(\mathsf{msk}_1, f)}_{\mathsf{sk}_f^{(1)}} = \mathsf{KGen}(\mathsf{msk}_0 + \mathsf{msk}_1, f).$$

The exact specifications of the encoding function Encode and the group operation $*$ are irrelevant for this explanation. To define the master public key of the scheme, each user samples a local key pair $(\mathsf{mpk}_i, \mathsf{msk}_i)$ and we define the *global* master public key as

$$\widetilde{\mathsf{mpk}} = \mathsf{mpk}_1 * \ldots * \mathsf{mpk}_L = \mathsf{Encode}(\mathsf{msk}_1 + \ldots + \mathsf{msk}_L)$$

which can be computed publicly using the master public keys published by each user. In effect, the L users are sharing (in the sense of additive secret-sharing) the master secret key of the new combined key $\widetilde{\mathsf{mpk}}$. The users will then also publish enough information to help the i-th user computing a functional secret key under the new master public key. Here is where we leverage the fact that all functions are known in advance, and we ask each user to publish, along with their mpk_j all functional keys, except for their own function. In other words, the j-th user also outputs

$$\left\{\mathsf{sk}_{f_i}^{(j)} = \mathsf{KGen}(\mathsf{msk}_j, f_i)\right\}_{i \neq j}.$$

Arranging all of these public information in matrix form, and applying the homomorphic operator row-wise, we obtain:

$$\begin{pmatrix} \perp & \mathsf{sk}_{f_2}^{(1)} & \ldots & \mathsf{sk}_{f_L}^{(1)} \\ \mathsf{sk}_{f_1}^{(2)} & \perp & \ldots & \mathsf{sk}_{f_L}^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ \mathsf{sk}_{f_1}^{(L)} & \mathsf{sk}_{f_2}^{(L)} & \ldots & \perp \end{pmatrix} \xrightarrow{*} \begin{pmatrix} \mathsf{KGen}(\sum_{j \neq 1} \mathsf{msk}_j, f_1) \\ \mathsf{KGen}(\sum_{j \neq 2} \mathsf{msk}_j, f_2) \\ \vdots \\ \mathsf{KGen}(\sum_{j \neq L} \mathsf{msk}_j, f_L) \end{pmatrix}$$

Note that the i-th combined key is almost a valid functional secret key for f_i under $\widetilde{\mathsf{mpk}}$, except that it is missing the contribution of msk_i. However, the i-th user is the one that sampled msk_i in the first place, and therefore it can easily fill the missing value to obtain

$$\begin{aligned} \widetilde{\mathsf{sk}}_{f_i} &= \mathsf{KGen}\left(\sum_{j \neq i} \mathsf{msk}_j, f_i\right) * \mathsf{KGen}(\mathsf{msk}_i, f_i) \\ &= \mathsf{KGen}\left(\sum_{j} \mathsf{msk}_j, f_i\right). \end{aligned}$$

At this point, decryption and encryption correctness simply follow by the correctness of the original FE scheme, except that we now have substituted the key authority with a fully distributed setup.

Instantiating the Transformation. Given this general template outlined above, all that is left is to look into the literature of traditional QFE schemes, and find a compatible one. It turns out that a handful of schemes satisfy this homomorphic property. However, while all schemes obtained via this transform are correct, not all of them can be proven secure. For instance, the RQFE scheme obtained by transforming the QFE of Wee [Wee20] is unfortunately broken due to linear attacks. The only QFE scheme which we are aware of that survives the transformation is that of Baltico et al. [BCFG17], which was only proven to be secure in the GGM. Consequently, our RQFE inherits the security in the GGM. Proving security of this template turns out to be a nuanced task, since we have to deal with potentially malformed keys and adaptive corruption queries. We refer to Section 4.2 for more technical details.

2.3 RLFE in the Standard Model

Our RFE construction for linear functions is conceptually similar to the recent works of [HLWW23, ZZGQ23] on registered attribute-based encryption (RABE)[5] and can be summarised by the following idea. Starting with a base FE scheme, the major challenge is to construct a one-user RFE which is correct and secure. Given this, we can construct an L-user RFE by running L parallel instances of the one-user RFE, where the digest mpk aggregates all individual mpk_i from the L instances. To ensure decryption correctness, the Aggr algorithm outputs helper keys which correspond to the cross-terms due to $\left(\mathsf{mpk}_j\right)_{j \neq i}$ in mpk for each user i.

Our RLFE. While the above general strategy can be applied to various existing linear FEs, adapting their security proofs to the registered setting is tricky. We settle at basing on the scheme of [ABDP15] due to its simplicity, which we recall:

$$\mathsf{mpk} = [\mathbf{w}^{\mathsf{T}}], \quad \mathsf{msk} = \mathbf{w}^{\mathsf{T}}, \quad \mathsf{sk}_{\mathbf{y}} = \mathbf{w}^{\mathsf{T}}\mathbf{y}, \quad \mathsf{ct}_{\mathbf{x}} = \left([s], [s\mathbf{w}^{\mathsf{T}} + \mathbf{x}^{\mathsf{T}}]\right)$$

for some $\mathbf{w}, \mathbf{x}, \mathbf{y} \in \mathbb{Z}_p^n$ and $s \in \mathbb{Z}_p$. To decrypt, compute

$$[s\mathbf{w}^{\mathsf{T}} + \mathbf{x}^{\mathsf{T}}]\mathbf{y} - [s]\mathbf{w}^{\mathsf{T}}\mathbf{y} = [\mathbf{x}^{\mathsf{T}}\mathbf{y}].$$

We turn this into a one-user RLFE for a given function $\mathbf{y}$ with the following steps. Fix $[\mathbf{w}^{\mathsf{T}}]$ in the crs and let $\mathsf{mpk}' = [\mathbf{w}^{\mathsf{T}}\mathbf{y}]$. Correspondingly, let $\mathsf{ct}_{\mathbf{x}} = \left([s\mathbf{w}^{\mathsf{T}}\mathbf{y}], [s\mathbf{w}^{\mathsf{T}} + \mathbf{x}^{\mathsf{T}}]\right)$, so that the same decryption equation (and both correctness and security of the base scheme) applies:

$$[s\mathbf{w}^{\mathsf{T}} + \mathbf{x}^{\mathsf{T}}]\mathbf{y} - [s\mathbf{w}^{\mathsf{T}}\mathbf{y}] = [\mathbf{x}^{\mathsf{T}}\mathbf{y}].$$

Crucially, mpk' "Pedersen-commits" the function $\mathbf{y}$ using the "key" $\mathbf{w}^{\mathsf{T}}$, which is then inherited in ct, so that no one can decrypt to $\mathbf{x}^{\mathsf{T}}\mathbf{y}'$ for $\mathbf{y}' \neq \mathbf{y}$. This yields a "public RLFE" that anyone can decrypt via the above equation, and to make this available only to the registered user, the idea is to (additively) secret-share the commitment key. We let $\mathsf{pk} = [\mathbf{v}], \mathsf{sk} = \mathbf{v}$, and the new commitment key be $\mathbf{w} + \mathbf{v}$, shared by crs and the user. The resulting one-user RLFE has

$$\begin{aligned} &\mathsf{crs} = [\mathbf{w}^{\mathsf{T}}], \quad \mathsf{mpk} = [(\mathbf{w}^{\mathsf{T}} + \mathbf{v}^{\mathsf{T}})\mathbf{y}], \quad \mathsf{pk} = [\mathbf{v}^{\mathsf{T}}], \\ &\mathsf{sk} = \mathbf{v}^{\mathsf{T}}, \quad \mathsf{ct}_{\mathbf{x}} = \left([s], [s(\mathbf{w}^{\mathsf{T}} + \mathbf{v}^{\mathsf{T}})\mathbf{y}], [s\mathbf{w}^{\mathsf{T}} + \mathbf{x}^{\mathsf{T}}]\right) \end{aligned}$$

and decryption follows from

$$[s\mathbf{w}^{\mathsf{T}} + \mathbf{x}^{\mathsf{T}}]\mathbf{y} + [s]\mathbf{v}^{\mathsf{T}}\mathbf{y} - [s(\mathbf{w}^{\mathsf{T}} + \mathbf{v}^{\mathsf{T}})\mathbf{y}] = [\mathbf{x}^{\mathsf{T}}\mathbf{y}].$$

[5] Actually our RLFE setting is slightly simpler than their RABE setting, since composite-order group (or an additional layer of transformation to prime-order group) is not necessary for security.

In a nutshell, security follows from two facts: Only the user who knows the share $\mathbf{v}$ can access the "public RLFE"; furthermore decrypting to only $\mathbf{x}^{\mathsf{T}}\mathbf{y}$ is safeguarded by the other share $\mathbf{w}$. From here, we apply the L-parallel-instances compiler to obtain an L-user RLFE. To prevent mix-and-match of helper keys, i.e. cross-terms across the L instances, a randomisation factor for each user is introduced and bound to their helper keys, which is done via pairing. The scheme of [ABDP15] is proven from DDH with selective-security. Our scheme inherits the same security and the randomisation in helper keys lead to our q-type assumption (for $q = L$ number of users), which is essentially a q-type variant of DDH generalised into the pairing setting. We give our full RLFE construction in Section 4.3.

2.4 Registered Threshold Encryption

As a bonus application, we discuss how RLFE helps in removing the trusted setup in threshold encryption. In other words, we show how to build *registered threshold encryption* (RTE). Recall that, in traditional threshold encryption, the public parameters of the system are generated together with L users' secret keys. Given an encryption ct of a message m, each user can compute partial decryption shares using its secret key. Once we have t partial decryption shares, where the recovery threshold $t \le L$ is specified in the public parameters, the message m can be recovered. In terms of security, we want that an adversary holding less than t secret keys is unable to break semantic security of the scheme. In RTE, parties generate their own public keys and these are later aggregated into a short master public key. The system should preserve the "threshold decryption" functionality as in traditional threshold encryption.

To compile an RLFE into an RTE, each party i simply runs the RLFE key generation on a vector $\mathbf{i} = (1, i, \ldots, i^{t-1}) \in \mathbb{Z}_p^t$. To encrypt a message $m \in \{0, 1\}$, the encryptor first performs Shamir secret sharing, i.e. sampling a random degree-$(t-1)$ polynomial P over $\mathbb{Z}_p$ such that $P(0) = m$. Let $\mathbf{p} \in \mathbb{Z}_p^t$ be the coefficient vector of P. The encryptor encrypts $\mathbf{p}$ using the underlying RLFE scheme. By the security of the RLFE, a party holding a secret key sk_i learns $\langle \mathbf{i}, \mathbf{p} \rangle = P(i)$ and nothing else about the polynomial P. Once we have t different evaluations of the polynomial, we can recover $P(0) = m$ by Lagrange interpolation. In Section 6.2 we detail our RTE construction.

One subtle issue that we omitted so far is that the RLFE decryption actually allows a party i to recover the inner product $\langle \mathbf{i}, \mathbf{p} \rangle = P(i)$ *in the exponent* of a target group $\mathbb{G}_{\mathsf{T}}$ element $[P(i)]_{\mathsf{T}}$ from the underlying bilinear pairing. This does not create an issue as Lagrange interpolation is a linear function and thus, we can perform it in the exponent to recover $[P(0)]_{\mathsf{T}}$. Since $P(0) = m \in \{0, 1\}$, we can brute-force m from $[m]_{\mathsf{T}}$.

3 Preliminaries

Notation. We denote the security parameter by $\lambda \in \mathbb{N}$ throughout this paper and assume it as an implicit input to all algorithms. We write $[n] = \{1, \ldots, n\}$ and $[0, n] = \{0\} \cup [n]$ for any $n \in \mathbb{N}$. Capital and small bold-face letters (like $\mathbf{M}$ and $\mathbf{x}$) denote matrices and (column) vectors respectively. Capital and small letters (such as S and x) in general denote sets and concrete algebraic variables respectively (with any exceptions being stated explicitly). A tuple $T = (t_i)_{i \in [n]}$ defines an ordered set with elements indexed from $[n]$ for any $n \in \mathbb{N}$. Accordingly, $|S|$ and $|\mathbf{x}|$ respectively denotes the cardinality of set S and the length of a vector $\mathbf{x}$. We write $x \leftarrow\!\!\$\; X$ to denote sampling an element x from X uniformly at random. We write $\mathcal{A}$ for a probabilistic polynomial time (PPT) adversary that runs in time polynomial in λ. A function in λ, denoted by $\mathsf{negl}(\lambda) : \mathbb{N} \mapsto \mathbb{R}$, is called negligible if it vanishes faster than the inverse of any polynomial in λ, i.e. $\mathsf{negl}(\lambda) \in \mathcal{O}(1/p(\lambda))$ for all positive polynomials $p(\lambda)$.

Prime-Order Bilinear Groups. Throughout this work, we use cyclic groups of prime order p with an asymmetric bilinear map endowed on them. We assume a PPT bilinear group generator algorithm GGen that takes $\lambda \in \mathbb{N}$ as input and outputs $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_\mathsf{T}, p, g_1, g_2, e)$, where p is a prime of $\Theta(\lambda)$ bits, $\mathbb{G}_1 = \langle g_1 \rangle, \mathbb{G}_2 = \langle g_2 \rangle, \mathbb{G}_\mathsf{T} = \langle g_\mathsf{T} \rangle = \langle e(g_1, g_2) \rangle$ are cyclic groups of order p with $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_\mathsf{T}$ being a non-degenerate bilinear map. We use the implicit (bracket) notation for group elements: for $\mathbf{M}, \mathbf{M}' \in \mathbb{Z}_p^{k_1 \times k_2}$, define $[\mathbf{M}]_\mathsf{t} = g_\mathsf{t}^{\mathbf{M}} := \left(g_\mathsf{t}^{m_{i,j}}\right)$ and $[\mathbf{M}]_\mathsf{t} + [\mathbf{M}']_\mathsf{t} := [\mathbf{M} + \mathbf{M}' \bmod p]_\mathsf{t}$ for $\mathsf{t} \in \{1, 2, \mathsf{T}\}$ and $k_1, k_2 \in \mathbb{N}$. We also denote $[1]_1 := g_1$, $[1]_2 := g_2$, and abbreviate "e" with "$\cdot$", i.e. for matrices $\mathbf{M}_1, \mathbf{M}_2$ of appropriate dimensions, $e\left([\mathbf{M}_1]_1, [\mathbf{M}_2]_2\right)$ is written as $[\mathbf{M}_1]_1 [\mathbf{M}_2]_2 = [\mathbf{M}_1\mathbf{M}_2]_\mathsf{T} = g_\mathsf{T}^{\mathbf{M}_1\mathbf{M}_2}$. We express sampling a bilinear group instance as $\mathcal{G} := (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_\mathsf{T}, p, [1]_1, [1]_2, \cdot) \leftarrow \mathsf{GGen}(1^\lambda)$.

4 Registered Functional Encryption

We define and construct the core building blocks for our applications, namely registered functional encryption (RFE) for quadratic and linear functions. In particular, we define the syntax and security of RFE in Section 4.1. Then, Sections 4.2 and 4.3 provide schemes for weak RFE and RFE for quadratic and linear functions respectively. Both our RFE schemes are proven secure in presence of well-formed keys. Appendix A describes generic and concrete ways of tackling malicious keys to transcend this limitation.

4.1 Definitions

We define RFE and a variant which we call weak RFE. The main difference between the two is that, in the weak variant, the set of functions to be registered is known already at setup time. Below we primarily define RFE and describe the difference of the weak variant inline.

Definition 4.1 (Registered Functional Encryption)**.** A registered functional encryption (RFE) scheme for message space $\mathcal{M}$, ciphertext space $\mathcal{C}$, function class $\mathcal{F}$ and number of users L consists of the following tuple of PPT algorithms (Setup, KGen, Aggr, Enc, Dec):

- $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$: On input the security parameter 1^λ, the setup algorithm outputs a common reference string crs.
- $(\mathsf{pk}_\ell, \mathsf{sk}_\ell) \leftarrow \mathsf{KGen}(\mathsf{crs}, \ell \in [L])$: The key generation algorithm outputs a pair of public and secret keys $(\mathsf{pk}_\ell, \mathsf{sk}_\ell)$ for user ℓ.
- $(\mathsf{mpk}, (\mathsf{hsk}_\ell)_{\ell \in [L]}) \leftarrow \mathsf{Aggr}(\mathsf{crs}, (\mathsf{pk}_\ell, f_\ell)_{\ell \in [L]})$: On input crs and the tuple of public key pk_ℓ and function $f_\ell \in \mathcal{F}$ of all users $\ell \in [L]$, the deterministic aggregation algorithm outputs a master public key mpk and a tuple of helper secret keys $(\mathsf{hsk}_\ell)_{\ell \in [L]}$.
- $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mu)$: On input mpk and a message $\mu \in \mathcal{M}$, the encryption algorithm outputs a ciphertext $\mathsf{ct} \in \mathcal{C}$.

- $\mu' \leftarrow \mathsf{Dec}(\mathsf{sk}_\ell, \mathsf{hsk}_\ell, \mathsf{ct})$: On input a ciphertext ct together with a secret key sk_ℓ and a helper secret key hsk_ℓ, the decryption algorithm outputs μ'.

A weak RFE has the same syntax as an RFE, except that the tuple of functions $(f_\ell)_{\ell\in[L]}$ is input to Setup instead of to Aggr.

Definition 4.2 (Correctness)**.** An RFE scheme is said to be correct, if for all $\lambda \in \mathbb{N}$, $L \in \mathsf{poly}(\lambda)$, $\mu \in \mathcal{M}, k \in [L], (f_\ell)_{\ell\in[L]} \in \mathcal{F}^L, \mathsf{crs} \in \mathsf{Setup}(1^\lambda)$, $(\mathsf{pk}_k, \mathsf{sk}_k) \in \mathsf{KGen}(\mathsf{crs}, k)$, it holds that

$$\Pr\left[\mu' = f_k(\mu) \;\middle|\; \begin{array}{l} (\mathsf{mpk}, (\mathsf{hsk}_\ell)_{\ell\in[L]}) \leftarrow \mathsf{Aggr}(\mathsf{crs}, (\mathsf{pk}_\ell, f_\ell)_{\ell\in[L]}) \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mu) \\ \mu' \leftarrow \mathsf{Dec}(\mathsf{sk}_k, \mathsf{hsk}_k, \mathsf{ct}) \end{array}\right] = 1.$$

Correctness of a weak RFE is defined analogously with the only differences being that $(f_\ell)_{\ell\in[L]}$ is input to Setup instead of to Aggr.

Definition 4.3 (Strong Compactness)**.** An RFE is said to be strongly compact, if for all $\lambda \in \mathbb{N}$, $L \in \mathsf{poly}(\lambda)$, $(f_\ell)_{\ell\in[L]} \in \mathcal{F}^L$, $\mathsf{crs} \in \mathsf{Setup}(1^\lambda)$, $(\mathsf{pk}_\ell, \mathsf{sk}_\ell) \in \mathsf{KGen}(\mathsf{crs}, \ell)$, and $(\mathsf{mpk}, (\mathsf{hsk}_\ell)_{\ell\in[L]}) \in \mathsf{Aggr}(\mathsf{crs}, (\mathsf{pk}_\ell, f_\ell)_{\ell\in[L]})$, it holds that $|\mathsf{mpk}|$, $|\mathsf{hsk}_\ell|$, $|\mathsf{ct}|$ are of size $\mathsf{poly}(\lambda, \log L)$.[6] Strong compact weak RFEs are defined analogously.

Definition 4.4 (Security)**.** An RFE scheme Π is said to be secure, if for any PPT $\mathcal{A}$ it holds that

$$\left|\Pr\left[\mathsf{Exp}^0_{\Pi,\mathcal{A}}(1^\lambda) = 1\right] - \Pr\left[\mathsf{Exp}^1_{\Pi,\mathcal{A}}(1^\lambda) = 1\right]\right| \leq \mathsf{negl}(\lambda),$$

where $\mathsf{Exp}^b_{\Pi,\mathcal{A}}$ is defined in Fig. 1. The security of a weak RFE is defined analogously, with the only difference that $(f_\ell)_{\ell\in[L]}$ is declared by $\mathcal{A}$ upfront and input to Setup instead of to Aggr.

We also consider the notion of *selective-security with static corruption*, where the experiment is same as that in Fig. 1, except that $\mathcal{A}$ declares the messages (μ_0, μ_1) and the set of corrupt users $C \subseteq [L]$ at the beginning of the experiment (i.e. the corruption oracle CorrO is withheld from $\mathcal{A}$).

Remark 4.5 (On Malicious Keys and Key Queries)**.** In Definition 4.4 we require $\mathcal{A}$ to output the randomness $(\mathbf{r}_\ell)_{\ell\in M}$ for keys generated by $\mathcal{A}$, the setting which our RFEs will be proven secure. We defer handling malicious keys without this requirement to Appendix A, where the relevant $\mathsf{IsValid}$ algorithm and completeness property are also introduced. For simplicity we only allow a single key query per user ℓ. The mildly stronger notion of allowing multiple key queries per user is implied so long as KGen is stateless (so that the same reduction still applies when simulating multiple keys for the same user) and holds true for both of our RFE schemes.

4.2 Weak RFE for Quadratic Functions

We build a weak RFE scheme for quadratic functions with a *transparent* setup, i.e. the crs is constructed with public randomness.

[6] Our definition is stronger than existing RFE compactness [FFM+23], since it additionally requires *succinct ciphertexts*.

$\mathsf{Exp}^{b}_{\Pi,\mathcal{A}}(1^\lambda)$	$\mathsf{KGenO}(\ell)$
$\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$ $\big(\mu_0, \mu_1, (\mathsf{pk}_\ell, f_\ell)_{\ell\in[L]}, (\mathbf{r}_\ell)_{\ell\in M}\big) \leftarrow \mathcal{A}^{\mathsf{CorrO}(\cdot),\mathsf{KGenO}(\cdot)}(\mathsf{crs})$ // $\mathcal{A}$ provides randomness for set M of maliciously generated keys **assert** $[L] \setminus M \subseteq K$ **assert** $\mathsf{pk}_\ell \in \mathsf{KGen}(\mathsf{crs}, \ell; \mathbf{r}_\ell) \ \ \forall \ell \in M$ **assert** $f_\ell(\mu_0) = f_\ell(\mu_1) \ \ \forall \ell \in C \cup M$ $(\mathsf{mpk}, (\mathsf{hsk}_\ell)_{\ell\in[L]}) \leftarrow \mathsf{Aggr}(\mathsf{crs}, (\mathsf{pk}_\ell, f_\ell)_{\ell\in[L]})$ $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mu_b)$ $b' \leftarrow \mathcal{A}(\mathsf{ct}^*)$ **return** b'	**if** $K[\ell] = \bot$ $(\mathsf{pk}_\ell, \mathsf{sk}_\ell) \leftarrow \mathsf{KGen}(\mathsf{crs}, \ell)$ $K[\ell] := (\mathsf{pk}_\ell, \mathsf{sk}_\ell)$ $(\mathsf{pk}_\ell, \mathsf{sk}_\ell) \leftarrow K[\ell]$ **return** pk_ℓ $\mathsf{CorrO}(\ell)$ $C := C \cup \{\ell\}$ $(\mathsf{pk}_\ell, \mathsf{sk}_\ell) \leftarrow K[\ell]$ **return** sk_ℓ

Figure 1: Security experiment for RFE.

Let $n_1, n_2, L \in \mathsf{poly}(\lambda)$. For any $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_\mathsf{T}, p, [1]_1, [1]_2, \cdot)$ output by $\mathsf{GGen}(1^\lambda)$, we construct an RFE for the message space $\mathcal{M} = \mathbb{Z}_p^{n_1} \times \mathbb{Z}_p^{n_2}$, the class of quadratic functions $\mathcal{F}$ being

$$\left\{\left(f : \mathcal{M} \to [\mathbb{Z}_p]_\mathsf{T}, f(\mathbf{x}, \mathbf{y}) \mapsto \left[\mathbf{x}^\mathsf{T}\mathbf{F}\mathbf{y} \bmod p\right]_\mathsf{T}\right) : \mathbf{F} \in \mathbb{Z}_p^{n_1 \times n_2}\right\},$$

and (an upper bound of) L number of users. Since for any $f \in \mathcal{F}$, $f(\mathbf{x}, \mathbf{y}) \mapsto \left[\mathbf{x}^\mathsf{T}\mathbf{F}\mathbf{y} \bmod p\right]_\mathsf{T}$ is fully described by $\mathbf{F}, \mathbb{G}_\mathsf{T}$ and p whereas $\mathbb{G}_\mathsf{T}, p$ are publicly fixed, we simply write $\mathbf{F}$ for such. Further, for any $\ell \in \mathbb{N}$ and $\mathbf{F}_\ell \in \mathcal{F}$ we denote its (i, j)-th entry as $f_{i,j}^{(\ell)} \in \mathbb{Z}_p$.

Theorem 4.6. RQFE (Fig. 2) is strongly compact (Definition 4.3).

Proof. Assuming that the groups description $\mathcal{G}$ and each element in $\mathbb{Z}_p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_\mathsf{T}$ are of description size $\mathsf{poly}(\lambda)$, we count the size of mpk, hsk_ℓ, and ct: $|\mathsf{mpk}|, |\mathsf{ct}| = (n_1 + n_2) \cdot \mathsf{poly}(\lambda)$, and $|\mathsf{hsk}_\ell| = n_1 n_2 \cdot \mathsf{poly}(\lambda)$. Notably, they are of size independent of L. □

Theorem 4.7. RQFE (Fig. 2) is correct (Definition 4.2).

Proof. Recall $[\mathbf{s}]_1 = \left[\sum_{\ell\in[L]} \mathbf{s}_\ell\right]_1, [w]_1 = \left[\sum_{\ell\in[L]} w_\ell\right]_1$. For a user $k \in [L]$, its secret key is $\mathsf{sk}_k = \left[\mathbf{s}_k^\mathsf{T}\mathbf{F}_k\mathbf{t} + \gamma_k w_k\right]_2$ and its helper key $\mathsf{hsk}_k = \left(\left[\sum_{\ell\in[L]\setminus\{k\}} \mathbf{s}_\ell^\mathsf{T}\mathbf{F}_k\mathbf{t} + \gamma_k w_\ell\right]_2, [\gamma_k]_2, \mathbf{F}_k\right)$. User $k \in [L]$ decrypts to

$$\begin{aligned}
[D_0]_\mathsf{T} = \ & [\alpha]_1 \left(\left[\mathbf{s}_k^\mathsf{T}\mathbf{F}_k\mathbf{t} + \gamma_k w_k\right]_2 + \left[\sum_{\ell\in[L]\setminus\{k\}} \mathbf{s}_\ell^\mathsf{T}\mathbf{F}_k\mathbf{t} + \gamma_k w_\ell\right]_2\right) \\
= \ & [\alpha]_1 \left[\sum_{\ell\in[L]} \mathbf{s}_\ell^\mathsf{T}\mathbf{F}_k\mathbf{t} + \gamma_k w_\ell\right]_2 = \left[\alpha\mathbf{s}^\mathsf{T}\mathbf{F}_k\mathbf{t} + \alpha\gamma_k w\right]_\mathsf{T}, \\
[D_1]_\mathsf{T} = \ & [\alpha w]_1 [\gamma_k]_2 = [\alpha\gamma_k w]_\mathsf{T}
\end{aligned}$$

$\mathsf{Setup}(1^\lambda, (\mathbf{F}_\ell)_{\ell\in[L]})$

$\mathcal{G} := (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_\mathsf{T}, p, [1]_1, [1]_2, \cdot) \leftarrow \mathsf{GGen}(1^\lambda)$
$\textbf{for } \ell \in [L] : \gamma_\ell \leftarrow\!\!\$\ \mathbb{Z}_p$
$\mathbf{t} \leftarrow\!\!\$\ \mathbb{Z}_p^{n_2}$
$\textbf{return } \mathsf{crs} := \left(\mathcal{G}, (\mathbf{F}_\ell)_{\ell\in[L]}, ([\gamma_\ell]_2)_{\ell\in[L]}, [\mathbf{t}]_2\right)$

$\mathsf{Aggr}(\mathsf{crs}, (\mathsf{pk}_\ell)_{\ell\in[L]})$

$[\mathbf{s}]_1 := \left[\sum_{\ell\in[L]} \mathbf{s}_\ell\right]_1; \ [w]_1 := \left[\sum_{\ell\in[L]} w_\ell\right]_1$
$\textbf{for } k \in [L] :$
$\quad [h_{1,k}]_2 := \left[\sum_{\ell\in[L]\setminus\{k\}} \mathsf{dk}_{\ell,k}\right]_2$
$\quad [h_{2,k}]_2 := [\gamma_k]_2$
$\mathsf{mpk} := (\mathcal{G}, [\mathbf{s}]_1, [w]_1, [\mathbf{t}]_2)$
$\mathsf{hsk}_k := ([h_{1,k}]_2, [h_{2,k}]_2, \mathbf{F}_k)$
$\textbf{return } (\mathsf{mpk}, (\mathsf{sk}_k)_{k\in[L]})$

$\mathsf{KGen}(\mathsf{crs}, \ell)$

$\mathbf{s}_\ell \leftarrow\!\!\$\ \mathbb{Z}_p^{n_1}; \ w_\ell \leftarrow\!\!\$\ \mathbb{Z}_p$
$\textbf{for } k \in [L] :$
$\quad [\mathsf{dk}_{\ell,k}]_2 := \left[\mathbf{s}_\ell^\mathsf{T}\mathbf{F}_k\mathbf{t} + \gamma_k w_\ell\right]_2$
$\mathsf{pk}_\ell := \left([\mathbf{s}_\ell]_1, [w_\ell]_1, ([\mathsf{dk}_{\ell,k}]_2)_{k\in[L]\setminus\{\ell\}}\right)$
$\mathsf{sk}_\ell := [\mathsf{dk}_{\ell,\ell}]_2$
$\textbf{return } (\mathsf{pk}_\ell, \mathsf{sk}_\ell)$

$\mathsf{Dec}(\mathsf{sk}_k = [K]_2, \mathsf{hsk}_k, \mathsf{ct})$

$[D_0]_\mathsf{T} := [C_1]_1 ([K]_2 + [h_1]_2)$
$[D_1]_\mathsf{T} := [C_2]_1 [h_2]_2$
$[D_2]_\mathsf{T} := \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} f_{i,j}^{(k)} \cdot \left([\mathbf{C}_{3,i}^\mathsf{T}]_1 [\mathbf{C}_{4,j}]_2\right)$
$\textbf{return } [D_0]_\mathsf{T} - [D_1]_\mathsf{T} + [D_2]_\mathsf{T}$

$\mathsf{Enc}(\mathsf{mpk}, (\mathbf{x}, \mathbf{y}))$

$\textbf{parse } [\mathbf{s}]_1 = ([s_1]_1, \ldots, [s_{n_1}]_1)$
$\textbf{parse } [\mathbf{t}]_2 = ([t_1]_2, \ldots, [t_{n_2}]_2)$
$\textbf{parse } (\mathbf{x}, \mathbf{y}) = (x_1, \ldots, x_{n_1}, y_1, \ldots, y_{n_2})$
$\alpha \leftarrow\!\!\$\ \mathbb{Z}_p$
$\mathbf{M} \leftarrow\!\!\$\ \mathsf{GL}_2(\mathbb{Z}_p)$
$[C_1]_1 := [\alpha]_1$
$[C_2]_1 := [\alpha w]_1$
$[\mathbf{C}_{3,i}]_1 := \left[(\mathbf{M}^{-1})^\mathsf{T} \cdot \begin{pmatrix} x_i \\ \alpha s_i \end{pmatrix}\right]_1, \forall i \in [n_1]$
$[\mathbf{C}_{4,j}]_2 := \left[\mathbf{M} \cdot \begin{pmatrix} y_j \\ -t_j \end{pmatrix}\right]_2, \forall j \in [n_2]$
$\mathsf{ct} := \left([C_1]_1, [C_2]_1, ([\mathbf{C}_{3,i}]_1)_{i\in[n_1]}, ([\mathbf{C}_{4,j}]_2)_{j\in[n_2]}\right)$
$\textbf{return } \mathsf{ct}$

Figure 2: Weak RQFE construction.

$$\begin{aligned}
[D_2]_\mathsf{T} &= \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} f_{i,j}^{(k)} \left(\left[(x_i, \alpha s_i)\mathbf{M}^{-1}\right]_1 \left[\mathbf{M} \cdot \begin{pmatrix} y_j \\ -t_j \end{pmatrix}\right]_2 \right) \\
&= \left[\sum_{i=1}^{n_1}\sum_{j=1}^{n_2} x_i y_j f_{i,j}^{(k)} - \alpha s_i t_j f_{i,j}^{(k)}\right]_\mathsf{T} = \left[\mathbf{x}^\mathsf{T}\mathbf{F}_k\mathbf{y} - \alpha\mathbf{s}^\mathsf{T}\mathbf{F}_k\mathbf{t}\right]_\mathsf{T},
\end{aligned}$$

hence yielding the desired output

$$[D_0]_\mathsf{T} - [D_1]_\mathsf{T} + [D_2]_\mathsf{T} = \left[\alpha\mathbf{s}^\mathsf{T}\mathbf{F}_k\mathbf{t} + \alpha\gamma_k w\right]_\mathsf{T} - [\alpha\gamma_k w]_\mathsf{T} + \left[\mathbf{x}^\mathsf{T}\mathbf{F}_k\mathbf{y} - \alpha\mathbf{s}^\mathsf{T}\mathbf{F}_k\mathbf{t}\right]_\mathsf{T} = \left[\mathbf{x}^\mathsf{T}\mathbf{F}_k\mathbf{y}\right]_\mathsf{T}.$$

□

Theorem 4.8. RQFE (Fig. 2) is secure (Definition 4.4) in GGM.

Proof. We start with some notations and definitions for generic and symbolic bilinear group models.

Generic Bilinear Group Model. Our definitions for generic bilinear group model is adapted from [BCFG17]. Let $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_\mathsf{T}, p, [1]_1, [1]_2, \cdot)$ be a bilinear group setting, $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_\mathsf{T}$ be lists of group elements in $\mathbb{G}_1, \mathbb{G}_2$, and $\mathbb{G}_\mathsf{T}$ respectively. Let $\mathcal{D}$ be a distribution over $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_\mathsf{T}$. The generic group model for a bilinear group setting $\mathcal{G}$ and a distribution $\mathcal{D}$ is described in Fig. 3. In this model, the challenger first initialises the lists $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_\mathsf{T}$ by sampling the group elements according to $\mathcal{D}$, and the adversary receives handles for the elements in the lists. For $\mathsf{t} \in \{1, 2, \mathsf{T}\}$, $\mathcal{L}_\mathsf{t}[h]$ denotes the h-th element in the list $\mathcal{L}_\mathsf{t}$. The handle to this element is simply the pair (t, h). An adversary $\mathcal{A}$ running in the generic bilinear group model can apply group operations and the bilinear map to the elements in the lists. To do this, $\mathcal{A}$ has to call the appropriate oracle specifying handles for the input elements. $\mathcal{A}$ also gets access to the internal state variables of the challenger via handles, and we assume that the equality queries are "free", in the sense that they do not count when measuring the computational complexity of $\mathcal{A}$. For $\mathsf{t} \in \{1, 2, \mathsf{T}\}$, the challenger computes the result of a query, say $\delta \in \mathbb{G}_\mathsf{t}$, and stores it in the corresponding list as $\mathcal{L}_\mathsf{t}[\mathsf{pos}] = \delta$ where pos is its next *empty* position in $\mathcal{L}_{\mathsf{t}}$, and returns to $\mathcal{A}$ its (newly created) handle $(\mathsf{t}, \mathsf{pos})$. Handles are not unique (i.e. the same group element may appear more than once in a list under different handles). The equality test oracle in [BCFG17] is replaced with the zero-test oracle $\mathsf{Zt}_{\mathsf{T}}(\cdot)$ that, on input a handle (t, h), returns 1 if $\mathcal{L}_{\mathsf{t}}[h] = 0$ and 0 otherwise only for the case $\mathsf{t} = \mathsf{T}$.

State: Lists $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_{\mathsf{T}}$ over $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_{\mathsf{T}}$ respectively.

Initializations: Lists $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_{\mathsf{T}}$ sampled according to distribution $\mathcal{D}$.

Oracles: The oracles provide black-box access to the group operations, the bilinear map, and zero-tests.

- $\forall \mathsf{t} \in \{1, 2, \mathsf{T}\}$: $\mathsf{Add}_{\mathsf{t}}(h_1, h_2)$ appends $\mathcal{L}_{\mathsf{t}}[h_1] + \mathcal{L}_{\mathsf{t}}[h_2]$ to $\mathcal{L}_{\mathsf{t}}$ and returns its handle $(\mathsf{t}, |\mathcal{L}_{\mathsf{t}}|)$.
- $\forall \mathsf{t} \in \{1, 2, \mathsf{T}\}$: $\mathsf{Neg}_{\mathsf{t}}(h)$ appends $-\mathcal{L}_{\mathsf{t}}[h]$ and returns its handle $(\mathsf{t}, |\mathcal{L}_{\mathsf{t}}|)$.
- $\mathsf{Map}(h_1, h_2)$ appends $[\mathcal{L}_1[h_1]]_1 \, [\mathcal{L}_2[h_2]]_2$ and returns its handle $(\mathsf{T}, |\mathcal{L}_{\mathsf{T}}|)$.
- $\mathsf{Zt}_{\mathsf{T}}(h)$ returns 1 if $\mathcal{L}_{\mathsf{T}}[h] = 0$ and 0 otherwise.

All oracles return $\perp$ when given invalid indices.

Figure 3: GGM for bilinear group setting $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_{\mathsf{T}}, p, [1]_1, [1]_2, \cdot)$ and distribution $\mathcal{D}$.

Symbolic Bilinear Group Model. The symbolic bilinear group model (SGM) for a bilinear group setting $\mathcal{G}$ and a distribution $\mathcal{D}$ gives to the adversary the same interface as the corresponding generic group model (GGM), except that internally the challenger stores lists of elements from the ring $\mathbb{Z}_q[\mathrm{x}_1, \ldots, \mathrm{x}_k]$ instead of lists of group elements, where $\{\mathrm{x}_k\}_{k \in \mathbb{N}}$ are indeterminates. The oracles $\mathsf{Add}_{\mathsf{t}}(\cdot,\cdot), \mathsf{Neg}_{\mathsf{t}}(\cdot), \mathsf{Map}(\cdot,\cdot), \mathsf{Zt}_{\mathsf{T}}(\cdot)$ compute addition, negation, multiplication, and zero tests respectively in the ring. For our proof, we will work in the ring $\mathbb{Z}_q[\mathrm{x}_1, \ldots, \mathrm{x}_k]$. Note that any element $\Phi \in \mathbb{Z}_q[\mathrm{x}_1, \ldots, \mathrm{x}_k]$ can be represented as

$$\Phi(\mathrm{x}_1, \ldots, \mathrm{x}_k) = \sum_{\in \mathbb{Z}^k} \eta_{\mathbf{c}} \prod_{i=1}^{k} \mathrm{x}_i^{c_i} \qquad \text{with } \mathbf{c} = (c_1, \ldots, c_k) \in \mathbb{Z}^k$$

using $\{\eta_{\mathbf{c}} \in \mathbb{Z}_q\}_{\mathbf{c} \in \mathbb{Z}^k}$, where $\eta_{\mathbf{c}} = 0$ for all but finite $\mathbf{c} \in \mathbb{Z}^k$. Note that this expression is unique. We now begin our proof for Theorem 4.8 below.

At a high level, the proof proceeds in a sequence of hybrids, where the first one (resp. the last one) encrypts $(\mathbf{x}^{(0)}, \mathbf{y}^{(0)}) \in \mathbb{Z}_p^{n_1+n_2}$ (resp., $(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}) \in \mathbb{Z}_p^{n_1+n_2}$). We will show that these hybrids are statistically indistinguishable from each other.

W.l.o.g., the challenger $\mathcal{C}$ simulates all the generic bilinear group oracle queries for $\mathcal{A}$. In particular, $\mathcal{C}$ stores actual computed elements in the list $\mathcal{L}_{\mathsf{t}}$ based on its group type $\mathsf{t} \in \{1, 2, \mathsf{T}\}$. Note that there is no element from $\mathbb{G}_{\mathsf{T}}$ in our scheme. So w.l.o.g., we will explicitly specify only the elements that $\mathcal{C}$ stores in $\mathcal{L}_1$ or $\mathcal{L}_2$. However, the only way $\mathcal{A}$ can learn information in the GGM are via calls to $\mathsf{Zt}_{\mathsf{T}}(\cdot)$ by providing handles (T, h) to elements in $\mathcal{L}_{\mathsf{T}}$. Therefore, we will specify such elements from $\mathbb{G}_{\mathsf{T}}$ explicitly as and when needed in the proof. The handle to an actual element stored in any of these lists are just a tuple $(\mathsf{t}, \mathsf{pos})$ specifying the group type t and its position in the table $\mathcal{L}_{\mathsf{t}}$. Since our scheme contains several variables, we will refrain from explicitly specifying the handles to the actual elements for convenience. Further, when we move to the SGM, we will denote any literal variable v as $\mathtt{v}$ and composite terms like $v_1 v_2$ (resp., $\frac{v_1}{v_2}$) as $\mathtt{v_1 v_2}$ (resp., $\frac{\mathtt{v_1}}{\mathtt{v_2}}$) to represent an individual monomial in a (possibly multivariate) polynomial. For variables denoted with Greek alphabets, say α, β, γ, we represent their corresponding formal variables as $\upalpha, \upbeta, \upgamma$. Assume $\mathcal{A}$ issues an arbitrary polynomial number $Q_{\mathsf{zt}}(\lambda)$ of $\mathsf{Zt}_{\mathsf{T}}(\cdot)$ queries in each of these three hybrids.

Hybrid $\mathcal{H}_0$: This is the game corresponding to bit $b = 0$ in the GGM which goes as follows.

- **Setup phase:** $\mathcal{A}$ declares the set of functions $(\mathbf{F}_\ell)_{\ell \in [L]}$ to be registered. The challenger samples $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_{\mathsf{T}}, p, [1]_1, [1]_2, \cdot) \leftarrow \mathsf{GGen}(1^\lambda)$ and initialises three tables $\mathcal{L}_{\mathsf{t}}[1]$ for all $\mathsf{t} \in \{1, 2, \mathsf{T}\}$ with the respective group generators $[1]_1, [1]_2$ and $[1]_{\mathsf{T}}$. It then prepares a tuple $\mathcal{G}' = \big(p, \{(\mathsf{t}, 1)\}_{\mathsf{t} \in \{1,2\}}\big)$, where $(\mathsf{t}, 1)$ represents the handle to these respective generators. Simulating the generic group oracle, it prepares to returns the crs as follows:
  1. For all $\ell \in [L]$, it computes $\gamma_\ell \in \mathbb{Z}_p$ and also $\mathbf{t} \in \mathbb{Z}_p^{n_2}$ as in the real Setup algorithm. Update $\mathcal{L}_2$ with the elements $[\gamma_\ell]_2$ for all $\ell \in [L]$ and each entry from $[\mathbf{t}]_2$. Set $\mathsf{crs} = (\mathcal{G}, (\mathbf{F}_\ell)_{\ell \in [L]}, \{[\gamma_\ell]_2\}_{\ell \in [L]}, [\mathbf{t}]_2)$.
  2. Return to $\mathcal{A}$ a tuple crs' that includes $\mathcal{G}'$ along with the handles to all elements in the same order as they are arranged in the crs above.
- **Query phase:** $\mathcal{A}$ issues its key queries and corruption queries.
  - For a key query on ℓ, if $K[\ell] = \perp$, the challenger does the following:[7]
    1. Sample $\mathbf{s}_\ell \leftarrow \mathbb{Z}_p^{n_1}, w_\ell \in \mathbb{Z}_p$.
    2. Compute sk_ℓ and $\mathsf{pk}_\ell = \big([\mathbf{s}_\ell]_1, [w_\ell]_1, \{[\mathsf{dk}_{\ell,k}]_2\}_{k \in [L] \setminus \{\ell\}}\big)$ as in the real KGen.
    3. Update $\mathcal{L}_1$ with $([\mathbf{s}_\ell]_1, [w_\ell]_1)$. Recall for all $k \in [L] \setminus \{\ell\}$, $[\mathsf{dk}_{\ell,k}]_2$ has the following structure:
    $$[\mathsf{dk}_{\ell,k}]_2 = \mathbf{s}_\ell^{\mathsf{T}} \mathbf{F}_k [\mathbf{t}]_2 + w_\ell [\gamma_k]_2 .$$
    Even given handles to $\mathbf{s}_\ell$ and w_ℓ in $\mathbb{G}_1$ (from pk_ℓ) (along with the handles to $\mathbf{t}$ and γ_k in $\mathbb{G}_2$ from crs'), it is easy to see that $\mathcal{A}$ cannot compute a handle for $\mathsf{dk}_{\ell,k}$ in $\mathbb{G}_2$ on its own. Thus the challenger adds $[\mathsf{dk}_{\ell,k}]_2$ to $\mathcal{L}_2$ for each $k \in [L] \setminus \{\ell\}$.
    4. Set $K[\ell] = (\mathsf{pk}'_\ell, \mathsf{pk}_\ell, \mathsf{sk}_\ell)$, where pk'_ℓ is a sequence of handles to all elements in the same order as they are arranged in pk_ℓ.

    Then it parses $(\mathsf{pk}'_\ell, \mathsf{pk}_\ell, \mathsf{sk}_\ell)$ from $K[\ell]$ and return pk'_ℓ to $\mathcal{A}$.
  - For a corrupt query on ℓ, the challenger returns sk_ℓ from $K[\ell] = (\mathsf{pk}'_\ell, \mathsf{pk}_\ell, \mathsf{sk}_\ell)$ and update the set of corrupt indices $C \coloneqq C \cup \{\ell\}$. Recall sk_ℓ has the following structure:
  $$\mathsf{sk}_\ell = \mathbf{s}_\ell^{\mathsf{T}} \mathbf{F}_\ell [\mathbf{t}]_2 + w_\ell [\gamma_\ell]_2 .$$
- **Challenge phase:** $\mathcal{A}$ specifies the following challenge information:
$$(\mathsf{pk}_\ell, \mathsf{sk}_\ell)_{\ell \in [L]}, \quad (\mathbf{s}_\ell, w_\ell)_{\ell \in M} \quad \text{and} \quad ((\mathbf{x}^{(0)}, \mathbf{y}^{(0)}), (\mathbf{x}^{(1)}, \mathbf{y}^{(1)})) \in (\mathbb{Z}_p^{n_1+n_2})^2.$$
  <u>*Check admissibility.*</u> Denote $H \coloneqq [L] \setminus M$ the set of indices whose keys are honestly generated by the challenger. For each $\ell \in [L]$, the challenger checks that:

[7] As mentioned earlier in Definition 4.4, we only allow $\mathcal{A}$ to one key-query per slot. Going ahead however, for a more complete treatment, our analysis below in Table 1 and some of the later hybrids allows $\mathcal{A}$ to query multiple keys per slot. We show this explictly with a counter $c \in [Q_{\mathsf{k}}]$, where w.l.o.g., we assumed Q_{k} as the maximum number of key queries per slot.

- For all $\ell \in [L]$, pk_ℓ is either honestly generated by the challenger, or maliciously generated by $\mathcal{A}$ but the key generation randomness is provided, i.e. it checks $H \subseteq K$.
- For all $\ell \in M$ whose key is maliciously generated, run $(\mathsf{pk}'_\ell, \mathsf{sk}'_\ell) \leftarrow \mathsf{KGen}(\mathsf{crs}, \ell; \mathbf{s}_\ell, w_\ell)$ using the provided randomness $\mathbf{r}_\ell$ and check that $\mathsf{pk}_\ell = \mathsf{pk}'_\ell$.
- For all $\ell \in C \cup M$, it holds that $(\mathbf{x}^{(0)})^\intercal \mathbf{F}_\ell \mathbf{y}^{(0)} = (\mathbf{x}^{(1)})^\intercal \mathbf{F}_\ell \mathbf{y}^{(1)}$.

It aborts if any of the above is false.

Key aggregation. The challenger runs

$$\left(\mathsf{mpk}, (\mathsf{hsk}_k)_{k\in[L]}\right) \leftarrow \mathsf{Aggr}\left(\mathsf{crs}, \{\mathsf{pk}^*_1, \ldots, \mathsf{pk}^*_L\}\right), \text{ where}$$

$$\mathsf{mpk} = (\mathcal{G}, [\mathbf{s}]_1, [w]_1, [\mathbf{t}]_2), \text{ and } \mathsf{hsk}_k = \left(\left[\textstyle\sum_{\ell\in[L]\setminus\{k\}} \mathsf{dk}_{\ell,k}\right]_2, [\gamma_k]_2, \mathbf{F}_k\right) \text{ for all } k \in [L].$$

Since Aggr is deterministic, $\mathcal{A}$ is able to compute $\left(\mathsf{mpk}, (\mathsf{hsk}_\ell)_{\ell\in[L]}\right)$ on its own. In the GGM, $\mathcal{A}$ computes handles for the elements in mpk and $(\mathsf{hsk}_\ell)_{\ell\in[L]}$. To this end, it queries the appropriate group oracles *iteratively* as per the Aggr algorithm to generate the tuples mpk' and each $\mathsf{hsk}_{\ell'}$ as sequences of handles to all elements (except ℓ and for the ones it already had from before) in the same order as arranged in mpk and each hsk_ℓ for all $\ell \in [L]$.

Compute challenge ciphertext. The challenger does the following:

1. Generate $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{mpk}, (\mathbf{x}^{(0)}, \mathbf{y}^{(0)}))$ where $\mathsf{ct}^* = \left([C_1]_1, [C_2]_1, \{[\mathbf{C}_{3,i}]_1\}_{i\in[n_1]}, \{[\mathbf{C}_{4,j}]_2\}_{i\in[n_2]}\right)$.
2. Recall $[C_1]_1 = [\alpha]_1, [C_2]_1 = [\alpha w]_1 \in \mathbb{G}_1$. Accordingly, update $\mathcal{L}_1$ with $[\alpha]_1$ and $[\alpha w]_1$.
3. Parse the message as $\left(\mathbf{x}^{(0)}, \mathbf{y}^{(0)}\right) = \left(x_1^{(0)}, \ldots, x_{n_1}^{(0)}, y_1^{(0)}, \ldots, y_{n_2}^{(0)}\right)$ and
$$[\mathbf{s}]_1 = ([s_1]_1, \ldots, [s_{n_1}]_1) \quad , \quad [\mathbf{t}]_2 = ([t_1]_2, \ldots, [t_{n_2}]_2).$$
4. Recall that the elements $[\mathbf{C}_{3,i}]_1 \in \mathbb{G}_1$ and $[C_{3,j}]_1 \in \mathbb{G}_2$ have the following structure:
$$\begin{aligned}
\forall i \in [n_1], \quad [\mathbf{C}_{3,i}]_1 &= \left[(\mathbf{M}^{-1})^\intercal \cdot \begin{pmatrix} x_i^{(0)} \\ \alpha s_i \end{pmatrix}\right]_1 = \left[\frac{1}{\Delta_\mathbf{M}} \begin{pmatrix} d & -c \\ -b & a \end{pmatrix} \cdot \begin{pmatrix} x_i^{(0)} \\ \alpha s_i \end{pmatrix}\right]_1 \\
\forall j \in [n_2], \quad [\mathbf{C}_{4,j}]_2 &= \left[\mathbf{M} \cdot \begin{pmatrix} y_j^{(0)} \\ -t_j \end{pmatrix}\right]_2 = \left[\begin{pmatrix} a & b \\ c & d \end{pmatrix} \cdot \begin{pmatrix} y_j^{(0)} \\ -t_j \end{pmatrix}\right]_2
\end{aligned}$$
where $\Delta_\mathbf{M} = (ad - bc)$ denotes the determinant of $\mathbf{M} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \leftarrow \mathsf{GL}_2(\mathbb{Z}_p)$.
Accordingly, update $\mathcal{L}_1$ with $\left\{\left[\Delta_\mathbf{M}^{-1}\left(dx_i^{(0)} - c\alpha s_i\right)\right]_1, \left[\Delta_\mathbf{M}^{-1}\left(-bx_i^{(0)} + a\alpha s_i\right)\right]_1\right\}_{i\in[n_1]}$ and $\mathcal{L}_2$ with $\left\{\left[ay_j^{(0)} - bt_j\right]_2, \left[cy_j^{(0)} - dt_j\right]_2\right\}_{j\in[n_2]}$ in order. The challenger outputs $\mathsf{ct}^{*\prime}$ to $\mathcal{A}$ that includes the handles to elements in ct^* arranged in the same order as described above.

- **Output phase:** $\mathcal{A}$ outputs a bit $b' \in \{0, 1\}$.

Hybrid $\mathcal{H}_1$: In this hybrid, fix $\mathbf{M}$ at the outset of the experiment and replaces the generator $[1]_1 \in \mathbb{G}_1$ with $[\Delta_{\mathbf{M}}]_1$. Accordingly, this changes all the elements in the scheme that are generated in $\mathbb{G}_1$. In particular, the elements that mainly change in the actual scheme are as follows:

1. The handle to $[1]_1$ in crs' points to $[\Delta_{\mathbf{M}}]_1$.
2. For all $\ell \in [L]$, $([\mathbf{s}_\ell]_1, [w_\ell]_1) \in \mathsf{pk}_\ell$ changes to $([\Delta_{\mathbf{M}}\mathbf{s}_\ell]_1, [\Delta_{\mathbf{M}} w_\ell]_1)$.
3. The modified $\mathsf{mpk} = (\mathcal{G}, [\Delta_{\mathbf{M}}\mathbf{s}]_1, [\Delta_{\mathbf{M}} w]_1, [\mathbf{t}]_2)$, where $\mathbf{s} = \sum_{\ell=1}^{L} \mathbf{s}_\ell = (s_1, \ldots, s_{n_1}), w = \sum_{\ell=1}^{L} w_\ell$.
4. Finally, the modified ciphertext elements are:

$$[C_1]_1 = [\Delta_{\mathbf{M}}\alpha]_1 \quad , \quad [C_2]_1 = [\Delta_{\mathbf{M}}\alpha w]_1 \quad , \quad [\mathbf{C}_{3,i}]_1 = \left[\begin{pmatrix} d & -c \\ -b & a \end{pmatrix} \cdot \begin{pmatrix} x_i^{(0)} \\ \alpha s_i \end{pmatrix}\right]_1$$

The rest of the experiment remains the same as $\mathcal{H}_0$. Note that $a, b, c, d \leftarrow \mathbb{Z}_p$ were sampled randomly. Thus, the above change amounts to a shift in $\mathcal{H}_1$'s output distribution only by a statistical distance of *at most* $\frac{3}{p}$ (obtained from $\Pr[\Delta_{\mathbf{M}} = 0]$), which is negligible. Hence, $\mathcal{H}_0 \approx_s \mathcal{H}_1$.

For ease of presentation, in Table 1 we show all unit and composite terms generated in the scheme itself, and stored in the respective lists.

Hybrid $\mathcal{H}_2$: In this hybrid, the challenger moves *partially* to the SGM. Namely, the interaction with $\mathcal{A}$ remains the same as in $\mathcal{H}_1$, except that now the challenger stores formal variables instead of the actual elements in the respective lists $\mathcal{L}_{\mathtt{t}}$ for all $\mathtt{t} \in \{1, 2, \mathsf{T}\}$. Thus, all the handles that $\mathcal{A}$ receives refer to multivariate polynomials from the following ring:

$$\zeta = \mathbb{Z}_p\left[\{\gamma_\ell\}_{\ell\in[L]}, (\mathtt{t}_1, \ldots, \mathtt{t}_{n_2}), \{(\mathtt{s}_{\ell,1}^c, \ldots, \mathtt{s}_{\ell,n_1}^c)\}_{\ell\in H, c\in[Q_{\mathsf{k}}]}, \{\mathtt{w}_\ell^c\}_{\ell\in H, c\in[Q_{\mathsf{k}}]}, \alpha, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\right].$$

Concretely, $\mathcal{A}$ gets handles to formal polynomials (from the scheme) from $\mathcal{L}_{\mathtt{t}}$ for each $\mathtt{t} \in \{1, 2\}$, where:

1. $\mathcal{L}_1 = \mathcal{L}_1^{\mathsf{crs}} \cup \mathcal{L}_1^{\mathsf{key}} \cup \mathcal{L}_1^{\mathsf{ct}}$, where
   (a) $\mathcal{L}_1^{\mathsf{crs}} = \{(\mathtt{ad} - \mathtt{bc})\}$,
   (b) $\mathcal{L}_1^{\mathsf{key}} = \left\{\left((\mathtt{ad} - \mathtt{bc})\,\mathtt{s}_{\ell,1}^c, \ldots, (\mathtt{ad} - \mathtt{bc})\,\mathtt{s}_{\ell,n_1}^c\right), (\mathtt{ad} - \mathtt{bc})\,\mathtt{w}_\ell^c\right\}_{c\in[Q_{\mathsf{k}}],\ell\in H}$, and
   (c) $\mathcal{L}_1^{\mathsf{ct}} = \left\{(\mathtt{ad} - \mathtt{bc})\,\alpha, (\mathtt{ad} - \mathtt{bc})\,(\alpha\mathtt{w}_1 + \cdots + \alpha\mathtt{w}_L), \left\{\left(\mathtt{d}x_i^{(0)} - \mathtt{c}\alpha\mathtt{s}_i\right), \left(-\mathtt{b}x_i^{(0)} + \mathtt{a}\alpha\mathtt{s}_i\right)\right\}_{i\in[n_1]}\right\}$.
2. $\mathcal{L}_2 = \mathcal{L}_2^{\mathsf{crs}} \cup \mathcal{L}_2^{\mathsf{key}} \cup \mathcal{L}_2^{\mathsf{ct}}$, where
   (a) $\mathcal{L}_2^{\mathsf{crs}} = \left\{1, \{\gamma_\ell\}_{\ell\in[L]}, (\mathtt{t}_1, \ldots, \mathtt{t}_{n_2})\right\}$,
   (b) $\mathcal{L}_2^{\mathsf{key}} = \left\{\left(\mathtt{s}_{\ell,1}^c, \ldots, \mathtt{s}_{\ell,n_1}^c\right)\mathbf{F}_k(\mathtt{t}_1, \ldots, \mathtt{t}_{n_2})^{\mathtt{T}} + \gamma_k\mathtt{w}_\ell^c = \gamma_k\mathtt{w}_\ell^c + \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} f_{i,j}^{(k)}\mathtt{s}_{\ell,i}^c\mathtt{t}_j\right\}_{\substack{c\in[Q_{\mathsf{k}}],\ell\in H \\ k\in[L]\setminus\{\ell\}}}$.
   (c) $\mathcal{L}_2^{\mathsf{ct}} = \left\{\left(\mathtt{a}y_j^{(0)} - \mathtt{b}\alpha\mathtt{t}_j\right), \left(\mathtt{c}y_j^{(0)} - \mathtt{d}\alpha\mathtt{t}_j\right)\right\}_{j\in[n_2]}$.

	$\mathcal{L}_1$	$\mathcal{L}_2$
crs	$\boxed{\Delta_{\mathbf{M}} = ad - bc}$	$\boxed{g_2}\ ,\ \left\{\boxed{\gamma_1}, \dots, \boxed{\gamma_L}\right\}$ $\boxed{\mathbf{t}} = \left(\boxed{t_1}, \dots, \boxed{t_{n_2}}\right)$
$\underset{c \in [Q_{\mathsf{k}}]}{\{\mathsf{pk}_c\}}$	$\left\{ \begin{matrix} \boxed{\Delta_{\mathbf{M}} \mathbf{s}_\ell^c} = \boxed{(ad - bc)\mathbf{s}_\ell^c} \\ \boxed{\Delta_{\mathbf{M}} w_\ell^c} = \boxed{(ad - bc) w_\ell^c} \end{matrix} \right\}_{\ell \in H, c \in [Q_{\mathsf{k}}]}$	$\left\{ \boxed{\mathbf{F}_k(\mathbf{s}_\ell^c, \mathbf{t}) + \gamma_k w_\ell^c} \right\}_{\ell \in H, k \in [L] \setminus \{\ell\}, c \in [Q_{\mathsf{k}}]}$ $\left(\text{for } \left\{\mathsf{dk}_{\ell,k}^c\right\}_{\ell \in H, k \in [L] \setminus \{\ell\}, c \in [Q_{\mathsf{k}}]}\right)$
ct^*	$\boxed{(ad - bc)\alpha}$ (for C_1) , $\boxed{(ad - bc)\alpha(w_1 + \cdots + w_L)}$ (for C_2) , $\left\{ \begin{matrix} \boxed{d x_i^{(0)} - c\alpha s_i} \\ \boxed{-b x_i^{(0)} + a\alpha s_i} \end{matrix} \right\}_{i \in [n_1]}$ (for $\mathbf{C}_{3,i}, i \in [n_1]$)	$\left\{ \begin{matrix} \boxed{a y_j^{(0)} - b t_j} \\ \boxed{c y_j^{(0)} - d t_j} \end{matrix} \right\}_{j \in [n_2]}$ (for $\mathbf{C}_{4,j}, j \in [n_2]$)

Table 1: The above table shows all terms from the scheme for which handles are stored in the respective lists $\mathcal{L}_1$ and $\mathcal{L}_2$. Assume $\mathcal{A}$ issues some arbitrary polynomial number, Q_{k}, of key queries in the pre-challenge query phase (some of which may be corrupted). The table lists all the terms for each of these honestly sampled keys $\{\mathsf{pk}_c\}_{c \in [Q_{\mathsf{k}}]}$ received by $\mathcal{A}$ in the second row. Hence, these terms are also indexed with superscripts for the key query count $c \in [Q_{\mathsf{k}}]$ (along with the slot index, say $\ell \in H$, whose keys are honestly generated by the experiment). The terms corresponding to mpk and hsk_i are not shown, since their handles are publicly computable by $\mathcal{A}$ using the group oracles. Note that such terms correspond to keys for all the registered L slots, all of which (except at least one) may possibly be corrupted or maliciously generated. Hence, the individual variables in each of those terms in mpk and hsk_i are independent of the counter variable $c \in [Q_{\mathsf{k}}]$ respectively. The third row corresponds to the elements stored in the respective lists available from the challenge ciphertext ct^*.

However, when $\mathcal{A}$ issues any zero-test query via Zt_{T} oracle, the challenger replaces the formal variables with their corresponding elements from $\mathbb{Z}_p$. In this case, if the variable is not assigned a value in $\mathbb{Z}_p$, it samples the corresponding value from the same distribution as it did in $\mathcal{H}_1$ (except the elements in the functions $\{\mathbf{F}_k\}_{k \in [L] \setminus \{\ell\}}$ and the challenge message $\left(\mathbf{x}^{(0)}, \mathbf{y}^{(0)}\right)$ which are either fixed coefficients or constants in these polynomials). However, once a value is assigned to a variable, it is fixed throughout the rest of $\mathcal{H}_2$. We show in Lemma 4.9 that $\mathcal{H}_1 \equiv \mathcal{H}_2$.

Given the tuple $\mathsf{P} = (\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_{\mathsf{T}})$, we define the closure $\mathsf{C}(\mathcal{L}_{\mathsf{T}}) = \mathcal{L}_{\mathsf{T}} \cup \{V_1 \cdot V_2 \mid \forall V_1 \in \mathcal{L}_1, V_2 \in \mathcal{L}_2\}$. Basically, it is the set of (handles of) all multivariate polynomials from ζ

with variables representing elements in $\mathbb{G}_{\mathsf{T}}$ that $\mathcal{A}$ can compute querying Map on the handles it received for elements in $\mathcal{L}_1, \mathcal{L}_2$. We estimate the size of $\mathsf{C}(\mathcal{L}_{\mathsf{T}})$. By definition, we have $|\mathsf{C}(\mathcal{L}_{\mathsf{T}})| = |\mathcal{L}_{\mathsf{T}}| + |\mathcal{L}_1| \cdot |\mathcal{L}_2| = |\mathcal{L}_1| \cdot |\mathcal{L}_2|$ (as $|\mathcal{L}_{\mathsf{T}}| = 0$ in our scheme).

$$\begin{aligned} |\mathcal{L}_1| &= |\mathcal{L}_1^{\mathsf{crs}}| + \left|\mathcal{L}_1^{\mathsf{key}}\right| + \left|\mathcal{L}_1^{\mathsf{ct}}\right| \\ &\le 1 + \{(n_1+1)\cdot Q_{\mathsf{k}} \cdot |H|\} + (2n_1+2) = (n_1+1)\cdot Q_{\mathsf{k}}\cdot|H| + 2n_1 + 3, \text{ and} \\ |\mathcal{L}_2| &= |\mathcal{L}_2^{\mathsf{crs}}| + \left|\mathcal{L}_2^{\mathsf{key}}\right| + \left|\mathcal{L}_2^{\mathsf{ct}}\right| \\ &\le (1+L+n_2) + \{(L-1)\cdot Q_{\mathsf{k}}\cdot|H|\} + 2n_2 = (L-1)\cdot Q_{\mathsf{k}}\cdot|H| + L + 3n_2 \end{aligned}$$

For brevity, we do not state $\mathsf{C}(\mathcal{L}_{\mathsf{T}})$ explicitly with all possible cross combinations of the terms from $\mathcal{L}_1, \mathcal{L}_2$. But by inspection, we can see that the maximal total degree of a term in $\mathsf{C}(\mathcal{L}_{\mathsf{T}})$ is $d = 6$. In particular, these corresponding terms are as follows:

1. $\left[(\mathtt{ad}-\mathtt{bc})\alpha\mathtt{w}\cdot\left(\gamma_k \mathtt{w}_\ell^c + \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} f_{i,j}^{(k)}\mathtt{s}_{\ell,i}^c\mathtt{t}_j\right)\right]_{\mathsf{T}}$ for any $\ell \in H, k \in [L]\setminus\{\ell\}, c \in [Q_{\mathsf{k}}]$,
2. $[(\mathtt{ad}-\mathtt{bc})\alpha\mathtt{w}\cdot(-\mathtt{bt}_j)]_{\mathsf{T}}$ and $[(\mathtt{ad}-\mathtt{bc})\alpha\mathtt{w}\cdot(-\mathtt{dt}_j)]_{\mathsf{T}}$ for any $j \in [n_2]$,

where $\mathtt{w} = \sum_{\ell\in[L]}\mathtt{w}_\ell$ corresponds to sum of the terms $\mathtt{w}_\ell$ from the actual aggregated keys. Further, any handle submitted by $\mathcal{A}$ to the Zt_{T} oracle during its interaction refers to a polynomial $\Phi \in \zeta$ as

$$\Phi\left(\{\gamma_\ell\}_{\ell\in[L]}, (\mathtt{t}_1,\ldots,\mathtt{t}_{n_2}), \{(\mathtt{s}_{\ell,1}^c,\ldots,\mathtt{s}_{\ell,n_1}^c)\}_{\ell\in H, c\in[Q_{\mathsf{k}}]}, \{\mathtt{w}_\ell^c\}_{\ell\in H,c\in[Q_{\mathsf{k}}]}, \alpha,\mathtt{a},\mathtt{b},\mathtt{c},\mathtt{d}\right) = \sum_{\theta\in\mathsf{C}(\mathcal{L}_{\mathsf{T}})} \eta_\Theta\Theta,$$

where the coefficients $\{\eta_\Theta \in \mathbb{Z}_p\}_{\Theta\in\mathsf{C}(\mathcal{L}_{\mathsf{T}})}$ can be computed efficiently. Note that all the terms in $\mathsf{C}(\mathcal{L}_{\mathsf{T}})$ are distinct, so the coefficients η_Θ are unique.

Hybrid $\mathcal{H}_3$: In this hybrid, *all* queries to Zt_{T} oracle are answered using formal variables. Namely, the challenger returns 1 for any Zt_{T} query on a handle to some polynomial $\Phi \in \zeta$ (with fixed elements from $\{\mathbf{F}_k\}_{k\in[L]\setminus\{\ell\}}$ or $\left(\mathbf{x}^{(0)},\mathbf{y}^{(0)}\right)$), if:

$$\Phi\left(\{\gamma_\ell\}_{\ell\in[L]}, (\mathtt{t}_1,\ldots,\mathtt{t}_{n_2}), \{(\mathtt{s}_{\ell,1}^c,\ldots,\mathtt{s}_{\ell,n_1}^c)\}_{\ell\in H, c\in[Q_{\mathsf{k}}]}, \{\mathtt{w}_\ell^c\}_{\ell\in H,c\in[Q_{\mathsf{k}}]}, \alpha,\mathtt{a},\mathtt{b},\mathtt{c},\mathtt{d}\right) = 0$$

We show in Lemma 4.10 that $\mathcal{H}_2 \approx_s \mathcal{H}_3$.

Hybrid $\mathcal{H}_4$: In this hybrid, we switch the encryption of $(\mathbf{x}^{(0)},\mathbf{y}^{(0)})$ to $(\mathbf{x}^{(1)},\mathbf{y}^{(1)})$. This is the game corresponding to bit $b = 1$ in the SGM. We show in Lemma 4.11 that $\mathcal{H}_3 \approx_s \mathcal{H}_4$.

Hybrid $\mathcal{H}_5$: In this hybrid, the challenger moves from the SGM to GGM. Hence, we have $\mathcal{H}_4 \approx_s \mathcal{H}_5$ with a proof similar to that of Lemmas 4.9 and 4.10, but in the reverse order.

Hybrid $\mathcal{H}_6$: Scale everything back by $\Delta_{\mathbf{M}}^{-1}$ like $\mathcal{H}_0$. This is the game corresponding to bit $b = 1$ in the GGM. Following a similar transition from $\mathcal{H}_0$ to $\mathcal{H}_1$, but in the reverse order, we have $\mathcal{H}_5 \approx_s \mathcal{H}_6$.

Lemma 4.9. $\mathcal{H}_1$ and $\mathcal{H}_2$ are perfectly indistinguishable.

Proof. Note that $\mathcal{A}$ sees the same handles in both $\mathcal{H}_1$ and $\mathcal{H}_2$. So it can notice a difference between the hybrids only if some zero-test query via the $\mathsf{Zt_T}$ oracle is answered differently. However, these zero-test queries are answered using values sampled from the same distribution in both the hybrids. Thus $\mathcal{A}$'s view remains the same in both the hybrids. $\square$

Lemma 4.10. $\mathcal{H}_2$ and $\mathcal{H}_3$ are statistisally indistinguishable.

Proof. $\mathcal{H}_2$ and $\mathcal{H}_3$ differs only when $\mathcal{A}$ submits a handle for some $\Phi \in \zeta$ satisfying

$$\Phi\left(\{\gamma_\ell\}_{\ell\in[L]}, (t_1,\ldots,t_{n_2}), \{(s^c_{\ell,1},\ldots,s^c_{\ell,n_1})\}_{\ell\in H, c\in[Q_\mathsf{k}]}, \{w^c_\ell\}_{\ell\in H,c\in[Q_\mathsf{k}]}, \alpha, a, b, c, d\right) = 0, \text{ and}$$
$$\Phi\left(\{\gamma_\ell\}_{\ell\in[L]}, (\mathtt{t}_1,\ldots,\mathtt{t}_{n_2}), \{(\mathtt{s}^c_{\ell,1},\ldots,\mathtt{s}^c_{\ell,n_1})\}_{\ell\in H, c\in[Q_\mathsf{k}]}, \{\mathtt{w}^c_\ell\}_{\ell\in H,c\in[Q_\mathsf{k}]}, \alpha, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\right) \neq 0$$

to the $\mathsf{Zt_T}$ oracle. Denote this event as $\mathbf{E}_{2,3}$. It suffices to bound the probability of $\mathbf{E}_{2,3}$ occurring in $\mathcal{H}_2(\lambda)$. For this, recall the maximal total degree of any polynomial $\Phi \in \zeta$ that could be formed by linear combinations of the terms in $\mathsf{C}(\mathcal{L}_\mathsf{T})$ is $d = 6$ (see Items 1 and 2 on the preceding page). Further, note that all the variables in all such polynomials are answered with independent and uniformly random values from $\mathbb{Z}_p$ in $\mathcal{H}_2$. Thus, by Schwartz-Zippel lemma, we have $\Pr[\mathbf{E}_{2,3}] \leq \frac{6}{p}$. As $\mathcal{A}$ issues $Q_{\mathtt{zt}}(\lambda)$ many $\mathsf{Zt_T}$ queries, a union bound implies that $\mathcal{A}$ can distinguish the two hybrids with probability at most $\frac{6\cdot Q_{\mathtt{zt}}(\lambda)}{p}$. Thus, $\mathcal{H}_2 \approx_s \mathcal{H}_3$. $\square$

Lemma 4.11. $\mathcal{H}_3$ and $\mathcal{H}_4$ are statistically indistinguishable, given $(\mathbf{x}^{(0)})^\mathtt{T}\mathbf{F}_k\mathbf{y}^{(0)} = (\mathbf{x}^{(1)})^\mathtt{T}\mathbf{F}_k\mathbf{y}^{(1)}$ for all $k \in C \cup M$.

Proof. In both hybrids $\mathcal{H}_3$ and $\mathcal{H}_4$, $\mathcal{A}$ interacts with $\mathcal{C}$ in the SGM. In particular, all elements from $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_\mathsf{T}$ are treated symbolically and indexed by their discrete logarithms. The only information that $\mathcal{A}$ can learn in the SGM is by querying the $\mathsf{Zt_T}$ oracle. Note that the only change from $\mathcal{H}_3$ to $\mathcal{H}_4$ is in the challenge ciphertext components. Hence, w.l.o.g., we focus mainly only on *successful* queries to the $\mathsf{Zt_T}$ oracle related to the coefficients of the ciphertext elements where the challenge message is embedded. Recall the challenge ciphertext $\mathsf{ct}^* = \left([C_1]_1, [C_2]_1, \{[\mathbf{C}_{3,i}]_1\}_{i\in[n_1]}, \{[\mathbf{C}_{4,j}]_2\}_{j\in[n_2]}\right)$ for the message $(\mathbf{x}^{(\beta)}, \mathbf{y}^{(\beta)}), \beta \in \{0,1\}$:

$$\mathsf{ct}^* = \left([(\mathtt{ad}-\mathtt{bc})\alpha]_1, [(\mathtt{ad}-\mathtt{bc})\alpha\mathtt{w}]_1, \left\{ \left[\begin{pmatrix} \mathtt{d}x_i^{(\beta)} - \mathtt{c}\alpha\mathtt{s}_i \\ -\mathtt{b}x_i^{(\beta)} + \mathtt{a}\alpha\mathtt{s}_i \end{pmatrix} \right]_1 \right\}_{i\in[n_1]}, \left\{ \left[\begin{pmatrix} \mathtt{a}y_j^{(\beta)} - \mathtt{b}\mathtt{t}_j \\ \mathtt{c}y_j^{(\beta)} - \mathtt{d}\mathtt{t}_j \end{pmatrix} \right]_2 \right\}_{j\in[n_2]} \right),$$

where $\mathbf{M} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \leftarrow \mathsf{GL}_2(\mathbb{Z}_p)$. The matrix $\mathbf{M}$ occurs only in the terms $\{[\mathbf{C}_{4,j}]_2\}_{j\in[n_2]}$ and $\mathbf{M}^{-1}$ only in the elements $\{[\mathbf{C}_{3,i}]_1\}_{i\in[n_1]}$ of ct^*. We therefore first show that the *only* way to annihilate terms related to a, b, c, d (i.e. the matrices $\mathbf{M}$ and $\mathbf{M}^{-1}$) is to pair the elements $[\mathbf{C}_{3,i}]_1$ with $[\mathbf{C}_{4,j}]_2$. For this, let us define:

$$\begin{aligned} [\mathbf{C}_{3,i}]_1^\mathtt{T} = \left(\left[\mathtt{d}x_i^{(\beta)} - \mathtt{c}\alpha\mathtt{s}_i\right]_1, \left[-\mathtt{b}x_i^{(\beta)} + \mathtt{a}\alpha\mathtt{s}_i\right]_1 \right) &\quad:=\quad \left(\left[c_{3,i}^{(1)}\right]_1, \left[c_{3,i}^{(2)}\right]_1 \right) \\ \text{and } [\mathbf{C}_{4,j}]_2^\mathtt{T} = \left(\left[\mathtt{a}y_j^{(\beta)} - \mathtt{b}\mathtt{t}_j\right]_2, \left[\mathtt{c}y_j^{(\beta)} - \mathtt{d}\mathtt{t}_j\right]_2 \right) &\quad:=\quad \left(\left[c_{4,j}^{(1)}\right]_2, \left[c_{4,j}^{(2)}\right]_2 \right). \end{aligned}$$

Claim 4.12. For all $i \in [n_1], j \in [n_2], z \in [2]$, the coefficients of the terms $\left[c_{4,j}^{(z)}\right]_2$ that are *not* paired with matching terms $\left[c_{3,i}^{(z)}\right]_1$ must be equal to 0.

Proof. Recall the lists from Items 1 and 2 on page 18. The only symbolic terms that $\mathcal{A}$ can access in $\mathcal{L}_1$, apart from the terms in $[\mathbf{C}_{3,i}]_1$ are:

1. $[\mathtt{ad} - \mathtt{bc}]_1$.
2. $\left(\left[(\mathtt{ad} - \mathtt{bc})\mathtt{s}^c_{\ell,1}\right]_1, \ldots, \left[(\mathtt{ad} - \mathtt{bc})\mathtt{s}^c_{\ell,n_1}\right]_1\right)$ and $[(\mathtt{ad} - \mathtt{bc})\mathtt{w}^c_\ell]_1$ for all $c \in [Q_k]$ and $\ell \in H$.
3. $[(\mathtt{ad} - \mathtt{bc})\alpha]_1$.
4. $[(\mathtt{ad} - \mathtt{bc})\alpha\mathtt{w}]_1$.
5. Any arbitrary linear combination among the above items (and possibly with $\left[c^{(1)}_{3,i}\right]_1$ or $\left[c^{(2)}_{3,i}\right]_1$).

Observe that Items 1 to 4 above all provide linearly independent terms symbolically. In particular, they do not cancel out internally as well as with each other. We will now establish that they cannot cancel even when $\mathcal{A}$ uses the Map oracle to form products with the terms in $[\mathbf{C}_{4,j}]_2 \in \mathcal{L}_2$. Below we inspect all possible pairings of the terms in $[\mathbf{C}_{4,j}]_2$ and show that they have linearly independent symbolic terms that cannot be cancelled out, so long as we forbid the correct terms $[\mathbf{C}_{3,i}]_1 \in \mathcal{L}_1$ in the pairing. In other words, we focus on the polynomial $\left[(\mathtt{ad} - \mathtt{bc}) \cdot \left(x^{(\beta)}_i y^{(\beta)}_j - \alpha\mathtt{s}_i\mathtt{t}_j\right)\right]_{\mathsf{T}}$, which is present only in the terms representing a *correctly formed* pairing $\left(\left[c^{(1)}_{3,i}\right]_1 \left[c^{(1)}_{4,j}\right]_2 + \left[c^{(2)}_{3,i}\right]_1 \left[c^{(1)}_{4,j}\right]_2\right)$ between $[\mathbf{C}_{3,i}]_1$ and $[\mathbf{C}_{4,j}]_2$. We divide the inspection in three cases.

Case 1 – $\left[c^{(1)}_{4,j}\right]_2$ is paired with terms from Items 1 to 4: In this case, we pair $\left[c^{(1)}_{4,j}\right]_2$ with any term that is not in $[\mathbf{C}_{3,i}]_1$.

1. $[\mathtt{ad} - \mathtt{bc}]_1 \left[c^{(1)}_{4,j}\right]_2$: Here we have *unique* terms $\mathtt{a}^2\mathtt{d}$, $-\mathtt{abc}$, $-\mathtt{abdt}_j$ and $\mathtt{b}^2\mathtt{ct}_j$.
2. $\left[(\mathtt{ad} - \mathtt{bc})\mathtt{s}^c_{\ell,i}\right]_1 \left[c^{(1)}_{4,j}\right]_2$: Here we have *unique* terms $\mathtt{a}^2\mathtt{ds}^c_{\ell,i}$, $-\mathtt{abcs}^c_{\ell,i}$, $-\mathtt{abds}^c_{\ell,i}\mathtt{t}_j$ and $\mathtt{b}^2\mathtt{cs}^c_{\ell,i}\mathtt{t}_j$.
3. $[(\mathtt{ad} - \mathtt{bc})\mathtt{w}^c_\ell]_1 \left[c^{(1)}_{4,j}\right]_2$: Here we have *unique* terms $\mathtt{a}^2\mathtt{dw}^c_\ell$, $-\mathtt{abcw}^c_\ell$, $-\mathtt{abdw}^c_\ell\mathtt{t}_j$ and $\mathtt{b}^2\mathtt{cw}^c_\ell\mathtt{t}_j$.
4. $[(\mathtt{ad} - \mathtt{bc})\alpha]_1 \left[c^{(1)}_{4,j}\right]_2$: Here we have *unique* terms $\mathtt{a}^2\mathtt{d}\alpha$, $-\mathtt{abc}\alpha$, $-\mathtt{abd}\alpha\mathtt{t}_j$ and $\mathtt{b}^2\mathtt{c}\alpha\mathtt{t}_j$.
5. $[(\mathtt{ad} - \mathtt{bc})\alpha\mathtt{w}]_1 \left[c^{(1)}_{4,j}\right]_2$: Here we have *unique* terms $\mathtt{a}^2\mathtt{d}\alpha\mathtt{w}$, $-\mathtt{abc}\alpha\mathtt{w}$, $-\mathtt{abd}\alpha\mathtt{t}_j\mathtt{w}$ and $\mathtt{b}^2\mathtt{c}\alpha\mathtt{t}_j\mathtt{w}$.
6. $\left[c^{(1)}_{3,i}\right]_1 \left[c^{(1)}_{4,j}\right]_2$: Here we have *unique* terms $\mathtt{ad}$, $-\mathtt{bdt}_j$, $-\mathtt{ac}\alpha\mathtt{s}_i$ and $\mathtt{bc}\alpha\mathtt{s}_i\mathtt{t}_j$. Note that this is the one that we exclude, but we must still need to make sure that it does not cancel out with the other pairings from Items 1 to 5 above. Due to the absence of degree 2 "literals" (like $\mathtt{a}^2$ or $\mathtt{b}^2$) and the presence of unique combinations of $\alpha, \mathtt{s_i}$ and $\mathtt{t_j}$, these monomials cannot be cancelled by any of the terms generated above.

Case 2 – $\left[c^{(2)}_{4,j}\right]_2$ is paired with terms from Items 1 to 4: In this case, we pair $\left[c^{(2)}_{4,j}\right]_2$ with any term that is not in $[\mathbf{C}_{3,i}]_1$.

1. $[\mathtt{ad} - \mathtt{bc}]_1 \left[c^{(2)}_{4,j}\right]_2$: Here we have *unique* terms $\mathtt{acd}$, $-\mathtt{ad}^2\mathtt{t}_j$, $-\mathtt{bc}^2$ and $\mathtt{bcdt}_j$.

2. $\left[(\mathtt{ad}-\mathtt{bc})\mathtt{s}^c_{\ell,i}\right]_1 \left[c^{(2)}_{4,j}\right]_2$: Here we have *unique* terms $\mathtt{acds}^c_{\ell,i}$, $-\mathtt{ad}^2\mathtt{s}^c_{\ell,i}\mathtt{t}_j$, $-\mathtt{bc}^2\mathtt{s}^c_{\ell,i}$ and $\mathtt{bcds}^c_{\ell,i}\mathtt{t}_j$.
3. $[(\mathtt{ad}-\mathtt{bc})\mathtt{w}^c_\ell]_1 \left[c^{(2)}_{4,j}\right]_2$: Here we have *unique* terms $\mathtt{acdw}^c_\ell$, $-\mathtt{ad}^2\mathtt{w}^c_\ell\mathtt{t}_j$, $-\mathtt{bc}^2\mathtt{w}^c_\ell$ and $\mathtt{bcdw}^c_\ell\mathtt{t}_j$.
4. $[(\mathtt{ad}-\mathtt{bc})\alpha]_1 \left[c^{(2)}_{4,j}\right]_2$: Here we have *unique* terms $\mathtt{acd}\alpha$, $-\mathtt{ad}^2\alpha\mathtt{t}_j$, $-\mathtt{bc}^2\alpha$ and $\mathtt{bcd}\alpha\mathtt{t}_j$.
5. $[(\mathtt{ad}-\mathtt{bc})\alpha\mathtt{w}]_1 \left[c^{(2)}_{4,j}\right]_2$: Here we have *unique* terms $\mathtt{acd}\alpha\mathtt{w}$, $-\mathtt{ad}^2\alpha\mathtt{t}_j\mathtt{w}$, $-\mathtt{bc}^2\alpha\mathtt{w}$ and $\mathtt{bcd}\alpha\mathtt{t}_j\mathtt{w}$.
6. $\left[c^{(2)}_{3,i}\right]_1 \left[c^{(2)}_{4,j}\right]_2$: Here we have *unique* terms $\mathtt{ac}\alpha\mathtt{s}_i$, $-\mathtt{ad}\alpha\mathtt{s}_i\mathtt{t}_j$, $-\mathtt{bc}$ and $\mathtt{bdt}_j$. Similar to case 1, though we exclude this, we must still need to make sure that it does not cancel out with the other pairings from Items 1 to 5 on the previous page in case 1 and Items 1 to 5 on pages 22–23 in this case above. Due to the absence of degree 2 "literals" (like $\mathtt{c}^2$ or $\mathtt{d}^2$) and the presence of unique combinations of α, $\mathtt{s_i}$ and $\mathtt{t_j}$, these monomials cannot be cancelled by any of the terms generated above.

Case 3 – $\left[c^{(1)}_{4,j}\right]_2$ is paired with $\left[c^{(2)}_{3,i}\right]_1$ or vice-versa: This is a simpler case, where we consider pairing the terms in $[\mathbf{C}_{3,i}]_1$ and $[\mathbf{C}_{4,j}]_2$, but in the wrong order.

1. $\left[c^{(2)}_{3,i}\right]_1 \left[c^{(1)}_{4,j}\right]_2$: Here we have *unique* terms $-\mathtt{ab}$, $\mathtt{b}^2\mathtt{t}_j$, $\mathtt{a}^2\alpha\mathtt{s}_i$ and $-\mathtt{ab}\alpha\mathtt{s}_i\mathtt{t}_j$.
2. $\left[c^{(1)}_{3,i}\right]_1 \left[c^{(2)}_{4,j}\right]_2$: Here we have *unique* terms $\mathtt{cd}$, $-\mathtt{d}^2\mathtt{t}_j$, $-\mathtt{c}^2\alpha\mathtt{s}_i$ and $\mathtt{cd}\alpha\mathtt{s}_i\mathtt{t}_j$.

These terms are again linearly independent symbolically from all the elements resulting from pairing the terms different than $\left[c^{(1)}_{4,j}\right]_2$ and $\left[c^{(2)}_{3,i}\right]_1$.

Because of symbolic linear independence, any arbitrary linear combination of the above terms cannot cancel each other, unless all their coefficients are identically 0. This proves Claim 4.12. □

We now note that upon pairing any arbitrarily scaled terms $[\eta_j\mathbf{C}_{4,j}]_2$ with $[\eta_i\mathbf{C}_{3,i}]_1$ for some $\eta_i, \eta_j \in \mathbb{Z}_p$ and further combining them linearly with more arbitrary scalars, say $\theta_{i,j} \in \mathbb{Z}_p$, any adversary can get access to the following polynomial in $\mathcal{L}_\mathsf{T}$:

$$
\begin{aligned}
&\sum_{\substack{i\in[n_1]\\ j\in[n_2]}} \theta_{i,j}\cdot\left(\left[\eta_i c^{(1)}_{3,i}\right]_1\left[\eta_j c^{(1)}_{4,j}\right]_2 + \left[\eta_i c^{(2)}_{3,i}\right]_1\left[\eta_j c^{(2)}_{4,j}\right]_2\right)\\
=&\sum_{\substack{i\in[n_1]\\ j\in[n_2]}} \theta_{i,j}\eta_i\eta_j\left(\left[\eta_i c^{(1)}_{3,i}\right]_1\left[\eta_j c^{(1)}_{4,j}\right]_2 + \left[\eta_i c^{(2)}_{3,i}\right]_1\left[\eta_j c^{(2)}_{4,j}\right]_2\right)\\
=&\sum_{\substack{i\in[n_1]\\ j\in[n_2]}} \theta_{i,j}\eta_i\eta_j\left\{(\mathtt{ad}-\mathtt{bc})\left(x^{(\beta)}_i y^{(\beta)}_j - \alpha\mathtt{s}_i\mathtt{t}_j\right)\right\}\\
=&\sum_{\substack{i\in[n_1]\\ j\in[n_2]}} \left\{x^{(\beta)}_i\cdot\eta_{i,j}(\mathtt{ad}-\mathtt{bc})\cdot y^{(\beta)}_j - \eta_{i,j}\cdot(\mathtt{ad}-\mathtt{bc})\alpha\mathtt{s}_i\mathtt{t}_j\right\},\\
&\text{where}\quad \eta_{i,j} := \theta_{i,j}\eta_i\eta_j
\end{aligned}
$$

Recall $\mathtt{s}_i$ as the symbolic term for $s_i = \sum_{\ell \in H} s_{\ell,i}$ where H is the set of honestly sampled keys (all of which, except one, may be corrupt).[8] As there exists one honest, registered party, we have $\mathtt{s}_i \neq 0, \forall i \in [n_1]$. For $\mathcal{A}$ to obtain information about $x_i^{(\beta)}$ and $y_j^{(\beta)}$ for any i, j, it must annihilate the term involving $(\mathtt{ad} - \mathtt{bc})\alpha\mathtt{s}_i\mathtt{t}_j$ in the above expression. Hence, it is enough to consider Zt_T queries of the form

$$\Omega + \sum_{i \in [n_1], j \in [n_2]} \left\{ -\eta_{i,j} \cdot (\mathtt{ad} - \mathtt{bc})\alpha\mathtt{s}_i\mathtt{t}_\mathtt{j} \right\} = 0, \text{ for some } \Omega \in \zeta. \tag{1}$$

We now show structural properties of $\Omega \in \zeta$ required to make Eq. (1) a successful Zt_T query.

Claim 4.13. For any $k \in [L], \ell \in H \setminus \{k\}$, fix the polynomial $\Lambda_{\ell,k} \in \mathcal{L}_2^{\mathsf{key}}$ (from Item 2 on page 18) such that

$$\Lambda_{\ell,k} := (\mathtt{s}_{\ell,1}, \ldots, \mathtt{s}_{\ell,n_1})\, \mathbf{F}_k(\mathtt{t}_1, \ldots, \mathtt{t}_{n_2})^\mathsf{T} + \gamma_k \mathtt{w}_\ell = \gamma_k \mathtt{w}_\ell + \sum_{i \in [n_1], j \in [n_2]} f_{i,j}^{(k)} \mathtt{s}_{\ell,i}\mathtt{t}_j.$$

Then the polynomial Ω in Eq. (1) is of the form $\displaystyle\sum_{\substack{k \in C \cup M \\ \ell \in H}} \xi_{k,\ell} \cdot (\mathtt{ad} - \mathtt{bc})\alpha \cdot (\Lambda_{\ell,k} - \gamma_k \mathtt{w}_\ell)$ for $\xi_{k,\ell} \in \mathbb{Z}_p$.

Proof. We first note that the polynomial $\Omega \in \zeta$ is of the form

$$\Omega = \sum_{k \in [L], \ell \in H \setminus \{k\}} \xi_{k,\ell} \cdot (\mathtt{ad} - \mathtt{bc})\alpha \cdot \Lambda_{\ell,k} + \psi_k \cdot (\mathtt{ad} - \mathtt{bc})\alpha\mathtt{w} \cdot \gamma_k \tag{2}$$

for $\psi_k \in \mathbb{Z}_p$, where $\mathtt{w} = \sum_{\ell \in H} \mathtt{w}_\ell$. Eq. (2) follows from the observations below:

1. One way $\mathcal{A}$ can attempt to nullify information about the monomials $\eta_{i,j} \cdot (\mathtt{ad} - \mathtt{bc})\alpha\mathtt{s}_i\mathtt{t}_j$ in Eq. (1) is by querying the oracle Map on handles for (appropriately scaled) $[\mathtt{t}_j]_2 \in \mathcal{L}_2$ (available from crs) along with the same for either $\left[c_{3,i}^{(1)}\right]_1 = \left[\left(\mathtt{d}x_i^{(\beta)} - \mathtt{c}\alpha\mathtt{s}_i\right)\right]_1 \in \mathcal{L}_1$ or $\left[c_{3,i}^{(2)}\right]_1 = \left[\left(-\mathtt{b}x_i^{(\beta)} + \mathtt{a}\alpha\mathtt{s}_i\right)\right]_1 \in \mathcal{L}_1$ (available from ct^*). However, by inspection, we see that these create new linearly independent monomials $\mathtt{d}\mathtt{t}_j x_i^{(\beta)} \in \mathcal{L}_\mathsf{T}$ or $-\mathtt{b}\mathtt{t}_j x_i^{(\beta)} \in \mathcal{L}_\mathsf{T}$ that cannot be cancelled with other terms.

2. The other (potentially more useful) way to annihilate $\eta_{i,j} \cdot (\mathtt{ad} - \mathtt{bc})\alpha\mathtt{s}_i\mathtt{t}_j$ in Eq. (1) is to query the Map oracle on (appropriately scaled) handles for $[(\mathtt{ad} - \mathtt{bc})\alpha]_1 \in \mathcal{L}_1$ and that of $[\Lambda_{\ell,k}]_2 \in \mathcal{L}_2$. Invoking this pairing operation also leads to the extra monomials $[(\mathtt{ad} - \mathtt{bc})\alpha \cdot \gamma_k \mathtt{w}_\ell]_\mathsf{T} \in \mathcal{L}_\mathsf{T}$. By inspection, we see that these extra monomials can be cancelled out only by querying the oracle Map again on the (appropriately scaled) handles for $[\gamma_k]_2 \in \mathcal{L}_2$ (available from crs) with that of $[(\mathtt{ad} - \mathtt{bc})\alpha\mathtt{w}]_1 \in \mathcal{L}_1$ (available from ct^*). This leads to Eq. (2) above with the scaling factors $\xi_{k,\ell}, \psi_k \in \mathbb{Z}_p$.

Below, we analyse further structural properties of the scalars $\xi_{k,\ell}$ and ψ_k in Eq. (2).

[8] W.l.o.g., the coordinates of only the honestly sampled vectors $\mathtt{s}_\ell$ are considered, as $\mathcal{A}$ knows all other $\mathtt{s}_\ell$, for $\ell \in \mathcal{MC}$.

1. $\underline{\psi_k = -\xi_{k,\ell}, \forall k \in [L],\ \ell \in H \setminus \{k\}}$: First observe that for any fixed $k \in [L]$, we have

$$\psi_k \cdot (\mathtt{ad} - \mathtt{bc})\alpha\mathtt{w} \cdot \gamma_k = (\mathtt{ad} - \mathtt{bc})\alpha\gamma_k \cdot \psi_k\mathtt{w} = (\mathtt{ad} - \mathtt{bc})\alpha\gamma_k \cdot \sum_{\ell \in H} \psi_k \mathtt{w}_\ell. \tag{3}$$

Recall from the prior discussion that the sum of extra monomials $\sum_{k\in[L],\ell\in H\setminus\{k\}} \xi_{k,\ell}\cdot(\mathtt{ad} - \mathtt{bc})\alpha\gamma_k\mathtt{w}_\ell$ needs to be nullified by $\sum_{k\in[L]} \psi_k \cdot (\mathtt{ad} - \mathtt{bc})\alpha\mathtt{w} \cdot \gamma_k$. This is mainly because the terms $(\mathtt{ad} - \mathtt{bc})\alpha\gamma_k\mathtt{w}_\ell$ do not appear anywhere else. Now note that for a fixed choice of k, ℓ, the term $(\mathtt{ad} - \mathtt{bc})\alpha\gamma_k\mathtt{w}_\ell$ defines linearly independent monomials. Therefore, from Eq. (3) it must hold that

$$(\mathtt{ad} - \mathtt{bc})\alpha\gamma_k \sum_{\ell\in H} \psi_k\mathtt{w}_\ell = -\sum_{\ell\in H\setminus\{k\}} \xi_{k,\ell}(\mathtt{ad} - \mathtt{bc})\alpha\gamma_k\mathtt{w}_\ell = (\mathtt{ad} - \mathtt{bc})\alpha\gamma_k \sum_{\ell\in H\setminus\{k\}} -\xi_{k,\ell}\mathtt{w}_\ell.$$

This implies that $\psi_k = -\xi_{k,\ell}, \forall k \in [L], \ell \in H \setminus \{k\}$.

2. $\underline{\psi_k = \xi_{k,\ell} = 0, \forall k \in H \setminus C}$: First note that for any $k \in H \setminus C$ (i.e. an index k whose keys are sampled honestly by the challenger and who is not corrupt by $\mathcal{A}$), we have

$$\psi_k \cdot (\mathtt{ad} - \mathtt{bc})\alpha\gamma_k\mathtt{w} = \psi_k \cdot (\mathtt{ad} - \mathtt{bc})\alpha\gamma_k\mathtt{w}_k + \psi_k \cdot (\mathtt{ad} - \mathtt{bc})\alpha\gamma_k\mathtt{w}_{\neq k}, \text{ where } \mathtt{w}_{\neq k} = \sum_{\ell\in H\setminus\{k\}} \mathtt{w}_\ell.$$

By inspection, note that $\mathcal{A}$ cannot obtain the unique monomial $(\mathtt{ad} - \mathtt{bc})\alpha\gamma_k\mathtt{w}_k$ from the handles to its available elements for all $k \in H\setminus C$. Particularly, it has access only to the handles for elements $[\gamma_k]_2 \in \mathcal{L}_2$, $[(\mathtt{ad} - \mathtt{bc})]_1, [(\mathtt{ad} - \mathtt{bc})\alpha]_1, [(\mathtt{ad} - \mathtt{bc})\alpha\mathtt{w}]_1 \in \mathcal{L}_1$ and $[(\mathtt{ad} - \mathtt{bc})\mathtt{w}_k]_1 \in \mathcal{L}_1, \forall k \in H \setminus C$. This implies that $\psi_k = 0, \forall k \in H \setminus C$. As a direct consequence, we also have $\xi_{k,\ell} = 0, \forall k \in H \setminus C$.

The above analysis implies that the only *non-zero* coefficients $\xi_{k,\ell}$ in Eq. (2) are for the terms $(\Lambda_{\ell,k} - \gamma_k\mathtt{w}_\ell) = \sum_{\substack{i\in[n_1]\\ j\in[n_2]}} f^{(k)}_{i,j}\mathtt{s}_{\ell,i}\mathtt{t}_j$, where $k \in C \cup M$ and $\ell \in H$. This proves Claim 4.13. ☐

Claim 4.13 establishes the structural form of Ω for Eq. (1) to be a successful $\mathsf{Zt_T}$ query. Plugging in the value of $\Lambda_{\ell,k}$ in the expression for Ω in Claim 4.13, we get

$$\Omega = \sum_{\substack{k\in C\cup M\\ \ell\in H}} \xi_{k,\ell} \cdot (\mathtt{ad} - \mathtt{bc})\alpha \cdot \sum_{\substack{i\in[n_1]\\ j\in[n_2]}} f^{(k)}_{i,j}\mathtt{s}_{\ell,i}\mathtt{t}_j = (\mathtt{ad} - \mathtt{bc})\alpha \sum_{\substack{i\in[n_1]\\ j\in[n_2]}} \sum_{\substack{k\in C\cup M\\ \ell\in H}} \xi_{k,\ell} f^{(k)}_{i,j}\mathtt{s}_{\ell,i}\mathtt{t}_j \tag{4}$$

Recall that the purpose of Ω in Eq. (1) was to cancel out the term involving $(\mathtt{ad} - \mathtt{bc})\alpha\mathtt{s}_i\mathtt{t}_j$. In the following and final claim, we establish the relation between coefficients $\eta_{i,j}$ and $\xi_{k,\ell}$ from Eqs. (1) and (2) respectively.

Claim 4.14. $\eta_{i,j} = \sum_{k\in C\cup M} \xi_{k,\ell} f^{(k)}_{i,j}$.

Proof. Plugging in Ω from Eq. (4) to Eq. (1), we have:

$$\begin{aligned}
&(\mathtt{ad} - \mathtt{bc})\alpha \sum_{\substack{i\in[n_1]\\ j\in[n_2]}} \sum_{\substack{k\in C\cup M\\ \ell\in H}} \xi_{k,\ell} f^{(k)}_{i,j}\mathtt{s}_{\ell,i}\mathtt{t}_j + \sum_{\substack{i\in[n_1]\\ j\in[n_2]}} \left\{-\eta_{i,j} \cdot (\mathtt{ad} - \mathtt{bc})\alpha\mathtt{s}_i\mathtt{t}_\mathtt{j}\right\} = 0 \\
\Rightarrow &\sum_{\substack{i\in[n_1]\\ j\in[n_2]}} \eta_{i,j} \cdot \mathtt{s}_i\mathtt{t}_\mathtt{j} = \sum_{\substack{i\in[n_1]\\ j\in[n_2]}} \sum_{\substack{k\in C\cup M\\ \ell\in H}} \xi_{k,\ell} f^{(k)}_{i,j}\mathtt{s}_{\ell,i}\mathtt{t}_j \quad [\text{as } (\mathtt{ad} - \mathtt{bc})\alpha \text{ is not identically } 0]
\end{aligned}$$

Above, each monomial $\mathtt{s}_i\mathtt{t}_j$ are linearly independent. Thus, we have:

$$\eta_{i,j}\cdot \mathtt{s}_i\mathtt{t_j} = \sum_{\substack{k\in C\cup M\\ \ell\in H}} \xi_{k,\ell} f_{i,j}^{(k)} \mathtt{s}_{\ell,i}\mathtt{t}_j \quad \Longrightarrow \quad \eta_{i,j}\cdot \mathtt{s}_i = \sum_{\substack{k\in C\cup M\\ \ell\in H}} \xi_{k,\ell} f_{i,j}^{(k)} \mathtt{s}_{\ell,i} \quad \text{[as } \mathtt{t}_j \text{ is not identically 0]}$$

Plugging in $\mathtt{s}_i = \sum_{\ell\in H} \mathtt{s}_{\ell,i}$ above, we further get:

$$\sum_{\ell\in H} \eta_{i,j}\cdot \mathtt{s}_{\ell,i} = \sum_{\ell\in H}\left(\sum_{k\in C\cup M} \xi_{k,\ell} f_{i,j}^{(k)}\right)\mathtt{s}_{\ell,i} \quad \Longrightarrow \quad \sum_{\ell\in H} \mathtt{s}_{\ell,i}\left(\eta_{i,j} - \sum_{k\in C\cup M} \xi_{k,\ell} f_{i,j}^{(k)}\right) = 0$$

Again, since each $\mathtt{s}_{\ell,i}$ is linearly independent and there exists at least one registered, uncorrupted party, we have $\eta_{i,j} = \sum_{k\in C\cup M} \xi_{k,\ell} f_{i,j}^{(k)}$ as desired. □

Claim 4.14 shows that any successful Zt_T query in $\mathcal{H}_3$ will also be so in $\mathcal{H}_4$. Recall that for any slot $k\in C\cup M$ we have $(\mathbf{x}^{(0)})^\mathsf{T}\mathbf{F}_k\mathbf{y}^{(0)} = (\mathbf{x}^{(1)})^\mathsf{T}\mathbf{F}_k\mathbf{y}^{(1)}$. This, along with Claim 4.14, implies the following:

$$\begin{aligned}
\sum_{i\in[n_1],j\in[n_2]} x_i^{(0)}\cdot\eta_{i,j}(\mathtt{ad}-\mathtt{bc})\cdot y_j^{(0)} &= (\mathtt{ad}-\mathtt{bc})\sum_{i\in[n_1],j\in[n_2]} x_i^{(0)}\cdot\left(\sum_{k\in C\cup M}\xi_{k,\ell} f_{i,j}^{(k)}\right)\cdot y_j^{(0)}\\
&= (\mathtt{ad}-\mathtt{bc})\sum_{k\in C\cup M}\xi_{k,\ell}\left(\sum_{i\in[n_1],j\in[n_2]} f_{i,j}^{(k)} x_i^{(0)} y_j^{(0)}\right)\\
&= (\mathtt{ad}-\mathtt{bc})\sum_{k\in C\cup M}\xi_{k,\ell}(\mathbf{x}^{(0)})^\mathsf{T}\mathbf{F}_k\mathbf{y}^{(0)}\\
&= (\mathtt{ad}-\mathtt{bc})\sum_{k\in C\cup M}\xi_{k,\ell}(\mathbf{x}^{(1)})^\mathsf{T}\mathbf{F}_k\mathbf{y}^{(1)}\\
&= \sum_{i\in[n_1],j\in[n_2]} x_i^{(1)}\cdot\eta_{i,j}(\mathtt{ad}-\mathtt{bc})\cdot y_j^{(1)}
\end{aligned}$$

Thus, switching $\beta=0$ (in $\mathcal{H}_3$) to $\beta=1$ (in $\mathcal{H}_4$) does not yield any distinguishing advantage for $\mathcal{A}$. Hence, $\mathcal{H}_3 \approx_s \mathcal{H}_4$. □

This ends the proof of Theorem 4.8. □

4.3 RFE for Linear Functions

Let $n, L\in \mathsf{poly}(\lambda)$. For any $(\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_\mathsf{T},p,[1]_1,[1]_2,\cdot)$ output by $\mathsf{GGen}(1^\lambda)$, we construct in Fig. 4 an RFE for the message space $\mathcal{M}=\mathbb{Z}_p^n$, the class of linear functions $\mathcal{F}$ being

$$\left\{\left(f:\mathbb{Z}_p^n\to[\mathbb{Z}_p]_\mathsf{T}, f(\mathbf{x})\mapsto\left[\mathbf{x}^\mathsf{T}\mathbf{y} \bmod p\right]_\mathsf{T}\right):\mathbf{y}\in\mathbb{Z}_p^n\right\},$$

and (an upper bound of) L number of users. Since any $f\in\mathcal{F}, f(\mathbf{x})\mapsto\left[\mathbf{x}^\mathsf{T}\mathbf{y} \bmod p\right]_\mathsf{T}$ is fully described by $\mathbf{y}, \mathbb{G}_\mathsf{T}$ and p whereas $\mathbb{G}_\mathsf{T}, p$ are publicly fixed, we simply write $\mathbf{y}$ for such. We remark that the scheme can be trivially extended to one support the function class mapping to $\mathbf{x}^\mathsf{T}\mathbf{y} \bmod p$, i.e. in plain instead of as target group element, with appropriate bound B on the image space, by letting the decryption algorithm solving for the discrete log solution.

$\mathsf{Setup}(1^\lambda)$

$\mathcal{G} := (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_\mathsf{T}, p, [1]_1, [1]_2, \cdot) \leftarrow \mathsf{GGen}(1^\lambda)$

for $\ell \in [L]$: $\mathbf{w}_\ell \leftarrow\!\!\$\ \mathbb{Z}_p^n$; $r_\ell \leftarrow\!\!\$\ \mathbb{Z}_p$

return $\mathsf{crs} := \begin{pmatrix} \mathcal{G}, ([\mathbf{w}_\ell]_1)_{\ell\in[L]}, ([r_\ell]_2)_{\ell\in[L]} \\ ([r_k \mathbf{w}_\ell]_2)_{k,\ell\in[L],k\neq\ell} \end{pmatrix}$

$\mathsf{Dec}(\mathsf{sk}_k, \mathsf{hsk}_k, \mathsf{ct})$

$[d_0]_\mathsf{T} := [c_1]_1 [h_0]_2 - [c_0]_1 [h_1]_2$

$[d_1]_\mathsf{T} := [c_0]_1 [h_0]_2 \mathbf{v}_k^\mathsf{T} \mathbf{y}_k$

$[\mathbf{d}_2^\mathsf{T}]_\mathsf{T} := [\mathbf{c}_2^\mathsf{T}]_1 [h_0]_2 - [c_0]_1 [\mathbf{h}_2^\mathsf{T}]_2$

return $[\mathbf{d}_2^\mathsf{T}]_\mathsf{T} \cdot \mathbf{y}_k - ([d_0]_\mathsf{T} - [d_1]_\mathsf{T})$

$\mathsf{KGen}(\mathsf{crs}, \ell)$

$\mathsf{sk}_\ell := \mathbf{v}_\ell \leftarrow\!\!\$\ \mathbb{Z}_p^n$

$\mathsf{pk}_\ell := \left([\mathbf{v}_\ell]_1, ([r_k \mathbf{v}_\ell]_2)_{k\in[L]\setminus\ell}\right)$

return $(\mathsf{pk}_\ell, \mathsf{sk}_\ell)$

$\mathsf{Enc}(\mathsf{mpk}, \mathbf{x} \in \mathbb{Z}_p^n)$

$s \leftarrow\!\!\$\ \mathbb{Z}_p$

$[c_0]_1 := [s]_1$

$[c_1]_1 := \left[s \sum_{\ell\in[L]} (\mathbf{w}_\ell^\mathsf{T} + \mathbf{v}_\ell^\mathsf{T}) \mathbf{y}_\ell\right]_1$

$[\mathbf{c}_2^\mathsf{T}]_1 := \left[s \sum_{\ell\in[L]} \mathbf{w}_\ell^\mathsf{T} + \mathbf{x}^\mathsf{T}\right]_1$

return $\mathsf{ct} := ([c_0]_1, [c_1]_1, [\mathbf{c}_2^\mathsf{T}]_1)$

$\mathsf{Aggr}(\mathsf{crs}, (\mathsf{pk}_\ell, \mathbf{y}_\ell)_{\ell\in[L]})$

for $k \in [L]$:

$[h_{0,k}]_2 := [r_k]_2$

$[h_{1,k}]_2 := \left[r_k \sum_{\ell\in[L]\setminus\{k\}} (\mathbf{w}_\ell^\mathsf{T} + \mathbf{v}_\ell^\mathsf{T}) \mathbf{y}_\ell\right]_2$

$[\mathbf{h}_{2,k}^\mathsf{T}]_2 := \left[r_k \sum_{\ell\in[L]\setminus\{k\}} \mathbf{w}_\ell^\mathsf{T}\right]_2$

$\mathsf{mpk} := \left(\left[\sum_{\ell\in[L]} (\mathbf{w}_\ell^\mathsf{T} + \mathbf{v}_\ell^\mathsf{T}) \mathbf{y}_\ell\right]_1, \left[\sum_{\ell\in[L]} \mathbf{w}_\ell^\mathsf{T}\right]_1\right)$

$\mathsf{hsk}_k := ([h_{0,k}]_2, [h_{1,k}]_2, [\mathbf{h}_{2,k}^\mathsf{T}]_2)$

return $(\mathsf{mpk}, (\mathsf{sk}_k)_{k\in[L]})$

Figure 4: RLFE construction.

Theorem 4.15. RLFE (Fig. 4) is strongly compact (Definition 4.3).

Proof. Assuming that the groups description $\mathcal{G}$ and each element in $\mathbb{Z}_p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_\mathsf{T}$ are of description size $\mathsf{poly}(\lambda)$, we count the size of mpk, hsk_ℓ, and ct: $|\mathsf{mpk}|, |\mathsf{hsk}_\ell|, |\mathsf{ct}| = n \cdot \mathsf{poly}(\lambda)$. Notably, they are of size independent of L. ∎

Theorem 4.16. RLFE (Fig. 4) is correct (Definition 4.2).

Proof. Observe that for any decryptor $k \in [L]$,

$$\begin{aligned}
[d_0]_\mathsf{T} &= \left[s \sum_{\ell\in[L]} (\mathbf{w}_\ell^\mathsf{T} + \mathbf{v}_\ell^\mathsf{T}) \mathbf{y}_\ell\right]_1 [r_k]_2 - [s]_1 \left[r_k \sum_{\ell\in[L]\setminus\{k\}} (\mathbf{w}_\ell^\mathsf{T} + \mathbf{v}_\ell^\mathsf{T}) \mathbf{y}_\ell\right]_2 \\
&= \left[s r_k \sum_{\ell\in[L]} (\mathbf{w}_\ell^\mathsf{T} + \mathbf{v}_\ell^\mathsf{T}) \mathbf{y}_\ell\right]_\mathsf{T} - \left[s r_k \sum_{\ell\in[L]\setminus\{k\}} (\mathbf{w}_\ell^\mathsf{T} + \mathbf{v}_\ell^\mathsf{T}) \mathbf{y}_\ell\right]_\mathsf{T} = \left[s r_k (\mathbf{w}_k^\mathsf{T} + \mathbf{v}_k^\mathsf{T}) \mathbf{y}_k\right]_\mathsf{T}, \\
[d_1]_\mathsf{T} &= [s]_1 [r_k]_2 \mathbf{v}_k^\mathsf{T} \mathbf{y}_k = \left[s r_k \mathbf{v}_k^\mathsf{T} \mathbf{y}_k\right]_\mathsf{T}, \\
[\mathbf{d}_2^\mathsf{T}]_\mathsf{T} &= \left[s \sum_{\ell\in[L]} \mathbf{w}_\ell^\mathsf{T} + \mathbf{x}^\mathsf{T}\right]_1 [r_k]_2 - [s]_1 \left[r_k \sum_{\ell\in[L]\setminus\{k\}} \mathbf{w}_\ell^\mathsf{T}\right]_2 \\
&= \left[s r_k \sum_{\ell\in[L]} \mathbf{w}_\ell^\mathsf{T} + r_k \mathbf{x}^\mathsf{T}\right]_\mathsf{T} - \left[s r_k \sum_{\ell\in[L]\setminus\{k\}} \mathbf{w}_\ell^\mathsf{T}\right]_\mathsf{T} = \left[s r_k \mathbf{w}_k^\mathsf{T} + r_k \mathbf{x}^\mathsf{T}\right]_\mathsf{T}.
\end{aligned}$$

Therefore decryption outputs

$$\begin{aligned}
[\mathbf{d}_2^\mathsf{T}]_\mathsf{T} \cdot \mathbf{y}_k - ([d_0]_\mathsf{T} - [d_1]_\mathsf{T}) &= \left[s r_k \mathbf{w}_k^\mathsf{T} + r_k \mathbf{x}^\mathsf{T}\right]_\mathsf{T} \cdot \mathbf{y}_k - \left(\left[s r_k (\mathbf{w}_k^\mathsf{T} + \mathbf{v}_k^\mathsf{T}) \mathbf{y}_k\right]_\mathsf{T} - \left[s r_k \mathbf{v}_k^\mathsf{T} \mathbf{y}_k\right]_\mathsf{T}\right) \\
&= \left[s r_k \mathbf{w}_k^\mathsf{T} \mathbf{y}_k + r_k \mathbf{x}^\mathsf{T} \mathbf{y}_k\right]_\mathsf{T} - \left[s r_k \mathbf{w}_k^\mathsf{T} \mathbf{y}_k\right]_\mathsf{T} = \left[r_k \mathbf{x}^\mathsf{T} \mathbf{y}_k\right]_\mathsf{T},
\end{aligned}$$

as desired. ∎

Our security proof relies on the following assumption.

Assumption 4.17. Let $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_\mathsf{T}, p, [1]_1, [1]_2) \leftarrow \mathsf{GGen}(1^\lambda)$. It holds that for any PPT $\mathcal{A}$

$$\left| \begin{array}{l} \Pr\left[\mathcal{A}\left([s]_1, ([a_\ell]_1, [r_\ell]_2)_{\ell\in[L]}, ([r_k a_\ell]_2)_{k,\ell\in[L],k\neq\ell}, \left[s\sum_{\ell\in[L]} a_\ell\right]_1\right) = 1\right] \\ - \Pr\left[\mathcal{A}\left([s]_1, ([a_\ell]_1, [r_\ell]_2)_{\ell\in[L]}, ([r_k a_\ell]_2)_{k,\ell\in[L],k\neq\ell}, [u]_1\right) = 1\right] \end{array} \right| \leq \mathsf{negl}(\lambda),$$

where $s, u, a_\ell, r_\ell \leftarrow\$\ \mathbb{Z}_p$ for all $\ell \in [L]$.

The above can be seen as a q-type variant (where $q = L$) of the DDH assumption generalised into the bilinear group setting: Removing all elements in $\mathbb{G}_2$, the statement is implied by DDH (over $\mathbb{G}_1$) which says that $[s]_1 [a_\ell]_1 \approx_c \$$ for any $\ell \in [L]$.

Remark 4.18. In Assumption 4.17, it is important that $\left[s\sum_{\ell\in[L]} a_\ell\right]_1$ sums over all $\ell \in [L]$ instead of any subset $S \subset [L]$. Otherwise, picking any $k \notin S$, one can distinguish $\left[s\sum_{\ell\in S} a_\ell\right]_1$ from random via the pairing equation $\left[s\sum_{\ell\in S} a_\ell\right]_1 [r_k]_2 \stackrel{?}{=} \sum_{\ell\in S} [s]_1 [r_k a_\ell]_2$. With $\left[s\sum_{\ell\in[L]} a_\ell\right]_1$ instead, since $[r_k a_k]_2$ for any $k \in [L]$ is not given out, the same attack does not apply.

In Appendix B, we prove that Assumption 4.17 holds in the generic group model.

Theorem 4.19. RLFE (Fig. 4) is selectively secure with static corruption (Definition 4.4) under Assumption 4.17.

Proof. We define the following hybrids:

- $\mathcal{H}_{b,0}$: This is same as the selective-security experiment for $b \in \{0, 1\}$, i.e. the distribution as in Fig. 4 encrypting $\mathbf{x}_b^\mathsf{T}$.
- $\mathcal{H}_{b,1}$: Same as $\mathcal{H}_{b,1}$, except that we compute $\left[\mathbf{c}_2^\mathsf{T}\right]_1$ as $\left[t\mathbf{x}_1^\mathsf{T} + (1-t)\mathbf{x}_0^\mathsf{T} + s\sum_{\ell\in[L]} \mathbf{k}_\ell^\mathsf{T}\overline{\mathbf{Z}}\right]_1$, where $(\mathbf{x}_0, \mathbf{x}_1)$ are the challenge messages from the adversary interacting with the experiment, $t, s \in \mathbb{Z}_p$ and $\mathbf{k}_\ell \in \mathbb{Z}_p^{n-1}$ are uniformly random, and $\overline{\mathbf{Z}} \in \mathbb{Z}_p^{n-1\times n}$ independent of b (defined below).

Notice that $\mathcal{H}_{0,1} \equiv \mathcal{H}_{1,1}$, since the distribution of all terms in $\left[\mathbf{c}_2^\mathsf{T}\right]_1$, hence also $\left[\mathbf{c}_2^\mathsf{T}\right]_1$, are independent of b. We show in the remaining that $\mathcal{H}_{b,0} \approx_c \mathcal{H}_{b,1}$ under Assumption 4.17, which completes the proof.

Suppose there exists a PPT $\mathcal{A}$ that distinguishes $\mathcal{H}_{b,0}$ and $\mathcal{H}_{b,1}$ with non-negligible probability. We construction a PPT $\mathcal{B}$ against Assumption 4.17.

On input a problem instance $\left([s]_1, ([a_\ell]_1, [r_\ell]_2)_{\ell\in[L]}, ([r_k a_\ell]_2)_{k,\ell\in[L],k\neq\ell}, [u]_1\right)$ where $[u]_1$ is either $\left[s\sum_{\ell\in[L]} a_\ell\right]_1$ or uniformly random, $\mathcal{B}$ proceeds as follows:

- Receive the pair of challenge messages $(\mathbf{x}_0, \mathbf{x}_1)$ and the set of corrupt users $C \subseteq [L]$ from $\mathcal{A}$.
- Let $\hat{\mathbf{x}} := \mathbf{x}_1 - \mathbf{x}_0$, let $\mathbf{B} := (\hat{\mathbf{x}} \mid \mathbf{Z})$ a basis of $\mathbb{Z}_p^n$, where $\mathbf{Z}$ is arbitrary basis of the kernel space $\hat{\mathbf{x}}^\perp$ of $\hat{\mathbf{x}}$.
- Sample random $\mathbf{k}_\ell \leftarrow \mathbb{Z}_p^{n-1}$ for all $\ell \in [L]$.
- Pass crs to $\mathcal{A}$ which is simulated as follows:

– For each $\ell \in [L]$, fetch $[a_\ell]_1$ and $([r_k]_2)_{k\in[L]\setminus\{\ell\}}$ from input and let

$$[\mathbf{w}_\ell]_1 :=([a_\ell]_1 \mid [\mathbf{k}_\ell^{\mathsf{T}}]_1)\mathbf{B}^{-1}, \qquad [r_k\mathbf{w}_\ell^{\mathsf{T}}]_2 := ([r_k a_\ell]_2 \mid [r_k]_2\, \mathbf{k}_\ell^{\mathsf{T}})\mathbf{B}^{-1} \text{ for all } k \in [L] \setminus \{\ell\}.$$

– Let $\mathsf{crs} := \big(\mathcal{G}, \{[\mathbf{w}_\ell]_1\}_{\ell\in[L]}, \{[r_\ell]_2\}_{\ell\in[L]}, \{[r_k\mathbf{w}_\ell]_2\}_{k,\ell\in[L],k\neq\ell}\big)$.

- For key query on user $\ell \in [L]$, if $K[\ell] = \perp$, same keys as follows:
  - If $\ell \in [L] \setminus C$ is not corrupt: Sample random $\mathbf{d}_\ell \leftarrow \mathbb{Z}_p^n$, fetch $[a_\ell]_1$ and $([r_k]_2)_{k\in[L]\setminus\{\ell\}}$ from input, and compute $\mathsf{pk}_\ell := \big([\mathbf{v}_\ell^{\mathsf{T}}]_1, [r_k\mathbf{v}_\ell^{\mathsf{T}}]_2\big)$ as

$$[\mathbf{v}_\ell^{\mathsf{T}}]_1 := [\mathbf{d}_\ell^{\mathsf{T}}]_1 - [(a_\ell|\mathbf{k}_\ell^{\mathsf{T}})]_1\,\mathbf{B}^{-1}, \qquad [r_k\mathbf{v}_\ell^{\mathsf{T}}]_2 := [r_k]_2\,\mathbf{d}_\ell^{\mathsf{T}} - ([r_k a_\ell]_2 \,|\, [r_k]_2\,\mathbf{k}_\ell^{\mathsf{T}})\mathbf{B}^{-1}.$$

  - If $\ell \in C$ is corrupt: Sample random $\mathbf{v}_\ell \in \mathbb{Z}_p^n$, fetch $([r_k]_2)_{k\in[L]\setminus\{\ell\}}$ from input, let $\mathsf{pk}_\ell := \big([\mathbf{v}_\ell^{\mathsf{T}}]_1, [r_k\mathbf{v}_\ell^{\mathsf{T}}]_2\big)$ and $\mathsf{sk}_\ell := \mathbf{v}_\ell$.

  Write the above to $K[\ell]$ and answer accordingly.

- Receive the message $(\mathbf{x}_0, \mathbf{x}_1)$, the registrations $(\mathsf{pk}_\ell, \mathbf{y}_\ell)_{\ell\in[L]}$, and randomness $(\mathbf{v}_\ell)_{\ell\in M}$ for malicious parties from $\mathcal{A}$. Verify that (1) $[L] \setminus M \subseteq K$, (2) for each $\ell \in M$ it holds that $\mathsf{pk}_\ell = \big([\mathbf{v}_\ell^{\mathsf{T}}]_1, [r_k\mathbf{v}_\ell^{\mathsf{T}}]_2\big)$ for $\mathbf{v}_\ell$ provided by $\mathcal{A}$, and (3) for all $\ell \in C\cup M$ it holds that $\mathbf{x}_0^{\mathsf{T}}\mathbf{y} = \mathbf{x}_1^{\mathsf{T}}\mathbf{y}$. If all checks pass, let $\mathsf{sk}_\ell = \mathbf{v}_\ell$ for the malicious $\ell \in M$, and simulate the challenge ciphertext ct^* as follows:
  - For each $\ell \in C \cup M$, write $\mathbf{y}_\ell =: \mathbf{B}\begin{pmatrix}0\\ \tilde{\mathbf{y}}_\ell\end{pmatrix}$ where $\tilde{\mathbf{y}}_\ell \in \mathbb{Z}_p^{n-1}$ (which is possible since $\mathbf{x}_0^{\mathsf{T}}\mathbf{y}_\ell = \mathbf{x}_1^{\mathsf{T}}\mathbf{y}_\ell$, equivalently $\mathbf{y}_\ell \in \hat{\mathbf{x}}^{\perp}$).
  - Fetch $[s]_1$ and $[u]_1$ from input, let $\mathsf{ct}^* := ([c_0]_1, [c_1]_1, [\mathbf{c}_2]_1)$ where $[c_0]_1 := [s]_1$ and

$$[c_1]_1 := [s]_1 \sum_{\ell\in[L]\setminus C\cup M} \mathbf{d}_\ell^{\mathsf{T}}\mathbf{y}_\ell + [s]_1 \sum_{\ell\in C\cup M} (\mathbf{k}_\ell^{\mathsf{T}}\tilde{\mathbf{y}}_\ell + \mathbf{v}_\ell^{\mathsf{T}}\mathbf{y}_\ell), \quad [\mathbf{c}_2^{\mathsf{T}}]_1 := \left([u]_1 \,\middle|\, [s]_1 \sum_{\ell\in[L]} \mathbf{k}_\ell^{\mathsf{T}}\right)\mathbf{B}^{-1} + [\mathbf{x}_b^{\mathsf{T}}]_1.$$

- Pass ct^* to $\mathcal{A}$ and return whatever $\mathcal{A}$ returns.

We analyse the outputs of $\mathcal{B}$. First, notice that the simulated outputs can be expressed as setting

$$\begin{aligned}\mathbf{w}_\ell^{\mathsf{T}} &= (a_\ell \mid \mathbf{k}_\ell^{\mathsf{T}})\mathbf{B}^{-1} \quad \text{for all } \ell \in [L]\\ \mathbf{v}_\ell^{\mathsf{T}} &= \mathbf{d}_\ell^{\mathsf{T}} - (a_\ell \mid \mathbf{k}_\ell^{\mathsf{T}})\mathbf{B}^{-1} \quad \text{for all } \ell \in [L] \setminus (C \cup M),\end{aligned}$$

then computing all components except $[\mathbf{c}_2]_1$ in the same way as in the scheme. In more details, using the above two equations, the outputs can be expressed as

$$\begin{aligned}\mathsf{crs}: \quad [r_k\mathbf{w}_\ell^{\mathsf{T}}]_2 &= [r_k(a_\ell \mid \mathbf{k}_\ell^{\mathsf{T}})\mathbf{B}^{-1}]_2 = ([r_k a_\ell]_2 \mid [r_k]_2\,\mathbf{k}_\ell^{\mathsf{T}})\mathbf{B}^{-1},\\ \mathsf{pk}_\ell, \ell \in [L] \setminus (C \cup M): \quad [r_k\mathbf{v}_\ell^{\mathsf{T}}]_2 &= [r_k(\mathbf{d}_\ell^{\mathsf{T}} - (a_\ell \mid \mathbf{k}_\ell^{\mathsf{T}}))\mathbf{B}^{-1}]_2 = [r_k]_2\,\mathbf{d}_\ell^{\mathsf{T}} - ([r_k a_\ell]_2 \mid [r_k]_2\,\mathbf{k}_\ell^{\mathsf{T}})\mathbf{B}^{-1}\end{aligned}$$

and for the challenge ciphertext ct^*,

$$\begin{aligned}
[c_1]_1 = &\ [s]_1 \sum_{\ell\in[L]\setminus(C\cup M)} \mathbf{d}_\ell^\mathsf{T}\mathbf{y}_\ell + [s]_1 \sum_{\ell\in(C\cup M)} (\mathbf{k}_\ell^\mathsf{T}\tilde{\mathbf{y}}_\ell + \mathbf{v}_\ell^\mathsf{T}\mathbf{y}_\ell) \\
= &\ [s]_1 \sum_{\ell\in[L]\setminus(C\cup M)} (\mathbf{v}_\ell^\mathsf{T} + (a_\ell \mid \mathbf{k}_\ell^\mathsf{T})\mathbf{B}^{-1})\mathbf{y}_\ell + [s]_1 \sum_{\ell\in(C\cup M)} ((a_\ell|\mathbf{k}_\ell^\mathsf{T})\mathbf{B}^{-1}\mathbf{y}_\ell + \mathbf{v}_\ell^\mathsf{T}\mathbf{y}_\ell) \\
= &\ \left[s\sum_{\ell\in[L]}((a_\ell|\mathbf{k}_\ell^\mathsf{T})\mathbf{B}^{-1} + \mathbf{v}_\ell^\mathsf{T})\mathbf{y}_\ell\right]_1 = \left[s\sum_{\ell\in[L]}(\mathbf{w}_\ell^\mathsf{T} + \mathbf{v}_\ell^\mathsf{T})\mathbf{y}_\ell\right]_1
\end{aligned}$$

where the second term in the second equality is due to $\mathbf{k}_\ell^\mathsf{T}\tilde{\mathbf{y}}_\ell = (a_\ell|\mathbf{k}_\ell^\mathsf{T})\mathbf{B}^{-1}\mathbf{B}\begin{pmatrix}0\\ \tilde{\mathbf{y}}_\ell\end{pmatrix} = (a_\ell|\mathbf{k}_\ell^\mathsf{T})\mathbf{B}^{-1}\mathbf{y}_\ell$.

Since $(a_\ell)_{\ell\in[L]}, (\mathbf{k}_\ell)_{\ell\in[L]}$ and $(\mathbf{d}_\ell)_{\ell\in[L]\setminus C}$ are all uniformly random, so are $(\mathbf{w}_\ell)_{\ell\in[L]}$ and $(\mathbf{v}_\ell)_{\ell\in[L]\setminus C}$. The keys $(\mathsf{pk}_\ell, \mathsf{sk}_\ell)$ for the corrupt users $\ell \in C$ are computed honestly. Therefore, all components of the simulated crs, $\mathsf{pk}_\ell, \mathsf{sk}_\ell$ as well as $[c_0]_1, [c_1]_1$ in ct^* are distributed same as in the scheme.

Finally we inspect $[\mathbf{c}_2^\mathsf{T}]_1$. Suppose $[u]_1 = \left[s\sum_{\ell\in[L]} a_\ell\right]_1$, then

$$[\mathbf{c}_2^\mathsf{T}]_1 = \left(\left[s\sum_{\ell\in[L]} a_\ell\right]_1 \middle| [s]_1 \sum_{\ell\in[L]} \mathbf{k}_\ell^\mathsf{T}\right)\mathbf{B}^{-1} + [\mathbf{x}_b^\mathsf{T}]_1 = \left[s\sum_{\ell\in[L]}(a_\ell|\mathbf{k}_\ell^\mathsf{T})\mathbf{B}^{-1} + \mathbf{x}_b^\mathsf{T}\right]_1 = \left[s\sum_{\ell\in[L]}\mathbf{w}_\ell^\mathsf{T} + \mathbf{x}_b^\mathsf{T}\right]_1$$

which is exactly the ciphertext component encrypting $\mathbf{x}_b$ as in the real scheme, or $\mathcal{H}_{b,0}$. Else if $[u]_1$ is uniform, then write $\mathbf{B}^{-1} \eqqcolon \begin{pmatrix}\overline{\mathbf{x}}^\mathsf{T}\\ \overline{\mathbf{Z}}\end{pmatrix}$, and we have

$$[\mathbf{c}_2^\mathsf{T}]_1 = \left([u]_1 \middle| [s]_1 \sum_{\ell\in[L]} \mathbf{k}_\ell^\mathsf{T}\right)\mathbf{B}^{-1} + [\mathbf{x}_b^\mathsf{T}]_1 = \left[u\overline{\mathbf{x}}^\mathsf{T} + s\sum_{\ell\in[L]}\mathbf{k}_\ell^\mathsf{T}\overline{\mathbf{Z}} + \mathbf{x}_0^\mathsf{T} + b(\mathbf{x}_1^\mathsf{T} - \mathbf{x}_0^\mathsf{T})\right]_1.$$

Now observe $\hat{\mathbf{x}}^\mathsf{T} = \hat{\mathbf{x}}^\mathsf{T}(\hat{\mathbf{x}} \mid \mathbf{Z})\begin{pmatrix}\overline{\mathbf{x}}^\mathsf{T}\\ \overline{\mathbf{Z}}\end{pmatrix} = (\|\hat{\mathbf{x}}\|^2 \mid \hat{\mathbf{x}}^\mathsf{T}\mathbf{Z})\begin{pmatrix}\overline{\mathbf{x}}^\mathsf{T}\\ \overline{\mathbf{Z}}\end{pmatrix} = \|\hat{\mathbf{x}}\|^2\overline{\mathbf{x}}^\mathsf{T} + \underbrace{\hat{\mathbf{x}}^\mathsf{T}\mathbf{Z}\overline{\mathbf{Z}}}_{=0}$, where $\|\hat{\mathbf{x}}\|$ denotes the L_2-norm of $\hat{\mathbf{x}}$. Equivalently $\overline{\mathbf{x}}^\mathsf{T} = c\hat{\mathbf{x}}^\mathsf{T} = c(\mathbf{x}_1^\mathsf{T} - \mathbf{x}_0^\mathsf{T})$ where $c \coloneqq \|\hat{\mathbf{x}}\|^{-2}$. Hence

$$[\mathbf{c}_2^\mathsf{T}]_1 = \left[uc(\mathbf{x}_1^\mathsf{T} - \mathbf{x}_0^\mathsf{T}) + \mathbf{x}_0^\mathsf{T} + b(\mathbf{x}_1^\mathsf{T} - \mathbf{x}_0^\mathsf{T}) + s\sum_{\ell\in[L]}\mathbf{k}_\ell^\mathsf{T}\overline{\mathbf{Z}}\right]_1 = \left[t\mathbf{x}_1^\mathsf{T} + (1-t)\mathbf{x}_0^\mathsf{T} + s\sum_{\ell\in[L]}\mathbf{k}_\ell^\mathsf{T}\overline{\mathbf{Z}}\right]_1,$$

where $t \coloneqq uc + b$ is uniform over $\mathbb{Z}_p$ since u is uniform and $c \neq 0$ (since w.l.o.g. $\hat{\mathbf{x}} \neq \mathbf{0}$). Therefore $[\mathbf{c}_2^\mathsf{T}]_1$ is distributed same as in $\mathcal{H}_{b,1}$.[9]

We conclude that $\mathcal{B}$ perfectly simulates $\mathcal{H}_{b,0}$ if the input $[u]_1 = \left[s\sum_{\ell\in[L]} a_\ell\right]_1$, and perfectly simulates $\mathcal{H}_{b,1}$ if $[u]_1$ is uniformly random. The proof is completed. □

[9] For any malicious user $\ell \in M$, decrypting $[\mathbf{c}_2^\mathsf{T}]_1$ in this case correctly yields $\mathbf{x}_0^\mathsf{T}\mathbf{y}_\ell = \mathbf{x}_1^\mathsf{T}\mathbf{y}_\ell$ since $(\mathbf{x}_1^\mathsf{T} - \mathbf{x}_0^\mathsf{T})\mathbf{y}_\ell = \overline{\mathbf{Z}}\mathbf{y}_\ell = 0$ and $\mathbf{c}_2^\mathsf{T}\mathbf{y}_\ell = \left(\mathbf{x}_0^\mathsf{T} + t(\mathbf{x}_1^\mathsf{T} - \mathbf{x}_0^\mathsf{T}) + s\sum_{\ell\in[L]}\mathbf{k}_\ell^\mathsf{T}\overline{\mathbf{Z}}\right)\mathbf{y}_\ell = \mathbf{x}_0^\mathsf{T}\mathbf{y}_\ell$.

5 Registered Traitor-Tracing

Traitor-tracing [CFN94] is a cryptographic primitive that allows to identify users involved in illegal distribution of content. Below, we define and construct a registered version of traitor-tracing.

Our scheme is obtained by adapting existing transformations from quadratic functional encryption to traitor-tracing to the registered setting. We first show that the RQFE scheme of Section 4.2 implies predicate encryption for comparison (PEC) following [Gay16]. The next step is just to recast PEC as a private linear broadcast encryption, a primitive first introduced in [BSW06]. This in turn yields registered traitor-tracing by adapting the transformation presented in [BSW06] to the registered setting.

5.1 Registered Private Linear Broadcast Encryption

We define and build a registered version of private linear broadcast encryption (PLBE), a primitive that was first defined in [BSW06].

Definition 5.1 (Registered Private Linear Broadcast Encryption)**.** A registered private linear broadcast encryption (RPLBE) scheme for message space $\mathcal{M}$, ciphertext space $\mathcal{C}$ and number of users L is a tuple of PPT algorithms $(\mathsf{Setup}, \mathsf{KGen}, \mathsf{Aggr}, \mathsf{Enc}, \mathsf{TrEnc}, \mathsf{Dec})$:

- $\mathsf{Setup}(1^\lambda)$ inputs the security parameter. It outputs a crs.
- $\mathsf{KGen}(\mathsf{crs}, \ell)$ inputs the crs and an index $\ell \in [L]$. It outputs a pair of public and secret keys $(\mathsf{pk}_\ell, \mathsf{sk}_\ell)$ associated with the index ℓ.
- $\mathsf{Aggr}(\mathsf{crs}, (\mathsf{pk}_\ell)_{\ell\in[L]})$ inputs crs and public keys $(\mathsf{pk}_\ell)_{\ell\in[L]}$. It outputs a master public key mpk and helper secret keys $(\mathsf{hsk}_\ell)_{\ell\in[L]}$.
- $\mathsf{Enc}(\mathsf{mpk}, m)$ inputs mpk and a message $m \in \mathcal{M}$. It outputs a ciphertext $\mathsf{ct} \in \mathcal{C}$.
- $\mathsf{TrEnc}(\mathsf{mpk}, i, m)$ inputs mpk an index $i \in [L]$, and a message $m \in \mathcal{M}$. It outputs a ciphertext $\mathsf{ct} \in \mathcal{C}$.
- $\mathsf{Dec}(\mathsf{sk}_\ell, \mathsf{hsk}_\ell, \mathsf{ct})$ inputs a secret key sk_ℓ, a helper secret key hsk_ℓ and a ciphertext ct. It outputs a message m'.

Definition 5.2 (Correctness)**.** An RPLBE is said to be correct if for all $\lambda \in \mathbb{N}$, $L \in \mathsf{poly}(\lambda)$, $m \in \mathcal{M}, \mathsf{crs} \in \mathsf{Setup}(1^\lambda)$, $(\mathsf{pk}_\ell, \mathsf{sk}_\ell) \in \mathsf{KGen}(\mathsf{crs}, \ell)$ where $\ell \in [L]$, and all $i, j \in [L]$ such that $i \le j \le L$,

$$\Pr\left[m = m' \;\middle|\; \begin{array}{l} (\mathsf{mpk}, (\mathsf{hsk}_\ell)_{\ell\in[L]}) \leftarrow \mathsf{Aggr}(\mathsf{crs}, (\mathsf{pk}_\ell)_{\ell\in[L]}) \\ \mathsf{ct} \leftarrow \mathsf{TrEnc}(\mathsf{mpk}, i, m) \\ m' \leftarrow \mathsf{Dec}(\mathsf{sk}_j, \mathsf{hsk}_j, \mathsf{ct}) \end{array} \right] = 1.$$

Definition 5.3 (Indistinguishability, Message-Hiding, Index-Hiding [BSW06])**.** An RPLBE scheme Π is said to be indistinguishable, message-hiding, and index-hiding respectively, if for all PPT $\mathcal{A}$ it holds that

$$\left| \Pr\left[\mathsf{ExpRBLPE}^{\mathtt{x},0}_{\Pi,\mathcal{A}}(1^\lambda) = 1 \right] - \Pr\left[\mathsf{ExpRBLPE}^{\mathtt{x},1}_{\Pi,\mathcal{A}}(1^\lambda) = 1 \right] \right|$$

is negligible in λ, for $\mathtt{x} \in \{\mathtt{Ind}, \mathtt{MsgHide}, \mathtt{IndexHide}\}$ respectively, where $\mathsf{Exp}^{\mathtt{x},b}_{\Pi,\mathcal{A}}$ is defined in Fig. 5.

$\mathsf{ExpRPLBE}^{\mathtt{x},b}_{\Pi,\mathcal{A}}(1^\lambda)$

$\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$
$((\mathsf{pk}_\ell)_{\ell\in[L]}, (\mathbf{r}_\ell)_{\ell\in M}, i \in [L], (m_0, m_1)) \leftarrow \mathcal{A}^{\mathsf{KGen}(\cdot),\mathsf{Corr}(\cdot)}(\mathsf{crs})$
assert $[L] \setminus M \subseteq K$
assert $\mathsf{pk}_\ell \in \mathsf{KGen}(\mathsf{crs}, \ell; \mathbf{r}_\ell) \quad \forall \ell \in M$
$(\mathsf{mpk}, (\mathsf{hsk}_\ell)_{\ell\in[L]}) \leftarrow \mathsf{Aggr}(\mathsf{crs}, (\mathsf{pk}_\ell)_{\ell\in[L]})$
if $\mathtt{x} = \mathtt{Ind}$:
  if $b = 0 : \mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{mpk}, m_0)$
  else : $\mathsf{ct}^* \leftarrow \mathsf{TrEnc}(\mathsf{mpk}, 1, m_0)$
if $\mathtt{x} = \mathtt{MsgHide} : \mathsf{ct}^* \leftarrow \mathsf{TrEnc}(\mathsf{mpk}, L+1, m_b)$
if $\mathtt{x} = \mathtt{IndexHide} : \mathsf{ct}^* \leftarrow \mathsf{TrEnc}(\mathsf{mpk}, i+b, m_0)$
$b' \leftarrow \mathcal{A}(\mathsf{ct}^*)$
return b'

$\mathsf{KGenO}(\ell)$

if $K[\ell] = \perp$
  $(\mathsf{pk}_\ell, \mathsf{sk}_\ell) \leftarrow \mathsf{KGen}(\mathsf{crs}, \ell)$
  $K[\ell] := (\mathsf{pk}_\ell, \mathsf{sk}_\ell)$
$(\mathsf{pk}_\ell, \mathsf{sk}_\ell) \leftarrow K[\ell]$
return pk_ℓ

$\mathsf{CorrO}(\ell)$

$C := C \cup \{\ell\}$
$(\mathsf{pk}_\ell, \mathsf{sk}_\ell) \leftarrow K[\ell]$
return sk_ℓ

Figure 5: Security experiments for RPLBE.

To construct RPLBE, we first recall a lemma from [Gay16] expressing the comparison predicate as a quadratic function.

Lemma 5.4 ([Gay16])**.** Let $L \in \mathsf{poly}(\lambda), \ell \in [L]$. Define the predicate $F_\ell : [L+1] \times \{1,2\} \to \{0,1,2\}$,

$$F_\ell(i,m) = \begin{cases} m, & \text{if } i \le \ell \\ 0, & \text{else} \end{cases} .$$

Then $F_\ell(i,m) = \mathbf{x}_{i,m}^\intercal \mathbf{M}_\ell \mathbf{y}_{i,m}$ for some $\mathbf{M}_\ell \in \{0,1\}^{2\sqrt{L}\times(\sqrt{L}+1)}$ and

$$\mathbf{x}_{i,m} \in \{0,1,2\}^{2\sqrt{L}} \quad \text{and} \quad \mathbf{y}_{i,m} \in \{0,1,2\}^{\sqrt{L}+1}.$$

Moreover, $\mathbf{M}_\ell$ is efficiently computable given ℓ, and $\mathbf{x}_{i,m}, \mathbf{y}_{i,m}$ are efficiently computable given (i,m). The latter is denoted by $(\mathbf{x}_{i,m}, \mathbf{y}_{i,m}) \leftarrow \mathcal{Z}(i,m)$.

We sketch the proof and refer to [Gay16] for a detailed analysis.

Proof Sketch. Let us assume that $L \in \mathbb{N}$ is a perfect square for convenience and let $\mathcal{Z}$ output $(0^{2\sqrt{L}}, 0^{\sqrt{L}+1})$ if the input is $(L+1, m)$ for any m. Clearly this yields $F_\ell(L+1, m) = 0$ as wanted. In the rest we consider $i \in [L]$.

Fix any $i \in [L]$. Let $(i_1, i_2) \in [\sqrt{L}] \times [\sqrt{L}]$ be such that $i = (i_1 - 1)\sqrt{L} + i_2$ and define (ℓ_1, ℓ_2) analogously for ℓ. Let

$$\widetilde{\mathbf{v}} = (\mathbf{0}^{i_1}, \mathbf{1}^{\sqrt{L}-i_1}) \in \{0,1\}^{\sqrt{L}} \quad \text{and} \quad \widehat{\mathbf{v}} = \mathbf{e}_{i_1} \in \{0,1\}^{\sqrt{L}}$$

where $\mathbf{e}_{i_1}$ is the i_1-th unit vector. Furthermore, let

$$\overline{\mathbf{v}} = (\mathbf{0}^{i_2-1}, \mathbf{1}^{\sqrt{L}-i_2+1}) \in \{0,1\}^{\sqrt{L}}.$$

$\mathsf{Setup}(1^\lambda)$	$\mathsf{KGen}(\mathsf{crs}, \ell)$	$\mathsf{Aggr}(\mathsf{crs}, (\mathsf{pk}_\ell)_{\ell\in[L]})$
$\mathsf{crs} \leftarrow \mathsf{RQFE.Setup}(1^\lambda, (\mathbf{M}_\ell)_{\ell\in[L]})$ **return** crs	$(\mathsf{pk}_\ell, \mathsf{sk}_\ell) \leftarrow \mathsf{RQFE.KGen}(\mathsf{crs}, \ell)$ **return** $(\mathsf{pk}_\ell, \mathsf{sk}_\ell)$	$(\mathsf{mpk}, \{\mathsf{hsk}_\ell\}_{\ell\in L}) \leftarrow \mathsf{RQFE.Aggr}(\mathsf{crs}, (\mathsf{pk}_\ell)_{\ell\in[L]})$ **return** $(\mathsf{mpk}, \{\mathsf{hsk}_\ell\}_{\ell\in L})$
$\mathsf{Enc}(\mathsf{mpk}, m \in \{1,2\})$	$\mathsf{TrEnc}(\mathsf{mpk}, i, m \in \{1,2\})$	$\mathsf{Dec}(\mathsf{sk}_\ell, \mathsf{hsk}_\ell, \mathsf{ct})$
$(\mathbf{x}_{1,m}, \mathbf{y}_{1,m}) \leftarrow \mathcal{Z}(1, m)$ $\mathsf{ct} \leftarrow \mathsf{RQFE.Enc}(\mathsf{mpk}, (\mathbf{x}_{1,m}, \mathbf{y}_{1,m}))$ **return** ct	$(\mathbf{x}_{i,m}, \mathbf{y}_{i,m}) \leftarrow \mathcal{Z}(i, m)$ $\mathsf{ct} \leftarrow \mathsf{RQFE.Enc}(\mathsf{mpk}, (\mathbf{x}_{i,m}, \mathbf{y}_{i,m}))$ **return** ct	$[m]_\mathsf{T} \leftarrow \mathsf{RQFE.Dec}(\mathsf{sk}_\ell, \mathsf{hsk}_\ell, \mathsf{ct})$ **if** $[1]_\mathsf{T} = [m]_\mathsf{T}$: **return** 1 **else** : **return** 2

Figure 6: RPLBE construction.

For any $j \in [\sqrt{L}]$, denote $\widetilde{v}_j$ the j-th entry of $\widetilde{\mathbf{v}}$ and analogously for $\widehat{v}_j, \overline{v}_j$. Now $F_\ell(i, m) = m$ if and only if $i \le \ell$, which implies either (1) $i_1 < \ell_1$, equivalently $\widetilde{v}_{\ell_1} = 1$, or (2) $i_1 = \ell_1$ and $i_2 \le \ell_2$, equivalently $\widehat{v}_{\ell_1} \cdot \overline{v}_{\ell_2} = 1$. That is, $\widetilde{v}_{\ell_1} + \widehat{v}_{\ell_1}\overline{v}_{\ell_2} = 1$. Thus, for any $m \in \{1,2\}$ and (ℓ_1, ℓ_2), we can express m as $m = m(\widetilde{v}_{\ell_1} + \widehat{v}_{\ell_1}\overline{v}_{\ell_2}) = \mathbf{x}_{i,m}^\mathsf{T}\mathbf{M}_\ell\mathbf{y}_{i,m}$, for $\mathbf{x}_{i,m}^\mathsf{T} := (m\widetilde{\mathbf{v}}^\mathsf{T}, m\widehat{\mathbf{v}}^\mathsf{T}) \in \{0,1,2\}^{2\sqrt{L}}, \mathbf{y}_{i,m}^\mathsf{T} := (1, \overline{\mathbf{v}}^\mathsf{T}) \in \{0,1,2\}^{\sqrt{L}+1}$ and $\mathbf{M}_\ell \in \{0,1\}^{2\sqrt{L}\times(\sqrt{L}+1)}$ is as follows:

$$\mathbf{M}_\ell(r, c) = \begin{cases} 1, & \text{if } (r,c) = (\ell_1, 1) \text{ or } (r,c) = (\ell_1 + \sqrt{L}, \ell_2 + 1) \\ 0, & \text{else} \end{cases} . \qquad \square$$

We show that an RPLBE can be constructed using our weak RQFE in Fig. 2. Let $L \in \mathsf{poly}(\lambda)$ and $\mathcal{M} = \{1,2\}$. For each $\ell \in [L]$, let function F_ℓ and its corresponding matrix $\mathbf{M}_\ell$ be as defined in Lemma 5.4. Also let $\mathcal{Z}$ be as defined in Lemma 5.4. Let RQFE be the weak RQFE constructed in Fig. 2, with parameters $n_1 = 2\sqrt{L}$, $n_2 = \sqrt{L} + 1$, $p > 2$, and number of users L. In Fig. 6 we describe an RPLBE for the message space $\mathcal{M}$ and L users.

Correctness of our construction follows directly from Lemma 5.4 and the correctness of RQFE. The next theorem states its security.

Theorem 5.5 (Security)**.** RPLBE (Fig. 6) is indistinguishable, message-hiding and index-hiding (Definition 5.3) if RQFE is secure.

Proof. Indistinguishability follows trivially since both algorithms $\mathsf{TrEnc}(\mathsf{mpk}, 1, m)$ and $\mathsf{Enc}(\mathsf{mpk}, m)$ are exactly the same.

Message-hiding follows from the security of RQFE: By definition of F_{L+1} from Lemma 5.4, for any messages $m_0, m_1 \in \mathcal{M}$ and $\ell \in [L]$ we have $F_\ell(L+1, m_0) = F_\ell(L+1, m_1) = 0$, hence by security of RQFE, the adversary learns nothing more than 0 in either experiment.

Index-hiding also follows from the security of RQFE: The index i or $i+1$ is encoded only in the RQFE message as $(\mathbf{x}_{i,m}, \mathbf{y}_{i,m})$ or $(\mathbf{x}_{i+1,m}, \mathbf{y}_{i+1,m})$. For any index $\ell \in C \cup M$ of which the adversary has the secret key, it holds that $i \ne \ell$, therefore either $i < i+1 \le \ell$ so that $F_\ell(i, m) = F_\ell(i+1, m) = m$, or $i+1 > i > \ell$ so that $F_\ell(i) = F_\ell(i+1) = 0$. Thus by security of RQFE, a ciphertext encrypting $(\mathbf{x}_{i,m}, \mathbf{y}_{i,m})$ is indistinguishable from one encrypting $(\mathbf{x}_{i+1,m}, \mathbf{y}_{i+1,m})$. $\square$

Optimizations. In practice, a *short, random* $\mathsf{seed} \in \{0,1\}^\lambda$ can be used as the crs along with a pseudorandom generator with a sufficient stretch, which gives a transparent setup for the RQFE.

Further, our RPLBE requires RQFE to compute quadratic functions associated to highly *sparse, binary* matrices. Lemma 5.4 explicitly characterises this: $\forall k \in [L]$, $\mathbf{M}_k$ contains exactly two 1s at positions $(k_1, 1)$ and $(k_1 + \sqrt{L}, k_2 + 1)$ for the natural map $k \mapsto (k_1, k_2)$ as specified above. We show how this significantly reduces the number of operations in the KGen and Dec algorithms:

The KGen algorithm (Fig. 2) for each user $\ell \in [L]$ can compute the terms

$$[\mathsf{dk}_{\ell,k}]_2 = s_{\ell,k_1}\,[t_1]_2 + s_{\ell,k_1+\sqrt{L}}\,[t_{k_2+1}]_2 + w_\ell\,[\gamma_k]_2$$

for matrix $\mathbf{M}_k$ with randomness

$$(\mathbf{s}_\ell, w_\ell) \in \mathbb{Z}_p^{2\sqrt{L}+1},$$

where $s_{\ell,i}, [t_j]_2$ denote the i-th and j-th elements in $\mathbf{s}_\ell$ and $[\mathbf{t}]_2$ respectively. This reduces computing *each* cross-term to only a constant number of operations (precisely, 3 exponentiations and 2 group operations in $\mathbb{G}_2$). Similarly, the slot k decryptor (Fig. 2) can avoid computing the full pairing-product in the term $[D_2]_\mathsf{T}$. Instead, it can simply compute it as

$$\left(\left[\mathbf{C}_{3,k_1}^\intercal\right]_1 [\mathbf{C}_{4,1}]_2\right) + \left(\left[\mathbf{C}_{3,k_1+\sqrt{L}}^\intercal\right]_1 [\mathbf{C}_{4,k_2+1}]_2\right).$$

So the decryptor also need not parse the full ciphertext (that grows with $\sqrt{L}$). Rather, it needs to parse only 10 group elements, namely:

$$[\mathbf{C}_1]_1, [\mathbf{C}_2]_1, \left([\mathbf{C}_{3,k_1}]_1, \left[\mathbf{C}_{3,k_1+\sqrt{L}}\right]_1\right), \left([\mathbf{C}_{4,1}]_2, [\mathbf{C}_{4,k_2+1}]_2\right)$$

Computing $[D_2]_\mathsf{T}$ requires just 4 pairings reducing its total count to only 6 during decryption (along with 5 group operations in $\mathbb{G}_\mathsf{T}$ and 1 in $\mathbb{G}_2$). Crucially, the total number of operations is *independent* of all $\sqrt{L}$ factors and is a constant.

Further, note that an index $i \in [L]$ is encoded during encryption using *binary* vectors $(\mathbf{x}_{i,m}, \mathbf{y}_{i,m})$ (where $\mathbf{x}_{i,m}$ is also scaled with the message $m \in \{1, 2\}$). Hence, one can further optimise the number of operations in the $\mathsf{Enc}, \mathsf{TrEnc}$ algorithms based on i and its equivalently encoded vectors $\widetilde{\mathbf{v}}, \widehat{\mathbf{v}}$ and $\overline{\mathbf{v}}$ as shown in Lemma 5.4.

5.2 Registered Traitor-Tracing

We are now ready to define and build registered traitor-tracing. The definitions and construction from RPLBE largely follow the one from [BSW06], except that we now work in the registered setting. We provide the definitions, construction and proofs below.

Definition 5.6 (Registered Traitor-Tracing)**.** A registered traitor-tracing (RTT) scheme for a message space $\mathcal{M}$, ciphertext space $\mathcal{C}$ and number of users L consists of the following tuple of PPT algorithms $(\mathsf{Setup}, \mathsf{KGen}, \mathsf{Aggr}, \mathsf{Enc}, \mathsf{Trace}^\mathsf{D}, \mathsf{Dec})$:

- $\mathsf{Setup}(1^\lambda)$ inputs the security parameter. It outputs a crs.
- $\mathsf{KGen}(\mathsf{crs}, \ell)$ inputs the crs and an index $\ell \in [L]$. It outputs a pair of public and secret keys $(\mathsf{pk}_\ell, \mathsf{sk}_\ell)$ associated with the index ℓ.
- $\mathsf{Aggr}(\mathsf{crs}, (\mathsf{pk}_\ell)_{\ell\in[L]})$ inputs the crs and public keys $(\mathsf{pk}_\ell)_{\ell\in[L]}$. It outputs a master public key mpk and helper secret keys $(\mathsf{hsk}_\ell)_{\ell\in[L]}$.
- $\mathsf{Enc}(\mathsf{mpk}, m)$ inputs mpk and a message $m \in \mathcal{M}$. It outputs a ciphertext $\mathsf{ct} \in \mathcal{C}$.

- $\mathsf{Trace}^{\mathsf{D}}(\mathsf{mpk}, \epsilon)$ inputs mpk and a parameter ϵ. It has oracle access to a decoder D. It outputs an identity $i \in [L]$.

- $\mathsf{Dec}(\mathsf{sk}_\ell, \mathsf{hsk}_\ell, \mathsf{ct})$ inputs a secret key sk_ℓ, a helper secret key hsk_ℓ and a ciphertext ct. It outputs a message m'.

Definition 5.7 (Correctness)**.** An RTT is said to be correct if for all $\lambda \in \mathbb{N}$, $L \in \mathsf{poly}(\lambda)$, $m \in \mathcal{M}$, $k \in [L]$, $\mathsf{crs} \in \mathsf{Setup}(1^\lambda)$, $(\mathsf{pk}_\ell, \mathsf{sk}_\ell) \in \mathsf{KGen}(\mathsf{crs}, \ell)$ where $\ell \in [L]$, it holds that

$$\Pr\left[m = m' \;\middle|\; \begin{array}{l} (\mathsf{mpk}, (\mathsf{hsk}_\ell)_{\ell \in [L]}) \leftarrow \mathsf{Aggr}(\mathsf{crs}, (\mathsf{pk}_\ell)_{\ell \in [L]}) \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, m) \\ m' \leftarrow \mathsf{Dec}(\mathsf{sk}_k, \mathsf{hsk}_k, \mathsf{ct}) \end{array} \right] = 1.$$

Definition 5.8 (Semantic Security and Traceability)**.** An RTT is said to be semantically secure, if for any PPT $\mathcal{A}$ it holds that

$$\left| \Pr\left[\mathsf{ExpRTT\text{-}Security}^0_{\Pi,\mathcal{A}}(1^\lambda) = 1 \right] - \Pr\left[\mathsf{ExpRTT\text{-}Security}^1_{\Pi,\mathcal{A}}(1^\lambda) = 1 \right] \right| \leq \mathsf{negl}(\lambda),$$

and traceable against arbitrary collusion, if for any PPT $\mathcal{A}$

$$\Pr\left[\mathsf{ExpRTT\text{-}Traceability}_{\Pi,\mathcal{A}}(1^\lambda) = 1 \right] \leq \mathsf{negl}(\lambda),$$

where $\mathsf{ExpRTT\text{-}Security}^b_{\Pi,\mathcal{A}}$ and $\mathsf{ExpRTT\text{-}Traceability}_{\Pi,\mathcal{A}}$ are defined in Fig. 7.

We also consider a selective security with static corruption version of the $\mathsf{ExpRTT\text{-}Security}^b_{\Pi,\mathcal{A}}(1^\lambda)$ where the adversary $\mathcal{A}$ announces the messages (m_0, m_1) and the corruption set at the beginning of the experiment, i.e. before seeing crs. Similarly, a scheme is said to be traceable with static corruption if the adversary in $\mathsf{ExpRTT\text{-}Traceability}_{\Pi,\mathcal{A}}$ announces the corruption set at the beginning of the experiment.

In Fig. 8 we present an RTT scheme based on an RPLBE scheme RPLBE, which is similar to that in [BSW06] but recast in the registered setting. Its correctness follows directly from that of RPLBE.

Theorem 5.9 (Semantic security)**.** RTT (Fig. 8) is semantically secure (Definition 5.8) if RPLBE is indistinguishable, message-hiding and index-hiding.

Proof. The proof follows the same reasoning as the one from [BSW06].

Hybrid $\mathcal{H}_0$. In this hybrid, the challenger sets $b = 0$.

Hybrid $\mathcal{H}_1$. This hybrid is identical to the previous one except that we set $\mathsf{ct} \leftarrow \mathsf{RPLBE.TrEnc}(\mathsf{mpk}, 1, m_0)$. Indistinguishability of hybrids follow from the indistinguishability of RPLBE.

Hybrid $\mathcal{H}_2$. This hybrid is identical to the previous one except that we set $\mathsf{ct} \leftarrow \mathsf{RPLBE.TrEnc}(\mathsf{mpk}, L+1, m_0)$. This is done via a sequence of sub-hybrids, where we replace $\mathsf{ct} \leftarrow \mathsf{RPLBE.TrEnc}(\mathsf{mpk}, i, m_0)$ by $\mathsf{ct} \leftarrow \mathsf{RPLBE.TrEnc}(\mathsf{mpk}, i+1, m_0)$, for all $i \in [L]$. Indistinguishability of hybrids follow from the index-hiding of RPLBE.

$\mathsf{ExpRTT\text{-}Security}^b_{\Pi,\mathcal{A}}(1^\lambda)$	$\mathsf{ExpRTT\text{-}Traceability}_{\Pi,\mathcal{A}}(1^\lambda)$
$\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$	$\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$
$(m_0, m_1) \leftarrow \mathcal{A}(\mathsf{crs})$	$((\mathsf{pk}_\ell)_{\ell\in[L]}, (\mathbf{r}_\ell)_{\ell\in M}, \mathsf{D}) \leftarrow \mathcal{A}^{\mathsf{KGen}(\cdot),\mathsf{Corr}(\cdot)}(\mathsf{crs})$
$(\mathsf{pk}_\ell, \mathsf{sk}_\ell) \leftarrow \mathsf{KGen}(\mathsf{crs}, \ell)\ \forall \ell \in [L]$	$\textbf{assert } [L] \setminus M \subseteq K$
$(\mathsf{mpk}, (\mathsf{hsk})_{\ell\in[L]}) \leftarrow \mathsf{Aggr}(\mathsf{crs}, (\mathsf{pk}_\ell)_{\ell\in[L]})$	$\textbf{assert } \mathsf{pk}_\ell \in \mathsf{KGen}(\mathsf{crs}, \ell; \mathbf{r}_\ell)\quad \forall \ell \in M$
$\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{mpk}, m_b)$	$(\mathsf{mpk}, (\mathsf{hsk}_\ell)_{\ell\in[L]}) \leftarrow \mathsf{Aggr}(\mathsf{crs}, (\mathsf{pk}_\ell)_{\ell\in[L]})$
$\textbf{return } \mathcal{A}(\mathsf{ct}^*)$	$S^* \leftarrow \mathsf{Trace}^{\mathsf{D}}(\mathsf{mpk}, \epsilon)$
	$b_1 := (\Pr[m \leftarrow \mathsf{D}(\mathsf{Enc}(\mathsf{mpk}, m)) \mid m \leftarrow\!\!\$\ \mathcal{M}] > \epsilon)$
$\underline{\mathsf{KGenO}(\ell)}$	$b_2 := (S^* = \emptyset \vee S^* \not\subseteq S)$
$\textbf{if } K[\ell] = \bot$	$\textbf{return } (b_1 \wedge b_2)$
$\quad (\mathsf{pk}_\ell, \mathsf{sk}_\ell) \leftarrow \mathsf{KGen}(\mathsf{crs}, \ell)$	
$\quad K[\ell] := (\mathsf{pk}_\ell, \mathsf{sk}_\ell)$	$\underline{\mathsf{CorrO}(\ell)}$
$(\mathsf{pk}_\ell, \mathsf{sk}_\ell) \leftarrow K[\ell]$	$C := C \cup \{\ell\};\ (\mathsf{pk}_\ell, \mathsf{sk}_\ell) \leftarrow K[\ell]$
$\textbf{return } \mathsf{pk}_\ell$	$\textbf{return } \mathsf{sk}_\ell$

Figure 7: Security experiments for RTT.

$\underline{\mathsf{Trace}^{\mathsf{D}}(\mathsf{mpk}, \epsilon)}$

$W := 8\lambda(L/\epsilon)^2$
for $i \in [L+1]$:
  $\mathsf{count} := 0$
  for $w \in [W]$:
    $m \leftarrow\!\!\$\ \{0,1\}$
    $\mathsf{ct} \leftarrow \mathsf{RPLBE.TrEnc}(\mathsf{mpk}, i, m)$
    if $\mathsf{D}(\mathsf{ct}) = m$: $\mathsf{count} := \mathsf{count} + 1$
  $\hat{p}_i := \mathsf{count}/W$
$S := \{i \in [L] : \hat{p}_i - \hat{p}_{i+1} \geq \epsilon/(4L)\}$
return $i \leftarrow\!\!\$\ S$

Figure 8: $\mathsf{Trace}^{\mathsf{D}}$ algorithm of the RTT construction from RPLBE. Other algorithms (Setup, KGen, Enc, Dec, Aggr) of the RTT are identical to those of RPLBE.

Hybrid $\mathcal{H}_3$. This hybrid is identical to the previous one except that we set $\mathsf{ct} \leftarrow \mathsf{RPLBE.TrEnc}(\mathsf{mpk}, L+1, m_1)$. Indistinguishability of hybrids follow from the message-hiding of RPLBE.

Hybrid $\mathcal{H}_4$. This hybrid is identical to the previous one except that we set $\mathsf{ct} \leftarrow \mathsf{RPLBE.TrEnc}(\mathsf{mpk}, 1, m_1)$. Indistinguishability of hybrids follow from the index-hiding of RPLBE.

Hybrid $\mathcal{H}_5$. This hybrid is identical to the previous one except that we set $\mathsf{ct} \leftarrow \mathsf{RPLBE.Enc}(\mathsf{mpk}, m_1)$. Indistinguishability of hybrids follow from the indistinguishability of RPLBE. $\square$

Theorem 5.10 (Traceability)**.** RTT (Fig. 8) is traceable against arbitrary collusion (Definition 5.8) if RPLBE is indistinguishable, message-hiding and index-hiding.

Proof. The proof follows the same reasoning as the one from [BSW06]. We sketch the main ideas here and refer to [BSW06] for a more detailed analysis (it is straightforward to adapt their proof to the registered setting).

Let $p_i = \Pr[\mathsf{D}(\mathsf{RPLBE.TrEnc}(\mathsf{mpk}, i, m)) = m]$ and $p = \Pr[\mathsf{D}(\mathsf{RPLBE.Enc}(\mathsf{mpk}, m)) = m]$. Let $\epsilon > 0$ be a constant. The proof is divided into 3 different types of adversaries.

- Type 1: D is a ϵ-useful decoder for which $|p - p_1| > 1/P(\lambda)$ for some polynomial P.
- Type 2: D is a ϵ-useful decoder for which $|p - p_1| \leq \mathsf{negl}(\lambda)$ but Trace outputs an empty set.
- Type 3: D is a ϵ-useful decoder for which $|p - p_1| \leq \mathsf{negl}(\lambda)$ but Trace outputs a set which is not contained in the set of colluders.

An adversary of type 1 can be used to break indistinguishability of the underlying RPLBE. An adversary of type 2 can be used to break message-hiding of the underlying RPLBE. Finally, an adversary of type 3 can be used to break the index-hiding of the underlying RPLBE. □

Efficiency. Instantiating Fig. 8 with the RPLBE in Fig. 6 via our weak RQFE (Fig. 2), we obtain a concretely efficient RTT scheme. The concrete costs are presented in Table 2. Recall that the functions $\mathbf{F}_\ell$ used for RTT can be succinctly described from Lemma 5.4. Moreover, the crs consists of random elements which can be succinctly described by a short seed to be expanded using a random oracle.

5.3 RTT with Bounded Collusion

We also present an RTT based on an RLFE. Instantiating with our RLFE in Section 4.3, we obtain an RTT with security in the standard model, at the cost of supporting only bounded number of collusions and a ciphertext size that grows with the collusion-bound.

A t-bounded-collusion RTT has the same syntax as in Definition 5.6 except that t is fixed before Setup and Trace additionally inputs a set S of at most t suspect identities. Its semantic security and traceability are defined alike Definition 5.8, except that in the traceability experiment the adversary can corrupt at most t users.

Let $L, t \in \mathsf{poly}(\lambda)$ with $t < L$. Let RLFE be the RLFE in Fig. 4 with parameter $n = t + 1$, any prime p, and L number of users. We construct a t-collusion-bounded RTT for the message space $\mathcal{M} = \{0, 1\}$ and L users. Let $\mathbf{x}_1, \ldots, \mathbf{x}_L, \mathbf{y} \in \mathbb{Z}_p^{t+1}$ be arbitrary fixed vectors such that 1) $\mathbf{x}_\ell^\mathsf{T}\mathbf{y} = 1$ for all $\ell \in [L]$, and 2) any t choices of the $\mathbf{x}_\ell$'s are linearly independent, which are hardwired in all algorithms below. Our construction is given in Fig. 9.

Both the scheme and its security proofs are conceptually similar to that of [ABP+17], but ours in the registered setting. Correctness is easy to see: For any $\ell \in [L]$ we have $\mathbf{x}_\ell \mathbf{y} m = m$ by design of $\mathbf{x}_\ell, \mathbf{y}$. Semantic security follows from the security of RLFE.

Theorem 5.11 (Semantic security)**.** Fig. 9 is a selective semantically secure with static corruption RTT if RLFE is selective secure with static corruption.

Theorem 5.12 (Traceability against bounded collusion)**.** Fig. 9 is traceable against a t collusion and static corruption if RLFE is secure with static corruption.

$$
\begin{array}{l}
\underline{\mathsf{Aggr}(\mathsf{crs}, (\mathsf{pk}_\ell)_{\ell\in[L]})} \\
\textbf{return } (\mathsf{mpk}, (\mathsf{hsk}_\ell)_{\ell\in L}) \leftarrow \mathsf{RLFE.Aggr}(\mathsf{crs}, (\mathsf{pk}_\ell, \mathbf{x}_\ell)_{\ell\in[L]}) \\
\\
\begin{array}{ll}
\underline{\mathsf{Enc}(\mathsf{mpk}, m)} & \underline{\mathsf{Dec}(\mathsf{sk}_\ell, \mathsf{hsk}_\ell, \mathsf{ct})} \\
\mathsf{ct} \leftarrow \mathsf{RLFE.Enc}(\mathsf{mpk}, \mathbf{y}m) & [m]_\mathsf{T} \leftarrow \mathsf{RLFE.Dec}(\mathsf{sk}_\ell, \mathsf{hsk}_\ell, \mathsf{ct}) \\
\textbf{return } \mathsf{ct} & \textbf{if } \ [m]_\mathsf{T} = [0]_\mathsf{T} : \textbf{return } 0 \\
 & \textbf{else } \ : \textbf{return } 1
\end{array} \\
\underline{\mathsf{Trace}^\mathsf{D}(\mathsf{mpk}, \epsilon, S)} \\
W := 8\lambda(L/\epsilon)^2; \ \textbf{parse } S = \{\mathsf{id}_1, ..., \mathsf{id}_t\} \\
\textbf{for } i \in [0, t] : \\
\quad \mathsf{count} := 0; \ S_i := \{\mathsf{id}_{i+1}, \ldots, \mathsf{id}_t\} \\
\quad \textbf{for } w \in [W] : \\
\quad\quad m \leftarrow\!\$ \ \{0, 1\} \\
\quad\quad \mathbf{v} \leftarrow\!\$ \ \mathbb{Z}_p^{t+1} : \mathbf{x}_\ell^\mathsf{T}\mathbf{v} = 0 \ \forall \ell \in S_i \ \wedge \ \mathbf{x}_\ell^\mathsf{T}\mathbf{v} = 1 - m \ \forall \ell \in S \setminus S_i \\
\quad\quad \mathsf{ct} \leftarrow \mathsf{RLFE.Enc}(\mathsf{mpk}, \mathbf{v}) \\
\quad\quad \textbf{if } \mathsf{D}(\mathsf{ct}) = m : \mathsf{count} := \mathsf{count} + 1 \\
\quad \hat{p}_i := \mathsf{count}/W \\
\textbf{for } i \in [t] : \textbf{if } \hat{p}_i - \hat{p}_{i+1} \geq \epsilon/(4L) \ \ \textbf{then } \ \textbf{return } \mathsf{id}_i
\end{array}
$$

Figure 9: Bounded-collusion RTT from RLFE. The algorithms (Setup, KGen) are identical to those of RLFE.

The proof follows the same rationale as that of [ABP+17], we sketch its main ideas for completeness. Assume that the adversary $\mathcal{A}$ is able to produce a decoder box for which Trace outputs a identity *not* contained in the set of corrupted parties. The reduction $\mathcal{R}$ (against the selective security of the underlying RLFE) starts by querying secret keys and forwarding them to $\mathcal{A}$. Upon receiving a decoder box D from $\mathcal{A}$, $\mathcal{R}$ does the following: It behaves similarly as the Trace algorithm by finding two messages m, m' that decode successfully and finding an index $i \in S$ such that $|p_i - p_{i-1}|$. The reduction prepares two messages $\mathbf{y}_0 = \mathbf{v}_{i-1}$ and $\mathbf{y}_1 = \mathbf{v}_i$, where $\mathbf{v}_j$ such that i) $\mathbf{x}_\ell \mathbf{v}_j = m$ for $\ell \in S_j$ and ii) $\mathbf{x}_\ell \mathbf{v}_j = m'$ for $\ell \notin S_j$. Note that the Trace should not have any information about the secret key $\mathsf{sk}_{\mathbf{x}_{\mathsf{id}_i}}$, so from the perspective of the adversary and the reduction, they should not be able to distinguish encryptions of these two messages. Upon receiving an encryption ct from the challenger, the reduction runs ct on D and outputs whatever it outputs. For a detailed proof, we refer to [ABP+17].

Efficiency. Instantiating Fig. 9 with the RLFE from Fig. 4, we obtain a t bounded-collusion RTT scheme with the the concrete parameters presented in Table 2.

5.4 On Revocation Mechanisms

Traitor tracing enables tracking misbehaving users. However, once such users are caught, we may require to revoke those users' keys from the system. One simple solution for this is to re-initialize the whole system once a traitor is caught. Here, we discuss some alternative approaches to add

	crs	pk_ℓ	sk_ℓ	mpk	hsk_ℓ	ct
RTT (Section 5.2)	λ $^{(\dagger)}$	$(2\sqrt{L}+1)\mathbb{G}_1$ $(L-1)\mathbb{G}_2$	$(1)\mathbb{G}_2$	$(2\sqrt{L}+1)\mathbb{G}_1$ $(\sqrt{L}+1)\mathbb{G}_2$	$(2)\mathbb{G}_2$	$(4\sqrt{L}+2)\mathbb{G}_1$ $(2\sqrt{L}+2)\mathbb{G}_2$
t-RTT (Section 5.3)	$(L(t+1))\mathbb{G}_1$ $(L^2(t+1))\mathbb{G}_2$	$(t+1)\mathbb{G}_1$ $((L-1)(t+1))\mathbb{G}_2$	$(t+1)\mathbb{Z}_p$	$(t+2)\mathbb{G}_1$	$(t+3)\mathbb{G}_2$	$(t+3)\mathbb{G}_1$

Table 2: Comparing parameter sizes of our RTT schemes. The notation $(d)\mathbb{G}_b$ denotes d elements of group $\mathbb{G}_b$. $^{(\dagger)}$Recall that a λ bit seed as the CRS can be expanded to $(L+\sqrt{L}+1)\mathbb{G}_2$ using, e.g. a PRG or a hash.

revocation in the registered setting (possibly with mild modifications to the syntax) that are more efficient that bootstrapping the system from scratch.

Generic Solution. Recall that in both (weak) RQFE (Fig. 2) and RLFE (Fig. 4), the master public key mpk and each helper secret key hsk is computed as a product of source group elements contributed through the public keys of different users. Hence, *revoking* any subset of users amounts to removing their contributions from the respective components in the (deterministically) aggregated $(\mathsf{mpk}, \{\mathsf{hsk}_i\}_i)$. Thus, given mpk and $\mathcal{R}$ at encryption, one can do the following: First, compute a fresh mpk' by removing the net contribution of the users' public keys based on $\mathcal{R}$ (e.g. by adding $\left[-\sum_{i\in\mathcal{R}}\mathbf{s}_i\right]_1$ and $\left[-\sum_{i\in\mathcal{R}}w_i\right]_1$ to $[\mathbf{s}]_1$ and $[w]_1$ respectively in the mpk of our RQFE (Fig. 2)). Next, encrypt the message with mpk' as some ct' and publish $\mathsf{ct} = (\mathsf{ct}', \mathcal{R})$ as the final ciphertext. Assuming the decryptor (for any slot i) has access to crs, it uses $\mathcal{R}$ in ct to recompute the correct hsk'_i similarly and use it to execute decryption. This is a generic solution and works for both our RTT schemes in the unbounded and bounded collusion settings (based on RQFE and RLFE respectively).

Bounded Collusion. Our generic approach requires $\mathcal{R}$ to contain respective public keys for revoked users and the decryptor to access crs. There is another efficient way to revoke any set of users (of bounded size) in our bounded-collusion RTT (Section 5.3). This solution follows from [ABP+17] which also works in our registered setting with an additional hash function modeled as a random oracle. Thus, we only sketch it here and refer to [ABP+17] for more details. Recall that [ABP+17] ties each user $i \in [L]$ with a *random* vector $\mathbf{x}_i \in \mathbb{Z}_p^\ell$, where $\ell = t + r + 1$, t and r are upper bounds on the number of traitors and revoked users. To revoke a set of users $\mathcal{R}$, an encryptor requires knowing the set $\{\mathbf{x}_i\}_{i\in\mathcal{R}}$. With this, it deterministically computes a vector $\mathbf{v}_\mathcal{R}$ s.t. for all $i \in \mathcal{R}$, $\langle \mathbf{x}_i, \mathbf{v}_\mathcal{R}\rangle = 0$ and encrypts a message m encoded as a vector $m \cdot \mathbf{v}_\mathcal{R}$ using an underlying linear FE scheme. Observe that a valid decryptor can still decrypt successfully, whereas a revoked user cannot decrypt and learn m anymore. The underlying linear FE scheme must be secure only against bounded collusion. To keep the scheme compact, we can sample the vectors $\{\mathbf{x}_i\}$ as $\mathbf{x}_i \leftarrow \mathsf{H}(i)$, where H is a hash function modeled as a random oracle, which allows us to recompute $\{\mathbf{x}_i\}$ on-the-fly.

6 More Applications

We describe more applications of our RFE schemes for inner products.

6.1 Single-Key Registered FE for Circuits

In the following we show that a registered FE for inner products can be generically transformed into a registered FE for *any polynomial-sized circuit* although with security only against an attacker that corrupts a single key. Thus, we obtain the same result as [SS10], except in the more desirable registered settings.[10] Our transformation is inspired by [ALS16], except that we manage to use *any* RFE for inner products over $\mathbb{Z}_p$. I.e., we can use any of the schemes introduced in this work to instantiate our transformation, whereas the transformation [ALS16] require inner products over $\mathbb{Z}_2$. In fact, our main technical innovation will consist in emulating inner products of $\mathbb{Z}_2$ with a scheme supporting inner products over $\mathbb{Z}_p$. Our transformation will consist of two main steps:

- In the first step, we leverage the well-known fact that any circuit C and input $\mathbf{x}$ admits a randomized encoding [AIK06] $\tilde{C}(\mathbf{x}; R)$ for the computation of $C(\mathbf{x})$, that can be computed as a constant-degree polynomial (where all computations are done over $\mathbb{Z}_2$) as a function of the input $\mathbf{x}$ and the randomness R. Linearizing, this function can be computed by an inner-product for vectors of polynomial dimension. Since this is a standard transformation, we omit further details here and we refer the reader to [ALS16] for a more precise treatment.
- We show how to use any RFE for inner products over $\mathbb{Z}_p$ (for a prime p) to build an RFE for inner products over $\mathbb{Z}_2$. One caveat of this transformation is that the resulting scheme is secure against an adversary that corrupts a single key, even if the starting scheme is collusion-resistant.

Putting together these two observations, we obtain our result. The remainder of this section is devoted to the second transformation. First, note that trivial solutions (i.e. just performing the computation over $\mathbb{Z}_p$) do not work, since they leak information about the inputs. To see this, note that

$$0 = 0 + 0 \pmod 2 = 1 + 1 \pmod 2 \quad \text{but} \quad 0 + 0 \pmod p \neq 1 + 1 \pmod p$$

which means that these two cases can be easily distinguished. To solve this issue we will use a gaussian rounding in the exponent trick by [BBDP22], which allows us to *emulate* a $\mathbb{Z}_2$ subgroup inside a $\mathbb{Z}_p$ group in a private manner. We first recall the definition of gaussian rounding.

Definition 6.1 ([Pei10])**.** Let $\sigma > 0$. For any $x \in \mathbb{R}$, the gaussian rounding $\lceil x \rfloor_\sigma$ is a random variable supported on $\mathbb{Z}$ defined by $\lceil x \rfloor_\sigma = x + D_{\mathbb{Z}-x,\sigma}$.

In other words, $\lceil x \rfloor_\sigma$ is a discrete gaussian centered on $x \in \mathbb{R}$ but supported on $\mathbb{Z}$. We will use the following convolution lemma which provides a *simulation property* for gaussian rounding.

Lemma 6.2 ([BBDP22])**.** Let $\epsilon > 0$ be bounded by a sufficiently small constant and let $\sigma_1, \sigma_2 \geq \eta_\epsilon(\mathbb{Z})$. Then it holds for all $x, y \in \mathbb{R}$ that

$$\lceil x \rfloor_{\sigma_1} + \lceil y \rfloor_{\sigma_2} \approx_s \lceil x + y \rfloor_{\sqrt{\sigma_1^2 + \sigma_2^2}}.$$

We bootstrap an RLFE over $\mathbb{Z}_p$ into a single-key RLFE over $\mathbb{Z}_2$. Let $\mathsf{RLFE} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Aggr}, \mathsf{Enc}, \mathsf{Dec})$ be a single-key RLFE scheme over $\mathbb{Z}_p$ (as the one from Section 4.3). We will show how to modify it into a scheme $\mathsf{RLFE}_2 = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Aggr}, \mathsf{Enc}, \mathsf{Dec})$ that supports computations over $\mathbb{Z}_2$. The new scheme is identical to RLFE except for the algorithm Enc which works as follows:

[10] Unlike traditional bounded-collusion FE which deals with notions of simulation-based security, we focus only on the indistiguishability based security here. Studying simulation security for RFE is out of scope and focus for this paper.

$\mathsf{RLFE}_2.\mathsf{Enc}(\mathsf{mpk}, \mathbf{x} \in \{0,1\}^n)$:

1. Parse $\mathbf{x} = (x_1, \ldots, x_n)$. Set $\widehat{\mathbf{x}} = (\lceil x_1 \cdot p/2 \rfloor_\sigma, \ldots, \lceil x_n \cdot p/2 \rfloor_\sigma)$ where $\sigma = \mathsf{poly}(\lambda)$.
2. Output $\mathsf{RLFE}.\mathsf{Enc}(\mathsf{mpk}, \widehat{\mathbf{x}})$.

Theorem 6.3 (Correctness)**.** If RLFE is correct, then RLFE_2 is correct.

Proof. Decryption works similarly as before: the decryptor uses $\mathsf{RLFE}.\mathsf{Dec}(\mathsf{sk}_\mathbf{y}, \mathsf{ct})$ and obtains $[\widehat{\mathbf{x}} \cdot \mathbf{y}^\mathsf{T}]_\mathsf{T}$. Let $B = \mathsf{poly}(\lambda)$. If there is a $i \in]-B, B[$ such that $[i]_\mathsf{T} = [\widehat{\mathbf{x}} \cdot \mathbf{y}^\mathsf{T}]_\mathsf{T}$, then output 0. Else if $[\lceil p/2 \rceil + i]_\mathsf{T} = [\widehat{\mathbf{x}} \cdot \mathbf{y}^\mathsf{T}]_\mathsf{T}$, output 1. Else, output $\perp$. Correctness holds as long as σ is chosen such that a sample $\lceil 0 \rfloor_{\sqrt{n}\sigma}$ has norm lower than B except with negligible probability. $\square$

Theorem 6.4 (Security)**.** Assuming that there is a secure single-key RLFE scheme over $\mathbb{Z}_p$, there is a secure single-key RLFE over $\mathbb{Z}_2$.

Proof. The proof follows the sequence of hybrids:

Hybrid $\mathcal{H}_0$. This is the game where $b = 0$ and $\mathsf{ct} \leftarrow \mathsf{RLFE}.\mathsf{Enc}(\mathsf{mpk}, \mathbf{x}_0)$.

Hybrid $\mathcal{H}_1$. Let $\mathbf{y} \in \{0,1\}$ be the vector such that $\mathsf{sk}_\mathbf{y}$ is held by the adversary. Let $\mathsf{wt}(\mathbf{y}) = t$ be the Hamming weight of the vector and let $y_i \neq 0$ be a coordinate which is different than 0. We set $\widetilde{\mathbf{x}}_0 = (0, \ldots, \lceil \mathbf{x}_0 \cdot \mathbf{y}^\mathsf{T} \rfloor_{\sqrt{t\sigma^2}}, \ldots, 0)$ which is 0 everywhere except in the i-th position, and compute $\mathsf{ct} \leftarrow \mathsf{RLFE}.\mathsf{Enc}(\mathsf{mpk}, \widetilde{\mathbf{x}}_0)$. We have that

$$\sum y_i \cdot \lceil x_{0,i} \cdot p/2 \rfloor_\sigma \approx_s \lceil \mathbf{x}_0 \cdot \mathbf{y}^\mathsf{T} \rfloor_{\sqrt{t\sigma^2}}$$

by Lemma 6.2. We can additionally invoke the security of the RLFE to establish indistinguishability of hybrids.

Hybrid $\mathcal{H}_2$. This hybrid is identical to the previous one except that $\mathsf{ct} \leftarrow \mathsf{RLFE}.\mathsf{Enc}(\mathsf{mpk}, \widetilde{\mathbf{x}}_1)$ where $\widetilde{\mathbf{x}}_1$ is defined similarly as before. Indistinguishability of hybrids follow from the security of the RLFE.

Hybrid $\mathcal{H}_3$. In this hybrid we replace $\mathsf{ct} \leftarrow \mathsf{RLFE}.\mathsf{Enc}(\mathsf{mpk}, \mathbf{x}_1)$. Indistinguishability from $\mathcal{H}_2$ follows from Lemma 6.2 and the security of RLFE, in a similar way as before. $\square$

6.2 Registered Threshold Encryption

Definition 6.5 (Registered Threshold Encryption)**.** A registered threshold encryption (RTE) scheme for for a message space $\mathcal{M}$, ciphertext space $\mathcal{C}$, number of users L and threshold $t \leq L$ consists of the tuple of PPT algorithms $(\mathsf{Setup}, \mathsf{KGen}, \mathsf{Aggr}, \mathsf{Enc}, \mathsf{PartDec}, \mathsf{Dec})$:

- $\mathsf{Setup}(1^\lambda)$ inputs the security parameter. It outputs a crs.
- $\mathsf{KGen}(\mathsf{crs}, \ell)$ inputs a crs and an index $\ell \in [L]$. It outputs a pair of public and secret keys $(\mathsf{pk}_\ell, \mathsf{sk}_\ell)$ associated with the index ℓ.
- $\mathsf{Aggr}(\mathsf{crs}, (\mathsf{pk}_\ell)_{\ell \in [L]})$ inputs a crs and public keys $(\mathsf{pk}_\ell)_{\ell \in [L]}$. It outputs a master public key mpk and helper decryption keys $(\mathsf{hsk}_\ell)_{\ell \in [L]}$.

- $\mathsf{Enc}(\mathsf{mpk}, m)$ inputs mpk and a message $m \in \mathcal{M}$. It outputs a ciphertext $\mathsf{ct} \in \mathcal{C}$.
- $\mathsf{PartDec}(\mathsf{sk}_\ell, \mathsf{hsk}_\ell, \mathsf{ct})$ inputs a secret key sk_ℓ, a helper decryption key hsk_ℓ and a ciphertext ct. It outputs a share share_ℓ.
- $\mathsf{Dec}((\mathsf{share}_\ell)_{\ell \in T})$ inputs a set of shares $(\mathsf{share}_\ell)_{\ell \in T}$. It outputs a message m'.

Definition 6.6 (Correctness)**.** An RTE is said to be correct if for all $\lambda \in \mathbb{N}$, $L \in \mathsf{poly}(\lambda)$, $m \in \mathcal{M}$, $t, k \in [L]$, $\mathsf{crs} \in \mathsf{Setup}(1^\lambda)$, $(\mathsf{pk}_\ell, \mathsf{sk}_\ell) \in \mathsf{KGen}(\mathsf{crs}, \ell)$ where $\ell \in [L]$, and $T \subseteq [L]$ with $|T| \geq t$, it holds that

$$\Pr\left[m = m' \;\middle|\; \begin{array}{l} (\mathsf{mpk}, (\mathsf{hsk}_\ell)_{\ell \in [L]}) \leftarrow \mathsf{Aggr}(\mathsf{crs}, (\mathsf{pk}_\ell)_{\ell \in [L]}) \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, m) \\ \mathsf{share}_\ell \leftarrow \mathsf{PartDec}(\mathsf{sk}_\ell, \mathsf{hsk}_\ell, \mathsf{ct}) \ \forall \ell \in T \\ m' \leftarrow \mathsf{Dec}((\mathsf{share}_\ell)_{i \in T}) \end{array} \right] = 1.$$

We define both semantic and simulation security of an RTE. The latter is defined in a similar fashion as in [BGG+18] which, informally, states that partial decryptions by honest users leak nothing about secret keys of honest users. This is captured by the existence of a simulator that can simulate partial decryptions without using honest parties secret keys.

Definition 6.7 (Semantic and Simulation Security)**.** An RTE scheme Π is said to be semantically secure, if for all PPT $\mathcal{A}$ it holds that

$$\left| \Pr\left[\mathsf{ExpRTE}^0_{\Pi,\mathcal{A}}(1^\lambda) = 1 \right] - \Pr\left[\mathsf{ExpRTE}^1_{\Pi,\mathcal{A}}(1^\lambda) = 1 \right] \right| \leq \mathsf{negl}(\lambda),$$

and it is said to be simulation-secure, if there exists a PPT, stateless $\mathsf{PartDecSim}$ such that for all PPT $\mathcal{A}$ and $k \in \mathsf{poly}(\lambda)$ it holds that

$$\left| \Pr\left[\mathsf{ExpRTE\text{-}Sim}^0_{\Pi,\mathcal{A}}(1^\lambda) = 1 \right] - \Pr\left[\mathsf{ExpRTE\text{-}Sim}^1_{\Pi,\mathcal{A}}(1^\lambda) = 1 \right] \right|$$

is at most $\mathsf{negl}(\lambda)$, where $\mathsf{ExpRTE}^b_{\Pi,\mathcal{A}}$ and $\mathsf{ExpRTE\text{-}Sim}^b_{\Pi,\mathcal{A}}$ are defined in Fig. 10.

Our RTE construction makes use of Shamir's secret-sharing, which we recall: to "t-out-of-L" share a secret m, sample $t-1$ random field elements, i.e. some random $\mathbf{p} \in \mathbb{Z}_p^{t-1}$ for some prime p. Let each part $\ell \in [L]$ be assigned a label ℓ, the share for party ℓ is $\mathsf{share}_\ell \coloneqq (m, \mathbf{p}^\mathsf{T})\mathbf{y}_\ell$ where $\mathbf{y}_\ell^\mathsf{T} = (1, \ell, \ldots, \ell^{t-1})$. To reconstruct the secret m given any set T of shares where $|T| = t$, say from users $\ell_1, \ldots, \ell_t$, compute

$$g\left((\mathsf{share}_\ell)_{\ell \in T}\right) = (\mathsf{share}_{\ell_1}, \ldots, \mathsf{share}_{\ell_t}) \left(\mathbf{y}_{\ell_1} \ldots \mathbf{y}_{\ell_t}\right)^{-1} (1, 0, \ldots, 0)^\mathsf{T} \tag{5}$$

which yields $(m, \mathbf{p}^\mathsf{T})(1, 0, \ldots, 0)^\mathsf{T} = m$. Security of secret-sharing says that m is information-theoretically hidden given any set of less than t shares.

Equipped with this, we construct below an RTE scheme for the message space $\mathbb{Z}_p$, for $L \in \mathsf{poly}(\lambda)$ number of users and a threshold $t \leq L$. Fix $\mathbf{y}_\ell = (1, \ell, \ldots, \ell^{t-1})$ for all $\ell \in [L]$. Let function g be as defined in Eq. (5). Let RLFE be a linear RFE for the function class $\mathcal{F}$ containing $f_\ell : \mathbb{Z}_p^t \times \mathbb{Z}_p^t \to \mathbb{Z}_p : f_\ell(\mathbf{x}) = \mathbf{x}^\mathsf{T}\mathbf{y}_\ell \bmod p$ (or else outputting some representation of $\mathbf{x}^\mathsf{T}\mathbf{y}_\ell \bmod p$, e.g. as a group element) for all $\mathbf{x} \in \mathbb{Z}_p^t$ and $\ell \in [L]$. For example, both Fig. 4 and Fig. 2 can be used to instantiate our construction, and with the latter giving a transparent setup.

$\mathsf{ExpRTE}^b_{\Pi,\mathcal{A}}(1^\lambda)$

$\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$
$((\mathsf{pk}_\ell)_{\ell\in[L]}, (\mathbf{r}_\ell)_{\ell\in M}, (m_0, m_1)) \leftarrow \mathcal{A}^{\mathsf{KGen}(\cdot),\mathsf{Corr}(\cdot)}(\mathsf{crs})$
assert $[L] \setminus M \subseteq K$
assert $\mathsf{pk}_\ell \in \mathsf{KGen}(\mathsf{crs}, \ell; \mathbf{r}_\ell) \quad \forall \ell \in M$
assert $|C \cup M| < t$
$(\mathsf{mpk}, (\mathsf{hsk}_\ell)_{\ell\in[L]}) \leftarrow \mathsf{Aggr}(\mathsf{crs}, (\mathsf{pk}_\ell)_{\ell\in[L]})$
$\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{mpk}, m_b)$
return $\mathcal{A}(\mathsf{ct}^*)$

$\mathsf{ExpRTE\text{-}Sim}^b_{\Pi,\mathcal{A}}(1^\lambda)$

$\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$
$((\mathsf{pk}_\ell)_{\ell\in[L]}, (\mathbf{r}_\ell)_{\ell\in M}, (m_i)_{i\in[k]}) \leftarrow \mathcal{A}^{\mathsf{KGen}(\cdot),\mathsf{Corr}(\cdot)}(\mathsf{crs})$
assert $[L] \setminus M \subseteq K$
assert $\mathsf{pk}_\ell \in \mathsf{KGen}(\mathsf{crs}, \ell; \mathbf{r}_\ell) \quad \forall \ell \in M$
$(\mathsf{mpk}, (\mathsf{hsk}_\ell)_{\ell\in[L]}) \leftarrow \mathsf{Aggr}(\mathsf{crs}, (\mathsf{pk}_\ell)_{\ell\in[L]})$
$\mathsf{ct}^*_i \leftarrow \mathsf{Enc}(\mathsf{mpk}, m_i) \quad \forall i \in [k]$
$D[i] := (m_i, \mathsf{ct}^*_i) \quad \forall i \in [k]$
if $b = 0 : b' \leftarrow \mathcal{A}^{\mathsf{PartDec}((\mathbf{r}_\ell)_{\ell\in M},\cdot,\cdot)}((\mathsf{ct}^*_i)_{i\in[k]})$
if $b = 1 : b' \leftarrow \mathcal{A}^{\mathsf{PartDecSim}((\mathsf{pk}_\ell)_{\ell\in[L]},(\mathsf{sk}_\ell)_{\ell\in C},(\mathbf{r}_\ell)_{\ell\in M},D,\cdot,\cdot)}((\mathsf{ct}^*_i)_{i\in[k]})$
return b'

$\mathsf{KGenO}(\ell)$

if $K[\ell] = \bot$
  $(\mathsf{pk}_\ell, \mathsf{sk}_\ell) \leftarrow \mathsf{KGen}(\mathsf{crs}, \ell)$
  $K[\ell] := (\mathsf{pk}_\ell, \mathsf{sk}_\ell)$
$(\mathsf{pk}_\ell, \mathsf{sk}_\ell) \leftarrow K[\ell]$
return pk_ℓ

$\mathsf{CorrO}(\ell)$

$C := C \cup \{\ell\}$
$(\mathsf{pk}_\ell, \mathsf{sk}_\ell) \leftarrow K[\ell]$
return sk_ℓ

$\mathsf{PartDecO}((\mathbf{r}_\ell)_{\ell\in M}, \ell, \mathsf{ct})$

if $\mathsf{ct} \notin D$: **return** $\bot$
if $\ell \in M$:
  $(\mathsf{pk}_\ell, \mathsf{sk}_\ell) \leftarrow \mathsf{KGen}(\mathsf{crs}, \ell; \mathbf{r}_\ell)$
else : $(\mathsf{pk}_\ell, \mathsf{sk}_\ell) \leftarrow K[\ell]$
$\mathsf{share}_\ell \leftarrow \mathsf{PartDec}(\mathsf{sk}_\ell, \mathsf{ct})$
return share_ℓ

Figure 10: Security experiment for RTE.

$\mathsf{Setup}(1^\lambda)$

$\mathsf{crs} \leftarrow \mathsf{RLFE.Setup}(1^\lambda)$
return crs

$\mathsf{KGen}(\mathsf{crs}, \ell)$

$(\mathsf{pk}_\ell, \mathsf{sk}_\ell) \leftarrow \mathsf{RLFE.KGen}(\mathsf{crs}, \ell)$
return $(\mathsf{pk}_\ell, \mathsf{sk}_\ell)$

$\mathsf{Aggr}(\mathsf{crs}, (\mathsf{pk}_\ell)_{\ell\in[L]})$

$(\mathsf{mpk}, (\mathsf{hsk}_\ell)_{\ell\in L}) \leftarrow \mathsf{RLFE.Aggr}(\mathsf{crs}, (\mathsf{pk}_\ell, \mathbf{y}_\ell)_{\ell\in[L]})$
return $(\mathsf{mpk}, (\mathsf{hsk}_\ell)_{\ell\in L})$

$\mathsf{PartDec}(\mathsf{sk}_\ell, \mathsf{hsk}_\ell, \mathsf{ct})$

return $\mathsf{share}_\ell \leftarrow \mathsf{RLFE.Dec}(\mathsf{sk}_\ell, \mathsf{hsk}_\ell, \mathsf{ct})$

$\mathsf{Enc}(\mathsf{mpk}, m)$

$\mathbf{p} \leftarrow_\$ \mathbb{Z}_p^{t-1}$
$\mathsf{ct} \leftarrow \mathsf{RLFE.Enc}(\mathsf{mpk}, (m, \mathbf{p}^\intercal))$
return ct

$\mathsf{Dec}((\mathsf{share}_\ell)_{\ell\in T})$

return $g\left((\mathsf{share}_\ell)_{\ell\in T}\right)$

Figure 11: RTE construction.

Remark 6.8. In case RLFE is weak, we modify the construction such that $(\mathbf{y}_\ell)_{\ell\in[L]})$ is input to $\mathsf{RLFE.Setup}$ instead of to $\mathsf{RLFE.Aggr}$.

Remark 6.9. In the group setting where RLFE may output some group elements, e.g. in the target group for both of our RFE constructions, the Dec algorithm is modified to evaluate the function g in the corresponding group as well. For example, given $\left[(\mathsf{share}_\ell)_{\ell\in T}\right]_\mathsf{T}$, it outputs $\left[g\left((\mathsf{share}_\ell)_{\ell\in T}\right)\right]_\mathsf{T} =$

$[m]_{\mathsf{T}}$.

Theorem 6.10 (Correctness)**.** Fig. 11 is correct if RLFE is correct.

Proof. By correctness of RLFE we have $\mathsf{share}_\ell = (m, \mathbf{p}^\intercal) \cdot \mathbf{y}_\ell$. The rest follows from linear algebra. ☐

Theorem 6.11 (Semantic security)**.** Fig. 11 is semantically secure if RLFE is secure.

Proof. The proof follows from the following hybrids.

Hybrid $\mathcal{H}_{b,0}$. This is the real security game.

Hybrid $\mathcal{H}_{b,1}$. In this hybrid, we change how ct is generated: first, for each $\ell \in C \cup M$, sample a simulated ℓ-th share $s_\ell \leftarrow\$ \mathbb{Z}_p$. Then, sample $\mathbf{p} \leftarrow\$ \mathbb{Z}_p^{t-1}$ subject to the constraint $(m_0, \mathbf{p}_0^\intercal) \cdot \mathbf{y}_\ell = s_\ell$. Finally, compute $\mathsf{ct} \leftarrow \mathsf{RLFE.Enc}(\mathsf{mpk}, (m_b, \mathbf{p}))$.

Since $|C \cup M| < t$, the distributions $\mathcal{H}_{b,0}$ and $\mathcal{H}_{b,1}$ are identical (from the security of Shamir's secret-sharing). Then, the indistinguishability of $\mathcal{H}_{0,1}$ and $\mathcal{H}_{1,1}$ follows from the security of RLFE. ☐

Theorem 6.12 (Simulation security)**.** Fig. 11 is simulation secure.

Proof. Let $\mathcal{L}$ be a list which is initially empty. Consider the following simulator:

$\mathsf{PartDecSim}((\mathbf{r}_\ell)_{\ell \in M}, \ell, \mathsf{ct})$**:**

- If $\mathsf{ct} \notin D$ then abort. Else look up $(m_i, \mathsf{ct}_i^*) = D[i]$ such that $\mathsf{ct} = \mathsf{ct}_i^*$.
- If $\mathcal{L}[i] = \bot$:
  - If $\ell \in M$, use provided randomness to generate keys, i.e. run $(\mathsf{pk}_\ell, \mathsf{sk}_\ell) \leftarrow \mathsf{KGen}(\mathsf{crs}, \ell; \mathbf{r}_\ell)$. If $\ell \in C \setminus M$, look up $(\mathsf{pk}_\ell, \mathsf{sk}_\ell) \leftarrow K[\ell]$.
  - For all $\ell \in C \cup M$, compute $\mathsf{share}_{i,\ell} \leftarrow \mathsf{PartDec}(\mathsf{sk}_\ell, \mathsf{ct})$.
  - Sample random $\mathbf{p}_i \in \mathbb{Z}_p^{t-1}$ subject to $(m_i, \mathbf{p}_i^\intercal) \cdot \mathbf{y}_\ell = \mathsf{share}_{i,\ell}$ for all $\ell \in C \cup M$. Note that if $|C \cup M| \geq t$ then $\mathbf{p}_i$ is uniquely determined.
  - Write $\mathcal{L}[i] \coloneqq (m_i, \mathbf{p}_i^\intercal)$.
- Output $(m_i, \mathbf{p}_i^\intercal) \cdot \mathbf{y}_\ell$.

Clearly the above simulator does not require sk_ℓ for users $\ell \in K \setminus C$ who are honest. By the correctness of RLFE, for all $\ell \in K$ and $\mathsf{share}_{i,\ell} \leftarrow \mathsf{PartDec}(\mathsf{sk}_\ell, \mathsf{ct}_j)$, it holds that $\mathsf{share}_{i,\ell} = (m_i, \mathbf{p}_i^\intercal)$ for some random $\mathbf{p}_i$ subject to the above evaluation constraint. Hence the output of PartDecSim constructed above has the same distribution as PartDecO. ☐

Efficiency. Instantiating Fig. 11 with our weak RQFE (Fig. 2) (serving as an RLFE), we obtain an RTE scheme with the following efficiency:

$$|\mathsf{crs}|\ ,\ |\mathsf{pk}_\ell| = L \cdot \mathsf{poly}(\lambda), \quad |\mathsf{mpk}|\ ,\ |\mathsf{sk}_\ell|\ ,\ |\mathsf{hsk}_\ell|\ \ |\mathsf{share}_\ell| = \mathsf{poly}(\lambda), \quad |\mathsf{ct}| = t \cdot \mathsf{poly}(\lambda).$$

Remark 6.13. The scheme presented above can be upgraded to allow for an *adaptive* (or *dynamic*) choice of the threshold at encryption time (instead of restricting it at the setup), as long as the chosen threshold $T < t$. This can be done by choosing a random $\widetilde{\mathbf{p}} \in \mathbb{Z}_p^{T-1}$ and letting the trailing be zeros, i.e. use $\mathbf{p}^\intercal = (\widetilde{\mathbf{p}}^\intercal, \mathbf{0}^\intercal) \in \mathbb{Z}_p^{t-1}$ during encryption. We formalise this below.

6.2.1 Choosing Threshold Dynamically at Encryption

In Fig. 11, the threshold t is fixed as a parameter of the scheme, and Remark 6.13 suggests a way for adaptive threshold choice at encryption time as long as the chosen threshold is $T \leq t$. A trivial way to extend this to allow for arbitrary threshold $T \leq L$ would be to instantiate Fig. 11 with $t = L$, which would however blow up the ciphertext size to $O(L)$, resulting in a trivial efficiency. Below, we sketch an alternative solution for an adaptive choice of threshold during encryption for arbitrary $T \leq L$ which still achieves short ciphertext, i.e. $|\mathsf{ct}| = T \cdot \mathsf{poly}(\lambda)$.

Let RTE_t be an RTE with fixed threshold $t \in [L]$. For simplicity, we assume that $L = 2^k$ for some $k \in \mathbb{N}$ (the most general case follows by assuming $k = \lceil \log L \rceil$). At a high level, parties run k independent executions of the protocol setting $t = 2^i$ for each $i \in [k]$. Each of the components grow only by a factor of $k = \mathcal{O}(\log L)$. If an encryptor wants to encrypt a message with respect to a threshold T, it encrypts the message with respect to the instance i such that $2^{i-1} < T \leq 2^i$. Note that the ciphertext only grows by a factor of at most 2 comparing to Fig. 11. In more detail, the scheme works as follows:

- The new $\mathsf{crs} = \{\mathsf{crs}_i\}_{i \in [k]}$ where $\mathsf{crs}_i \leftarrow \mathsf{RTE}_{2^i}.\mathsf{Setup}(1^\lambda)$.
- Each party computes $\mathsf{pk}_\ell = \{\mathsf{pk}_\ell^{(i)}\}_{i \in [k]}$ and $\mathsf{sk}_\ell = \{\mathsf{sk}_\ell^{(i)}\}_{i \in [k]}$ where $(\mathsf{pk}_\ell^{(i)}, \mathsf{sk}_\ell^{(i)}) \leftarrow \mathsf{RTE}_{2^i}.\mathsf{KGen}(\mathsf{crs}_i, \ell)$.
- The new master public key is $\mathsf{mpk} = \{\mathsf{mpk}_i\}_{i \in [k]}$ and the helper secret keys $\mathsf{hsk}_\ell = \{\mathsf{hsk}_\ell^{(i)}\}_{i \in [k]}$ where $(\mathsf{mpk}_i, \mathsf{hsk}_1^{(i)}, \ldots, \mathsf{hsk}_L^{(i)}) \leftarrow \mathsf{RTE}_{2^i}.\mathsf{Aggr}(\mathsf{crs}_i, \mathsf{pk}_1^{(i)}, \ldots, \mathsf{pk}_L^{(i)})$.
- Let $T \in [L]$ be the threshold chosen at encryption time. Let i be such that $2^{i-1} < T \leq 2^i$. Compute $\mathsf{ct} \leftarrow \mathsf{RTE}_{2^i}.\mathsf{Enc}(\mathsf{mpk}_i, m)$.
- Each party ℓ can compute partial decryptions using their own secret key $\mathsf{sk}_\ell^{(i)}$.

It is easy to see that the sizes of crs, mpk, pk_ℓ and sk_ℓ are a factor of $O(\log L)$ larger than in the original scheme. The ciphertext has size $2^i \cdot \mathsf{poly}(\lambda)$ where $2^i \leq 2T$ since $T > 2^{i-1}$.

6.2.2 Outlook: Broadcast-Efficient Secret Sharing

Other than being a distributed threshold encryption, Fig. 11 additionally provides a mechanism to broadcast secret shares efficiently. We envision the following scenario:

- A dealer wants to share a secret message m to L parties using a t-out-of-L secret sharing scheme.
- The dealer is connected to the parties via a broadcast channel (e.g. a WiFi network) and knows the public key of each party.
- The dealer wants to send each share to the respective party, while minimising the overall communication complexity.

A trivial solution to this setting is to encrypt each share individually, which incurs a total communication cost of $O(L)$. A more economical solution is to use our RTE Fig. 11 with threshold t and broadcast the ciphertext $\mathsf{ct} \leftarrow \mathsf{RTE}.\mathsf{Enc}(\mathsf{mpk}, m)$, so that each party recovers its secret share of m by running partial decryption. The communication complexity of this approach is $O(t)$.

6.3 Distributed Broadcast with Transparent Setup

We now outline how a registered threshold encryption generically implies a distributed broadcast encryption [FWW23, KMW23]. In particular, instantiating Fig. 11 with our weak RQFE (Fig. 2), we obtain a pairing-based distributed broadcast encryption scheme with a *transparent setup*. At first glance, this may appear to be a trivial implication, by simply setting $t = 1$. However there is a slight syntax mismatch: distributed broadcast encryption allows one to encrypt with respect to any *subset* of users, whereas registered threshold encryption only allows one to encrypt with respect to the full set of users (though any t-sized subset can jointly decrypt). Fortunately, our scheme allows one to aggregate a subset of keys, without affecting the correctness of the construction. Concretely, let RTE be an RTE with $t = 1$.

$\mathsf{Setup}(1^\lambda)$: The setup algorithm simply runs $\mathsf{crs} \leftarrow \mathsf{RTE.Setup}(1^\lambda)$.

$\mathsf{KGen}(\mathsf{crs}, \ell)$: The key generation runs $(\mathsf{pk}_\ell, \mathsf{sk}_\ell) \leftarrow \mathsf{RTE.KGen}(\mathsf{crs}, \ell)$.

$\mathsf{Enc}((\mathsf{pk}_s)_{s\in S}, S, m)$: On input a set of public keys $(\mathsf{pk}_s)_{s\in S}$ for $S \subseteq [L]$ and a message m, the encryption algorithm proceeds as follows.

- Compute $(\mathsf{mpk}, \mathsf{hsk}_1, \ldots, \mathsf{hsk}_L) \leftarrow \mathsf{RTE.Aggr}(\mathsf{crs}, (\mathsf{pk}_s)_{s\in S})$.
- Return $\mathsf{ct} \leftarrow \mathsf{RTE.Enc}(\mathsf{mpk}, m)$.

$\mathsf{Dec}(\mathsf{sk}_k, S, \mathsf{ct})$: On input a secret key for user k, the set S and a ciphertext ct with $k \in S$, the decryption algorithm of the k user returns $\mathsf{RTE.PartDec}(\mathsf{sk}_\ell, \mathsf{hsk}_\ell, \mathsf{ct})$.

Correctness and security follow by a straightforward reduction to the underlying RTE which we omit here. We remark that, although the functionality of RTE does not generically support aggregation of subsets of keys (as opposed to the full set $\mathsf{pk}_1, \ldots, \mathsf{pk}_L$), our RTE Fig. 11, when instantiated with either of our RFE schemes, allows this. Therefore, one can improve the concrete efficiency of the scheme by leveraging this additional property to avoid sampling "dummy" public keys in the encryption/decryption algorithms.

7 Benchmarks

We implemented a prototype[11] of our RPLBE scheme (Section 5.1) in python. As explained in Section 5, RPLBE immediately implies a register traitor-tracing, without any modification to the algorithms. For the implementation we set L to be a perfect square, and we ran our benchmarks with different values of $L \in \{16, 64, 256, 1024\}$. We calculated the time it took to run the Setup and Aggr for each L. For the KGen and Enc and Dec we calculated the average times for each slot, over 100 repetitions of the experiment. The benchmarks were conducted on a personal computer with a AMD Ryzen 5 5600X 3.7GHz CPU and 32GB of RAM running Arch Linux with kernel 6.7.1-arch1-1. In Table 3 we report the measurements for our benchmarks plotted in Fig. 12.

Storage. The storage requirement of our RPLBE is quite modest: In the RPLBE scheme with $L = 1024$, we calculated the sizes of the expanded crs, mpk, and the ciphertext, and they were 135KB, 6.6KB and 6.7KB respectively. Furthermore, the sizes of a user's public key, secret key, and helper secret key are 102.5KB, 97B, and 194B, respectively.

[11] https://anonymous.4open.science/r/RPLBE-2F21/

Group Operations. For the choice of pairings, we used the BLS12-381 elliptic curve via the petrelic [LG22] Python wrapper around RELIC [AGM+20]: each element in $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ is represented with 49, 97, and 384 bytes, respectively. On our machine, exponentiation in $\mathbb{G}_1$ and $\mathbb{G}_2$ takes an average of 6.6 and 5.8 microseconds, respectively, and each pairing evaluation takes 0.64 milliseconds.

	Time (ms)				
L	Setup	KGen	Aggr	Enc	Dec
16	3.86	9.04	1.06	7.26	4.04
64	13.31	35.14	14.56	13.53	4.04
256	48.94	138.17	226.93	26.11	4.04
1024	189.57	553.87	3576.37	51.2428	4.04

Table 3: Runtimes of our RPLBE algorithms for different L.

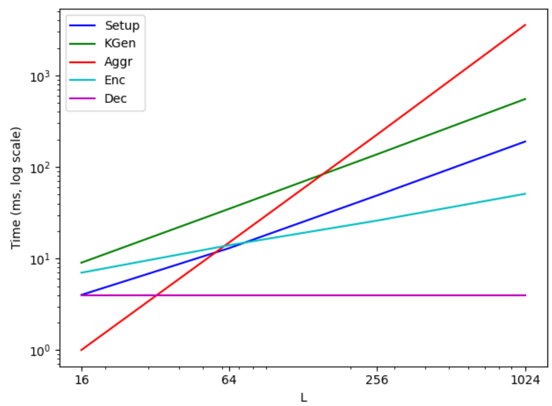

Figure 12: Runtime plots of RPLBE algorithms with a growing number of users, interpolated from the measurements taken from $L = \{16, 64, 256, 1024\}$. Both axes are in log-scale.

8 Conclusions

In this work we introduced the concept of registered traitor-tracing, a new model for traitor-tracing without a trusted authority, where each user samples their own key locally. We proposed two schemes based on bilinear maps in the bounded and unbounded collusion settings. Our benchmarks suggest that our schemes can be used in real-world applications, without adding exorbitant computational costs. An important future direction is to explore constructions of registered traitor-tracing that are post-quantum secure. Another potential direction is to study registered trace and revoke systems more formally and build schemes from different assumptions.

Acknowledgments

Pedro Branco is funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project number 537717419 and partially funded by the German Federal Ministry of Education and Research (BMBF) in the course of the 6GEM research hub under grant number

16KISK038. M.M. is supported by the European Union (ERC AdG REWORC - 101054911). G.M. is supported by the European Research Council through an ERC Starting Grant (Grant agreement No. 101077455, ObfusQation). G.M. wishes to thank Hoeteck Wee for many discussions on traitor tracing and functional encryption and for pointing to [Gay16].

References

[ABDP15] Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval. Simple functional encryption schemes for inner products. In Jonathan Katz, editor, *PKC 2015: 18th International Conference on Theory and Practice of Public Key Cryptography*, volume 9020 of *Lecture Notes in Computer Science*, pages 733–751, Gaithersburg, MD, USA, March 30 – April 1, 2015. Springer, Heidelberg, Germany. 5, 9, 10

[ABP+17] Shweta Agrawal, Sanjay Bhattacherjee, Duong Hieu Phan, Damien Stehlé, and Shota Yamada. Efficient public trace and revoke from standard assumptions: Extended abstract. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017: 24th Conference on Computer and Communications Security*, pages 2277–2293, Dallas, TX, USA, October 31 – November 2, 2017. ACM Press. 7, 37, 38, 39

[AGM+20] D. F. Aranha, C. P. L. Gouvêa, T. Markmann, R. S. Wahby, and K. Liao. RELIC is an Efficient LIbrary for Cryptography. https://github.com/relic-toolkit/relic, 2020. 47

[AIK06] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Computationally private randomizing polynomials and their applications. *Comput. Complex.*, 15(2):115–162, 2006. 40

[AKYY23] Shweta Agrawal, Simran Kumari, Anshu Yadav, and Shota Yamada. Broadcast, trace and revoke with optimal parameters from polynomial hardness. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023*, pages 605–636, Cham, 2023. Springer Nature Switzerland. 2, 4

[ALS16] Shweta Agrawal, Benoît Libert, and Damien Stehlé. Fully secure functional encryption for inner products, from standard assumptions. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part III*, volume 9816 of *Lecture Notes in Computer Science*, pages 333–362, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany. 40

[BBDP22] Zvika Brakerski, Pedro Branco, Nico Döttling, and Sihang Pu. Batch-OT with optimal rate. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology – EUROCRYPT 2022*, pages 157–186, Cham, 2022. Springer International Publishing. 40

[BCFG17] Carmen Elisabetta Zaira Baltico, Dario Catalano, Dario Fiore, and Romain Gay. Practical functional encryption for quadratic functions with applications to predicate encryption. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages

67–98, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany. 9, 14, 15

[BGG+18] Dan Boneh, Rosario Gennaro, Steven Goldfeder, Aayush Jain, Sam Kim, Peter M. R. Rasmussen, and Amit Sahai. Threshold cryptosystems from threshold fully homomorphic encryption. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018*, pages 565–596, Cham, 2018. Springer International Publishing. 42

[BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 1–18, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Heidelberg, Germany. 4

[BSW06] Dan Boneh, Amit Sahai, and Brent Waters. Fully collusion resistant traitor tracing with short ciphertexts and private keys. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 573–592, St. Petersburg, Russia, May 28 – June 1, 2006. Springer, Heidelberg, Germany. 2, 4, 6, 31, 34, 35, 37

[BW06] Dan Boneh and Brent Waters. A fully collusion resistant broadcast, trace, and revoke system. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006: 13th Conference on Computer and Communications Security*, pages 211–220, Alexandria, Virginia, USA, October 30 – November 3, 2006. ACM Press. 2, 4

[BZ14] Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 480–499, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany. 2, 4

[CES21] Kelong Cong, Karim Eldefrawy, and Nigel P. Smart. Optimizing registration based encryption. In Maura B. Paterson, editor, *Cryptography and Coding*, pages 129–157, Cham, 2021. Springer International Publishing. 4

[CFN94] Benny Chor, Amos Fiat, and Moni Naor. Tracing traitors. In Yvo Desmedt, editor, *Advances in Cryptology – CRYPTO'94*, volume 839 of *Lecture Notes in Computer Science*, pages 257–270, Santa Barbara, CA, USA, August 21–25, 1994. Springer, Heidelberg, Germany. 1, 4, 31

[DF03] Yevgeniy Dodis and Nelly Fazio. Public key trace and revoke scheme secure against adaptive chosen ciphertext attack. In Yvo Desmedt, editor, *PKC 2003: 6th International Workshop on Theory and Practice in Public Key Cryptography*, volume 2567 of *Lecture Notes in Computer Science*, pages 100–115, Miami, FL, USA, January 6–8, 2003. Springer, Heidelberg, Germany. 4

[DKL+23] Nico Döttling, Dimitris Kolonelos, Russell W. F. Lai, Chuanwei Lin, Giulio Malavolta, and Ahmadreza Rahimi. Efficient laconic cryptography from learning with errors. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023*, pages 417–446, Cham, 2023. Springer Nature Switzerland. 2, 4

[DP23] Pratish Datta and Tapas Pal. Registration-based functional encryption. *IACR Cryptol. ePrint Arch.*, page 457, 2023. 2, 3, 4, 7

[DPY23] Pratish Datta, Tapas Pal, and Shota Yamada. Registered fe beyond predicates: (attribute-based) linear functions and more. Cryptology ePrint Archive, Paper 2023/457, 2023. https://eprint.iacr.org/2023/457. 5

[FFM+23] Danilo Francati, Daniele Friolo, Monosij Maitra, Giulio Malavolta, Ahmadreza Rahimi, and Daniele Venturi. Registered (inner-product) functional encryption. In *Advances in Cryptology - ASIACRYPT 2023 - 29th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 4-8, 2023, Proceedings*, Lecture Notes in Computer Science. Springer, 2023. 2, 3, 4, 5, 6, 7, 12

[FKdP23] Dario Fiore, Dimitris Kolonelos, and Paola de Perthuis. Cuckoo commitments: Registration-based encryption and key-value map commitments for large spaces. In *Advances in Cryptology - ASIACRYPT 2023 - 29th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 4-8, 2023, Proceedings*, Lecture Notes in Computer Science. Springer, 2023. 2, 4

[FWW23] Cody Freitag, Brent Waters, and David J. Wu. How to use (plain) witness encryption: Registered abe, flexible broadcast, and more. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023*, pages 498–531, Cham, 2023. Springer Nature Switzerland. 2, 4, 46

[Gay16] Romain Gay. Functional encryption for quadratic functions, and applications to predicate encryption. Cryptology ePrint Archive, Report 2016/1106, 2016. http://eprint.iacr.org/2016/1106. 7, 31, 32, 48

[GHM+19] Sanjam Garg, Mohammad Hajiabadi, Mohammad Mahmoody, Ahmadreza Rahimi, and Sruthi Sekar. Registration-based encryption from standard assumptions. In Dongdai Lin and Kazue Sako, editors, *PKC 2019: 22nd International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 11443 of *Lecture Notes in Computer Science*, pages 63–93, Beijing, China, April 14–17, 2019. Springer, Heidelberg, Germany. 2, 4, 6

[GHMR18] Sanjam Garg, Mohammad Hajiabadi, Mohammad Mahmoody, and Ahmadreza Rahimi. Registration-based encryption: Removing private-key generator from IBE. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018: 16th Theory of Cryptography Conference, Part I*, volume 11239 of *Lecture Notes in Computer Science*, pages 689–718, Panaji, India, November 11–14, 2018. Springer, Heidelberg, Germany. 2, 4, 6, 7

[GKMR22] Noemi Glaeser, Dimitris Kolonelos, Giulio Malavolta, and Ahmadreza Rahimi. Efficient registration-based encryption. Cryptology ePrint Archive, Paper 2022/1505, 2022. https://eprint.iacr.org/2022/1505. 2, 4, 6

[GKW18] Rishab Goyal, Venkata Koppula, and Brent Waters. Collusion resistant traitor tracing from learning with errors. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *50th Annual ACM Symposium on Theory of Computing*, pages 660–670, Los Angeles, CA, USA, June 25–29, 2018. ACM Press. 2, 4

[GKW19] Rishab Goyal, Venkata Koppula, and Brent Waters. New approaches to traitor tracing with embedded identities. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019: 17th Theory of Cryptography Conference, Part II*, volume 11892 of *Lecture Notes in Computer Science*, pages 149–179, Nuremberg, Germany, December 1–5, 2019. Springer, Heidelberg, Germany. 2, 4

[GLW23] Junqing Gong, Ji Luo, and Hoeteck Wee. Traitor tracing with $N^{1/3}$-size ciphertexts and O(1)-size keys from k-Lin. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023*, pages 637–668, Cham, 2023. Springer Nature Switzerland. 2, 4

[GV20] Rishab Goyal and Satyanarayana Vusirikala. Verifiable registration-based encryption. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2020, Part I*, Lecture Notes in Computer Science, pages 621–651, Santa Barbara, CA, USA, August 16–20, 2020. Springer, Heidelberg, Germany. 4

[HLWW23] Susan Hohenberger, George Lu, Brent Waters, and David J. Wu. Registered attribute-based encryption. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023*, pages 511–542, Cham, 2023. Springer Nature Switzerland. 2, 4, 6, 9

[JLS21] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 60–73, 2021. 4

[KHL03] Chong Hee Kim, Yong Ho Hwang, and Pil Joong Lee. An efficient public key trace and revoke scheme secure against adaptive chosen ciphertext attack. In Chi-Sung Laih, editor, *Advances in Cryptology – ASIACRYPT 2003*, volume 2894 of *Lecture Notes in Computer Science*, pages 359–373, Taipei, Taiwan, November 30 – December 4, 2003. Springer, Heidelberg, Germany. 4

[KMW23] Dimitris Kolonelos, Giulio Malavolta, and Hoeteck Wee. Distributed broadcast encryption from bilinear groups. Cryptology ePrint Archive, Paper 2023/874, 2023. https://eprint.iacr.org/2023/874. 2, 4, 6, 46

[KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 146–162, Istanbul, Turkey, April 13–17, 2008. Springer, Heidelberg, Germany. 5

[KW20] Sam Kim and David J. Wu. Collusion resistant trace-and-revoke for arbitrary identities from standard assumptions. In *Advances in Cryptology – ASIACRYPT 2020, Part II*, Lecture Notes in Computer Science, pages 66–97. Springer, Heidelberg, Germany, December 2020. 2, 4

[LG22] Wouter Lueks Laurent Girod. petrelic is a python wrapper around relic. https://github.com/spring-epfl/petrelic, 2022. 47

[Luo22] Ji Luo. Ad hoc (decentralized) broadcast, trace, and revoke. Cryptology ePrint Archive, Paper 2022/925, 2022. https://eprint.iacr.org/2022/925. 4

[NNL01] Dalit Naor, Moni Naor, and Jeffery Lotspiech. Revocation and tracing schemes for stateless receivers. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 41–62, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Heidelberg, Germany. 4

[NP01] Moni Naor and Benny Pinkas. Efficient trace and revoke schemes. In Yair Frankel, editor, *FC 2000: 4th International Conference on Financial Cryptography*, volume 1962 of *Lecture Notes in Computer Science*, pages 1–20, Anguilla, British West Indies, February 20–24, 2001. Springer, Heidelberg, Germany. 4

[NWZ16] Ryo Nishimaki, Daniel Wichs, and Mark Zhandry. Anonymous traitor tracing: How to embed arbitrary information in a key. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 388–419, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany. 2, 4

[Pei10] Chris Peikert. An efficient and parallel Gaussian sampler for lattices. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 80–97, Santa Barbara, CA, USA, August 15–19, 2010. Springer, Heidelberg, Germany. 40

[Rog15] Phillip Rogaway. The moral character of cryptographic work. Cryptology ePrint Archive, Report 2015/1162, 2015. http://eprint.iacr.org/2015/1162. 2

[SS10] Amit Sahai and Hakan Seyalioglu. Worry-free encryption: functional encryption with public keys. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *ACM CCS 2010: 17th Conference on Computer and Communications Security*, pages 463–472, Chicago, Illinois, USA, October 4–8, 2010. ACM Press. 40

[Wee20] Hoeteck Wee. Functional encryption for quadratic functions from k-lin, revisited. In *TCC 2020: 18th Theory of Cryptography Conference, Part I*, Lecture Notes in Computer Science, pages 210–228. Springer, Heidelberg, Germany, March 2020. 9

[WQZDF10] Qianhong Wu, Bo Qin, Lei Zhang, and Josep Domingo-Ferrer. Ad hoc broadcast encryption. In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, CCS '10, page 741–743, New York, NY, USA, 2010. Association for Computing Machinery. 2, 4

[Zha20] Mark Zhandry. New techniques for traitor tracing: Size $N^{1/3}$ and more from pairings. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2020, Part I*, Lecture Notes in Computer Science, pages 652–682, Santa Barbara, CA, USA, August 16–20, 2020. Springer, Heidelberg, Germany. 2, 4

[ZZGQ23] Ziqi Zhu, Kai Zhang, Junqing Gong, and Haifeng Qian. Registered abe via predicate encodings. In *Advances in Cryptology - ASIACRYPT 2023 - 29th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 4-8, 2023, Proceedings*, Lecture Notes in Computer Science. Springer, 2023. 2, 4, 9

A On Handling Malicious Public Keys

We continue our discussion in Remark 4.5 on tackling malicious public keys in our RFE schemes from Sections 4.2 and 4.3. Our schemes from Sections 4.2 and 4.3 are proven secure in a setting that requires the adversary to declare to the security experiment its randomness for maliciously chosen keys. We provide three different solutions to overcome this. The first one is generic and uses NIZKs that works for any RFE with honestly formed keys. The remaining two solutions are specific to our weak RQFE scheme from Section 4.2.

For this, we introduce the additional algorithm to the syntax of an RFE:

- $b \leftarrow \mathsf{IsValid}(\mathsf{crs}, \ell, \mathsf{pk}_\ell)$: On input crs, a public key pk_ℓ and a user index $\ell \in [L]$, the deterministic verification algorithm outputs a bit $b \in \{0, 1\}$.

First we recall that, in the malicious key setting, the additional $\mathsf{IsValid}$ algorithm acts as an argument system to verify the validity of public keys, for which it should satisfy the completeness property defined as follows.

Definition A.1 (Completeness)**.** An RFE scheme is said to be complete, if for all $\lambda, \in \mathbb{N}, \ell \in [L]$,

$$\Pr\Big[\mathsf{IsValid}(\mathsf{crs}) = 1 \,\Big|\, \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda);\ (\mathsf{pk}_\ell, \mathsf{sk}_\ell) \leftarrow \mathsf{KGen}(\mathsf{crs}, \ell)\Big] = 1.$$

Completeness for a weak RFE is defined analogously.

It is direct that all strategies that we discuss in the following satisfy completeness.

Definition A.2 (Security against Malicious-Key Generation)**.** This is same as Definition 4.4, except that $\mathcal{A}$ does not output the randomness $(\mathbf{r}_\ell)_{\ell \in M}$ to the experiment, conditioned on $1 \leftarrow \mathsf{IsValid}(\mathsf{crs}, \ell, \mathsf{pk}_\ell)$.

A.1 Generic Solution using NIZKs

The first generic approach is to have KGen output a public key pk_ℓ which has a NIZK proof attached, proving that pk_ℓ is indeed generated honestly, and the $\mathsf{IsValid}$ algorithm checks that validity of the NIZK proof. Here, we need the NIZK to be straight-line simulation extractable. To argue that the basic security (as per Definition 4.4) implies security against malicious key generation, the reduction does the following: for any malicious user $\ell \in M$, upon receiving a key-proof pair (pk_ℓ, π) for which the malicious-key-generating adversary $\mathcal{A}$ asks to register pk_ℓ, the reduction extracts randomness

$\mathbf{r}_\ell$, and passes $(\mathsf{pk}_\ell, \mathbf{r}_\ell)$ to the basic security adversary $\mathcal{B}$; for any user $\ell \notin M$, upon receiving a key pk_ℓ from $\mathcal{B}$, it simulates a proof π and passes (pk_ℓ, π) to $\mathcal{A}$ as the public key. The straight-line property ensures the reduction runs in polynomial time.

A.2 Handling Malicious Keys in ROM

We briefly explain how to use a random oracle (RO) to extend our weak RFE scheme for quadratic functions (Fig. 2) to allow for maliciously generated public keys.

Intuitively, Fig. 2 fails to handle malicious key generation because of the ability to *adaptively* construct keys *depending* on the honestly generated keys. In more detail, it can set $[\mathbf{s}_i]_1 = \left[\sum_{\ell\in\mathcal{H}} \mathbf{s}_\ell + \mathbf{s}'\right]_1$ for some slot index $i \in [L]$, where $\mathcal{H}$ is the set of honestly formed keys. The security proof fails in this case as we cannot treat $\sum_{\ell\in[L]} \mathbf{s}_\ell \in \mathbb{Z}_p^{n_1}$ as a variable unknown to the adversary.

In a nutshell, this can be prevented by computing the master public key as a *random* linear combination (output by an RO) of the users' public keys (instead of simply summing it).

Construction 1 (Weak RQFE with Malicious Keys in ROM)**.** Let $\mathsf{H} : \{0,1\}^* \to \mathbb{Z}_p^L$ be a random oracle. The scheme is identical to Fig. 2 except with a slightly modified aggregation algorithm:

$\mathsf{IsValid}(\mathsf{crs}, \ell, \mathsf{pk}_\ell)$:

1. Parse $\mathsf{crs} = \Big(\mathcal{G}, (\mathbf{F}_\ell)_{\ell\in[L]}, ([\gamma_\ell]_2)_{\ell\in[L]}, [\mathbf{t}]_2\Big)$ and $\mathsf{pk}_\ell = \Big([\mathbf{s}_\ell]_1, [w_\ell]_1, ([\mathsf{dk}_{\ell,k}]_2)_{k\in[L]\setminus\{\ell\}}\Big)$.
2. Output 1 if $[1]_1 [\mathsf{dk}_{\ell,k}]_2 = \left[\mathbf{s}_\ell^\intercal\right]_1 \mathbf{F}_k [\mathbf{t}]_2 + [w_\ell]_1 [\gamma_k]_2$ for all $k \in [L] \setminus \{\ell\}$. Else output 0.

$\mathsf{Aggr}(\mathsf{crs}, (\mathsf{pk}_\ell)_{\ell\in[L]})$:

1. Parse $\mathsf{crs} = (\mathcal{G}, (\mathbf{F}_\ell)_{\ell\in[L]}, \{[\gamma_\ell]_2\}_{\ell\in[L]}, [\mathbf{t}]_2)$, and $\mathsf{pk}_\ell = \big([\mathbf{s}_\ell]_1, [w_\ell]_1, \{[\mathsf{dk}_{\ell,k}]_2\}_{k\in[L]\setminus\{\ell\}}\big)$ for each $\ell \in [L]$.
2. Compute $(\alpha_1, \ldots, \alpha_L) \leftarrow \mathsf{H}(\mathsf{pk}_1, \ldots, \mathsf{pk}_L)$.
3. Output $\mathsf{mpk} \coloneqq (\mathcal{G}, [\mathbf{s}]_1, [w]_1, [\mathbf{t}]_2)$ and $\mathsf{hsk}_k \coloneqq \big([h_{1,k}]_2, [h_{2,k}]_2, \mathbf{F}_k\big)$ for all $k \in [L]$, where

$$[\mathbf{s}]_1 \coloneqq \left[\sum_{\ell\in[L]} \alpha_\ell \mathbf{s}_\ell\right]_1, \quad [w]_1 \coloneqq \left[\sum_{\ell\in[L]} \alpha_\ell w_\ell\right]_1, \quad [h_{1,k}]_2 \coloneqq \left[\sum_{\ell\in[L]\setminus\{k\}} \alpha_k \mathsf{dk}_{\ell,k}\right]_2, \quad [h_{2,k}]_2 \coloneqq [\gamma_k]_2 .$$

Note that the above solution makes the validity of (maliciously) sampled keys vacously true as every key is valid as a group element. So there is no specific validity test required in this case and the security experiment does not need to check for well-formedness of keys explicitly. Therefore, completeness and correctness hold trivially as per Definitions 4.2 and A.1. We now sketch the security proof.[12] Informally, we need to show that $[\mathbf{s}]_1$ and $[w]_1$ can be treated as

[12] A careful reader may have already noticed that we want to prove our scheme secure in the GGM in addition to a hash function H that is modelled as a random oracle during the proof. To tackle this practically unexciting yet technical issue, one may define a hybrid model GGROM (or SGROM), between GGM (or equivalently SGM) and ROM as follows. An adversary $\mathcal{A}$ in the GGROM gets the same interface as GGM with an additional access to a random oracle in the generic (or symbolic) bilinear group GGRO (or SGRO), which has the following syntax: on input handles to finite sequences of $\mathbb{G}_1, \mathbb{G}_2$, and $\mathbb{G}_T$ elements, and a finite bit string, the GGRO fetches the group elements defined by these handles from the generic group oracles, and query the RO on the corresponding sequence of group elements and the bit string. The GGRO returns whatever the RO outputs. In particular, $\mathcal{A}$ can only indirectly access H through GGRO, which relays queries (and their answers) to/from H.

symbolic variables, and not constants, in the view of an adversary $\mathcal{A}$, so that we can then reuse the proof ideas of Theorem 4.8.

The proof proceeds similar to that of Theorem 4.8 by first lifting the game from GGM to SGM. We then show that $\mathcal{A}$ has a negligible probability of succeeding in certains forms of Zt_T queries. One such Zt_T query, for instance, looks like $\eta \mathtt{w} + c = 0$, where η, c are coefficients chosen by $\mathcal{A}$. Note that

$$\begin{aligned}\mathtt{w} &= \sum_{i\in[L]} \alpha_i \mathtt{w}_i \\ &= \sum_{i\in H} \alpha_i \mathtt{w}_i + \sum_{k\in C} \alpha_k \mathtt{w}_k + \sum_{j\in M} \alpha_j \sum_{i\in H} \psi_{j,i}\mathtt{w}_i + c_j\end{aligned}$$

since malicious keys can depend on honestly generated keys. Since c is a constant chosen by the adversary, it is different from all w_i for $i \in H$. Thus, for the zero-test to succeed, it must hold that for all $i \in H, \alpha_i = \alpha_j \sum_{j\in M} \psi_{j,i}$. In more detail, for all $i \in H$ the coefficients of $\mathtt{w}_i$ needs to be annihilated. This happens with probability at most $1/p^L = 1/2^{\lambda L}$. If the adversary performs Q_{zt} queries to the random oracle, by a union bound, we can upper bound the probability of success by $Q_{\mathsf{zt}}/2^{\lambda L}$, which is negligible in λ.

A similar argument can be shown for another Zt_T query of the form $\eta_i \mathtt{s}_i + c_i = 0$ (where $\mathtt{s}_i$ symbolically represents the i-th coordinate of $\sum_{j\in[L]} \alpha_j \mathbf{s}_j$) for some adversarially chosen coefficients η_i, c_i, so that $\mathcal{A}$ again has a negligible probability of succeeding. Rest of the proof remains same as before.

A.3 Handling Malicious Keys *without* ROM

We now show our third and final, albeit tailor-made, solution to provide security against maliciously computed keys for our weak RFE for quadratic functions. In particular, this ad hoc solution lifts Fig. 2 at the cost of *losing transparent setup*, but relies neither on NIZK nor ROM. The new scheme has modified Setup, KGen and IsValid algorithms, while the other procedures, i.e. Aggr, Enc, Dec, remain the same as in Fig. 2 (except some syntactic changes) and are left out.

Construction 2 (Weak RQFE with Malicious Keys)**.**

$\mathsf{Setup}(1^\lambda, 1^L, (\mathbf{F}_\ell)_{\ell\in[L]})$:

1. Sample $\mathcal{G} := (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_\mathsf{T}, p, [1]_1, [1]_2, \cdot) \leftarrow \mathsf{GGen}(1^\lambda)$.
2. Sample $\rho_\ell, \gamma_\ell \leftarrow\!\!\$\ \mathbb{Z}_p$ for all $\ell \in [L]$, and $\mathbf{t} \leftarrow\!\!\$\ \mathbb{Z}_p^{n_2}, \delta \leftarrow \mathbb{Z}_p$.
3. Output $\mathsf{crs} := \Big(\mathcal{G}, (\mathbf{F}_\ell)_{\ell\in[L]}, ([\rho_\ell]_1, [\rho_\ell \mathbf{t}]_2)_{\ell\in[L]}, ([\rho_\ell\gamma_k]_2)_{\ell,k\in[L]}, [\delta]_2, ([\gamma_\ell]_2)_{\ell\in[L]}, [\mathbf{t}]_2\Big)$.

$\mathsf{KGen}(\mathsf{crs}, \ell)$:

1. Parse $\mathsf{crs} = \Big(\mathcal{G}, (\mathbf{F}_\ell)_{\ell\in[L]}, ([\rho_\ell]_1, [\rho_\ell \mathbf{t}]_2)_{\ell\in[L]}, ([\rho_\ell\gamma_k]_2)_{\ell,k\in[L]}, [\delta]_2, ([\gamma_\ell]_2)_{\ell\in[L]}, [\mathbf{t}]_2\Big)$.
2. Sample $\mathbf{s}_\ell \leftarrow\!\!\$\ \mathbb{Z}_p^{n_1}$ and $w_\ell \leftarrow\!\!\$\ \mathbb{Z}_p$.
3. For all $k \in [L]$, let $[\mathsf{dk}_{\ell,k}]_2 := \left[\rho_\ell \mathbf{s}_\ell^\intercal \mathbf{F}_k \mathbf{t} + \rho_\ell\gamma_k w_\ell\right]_2$.
4. Output $\mathsf{pk}_\ell := \Big([\rho_\ell \mathbf{s}_\ell]_1, [\rho_\ell w_\ell]_1, \big([\mathsf{dk}_{\ell,k}]_2\big)_{k\in[L]\setminus\{\ell\}}, [\delta \mathbf{s}_\ell]_2, [\delta w_\ell]_2\Big)$ and $\mathsf{sk}_\ell := [\mathsf{dk}_{\ell,\ell}]_2$.

$\mathsf{IsValid}(\mathsf{crs}, \ell, \mathsf{pk}_\ell)$:

1. Parse $\mathsf{crs} = \Big(\mathcal{G}, (\mathbf{F}_\ell)_{\ell\in[L]}, ([\rho_\ell]_1, [\rho_\ell \mathbf{t}]_2)_{\ell\in[L]}, ([\rho_\ell\gamma_k]_2)_{\ell,k\in[L]}, [\delta]_2, ([\gamma_\ell]_2)_{\ell\in[L]}, [\mathbf{t}]_2\Big)$.
2. Parse $\mathsf{pk}_\ell = \Big([\rho_\ell \mathbf{s}_\ell]_1, [\rho_\ell w_\ell]_1, \big([\mathsf{dk}_{\ell,k}]_2\big)_{k\in[L]\setminus\{\ell\}}, [\delta \mathbf{s}_\ell]_2, [\delta w_\ell]_2\Big)$.
3. Output 1, if
   (a) $[1]_1 [\mathsf{dk}_{\ell,k}]_2 = \left[\rho_\ell \mathbf{s}_\ell^{\mathtt{T}}\right]_1 \mathbf{F}_k [\mathbf{t}]_2 + [\rho_\ell w_\ell]_1 [\gamma_k]_2, \forall k \in [L] \setminus \{\ell\}$,
   (b) $[\rho_\ell \mathbf{s}_\ell]_1 [\delta]_2 = [\rho_\ell]_1 [\delta \mathbf{s}_\ell]_2$ and $[\rho_\ell w_\ell]_1 [\delta]_2 = [\rho_\ell]_1 [\delta w_\ell]_2$.

Compared to Fig. 2, Construction 2 introduces $\{\rho_\ell\}_{\ell\in[L]}, \delta \in \mathbb{Z}_p$ into the crs, where ρ_ℓ is multiplied to other terms in crs, pk_ℓ and sk_ℓ. Additionally pk_ℓ includes new elements $[\delta \mathbf{s}_\ell]_2, [\delta w_\ell]_2$. Completeness and correctness (Definitions 4.2 and A.1) are easy to see and follows immediately with mild syntactic changes.

We now sketch the security proof for the scheme presented above. As explained before in Appendix A.2, Fig. 2 falls prey to an adaptive attack, where an adversary can control the (discrete logarithm of the) master public key if it chooses keys after seeing the honestly generated ones. To prevent this, Construction 2 ensures in an ad hoc way above that if such malicious keys pass the $\mathsf{IsValid}$ test, the adversary cannot know the discrete logarithm of the master public key. We can then reuse the proof ideas from that of Fig. 2.

The proof proceeds similar to that of Theorem 4.8 by first lifting the game from GGM to SGM. Next, we show that $[\mathbf{s}]_1$ and $[w]_1$ are not constants in the view of an adversary $\mathcal{A}$. Recall M is the set of slot indices for maliciously chosen public keys. For any $j \in M$, let

$$\mathsf{pk}_j \coloneqq \left(\left[\mathbf{s}_j^*\right]_1, \left[w_j^*\right]_1, \left\{\left[\mathsf{dk}_{j,k}^*\right]_2\right\}_{k\in[L]\setminus\{\ell\}}, [\widehat{\mathbf{s}_j}]_2, [\widehat{w_j}]_2\right)$$

Given $1 \leftarrow \mathsf{IsValid}(\mathsf{crs}, j, \mathsf{pk}_j)$, using the crs and $\left[\mathbf{s}_j^*\right]_1, \left[w_j^*\right]_1$, it is easy to verify that:

$$(i)\ \left[\mathsf{dk}_{j,k}^*\right]_2 = \left[\rho_j \mathbf{s}_j^{*T} \mathbf{F}_k \mathbf{t} + \rho_j \gamma_k w_j\right]_2 \quad , \quad (ii)\ [\rho_j \widehat{\mathbf{s}_j}]_2 = \left[\delta \mathbf{s}_j^*\right]_2 \quad , \quad (iii)\ [\rho_j \widehat{w_j}]_2 = \left[\delta w_j^*\right]_2$$

In the SGM, note that the validity test implies for all $j \in M$, $\rho_j \widehat{\mathtt{s}}_{j,i} = \delta \mathtt{s}_{j,i}^*, \forall i \in [n_1]$ and $\rho_j \widehat{\mathtt{w}_j} = \delta \mathtt{w}_j^*$. Since ρ and δ are symbolically linearly independent, this implies $\mathtt{s}_{j,i}^* = \rho \mathtt{s}_{j,i}, \forall i \in [n_1]$ and $\mathtt{w}_j^* = \rho \mathtt{w}_j$.

We then show that $\mathcal{A}$ has a negligible probability of succeeding in certains forms of Zt_T queries. One such Zt_T query, for instance, looks like: $\eta \mathtt{w} - c = 0$ is negligible, where $\mathtt{w}$ symbolically represents $w = \sum_{\ell\in[L]} w_\ell$ and η, c are constants chosen by $\mathcal{A}$. We skectch our arguments as follows. Note that we can express $\mathtt{w}$ as $\mathtt{w} = \sum_{i\in\mathcal{H}} \rho_i \mathtt{w}_i + \sum_{i\in M} \rho_i w_i$. Since the symbolic variables ρ_ℓ are all pairwise linearly independent, $\mathcal{A}$ has a negligible probability in succeeding in a Zt_T query of the above form. Similarly, we can show that $\mathcal{A}$ cannot succeed in a Zt_T query of the form $\eta_i \mathtt{s}_i - c_i = 0$, where $\mathtt{s}_i$ is the i-th symbolic coordinate representing the vector $\mathbf{s} = \sum_{\ell\in[L]} \mathbf{s}_\ell$ from the master public key and η_i, c_i are constants chosen by $\mathcal{A}$. This establishes that $\mathcal{A}$ cannot maliciously fix the master public key. From here on, we adopt similar proof ideas from Theorem 4.8. In particular, the ring of multivariate polynomials, as the source of all Zt_T queries, now becomes

$$\zeta = \mathbb{Z}_p\left[\{\gamma_\ell, \rho_\ell\}_{\ell\in[L]}, \delta, (\mathtt{t}_1, \ldots, \mathtt{t}_{n_2}), \{(\mathtt{s}_{\ell,1}^c, \ldots, \mathtt{s}_{\ell,n_1}^c)\}_{\ell\in H, c\in[Q_k]}, \{\mathtt{w}_\ell^c\}_{\ell\in H, c\in[Q_k]}, \alpha, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\right],$$

where the maximal total degree of a term in any polynomial $\Phi \in \mathsf{C}(\mathcal{L}_\mathsf{T}) \subset \zeta$ is $d = 8$ due to introducing $(\rho_\ell)_{\ell\in[L]}$ into $\mathtt{s}_i = \sum_{\ell\in[L]} \rho_\ell \mathtt{s}_{\ell,i}$ and $\mathtt{w} = \sum_{\ell\in[L]} \rho_\ell \mathtt{w}_\ell$. By inspection and observing symbolic linear independence between various terms, we can reason out that it is enough to consider Zt_{T} queries of the form

$$\Omega + \sum_{i\in[n_1],j\in[n_2]} \left\{-\eta_{i,j} \cdot (\mathtt{ad} - \mathtt{bc})\alpha \mathtt{s}_i \mathtt{t}_\mathtt{j}\right\} = 0, \text{ for some } \Omega \in \zeta.$$

Further reasoning on the structural properties of the coefficients of Ω with some inspection similar to Claim 4.13 leads us to a point where we have

$$\Omega = (\mathtt{ad} - \mathtt{bc})\alpha \sum_{\substack{i\in[n_1]\\ j\in[n_2]}} \sum_{\substack{k\in C\cup M\\ \ell\in H}} \xi_{k,\ell} f_{i,j}^{(k)} \left(\rho_\ell \mathtt{s}_{\ell,i}\right) \mathtt{t}_j$$

From here on, the proof follows exactly as that of Theorem 4.8.

B Analysis of Assumption 4.17

Theorem B.1. If $1/p = \mathsf{negl}(\lambda)$, then Assumption 4.17 holds in the generic bilinear group model (GGM).

Proof. We prove the claim in the symbolic group model (SGM) recalled in the proof of Theorem 4.8, which implies a proof in the GGM. For $b \in \{0, 1\}$, consider an SGM adversary $\mathcal{A}$ which inputs (the handles of)

$$\left([\mathtt{s}]_1, \{[\mathtt{a}_\ell]_1, [\mathtt{r}_\ell]_2\}_{\ell\in[L]}, \{[\mathtt{r}_k\mathtt{a}_\ell]_2\}_{k,\ell\in[L],k\neq\ell}, [\mathtt{u}_b]_1\right)$$

where $\mathtt{u}_0 = \mathtt{s}\sum_{\ell\in[L]} \mathtt{a}_\ell$ while $\mathtt{u}_1 = \mathtt{u}$ is an independent variable. We argue that the probabilities of $\mathcal{A}$ returning 1 in both the cases $b \in \{0, 1\}$ are negligibly close. Recall from Theorem 4.8 where we defined the closure $\mathsf{C}(\mathcal{L}_\mathsf{T}) = \mathcal{L}_\mathsf{T} \cup \{V_1 \cdot V_2 \mid \forall V_1 \in \mathcal{L}_1, V_2 \in \mathcal{L}_2\}$. Note that all the handles $\mathcal{A}$ receives in this case belongs to the following ring: $\zeta = \mathbb{Z}_p\left[\mathtt{s}, \{\mathtt{a}_\ell\}_{\ell\in[L]}, \{\mathtt{r}_k\}_{k\in[L]}\right]$. Hence, $\mathsf{C}(\mathcal{L}_\mathsf{T})$ provides the list of (handles of) polynomials from ζ representing elements from $\mathbb{G}_\mathsf{T}$ that $\mathcal{A}$ received during its execution.

In the case $b = 1$, observe that (the handles for) all polynomials $\Phi \in \mathsf{C}(\mathcal{L}_\mathsf{T})$ are linear combinations of the following (possibly repeating) monomials denoted by m with subscripts:

$$m_0 = 1, \qquad m_1 = \mathtt{s}, \qquad \{m_{2,\ell} = \mathtt{a}_\ell\}_{\ell\in[L]}, \qquad m_3 = \mathtt{u},$$

$$\{m_{4,k} = \mathtt{r}_k\}_{k\in[L]}, \quad \{m_{5,k} = \mathtt{sr}_k\}_{k\in[L]}, \quad \{m_{6,k,\ell} = \mathtt{r}_k\mathtt{a}_\ell\}_{k,\ell\in[L]}, \quad \{m_{7,k} = \mathtt{ur}_k\}_{k\in[L]},$$

$$\left\{\begin{array}{l} m_{8,k,\ell} \\ = \mathtt{r}_k\mathtt{a}_\ell \end{array}\right\}_{\substack{k,\ell\in[L],\\ :k\neq\ell}} \quad \left\{\begin{array}{l} m_{9,k,\ell} \\ = \mathtt{sr}_k\mathtt{a}_\ell \end{array}\right\}_{\substack{k,\ell\in[L],\\ :k\neq\ell}} \quad \left\{\begin{array}{l} m_{10,k,\ell,\ell'} \\ = \mathtt{r}_k\mathtt{a}_\ell\mathtt{a}_{\ell'} \end{array}\right\}_{\substack{k,\ell,\ell'\in[L],\\ :k\neq\ell}} \quad \left\{\begin{array}{l} m_{11,k,\ell} \\ = \mathtt{ur}_k\mathtt{a}_\ell \end{array}\right\}_{\substack{k,\ell\in[L]\\ :k\neq\ell}}$$

Note that, except for the monomials $m_{6,k,\ell}$ and $m_{8,k,\ell}$, all other monomials are linearly independent of each other. In particular, if $c_{i,\mathtt{index}}$ denotes the coefficient of $m_{i,\mathtt{index}}$ in any zero polynomial $\Phi \equiv 0$, it holds that $c_{6,k,\ell} = -c_{8,k,\ell} \in \mathbb{Z}_p$ for all $k, \ell \in [L]$ with $k \neq \ell$, and all other coefficients are zero.

In the case $b = 0$, observe that all polynomials $\Phi \in \mathsf{C}(\mathcal{L}_\mathsf{T})$ are linear combinations of the following (possibly repeating) monomials denoted by m with subscripts:

$$m_0 = 1, \qquad m_1 = \mathtt{s}, \qquad \{m_{2,\ell} = \mathtt{a}_\ell\}_{\ell \in [L]}, \qquad m_3 = \mathtt{s} \textstyle\sum_{j \in [L]} \mathtt{a}_j,$$

$$\{m_{4,k} = \mathtt{r}_k\}_{k \in [L]}, \quad \{m_{5,k} = \mathtt{sr}_k\}_{k \in [L]}, \quad \{m_{6,k,\ell} = \mathtt{r}_k \mathtt{a}_\ell\}_{k,\ell \in [L]}, \quad \left\{m_{7,k} = \mathtt{sr}_k \textstyle\sum_{j \in [L]} \mathtt{a}_j\right\}_{k \in [L]},$$

$$\left\{\begin{matrix} m_{8,k,\ell} \\ = \mathtt{r}_k \mathtt{a}_\ell \end{matrix}\right\}_{\substack{k,\ell \in [L] \\ :k \neq \ell}}, \quad \left\{\begin{matrix} m_{9,k,\ell} \\ = \mathtt{sr}_k \mathtt{a}_\ell \end{matrix}\right\}_{\substack{k,\ell \in [L] \\ :k \neq \ell}}, \quad \left\{\begin{matrix} m_{10,k,\ell,\ell'} \\ = \mathtt{r}_k \mathtt{a}_\ell \mathtt{a}_{\ell'} \end{matrix}\right\}_{\substack{k,\ell,\ell' \in [L] \\ :k \neq \ell}}, \quad \left\{\begin{matrix} m_{11,k,\ell} \\ = \mathtt{sr}_k \mathtt{a}_\ell \textstyle\sum_{j \in [L]} \mathtt{a}_j \end{matrix}\right\}_{\substack{k,\ell \in [L] \\ :k \neq \ell}}$$

Note that, except for the monomials $m_{6,k,\ell}$, $m_{7,k}$, $m_{8,k,\ell}$, and $m_{9,k,\ell}$, all other monomials are linearly independent of each other. Furthermore, among these exceptions, clearly (the subspace generated by) $\{m_{6,k,\ell}\}_{k,\ell \in [L]} \cup \{m_{8,k,\ell}\}_{k,\ell \in [L]:k \neq \ell}$ is linearly independent of (the subspace generated by) $\{m_{7,k}\}_{k \in [L]} \cup \{m_{9,k,\ell}\}_{k,\ell \in [L]:k \neq \ell}$.

Below, we argue that the set of monomials $\{m_{7,k}\}_{k \in [L]} \cup \{m_{9,k,\ell}\}_{k,\ell \in [L]:k \neq \ell}$ is actually linearly independent. Indeed, suppose

$$\begin{aligned} 0 &= \sum_{k \in [L]} c_{7,k} m_{7,k} + \sum_{k,\ell \in [L]:k \neq \ell} c_{9,k,\ell} m_{9,k,\ell} \\ &= \sum_{k,\ell \in [L]} c_{7,k} \mathtt{sr}_k \mathtt{a}_\ell + \sum_{k,\ell \in [L]:k \neq \ell} c_{9,k,\ell} \mathtt{sr}_k \mathtt{a}_\ell \\ &= \sum_{k,\ell \in [L]:k \neq \ell} (c_{7,k} + c_{9,k,\ell}) \mathtt{sr}_k \mathtt{a}_\ell + \sum_{k \in [L]} c_{7,k} \mathtt{sr}_k \mathtt{a}_k. \end{aligned}$$

The second term forces $c_{7,k} = 0$ for all $k \in [L]$, and the first term forces $c_{9,k,\ell} = -c_{7,k} = 0$ for all $k, \ell \in [L]$ with $k \neq \ell$.

To summarise, if $c_{i,\mathtt{index}}$ denotes the coefficient of $m_{i,\mathtt{index}}$ in any zero polynomial $\Phi \equiv 0$, it holds that $c_{6,k,\ell} = -c_{8,k,\ell} \in \mathbb{Z}_p$ for all $k, \ell \in [L]$ with $k \neq \ell$, and all other coefficients are zero, i.e. identical to the case $b = 1$.

It remains to analyse the behaviour of the Zt_T oracle on input a polynomial $\Phi \in \zeta$ in the cases $b \in \{0, 1\}$. From the above, Φ is a zero polynomial in the case $b = 0$ *iff* it is also a zero polynomial in the case $b = 1$. Therefore, for $\Phi \equiv 0$, the Zt_T oracle behaves identically in the cases $b \in \{0, 1\}$. Next, suppose $\Phi \not\equiv 0$ is a non-zero polynomial. If $b = 0$, the above analysis shows that Φ is of degree $d_0 = 4$. By the Schwartz-Zippel lemma, the probability that the zero-test oracle returns 1 is at most $4/p$. Similarly, if $b = 1$, Φ is of degree $d_1 = 3$, and thus the probability that the zero-test oracle returns 1 is at most $3/p$. Assuming that $\mathcal{A}$ queries the zero-test oracle $Q_{\mathsf{zt}}(\lambda) = \mathsf{poly}(\lambda)$ times, the difference in the probabilities of $\mathcal{A}$ returning 1 in the cases $b \in \{0, 1\}$ is at most $4Q_{\mathsf{zt}}(\lambda)/p + 3Q_{\mathsf{zt}}(\lambda)/p = \mathsf{negl}(\lambda)$. $\square$