Registered FE beyond Predicates: (Attribute-Based) Linear Functions and more

Pratish Datta[⋆], Tapas Pal[†], Shota Yamada[†⋆]

[⋆]NTT Research, Sunnyvale, CA 94085, USA
pratish.datta@ntt-research.com
[†]NTT Social Informatics Laboratories, Tokyo, Japan
tapas.pal@ntt.com
[†⋆]National Institute of Advanced Industrial Science and Technology (AIST), Tokyo, Japan 135-0064
yamada-shota@aist.go.jp

October 12, 2023

Abstract

This paper introduces the *first* registered functional encryption RFE scheme tailored for *linear functions*. Distinctly different from classical functional encryption (FE), RFE addresses the key-escrow issue and negates the master key exfiltration attack. Instead of relying on a centralized trusted authority, it introduces a "key curator" - a fully transparent entity that does not retain secrets. In an RFE framework, users independently generate secret keys and subsequently register their respective public keys, along with their authorized functions, with the key curator. This curator consolidates public keys from various users into a unified, concise master public key. For decryption, users occasionally secure helper decryption keys from the key curator, which they use in conjunction way with their private keys. It is imperative that the aggregate public key, helper decryption keys, ciphertexts, and the times for encryption/decryption are polylogarithmic in the number of registered users.

All existing RFE designs were confined to predicates where given the correct credentials a user can retrieve the entire payload from a ciphertext or gain no information about it otherwise. Contrarily, our RFE scheme facilitates the computation of linear functions on encrypted content and extraction of only the computation results. Recognizing potential leaks from linear functions, we further enhance our RFE by incorporating an attribute-based access control mechanism. The outcome is the *first* registered attribute-based linear FE (RABIPFE), which supports access policies depicted as linear secret sharing schemes LSSS. Our proposed schemes are realized in the common reference string (CRS) model as introduced by Hohenberger et al.[EUROCRYPT 2023], employ simple tools and black-box methods. Specifically, our constructs operate in asymmetric prime-order bilinear group regime setting and are proven secure in the generic bilinear group model. Aligning with all pre-existing black-box RFE designs within the CRS model, our schemes cater to a predetermined maximum user count. A notable variant of our RABIPFE scheme also yields the *first* efficient register ABE (RABE) system for LSSS access policies in asymmetric prime-order bilinear groups. Conclusively, demonstrating feasibility, we formulated an RFE blueprint that supports general functionalities and an infinite user base, leveraging indistinguishability obfuscation and one-way functions.

Contents

1 Introduction

Functional Encryption: Functional Encryption (FE) [BSW11, O'N10] expands upon the traditional public-key encryption paradigm by introducing fine-grained access control over encrypted data. In an FE scheme, a central authority possesses a master secret key and issues a corresponding master public key. Leveraging its master secret key, this authority furnishes users with secret keys corresponding to diverse legitimate functions. Conversely, any party can encrypt data using the master public key. Given a secret key for a function f and a ciphertext of a message x, decryption unveils $f(x)$ without revealing further details about x.

The FE paradigm holds vast potential, presenting myriad applications, both as a standalone solution and as a foundational element for other cryptographic primitives [GKP+13, HJO+16, GPSZ17, GHKW17, Bit17, BGJS17, BV15, AJ15, AJS15, NWZ16]. Given its broad utility, FE and its various subclasses have garnered significant attention in the research community. Below is a non-exhaustive list of notable results in the field [BF01, BB04, BGW05, SW05, GPSW06, KSW08, Wat09, LOS+10b, ABB10b, ABB10a, LW10, OT10, OT12, LW12, Wat12, GVW12, GVW13, BGG+14, Att14, Wee14, GVW15, ABDP15, ALS15, CGW15, BJK15, LV16, AS16, DDM16, BCFG17, BBL17, GKW17, Agr17, Wee17, DOT18, CLT18, CGKW18, TT18, GWW19, AV19, LL20a, LL20b, GW20, ACGU20a, CDSG+20, JLS21, ALMT20, AY20, KW20, Wee20, AGW20, Gay20, Wee21, AMVY21, Wee22, KNT21, DP21, DPT22, JLS22, AKM+22, GGLW22, Tom23]

The Key-Escrow Challenge: While FE offers a powerful means for achieving precise control over encrypted data, it distinctly alters the trust dynamic when juxtaposed with standard public-key encryption. Specifically, in FE, a central, trusted entity is tasked with distributing the secret decryption keys tailored to each user. This central entity must safely maintain a long-term master secret key. A compromise of this authority could grant adversaries the power to decrypt every ciphertext in the system, revealing all encrypted messages. This inherent vulnerability to key exfiltration attacks necessitates meticulous protection of the master secret key for the system's duration. By contrast, with traditional public-key encryption, users autonomously generate their own key pairs without entrusting their secret keys to a central figure. This decentralization eliminates a single point of failure. The amalgamation of inherent key escrow and susceptibility to key exfiltration remains a significant barrier to FE adoption.

Registered FE: Addressing the key-escrow and master key exfiltration vulnerabilities inherent in FE, recent efforts have delved into an innovative encryption framework known as Registered FE (RFE). RFE replaces the central authority with a fully transparent entity known as a "key curator", which does not retain any secrets. Contrary to issuing secret decryption keys, the key curator's primary role revolves around consolidating public keys from registered users into a concise master public key. Elaborating, within an RFE framework, users autonomously generate their public and secret key pairs (mirroring traditional public-key encryption practices). They subsequently register their public keys, along with the functions they are sanctioned for, with the key curator. This entity, in turn, refreshes the scheme's master public key. Analogous to conventional FE, this master public key can encrypt any message x in the system. A registered user, authorized for a specific function f, can decrypt the ciphertext, gleaning solely $f(x)$, utilizing their secret key. This is aided by a publicly computable helper decryption key, which connects the user's public key with the prevailing master public key. Given the dynamic nature of the RFE system, where the master public key evolves as new users are onboarded, it is imperative for users to intermittently update their helper decryption keys throughout the system's lifespan. It is worth noting that these updates for each user can be determined publicly. From an efficiency perspective, if L users are registered, each user should only be tasked with updating their decryption key a maximum of $O(\log L)$ times throughout the system's existence. Further, the magnitude of each update must remain succinct, preferably within the realm of $\mathsf{poly}(\lambda, \log L)$, where λ symbolizes a security parameter. Aligning with standard FE, it is also crucial that the master public key maintains a compact footprint, sized approximately $\mathsf{poly}(\lambda, \log L)$.

Initial Results: Non-Black-Box Constructions: Early research work spearheaded by Garg et al. [GHMR18, GHM+19, GV20, CES21], established RFE schemes within the context of Identity-Based Encryption (IBE),

a subclass of FE. These were developed using well-studied computational assumptions, including CDH, factoring, and LWE. This new primitive was named Registration-Based Encryption (RBE) in those works. However, these constructions extensively relied on non-black-box cryptographic techniques, making them largely infeasible, even when considering subsequent optimization efforts [CES21].

Non-Black-Box Constructions and the CRS Model: A seminal advancement in the domain of RFE was marked by the recent contributions of Hohenberger et al. [HLWW23]. Their work elucidated the concept of Registered ABE (RABE), examining RFE within the broader framework of Attribute-Based Encryption (ABE).

Moreover, the research devised innovative techniques for realizing RABE (with RBE as a specific case) using purely black-box cryptographic methods. Impressively, they designed an efficient RABE scheme for access structures represented as Linear Secret Sharing Schemes (LSSS) in composite order bilinear groups, based on the same established static assumptions for IBE [LW10] and ABE [LOS+10b].

Yet, this black-box approach did require one significant trade-off: the one-time trusted generation of a structured Common Reference String (CRS). At a cursory look, this might seem like a mere transposition of trust. However, it is crucial to underscore that this CRS setup is a one-off process, potentially executed via a multi-party computation protocol, and remains reusable across diverse systems. Importantly, post this setup, the CRS remains the sole trusted element. All subsequent activities of the key curator are both deterministic and auditable. The system's security remains intact unless the initial CRS setup is jeopardized. This is markedly different from conventional FE wherein the central authority's long-term master secret key demands perpetual trust. Any breach, resulting in the unauthorized acquisition of this secret key, grants the perpetrator unfettered access to decrypt every system ciphertext. Hence, this CRS-based RFE framework considerably reduces the inherent trust requisites compared to its traditional FE counterpart.

This innovative approach, focusing on creating efficient black-box RFE architectures within the CRS paradigm using simple cryptographic tools, has catalyzed a renaissance of interest in the cryptographic sphere. This has culminated in a flurry of very recent findings [KMW23, FWW23, FKdP23, DKL+23, ZZGQ23, FFM+23], majority of which are actually concurrent to our work [FKdP23, ZZGQ23, FFM+23, KMW23]. These investigations have spearheaded the development of registration-adapted variants of assorted FE subclasses, such as broadcast encryption [KMW23, FWW23], IBE (tailored for large identities) [DKL+23]([FKdP23]), ABE designed for access policies represented as LSSS and arithmetic branching programs [ZZGQ23], and Inner-Product Predicate Encryption (IPE) [ZZGQ23, FFM+23]. Notably, these constructions [KMW23, FKdP23, ZZGQ23, FFM+23] are characterized by their black-box nature, efficiency, and reliance on simple tools like bilinear pairings.

Limitation of the State of the Art: While the advancements in the field have been commendable, the subclasses of FE for which registration-based variations have been constructed predominantly fall within the realm of predicate encryption (PE). PE, a subclass of FE, associates a secret key with an ID/attribute string, while a ciphertext encrypts a predicate-payload pair (or vice-versa). The decryption process unveils the payload only if the predicate is satisfied by the attribute string; otherwise, a unique null symbol, $\perp$, is revealed. Although PE facilitates fine-grained access controls to encrypted content, its capabilities are restricted. Specifically, it can only expose encrypted data entirely or partially to eligible users and keep concealed from unauthorized ones. In contrast, more potent FE subclasses permit privacy-focused computations on encrypted data, emphasizing the extraction of computational outcomes over the raw data. Despite the massive developments in the field, currently, constructing RFE for functionalities beyond predicates remains an unresolved challenge.

Inner-Product FE: This study pivots to what is arguably one of the simplest function classes richly covered in literature: linear functions or inner-products [ABDP15, ALS15, KLM+18, BJK15, ABKW19, ACGU20a, DDM16, ALMT20, CLT18, Wee17, BBL17, WFL19, Tom19, TT18, TAO16, MKMS21, DP19]. An Inner-Product FE (IPFE) scheme involves encrypting vectors over a specific finite field, with secret keys also devised for vectors within that field. The decryption process discloses the inner product of the message and

secret key vectors. The practicality of inner product functions spans diverse applications, from computing weighted means in descriptive statistics, evaluating polynomials, determining exact thresholds [ABDP15], facilitating hidden-weight coin flips [CS19], to biometric authentication and encrypted data's nearest-neighbour search [KLM+18]. Moreover, IPFE can serve as foundational for creating FE schemes that support advanced function categories, such as quadratic functions [BCFG17, Gay20]. Regrettably, as with other FE subclasses, the key-escrow issue poses a significant challenge to the deployment of IPFE.

Attribute-Based IPFE: FE has further evolved to support even more advanced function classes, merging the attribute-based access control of PE with the evaluation of linear functions on encrypted data. Although IPFE offers broad practical applications, its underlying nature is fragile, with each new secret key release leaking sensitive information. To counteract this vulnerability, Abdalla et al. [ACGU20a] proposed attribute-based IPFE (ABIPFE), a concept that embeds access policies into ciphertext while facilitating linear function evaluations. In an ABIPFE framework, each vector is encrypted under certain access policies, while its secret key corresponds to a combination of an attribute stream and a vector. Successful decryption unveils the inner product of the message and key vector, contingent upon the access policy being met by the attribute. Numerous subsequent studies [AGW20, LLW21, PD21a, DP21, DPT22, DDM+23, DP23] have explored ABIPFE across diverse access policy categories using established tools such as bilinear groups and lattices. Despite such advancements, the threat of master key exfiltration attacks continues to overshadow ABIPFE. Hence, in this work, we pose a pivotal question that has largely remained elusive.
Open problem. *Is it possible to design efficient black-box RFE schemes for function classes beyond predicates, such as for Attribute-Based Linear Function Evaluation?*

Our Results: In this paper, we offer a positive response to the aforementioned open question. Indeed, for the first time in the literature, we formulate the notion of RFE [1], and the design of the primitive for function classes that transcend simple predicates. Specifically, we presented the *first* registered IPFE (RIPFE) and ABIPFE (RABIPFE) schemes. Developed within the CRS model and leveraging simple tools and black-box methodologies, our designs operate within asymmetric prime-order bilinear group setting, which is known to be faster and more secure compared to its other variants [BGJT14, GGMZ13, Jou13, Jou14]. Security is assured within the generic bilinear group model (GGM) [Sho97a]. The proposed RABIPFE system incorporates LSSS access policies, the zenith of policy expressiveness achieved by current ABE/ABIPFE constructions rooted in bilinear groups, even within conventional centralized frameworks. As a special case of our RABIPFE construction, we also present the first registered ABE (RABE) system in a prime-order bilinear group setting.

Our schemes are tailored to accommodate a pre-determined number of users. Specifically, the structured CRS dimensions are quadratically proportional to user numbers, while registration performance is linearly dependent on user count, aligning with existing black-box RFE structures within the CRS model. Further, analogous to the current CRS-based RFE schemes, we necessitate a thorough verification of user public keys before their registration in the system to address concerns regarding malicious public keys. Similar to the existing RABE scheme, our RABIPFE system can accommodate an attribute within access policies either singularly or within set limitations, with parameter dimensions expanding linearly based on repetition limits via a simple encoding technique similar to this [Wat11, LW11a]. Existing techniques for handling arbitrary repetitions of the same attributes within an access policy from centralized ABE literature [KW19, LL20b, LL20a] seemingly lack direct applicability in the registration-based context, leaving ample room for future investigation.

As our objective veers towards designing RFEs with functionalities surpassing predicates, we have to devise a different randomization strategy than existing works. While prior works like [HLWW23] naturally

[1] All existing studies in this field have specifically defined the concept of RFE in accordance with the particular function classes they explored, often under varied terminologies. For instance, Garg et al. [GHMR18] utilized the term "registration-based encryption (RBE)". Hohenberger et al. [HLWW23] introduced the term "registered ABE (RABE)". Kolonelos et al. [KMW23] put forth the concept of "distributed broadcast encryption (DBE)". Similarly, Freitag et al. [FWW23] coined the term "flexible broadcast encryption (FBE)", and the list goes on. What distinguishes our work is the formal definition of RFE in its broadest sense, encompassing all existing registration-based primitives as particular instances of this overarching notion.

separated the masking component from the CRS into random shares to oversee predicate validation and membership verification, this does not work in our setting. Broadly speaking, the challenge arises from the distinction that, in the context of predicates, each user's output is either zero or one. However, in our setting, the outputs for different users can diverge significantly. To resolve this issue, our approach directly randomizes the component concealing the inner-product value during decryption. This novel process however, inherently risks the intermingling of decryption threads, potentially leading to vulnerabilities. To navigate these challenges, our refined randomization process, in essence, links the key vector to pertinent users while concurrently managing functional evaluation and membership validation in an integrated fashion. Such technique was previously used implicitly by Waters [Wat11] in the context of centralized ABE. We not only extend this method beyond predicates but also adapt it to the registration-based framework, while presenting it in a more explicit manner (see Section 2 for further details).

Further, beyond the prime-order bilinear group-centric schemes, we also detail the blueprint for an RFE system catering to an indefinite number of users and supporting general functionalities via indistinguishability obfuscation (IO) and one-way functions (OWFs). An obfuscator, as defined in [BGI+01], is a tool that converts a circuit into an equivalent one, *i.e.* preserving its input-output behaviour, while concealing the original circuit's confidential data. Indistinguishability obfuscator [BGI+01] is a specific type of obfuscator ensuring that any two equivalent circuits' obfuscations are indistinguishable. Coupled with the seminal work of Jain et. al. [JLS21, JLS22], realizing IO from falsifiable assumptions, this leads to an RFE system for arbitrary functionalities and user counts grounded in falsifiable assumptions. This latter achievement stands as a testament to the potential of RFE systems to accommodate versatile functionalities and an expansive user base.

Concurrent Works: In parallel and independent research, Francati et al. [FFM+23] provides robust attribute hiding registered *zero* inner-product PE scheme, in prime-order bilinear group under the generic bilinear group model (GGM). The attribute hiding security ensures that the ciphertext obscures the associated attribute vector from all system users, even those with decryption rights. However, this RFE remains within the context of RFE for predicates as previously described, and fully discloses payload data to authorized individuals. Contrasting this, our RFE scheme for linear or inner-product functionality does not just reveal encrypted data. Instead, it computes and outputs the inner-product of encrypted content.

Further, just like our work, Francati et.al [FFM+23] also presented an RFE scheme for general functionalities and supporting an arbitrary number of users from IO and OWFs. The differences between the two constructions are as follows. Francati et.al [FFM+23] essentially observed that the construction of RABE due to Hohenberger et al. [HLWW23] for general access policies and an arbitrary number of users from IO and OWFs actually works as a full pleaded RFE that can support arbitrary functionalities beyond predicates. On the other hand, we tweak the RABE scheme of [HLWW23] by introducing a Naor-Yung style [NY90] "double-encryption" mechanism inspired by the techniques of Garg et.al [GHMR18]. However, rather than using a simulation *sound* non-interactive zero-knowledge (NIZK) proof system as [NY90, GHMR18], our construction only employs Lamport's one-time signatures [Lam79] which can be realized from OWFs.

In another synchronized study, Zhu et al. [ZZGQ23] introduced RABE schemes suitable for LSSS access policies and non-attribute-hiding inner-product predicates. They utilized asymmetric prime-order bilinear groups and proved security in the standard model under the k-linear assumption [EHK+13]. In contrast, while the RABE scheme, which emerges as a particular instance of our RABIPFE, is also constructed using asymmetric prime-order bilinear groups, its security is guaranteed solely within the generic bilinear group model (GGM). However, the RABE scheme of Zhu et al. [ZZGQ23] appears to leverage the well-known composite-to-prime-order conversion framework [CGKW18, CGW15] derived from centralized ABE literature on the RABE scheme of [HLWW23]. As a result, it inherits the inefficiencies of that framework, namely, the generation of large ciphertext/secret keys and diminished performance. In contrast, our RABE scheme is straightforward and delivers notably superior concrete performance. Furthermore, for user public key validation, Zhu et al. [ZZGQ23] deploy a strong cryptographic primitive: a pairing-based unbounded simulation-secure Quasi-adaptive non-interactive zero-knowledge proof system. In comparison, we propose a streamlined public key validation technique harnessing the power of GGM. Lastly, while both of their

schemes primarily address predicates, our registration-based renditions of IPFE and ABIPFE surpass this scope, facilitating the evaluation of linear functions on encrypted data.

2 Technical Overview

In this work, we construct registration-based FE schemes for general as well as specific function classes. Regarding the specific classes, in the registration-based setting, we consider FE for linear functions which is known as IPFE [ABDP15] and FE for linear functions with access control[2] which is noted as ABIPFE [ACGU20b]. Before going to the technical descriptions of the designing of these primitives in the registration-based setting, we first provide an overview of the definition of registered FE for general functions.

2.1 Definition of Registered FE

Let $\mathcal{U}_F$ be the universe of functions and $\mathcal{M}$ be the set of messages supported by the scheme. We also assume that $\mathcal{U}_F$ contains only polynomial-size functions having maximum size ℓ_f in bit-length. There is a one-time trusted setup which samples a common reference string (CRS) crs depending on the security parameter and the bound ℓ_f. We allow the size of crs to be $\mathsf{poly}(\lambda, \ell_f, L)$. The key curator first initializes an empty master public key MPK when there is no user in the system at the beginning. If a user wants to join the system then it first samples a public-secret key pair $(\mathsf{pk}, \mathsf{sk})$ using crs, and then sends the public key pk along with a function $f_{\mathsf{pk}} \in \mathcal{U}_F$ to the key curator for registration. The key curator then aggregates the pair $(\mathsf{pk}, f_{\mathsf{pk}})$ into the current master public key MPK and outputs an updated one MPK'. Additionally, the user also receives a helper decryption key hsk from the key curator.

The key curator does not hold any secret and the role can be played by anyone in the system as well since the process of aggregation is *deterministic*. The key generation and registration are both allowed to run in time $\mathsf{poly}(\lambda, \ell_f, L)$. However, in our actual constructions, the key generation process does not depend on ℓ_f. On the other hand, we require that the size of the secret key sk, the master public key MPK (at any stage) and the helper decryption key hsk for each user must be *polylogarithmic* in the total number of users, *i.e.*, $\mathsf{poly}(\lambda, \ell_f, \log L)$. Whenever a new user joins the system, the master public key is updated and as a result, the existing users might need an updated helper decryption key from the key curator. As in existing registration-based systems, we require that the actual number of updates needed for a helper decryption key of each user is essentially $O(\log L)$ throughout the existence of the system.

In our setting, the knowledge of MPK is sufficient for encrypting a message $m \in \mathcal{M}$. Any registered users whose public key-function pair $(\mathsf{pk}, f_{\mathsf{pk}})$ is integrated into the master public key MPK can decrypt the ciphertext ct using their secret keys sk and the helper decryption keys hsk. It is important to note that the large crs is not required at all during encryption or decryption, the information of crs is requested for generating keys of users and at the time of registering a user. This makes the encryption and decryption algorithm much more efficient, both of which run in time $\mathsf{poly}(\lambda, \ell_f, \log L)$ and are comparable to exiting (non-registered) FEs for specific class of functions such as IPFE. We formally define registered FE in Section 4.

Slotted Registered FE: We followed the blueprint of Hohenberger et al. [HLWW23] for constructing a registered encryption scheme. In particular, we first define and construct a slotted registered FE (SlotRFE) scheme and then use a transformation to achieve the full-fledged RFE scheme described above. A SlotRFE scheme is basically an RFE scheme where the total number of users L is *fixed* at the time of setup, and each user of the system is identified via a slot index $i \in [L]$. Therefore, each slot i is associated with a user-sampled key pair $(\mathsf{pk}_i, \mathsf{sk}_i)$ and a function $f_i \in \mathcal{U}_F$. The aggregation algorithm can be run only when a list of public key-function pair (pk_i, f_i) for all $i \in [L]$ is available. It uses the list to output the aggregated master public key MPK and a helper decryption key hsk_i for each user $i \in [L]$, that is all the users are registered in one shot. Note that, no update of the master public key or helper decryption keys is needed since no new user is allowed to join the system once the registration is over. For a SlotRFE, we require that

[2]It is like having ABE on top of IPFE.

the sizes of the master public key and helper decryption keys must grow at most poly-logarithmically with the total number of users in the system. The formal definition of SLotRFE is given in Section 5.

We present a transformation to go from SlotRFE to RFE. The idea of the conversion essentially follows from the similar transformation used for RABE by Hohenberger et al. [HLWW23]. In Section 9, we adapt their transformation into the setting of FE to construct RFE from our SlotRFE. It depends on a *power-of-two* approach that uses $\ell + 1$ many SlotRFE schemes for building an RFE scheme with $L = 2^\ell$ users. Just like the RABE, the public parameters $(\mathsf{crs}, \mathsf{MPK}, \mathsf{hsk})$, ciphertext size and encryption time of the resulting RFE carry an overhead of $O(\log L)$ compared to the underlying SlotRFE scheme. This means that if the CRS of the SlotRFE scales with at most $O(\log L)$ then the RFE can support an exponential number of users. It is exactly the case for our IO-based RFE scheme for general functions. However, our pairing-based RFE schemes can only support a (polynomially) bounded number of users[3] since corresponding slotted versions produce a CRS of size $O(L^2)$.

2.2 Registered FE for (Attribute-Based) Linear Functions

In this subsection, we first provide technical ideas of our registered IPFE where functions and messages are vectors and decryption recovers the inner product between the vectors. Then, we discuss the technical ideas of constructing a registered ABIPFE.

Recap of plain IPFE [ABDP15]: Let us first describe the IPFE construction by Abdalla et. al [ABDP15], which is not registration-based, but serves as our starting point. Their construction works over groups of prime order without pairings. In their construction, the master public key consists of group elements $\mathsf{MPK} = (g, g^{\boldsymbol{\alpha}})$, where $\boldsymbol{\alpha} \in \mathbb{Z}_p^n$ and n is the dimension of the vectors supported by the scheme. The ciphertext encrypting a vector $\boldsymbol{x} \in \mathbb{Z}_p^n$ is of the form $\mathsf{ct} = (g^{s\boldsymbol{\alpha}+\boldsymbol{x}}, g^s)$, where $s \leftarrow \mathbb{Z}_p$, and a secret key associated with a vector $\boldsymbol{y} \in \mathbb{Z}_p^n$ is $\mathsf{sk} = \langle \boldsymbol{\alpha}, \boldsymbol{y} \rangle$. To decrypt a ciphertext, we first compute $g^{s\langle \boldsymbol{\alpha}+\boldsymbol{x}, \boldsymbol{y}\rangle} = g^{s\langle \boldsymbol{\alpha}, \boldsymbol{y}\rangle + \langle \boldsymbol{x}, \boldsymbol{y}\rangle}$ from $g^{s\boldsymbol{\alpha}+\boldsymbol{x}}$ and remove the masking term $g^{s\langle \boldsymbol{\alpha}, \boldsymbol{y}\rangle}$ using g^s and the secret key $\langle \boldsymbol{\alpha}, \boldsymbol{y} \rangle$. Then, we recover $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ from $g^{\langle \boldsymbol{x}, \boldsymbol{y}\rangle}$ by the brute-force computation.

Let us briefly discuss the intuition behind the security of the construction. Suppose that the adversary is given secret keys $\{\langle \boldsymbol{\alpha}, \boldsymbol{y}_i \rangle\}_i$ corresponding to vectors $\{\boldsymbol{y}_i\}_i$. Intuitively, only meaningful way to get information of $\boldsymbol{x}$ from the ciphertext is to take linear combination between the ciphertext components to compute $g^{s\langle \boldsymbol{\alpha}+\boldsymbol{x}, \boldsymbol{z}\rangle} = g^{s\langle \boldsymbol{\alpha}, \boldsymbol{z}\rangle + \langle \boldsymbol{x}, \boldsymbol{z}\rangle}$ for some vector $\boldsymbol{z}$ and remove the masking term $g^{s\langle \boldsymbol{\alpha}, \boldsymbol{z}\rangle}$ to recover the information of $\langle \boldsymbol{x}, \boldsymbol{z} \rangle$. Since the adversary is given only $\{\langle \boldsymbol{\alpha}, \boldsymbol{y}_i \rangle\}_i$, it is impossible for her to obtain any information of $\langle \boldsymbol{\alpha}, \boldsymbol{z} \rangle$ when $\boldsymbol{z}$ is outside of the span of the vectors $\{\boldsymbol{y}_i\}_i$. This in turn means that the information of $\langle \boldsymbol{x}, \boldsymbol{z} \rangle$ cannot be obtained if $\boldsymbol{z}$ is outside of the span, as desired.

Attempt 1: In our first attempt, we consider a construction that supports only a single user. Even in this setting, we face the challenge that there is no obvious way to generate a secret key, because the secret key generation of the plain IPFE construction we explained above crucially requires the knowledge of the master secret key. Translated into the setting of registration-based IPFE, this means that the key generation requires the knowledge of a trapdoor corresponding to the CRS, which is not known to the user. To resolve the problem, we observe that what is necessary for the decryption is actually the masking term $g^{s\langle \boldsymbol{\alpha}, \boldsymbol{y}\rangle}$. We construct the scheme so that the masking term can be recovered by the decryptor by exploiting the fact that the master public key can depend on the vector $\boldsymbol{y}$ in the registration-based setting.

Concretely, the CRS of the construction is the same as the master public key of the IPFE we explained. Namely, we set $\mathsf{crs} = (g, g^{\boldsymbol{\alpha}})$. A user who joins the system chooses random $r \leftarrow \mathbb{Z}_p$ and sets the public key as $\mathsf{pk} = g^r$ and the secret key as $\mathsf{sk} = r$. We set the master public key as $\mathsf{MPK} = (g, g^{\boldsymbol{\alpha}}, W = g^{r+\langle \boldsymbol{\alpha}, \boldsymbol{y}\rangle})$, where $\boldsymbol{y}$ is the vector associated with the user. The ciphertext encrypting $\boldsymbol{x}$ is $\mathsf{ct} = (g^{s\boldsymbol{\alpha}+\boldsymbol{x}}, g^s, W^s = g^{sr+s\langle \boldsymbol{\alpha}, \boldsymbol{y}\rangle})$. A user can therefore recover $g^{s\langle \boldsymbol{\alpha}, \boldsymbol{y}\rangle}$ by computing g^{sr} from g^s and r and then compute W^s/g^{sr}. The rest of the decryption algorithm is the same as the plain IPFE construction explained above.

[3] In fact, all existing pairing-based RABEs have the limitation of supporting only a bounded number of users.

While the construction works for the single-user setting, the apparent problem is that there is no obvious way to extend the construction to the multi-user case. One could consider a natural attempt where we set $\mathsf{pk}_i = g^{r_i}$ for each user indexed with i and aggregate the public keys to set the master public key as $\mathsf{MPK} = (g, g^{\boldsymbol{\alpha}}, W = \prod_i g^{r_i + \langle \boldsymbol{\alpha}, \boldsymbol{y}_i \rangle} = g^{\sum_i r_i + \sum_i \langle \boldsymbol{\alpha}, \boldsymbol{y}_i \rangle})$. Suppose that we were able to set the ciphertext and helper decryption key so that the decryption is possible. Then, a collusion of two users should be able to recover $\langle \boldsymbol{x}, \boldsymbol{y}_1 + \boldsymbol{z} \rangle$ and $\langle \boldsymbol{x}, \boldsymbol{y}_1 - \boldsymbol{z} \rangle$ for arbitrary $\boldsymbol{z}$, which breaks the security of the scheme. This is because $\mathsf{sk}_1 = r_1$ and $\mathsf{sk}_2 = r_2$ are valid secret keys for vectors $\boldsymbol{y}_1 + \boldsymbol{z}$ and $\boldsymbol{y}_2 - \boldsymbol{z}$ respectively, since we have $W = g^{r_1 + \langle \boldsymbol{\alpha}, \boldsymbol{y}_1 + \boldsymbol{z} \rangle} \cdot g^{r_2 + \langle \boldsymbol{\alpha}, \boldsymbol{y}_2 - \boldsymbol{z} \rangle} \cdot g^{\sum_{i \neq 1,2} r_i + \sum_i \langle \boldsymbol{\alpha}, \boldsymbol{y}_i \rangle}$. At a high level, the construction is insecure, since each vector $\boldsymbol{y}_i$ is not bound to the corresponding index i.

Attempt 2: Based on the above observation, in our second attempt, we computationally bind each vector with the corresponding index. Namely, we set $\mathsf{crs} = (g, g^{\boldsymbol{\alpha}}, g^{\beta_1}, \ldots, g^{\beta_L}, g^{\beta_1 \boldsymbol{\alpha}}, \ldots g^{\beta_L \boldsymbol{\alpha}})$, where L is the number of users in the system and $\beta_i \leftarrow \mathbb{Z}_p$ for each i. We set the secret key for user i as $\mathsf{sk}_i = r_i$ and the corresponding public key as $\mathsf{pk}_i = g^{\beta_i r_i}$. Given the set of public keys $\{\mathsf{pk}_i\}_{i \in [L]}$ and corresponding vectors $\{\boldsymbol{y}_i\}_{i \in [L]}$, the master public key is set as $\mathsf{MPK} = (g, g^{\boldsymbol{\alpha}}, W = \prod_{i \in [L]} \mathsf{pk}_i \cdot g^{\beta_i \langle \boldsymbol{\alpha}, \boldsymbol{y}_i \rangle} = \prod_{i \in [L]} g^{\beta_i (r_i + \langle \boldsymbol{\alpha}, \boldsymbol{y}_i \rangle)})$. The difference from the previous attempt is that we separate the thread of the computation for each user by the individual randomness β_i. To encrypt the vector $\boldsymbol{x}$, we compute $\mathsf{ct} = (g^s, g^{s\alpha + \boldsymbol{x}}, \mathsf{MPK}^s = \prod_i g^{s \beta_i (r_i + \langle \boldsymbol{\alpha}, \boldsymbol{y}_i \rangle)})$. Although it seems that now the construction is secure, we do not know how to decrypt the ciphertext. During the decryption, a user indexed with i may want to unmask $g^{s \beta_i r_i}$, but she only knows g^s, g^{β_i}, and r_i and thus this task is impossible. This motivates us to use the (symmetric) pairings in our next attempt.

Attempt 3: In our third attempt, we construct the scheme so that a user who knows r_i can remove the masking term. Towards this goal, we change the CRS as $\mathsf{crs} = (g, g_T^{\boldsymbol{\alpha}}, g^{\beta_1}, \ldots, g^{\beta_L}, g^{\beta_1 \boldsymbol{\alpha}}, \ldots, g^{\beta_L \boldsymbol{\alpha}}, g^{1/\beta_1}, \ldots, g^{1/\beta_L})$. The forms of the public keys, secret keys, and master public key are the same as the previous attempt except that now the group components are in the source group. We change the form of the ciphertext as $\mathsf{ct} = (g_T^s, g_T^{s\boldsymbol{\alpha} + \boldsymbol{x}}, W^s = \prod_j g^{s \beta_j (r_j + \langle \boldsymbol{\alpha}, \boldsymbol{y}_j \rangle)})$, where $g_T = e(g, g)$. To decrypt the ciphertext, user i computes

$$e(W^s, g^{1/\beta_i}) = e(\prod_j g^{s \beta_j (r_j + \langle \boldsymbol{\alpha}, \boldsymbol{y}_j \rangle)}, g^{1/\beta_i}) = g_T^{s r_i} \cdot g_T^{s \langle \boldsymbol{\alpha}, \boldsymbol{y}_i \rangle} \cdot \underbrace{\prod_{j \neq i} g_T^{s \beta_j r_j / \beta_i} \cdot \prod_{j \neq i} g_T^{s \beta_j \langle \boldsymbol{\alpha}, \boldsymbol{y}_j \rangle / \beta_i}}_{= \text{Cross term}}.$$

Here, the user can unmask the term $g_T^{s r_i}$ using g_T^s and her secret key r_i. However, to retrieve the desired term $g_T^{s \langle \boldsymbol{\alpha}, \boldsymbol{y}_i \rangle}$, she also has to remove the cross term. In our next attempt, we enforce the users to compute extra components and include them into the public key when they register into the system. These extra components will enable the decryptor to compute the cross term.

Attempt 4: In our fourth attempt, we set $\mathsf{crs} = (g, g^{1/\gamma}, g_T^{\boldsymbol{\alpha}}, \{g^{\beta_j}, g^{\beta_j \boldsymbol{\alpha}}, g^{1/\beta_j}\}_j, \{g^{\gamma \beta_j / \beta_k}, g^{\gamma \beta_j \boldsymbol{\alpha} / \beta_k}\}_{j \neq k})$, where $\gamma \leftarrow \mathbb{Z}_q$ and the extra components will be used for computing the cross terms. We then enforce user i to compute and publicize $\{g^{\gamma \beta_i r_i / \beta_j}\}_{j \neq i}$ when it registers. Namely, we set $\mathsf{pk}_i = (g^{\beta_i r_i}, \{g^{\gamma \beta_i r_i / \beta_j}\}_{j \neq i})$ and $\mathsf{sk}_i = r_i$. The aggregation algorithm is going to be a bit more complex since it computes helper decryption keys $\{\mathsf{hsk}_i\}_i$ in addition to MPK. Concretely, given the public keys $\{\mathsf{pk}_i\}_i$ and corresponding vectors $\{\boldsymbol{y}_i\}_i$, the aggregation algorithm computes

$$\mathsf{MPK} = \left(g_T, g_T^{\boldsymbol{\alpha}}, g^{1/\gamma}, W = \prod_{i \in [L]} g^{\beta_i (r_i + \langle \boldsymbol{\alpha}, \boldsymbol{y}_i \rangle)} \right), \quad \mathsf{hsk}_i = \prod_{j \neq i} \left(g^{\beta_j \gamma r_j / \beta_i} \cdot g^{\beta_j \gamma \langle \boldsymbol{\alpha}, \boldsymbol{y}_j \rangle / \beta_i} \right).$$

The ciphertext is now $\mathsf{ct} = (g_T^s, g^{s/\gamma}, g_T^{s\boldsymbol{\alpha} + \boldsymbol{x}}, W^s)$. The cross term then can be recovered by computing

$$e(g^{s/\gamma}, \mathsf{hsk}_i) = \prod_{j \neq i} e\left(g^{s/\gamma}, g^{\beta_j \gamma r_j / \beta_i} \cdot g^{\beta_j \gamma \langle \boldsymbol{\alpha}, \boldsymbol{y}_j \rangle / \beta_i} \right) = \prod_{j \neq i} g_T^{s \beta_j r_j / \beta_i} \cdot \prod_{j \neq i} g_T^{s \beta_j \langle \boldsymbol{\alpha}, \boldsymbol{y}_j \rangle / \beta_i}$$

as desired.

One thing missing from the above discussion is how to check the validity of the public key registered by the user. Given the public key $\mathsf{pk}_i = (U_i, \{V_{i,j}\}_{j\neq i})$, we can check that it is in the valid form in the sense that there exists r_i such that $U_i = g^{\beta_i r_i}$ and $V_{i,j} = g^{\gamma\beta_i r_i/\beta_j}$ by checking $e(U_i, g^{1/\beta_j}) = e(V_{i,j}, g^{1/\gamma})$ for all $j \neq i$. However, this check does not ensure that the user actually followed the protocol to compute U_i: a malicious user might have deviated from the protocol and still have passed the verification. For example, the user might have used g^{β_i} and $g^{\beta_i\boldsymbol{\alpha}}$ to compute U_i as $U_i = g^{\beta_i r_i + \beta_i\langle\boldsymbol{\alpha},\boldsymbol{z}\rangle}$ for some $\boldsymbol{z}$ and computed corresponding $V_{i,j} = g^{\gamma\beta_i r_i/\beta_j + \beta_i\langle\boldsymbol{\alpha},\boldsymbol{z}\rangle/\beta_j}$ from $g^{\gamma\beta_i/\beta_j}$ and $g^{\beta_i\boldsymbol{\alpha}/\beta_j}$. Such a user can certainly pass the verification. However, the user is able to decrypt the ciphertext with respect to the vector $\boldsymbol{y}_i + \boldsymbol{z}$, namely, it can recover $\langle\boldsymbol{x}, \boldsymbol{y}_i + \boldsymbol{z}\rangle$, even though it registered into the system with the vector $\boldsymbol{y}_i$, since we have $W = g^{\beta_i(r_i+\langle\boldsymbol{\alpha},\boldsymbol{y}_i+\boldsymbol{z}\rangle)} \cdot \prod_{j\neq i} g^{\beta_j(r_j+\langle\boldsymbol{\alpha},\boldsymbol{y}_j\rangle)}$, which is problematic. Therefore, it is not enough to check that the public key is in a valid form. Rather, we have to check that the public key is computed following the exact procedure specified by the protocol. A straightforward solution to ensure this is to use non-interactive zero-knowledge proof of knowledge (NIZK-PoK), where we add the CRS of NIZK-PoK to the CRS (of registration-based IPFE) and let the user i prove the knowledge of r_i when it registers. While this idea may work, it is inefficient and indirect. In the next step, we provide a much more direct and efficient solution to the problem using asymmetric pairings.

Our Final Construction: We then explain our final construction. In the construction, we use the asymmetric pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ with generators $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$. In the construction, U_i resides in $\mathbb{G}_1$ and is computed as $U_i = g_1^{\beta_i r_i}$. We let the user compute the copy $\tilde{U}_i = g_2^{\beta_i r_i}$ of U_i in $\mathbb{G}_2$ when it registers, which is meant to serve as a proof that U_i is generated following the honest procedure of the protocol. By carefully placing the group components into $\mathbb{G}_1$ and $\mathbb{G}_2$, we can prevent the above attack. In more detail, we place $\{g_1^{\beta_i}\}_i$ and $\{g_2^{\gamma\beta_i/\beta_j}\}_{i,j}$ in the CRS so that the user can compute U_i and $V_{i,j}$. The CRS also includes $\{g_1^{\beta_i\boldsymbol{\alpha}}\}_i$, which is used for computing the master public key. We further include $\{g_2^{\beta_i}\}_i$ in the CRS so that the copy $\tilde{U}_i$ of U_i can be computed. However, we do *not* include $\{g_2^{\beta_i\boldsymbol{\alpha}}\}_i$ in the CRS and thus the adversary is not able to mount the above attack.

Here, we provide a concrete description of our construction. First, we set

$$\mathsf{crs} = \left(g_1, g_2, g_T^{\boldsymbol{\alpha}}, g_1^{1/\gamma}, \{g_1^{\beta_i}, g_2^{\beta_i}, g_2^{1/\beta_i}\}_i, \{g_2^{\gamma\beta_i/\beta_j}, g_2^{\gamma\beta_i\boldsymbol{\alpha}/\beta_j}\}_{i\neq j}\right). \tag{1}$$

When a user with index i registers, it sets the public key as $\mathsf{pk}_i = (U_i = g_1^{\beta_i r_i}, \tilde{U}_i = g_2^{\beta_i r_i}, \{V_{i,j} = g_2^{\gamma r_i\beta_i/\beta_j}\}_{j\neq i})$ and $\mathsf{sk}_i = r_i$. The verification of the public key is done by checking $e(U_i, g_2) = e(g_1, \tilde{U}_i)$ and $e(U_i, g_2^{1/\beta_j}) = e(g_1^{1/\gamma}, V_{i,j})$ for all $j \neq i$. The aggregation algorithm computes

$$\mathsf{MPK} = \left(g_T, g_T^{\boldsymbol{\alpha}}, g_1^{1/\gamma}, W = \prod_{i\in[L]} g_1^{\beta_i(r_i+\langle\boldsymbol{\alpha},\boldsymbol{y}_i\rangle)}\right), \quad \mathsf{hsk}_i = \prod_{j\neq i}\left(g_2^{\beta_j\gamma r_j/\beta_i} \cdot g_2^{\beta_j\gamma\langle\boldsymbol{\alpha},\boldsymbol{y}_j\rangle/\beta_i}\right). \tag{2}$$

A ciphertext encrypting a vector $\boldsymbol{x}$ is

$$\mathsf{ct} = (g_T^s, g_T^{s\boldsymbol{\alpha}+\boldsymbol{x}}, g_1^{-s/\gamma}, W^s).$$

We omit the description of the decryption algorithm here. We observe that the sizes of MPK and hsk_i are compact, both of which are $\mathsf{poly}(\lambda, n, \log L)$[4].

Overview of the Security Proof: We prove the security of our construction in the generic group model. Roughly speaking, in the generic group model, the only way for the adversary to obtain non-trivial information encoded on the exponents of the group elements is to find a non-trivial linear combination of pairing products that equals zero. In the first step of the proof, we show that (1) the only way for the adversary

[4]We assume that the master public key MPK is implicitly included in each user's helper decryption key hsk_i.

to pass the verification when it registers the public key is to follow the honest key generation procedure. We then show that (2) the only way for the adversary to obtain non-trivial information on the encrypted vectors (*i.e.*, messages) is to follow the honest decryption procedure or take a linear combination between them. These two facts immediately imply the security of the construction.

We first show (1). In particular, we show that when the adversary passes the verification, it should have computed U_i as $U_i = g_1^{\beta_i r_i}$ using r_i. In the proof, we show that if the adversary deviates from the correct procedure of computing U_i, then it cannot compute the associating copy $\tilde{U}_i$ or $\{V_{i,j}\}_{j\neq i}$. For example, suppose that the adversary inserts the term $g_1^{1/\gamma}$ into U_i as $U_i = g_1^{\beta_i r_i} \cdot g_1^{t/\gamma}$ for some $t \in \mathbb{Z}_q$. Then, in order to pass the verification, the adversary has to compute $\tilde{U}_i = g_2^{\beta_i r_i} \cdot g_2^{t/\gamma}$. However, the term $g_2^{1/\gamma}$ is missing from the CRS, there is no way for the adversary to compute $\tilde{U}_i$. Other cases can be dealt with in a similar manner.[5]

We then explain the overview of the proof of (2), which is shown in several steps. In the first step, we show that the ciphertext component W^s should be paired with (linear combination of) $\{g_2^{1/\beta_i}\}_i$ terms, since otherwise the result of the pairing computation includes terms that can never be cancelled by any other pairing products. For example, if we pair W^s with $g_2^{\beta_i}$, the pairing product includes the term of the form $g_T^{s\beta_i\beta_j r_i}$. However, this term cannot be cancelled inside the linear combination of the pairing products, since any other combination of the terms does not yield $g_T^{s\beta_i\beta_j r_i}$ as a result of the pairing computation. This means at a high level that there is no non-trivial information that is obtained by inserting $e(W^s, g_2^{\beta_i})$ into the linear combination.

We then focus the term $e(W^s, g_2^{1/\beta_i})$. By (1), all the public keys should be correctly generated including the ones that are generated by the adversary. In particular, we have $W = \prod_{i\in[L]} W_i = \prod_{i\in[L]} g_1^{\beta_i(r_i + \langle \boldsymbol{y}_i, \boldsymbol{\alpha}\rangle)}$ for some $\{r_i\}$. We therefore have

$$e(W^s, g_2^{1/\beta_i}) = g_T^{sr_i} \cdot g_T^{s\langle \boldsymbol{\alpha}, \boldsymbol{y}_i\rangle} \cdot \underbrace{\prod_{j\neq i} g_T^{s\beta_j r_j/\beta_i} \cdot \prod_{j\neq i} g_T^{s\beta_j \langle \boldsymbol{\alpha}, \boldsymbol{y}_j\rangle/\beta_i}}_{=\text{Cross term}}.$$

Ignoring the cross terms, the above component is similar to the message-carrying part of the plain IPFE we first introduced. Indeed, our proof from here closely follows the intuition of why the plain IPFE scheme is secure. First, we show that if r_i is not known to the adversary, then it cannot unmask the term $g_T^{sr_i}$. This means that the adversary can insert $e(W^s, g_2^{1/\beta_i})$ into the linear combination only when the index i is corrupted or the public key for this index is generated by the adversary herself. In both cases, the adversary can unmask the term $g_T^{sr_i}$ using the knowledge of r_i. However, she still has to compute and unmask the term $g_T^{s\langle \boldsymbol{\alpha}, \boldsymbol{y}_i\rangle}$. By inspection, we can show that the only possible way to unmask $g_T^{s\langle \boldsymbol{\alpha}, \boldsymbol{y}_i\rangle}$ is to compute $g_T^{s\langle \boldsymbol{\alpha}, \boldsymbol{y}_i\rangle + \langle \boldsymbol{x}, \boldsymbol{y}_i\rangle}$ using $g_T^{s\boldsymbol{\alpha}+\boldsymbol{x}}$ and then subtract the term from it. As a result, we will obtain $g_T^{\langle \boldsymbol{x}, \boldsymbol{y}_i\rangle}$, which only contains the information of $\langle \boldsymbol{x}, \boldsymbol{y}_i\rangle$. To sum up, if the adversary wants to obtain non-trivial information of the encrypted vector $\boldsymbol{x}$ from the ciphertext, it should take the linear combination among the ciphertext components in a way that the information of $\boldsymbol{x}$ is lost except for $\langle \boldsymbol{x}, \boldsymbol{y}_i\rangle$, where i is an index that is corrupted or the corresponding public key is generated by the adversary herself. This means that the information of $\boldsymbol{x}$ does not leak to the adversary more than necessary, since $\langle \boldsymbol{x}, \boldsymbol{y}_i\rangle$ for such i is revealed to the adversary anyway by the correctness of the protocol. The full construction and analysis are provided in Section 6.

Registered ABIPFE: Our pairing-based registered ABIPFE scheme provides attribute-based access control over IPFE. In the slotted version, each user is registered with a vector $\boldsymbol{y}_i$ and an attribute set Att_i whereas the encryption of $\boldsymbol{x}$ is performed under an access policy P which is represented by a linear secret sharing scheme (LSSS). We recall that an access structure of LSSS is specified by a matrix $\mathbf{M} \in \mathbb{Z}_p^{K\times N}$ and a mapping ρ which associates distinct attributes to the row indices of $\mathbf{M}$. To share a secret s, we first sample a

[5] Actually, the adversary can pass the verification by randomizing an honestly generated public key. However, there is no gain for the adversary to perform this type of malicious key generation as we will show in the formal proof. We ignore this subtle point in this overview and defer the full details to the formal proof.

random vector $\boldsymbol{v} = (s, v_2, \ldots, v_N)$ and compute the shares $\boldsymbol{u} = \mathbf{M}\boldsymbol{v}$. The i-th component of $\boldsymbol{u}$ is the share associated with the attribute $\rho(i)$. The reconstruction of the secret is possible with a set of attributes Att that satisfies the access structure. More specifically, there exists a vector $\boldsymbol{\omega}$ such that $\boldsymbol{\omega}^\top \boldsymbol{u}_{\mathsf{Att}} = \mathbf{M}_{\mathsf{Att}}\boldsymbol{v} = s$ where $\mathbf{M}_{\mathsf{Att}}$ is the matrix formed by the rows of $\mathbf{M}$ associated with the attributes in Att via the mapping ρ and $\boldsymbol{u}_{\mathsf{Att}}$ is the components of $\boldsymbol{u}$ associated with Att.

At a very high level, our slotted registered ABIPFE is a combination of the registered ABE of [HLWW23] and our registered IPFE discussed above. Combining the primitives ABE and IPFE, even in the non-registration-based setting, in a completely generic way might be insecure [ACGU20a, PD21b] since the ABE adversary is not allowed to query any secret key that decrypts the challenge ciphertext. However, this is crucial for the security of ABIPFE. Our approach aims to blend the *attribute-aggregation* procedure devised for the registered ABE of [HLWW23] with the *function-aggregation* technique developed in this work for our registered IPFE. The aggregated master public key consists of two aggregated components—one for attributes and another for function vectors—which are randomized using a newly sampled group element during encryption. This additional randomization adds an extra layer of security to the encryption process, making it (computationally) difficult for *unauthorized* users to gain access to the inner product values, even if they possess secret keys.

Let $\mathcal{U}_{\mathsf{att}}$ be the universe of attributes. Then, for each attribute $w \in \mathcal{U}_{\mathsf{att}}$ and slot $i \in [L]$, the setup randomly samples $t_{i,w} \leftarrow \mathbb{Z}_p$ and adds the additional elements $\{g_1^{1/\pi}, \{g_1^{\beta_i t_{i,w}}\}_{i,w}, \{g_2^{\pi\beta_i t_{i,w}/\beta_j}\}_{i\neq j}\}$ to the crs (given is Eq. 1) of the registered IPFE. The users can sample their individual key pairs similar to our registered IPFE. At the aggregation step, each user submits a pair $(\boldsymbol{y}_i, \mathsf{Att}_i)$ comprising of a vector and an attribute set along with its public key pk_i. The aggregation algorithm follows exactly the same way as in IPFE except it adds new components: $\{T_w = \prod_{i\in[L], w\notin\mathsf{Att}_i} g_1^{\beta_i t_{i,w}}\}_w$ to MPK and $\prod_{j\neq i: w\notin\mathsf{Att}_j} g_2^{\pi\beta_j t_{j,w}/\beta_i}$ to hsk_i of Eq. 2. Therefore, the sizes of MPK and hsk_i both are bounded by $\mathsf{poly}(\lambda, |\mathcal{U}_{\mathsf{att}}|, n, \log L)$, meeting the efficiency requirement of a slotted registered FE scheme. A ciphertext encrypting a vector $\boldsymbol{x}$ under a policy $(\mathbf{M}, \rho)$ is computed as follows. Our idea is to randomize the ciphertext component W^s of IPFE with a random element $h \leftarrow \mathbb{G}_1$ as $h^s \cdot W^s$. At the time of decryption, it eventually produces an additional masking factor $e(h, g_2)^{s/\beta_i}$ which can only be cancelled using a secret key sk_i that corresponds to Att_i satisfying the policy $(\mathbf{M}, \rho)$. More specifically, we first sample a random vector $\boldsymbol{v} = (s, v_2, \ldots, v_N)$ and then set the ciphertext

$$\mathsf{ct} = (\ g_T^s, g_T^{s\boldsymbol{\alpha}+\boldsymbol{x}}, g_1^{-s/\gamma}, g_1^{-s/\pi}, h^s W^s, h^{\langle \boldsymbol{v}, \boldsymbol{m}_k\rangle} T^s_{\rho(k)}\ )$$

where $\boldsymbol{m}_k$ denotes the k-th row of $\mathbf{M}$. To decrypt the ciphertext the i-th user first computes a slot-specific component $e(h, g_2)^{s/\beta_i} \cdot e(g_1, g_2)^{s\langle\boldsymbol{\alpha}, \boldsymbol{y}_i\rangle}$ using the ciphertext components $h^s W^s, g_1^{-s/\gamma}$, secret key sk_i and a component (same as the i-th helper decryption key of IPFE shown in Eq. 2) of hsk_i. Next, assuming that Att_i satisfies the policy, the user reconstruct the secret s in the form of $e(h, g_2)^{s/\beta_i}$ via pairing the ciphertext components $h^{\langle \boldsymbol{v}, \boldsymbol{m}_k\rangle} T^s_{\rho(k)}, g_1^{-s/\pi}$ with g_2^{1/β_i} and the newly added helper decryption key component $\prod_{j\neq i: w\notin\mathsf{Att}_j} g_2^{\pi\beta_j t_{j,w}/\beta_i}$ respectively. In this step, we avail a *cross-terms cancellation* approach similar to [HLWW23]. Finally, the user recovers the inner product value $\langle \boldsymbol{x}, \boldsymbol{y}_i\rangle$ from $g_T^{s\langle\boldsymbol{\alpha}, \boldsymbol{y}_i\rangle + \langle \boldsymbol{x}, \boldsymbol{y}_i\rangle}$ by unmasking it using the term $g_T^{s\langle\boldsymbol{\alpha}, \boldsymbol{y}_i\rangle}$. To prove the generic security of the scheme, we show that the *only* way for an adversary to recover the masking term $g_T^{s\langle\boldsymbol{\alpha}, \boldsymbol{y}_i\rangle}$ is to make use of a secret key sk_i which corresponds to an attribute set satisfying the challenge policy. We refer to Section 7 for a formal description of the construction and analysis of the slotted registered ABIPFE.

2.3 Registered FE for Polynomial-size Circuits

In this work, we build a registered FE for all polynomial-size circuits from indistinguishable obfuscation and one-way functions. While our pairing-based registered FEs for specific functionalities could only support a bounded number of users, the registered FE for general functions allows an *arbitrary* number of users to join the system. Our registered FE for all circuits generalizes the IO-based registered ABE of Hohenberger et al. [HLWW23] that provides access control using any arbitrary circuit predicates. In particular, it is based

on IO [BGI+01, GGH+13] and somewhere statistically binding hash functions (SSB) [HW15, OPWW15]. An overview of the slotted registered FE is as follows. The CRS is an SSB hash key hk. In the key generation phase, each user samples a seed s_i and sets the public key as $\mathsf{pk}_i = \mathsf{PRG}(s_i)$ where PRG is a pseudorandom generator. To register a set of L users, the key curator hashes the list of public key-function pairs $\{(\mathsf{pk}_i, f_i)\}_i$ using hk and sets the hash value h to be the master public key MPK. Additionally, it computes an SSB opening π_i for each slot index i, which serves as the helper decryption key hsk_i of the user. The ciphertext of the slotted scheme consists of a ciphertext CT encrypting the message m under a freshly sampled (symmetric) encryption key SK and an obfuscated circuit which is consistent with MPK and SK. The circuit first verifies (a) the opening π using MPK and (b) the public key pk_i by re-computing $\mathsf{PRG}(\mathsf{sk}_i)$, and if the checks pass then it outputs the message m by decrypting the ciphertext CT using SK. The correctness is immediate by the definition of the obfuscated circuit. The compactness of MPK and hsk_i follows from the *succinctness* of SSB. Since the CRS (or the hash key) size scales with $O(\log L)$, our slotted registered FE can be transformed into a registered FE supporting any arbitrary number of users. We give a detailed description of the scheme in Section 8.

3 Preliminaries

Notations: Throughout this paper, we use λ as the security parameter. Let $n, m \in \mathbb{Z}$ be two non-negative integers. Then $[n]$ denotes the set $\{1, 2, \ldots, n\}$ if $n > 0$ and $[n, m]$ denotes the set $\{n, n+1, \ldots, m\}$. We use the bold uppercase letters (e.g. $\mathbf{M}$) to denote matrices and the bold lowercase letters (e.g. $\boldsymbol{x}$) to denote vectors. The components of the vectors are denoted by non-boldface letters (e.g. $\boldsymbol{x} = (x_1, \ldots, x_n)$). We write $\mathsf{poly}(\lambda)$ as a polynomial function of λ if it is of the form $O(\lambda^c)$ for some constant $c \in \mathbb{N}$. We say a function $\mathsf{negl}(\lambda)$ is negligible function of λ if it is of the form $O(\lambda^{-c})$ for all $c > 0$.

Bilinear groups: Assume a bilinear group generator algorithm GG that takes as input 1^λ and outputs a tuple $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, e)$, where $\mathbb{G}_1, \mathbb{G}_2$ are the source groups and $\mathbb{G}_T$ is the target group of the same prime order $p = p(\lambda)$ with generators g_1, g_2 respectively. The map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ satisfies *non-degeneracy*, meaning that $e(g_1, g_2) = g_T$ generates $\mathbb{G}_T$. It also satisfies bilinearity, *i.e.*, for all $a, b \in \mathbb{Z}_p$ it holds that $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$. We require that the group operations and the bilinear map are efficiently computable.

3.1 Generic Bilinear Group Model

In this work, we prove the security of our pairing-based schemes in the generic bilinear group model. We recall the notations and definitions of generic bilinear group model adapted from [BCFG17, AY20]. In the generic bilinear group model, the adversary only receives *handles* of the group elements instead of the actual group elements. The adversary is also given a stateful oracle to perform some specific group and pairing operations on already queried group elements via the received handles. Note that, the adversary gets only the handles of the newly computed group elements. In order to perform the group operations, the generic bilinear group model maintains an internal mapping between the handles and their corresponding group elements. The handles are not unique meaning that the same group elements may appear more than once in a list under different handles. The motivation of proving the security of a group-based cryptographic scheme in the generic bilinear group model is to ensure that an adversary that applies group operations in a black-box manner can not break the security of the scheme. In the definition below, we consider the list that maintains the list of the group exponents rather than elements as in [BCFG17, AY20], since the specific representation of the group element is irrelevant in this model.

Definition 1 (Generic Bilinear Group Oracle) Let $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, e)$ be a bilinear group and L_1, L_2, L_T be the lists of exponents of group elements in $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ respectively. The challenger initializes L_1, L_2 and L_T according to a distribution $\mathcal{D}$ and the adversary receives handles for the elements in the lists. For $s \in \{1, 2, T\}$, $L_s[h]$ denotes the h-th element in the list L_s. The handle to this element is simply the pair (s, h). The adversary is provided with the following oracles:

- $\mathsf{add}(s, h_1, h_2)$: The adversary submits a tuple (s, h_1, h_2) such that $(s, h_1), (s, h_2)$ represent handles to elements in the list L_s for $s \in \{1, 2, T\}$. The challenger appends $L_s[h_1] + L_s[h_2]$ to L_s and returns its handle $(s, |L_s|)$.
- $\mathsf{neg}(s, h)$: The adversary submits a handle (s, h) for $s \in \{1, 2, T\}$. The challenger appends $-L_s[h]$ to L_s and returns its handle $(s, |L_s|)$.
- $\mathsf{map}(h_1, h_2)$: The adversary submits a tuple (h_1, h_2) such that $(1, h_1)$ and $(2, h_2)$ represent handles of elements in the lists L_1 and L_2 respectively. The challenger appends $L_1[h_1] \cdot L_2[h_2]$
- $\mathsf{zero\text{-}test}(h)$: The adversary submits h such that (T, h) represents a handle of an element in the lists L_T. The challenger returns 1 if $L_T[h] = 0$; and 0 otherwise.

Symbolic group model: The symbolic group model for a bilinear group $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, e)$ and a distribution $\mathcal{D}_p$ gives to the adversary the same interface as the corresponding generic bilinear group model, except that internally the challenger stores lists of element in the field $\mathbb{Z}_p(X_1, \ldots, X_n)$ instead of lists of group elements. The oracles $\mathsf{add}, \mathsf{neg}, \mathsf{map}$ and $\mathsf{zero\text{-}test}$ computes addition, negation, multiplication and equality in the field. In this work, we will use the subring $\mathbb{Z}_p[X_1, \ldots, X_n, 1/X_1, \ldots, 1/X_n]$ whose elements can be represented as

$$f(X_1, \ldots, X_n) = \sum_{(c_1, \ldots, c_n) \in \mathbb{Z}^n} a_{c_1, \ldots, c_n} X_1^{c_1} \cdots, X_n^{c_n}$$

using the coefficients $\{a_{c_1, \ldots, c_n} \in \mathbb{Z}_p\}_{(c_1, \ldots, c_n) \in \mathbb{Z}^n}$, where we have $a_{c_1, \ldots, c_n} = 0$ for all but finite $(c_1, \ldots, c_n) \in \mathbb{Z}^n$. This expression of f is unique.

We will require the Schwatrz-Zipple Lemma [Sho97b, Zip79] stated as follows.

Lemma 1 ([Sho97b, Zip79]) Fix a prime p and let $f \in \mathbb{Z}_p[X_1, \ldots, X_n]$ be an n-variate polynomial with degree at most d and which is not identical to zero. Then,

$$\Pr[f(x_1, \ldots, x_n) = 0 : x_1, \ldots, x_n \leftarrow \mathbb{Z}_p] \leq d/p.$$

4 Registered Functional Encryption

In this section, we introduce the notion of registered FE for general class of functions. We generalize the registration-based ABE notion of [HLWW23] into the setting of FE which goes beyond the all-or-nothing type paradigm.

Definition 2 (Registered Functional Encryption) Let $\mathcal{U}_F = \{\mathcal{F}_\lambda\}_{\lambda \in \mathbb{N}}$ be the universe of functions and $\mathcal{M}$ be the set of messages. A registered functional encryption scheme with function universe $\mathcal{U}_F$ and message space $\mathcal{M}$ is a tuple of efficient algorithms $\mathsf{RFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{RegPK}, \mathsf{Enc}, \mathsf{Update}, \mathsf{Dec})$ that work as follows:

$\mathsf{Setup}(1^\lambda, 1^{\ell_f}) \to \mathsf{crs}$: The setup algorithm takes the security parameter λ, the (maximum) size ℓ_f of the functions in $\mathcal{U}_F$ as inputs and outputs a common reference string crs.

$\mathsf{KeyGen}(\mathsf{crs}, \mathsf{aux}) \to (\mathsf{pk}, \mathsf{sk})$: The key generation algorithm takes the common reference string crs, and a (possibly empty) state aux as inputs and outputs a public key pk and a secret key sk.

$\mathsf{RegPK}(\mathsf{crs}, \mathsf{aux}, \mathsf{pk}, f_{\mathsf{pk}}) \to (\mathsf{MPK}, \mathsf{aux}')$: The registration algorithm takes a common reference string crs, a (possibly empty) state aux, a public key pk and a function $f_{\mathsf{pk}} \in \mathcal{F}_\lambda$ as inputs and outputs a master public key MPK and an updated state aux'. This is a *deterministic* algorithm.

$\mathsf{Enc}(\mathsf{MPK}, m) \to \mathsf{ct}$: The encryption algorithm takes a master public key MPK and a message $m \in \mathcal{M}$ as inputs and outputs a ciphertext ct.

Update$(\mathsf{crs}, \mathsf{aux}, \mathsf{pk}) \to \mathsf{hsk}$: The update algorithm takes a common reference string crs, a state aux and a public key pk as inputs, and outputs a helper decryption keys hsk. This is a *deterministic* algorithm.

Dec$(\mathsf{sk}, \mathsf{hsk}, \mathsf{ct}) \in \mathcal{M} \cup \{\mathsf{GetUpdate}, \perp\}$: The decryption algorithm takes a secret key sk, a helper decryption key hsk and ciphertext ct as inputs. The algorithm either outputs a message $m' \in \mathcal{M}$, a special symbol $\perp$ indicating decryption failure, or a special message $\mathsf{GetUpdate}$ indicating an updated helper decryption key is needed to decrypt the ciphertext. This is a *deterministic* algorithm.

The algorithms must satisfy the following properties:

Correctness, Compactness and Update efficiency: For all security parameters $\lambda \in \mathbb{N}$, all messages $m \in \mathcal{M}$, all functions $f \in \mathcal{F}_\lambda$, we define the following experiment between an adversary $\mathcal{A}$ and a challenger:

- **Setup phase:** The challenger starts by sampling the common reference string $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^{\ell_f})$. It then initializes the auxiliary input $\mathsf{aux} \leftarrow \perp$ and initial master public key $\mathsf{MPK}_0 \leftarrow \perp$. It also initializes a counter $\mathsf{ctr}[\mathsf{reg}] \leftarrow 0$ to keep track of the number of registration queries and another counter $\mathsf{ctr}[\mathsf{enc}] \leftarrow 0$ to keep track of the number of encryption queries. Finally, it initializes $\mathsf{ctr}[\mathsf{reg}]^* \leftarrow \infty$ as the index for the target key. It gives crs to $\mathcal{A}$.
- **Query phase:** During the query phase, the adversary $\mathcal{A}$ is able to make the following queries:
  - **Register non-target key query:** In a non-target-key registration query, the adversary $\mathcal{A}$ specifies a public key pk and a function $f \in \mathcal{U}_F$. The challenger first increments the counter $\mathsf{ctr}[\mathsf{reg}] \leftarrow \mathsf{ctr}[\mathsf{reg}] + 1$ and then registers the key by computing $(\mathsf{MPK}_{\mathsf{ctr}[\mathsf{reg}],\mathsf{aux}'}) \leftarrow \mathsf{RegPK}(\mathsf{crs}, \mathsf{aux}, \mathsf{pk}, f)$. The challenger updates its auxiliary data by setting $\mathsf{aux} \leftarrow \mathsf{aux}'$ and replies $\mathcal{A}$ with $(\mathsf{ctr}[\mathsf{reg}], \mathsf{MPK}_{\mathsf{ctr}[\mathsf{reg}]}, \mathsf{aux})$.
  - **Register target key query:** In a target-key registration query, the adversary specifies a function $f^* \in \mathcal{U}_F$. If the adversary has previously made a target-key registration query, then the challenger replies with $\perp$. Otherwise, the challenger increments the counter $\mathsf{ctr}[\mathsf{reg}] \leftarrow \mathsf{ctr}[\mathsf{reg}] + 1$, samples $(\mathsf{pk}^*, \mathsf{sk}^*) \leftarrow \mathsf{KeyGen}(1^\lambda, \mathsf{aux})$, and registers $(\mathsf{MPK}_{\mathsf{ctr}[\mathsf{reg}],\mathsf{aux}'}) \leftarrow \mathsf{RegPK}(\mathsf{crs}, \mathsf{aux}, \mathsf{pk}^*, f^*)$. It computes the helper decryption key $\mathsf{hsk}^* \leftarrow \mathsf{Update}(\mathsf{crs}, \mathsf{aux}, \mathsf{pk}^*)$. The challenger updates its auxiliary data by setting $\mathsf{aux} \leftarrow \mathsf{aux}'$, stores the index of the target identity $\mathsf{ctr}[\mathsf{reg}]^* \leftarrow \mathsf{ctr}[\mathsf{reg}]$, and replies to $\mathcal{A}$ with $(\mathsf{ctr}[\mathsf{reg}], \mathsf{MPK}_{\mathsf{ctr}[\mathsf{reg}]}, \mathsf{aux}, \mathsf{pk}^*, \mathsf{hsk}^*, \mathsf{sk}^*)$.
  - **Encryption query:** In an encryption query, the adversary submits the index $\mathsf{ctr}[\mathsf{reg}]^* \leq i \leq \mathsf{ctr}[\mathsf{reg}]$ of a public key[6] and a message $m_{\mathsf{ctr}[\mathsf{enc}]} \in \mathcal{M}$. If the adversary has not yet registered a target key the challenger replies with $\perp$. Otherwise, the challenger increments the counter $\mathsf{ctr}[\mathsf{enc}] \leftarrow \mathsf{ctr}[\mathsf{enc}] + 1$ and computes $\mathsf{ct}_{\mathsf{ctr}[\mathsf{enc}]} \leftarrow \mathsf{Enc}(\mathsf{MPK}_i, m)$. The challenger replies to $\mathcal{A}$ with $(\mathsf{ctr}[\mathsf{enc}], \mathsf{ct}_{\mathsf{ctr}[\mathsf{enc}]})$.
  - **Decryption query:** In a decryption query, the adversary submits a ciphertext index $1 \leq j \leq \mathsf{ctr}[\mathsf{enc}]$. The challenger computes $m'_j \leftarrow \mathsf{Dec}(\mathsf{sk}^*, \mathsf{hsk}^*, \mathsf{ct}_j)$. If $m'_j = \mathsf{GetUpdate}$, then the challenger computes an updated helper decryption key $\mathsf{hsk}^* \leftarrow \mathsf{Update}(\mathsf{crs}, \mathsf{aux}, \mathsf{pk}^*)$ and recomputes $m'_j \leftarrow \mathsf{Dec}(\mathsf{sk}^*, \mathsf{hsk}^*, \mathsf{ct}_j)$. If $m'_j \neq f^*(m_j)$, the experiment halts with outputs $b = 1$.

  If $\mathcal{A}$ has finished making queries and the experiment has not halted (as a result of a decryption query), then the experiment outputs $b = 0$.

We say that RFE is correct, compact and update efficient if for all adversaries $\mathcal{A}$ making at most polynomial number of queries, the following properties hold:

- **Correctness:** There exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $\Pr[b = 1] = \mathsf{negl}(\lambda)$ in the above experiment. We say that the scheme satisfies *perfect correctness* if $\Pr[b = 1] = 0$.

[6]The message is encrypted under a master public key which is registered only after the adversary registers a target key since we require the correctness to hold only for the target key.

- **Compactness:** Let N be the number of registration queries the adversary makes in the above experiment. There exists a universal polynomial $\mathsf{poly}(\cdot,\cdot,\cdot)$ such that for $i \in [N]$, $|\mathsf{MPK}_i| = \mathsf{poly}(\lambda, \ell_f, \log i)$. We also require that the size of the helper decryption key hsk^* satisfy $\mathsf{hsk}^* = \mathsf{poly}(\lambda, \ell_f, \log N)$ (at *all* point of the experiment).
- **Update efficiency:** Let N be the number of registration queries made by $\mathcal{A}$. Then, in the course of the above experiment, the challenger invokes the update algorithm Update at most $O(\log N)$ times where each invocation runs in $\mathsf{poly}(\log N)$ time in the RAM model of computation. Specially, we model Update as a RAM program that has *random* access to its input; thus, the running time of Update in the RAM model can be *smaller* than the input length.

Security: Let $\mathsf{coin} \in \{0,1\}$ be a bit. We define the following security experiment $\mathsf{Expt}^{\mathsf{RFE}}_{\mathcal{A}}(1^\lambda, \mathsf{coin})$ played between an adversary $\mathcal{A}$ and a challenger.

- **Setup phase:** The challenger samples a common reference string $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^{\ell_f})$. It then initializes the auxiliary input $\mathsf{aux} \leftarrow \bot$, the initial master public key $\mathsf{MPK} \leftarrow \bot$, a counter $\mathsf{ctr} \leftarrow 0$ for the number of honest-key-registration queries the adversary has made, an empty set of keys $\mathsf{Cor} \leftarrow \emptyset$ for tracking the honestly generated keys that are corrupted in course of the experiment, an empty set of keys $\mathsf{Mal} \leftarrow \emptyset$ which will be filled with the keys generated by the adversary and an empty dictionary $\mathsf{D} \leftarrow \emptyset$ mapping public keys to registered function. For notational convenience, if $\mathsf{pk} \notin \mathsf{D}$, then we define $\mathsf{D}[\mathsf{pk}] \coloneqq \emptyset$. The challenger gives the crs to $\mathcal{A}$.
- **Query phase:** The adversary $\mathcal{A}$ is allowed to query the following queries:
  - **Registered malicious key query:** In a corrupted key query, $\mathcal{A}$ specifies a public key pk and a function $f \in \mathcal{U}_F$. The challenger registers the key by computing $(\mathsf{MPK}', \mathsf{aux}') \leftarrow \mathsf{RegPK}(\mathsf{crs}, \mathsf{aux}, \mathsf{pk}, f)$. The challenger updates its copy of the public key $\mathsf{MPK} \leftarrow \mathsf{MPK}'$, its auxiliary data $\mathsf{aux} \leftarrow \mathsf{aux}'$, adds pk to Mal, and updates $\mathsf{D}[\mathsf{pk}] \leftarrow \mathsf{D}[\mathsf{pk}] \cup \{f\}$. It replies to $\mathcal{A}$ with $(\mathsf{MPK}', \mathsf{aux}')$.
  - **Registered honest key query:** In an honest key query, $\mathcal{A}$ specifies a function $f \in \mathcal{U}_F$. The challenger increments $\mathsf{ctr} \leftarrow \mathsf{ctr} + 1$ and samples $(\mathsf{pk}_{\mathsf{ctr}}, \mathsf{sk}_{\mathsf{ctr}}) \leftarrow \mathsf{KeyGen}(\mathsf{crs}, \mathsf{aux})$, and registers the key by computing $(\mathsf{MPK}', \mathsf{aux}') \leftarrow \mathsf{RegPK}(\mathsf{crs}, \mathsf{aux}, \mathsf{pk}_{\mathsf{ctr}}, f)$. The challenger updates its public key $\mathsf{MPK} \leftarrow \mathsf{MPK}'$, its auxiliary data $\mathsf{aux} \leftarrow \mathsf{aux}'$, adds $\mathsf{D}[\mathsf{pk}_{\mathsf{ctr}}] \leftarrow \mathsf{D}[\mathsf{pk}_{\mathsf{ctr}}] \cup \{f\}$. It replies to $\mathcal{A}$ with $(\mathsf{ctr}, \mathsf{MPK}', \mathsf{aux}', \mathsf{pk}_{\mathsf{ctr}})$.
  - **Corrupt honest key query:** In a corrupt-honest key query, $\mathcal{A}$ specifies an index $1 \le i \le \mathsf{ctr}$. Let $(\mathsf{pk}_i, \mathsf{sk}_i)$ be the i-th public/secret key the challenger samples when responding to the i-th honest-key-registration query. The challenger adds pk_i to Cor and replies to $\mathcal{A}$ with sk_i.
- **Challenge phase:** The adversary $\mathcal{A}$ chooses two messages $m_0^*, m_1^* \in \mathcal{M}$. The challenger replies with the challenge ciphertext $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{MPK}, m^*_{\mathsf{coin}})$.
- **Output phase:** At the end of the experiment, $\mathcal{A}$ outputs a bit $\mathsf{coin}' \in \{0,1\}$, which is the output of the experiment.

Let $\mathcal{S} = \{f_{\mathsf{pk}} \in \mathsf{D}[\mathsf{pk}] : \mathsf{pk} \in \mathsf{Cor} \cup \mathsf{Mal}\}$. We say an adversary $\mathcal{A}$ is admissible if for all functions $f_{\mathsf{pk}} \in \mathcal{S}$, it holds that $f_{\mathsf{pk}}(m_0^*) = f_{\mathsf{pk}}(m_1^*)$. The registration-based functional encryption scheme RFE is said to be secure if for all admissible adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

$$|\Pr[\mathsf{Expt}^{\mathsf{RFE}}_{\mathcal{A}}(1^\lambda, 0) = 1] - \Pr[\mathsf{Expt}^{\mathsf{RFE}}_{\mathcal{A}}(1^\lambda, 1) = 1]| = \mathsf{negl}(\lambda).$$

Definition 3 (Bounded Registered FE) We say that a registered FE scheme RFE is bounded if there is an *a-priori* bound on the number of registered users in the system. In a bounded RFE, the setup additionally takes a bound parameter 1^L which specifies the maximum number of registered users that can be joined to the system. Similarly, in the correctness and security definition, the adversary is asked to submit the bound 1^L at the beginning and it is restricted to query up to L queries.

Specific function classes of RFE: In this work, we construct RFE schemes for general (polynomial-size) functions from obfuscation as well as bounded RFE schemes for specific function classes from pairings. We consider the following class of registered FEs:

- *Registered Inner Product FE.* The inner product FE or IPFE [ABDP15, ALS16] is a specific class of FE which only allows linear computation over the encrypted data. The function space $\mathcal{U}_F$ and the message space $\mathcal{M}$ are vectors from the set $\mathbb{Z}^n$ for an integer $n \in \mathbb{N}$. In particular, a user registers the public key pk along with a function $f_{\mathsf{pk}} = \boldsymbol{y} \in \mathbb{Z}^n$ and a message $m = \boldsymbol{x} \in \mathbb{Z}^n$ is encrypted using the master public key. During decryption a user recovers the inner product $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ between the vectors. As in all existing pairing-based IPFE schemes of the literature, our registered IPFE scheme also requires that the inner product value to lie in a polynomial range for efficient extraction of it from the exponent of the target group.
- *Registered Attribute-Based Inner Product FE.* We consider the attribute-based IPFE or ABIPFE [ACGU20a] which provides attribute-based access control over IPFE. The secret key and message vectors are additionally associated with an attribute set $\mathsf{Att} \subseteq \mathcal{U}_{\mathsf{att}}$ and a policy $P \in \mathcal{P}$ where $\mathcal{U}_{\mathsf{att}}$ and $\mathcal{P}$ are attribute universe and a set of supported policies respectively, and the recovery of the inner product during decryption depends on whether the attribute set is satisfying the policy. In our registration-based setting, a user registers a public key pk with a function $f_{\mathsf{pk}} = (\mathsf{Att}, \boldsymbol{y}) \in \mathsf{PSet}(\mathcal{U}_{\mathsf{att}}) \times \mathbb{Z}^n (= \mathcal{U}_F)$[7] and the encryption of the message $m = (P, \boldsymbol{x}) \in \mathcal{P} \times \mathbb{Z}^n$ yields a ciphertext where P is made available with it in the clear. The decryption procedure computes $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ (also belonging to a polynomial range) using the secret key sk of the user if the associated attribute set Att satisfies the policy, *i.e.*, if $P(\mathsf{Att}) = 1$ holds.

5 Slotted Registered Functional Encryption

In this section, we introduce the notion of slotted registered FE which is the core building block for building the full-fledged registered FE scheme. We discuss the transformation from slotted registered FE to registered FE in Section 9.

Definition 4 (Slotted Registered Functional Encryption) Let $\mathcal{U}_F = \{\mathcal{F}_\lambda\}_{\lambda \in \mathbb{N}}$ be the universe of functions and $\mathcal{M}$ be the set of messages. A slotted registered functional encryption scheme with function universe $\mathcal{U}_F$ and message space $\mathcal{M}$ is a tuple of efficient algorithms $\mathsf{SlotRFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{IsValid}, \mathsf{Aggregate}, \mathsf{Enc}, \mathsf{Dec})$ that work as follows:

Setup$(1^\lambda, 1^{|\mathcal{U}_F|}, 1^L) \to \mathsf{crs}$: The setup algorithm takes the security parameter λ, the (maximum) size $|\mathcal{U}_F|$ of the functions in $\mathcal{U}_F$ and the number of slots L (in unary) as inputs and outputs a common reference string crs.

KeyGen$(\mathsf{crs}, i) \to (\mathsf{pk}_i, \mathsf{sk}_i)$: The key generation algorithm takes the common reference string crs, and a slot index $i \in [L]$ as inputs and outputs a public key pk_i and a secret key sk_i for the slot i.

IsValid$(\mathsf{crs}, i, \mathsf{pk}_i) \in \{0, 1\}$: The key-validation algorithm takes a common reference string crs, a slot index $i \in [L]$ and a public key pk_i as inputs and outputs a bit $b \in \{0, 1\}$. This is a *deterministic* algorithm.

Aggregate$(\mathsf{crs}, (\mathsf{pk}_1, f_1), \ldots, (\mathsf{pk}_L, f_L)) \to (\mathsf{MPK}, \mathsf{hsk}_1, \ldots, \mathsf{hsk}_L)$: The aggregate algorithm takes a common reference string crs, a list of L public key-function pairs $(\mathsf{pk}_1, f_1), \ldots, (\mathsf{pk}_L, f_L)$ as inputs such that $f_i \in \mathcal{F}_\lambda$ for all $i \in [L]$ and outputs a master public key MPK and a collection of helper decryption keys $\mathsf{hsk}_1, \ldots, \mathsf{hsk}_L$. We assume that the master public key is implicitly provided to the users along with their helper decryption keys. This is a *deterministic* algorithm.

Enc$(\mathsf{MPK}, m) \to \mathsf{ct}$: The encryption algorithm takes a master public key MPK and a message $m \in \mathcal{M}$ as inputs and outputs a ciphertext ct.

[7] Here, $\mathsf{PSet}(X)$ denotes the power set of the set X.

$\mathbf{Dec}(\mathsf{sk}, \mathsf{hsk}, \mathsf{ct}) \in \mathcal{M} \cup \{\perp\}$: The decryption algorithm takes a secret key sk, a helper decryption key hsk and ciphertext ct as inputs and outputs a message m'. This is a *deterministic* algorithm.

The algorithms must satisfy the following properties:

Completeness: For all $\lambda \in \mathbb{N}$, all property universes $\mathcal{U}_P$, and all indices $i \in [L]$,

$$\Pr\left[\mathsf{IsValid}(\mathsf{crs}, i, \mathsf{pk}_i) = 1 : \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^{|\mathcal{U}_F|}, 1^L); (\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{crs}, i)\right] = 1.$$

Correctness: The $\mathsf{SlotRFE}$ is said to be correct if for all security parameters $\lambda \in \mathbb{N}$, all possible lengths $L \in \mathbb{N}$, all indices $i \in [L]$, if we sample $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^{|\mathcal{U}_F|, 1^L})$, $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{crs}, i)$ and for all collections of public keys $\{\mathsf{pk}_j\}_{j \neq i}$ (which may be correlated to pk_i) where $\mathsf{IsValid}(\mathsf{crs}, j, \mathsf{pk}_j) = 1$, all messages $m \in \mathcal{M}$, all functions $f \in \mathcal{F}_\lambda$, the following holds

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}_i, \mathsf{hsk}_i, \mathsf{ct}) = f(m) : \begin{array}{c}(\mathsf{MPK}, \mathsf{hsk}_1, \ldots, \mathsf{hsk}_L) \leftarrow \mathsf{Aggregate}(\mathsf{MPK}, (\mathsf{pk}_1, f_1), \ldots, (\mathsf{pk}_L, f_L)); \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{MPK}, m)\end{array}\right] = 1.$$

Compactness: The $\mathsf{SlotRFE}$ is said to be compact if there exists a universal polynomial $\mathsf{poly}(\cdot,\cdot,\cdot)$ such that the length of the master public key and individual helper secret keys output by $\mathsf{Aggregate}$ are bounded by $\mathsf{poly}(\lambda, |\mathcal{U}_F|, \log L)$.

Security: Let $\mathsf{coin} \in \{0,1\}$ be a bit. We define the following security experiment $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{SlotRFE}}(1^\lambda, b)$ played between an adversary $\mathcal{A}$ and a challenger.

- **Setup phase:** The adversary sends a slot count 1^L to the challenger. The challenger samples $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^{|\mathcal{U}_F|}, 1^L)$ and sends crs to $\mathcal{A}$. The challenger initializes a counter $\mathsf{ctr} \leftarrow 0$, a dictionary D and a set of corrupted indices $\mathsf{Cor} \leftarrow \emptyset$ and a set of malicious indices $\mathsf{Mal} \leftarrow \emptyset$.
- **Pre-challenge query phase:** The adversary $\mathcal{A}$ is allowed to query the following queries:
  - **Key-generation query:** In a key-generation query, $\mathcal{A}$ specifies a slot index $i \in [L]$. The challenger samples $(\mathsf{pk}_{\mathsf{ctr}}, \mathsf{sk}_{\mathsf{ctr}}) \leftarrow \mathsf{KeyGen}(\mathsf{crs}, i)$ and increments $\mathsf{ctr} \leftarrow \mathsf{ctr} + 1$. Then, it sends $(\mathsf{ctr}, \mathsf{pk}_{\mathsf{ctr}})$ to $\mathcal{A}$. The challenger adds the mapping $\mathsf{ctr} \mapsto (i, \mathsf{pk}_{\mathsf{ctr}}, \mathsf{sk}_{\mathsf{ctr}})$ to the dictionary D.
  - **Corruption query:** In a corruption query, $\mathcal{A}$ specifies an index $1 \leq c \leq \mathsf{ctr}$. The challenger looks up the tuple $(i', \mathsf{pk}', \mathsf{sk}') \leftarrow \mathsf{D}[c]$ and sends sk' to $\mathcal{A}$.
- **Challenge phase:** For each slot $i \in [L]$, $\mathcal{A}$ specifies a tuple (c_i, pk_i^*) where either $c_i \in \{1, \ldots, \mathsf{ctr}\}$ to reference a challenger-generated key or $c_i = \perp$ to reference a key outside this set. $\mathcal{A}$ also specifies two challenge messages m_0^*, m_1^*. The challenger does the following:
  - If $c_i \in \{1, \ldots, \mathsf{ctr}\}$, then the challenger looks up the entry $\mathsf{D}[c_i] = (i', \mathsf{pk}', \mathsf{sk}')$. If $i = i'$, then the challenger sets $\mathsf{pk}_i \leftarrow \mathsf{pk}'$. Moreover, if $\mathcal{A}$ previously issues a *corruption query* on the index c_i, then the challenger adds the slot index i to Cor. Otherwise, if $i \neq i'$, then the experiment halts.
  - If $c_i = \perp$, then the challenger checks $\mathsf{IsValid}(\mathsf{crs}, i, \mathsf{pk}_i^*) = 1$. If not, the experiment halts. If the key is valid, the challenger sets $\mathsf{pk}_i \leftarrow \mathsf{pk}_i^*$ and adds the slot index i to Mal.

  The challenger computes $(\mathsf{MPK}, \mathsf{hsk}_1, \ldots, \mathsf{hsk}_L) \leftarrow \mathsf{Aggregate}(\mathsf{MPK}, (\mathsf{pk}_1, f_1), \ldots, (\mathsf{pk}_L, f_L)$ and then $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{MPK}, m^*_{\mathsf{coin}})$. Finally, it sends ct^* to $\mathcal{A}$. Note that, there is no need to additionally provide $(\mathsf{MPK}, \mathsf{hsk}_1, \ldots, \mathsf{hsk}_L)$ to $\mathcal{A}$ since $\mathsf{Aggregate}$ is a deterministic algorithm. Similarly, there is no advantage of allowing $\mathcal{A}$ to select the challenge messages after seeing the aggregated key.
- **Post-challenge query phase:** The adversary $\mathcal{A}$ is allowed to query the following queries:

– In a corruption query, $\mathcal{A}$ specifies a slot index $c \in \{1, \ldots, \mathsf{ctr}\}$. The challenger picks the tuple $(i', \mathsf{pk}', \mathsf{sk}') \leftarrow \mathsf{D}[c]$ and sends sk' to $\mathcal{A}$. Moreover, if $\mathcal{A}$ registered a tuple of the form (c, pk^*) in the challenge phase for some choice of pk^*, then the challenger adds the slot index $i' \in [L]$ to Cor.

- **Output phase:** At the end of the experiment, $\mathcal{A}$ outputs a bit $\mathsf{coin}' \in \{0, 1\}$, which is the output of the experiment.

We say an adversary $\mathcal{A}$ is admissible if for all corrupted slot indices $i \in \mathsf{Cor} \cup \mathsf{Mal}$, it holds that $f(m_0^*) = f(m_1^*)$. The slotted registration-based encryption scheme $\mathsf{SlotRFE}$ is said to be secure if for all polynomials $L = L(\lambda)$ and all efficient and admissible adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

$$|\Pr[\mathsf{Expt}_{\mathcal{A}}^{\mathsf{SlotRFE}}(1^\lambda, 0) = 1] - \Pr[\mathsf{Expt}_{\mathcal{A}}^{\mathsf{SlotRFE}}(1^\lambda, 1) = 1]| = \mathsf{negl}(\lambda).$$

Remark 1 (On post-challenge queries) The security definition above allows the adversary to make additional corruption queries in a post-challenge query phase. However, as shown in [HLWW23], the security in the setting without post-challenge queries implies the security in the setting with post-challenge queries since the aggregation algorithm is *deterministic*. For the same reason, we ignored post-challenge queries in the security definition of registered FE (Def. 2). Hence, we only consider (slotted) registered FE with a security notion that does not involve any post-challenge queries.

6 Slotted Registered IPFE from Pairing

In this section, we construct a slotted registered IPFE in the asymmetric bilinear pairing groups and prove its security in the GGM.

6.1 Construction

The slotted registered inner product functional encryption $\mathsf{SlotRIPFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{IsValid}, \mathsf{Aggregate}, \mathsf{Enc}, \mathsf{Dec})$ for a function universe $\mathcal{U}_F = \mathbb{Z}^n$, and message space $\mathcal{M} = \mathbb{Z}^n$ works as follows:

$\mathsf{Setup}(1^\lambda, 1^n, L)$: The setup algorithm takes the security parameter λ, the length n of vectors (in unary) and the number of users L (in binary) as inputs and samples $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, e) \leftarrow \mathsf{GG}(1^\lambda)$. The algorithm computes the following terms:

1. Sample $\boldsymbol{\alpha} \leftarrow \mathbb{Z}_p^n, \gamma, \beta_i \leftarrow \mathbb{Z}_p$ for all $i \in [L]$.
2. Compute $\mathbf{Z} \coloneqq g_T^{\boldsymbol{\alpha}}$ and $\Gamma \coloneqq g_1^{1/\gamma}$ where $g_T = e(g_1, g_2)$.
3. For each $i \in [L]$, compute $\mathbf{A}_i \coloneqq g_1^{\beta_i \boldsymbol{\alpha}}, B_i \coloneqq g_1^{\beta_i}, \widetilde{B}_i \coloneqq g_2^{\beta_i}, D_i \coloneqq g_2^{1/\beta_i}$.
4. For each slot $i, j \in [L]$ and $i \neq j$, compute $R_{i,j} \coloneqq g_2^{\gamma\beta_i/\beta_j}, \mathbf{S}_{i,j} \coloneqq g_2^{\gamma\beta_i\boldsymbol{\alpha}/\beta_j}$.
5. Output the common reference string as

$$\mathsf{crs} \coloneqq \begin{pmatrix} \mathcal{G}, \quad \mathbf{Z} = g_T^{\boldsymbol{\alpha}}, \quad \Gamma = g_1^{1/\gamma}, \\ \{\mathbf{A}_i = g_1^{\beta_i\boldsymbol{\alpha}}, \ B_i = g_1^{\beta_i}, \ \widetilde{B}_i = g_2^{\beta_i}, \ D_i = g_2^{1/\beta_i}\}_{i\in[L],\ell\in[n]}, \\ \{R_{i,j} = g_2^{\gamma\beta_i/\beta_j}, \ \mathbf{S}_{i,j} = g_2^{\gamma\beta_i\boldsymbol{\alpha}/\beta_j}\}_{i,j\in[L],i\neq j} \end{pmatrix}.$$

$\mathsf{KeyGen}(\mathsf{crs}, i)$: The key generation algorithm takes the common reference string crs, and a slot index $i \in [L]$ as inputs and works as follows:

1. Sample $r_i \leftarrow \mathbb{Z}_p$ and compute $U_i \coloneqq B_i^{r_i}, \widetilde{U}_i = \widetilde{B}_i^{r_i}, P_{i,j} \coloneqq R_{i,j}^{r_i}$ for all $j \in [L]$ and $j \neq i$.

2. Output the public and secret keys as

$$\mathsf{pk}_i \coloneqq \Big(U_i = g_1^{\beta_i r_i}, \widetilde{U}_i = g_2^{\beta_i r_i}, \ \ \{P_{i,j} = g_2^{\gamma \beta_i r_i / \beta_j}\}_{j \in [L], j \neq i} \Big) \ \text{ and } \ \mathsf{sk}_i \coloneqq r_i.$$

IsValid$(\mathsf{crs}, i, \mathsf{pk}_i)$: The public key verification algorithm takes the common reference string crs, a slot index $i \in [L]$ and a public key $\mathsf{pk}_i = (U_i, \widetilde{U}_i, \{P_{i,j}\}_{j \in [L], j \neq i})$, and checks the following:

$$e(U_i, g_2) \stackrel{?}{=} e(g_1, \widetilde{U}_i) \quad \text{and}$$
$$e(U_i, D_j) \stackrel{?}{=} e(\Gamma, P_{i,j}) \quad \forall j \in [L] \setminus \{i\}.$$

If the check passes then it outputs 1; otherwise 0.

Aggregate$(\mathsf{crs}, (\mathsf{pk}_1, \boldsymbol{y}_1), \ldots, (\mathsf{pk}_L, \boldsymbol{y}_L))$: The aggregate algorithm takes a common reference string crs, a list of L public key-function pairs $(\mathsf{pk}_1, \boldsymbol{y}_1), \ldots, (\mathsf{pk}_L, \boldsymbol{y}_L)$ as inputs such that $\boldsymbol{y}_i \in \mathbb{Z}^n$ and $\mathsf{pk}_i = (U_i, \widetilde{U}_i, \{P_{i,j}\}_{j \in [L], j \neq i})$ for all $i \in [L]$. It proceeds as follows:

1. Using $\mathbf{A}_i$, U_i and $\boldsymbol{y}_i$, compute $W_i \coloneqq U_i \cdot \prod_{\ell \in [n]} A_{i,\ell}^{y_{i,\ell}} = U_i \cdot g_1^{\beta_i \langle \boldsymbol{y}_i, \boldsymbol{\alpha} \rangle}$, where $A_{i,\ell}$ denotes the ℓ-th entry of $\mathbf{A}_i$.
2. Using $\mathbf{S}_{i,j}$ and $\boldsymbol{y}_i$, compute $S_{j,i} = \prod_{\ell \in [n]} S_{j,i,\ell}^{y_{j,\ell}} = g_2^{\gamma \beta_j \langle \boldsymbol{y}_i, \boldsymbol{\alpha} \rangle / \beta_i}$ for all $i, j \in [L]$ and $j \neq i$, where $S_{j,i,\ell}$ denotes the ℓ-th entry of $\mathbf{S}_{j,i}$.
3. Compute the component of MPK as $W = \prod_{i \in [L]} W_i$.
4. Compute the components of hsk_i as $S_i \coloneqq \prod_{j \in [L] \setminus \{i\}} S_{j,i}$ and $P_i \coloneqq \prod_{j \in [L] \setminus \{i\}} P_{j,i}$.
5. Output the master public key and slot-specific helper decryption keys as

$$\mathsf{MPK} \coloneqq (\mathcal{G}, \ \mathbf{Z}, \ \Gamma, \ W) \text{ and } \ \mathsf{hsk}_i \coloneqq S_i \cdot P_i.$$

Enc$(\mathsf{MPK}, \boldsymbol{x})$: The encryption algorithm takes a master public key MPK and a message $\boldsymbol{x} \in \mathbb{Z}^n$ as inputs and proceeds as follows:

1. Sample $s \leftarrow \mathbb{Z}_p$.
2. Compute $C_0 \coloneqq g_T^s$ and $\mathbf{C}_1 = (g_T^{x_1} \cdot Z_1^s, \ldots, g_T^{x_n} \cdot Z_n^s) = g_T^{\boldsymbol{x} + s\boldsymbol{\alpha}}$ where Z_ℓ denotes the ℓ-th entry of $\mathbf{Z}$.
3. Compute $C_2 \coloneqq W^{-s}$ and $C_3 \coloneqq \Gamma^s$.
4. Output the ciphertext

$$\mathsf{ct} \coloneqq (C_0, \mathbf{C}_1, C_2, C_3).$$

Dec$(\mathsf{sk}, \mathsf{hsk}, \mathsf{ct})$: The decryption algorithm takes a secret key $\mathsf{sk} = r$, a helper decryption key hsk for the i-th slot and a ciphertext $\mathsf{ct} \coloneqq (C_0, \mathbf{C}_1, C_2, C_3)$ as inputs and works as follows:

1. Compute the following terms

$$E \coloneqq e(C_2, D_i) \cdot e(C_3, \mathsf{hsk}) \text{ and } C = \prod_{\ell \in [n]} C_{1,\ell}^{y_{i,\ell}}$$

where $C_{1,\ell}$ denotes the ℓ-th entry of $\mathbf{C}_1$.

2. Output the message as $\log_{g_T} \big(C \cdot C_0^{\mathsf{sk}} \cdot E \big)$.

Completeness: Consider a key pair $(\mathsf{pk}_i, \mathsf{sk}_i)$ generated using $\mathsf{KeyGen}(\mathsf{crs}, i; r)$. Then by construction, we have $\mathsf{pk} = (U_i, \widetilde{U}_i, \{P_{i,j}\}_{j\in[L], j\neq i})$ where

$$U_i = B_i^{r_i} = g_1^{\beta_i r_i},\ \widetilde{U}_i = \widetilde{B}_i^{r_i} = g_2^{\beta_i r_i} \text{ and } P_{i,j} = R_{i,j}^{r_i} = g_2^{\gamma\beta_i r_i/\beta_j}.$$

Therefore, the validity of pk_i is verified using

$$\begin{aligned} e(U_i, g_2) &= e(g_1, g_2)^{\beta_i r_i} = e(g_1, \widetilde{U}_i) \quad \text{and} \\ e(U_i, D_j) &= e(g_1, g_2)^{\beta_i r_i/\beta_j} = e(\Gamma, P_{i,j}) \quad \forall j \in [L] \setminus \{i\} \end{aligned}$$

since $D_j = g_2^{1/\beta_j}$ and $\Gamma = g_1^{1/\gamma}$. The RIPFE satisfies completeness since the public key passes all the pairing equations defined by the $\mathsf{IsValid}$ algorithm, *i.e.* $\mathsf{IsValid}(\mathsf{crs}, \mathsf{pk}_i)$ outputs 1.

Correctness: Consider a secret key $\mathsf{sk} = r_i$, a helper decryption key $\mathsf{hsk} = S_i \cdot P_i$ and a ciphertext $\mathsf{ct} = (C_0, \mathbf{C}_1, C_2, C_3)$. Then, by construction, we have

$$\begin{aligned} \mathsf{hsk} &= \prod_{j\neq i} g_2^{\gamma r_j \beta_j/\beta_i} \cdot \prod_{\ell\in[n]} \prod_{j\neq i} g_2^{\gamma r_j y_{j,\ell}\alpha_\ell\beta_j/\beta_i} = \prod_{j\neq i} g_2^{\gamma r_j\beta_j/\beta_i} \cdot \prod_{j\neq i} g_2^{\gamma r_j\langle \boldsymbol{y}_j, \boldsymbol{\alpha}\rangle\beta_j/\beta_i}, \\ C_2 &= \prod_{i\in[L]} W_i^{-s} = \prod_{i\in[L]} U_i^{-s} \prod_{i\in[L]} \cdot \prod_{\ell\in[n]} A_{i,\ell}^{-s y_{i,\ell}} = \prod_{i\in[L]} g_1^{-s r_i\beta_i} \cdot \prod_{i\in[L]} g_1^{-s\langle \boldsymbol{y}_i, \boldsymbol{\alpha}\rangle\beta_i}, \end{aligned}$$

$$\begin{aligned} e(C_2, D_i) &= \prod_{j\in[L]} e(g_1, g_2)^{-s r_j\beta_j/\beta_i} \cdot \prod_{j\in[L]} e(g_1, g_2)^{-s\langle \boldsymbol{y}_j, \boldsymbol{\alpha}\rangle\beta_j/\beta_i}, \\ e(C_3, \mathsf{hsk}) &= \prod_{j\neq i} e(g_1, g_2)^{s r_j\beta_j/\beta_i} \cdot \prod_{j\neq i} e(g_1, g_2)^{s r_j\langle \boldsymbol{y}_j, \boldsymbol{\alpha}\rangle\beta_j/\beta_i}, \\ C &= \prod_{\ell\in[n]} C_{1,\ell}^{y_{i,\ell}} = g_T^{\langle \boldsymbol{x}, \boldsymbol{y}_i\rangle + s\langle \boldsymbol{y}_i, \boldsymbol{\alpha}\rangle}, \quad C_0^{\mathsf{sk}} = g_T^{s r_i}. \end{aligned}$$

Therefore, $E = e(C_2, D_i) \cdot e(C_3, \mathsf{hsk}) = e(g_1, g_2)^{-s(r_i + \langle \boldsymbol{y}_i, \boldsymbol{\alpha}\rangle)}$. Hence, the inner product value is obtained as $\log_{g_T}\left(C \cdot C_0^{\mathsf{sk}} \cdot E\right) = \langle \boldsymbol{x}, \boldsymbol{y}_i\rangle$.

Compactness: The master public key contains $O(n)$ group elements and each group element can be represented using $\mathsf{poly}(\lambda)$ bits. Therefore, the master public key size is bounded by $\mathsf{poly}(\lambda, |\mathcal{U}_F|, \log L)$ where $|\mathcal{U}_F| = n$. The helper decryption key contains a single group element. Since the information of the aggregated master public key is given with the helper decryption key the size is also bounded by $\mathsf{poly}(\lambda, |\mathcal{U}_F|, \log L)$.

6.2 Security Analysis

Theorem 1 *The slotted registered inner product functional encryption scheme is secure in the generic group model.*

Proof. Let a PPT adversary against our slotted registered IPFE be $\mathcal{A}$. Recall that it is sufficient to assume that there is no post-challenge queries from $\mathcal{A}$. Let Cor be the set of all corrupted slots, for which honest key pairs are generated by the challenger and later are corrupted by $\mathcal{A}$ and the corresponding secret keys are revealed to $\mathcal{A}$. Let Mal be the set of slots for which maliciously keys are provided by $\mathcal{A}$ itself. At the beginning, Cor is set an empty set and it is dynamically filled with the corrupted slot indices during the course of the security experiment. Similarly, Mal is set an empty set at the beginning of the game. It is defined during the challenge phase based on the keys provided by $\mathcal{A}$.

We consider the following sequence of hybrid games played between the adversary $\mathcal{A}$ and the challenger. Let E_i be the event of $\mathcal{A}$ outputting the correct bit coin in Game i.

Game 0**:** This is the real experiment in the generic group model. We also assume that the total number of corrupted slots is Q_{cor} and the total number of zero-test queries made by $\mathcal{A}$ is Q_{zt}. The challenger simulates $\mathcal{A}$ as follows.

Setup. The challenger samples $\boldsymbol{\alpha} \leftarrow \mathbb{Z}_p^n, \gamma, \beta_i \leftarrow \mathbb{Z}_p$ for all $i \in [L]$. It creates the following lists of group exponents:

- $L_1 = \{1, 1/\gamma, \{\beta_i \boldsymbol{\alpha}_i, \beta_i\}_{i\in[L]}\}$ which corresponds to the $\mathbb{G}_1$-elements in the CRS.
- $L_2 = \{1, \{\beta_i, 1/\beta_i\}_{i\in[L]}, \{\gamma\beta_i/\beta_j, \gamma\beta_i\boldsymbol{\alpha}/\beta_j\}_{i,j\in[L],i\neq j}\}$ which corresponds to the $\mathbb{G}_2$-elements in the CRS.
- $L_T = \{1, \boldsymbol{\alpha}\}$ which corresponds to the $\mathbb{G}_T$-elements in the CRS.

The adversary is given handles corresponding to all the elements of lists L_1, L_2 and L_T.

Pre-challenge query phase. The challenger initializes a dictionary D, a set of corrupted set $\mathsf{Cor} \leftarrow \emptyset$, a set of indices for which maliciously generated keys $\mathsf{Mal} \leftarrow \emptyset$, and sets $\mathsf{ctr} \leftarrow 0$. $\mathcal{A}$ makes the following queries:

1. *Key generation queries.* $\mathcal{A}$ specifies a slot index i. Then, the challenger samples $r_i \leftarrow \mathbb{Z}_p$, sets $\mathsf{sk}_{\mathsf{ctr}} = r_i$, updates the lists as
   - $L_1 \leftarrow L_1 \cup \{\beta_i r_i\}$,
   - $L_2 \leftarrow L_2 \cup \{\beta_i r_i, \{\gamma\beta_i r_i/\beta_j\}_{i,j\in[L],i\neq j}\}$,

   and provides $\mathcal{A}$ with the handles of newly computed group elements. The challenger increments $\mathsf{ctr} \leftarrow \mathsf{ctr} + 1$ and adds the mapping $\mathsf{ctr} \mapsto (i, h_{\mathsf{pk}_{\mathsf{ctr}}}, \mathsf{sk}_{\mathsf{ctr}})$ to the dictionary D. Here, $h_{\mathsf{pk}_{\mathsf{ctr}}}$ is the set of handles corresponding to the newly added elements in the list and is of the form $\{(1, h_i), (2, \widetilde{h}_i), \{(2, h_{i,j})\}_{j\in[L]\setminus\{i\}}\}$ with $L_1[h_i] = L_2[h_i] = \beta_i r_i$ and $L_2[h_{i,j}] = \gamma\beta_i r_i/\beta_j$.[8]
2. *Corruption queries.* In a corruption query, $\mathcal{A}$ specifies an index $1 \le c \le \mathsf{ctr}$. The challenger looks up the tuple $(i', h_{\mathsf{pk}'}, \mathsf{sk}') \leftarrow \mathsf{D}[c]$ and sends sk' to $\mathcal{A}$. It updates the set of corrupted indices as $\mathsf{Cor} \leftarrow \mathsf{Cor} \cup \{i'\}$.

Challenge phase. For each slot $i \in [L]$, $\mathcal{A}$ specifies a tuple $(c_i, h_{\mathsf{pk}_i^*}, \boldsymbol{y}_i)$ where either $c_i \in \{1, \ldots, \mathsf{ctr}\}$ to reference a challenger-generated key or $c_i = \perp$ to reference a key outside this set. Here, we denote $h_{\mathsf{pk}_i^*} = \{(1, h_i^*), (2, \widetilde{h}_i^*), \{(2, h_{i,j}^*)\}_{j\in[L]\setminus\{i\}}\}$ by the list of handles associated to the public key of the i-th slot. $\mathcal{A}$ also specifies two challenge messages $\boldsymbol{x}_0^*, \boldsymbol{x}_1^*$.

- If $c_i \in \{1, \ldots, \mathsf{ctr}\}$, then the challenger looks up the entry $\mathsf{D}[c_i] = (i', h_{\mathsf{pk}'}, \mathsf{sk}')$. If $i = i'$, then the challenger sets $h_{\mathsf{pk}_i} \leftarrow h_{\mathsf{pk}'}$. Otherwise, if $i \neq i'$, then the experiment halts.
- If $c_i = \perp$, then the challenger runs $\mathsf{IsValid}$ on the GGM. In more detail, it checks whether the following equations hold by the zero test oracle:[9]

$$L_1[h_i^*] - L_2[\widetilde{h}_i^*] \equiv 0 \bmod p, \tag{3}$$

$$L_1[h_i^*]/\beta_j - L_2[h_{i,j}^*]/\gamma \equiv 0 \bmod p \quad \forall j \in [L] \setminus \{i\}. \tag{4}$$

  If Equation (3) or (4) does not hold, the experiment halts. If the both equations hold, the challenger sets $h_{\mathsf{pk}_i} \leftarrow h_{\mathsf{pk}_i^*}$, adds the slot index i to Mal. Note that for each index i such that $c_i \neq \perp$, L additional zero-test queries are made by the above check.

[8] Recall that in the GGM, the adversary can only access the handles corresponding to the group elements in $\mathsf{pk}_{\mathsf{ctr}}$.

[9] We note that the verification of the equations requires a number of steps in GGM, which we supress here. For example, to check Eq. (4), a handle for $L_1[h_{i,j}^*]/\gamma$ is first generated by calling map on input $h_{i,j}^*$ and the handle for $1/\beta_j$. Then, a handle for $-L_1[h_{i,j}^*]/\gamma$ is generated by submitting the resulting handle to neg. A handle for $L_1[h_i^*]/\beta_j - L_2[h_{i,j}^*]/\gamma$ is generated by calling add on input the handle for $L_1[h_i^*]/\beta_j$ and $-L_1[h_{i,j}^*]/\gamma$, where the former handle is generated similarly to the latter. Finally, the obtained handle is submitted to $\mathsf{zero\text{-}test}$.

Let us denote $h_{\mathsf{pk}_i} = \{(1, h_i), (2, \widetilde{h}_i), \{(2, h_{i,j})\}_{j\in[L]\setminus\{i^*\}}\}$ for $i \in [L]$. Next, the challenger computes the handles corresponding to the challenge ciphertext by sampling $\mathsf{coin} \leftarrow \{0,1\}, s \leftarrow \mathbb{Z}_p$ and computing

$$c_0 := s, \quad \mathbf{c}_1 := \{c_{1,\ell}\}_{\ell\in[n]} = \boldsymbol{x}^*_{\mathsf{coin}} + s\boldsymbol{\alpha}, \quad c_2 := \sum_{i\in[L]} s(L_1[h_i] + \beta_i\langle \boldsymbol{y}_i, \boldsymbol{\alpha}\rangle), \quad c_3 := s/\gamma.$$

It then updates the lists as

- $L_1 \leftarrow L_1 \cup \{c_2, c_3\}$,
- $L_T \leftarrow L_T \cup \{s, \{c_{1,\ell}\}_{\ell\in[n]}\}$

and gives the handles of the newly added elements to the adversary $\mathcal{A}$.

Output phase. At the end of the experiment, $\mathcal{A}$ outputs a guess $\mathsf{coin}' \in \{0,1\}$, which is the output of the experiment.

Game 1: In this game, we partially switch to the symbolic group model and replace

$$\gamma, \quad \{\beta_i\}_{i\in[L]}, \quad \{r_i\}_{i\in[L]}, \quad s, \quad \boldsymbol{\alpha} = \{\alpha_\ell\}_{\ell\in[n]}, \quad \{c_{1,\ell}\}_{\ell\in[n]}, \quad c_2, \quad c_3$$

in $\mathbb{Z}_p$ with formal variables

$$\widehat{\Gamma}, \quad \{\widehat{\mathrm{B}}_i\}_{i\in[L]}, \quad \{\widehat{\mathrm{R}}_i\}_{i\in[L]}, \quad \widehat{\mathrm{S}}, \quad \{\widehat{\mathrm{A}}_\ell\}_{\ell\in[n]}, \quad \{\widehat{\mathrm{C}}_{1,\ell}\}_{\ell\in[n]}, \quad \widehat{\mathrm{C}}_2, \quad \widehat{\mathrm{C}}_3$$

respectively. Therefore, all the handles given to $\mathcal{A}$ refer to elements in the ring

$$\mathbb{T} := \mathbb{Z}_p[\widehat{\Gamma}, 1/\widehat{\Gamma}, \{\widehat{\mathrm{B}}_i, 1/\widehat{\mathrm{B}}_i\}_{i\in[L]}, \{\widehat{\mathrm{R}}_i\}_{i\in[L]}, \widehat{\mathrm{S}}, \{\widehat{\mathrm{A}}_\ell\}_{\ell\in[n]}, \{\widehat{\mathrm{C}}_{1,\ell}\}_{\ell\in[n]}, \widehat{\mathrm{C}}_2, \widehat{\mathrm{C}}_3]$$

where $\{\widehat{\mathrm{R}}_i\}_{i\in[L]}$ are needed to represent the secret keys sampled by the challenger. However, when $\mathcal{A}$ submits a zero-test query, the challenger substitutes the formal variables with corresponding elements in $\mathbb{Z}_p$.

In more detail, the challenger picks $\gamma, \{\beta_i\}_{i\in[L]}, s, \{\alpha_\ell\}_{\ell\in[n]}, \{r_i\}_{i\in[L]}$ at the beginning of the game as in the previous game (following the same distribution). It also computes $\{c_{1,\ell}\}_{\ell\in[n]}, c_2, c_3 \in \mathbb{Z}_p$ as in the previous game when it answers the challenge query. Once the $\mathbb{Z}_p$-elements are sampled, these remain fixed throughout the experiment. Furthermore, for an honest slot i, r_i is revealed to the adversary whenever it makes a corruption query for the slot and the associated index i is added to Cor. A zero-test query before and during the challenge phase[10] always corresponds to an element $f(\widehat{\Gamma}, \{\widehat{\mathrm{B}}_i\}_{i\in[L]}, \{\widehat{\mathrm{R}}_i\}_{i\in[L]}, \widehat{\mathrm{S}}, \{\widehat{\mathrm{A}}_\ell\}_{\ell\in[n]})$ in the sub-ring $\mathbb{T}'$ of $\mathbb{T}$ which is defined as

$$\mathbb{T}' := \mathbb{Z}_p[\widehat{\Gamma}, 1/\widehat{\Gamma}, \{\widehat{\mathrm{B}}_i, 1/\widehat{\mathrm{B}}_i\}_{i\in[L]}, \{\widehat{\mathrm{R}}_i\}_{i\in[L]}, \widehat{\mathrm{S}}, \{\widehat{\mathrm{A}}_\ell\}_{\ell\in[n]}].$$

For the query, the challenger returns 1 if

$$f(\gamma, \{\beta_i\}_{i\in[L]}, \{r_i\}_{i\in[L]}, s, \{\alpha_\ell\}_{\ell\in[n]}) = 0$$

holds over $\mathbb{Z}_p$, and 0 otherwise. A zero test query after the challenge phase corresponds to an element $f(\widehat{\Gamma}, \{\widehat{\mathrm{B}}_i\}_{i\in[L]}, \{\widehat{\mathrm{R}}_i\}_{i\in[L]}, \widehat{\mathrm{S}}, \{\widehat{\mathrm{A}}_\ell\}_{\ell\in[n]}, \{\widehat{\mathrm{C}}_{1,\ell}\}_{\ell\in[n]}, \widehat{\mathrm{C}}_2, \widehat{\mathrm{C}}_3)$ in $\mathbb{T}$. For the query, the challenger returns 1 if

$$f(\gamma, \{\beta_i\}_{i\in[L]}, \{r_i\}_{i\in[L]}, s, \{\alpha_\ell\}_{\ell\in[n]}, \{c_{1,\ell}\}_{\ell\in[n]}, c_2, c_3) = 0$$

holds over $\mathbb{Z}_p$, and 0 otherwise. We show in Lemma 1 that $\Pr[\mathsf{E}_0] = \Pr[\mathsf{E}_1]$.

[10] Recall that $\mathsf{IsValid}$ ia run during the challenge phase and it involves the zero-test queries.

We now list all the components in $\mathbb{T}$ for which the corresponding handles are given to $\mathcal{A}$ during the game. The components of $\mathbb{T}$ related to the group elements of $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are given by

$$\begin{aligned} V_1 &= \left\{1, 1/\widehat{\Gamma}, \{\widehat{\mathrm{B}}_i, \{\widehat{\mathrm{B}}_i\widehat{\mathrm{A}}_\ell\}_{\ell\in[L]}\}_{i\in[L]}, \{\widehat{\mathrm{B}}_i\widehat{\mathrm{R}}_i\}_{i\in[L]}, \widehat{\mathrm{C}}_2, \widehat{\mathrm{C}}_3\right\}, \\ V_2 &= \left\{1, \{\widehat{\mathrm{B}}_i, 1/\widehat{\mathrm{B}}_i, \widehat{\mathrm{B}}_i\widehat{\mathrm{R}}_i\}_{i\in[L]}, \{\widehat{\Gamma}\widehat{\mathrm{B}}_i/\widehat{\mathrm{B}}_j, \widehat{\Gamma}\widehat{\mathrm{B}}_i\widehat{\mathrm{R}}_i/\widehat{\mathrm{B}}_j, \{\widehat{\Gamma}\widehat{\mathrm{B}}_i\widehat{\mathrm{A}}_\ell/\widehat{\mathrm{B}}_j\}_{\ell\in[n]}\}_{i,j\in[L],i\neq j}\right\}, \\ V_T &= \left\{1, \widehat{\mathrm{S}}, \{\widehat{\mathrm{C}}_{1,\ell}\}_{\ell\in[n]}\right\}, \end{aligned} \tag{5}$$

respectively. We also define $V_3 \subset V_2$ as $V_3 = \{1/\widehat{\mathrm{B}}_i\}_{i\in[L]}$. Note that any handle refers to an element $f \in \mathbb{T}$ that can be represented as

$$\begin{aligned} f &= \sum_{(X,Y)\in V_1\times V_2} a_{X,Y} X\cdot Y + \sum_{Z\in V_T} a_Z Z \\ &= \underbrace{\sum_{(X,Y)\in(V_1\times V_2)\setminus(\{\widehat{\mathrm{C}}_2\}\times V_3)} a_{X,Y} X\cdot Y}_{:=f_1} + \underbrace{\sum_{i\in[L]} b_i\cdot\widehat{\mathrm{C}}_2\cdot(1/\widehat{\mathrm{B}}_i)}_{:=f_2} + \underbrace{d_0\cdot\widehat{\mathrm{S}} + \sum_{\ell\in[n]} d_\ell\cdot\widehat{\mathrm{C}}_{1,\ell}}_{:=f_3} \end{aligned} \tag{6}$$

using $a_{X,Y} \in \mathbb{Z}_p$ for $X \in V_1$ and $Y \in V_2$ and $a_Z \in \mathbb{Z}_p$ for $Z \in V_T$. In the last equation, we rename the coefficients as $b_i \coloneqq a_{\widehat{\mathrm{C}}_2,1/\widehat{\mathrm{B}}_i}$, $d_0 \coloneqq a_{\widehat{\mathrm{S}}}$, and $d_\ell \coloneqq a_{\widehat{\mathrm{C}}_{1,\ell}}$ and divide the sum into three parts for later analysis. Note that the above representation of f does not uniquely determine $\{a_{X,Y}\}_{X,Y}$, since different choices of X and Y may lead to the same product $X\cdot Y$.[11] However, we can see that $\{b_i\}_{i\in[L]}$, and $\{d_\ell\}_{\ell\in[0,n]}$ are uniquely determined from f.

Game 2: In this game, we change the way zero-test queries are answered. Namely, whenever a zero-test query corresponding to a handle of $f \in \mathbb{T}$ is made, the challenger defines $f'' \in \mathbb{T}'$ as $f'' \coloneqq f$ if the query is made during or before the challenge phase. Otherwise, the challenger defines f'' as

$$\begin{aligned} &f''(\widehat{\Gamma}, \{\widehat{\mathrm{B}}_i\}_{i\in[L]}, \{\widehat{\mathrm{R}}_i\}_{i\in[L]}, \widehat{\mathrm{S}}, \{\widehat{\mathrm{A}}_\ell\}_{\ell\in[n]}) \coloneqq \\ &\quad f(\widehat{\Gamma}, \{\widehat{\mathrm{B}}_i\}_{i\in[L]}, \{\widehat{\mathrm{R}}_i\}_{i\in[L]}, \widehat{\mathrm{S}}, \{\widehat{\mathrm{A}}_\ell\}_{\ell\in[n]}, \{C_{1,\ell}\}_{\ell\in[n]}, C_2, C_3) \end{aligned}$$

where $\{C_{1,\ell}\}_{\ell\in[n]}$, C_2, and C_3 are polynomials in $\mathbb{T}'$ defined as

$$C_{1,\ell} \coloneqq \widehat{\mathrm{S}}\widehat{\mathrm{A}}_\ell + x^*_{\ell,\mathsf{coin}}, \qquad C_2 \coloneqq \sum_{i\in[L]} \widehat{\mathrm{S}}(L_1[h_i] + \widehat{\mathrm{B}}_i\cdot\langle \boldsymbol{y}_i, \widehat{\mathbf{A}}\rangle), \qquad C_3 \coloneqq \widehat{\mathrm{S}}/\widehat{\Gamma} \tag{7}$$

respectively, where $\langle \boldsymbol{y}_i, \widehat{\mathbf{A}}\rangle$ denotes the linear combination of the vector $\boldsymbol{y}_i$ and the variables $\{\widehat{\mathrm{A}}_\ell\}_{\ell\in[n]}$, that is $\langle \boldsymbol{y}_i, \widehat{\mathbf{A}}\rangle = \sum_{\ell\in[n]} y_{i,\ell}\widehat{\mathrm{A}}_\ell$. Here, $L_1[h_i]$ is an element in $\mathbb{T}'$ defined as in **Game** 1. Then, $f' \in \mathbb{T}'$ is defined as

$$\begin{aligned} &f'(\widehat{\Gamma}, \{\widehat{\mathrm{B}}_i\}_{i\in[L]}, \{\widehat{\mathrm{R}}_i\}_{i\in[L]\setminus\mathsf{Cor}}, \widehat{\mathrm{S}}, \{\widehat{\mathrm{A}}_\ell\}_{\ell\in[n]}) \coloneqq \\ &\quad f''(\widehat{\Gamma}, \{\widehat{\mathrm{B}}_i\}_{i\in[L]}, \{\widehat{\mathrm{R}}_i\}_{i\in[L]\setminus\mathsf{Cor}}, \{r_i\}_{i\in\mathsf{Cor}}, \widehat{\mathrm{S}}, \{\widehat{\mathrm{A}}_\ell\}_{\ell\in[n]}). \end{aligned} \tag{8}$$

We emphasize that Cor is the set of corrupted indices at the time when the zero-test query is made. After that, it answers the zero-test query by substituting the formal variables with their values in $\mathbb{Z}_p$. In particular, the challenger returns 1 if it holds that

$$f'(\gamma, \{\beta_i\}_{i\in[L]}, \{r_i\}_{i\in[L]\setminus\mathsf{Cor}}, s, \{\alpha_\ell\}_{\ell\in[n]}) = 0$$

and 0 otherwise. We show in Lemma 2 that $\Pr[\mathsf{E}_1] = \Pr[\mathsf{E}_2]$.

[11] For example, we have $1\cdot\widehat{\mathrm{B}}_i = \widehat{\mathrm{B}}_i\cdot 1$ and $\widehat{\mathrm{B}}_i\cdot(1/\widehat{\mathrm{B}}_j) = (1/\widehat{\Gamma})\cdot(\widehat{\Gamma}\widehat{\mathrm{B}}_i/\widehat{\mathrm{B}}_j)$.

Game 3**:** In this game, we switch to the SGM completely and change the way the zero-test queries are answered by the challenger. The challenger no longer uses the integers $\gamma, \{\beta_i\}_{i\in[L]}, \{r_i\}_{i\in[L]\setminus\mathsf{Cor}}, s, \{\alpha_\ell\}_{\ell\in[n]}$ from $\mathbb{Z}_p$ in order to simulate the zero-test queries. Instead, when $\mathcal{A}$ submits a handle corresponding to $f \in \mathbb{T}$, the challenger first computes associated $f' \in \mathbb{T}'$ and returns 1 if

$$f'(\widehat{\Gamma}, \{\widehat{\mathrm{B}}_i\}_{i\in[L]}, \{\widehat{\mathrm{R}}_i\}_{i\in[L]\setminus\mathsf{Cor}}, \widehat{\mathrm{S}}, \{\widehat{\mathrm{A}}_\ell\}_{\ell\in[n]}) = 0 \tag{9}$$

holds over $\mathbb{T}'$, 0 otherwise. Note that this change in particular means that Eq. (3) and (4) that are checked during the challenge phase is replaced with the following check:

$$L'_1[h^*_i] - L'_2[\widetilde{h}^*_i] \equiv 0 \quad \text{over } \mathbb{T}' \tag{10}$$

and

$$L'_1[h^*_i]/\widehat{\mathrm{B}}_j - L'_2[h^*_{i,j}]/\widehat{\Gamma} \equiv 0 \quad \text{over } \mathbb{T}' \quad \forall j \in [L] \setminus \{i\} \tag{11}$$

where $L'_1[h^*_i]$ and $L'_2[h^*_{i,j}]$ are elements in $\mathbb{T}'$ that are obtained by replacing $\{\widehat{\mathrm{R}}_i\}_{i\in\mathsf{Cor}}$ with $\{r_i\}_{i\in\mathsf{Cor}}$ that appear in $L_1[h^*_i]$ and $L_2[h^*_{i,j}]$, respectively.[12] We show in Lemma 3 that

$$|\Pr[\mathsf{E}_2] - \Pr[\mathsf{E}_3]| \leq Q_{\mathsf{zt}}(L+7)^2/p.$$

Game 4**:** In this game, the challenger checks if the adversarially generated public keys are in a specific form before proceeding to computing the challenge ciphertext. More precisely, if $\mathcal{A}$ submits handles $\{(1, h^*_i), (2, \widetilde{h}^*_i), \{(2, h^*_{i,j})\}_{j\in[L]\setminus\{i\}}\}$ with $c_i = \perp$ at the challenge phase, the challenger verifies whether the handles satisfy

$$L'_1[h^*_i] = L'_2[\widetilde{h}^*_i] = r^*_i\widehat{\mathrm{B}}_i + t^*_i\widehat{\mathrm{B}}_i\widehat{\mathrm{R}}_i, \quad L'_2[h^*_{i,j}] = r^*_i\widehat{\Gamma}\widehat{\mathrm{B}}_i/\widehat{\mathrm{B}}_j + t^*_i\widehat{\Gamma}\widehat{\mathrm{B}}_i\widehat{\mathrm{R}}_i/\widehat{\mathrm{B}}_j, \ \forall j \neq i \tag{12}$$

for some $r^*_i, t^*_i \in \mathbb{Z}_p$ and aborts the experiment otherwise. We show in Lemma 4 that $\Pr[\mathsf{E}_3] = \Pr[\mathsf{E}_4]$.

Game 5**:** In this game, we further modify the way the zero-test queries are answered after the challenge phase. When $\mathcal{A}$ submits a handle corresponding to $f \in \mathbb{T}$ for a zero-test query, the challenger represents f as in Eq. (6) and returns 0 if there exists $i \notin \mathsf{Cor} \cup \mathsf{Mal}$ such that $b_i \neq 0$. Otherwise, the challenger proceeds as in the previous game for answering the zero-test queries. We show in Lemma 5 that $\Pr[\mathsf{E}_4] = \Pr[\mathsf{E}_5]$.

Game 6**:** In this game, we further modify the way the zero-test queries are answered after the challenge phase. When $\mathcal{A}$ submits a handle corresponding to $f \in \mathbb{T}$ for a zero-test query, the challenger represents f as in Eq. (6) and returns 0 if

$$\boldsymbol{d} = -\sum_{i\in\mathsf{Cor}\cup\mathsf{Mal}} b_i\boldsymbol{y}_i, \quad \text{where} \quad \boldsymbol{d} = (d_1 \ldots, d_n) \tag{13}$$

does not hold. Otherwise, the challenger proceeds as in the previous game for answering the zero-test queries. We show in Lemma 6 that $\Pr[\mathsf{E}_5] = \Pr[\mathsf{E}_6]$.

In Lemma 7, we show $\Pr[\mathsf{E}_6] = 1/2$. Combining the advantage of $\mathcal{A}$ in distinguishing between the adjacent games, we obtain

$$|\Pr[\mathsf{E}_0] - \frac{1}{2}| \leq Q_{\mathsf{zt}}(L+7)^2/p$$

which is negligible in the security parameter.

We now prove Lemma 1 to 6 to complete the security analysis.

[12] $L_1[h^*_i]$ and $L_2[h^*_{i,j}]$ do not contain the terms $\{\widehat{\mathrm{C}}_{1,\ell}\}_{\ell\in[n]}, \widehat{\mathrm{C}}_2, \widehat{\mathrm{C}}_3$ since the challenge ciphertext has not been generated yet at the point when these handles are specified. So, we can skip the step where they are replaced with the corresponding polynomials in $\mathbb{T}'$ when transforming $L_1[h^*_i]$ and $L_2[h^*_{i,j}]$ into $L'_1[h^*_i]$ and $L'_2[h^*_{i,j}]$.

Lemma 1 *For any adversary, it holds that* $\Pr[\mathsf{E}_0] = \Pr[\mathsf{E}_1]$.

Proof. The only difference between Game 0 and Game 1 is that the challenger switched to the symbolic group model in Game 1, however, it answers the zero-test queries of $\mathcal{A}$ in the exact same way as it does in Game 0. In particular, in both the games, the zero-test queries are answered using the same distribution of $\gamma, \{\beta_i\}_{i\in[L]}, \{r_i\}_{i\in[L]}, s, \{\alpha_\ell\}_{\ell\in[n]}, \{c_{1,\ell}\}_{\ell\in[n]}, c_2, c_3$ over $\mathbb{Z}_p$. Hence, the view of $\mathcal{A}$ remains the same in both the games. The lemma follows. □

Lemma 2 *For any adversary, it holds that* $\Pr[\mathsf{E}_1] = \Pr[\mathsf{E}_2]$.

Proof. The only difference between Game 1 and Game 2 is in how the zero-test queries corresponding to $f \in \mathbb{T}$ are answered. We only consider the case where the query is made after the challenge phase, since the other case is simpler. When the query corresponding to $f \in \mathbb{T}$ is made, the challenger replaces the formal variables $\{\widehat{\mathrm{C}}_{1,\ell}\}_{\ell\in[n]}, \widehat{\mathrm{C}}_2, \widehat{\mathrm{C}}_3$ with the polynomials $\{C_{1,\ell}\}_{\ell\in[n]}, C_2, C_3 \in \mathbb{T}'$ defined as Eq. (7) and $\{\widehat{\mathrm{R}}_i\}_{i\in\mathsf{Cor}}$ with $\{r_i\}_{i\in\mathsf{Cor}}$, represents f as an element f' in the subring $\mathbb{T}'$, and replaces the remaining formal variables with the $\mathbb{Z}_p$ values to see if the resulting value equals to 0 in Game 2, whereas the challenger directly replaces all the formal variables in f with the corresponding $\mathbb{Z}_p$ values without going through the intermediate polynomial f' in Game 1. However, this does not change the value that the zero-test oracle returns to the adversary since we have

$$\begin{aligned} & f'(\mathsf{values}') = f''(\mathsf{values}', \mathsf{values}'') = f''(\mathsf{values}) \\ = \;& f(\mathsf{values}, \{C_{1,\ell}(\mathsf{values})\}_{\ell\in[n]}, C_2(\mathsf{values}), C_3(\mathsf{values})) = f(\mathsf{values}, \{c_{1,\ell}\}_{\ell\in[n]}, c_2, c_3) \end{aligned}$$

where values, values', and values'' are short-hand expressions for $(\gamma, \{\beta_i\}_{i\in[L]}, \{r_i\}_{i\in[L]}, s, \{\alpha_\ell\}_{\ell\in[n]})$, $(\gamma, \{\beta_i\}_{i\in[L]}, \{r_i\}_{i\in[L]\setminus\mathsf{Cor}}, s, \{\alpha_\ell\}_{\ell\in[n]})$, and $\{r_i\}_{i\in\mathsf{Cor}}$, respectively. Hence, the view of $\mathcal{A}$ remains the same in both the games. The lemma follows. □

Lemma 3 *For any adversary, it holds that* $|\Pr[\mathsf{E}_2] - \Pr[\mathsf{E}_3]| \leq (Q_{\mathsf{zt}} + Q_{\mathsf{cor}}L)(L+5)/p$.

Proof. The only change of Game 3 from Game 2 is that the challenger uses the formal variables $\widehat{\Gamma}, \{\widehat{\mathrm{B}}_i\}_{i\in[L]}$, $\{\widehat{\mathrm{R}}_i\}_{i\in[L]\setminus\mathsf{Cor}}, \widehat{\mathrm{S}}, \{\widehat{\mathrm{A}}_\ell\}_{\ell\in[n]}$ instead of the $\mathbb{Z}_p$ values $\gamma, \{\beta_i\}_{i\in[L]}, \{r_i\}_{i\in[L]\setminus\mathsf{Cor}}, s, \{\alpha_\ell\}_{\ell\in[n]}$ to answer the zero-test queries of $\mathcal{A}$, where Cor is the set of corrputed indices at the time when the query is made. Therefore, the only possible case when Game 3 behaves differently from Game 2 in $\mathcal{A}$'s view is the following: $\mathcal{A}$ submits a handle for an element $f \in \mathbb{T}$ to the zero-test oracle satisfying

$$f'(\gamma, \{\beta_i\}_{i\in[L]}, \{r_i\}_{i\in[L]}, s, \{\alpha_\ell\}_{\ell\in[n]}) = 0 \;\bigwedge\; f'(\widehat{\Gamma}, \{\widehat{\mathrm{B}}_i\}_{i\in[L]}, \{\widehat{\mathrm{R}}_i\}_{i\in[L]\setminus\mathsf{Cor}}, \widehat{\mathrm{S}}, \{\widehat{\mathrm{A}}_\ell\}_{\ell\in[n]}) \neq 0$$

where $f' \in \mathbb{T}'$ is defined from associated $f \in \mathbb{T}$ as in Eq. (8). Let us denote the event as $\mathsf{E}_{\mathrm{bad}}$. It is sufficient to bound the probability of $\mathsf{E}_{\mathrm{bad}}$ in Game 2.

We now fix an element $f' \in \mathbb{T}'$ and define a *polynomial* $g(\widehat{\Gamma}, \{\widehat{\mathrm{B}}_i\}_{i\in[L]}, \{\widehat{\mathrm{R}}_i\}_{i\in[L]\setminus\mathsf{Cor}}, \widehat{\mathrm{S}}, \{\widehat{\mathrm{A}}_\ell\}_{\ell\in[n]}) \in \mathbb{Z}_p[\widehat{\Gamma}, \{\widehat{\mathrm{B}}_i\}_{i\in[L]}, \{\widehat{\mathrm{R}}_i\}_{i\in[L]\setminus\mathsf{Cor}}, \widehat{\mathrm{S}}, \{\widehat{\mathrm{A}}_\ell\}_{\ell\in[n]}]$ as

$$\begin{aligned} & g(\widehat{\Gamma}, \{\widehat{\mathrm{B}}_i\}_{i\in[L]}, \{\widehat{\mathrm{R}}_i\}_{i\in[L]\setminus\mathsf{Cor}}, \widehat{\mathrm{S}}, \{\widehat{\mathrm{A}}_\ell\}_{\ell\in[n]}) \\ & = \widehat{\Gamma} \cdot \textstyle\prod_{i\in[L]} \widehat{\mathrm{B}}_i \cdot f'(\widehat{\Gamma}, \{\widehat{\mathrm{B}}_i\}_{i\in[L]}, \{\widehat{\mathrm{R}}_i\}_{i\in[L]\setminus\mathsf{Cor}}, \widehat{\mathrm{S}}, \{\widehat{\mathrm{A}}_\ell\}_{\ell\in[n]}). \end{aligned}$$

The product $\widehat{\Gamma} \cdot \prod_{i\in[L]} \widehat{\mathrm{B}}_i$ is introduced to cancel out any denominator of $f' \in \mathbb{T}'$. Note that g is a *polynomial* in the ring $\mathbb{Z}_p[\widehat{\Gamma}, \{\widehat{\mathrm{B}}_i\}_{i\in[L]}, \{\widehat{\mathrm{R}}_i\}_{i\in[L]\setminus\mathsf{Cor}}, \widehat{\mathrm{S}}, \{\widehat{\mathrm{A}}_\ell\}_{\ell\in[n]}]$ rather than in $\mathbb{T}$. We have that the event $\mathsf{E}_{\mathrm{bad}}$ occurs if and only if

$$\begin{aligned} & g(\gamma, \{\beta_i\}_{i\in[L]}, \{r_i\}_{i\in[L]\setminus\mathsf{Cor}}, s, \{\alpha_\ell\}_{\ell\in[n]}) = 0 \;\bigwedge \\ & g(\widehat{\Gamma}, \{\widehat{\mathrm{B}}_i\}_{i\in[L]}, \{\widehat{\mathrm{R}}_i\}_{i\in[L]\setminus\mathsf{Cor}}, \widehat{\mathrm{S}}, \{\widehat{\mathrm{A}}_\ell\}_{\ell\in[n]}) \neq 0 \end{aligned}$$

since γ and $\{\beta_i\}_{i\in[L]}$ are non-zero over $\mathbb{Z}_p$. Using Schwartz-Zippel lemma (Lemma 1), we can bound the probability of $\mathsf{E}_{\mathrm{bad}}$ by $(L+5)^2/p$ since the maximum degree of g is $(L+5)$ over the ring $\mathbb{Z}_p[\widehat{\Gamma}, \{\widehat{\mathrm{B}}_i\}_{i\in[L]}, \{\widehat{\mathrm{R}}_i\}_{i\in[L]\setminus\mathsf{Cor}}, \widehat{\mathrm{S}}, \{\widehat{\mathrm{A}}_\ell\}_{\ell\in[n]}]$. The maximum degree is calculated by representing f as in Eq. (6) and noting that the degree of C_2 as a polynomial in $\mathbb{T}'$ does not exceed 3, since each $L_1[h_i]$ should be the linear combination of element in $V_1\setminus\{\widehat{\mathrm{C}}_2, \widehat{\mathrm{C}}_3\}$ (See Eq. (7).). Since there are a total of $Q_{\mathsf{zt}} + Q_{\mathsf{cor}}L$ number of zero-test queries, the lemma follows. □

Lemma 4 *For any adversary, it holds that* $\Pr[\mathsf{E}_3] = \Pr[\mathsf{E}_4]$.

Proof. The only difference between Game 3 and Game 4 is that the challenger aborts the experiment in Game 4 at the challenge phase if the adversarially sampled public keys do not match with a particular format. In particular, if $\mathcal{A}$ submits handles $\{(1, h_i^*), (2, \widetilde{h}_i^*), \{(2, h_{i,j}^*)\}_{j\in[L]\setminus\{i\}}\}$ representing the handles for (an adversarially generated) public key for the slot i at the challenge phase, then the challenger verifies whether the handles satisfy Eq. (12) for some $r_i^*, t_i^* \in \mathbb{Z}_p$ and aborts the game if not in Game 4. Note that in both the games, the challenger runs $\mathsf{IsValid}$ before proceeding to computing the challenge ciphertext, otherwise the challenger aborts the experiment. Thus, the two games only differ from $\mathcal{A}$'s view if $\mathsf{IsValid}$ ouputs 1 on those handles, but the handles do not match the particular format shown in Eq. (12). Here, we show that this cannot happen and thus the two games are in fact equivalent.

For proving this, we first define the set of monomials V_0 as $V_1\setminus\{\widehat{\mathrm{C}}_2, \widehat{\mathrm{C}}_3\}$. We can see that $L_1[h_i^*]$ should be an element in $\mathbb{T}'$ that can be represented as a linear combination of monomials in V_0 with coefficients in $\mathbb{Z}_p$.[13] This implies that $L_1'[h_i^*]$ can be represented as a linear combination of monomials in V_0' with coefficients in $\mathbb{Z}_p$, where V_0' is defined as

$$V_0' \coloneqq \left\{1, 1/\widehat{\Gamma}, \{\widehat{\mathrm{B}}_k, \{\widehat{\mathrm{B}}_k\widehat{\mathrm{A}}_\ell\}_{\ell\in[L]}\}_{k\in[L]}, \{\widehat{\mathrm{B}}_k\widehat{\mathrm{R}}_k\}_{k\in[L]\setminus\mathsf{Cor}}\right\}$$

Similarly, we can also see that $L_2[\widetilde{h}_i^*]$ and $L_2[h_{i,j}^*]$ for $j \in [L]\setminus\{i\}$ should correspond to elements in $\mathbb{T}'$ that can be represented as linear combinations of monomials in V_2 with coefficients in $\mathbb{Z}_p$. This implies that $L_2'[h_i^*]$ and $L_2'[h_{i,j}^*]$ for all $j \in [L]\setminus\{i\}$ can be represented as linear combinations of monomials in V_2' with coefficients in $\mathbb{Z}_p$, where V_2' is defined as

$$V_2' = \left\{1, \{\widehat{\mathrm{B}}_k, 1/\widehat{\mathrm{B}}_k\}_{k\in[L]}, \{\widehat{\Gamma}\widehat{\mathrm{B}}_k/\widehat{\mathrm{B}}_j, \{\widehat{\Gamma}\widehat{\mathrm{B}}_k\widehat{\mathrm{A}}_\ell/\widehat{\mathrm{B}}_j\}_{\ell\in[n]}\}_{k,j\in[L],k\neq j}, \{\widehat{\mathrm{B}}_k\widehat{\mathrm{R}}_k, \{\widehat{\Gamma}\widehat{\mathrm{B}}_k\widehat{\mathrm{R}}_k/\widehat{\mathrm{B}}_j\}_{j\neq k}\}_{k\in[L]\setminus\mathsf{Cor}}\right\}.$$

If the handles pass the verification, it in particular means that the handles satisfy Eq. (10) . This implies that $L_1'[h_i^*]$ (and thus $L_2'[\widetilde{h}_i^*]$) can be represented as a linear combination of monomials in $V_0' \cap V_2' = \{1\} \cup \{\widehat{\mathrm{B}}_k\}_{k\in[L]} \cup \{\widehat{\mathrm{B}}_k\widehat{\mathrm{R}}_k\}_{k\in[L]\setminus\mathsf{Cor}}$ with coefficients in $\mathbb{Z}_p$. Namely, we can write

$$L_1'[h_i^*] = e_0 + \sum_{k\in[L]} e_k\widehat{\mathrm{B}}_k + \sum_{k\in[L]\setminus\mathsf{Cor}} e_k'\widehat{\mathrm{B}}_k\widehat{\mathrm{R}}_k$$

using $e_k \in \mathbb{Z}_p$ for $k \in [0, L]$ and $e_k' \in \mathbb{Z}_p$ for $k \in [L]\setminus\mathsf{Cor}$. We then show that $e_k = 0$ and $e_k' = 0$ hold for all $k \neq i$ if the handles pass the verification shown in Eq. (11).

- We first show that $e_0 = 0$. For the sake of contradiction, let us assume that $e_0 \neq 0$ and take some index $k^* \neq [L]\setminus\{i\}$. Then, $L_2'[h_{i,k^*}^*]$ should contain the term $\widehat{\Gamma}/\widehat{\mathrm{B}}_{k^*}$, since Eq. (11) with $k^* \neq i$ implies $L_2'[h_{i,k^*}^*] = \widehat{\Gamma}\cdot L_1'[h_i^*]/\widehat{\mathrm{B}}_{k^*}$. However, this contradicts the fact that $L_2'[h_{i,k^*}^*]$ can be represented as a linear combination of the monomials in V_2' with coefficients in $\mathbb{Z}_p$, since $\widehat{\Gamma}/\widehat{\mathrm{B}}_{k^*} \notin V_2'$.
- We next show that $e_k = 0$ for all $k \in [L]\setminus\{i\}$. For the sake of contradiction, let us assume that $e_{k^*} \neq 0$ holds for some $k^* \neq i$. Then, $L_2'[h_{i,k^*}^*]$ should contain the term $\widehat{\Gamma}$, since Eq. (11) with $j = k^*$ implies $L_2'[h_{i,k^*}^*] = \widehat{\Gamma}\cdot L_1'[h_i^*]/\widehat{\mathrm{B}}_{k^*}$. However, this contradicts the fact that $L_2'[h_{i,k^*}^*]$ can be represented as a linear combination of the monomials in V_2' with coefficients in $\mathbb{Z}_p$, since $\widehat{\Gamma} \notin V_2'$.

[13] Note that the linear combination should not include $\widehat{\mathrm{C}}_2$ and $\widehat{\mathrm{C}}_3$, since the handle is submitted before the challenge phase.

- We finally show that $e'_k = 0$ for all $k \neq i$. For the sake of contradiction, let us assume that $e'_{k^*} \neq 0$ holds for some $k^* \neq i$. Then, $L'_2[h^*_{i,k^*}]$ should contain the term $\widehat{\Gamma}\widehat{\mathrm{R}}_{k^*}$, since Eq. (11) with $j = k^*$ implies $L'_2[h^*_{i,k^*}] = \widehat{\Gamma} \cdot L'_1[h^*_i]/\widehat{\mathrm{B}}_{k^*}$. However, this contradicts the fact that $L'_2[h^*_{i,k^*}]$ can be represented as a linear combination of the monomials in V'_2 with coefficients in $\mathbb{Z}_p$, since $\widehat{\Gamma}\widehat{\mathrm{R}}_{k^*} \notin V'_2$.

The above shows that $L'_1[h^*_i]$ can be written as $L'_1[h^*_i] = r^*_i\widehat{\mathrm{B}}_i + t^*_i\widehat{\mathrm{B}}_i\widehat{\mathrm{R}}_i$ using $r^*_i \in \mathbb{Z}_p$ and $t^*_i \in \mathbb{Z}_p$, if we rename e_i and e'_i as r^*_i and t^*_i, respectively. Then, $L'_2[\widetilde{h}^*_i] = r^*_i\widehat{\mathrm{B}}_i + t^*_i\widehat{\mathrm{B}}_i\widehat{\mathrm{R}}_i$ follows from Eq. (10) and $L'_2[h^*_{i,j}] = r^*_i\widehat{\Gamma}\widehat{\mathrm{B}}_i/\widehat{\mathrm{B}}_j + t^*_i\widehat{\Gamma}\widehat{\mathrm{B}}_i\widehat{\mathrm{R}}_i/\widehat{\mathrm{B}}_j$ for all $j \neq i$ follows from Eq.(11).

Lemma 5 *For any adversary, it holds that* $\Pr[\mathsf{E}_4] = \Pr[\mathsf{E}_5]$.

Proof. The only difference between Game 4 and Game 5 is in the way zero-test queries are answered. The only possible case when Game 5 behaves differently from Game 4 in $\mathcal{A}$'s view is the case where the following event occurs: $\mathcal{A}$ submits a handle for $f \in \mathbb{T}$ to the zero-test oracle such that $b_{i^*} \neq 0$ for some $i^* \in [L]\backslash(\mathsf{Cor} \cup \mathsf{Mal})$ and the associated polynomial $f' \in \mathbb{T}'$ equals to 0 in $\mathbb{T}'$. Here, we prove that f' cannot be 0 for such f and thus the two games are in fact equivalent.

Recall that f' is obtained by replacing $\{\widehat{\mathrm{C}}_{1,\ell}\}_{\ell\in[n]}$, $\widehat{\mathrm{C}}_2$, and $\widehat{\mathrm{C}}_3$ in Eq. (6) with polynomials $\{C_{1,\ell}\}_{\ell\in[n]}$, C_2, and C_3 defined as in Eq. (7) to obtain $f'' \in \mathbb{T}'$ and then replacing $\{\widehat{\mathrm{R}}_i\}_{i\in\mathsf{Cor}}$ that appear in f'' with $\{r_i\}_{i\in\mathsf{Cor}}$. We divide f into f_1, f_2, and f_3 as in Eq. (6) and define the corresponding polynomials f''_1, f''_2, f''_3, and f'_1, f'_2, and f'_3 similarly. First observe that we can write f''_2 as follows:

$$f''_2 = \sum_{i\in[L]} b_i \cdot C_2 \cdot (1/\widehat{\mathrm{B}}_i) = \sum_{i\in[L],j\in[L]} b_i \cdot \widehat{\mathrm{S}}(L_1[h_j] + \widehat{\mathrm{B}}_j \cdot \langle \boldsymbol{y}_j, \widehat{\mathbf{A}}\rangle)(1/\widehat{\mathrm{B}}_i).$$

If we replace $\{\widehat{\mathrm{R}}_i\}_{i\in\mathsf{Cor}}$ that (implicitly) appear in the above sum with $\{r_i\}_{i\in\mathsf{Cor}}$, we obtain the following:

$$\begin{aligned}
f'_2 &= \sum_{i\in[L],j\in[L]} b_i \cdot \widehat{\mathrm{S}}(L'_1[h_j] + \widehat{\mathrm{B}}_j \cdot \langle \boldsymbol{y}_j, \widehat{\mathbf{A}}\rangle)(1/\widehat{\mathrm{B}}_i) \\
&= \sum_{i\in[L],j\in[L]\backslash(\mathsf{Mal}\cup\mathsf{Cor})} b_i \cdot \widehat{\mathrm{S}}\widehat{\mathrm{B}}_j\widehat{\mathrm{R}}_j/\widehat{\mathrm{B}}_i + \sum_{i\in[L],j\in\mathsf{Cor}\backslash\mathsf{Mal}} b_i r_j \cdot \widehat{\mathrm{S}}\widehat{\mathrm{B}}_j/\widehat{\mathrm{B}}_i \\
&\qquad + \sum_{i\in[L],j\in\mathsf{Mal}} b_i \cdot \widehat{\mathrm{S}}\widehat{\mathrm{B}}_j(r^*_j + t^*_j\widehat{\mathrm{R}}_j)/\widehat{\mathrm{B}}_i + \sum_{i,j\in[L]} b_i \cdot \langle \boldsymbol{y}_j, \widehat{\mathbf{A}}\rangle \cdot \widehat{\mathrm{S}}\widehat{\mathrm{B}}_j/\widehat{\mathrm{B}}_i \\
&= \sum_{i\in[L]\backslash(\mathsf{Mal}\cup\mathsf{Cor})} b_i \cdot \widehat{\mathrm{S}}\widehat{\mathrm{R}}_i + \sum_{i\in\mathsf{Cor}\backslash\mathsf{Mal}} b_i r_i \cdot \widehat{\mathrm{S}} + \sum_{i\in\mathsf{Mal}} b_i \cdot \widehat{\mathrm{S}}(r^*_i + t^*_i\widehat{\mathrm{R}}_i) + \sum_{i\in[L]} b_i \cdot \langle \boldsymbol{y}_i, \widehat{\mathbf{A}}\rangle \cdot \widehat{\mathrm{S}} + \Phi \qquad (14)
\end{aligned}$$

where we define Φ as

$$\begin{aligned}
\Phi &= \sum_{i\in[L],j\in[L]\backslash(\mathsf{Mal}\cup\mathsf{Cor}),i\neq j} b_i \cdot \widehat{\mathrm{S}}\widehat{\mathrm{B}}_j\widehat{\mathrm{R}}_j/\widehat{\mathrm{B}}_i + \sum_{i\in[L],j\in\mathsf{Cor}\backslash\mathsf{Mal},i\neq j} b_i r_j \cdot \widehat{\mathrm{S}}\widehat{\mathrm{B}}_j/\widehat{\mathrm{B}}_i \\
&\qquad + \sum_{i\in[L],j\in\mathsf{Mal},i\neq j} b_i \cdot \widehat{\mathrm{S}}\widehat{\mathrm{B}}_j(r^*_j + t^*_j\widehat{\mathrm{R}}_j)/\widehat{\mathrm{B}}_i + \sum_{i,j\in[L],i\neq j} b_i \cdot \langle \boldsymbol{y}_j, \widehat{\mathbf{A}}\rangle \cdot \widehat{\mathrm{S}}\widehat{\mathrm{B}}_j/\widehat{\mathrm{B}}_i. \qquad (15)
\end{aligned}$$

In the second line of Eq. (14), we use the fact that $L'_1[h_j]$ can be represented as $L'_1[h_j] = r^*_j\widehat{\mathrm{B}}_j + t^*_j\widehat{\mathrm{B}}_j\widehat{\mathrm{R}}_j$ for $j \in \mathsf{Mal}$ due to the change introduced in **Game 4** and in the last line, we single out the terms with $i = j$ from each of the summations in the second line and put other terms into Φ. In particular, f'_2 contains the term $\widehat{\mathrm{S}}\widehat{\mathrm{R}}_{i^*}$ with non-zero coefficient b_{i^*}, which appears in the leftmost summation in the last line of Eq. (14). It can be seen by inspection that the term cannot be canceled inside the above sum. We then show that neither of f'_1 nor f'_3 contains the monomial $\widehat{\mathrm{S}}\widehat{\mathrm{R}}_{i^*}$.

- It is easy to observe that f'_3 can be represented as a linear combination of $\widehat{\mathrm{S}}$, 1, and $\{\widehat{\mathrm{S}}\widehat{\mathrm{A}}_\ell\}_\ell$. Therefore, it does not contain the term $\widehat{\mathrm{S}}\widehat{\mathrm{R}}_{i^*}$.

- We then show that f'_1 cannot contain the term either. To show this, we consider each combination of $(X, Y) \in (V_1 \times V_2) \backslash (\{\widehat{\mathrm{C}}_2\} \times V_3)$ and show that the resulting element that is obtained by replacing $\widehat{\mathrm{C}}_2, \widehat{\mathrm{C}}_3$ that appear in $X \cdot Y$ with the polynomials C_2 and C_3 and then replacing $\{\widehat{\mathrm{R}}_i\}_{i \in \mathsf{Cor}}$ with $\{r_i\}_{i \in \mathsf{Cor}}$ does not contain the desired term $\widehat{\mathrm{S}}\widehat{\mathrm{R}}_{i^*}$. We first observe that X should be either $\widehat{\mathrm{C}}_2$ or $\widehat{\mathrm{C}}_3$, since otherwise the resulting element does not contain the multiplicative factor $\widehat{\mathrm{S}}$.
  - We first consider the case of $X = \widehat{\mathrm{C}}_2$. In this case, Y should be chosen from $V_2 \backslash V_3$. However, then the resulting terms are multiplied by $\widehat{\mathrm{B}}_i$ for some i and thus does not yield the term.
  - We then consider the case of $X = \widehat{\mathrm{C}}_3$. In this case, since $Y \in V_2$, the resulting element is multiplied by either $1/\widehat{\Gamma}$ or $\widehat{\mathrm{B}}_i/\widehat{\mathrm{B}}_j$ with $i \neq j$. Therefore, it does not yield the term.

Therefore, the monomial $\widehat{\mathrm{S}}\widehat{\mathrm{R}}_{i^*}$ that appears in f'_1 cannot be canceled by any term in f'_2 nor f'_3. This implies $f' = f'_1 + f'_2 + f'_3 \neq 0$ over $\mathbb{T}'$ as desired. This completes the proof. □

Lemma 6 *For any adversary, it holds that* $\Pr[\mathsf{E}_5] = \Pr[\mathsf{E}_6]$.

Proof. The only difference between Game 5 and Game 6 is in the way zero-test queries are answered. The only possible case when Game 6 behaves differently from Game 5 in $\mathcal{A}$'s view is the case where the following event occurs: $\mathcal{A}$ submits a handle for a polynomial $f \in \mathbb{T}$ to the zero-test oracle such that $b_i = 0$ for all $i \notin \mathsf{Cor} \cup \mathsf{Mal}$ and $\boldsymbol{d} \neq -\sum_{i \in \mathsf{Cor} \cup \mathsf{Mal}} b_i \boldsymbol{y}_i$, but the associated polynomial $f' \in \mathbb{T}'$ equals to 0 in $\mathbb{T}'$. Here, we prove that this cannot happen and thus the two games are in fact equivalent.

We divide f into f_1, f_2, and f_3 as in Eq. (6) and define the corresponding polynomials f'_1, f'_2, and f'_3 as in the proof of Lemma 5. Recall that f'_2 can be expressed as Eq. (14). Noting that $b_i = 0$ for all $i \notin \mathsf{Cor} \cup \mathsf{Mal}$, Eq. (14) can be (slightly) simplified as follows:

$$f'_2 = \sum_{i \in \mathsf{Cor} \cup \mathsf{Mal}} b_i \cdot \langle \boldsymbol{y}_i, \widehat{\mathbf{A}} \rangle \cdot \widehat{\mathrm{S}} + \underbrace{\sum_{i \in \mathsf{Cor} \backslash \mathsf{Mal}} b_i r_i \cdot \widehat{\mathrm{S}} + \sum_{i \in \mathsf{Mal}} b_i \cdot \widehat{\mathrm{S}}(r_i^* + t_i^* \widehat{\mathrm{R}}_i) + \Phi}_{:= \Psi}$$

where Φ is defined as in Eq. (15). We also have

$$f'_3 = d_0 \widehat{\mathrm{S}} + \langle \boldsymbol{d}, \widehat{\mathbf{A}} \rangle \cdot \widehat{\mathrm{S}} + \langle \boldsymbol{x}^*_{\mathsf{coin}}, \boldsymbol{d} \rangle,$$

which is easy to observe. We then show that f'_1 does not contain the monomial of the form $\widehat{\mathrm{S}}\widehat{\mathrm{A}}_\ell$ for any ℓ. This is sufficient to complete the proof, since

$$f' = f'_1 + f'_2 + f'_3 = \left\langle \boldsymbol{d} + \sum_{i \in \mathsf{Cor} \cup \mathsf{Mal}} b_i \cdot \boldsymbol{y}_i, \widehat{\mathbf{A}} \right\rangle \cdot \widehat{\mathrm{S}} + f'_1 + \Psi + d_0 \widehat{\mathrm{S}} + \langle \boldsymbol{x}^*_{\mathsf{coin}}, \boldsymbol{d} \rangle,$$

does not equal to 0 unless $\boldsymbol{d} + \sum_{i \in \mathsf{Cor} \cup \mathsf{Mal}} b_i \cdot \boldsymbol{y}_i = 0$, which can be seen by observing that $f'_1 + \Psi + d_0 \widehat{\mathrm{S}} + \langle \boldsymbol{x}^*_{\mathsf{coin}}, \boldsymbol{d} \rangle$ does not contain any term of the form $\widehat{\mathrm{S}}\widehat{\mathrm{A}}_\ell$.

We then move to show that f'_1 does not contain the term of the form $\widehat{\mathrm{S}}\widehat{\mathrm{A}}_\ell$ for any ℓ. To show this, we consider each combination of $(X, Y) \in (V_1 \times V_2) \backslash (\{\widehat{\mathrm{C}}_2\} \times V_3)$ and show that the resulting element in $\mathbb{T}'$ that is obtained by replacing $\widehat{\mathrm{C}}_2, \widehat{\mathrm{C}}_3$ that appear in $X \cdot Y$ with the polynomials C_2 and C_3 and then replacing $\{\widehat{\mathrm{R}}_i\}_{i \in \mathsf{Cor}}$ with $\{r_i\}_{i \in \mathsf{Cor}}$ does not contain the desired term $\widehat{\mathrm{S}}\widehat{\mathrm{A}}_\ell$. We first observe that X should be either $\widehat{\mathrm{C}}_2$ or $\widehat{\mathrm{C}}_3$, since otherwise the resulting element does not contain the multiplicative factor $\widehat{\mathrm{S}}$.

- We first consider the case of $X = \widehat{\mathrm{C}}_2$. In this case, Y should be chosen from $V_2 \backslash V_3$. However, then the resulting terms are multiplied by $\widehat{\mathrm{B}}_i$ for some i and thus does not yield the term.
- We then consider the case of $X = \widehat{\mathrm{C}}_3$. In this case, since $Y \in V_2$, the resulting element is multiplied by either $1/\widehat{\Gamma}$ or $\widehat{\mathrm{B}}_i/\widehat{\mathrm{B}}_j$ with $i \neq j$. Therefore, it does not yield the term.

As discussed above, this completes the proof. □

Lemma 7 *For any adversary, it holds that* $\Pr[\mathsf{E}_6] = 1/2$.

Proof. We show that answer to any zero-test query made by $\mathcal{A}$ does not depend on the challenge bit coin and therefore the view of $\mathcal{A}$ is independent from the value of coin. To see this, let us consider $f \in \mathbb{T}$ that corresponds to a zero-test query made by $\mathcal{A}$ during the game and prove that f' refers to the same element in $\mathbb{T}'$ regardless of the value of coin if f satisfies the restrictions that are introduced in **Game 5** and **Game 6**. For the analysis, we divide f into f_1, f_2, and f_3 as in Eq. (6) and define the corresponding polynomials f'_1, f'_2, and f'_3 as in the proof of Lemma 5.

We first claim that regardless of the value of coin, f'_1 and f'_2 refer to the same elements in $\mathbb{T}'$. This can be easily seen by observing that f_1 and f_2 do not include the formal variables $\{\widehat{\mathrm{C}}_{1,\ell}\}_\ell$, which are the only variables that depend on the value of coin when they are replaced with the polynomials in $\mathbb{T}'$.

We then prove the same statement for f'_3. We observe:

$$f'_3 = d_0\widehat{\mathrm{S}} + \langle \boldsymbol{d}, \widehat{\mathbf{A}} \rangle \cdot \widehat{\mathrm{S}} + \langle \boldsymbol{x}^*_{\mathsf{coin}}, \boldsymbol{d} \rangle = d_0\widehat{\mathrm{S}} + \langle \boldsymbol{d}, \widehat{\mathbf{A}} \rangle \cdot \widehat{\mathrm{S}} - \sum_{i \in \mathsf{Cor} \cup \mathsf{Mal}} b_i \cdot \langle \boldsymbol{x}^*_{\mathsf{coin}}, \boldsymbol{y}_i \rangle.$$

Since $\langle \boldsymbol{x}^*_0, \boldsymbol{y}_i \rangle = \langle \boldsymbol{x}^*_1, \boldsymbol{y}_i \rangle$ for all $i \in \mathsf{Cor} \cup \mathsf{Mal}$ by the admissibility condition, f'_3 refers to the same element in $\mathbb{T}'$ regardless of whether $\mathsf{coin} = 0$ or $\mathsf{coin} = 1$.

Since each of f'_1, f'_2, and f'_3 refers to the same element in $\mathbb{T}'$ regardless of whether $\mathsf{coin} = 0$ or $\mathsf{coin} = 1$, the same holds for $f' = f'_1 + f'_2 + f'_3$. This completes the proof. □

Finally, the proof of Theorem 1 follows by combining the proofs of Lemma 1 to 7. □

Registered IPFE from pairings: Plugging our slotted registered IPFE into the transformation described in Section 9, we achieve the following corollary:

Corollary 1 (Bounded Registered IPFE) Let λ be a security parameter. Then, under generic bilinear group model, for every polynomial $L = L(\lambda)$ and an integer $n \in \mathbb{N}$, there exists a bounded registered IPFE scheme with function space $\mathcal{U}_F = \mathbb{Z}^n$, message space $\mathcal{M} = \mathbb{Z}^n$, and supporting up to L users with the following properties:

- The size of the common reference string and the size of the auxiliary data maintained by the key curator is $L^2 \cdot \mathsf{poly}(\lambda, n, \log L)$.
- The running times of key-generation and registration are $L \cdot \mathsf{poly}(\lambda, \log L)$ and $L \cdot \mathsf{poly}(\lambda, n, \log L)$ respectively.
- The size of the master public key and the helper decryption keys are both $\mathsf{poly}(\lambda, n, \log L)$.
- The size of a ciphertext is $\mathsf{poly}(\lambda, n, \log L)$.

7 Slotted Registered ABIPFE from Pairing

In this section, we present our slotted registered attribute-based IPFE scheme based on pairing. The attribute-based access control is provided by the policies represented by linear secret sharing access structures (LSSS). In many well-known ABE schemes [GPSW06, LOS+10b, LW11b] the access control is delivered by LSSS which also captures the class of monotone Boolean formulas [LW11a] as a special case. Before going ahead, we first present the formal definitions of access structures and linear secret-sharing schemes.

Definition 5 (Access Structures [Bei96]) Let $\mathcal{U}_{\mathsf{att}}$ be the attribute universe. An access structure on $\mathcal{U}_{\mathsf{att}}$ is a collection $\mathbb{A} \subseteq 2^{\mathcal{U}_{\mathsf{att}}} \setminus \emptyset$ of non-empty sets of attributes. The sets in $\mathbb{A}$ are called the *authorized* sets and the sets not in $\mathbb{A}$ are called the *unauthorized* sets. An access structure is called *monotone* if $\forall B, C \in 2^{\mathcal{U}_{\mathsf{att}}}$ if $B \in \mathbb{A}$ and $B \subseteq C$, then $C \in \mathbb{A}$.

Definition 6 (Linear Secret Sharing Schemes (LSSS) [Bei96]) Let $q = q(\lambda)$ be a prime and $\mathcal{U}_{\mathsf{att}}$ the attribute universe. For each monotone access structure $\mathbb{A}$ on $\mathcal{U}_{\mathsf{att}}$, there exists a matrix $\mathbf{M} \in \mathbb{Z}_q^{\ell \times n}$, called the share-generating matrix, and a function $\rho\colon [\ell] \to \mathcal{U}_{\mathsf{att}}$, that labels the rows of $\mathbf{M}$ with attributes from $\mathcal{U}_{\mathsf{att}}$. A secret sharing scheme with domain of secrets $\mathbb{Z}_p$ for a monotone access structure $\mathbb{A}$ over $\mathcal{U}_{\mathsf{att}}$, a.k.a. a monotone secret sharing scheme, is a randomized algorithm that has the following properties:

- **Share generation.** To share a secret $s \in \mathbb{Z}_p$, we define the vector $\boldsymbol{v} = (s, v_2, ..., v_n)$, by sampling $v_2, \ldots, v_n \leftarrow \mathbb{Z}_p$. Then the vector of ℓ shares of the secret s is given by $\boldsymbol{u} = \mathbf{M}\boldsymbol{v}^\top \in \mathbb{Z}_p^{\ell \times 1}$, where for all $i \in [\ell]$ the share u_i "belongs" to a party holding the attribute $\rho(i)$. We will be referring to the pair $(\mathbf{M}, \rho)$ as the LSSS policy of the access structure $\mathbb{A}$.
- **Reconstruction of secret and soundness.** Let S (resp. S') denote an authorized (resp. unauthorized) set of attributes according to some monotone access structure $\mathbb{A}$ and let $\mathcal{I}$ (resp. $\mathcal{I}'$) be the set of rows of the share generating matrix $\mathbf{M}$ of the LSSS policy pair $(\mathbf{M}, \rho)$ associated with $\mathbb{A}$ whose labels are in S (resp. S'). For reconstruction, there exist constants $\{w_i\}_{i\in\mathcal{I}}$ in $\mathbb{Z}_p$ such that for any valid shares $\{u_i = (\mathbf{M}\boldsymbol{v}^\top)_i\}_{i\in\mathcal{I}}$ of a secret $s \in \mathbb{Z}_p$, it is true that $\sum_{i\in\mathcal{I}} w_i u_i = s$ (equivalently, $\sum_{i\in\mathcal{I}} w_i \mathbf{M}_i = (1, \overbrace{0, \ldots, 0}^{n-1})$, where $\mathbf{M}_i$ is the ith row of $\mathbf{M}$ or $(1, 0, \ldots, 0) \in \mathsf{Span}\{\mathbf{M}_{\rho(i)} : \rho(i) \in S\}$). For soundness, there are no such w_i's, as above or equivalently, $(1, 0, \ldots, 0) \notin \mathsf{Span}\{\mathbf{M}_{\rho(i)} : \rho(i) \in S'\}$.

Remark 2 (One-use restriction) In this work, we construct a registered ABIPFE scheme where policies are represented by a linear secret sharing scheme (Definition 6), with the restriction that each attribute is associated with at most one row of the associated LSSS matrix $\mathbf{M}$. In other words, the row-labelling function ρ of the policies $(\mathbf{M}, \rho)$ is injective. It is well-known that a scheme with the one-use restriction can be extended into one where attributes can be used up to k times by expanding the public parameters and secret keys by a factor of k via a simple encoding (Lewko et al. [LOS+10a]).

7.1 Construction

The slotted registered attribute-based IPFE $\mathsf{SlotRABIPFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{IsValid}, \mathsf{Aggregate}, \mathsf{Enc}, \mathsf{Dec})$ for an attribute universe $\mathcal{U}_{\mathsf{att}}$, a set of policies $\mathcal{P}$ which contains (one-use) LSSS policies of a monotone access structure on $\mathcal{U}_{\mathsf{att}}$, and a function space $\mathcal{U}_F = \mathsf{PSet}(\mathcal{U}_{\mathsf{att}}) \times \mathbb{Z}^n$, message space $\mathcal{M} = \mathcal{P} \times \mathbb{Z}^n$ works as follows:

Setup$(1^\lambda, 1^n, \mathcal{U}_{\mathsf{att}}, L)$: The setup algorithm takes the security parameter λ, the length n of vectors (in unary), the attribute universe $\mathcal{U}_{\mathsf{att}}$ and the number of users L (in binary) as inputs and samples $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, e) \leftarrow \mathsf{GG}(1^\lambda)$. The algorithm computes the following terms:

1. Sample $\boldsymbol{\alpha} \leftarrow \mathbb{Z}_p^n, \gamma, \pi, \beta_i \leftarrow \mathbb{Z}_p$ for all $i \in [L]$.
2. Compute $\mathbf{Z} \coloneqq g_T^{\boldsymbol{\alpha}}$ and $\Gamma \coloneqq g_1^{1/\gamma}, \Pi \coloneqq g_1^{1/\pi}$ where $g_T = e(g_1, g_2)$.
3. For each $i \in [L]$, compute $\mathbf{A}_i \coloneqq g_1^{\beta_i \boldsymbol{\alpha}}, B_i \coloneqq g_1^{\beta_i}, \widetilde{B}_i \coloneqq g_2^{\beta_i}, D_i \coloneqq g_2^{1/\beta_i}$.
4. For each slot $i, j \in [L]$ and $i \neq j$, compute $R_{i,j} \coloneqq g_2^{\gamma\beta_i/\beta_j}, \mathbf{S}_{i,j} \coloneqq g_2^{\gamma\beta_i\boldsymbol{\alpha}/\beta_j}$.
5. Sample $t_{i,w} \leftarrow \mathbb{Z}_p$ for all $i \in [L], w \in \mathcal{U}_{\mathsf{att}}$.
6. For all $i, j \in [L]$ with $i \neq j$ and $w \in \mathcal{U}_{\mathsf{att}}$, compute $T_{i,w} \coloneqq B_i^{t_{i,w}}, H_{i,j,w} \coloneqq g_2^{\pi\beta_i t_{i,w}/\beta_j}$.
7. Output the common reference string as

$$\mathsf{crs} \coloneqq \left(\begin{array}{c} \mathcal{G}, \quad \mathbf{Z} = g_T^{\boldsymbol{\alpha}}, \quad \Gamma = g_1^{1/\gamma}, \ \Pi = g_1^{1/\pi}, \\ \left\{ \mathbf{A}_i = g_1^{\beta_i\boldsymbol{\alpha}}, \ B_i = g_1^{\beta_i}, \ \{T_{i,w} = g_1^{\beta_i t_{i,w}}\}_{w\in\mathcal{U}_{\mathsf{att}}}, \ \widetilde{B}_i = g_2^{\beta_i}, \ D_i = g_2^{1/\beta_i} \right\}_{i\in[L]}, \\ \left\{ R_{i,j} = g_2^{\gamma\beta_i/\beta_j}, \ \mathbf{S}_{i,j} = g_2^{\gamma\beta_i\boldsymbol{\alpha}/\beta_j}, \ H_{i,j,w} = g_2^{\pi\beta_i t_{i,w}/\beta_j} \right\}_{\substack{i,j\in[L], i\neq j, \\ w\in\mathcal{U}_{\mathsf{att}}}} \end{array} \right),$$

KeyGen(crs, i) : The key generation algorithm takes the common reference string crs, and a slot index $i \in [L]$ as inputs and works as follows:

1. Sample $r_i \leftarrow \mathbb{Z}_p$ and compute $U_i \coloneqq B_i^{r_i}, \widetilde{U} = \widetilde{B}_i^{r_i}, P_{i,j} \coloneqq R_{i,j}^{r_i}$ for all $j \in [L]$ and $j \neq i$.
2. Output the public and secret keys as

$$\mathsf{pk}_i \coloneqq \Big(U_i = g_1^{\beta_i r_i}, \widetilde{U}_i = g_2^{\beta_i r_i}, \ \ \{P_{i,j} = g_2^{\gamma \beta_i r_i / \beta_j}\}_{j \in [L], j \neq i} \Big) \ \text{ and } \ \mathsf{sk}_i \coloneqq r_i.$$

IsValid$(\mathsf{crs}, i, \mathsf{pk}_i)$: The public key verification algorithm takes the common reference string crs, a slot index $i \in [L]$ and a public key $\mathsf{pk}_i = (U_i, \{P_{i,j}\}_{j \in [L], j \neq i})$, and checks the following:

$$e(U_i, g_2) \stackrel{?}{=} e(g_1, \widetilde{U}_i) \quad \text{and}$$
$$e(U_i, D_j) \stackrel{?}{=} e(\Gamma, P_{i,j}) \quad \forall j \in [L] \setminus \{i\}.$$

If the check passes then it outputs 1; otherwise 0.

Aggregate$(\mathsf{crs}, (\mathsf{pk}_1, \mathsf{Att}_1, \boldsymbol{y}_1), \ldots, (\mathsf{pk}_L, \mathsf{Att}_L, \boldsymbol{y}_L))$: The aggregate algorithm takes a common reference string crs, a list of L public key, attribute, function tuple $(\mathsf{pk}_1, \mathsf{Att}_1, \boldsymbol{y}_1), \ldots, (\mathsf{pk}_L, \mathsf{Att}_L, \boldsymbol{y}_L)$ as inputs such that $\mathsf{Att}_i \subseteq \mathcal{U}_{\mathsf{att}}, \boldsymbol{y}_i \in \mathbb{Z}^n$ and $\mathsf{pk}_i = (U_i, \widetilde{U}_i, \{P_{i,j}\}_{j \in [L], j \neq i})$ for all $i \in [L]$. It proceeds as follows:

1. Using $\mathbf{A}_i$, U_i and $\boldsymbol{y}_i$, compute $W_i \coloneqq U_i \prod_{\ell \in [n]} A_{i,\ell}^{y_{i,\ell}} = g_1^{\beta_i (r_i + \langle \boldsymbol{y}_i, \boldsymbol{\alpha} \rangle)}$, where $A_{i,\ell}$ denotes the ℓ-th entry of $\mathbf{A}_i$.
2. Using $\mathbf{S}_{i,j}$ and $\boldsymbol{y}_i$, compute $S_{j,i} = \prod_{\ell \in [n]} S_{j,i,\ell}^{y_{j,\ell}} = g_2^{\gamma \beta_j \langle \boldsymbol{y}_i, \boldsymbol{\alpha} \rangle / \beta_i}$ for all $i, j \in [L]$ and $j \neq i$, where $S_{j,i,\ell}$ denotes the ℓ-th entry of $\mathbf{S}_{j,i}$.
3. Compute the component of MPK as $W = \prod_{i \in [L]} W_i = \prod_{i \in [L]} g_1^{\beta_i (r_i + \langle \boldsymbol{y}_i, \boldsymbol{\alpha} \rangle)}$.
4. Compute the component of hsk_i as $F_i = S_i \cdot P_i$ where

$$S_i \coloneqq \prod_{j \in [L] \setminus \{i\}} S_{j,i} = \prod_{j \in [L] \setminus \{i\}} g_2^{\gamma \beta_j \langle \boldsymbol{y}_i, \boldsymbol{\alpha} \rangle / \beta_i}, \quad P_i \coloneqq \prod_{j \in [L] \setminus \{i\}} P_{j,i} = \prod_{j \in [L] \setminus \{i\}} g_2^{\gamma \beta_i r_i / \beta_j}$$

.

5. For each $w \in \mathcal{U}_{\mathsf{att}}, i \in [L]$, compute

$$T_w = \prod_{j \in [L] : w \notin \mathsf{Att}_j} T_{j,w} = \prod_{j \in [L] : w \notin \mathsf{Att}_j} g_1^{\beta_j t_{j,w}}, \quad H_{i,w} = \prod_{j \neq i : w \notin \mathsf{Att}_j} H_{j,i,w} = \prod_{j \neq i : w \notin \mathsf{Att}_j} g_2^{\pi \beta_j t_{j,w} / \beta_i}.$$

6. Output the master public key and slot-specific helper decryption keys as

$$\mathsf{MPK} \coloneqq (\mathcal{G}, \ \ \mathbf{Z}, \Gamma, \ \ W, \ \ \{T_w\}_{w \in \mathcal{U}}) \ \text{ and } \ \mathsf{hsk}_i \coloneqq (\mathsf{Att}_i, D_i, \ F_i, \ \{H_{i,w}\}_{w \in \mathcal{U}_{\mathsf{att}}}) \, .$$

Enc$(\mathsf{MPK}, (\mathbf{M}, \rho), \boldsymbol{x})$: The encryption algorithm takes a master public key MPK, a policy $(\mathbf{M} \in \mathbb{Z}_p^{K \times N}, \rho : [K] \to \mathcal{U}_{\mathsf{att}})$ where ρ is an injective function mapping the row indices of $\mathbf{M}$ into the attributes in $\mathcal{U}_{\mathsf{att}}$ and a message $\boldsymbol{x} \in \mathbb{Z}^n$ as inputs and proceeds as follows:

1. Sample $h \leftarrow \mathbb{G}_1$ and $s \leftarrow \mathbb{Z}_p$.
2. Compute $C_0 \coloneqq g_T^s$ and $\mathbf{C}_1 \coloneqq (g_T^{x_1} \cdot Z_1^s, \ldots, g_T^{x_n} \cdot Z_n^s) = g_T^{\boldsymbol{x} + s\boldsymbol{\alpha}}$ where Z_ℓ denotes the ℓ-th entry of $\mathbf{Z}$.
3. Compute $C_2 \coloneqq h^{-s} W^{-s}$, $C_3 \coloneqq \Gamma^s$ and $C_4 \coloneqq \Pi^s$.
4. Sample $v_2, \ldots, v_N$ and set $\boldsymbol{v} \coloneqq (s, v_2, \ldots, v_N)$.
5. For each $k \in [K]$, compute $C_{5,k} \coloneqq h^{\langle \boldsymbol{v}, \boldsymbol{m}_k \rangle} T_{\rho(k)}^{-s}$.

6. Output the ciphertext

$$\mathsf{ct} \coloneqq \left((\mathbf{M}, \rho),\ C_0, \mathbf{C}_1, C_2, C_3, C_4, \{C_{5,k}\}_{k\in[K]}\right).$$

Dec$(\mathsf{sk}, \mathsf{hsk}, \mathsf{ct})$: The decryption algorithm takes a secret key $\mathsf{sk} = r$, a helper decryption key $\mathsf{hsk} = (\mathsf{Att}_i, D_i, F_i, \{H_{i,w}\}_{w\in\mathcal{U}_{\mathsf{att}}})$ for the i-th slot and a ciphertext $\mathsf{ct} \coloneqq ((\mathbf{M}, \rho),\ C_0, \mathbf{C}_1, C_2, C_3, C_4, \{C_{5,k}\}_{k\in[K]})$ as inputs and works as follows. If Att_i does not satisfy the policy $(\mathbf{M}, \rho)$ then output $\perp$. Otherwise, there exists $\boldsymbol{\omega} \in \mathbb{Z}_p^{|I|}$ such that $\boldsymbol{\omega}^\top \mathbf{M}_{\mathsf{Att}_i} = \boldsymbol{e}_1^\top$ where $I = \{k \in [K] : \rho(k) \in \mathsf{Att}_i\} = \{k_\iota : \iota \in [|I|]\}$ and $\mathbf{M}_{\mathsf{Att}_i}$ is formed by taking the subset of rows of $\mathbf{M}$ indexed by I.

1. Compute the following terms

$$E_{\mathsf{slot}} = e(C_2, D_i) \cdot e(C_3, F_i) \cdot C_0^{\mathsf{sk}},$$
$$E_{\mathsf{att}} = \prod_{\iota\in[|I|]} \left(e(C_{5,k_\iota}, D_i) \cdot e(C_4, H_{i,\rho(k_\iota)})\right)^{\omega_\iota} \text{ and } C = \prod_{\ell\in[n]} C_{1,\ell}^{y_{i,\ell}}$$

   where $C_{1,\ell}$ denotes the ℓ-th entry of $\mathbf{C}_1$.
2. Output the message as $\log_{g_T}(C \cdot E_{\mathsf{slot}} \cdot E_{\mathsf{att}})$.

Completeness: Consider a key pair $(\mathsf{pk}_i, \mathsf{sk}_i)$ generated using $\mathsf{KeyGen}(\mathsf{crs}, i; r)$. Then by construction, we have $\mathsf{pk} = (U_i, \widetilde{U}_i, \{P_{i,j}\}_{j\in[L], j\neq i})$ where

$$U_i = B_i^{r_i} = g_1^{\beta_i r_i},\ \widetilde{U}_i = \widetilde{B}_i^{r_i} = g_2^{\beta_i r_i} \text{ and } P_{i,j} = R_{i,j}^{r_i} = g_2^{\gamma\beta_i r_i/\beta_j}.$$

Therefore, the validity of pk_i is verified using

$$e(U_i, g_2) = e(g_1, g_2)^{\beta_i r_i} = e(g_1, \widetilde{U}_i) \quad \text{and}$$
$$e(U_i, D_j) = e(g_1, g_2)^{\beta_i r_i/\beta_j} = e(\Gamma, P_{i,j}) \quad \forall j \in [L] \setminus \{i\}$$

since $D_j = g_2^{1/\beta_j}$ and $\Gamma = g_1^{1/\gamma}$. The RAB-IPFE satisfies completeness since the public key passes all the pairing equations defined by the $\mathsf{IsValid}$ algorithm, *i.e.*, $\mathsf{IsValid}(\mathsf{crs}, \mathsf{pk}_i)$ outputs 1.

Correctness: Consider a secret key $\mathsf{sk} = r_i$, a helper decryption key $\mathsf{hsk} = (\mathsf{Att}_i, D_i, F_i, \{H_{i,w}\}_{w\in\mathcal{U}_{\mathsf{att}}})$ and a ciphertext $\mathsf{ct} \coloneqq ((\mathbf{M}, \rho),\ C_0, \mathbf{C}_1, C_2, C_3, C_4, \{C_{5,k}\}_{k\in[K]})$. Then, by construction, we have

$$F_i = S_i \cdot P_i = \prod_{j\neq i} g_2^{\gamma r_j \beta_j/\beta_i} \cdot \prod_{\ell\in[n]} \prod_{j\neq i} g_2^{\gamma r_j y_{j,\ell} \alpha_\ell \beta_j/\beta_i} = \prod_{j\neq i} g_2^{\gamma r_j \beta_j/\beta_i} \cdot \prod_{j\neq i} g_2^{\gamma r_j \langle \boldsymbol{y}_j, \boldsymbol{\alpha}\rangle \beta_j/\beta_i},$$
$$C_2 = h^{-s} \prod_{i\in[L]} W_i^{-s} = h^{-s} \prod_{i\in[L]} U_i^{-s} \prod_{i\in[L]} \cdot \prod_{\ell\in[n]} A_{i,\ell}^{-s y_{i,\ell}} = h^{-s} \prod_{i\in[L]} g_1^{-s r_i \beta_i} \cdot \prod_{i\in[L]} g_1^{-s\langle \boldsymbol{y}_i, \boldsymbol{\alpha}\rangle \beta_i},$$

and then we compute

$$e(C_2, D_i) = e(h, g_2)^{-s/\beta_i} \cdot \prod_{j\in[L]} e(g_1, g_2)^{-s r_j \beta_j/\beta_i} \cdot \prod_{j\in[L]} e(g_1, g_2)^{-s\langle \boldsymbol{y}_j, \boldsymbol{\alpha}\rangle \beta_j/\beta_i},$$
$$e(C_3, F_i) = \prod_{j\neq i} e(g_1, g_2)^{-s r_j \beta_j/\beta_i} \cdot \prod_{j\neq i} e(g_1, g_2)^{-s r_j \langle \boldsymbol{y}_j, \boldsymbol{\alpha}\rangle \beta_j/\beta_i},$$

Therefore,

$$E_{\mathsf{slot}} = e(C_2, D_i) \cdot e(C_3, F_i) \cdot C_0^{\mathsf{sk}_i} = e(h, g_2)^{-s/\beta_i} \cdot e(g_1, g_2)^{-s\langle \boldsymbol{y}_i, \boldsymbol{\alpha}\rangle}$$

Next, we check that the attributes of Att_i are satisfying the policy $(\mathbf{M}, \rho)$. First, by construction, we have

$$C_{5,k} = h^{\langle \boldsymbol{v}, \boldsymbol{m}_k \rangle} T_{\rho(k)}^{-s} = h^{\langle \boldsymbol{v}, \boldsymbol{m}_k \rangle} \cdot \prod_{j \in [L]: \rho(k) \notin \mathsf{Att}_j} g_1^{-s\beta_j t_{j,\rho(k)}},$$

$$H_{i,\rho(k)} = \prod_{j \in [L] \setminus \{i\}: \rho(k) \notin \mathsf{Att}_j} H_{j,i,w} = \prod_{j \in [L] \setminus \{i\}: \rho(k) \notin \mathsf{Att}_j} g_2^{\pi\beta_j t_{j,\rho(k)}/\beta_i},$$

and then we compute

$$e(C_{5,k}, D_i) = e(h, g_2)^{\langle \boldsymbol{v}, \boldsymbol{m}_k \rangle / \beta_i} \cdot \prod_{j \in [L]: \rho(k) \notin \mathsf{Att}_j} e(g_1, g_2)^{-s\beta_j t_{j,\rho(k)}/\beta_i},$$

$$e(C_4, H_{i,\rho(k)}) = \prod_{j \in [L] \setminus \{i\}: \rho(k) \notin \mathsf{Att}_j} e(g_1, g_2)^{s\beta_j t_{j,\rho(k)}/\beta_i}.$$

Let $I = \{k \in [K] : \rho(k) \in \mathsf{Att}_i\} = \{k_1, \ldots, k_{|I|}\}$. Then for all $\iota \in [|I|]$, we have $\rho(k_\iota) \in \mathsf{Att}_i$ and hence

$$\prod_{j \in [L]: \rho(k_\iota) \notin \mathsf{Att}_j} e(g_1, g_2)^{-s\beta_j t_{j,\rho(k_\iota)}/\beta_i} = \prod_{j \in [L] \setminus \{i\}: \rho(k_\iota) \notin \mathsf{Att}_j} e(g_1, g_2)^{-s\beta_j t_{j,\rho(k_\iota)}/\beta_i}.$$

Therefore, we can write

$$e(C_{5,k_\iota}, D_i) \cdot e(C_4, H_{i,\rho(k_\iota)}) = e(h, g_2)^{\langle \boldsymbol{v}, \boldsymbol{m}_{k_\iota} \rangle / \beta_i}.$$

Now, if Att_i satisfies the policy then there exists $\boldsymbol{\omega} \in \mathbb{Z}_p^{|I|}$ such that $\boldsymbol{\omega}^\top \mathbf{M}_{\mathsf{Att}_i} = \boldsymbol{e}_1^\top$. So, we get

$$\begin{aligned} E_{\mathsf{att}} = \prod_{\iota \in [|I|]} \left(e(C_{5,k_\iota}, D_i) \cdot e(C_4, H_{i,\rho(k_\iota)}) \right)^{\omega_\iota} &= e(h, g_2)^{\langle \boldsymbol{v}, \sum_\iota \omega_\iota \boldsymbol{m}_{k_\iota} \rangle / \beta_i} \\ &= e(h, g_2)^{\langle \boldsymbol{v}, \boldsymbol{\omega}^\top \mathbf{M}_{S_i} \rangle / \beta_i} \\ &= e(h, g_2)^{\langle \boldsymbol{v}, \boldsymbol{e} \rangle / \beta_i} = e(h, g_2)^{s/\beta_i}. \end{aligned}$$

Next, we compute $C = \prod_{\ell \in [n]} C_{1,\ell}^{y_{i,\ell}} = g_T^{\langle \boldsymbol{x}, \boldsymbol{y}_i \rangle + s\langle \boldsymbol{y}_i, \boldsymbol{\alpha} \rangle}$. Finally, the inner product value is obtained as $\log_{g_T}(C \cdot E_{\mathsf{slot}} \cdot E_{\mathsf{att}}) = \langle \boldsymbol{x}, \boldsymbol{y}_i \rangle$.

Compactness: The master public key contains $O(n + |\mathcal{U}_{\mathsf{att}}|)$ group elements and each group element can be represented using $\mathsf{poly}(\lambda)$ bits. Therefore, the master public key size is bounded by $\mathsf{poly}(\lambda, |\mathcal{U}_F|, \log L)$ where $|\mathcal{U}_F| = n + |\mathcal{U}_{\mathsf{att}}|$. The helper decryption key contains $O(|\mathcal{U}_{\mathsf{att}}|)$ group element. Since the information of aggregated master public key is given with the helper decryption key the size of it is also bounded by $\mathsf{poly}(\lambda, |\mathcal{U}_F|, \log L)$.

7.2 Security Analysis

Theorem 2 *The slotted registered attribute-based inner product functional encryption scheme is secure in the generic group model.*

Proof. Let a PPT adversary against our slotted registered ABIPFE be $\mathcal{A}$. Recall that it is sufficient to assume that there are no post-challenge queries from $\mathcal{A}$. Let Cor be the set of all corrupted slots, for which an honest key pair is generated by the challenger and later is corrupted by $\mathcal{A}$ and the corresponding secret key is revealed to $\mathcal{A}$. Let Mal be the set of slots for which malicious keys are provided by $\mathcal{A}$ itself. At the beginning, Cor is set as an empty set and it is dynamically filled with the corrupted slot indices during the course of the security experiment. Similarly, Mal is also set as an empty set at the beginning of the game. It is defined during the challenge phase based on the keys provided by $\mathcal{A}$.

We consider the following sequence of hybrid games played between the adversary $\mathcal{A}$ and the challenger. Let E_i be the event of $\mathcal{A}$ outputting the correct bit coin in Game i.

Game 0**:** This is the real experiment in the generic group model. We also assume that the total number of zero-test queries made by $\mathcal{A}$ is Q_{zt}. The challenger simulates $\mathcal{A}$ as follows.

Setup. The challenger samples $\boldsymbol{\alpha} \leftarrow \mathbb{Z}_p^n, \gamma, \pi, \beta_i, t_{i,w} \leftarrow \mathbb{Z}_p$ for all $i \in [L]$ and $w \in \mathcal{U}_{\mathsf{att}}$. It creates the following lists of group exponents:

- $L_1 = \{1, 1/\gamma, 1/\pi, \{\beta_i \boldsymbol{\alpha}, \beta_i, \{\beta_i t_{i,w}\}_{w \in \mathcal{U}_{\mathsf{att}}}\}_{i \in [L]}\}$ which corresponds to the $\mathbb{G}_1$-elements in the CRS.
- $L_2 = \{1, \{\beta_i, 1/\beta_i\}_{i \in [L]}, \{\gamma\beta_i/\beta_j, \gamma\beta_i\boldsymbol{\alpha}/\beta_j, \{\pi\beta_i t_{i,w}/\beta_j\}_{w \in \mathcal{U}_{\mathsf{att}}}\}_{i,j \in [L], i \neq j}\}$ which corresponds to the $\mathbb{G}_2$-elements in the CRS.
- $L_T = \{1, \boldsymbol{\alpha}\}$ which corresponds to the $\mathbb{G}_T$-elements in the CRS.

The adversary is given handles corresponding to all the elements of lists L_1, L_2 and L_T.

Pre-challenge query phase. The challenger initializes a dictionary D, a set of corrupted set $\mathsf{Cor} \leftarrow \emptyset$, a set of indices for which maliciously generated keys $\mathsf{Mal} \leftarrow \emptyset$, and set $\mathsf{ctr} \leftarrow 0$. $\mathcal{A}$ makes the following queries:

1. *Key generation queries.* $\mathcal{A}$ specifies a slot index i. Then, the challenger samples $r_i \leftarrow \mathbb{Z}_p$, sets $\mathsf{sk}_{\mathsf{ctr}} = r_i$, updates the lists as
   - $L_1 \leftarrow L_1 \cup \{\beta_i r_i\}$,
   - $L_2 \leftarrow L_2 \cup \{\beta_i r_i, \{\gamma\beta_i r_i/\beta_j\}_{i,j \in [L], i \neq j}\}$,

   and provides $\mathcal{A}$ with the handles of newly computed group elements. The challenger increments $\mathsf{ctr} \leftarrow \mathsf{ctr} + 1$ and adds the mapping $\mathsf{ctr} \mapsto (i, h_{\mathsf{pk}_{\mathsf{ctr}}}, \mathsf{sk}_{\mathsf{ctr}})$ to the dictionary D. Here, $h_{\mathsf{pk}_{\mathsf{ctr}}}$ is the set of handles corresponding to the newly added elements in the list and is of the form $\{(1, h_i), (2, \widetilde{h}_i), \{(2, h_{i,j})\}_{j \in [L] \setminus \{i\}}\}$ with $L_1[h_i] = L_2[h_i] = \beta_i r_i$ and $L_2[h_{i,j}] = \gamma\beta_i r_i/\beta_j$.[14]
2. *Corruption queries.* In a corruption query, $\mathcal{A}$ specifies an index $1 \leq c \leq \mathsf{ctr}$. The challenger looks up the tuple $(i', h_{\mathsf{pk}'}, \mathsf{sk}') \leftarrow \mathsf{D}[c]$ and sends sk' to $\mathcal{A}$. It updates the set of corrupted indices as $\mathsf{Cor} \leftarrow \mathsf{Cor} \cup \{i'\}$.

Challenge phase. For each slot $i \in [L]$, $\mathcal{A}$ specifies a tuple $(c_i, h_{\mathsf{pk}_i^*}, \mathsf{Att}_i, \boldsymbol{y}_i)$ where either $c_i \in \{1, \ldots, \mathsf{ctr}\}$ to reference a challenger-generated key or $c_i = \perp$ to reference a key outside this set. Here, we denote $h_{\mathsf{pk}_i^*} = \{(1, h_i^*), (2, \widetilde{h}_i^*), \{(2, h_{i,j}^*)\}_{j \in [L] \setminus \{i\}}\}$ by the list of handles associated to the public key of the i-th slot provided by $\mathcal{A}$. $\mathcal{A}$ also specifies a challenge policy $(\mathbf{M} \in \mathbb{Z}_p^{K \times N}, \rho : [K] \to \mathcal{U}_{\mathsf{att}})$ and two challenge messages $\boldsymbol{x}_0^*, \boldsymbol{x}_1^*$.

- If $c_i \in \{1, \ldots, \mathsf{ctr}\}$, then the challenger looks up the entry $\mathsf{D}[c_i] = (i', h_{\mathsf{pk}'}, \mathsf{sk}')$. If $i = i'$, then the challenger sets $h_{\mathsf{pk}_i} \leftarrow h_{\mathsf{pk}'}$. Otherwise, if $i \neq i'$, then the experiment halts.
- If $c_i = \perp$, then the challenger runs $\mathsf{IsValid}$ on the GGM. In more detail, it checks whether the following equations hold by the zero test oracle:

$$L_1[h_i^*] - L_2[\widetilde{h}_i^*] \equiv 0 \bmod p, \tag{16}$$

$$L_1[h_i^*]/\beta_j - L_2[h_{i,j}^*]/\gamma \equiv 0 \bmod p \quad \forall j \in [L] \setminus \{i\}. \tag{17}$$

  If Equation (16) or (17) does not hold, the experiment halts. If the key is valid, the challenger sets $h_{\mathsf{pk}_i} \leftarrow h_{\mathsf{pk}_i^*}$, adds the slot index i to Mal.

[14] Recall that in the GGM, the adversary can only access the handles corresponding to the group elements in $\mathsf{pk}_{\mathsf{ctr}}$.

Let us denote $h_{\mathsf{pk}_i} = \{(1, h_i), (2, \widetilde{h}_i), \{(2, h_{i,j})\}_{j\in[L]\setminus\{i^*\}}\}$ for $i \in [L]$. Next, the challenger computes the handles corresponding to the challenge ciphertext by sampling $\mathsf{coin} \leftarrow \{0,1\}, \delta, s$, $v_2, \ldots, v_N \leftarrow \mathbb{Z}_p$, setting $h = g_1^{\delta}, \boldsymbol{v} = (s, v_2, \ldots, v_N)$ and computing

$$\begin{array}{lll} c_0 := s, & \mathbf{c}_1 := \{c_{1,\ell}\}_{\ell\in[n]} = \boldsymbol{x}^*_{\mathsf{coin}} + s\boldsymbol{\alpha}, & c_2 := -s\delta - \sum_{i\in[L]} s(L_1[h_i] + \beta_i\langle \boldsymbol{y}_i, \boldsymbol{\alpha}\rangle), \\ c_3 := s/\gamma, \quad c_4 := s/\pi, & \{c_{5,k} := \delta \cdot \langle \boldsymbol{v}, \boldsymbol{m}_k\rangle - s\sum_{j\in[L]:\rho(k)\notin\mathsf{Att}_j} \beta_j t_{j,\rho(k)}\}_{k\in[K]} & \end{array}$$

It then updates the lists as

- $L_1 \leftarrow L_1 \cup \{c_2, c_3, c_4, \{c_{5,k}\}_{k\in[K]}\}$,
- $L_T \leftarrow L_T \cup \{s, \{c_{1,\ell}\}_{\ell\in[n]}\}$

and gives the handles of the newly added elements to the adversary $\mathcal{A}$.

Output phase. At the end of the experiment, $\mathcal{A}$ outputs a guess $\mathsf{coin}' \in \{0,1\}$, which is the output of the experiment.

Game 1: In this game, we partially switch to the symbolic group model and replace

$$\begin{array}{llllll} \gamma, & \{\beta_i, r_i\}_{i\in[L]}, & \delta, & \boldsymbol{\alpha} = \{\alpha_\ell\}_{\ell\in[n]}, & \{c_{1,\ell}\}_{\ell\in[n]}, & c_2, \\ \pi, & \{t_{i,w}\}_{i\in[L],w\in\mathcal{U}_{\mathsf{att}}}, & s, & \{v_\mu\}_{\mu\in[2,N]}, & c_3,\ c_4, & \{c_{5,k}\}_{k\in[K]} \end{array}$$

in $\mathbb{Z}_p$ with formal variables

$$\begin{array}{llllll} \widehat{\Gamma}, & \{\widehat{\mathrm{B}}_i, \widehat{\mathrm{R}}_i\}_{i\in[L]}, & \widehat{\Delta}, & \widehat{\mathbf{A}} = \{\widehat{\mathrm{A}}_\ell\}_{\ell\in[n]}, & \{\widehat{\mathrm{C}}_{1,\ell}\}_{\ell\in[n]}, & \widehat{\mathrm{C}}_2, \\ \widehat{\Pi}, & \{\widehat{\mathrm{T}}_{i,w}\}_{i\in[L],w\in\mathcal{U}_{\mathsf{att}}}, & \widehat{\mathrm{S}}, & \{\widehat{\mathrm{V}}_\mu\}_{\mu\in[2,N]}, & \widehat{\mathrm{C}}_3,\ \widehat{\mathrm{C}}_4, & \{\widehat{\mathrm{C}}_{5,k}\}_{k\in[K]} \end{array}$$

respectively. Therefore, all the handles given to $\mathcal{A}$ refer to elements in the ring

$$\mathbb{T} := \mathbb{Z}_p\left[\begin{array}{llllll} \widehat{\Gamma}, & \widehat{\Pi}, & \{\widehat{\mathrm{B}}_i, 1/\widehat{\mathrm{B}}_i, \widehat{\mathrm{R}}_i\}_{i\in[L]}, & \widehat{\Delta}, & \widehat{\mathbf{A}} = \{\widehat{\mathrm{A}}_\ell\}_{\ell\in[n]}, & \{\widehat{\mathrm{C}}_{1,\ell}\}_{\ell\in[n]}, \\ 1/\widehat{\Gamma}, & 1/\widehat{\Pi}, & \{\widehat{\mathrm{T}}_{i,w}\}_{i\in[L],w\in\mathcal{U}_{\mathsf{att}}}, & \widehat{\mathrm{S}}, & \{\widehat{\mathrm{V}}_\mu\}_{\mu\in[2,N]}, & \widehat{\mathrm{C}}_3,\ \widehat{\mathrm{C}}_4, \end{array} \begin{array}{l} \widehat{\mathrm{C}}_2, \\ \{\widehat{\mathrm{C}}_{5,k}\}_{k\in[K]} \end{array}\right]$$

where $\{\widehat{\mathrm{R}}_i\}_{i\in[L]}$ are needed to represent the secret keys sampled by the challenger. However, when $\mathcal{A}$ submits a zero-test query, the challenger substitutes the formal variables with corresponding elements in $\mathbb{Z}_p$.

In more detail, the challenger picks $\gamma, \pi, \{\beta_i\}_{i\in[L]}, \{r_i\}_{i\in[L]}, \{t_{i,w}\}_{i\in[L],w\in\mathcal{U}_{\mathsf{att}}}, \delta, s, \{\alpha_\ell\}_{\ell\in[n]}, \{v_\mu\}_{\mu\in[2,N]}$ at the beginning of the game as in the previous game (following the same distribution). It also computes $\{c_{1,\ell}\}_{\ell\in[n]}, c_2, c_3, c_4, \{c_{5,k}\}_{k\in[K]} \in \mathbb{Z}_p$ as in the previous game when it answers the challenge query. Once the $\mathbb{Z}_p$-elements are sampled, these remain fixed throughout the experiment. Furthermore, for an honest slot i, r_i is revealed to the adversary whenever it makes a corruption query for the slot and the associated index i is added to Cor. A zero-test query before and during the challenge phase[15] always corresponds to an element $f(\widehat{\Gamma}, \widehat{\Pi}, \{\widehat{\mathrm{B}}_i\}_{i\in[L]}, \{\widehat{\mathrm{R}}_i\}_{i\in[L]}, \{\widehat{\mathrm{T}}_{i,w}\}_{i\in[L],w\in\mathcal{U}_{\mathsf{att}}}, \widehat{\Delta}, \widehat{\mathrm{S}}, \{\widehat{\mathrm{A}}_\ell\}_{\ell\in[n]}, \{\widehat{\mathrm{V}}_\mu\}_{\mu\in[2,N]})$ in the sub-ring $\mathbb{T}'$ of $\mathbb{T}$ which is defined as

$$\mathbb{T}' := \mathbb{Z}_p\left[\begin{array}{llllll} \widehat{\Gamma}, & \widehat{\Pi}, & \{\widehat{\mathrm{B}}_i, 1/\widehat{\mathrm{B}}_i, \widehat{\mathrm{R}}_i\}_{i\in[L]}, & \widehat{\Delta}, & \widehat{\mathbf{A}} = \{\widehat{\mathrm{A}}_\ell\}_{\ell\in[n]}, \\ 1/\widehat{\Gamma}, & 1/\widehat{\Pi}, & \{\widehat{\mathrm{T}}_{i,w}\}_{i\in[L],w\in\mathcal{U}_{\mathsf{att}}}, & \widehat{\mathrm{S}}, & \{\widehat{\mathrm{V}}_\mu\}_{\mu\in[2,N]} \end{array}\right].$$

For the query, the challenger returns 1 if

$$f(\gamma, \pi, \{\beta_i\}_{i\in[L]}, \{r_i\}_{i\in[L]}, \{t_{i,w}\}_{i\in[L],w\in\mathcal{U}_{\mathsf{att}}}, \delta, s, \{\alpha_\ell\}_{\ell\in[n]}, \{v_\mu\}_{\mu\in[2,N]}) = 0$$

[15] Recall that $\mathsf{IsValid}$ ia run during the challenge phase and it involves the zero-test queries.

holds over $\mathbb{Z}_p$, and 0 otherwise. A zero test query after the challenge phase corresponds to an element

$$f(\widehat{\Gamma},\widehat{\Pi},\{\widehat{\mathrm{B}}_i\}_{i\in[L]},\{\widehat{\mathrm{R}}_i\}_{i\in[L]},\{\widehat{\mathrm{T}}_{i,w}\}_{i\in[L],w\in\mathcal{U}_{\mathsf{att}}},\widehat{\Delta},\widehat{\mathrm{S}},\{\widehat{\mathrm{A}}_\ell\}_{\ell\in[n]},\{\widehat{\mathrm{V}}_\mu\}_{\mu\in[2,N]},\{\widehat{\mathrm{C}}_{1,\ell}\}_{\ell\in[n]},\widehat{\mathrm{C}}_2,\widehat{\mathrm{C}}_3,\widehat{\mathrm{C}}_4,\{\widehat{\mathrm{C}}_{5,k}\}_{k\in[K]})$$

in $\mathbb{T}$. For the query, the challenger returns 1 if

$$f(\gamma,\pi,\{\beta_i\}_{i\in[L]},\{r_i\}_{i\in[L]},\{t_{i,w}\}_{i\in[L],w\in\mathcal{U}_{\mathsf{att}}},\delta,s,\{\alpha_\ell\}_{\ell\in[n]},\{v_\mu\}_{\mu\in[2,N]},\{c_{1,\ell}\}_{\ell\in[n]},c_2,c_3,c_4,\{c_{5,k}\}_{k\in[K]})=0$$

holds over $\mathbb{Z}_p$, and 0 otherwise. We show in Lemma 1 that $\Pr[\mathsf{E}_0]=\Pr[\mathsf{E}_1]$.

We now list all the components in $\mathbb{T}$ for which the corresponding handles are given to $\mathcal{A}$ during the game. The components of $\mathbb{T}$ related to the group elements of $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are given by

$$\begin{aligned}
V_1 &= \Big\{1,1/\widehat{\Gamma},1/\widehat{\Pi},\{\widehat{\mathrm{B}}_i,\{\widehat{\mathrm{B}}_i\widehat{\mathrm{A}}_\ell\}_{\ell\in[L]},\{\widehat{\mathrm{B}}_i\widehat{\mathrm{T}}_{i,w}\}_{w\in\mathcal{U}_{\mathsf{att}}}\}_{i\in[L]},\{\widehat{\mathrm{B}}_i\widehat{\mathrm{R}}_i\}_{i\in[L]},\widehat{\mathrm{C}}_2,\widehat{\mathrm{C}}_3,\widehat{\mathrm{C}}_4,\{\widehat{\mathrm{C}}_{5,k}\}_{k\in[K]}\Big\},\\
V_2 &= \Big\{1,\{\widehat{\mathrm{B}}_i,1/\widehat{\mathrm{B}}_i,\widehat{\mathrm{B}}_i\widehat{\mathrm{R}}_i\}_{i\in[L]},\{\widehat{\Gamma}\widehat{\mathrm{B}}_i/\widehat{\mathrm{B}}_j,\widehat{\Gamma}\widehat{\mathrm{B}}_i\widehat{\mathrm{R}}_i/\widehat{\mathrm{B}}_j,\{\widehat{\Gamma}\widehat{\mathrm{B}}_i\widehat{\mathrm{A}}_\ell/\widehat{\mathrm{B}}_j\}_{\ell\in[n]},\{\widehat{\Pi}\widehat{\mathrm{B}}_i\widehat{\mathrm{T}}_{i,w}/\widehat{\mathrm{B}}_j\}_{w\in\mathcal{U}_{\mathsf{att}}}\}_{i,j\in[L],i\neq j}\Big\},\\
V_T &= \Big\{1,\widehat{\mathrm{S}},\{\widehat{\mathrm{C}}_{1,\ell}\}_{\ell\in[n]}\Big\},
\end{aligned}\tag{18}$$

respectively. We also define $\widetilde{V}_1=\{\widehat{\mathrm{C}}_2,\{\widehat{\mathrm{C}}_{5,k}\}_{k\in[K]}\}\subset V_1$ and $\widetilde{V}_2=\{1/\widehat{\mathrm{B}}_i\}_{i\in[L]}\subset V_2$.Note that any handle refers to an element $f\in\mathbb{T}$ that can be represented as

$$\begin{aligned}
f &= \textstyle\sum_{(X,Y)\in V_1\times V_2} a_{X,Y}X\cdot Y + \sum_{Z\in V_T} a_Z Z\\
&= \underbrace{\sum_{(X,Y)\in(V_1\times V_2)\setminus(\widetilde{V}_1\times\widetilde{V}_2)} a_{X,Y}X\cdot Y}_{:=f_1} + \underbrace{\sum_{i\in[L]} b_i\cdot\widehat{\mathrm{C}}_2\cdot(1/\widehat{\mathrm{B}}_i)}_{:=f_2} + \underbrace{\sum_{i\in[L],k\in[K]}\widetilde{b}_{i,k}\cdot\widehat{\mathrm{C}}_{5,k}\cdot(1/\widehat{\mathrm{B}}_i)}_{:=f_3}\\
&\quad + \underbrace{d_0\cdot\widehat{\mathrm{S}}+\sum_{\ell\in[n]} d_\ell\cdot\widehat{\mathrm{C}}_{1,\ell}}_{:=f_4}
\end{aligned}\tag{19}$$

using $a_{X,Y}\in\mathbb{Z}_p$ for $X\in V_1$ and $Y\in V_2$ and $a_Z\in\mathbb{Z}_p$ for $Z\in V_T$. In the last equation, we rename the coefficients as $b_i\coloneqq a_{\widehat{\mathrm{C}}_2,1/\widehat{\mathrm{B}}_i}$, $\widetilde{b}_{i,k}=a_{\widehat{\mathrm{C}}_{5,k},1/\widehat{\mathrm{B}}_i}$, $d_0\coloneqq a_{\widehat{\mathrm{S}}}$, and $d_\ell\coloneqq a_{\widehat{\mathrm{C}}_{1,\ell}}$ and single out the important terms for the analysis. Note that the above representation of f does not uniquely determine $\{a_{X,Y}\}_{X,Y}$, since different choices of X and Y may lead to the same product $X\cdot Y$.[16] However, we can see that $\{b_i\}_{i\in[L]}$, $\{\widetilde{b}_{i,k}\}_{i\in[L],k\in[K]}$, and $\{d_\ell\}_{\ell\in[0,n]}$ are uniquely determined from f.

Game 2: In this game, we change the way zero-test queries are answered after the challenge phase. Namely, whenever $\mathcal{A}$ submits a zero-test query corresponding to a handle of $f\in\mathbb{T}$, the challenger defines $f''\in\mathbb{T}'$ such that

$$\begin{aligned}
&f''(\widehat{\Gamma},\widehat{\Pi},\{\widehat{\mathrm{B}}_i\}_{i\in[L]},\{\widehat{\mathrm{R}}_i\}_{i\in[L]},\{\widehat{\mathrm{T}}_{i,w}\}_{i\in[L],w\in\mathcal{U}_{\mathsf{att}}},\widehat{\mathrm{S}},\{\widehat{\mathrm{A}}_\ell\}_{\ell\in[n]},\{\widehat{\mathrm{V}}_\mu\}_{\mu\in[2,N]})\\
&\coloneqq f(\widehat{\Gamma},\widehat{\Pi},\{\widehat{\mathrm{B}}_i\}_{i\in[L]},\{\widehat{\mathrm{R}}_i\}_{i\in[L]},\{\widehat{\mathrm{T}}_{i,w}\}_{i\in[L],w\in\mathcal{U}_{\mathsf{att}}},\widehat{\mathrm{S}},\{\widehat{\mathrm{A}}_\ell\}_{\ell\in[n]},\{\widehat{\mathrm{V}}_\mu\}_{\mu\in[2,N]},\{C_{1,\ell}\}_{\ell\in[n]},C_2,C_3,C_4,\{C_{5,k}\}_{k\in[K]})
\end{aligned}$$

where $\{C_{1,\ell}\}_{\ell\in[n]}$, C_2, C_3, C_4 and $\{C_{5,k}\}_{k\in[K]}$ are polynomials in $\mathbb{T}'$ defined as

$$\begin{aligned}
C_{1,\ell} &\coloneqq \widehat{\mathrm{S}}\widehat{\mathrm{A}}_\ell + x^*_{\ell,\mathsf{coin}}, \qquad C_2\coloneqq -\widehat{\mathrm{S}}\widehat{\Delta}-\sum_{i\in[L]}\widehat{\mathrm{S}}(L_1[h_i]+\widehat{\mathrm{B}}_i\cdot\langle\boldsymbol{y}_i,\widehat{\mathbf{A}}\rangle),\\
C_3 &\coloneqq \widehat{\mathrm{S}}/\widehat{\Gamma},\ C_4\coloneqq\widehat{\mathrm{S}}/\widehat{\Pi}, \qquad C_{5,k}\coloneqq\widehat{\Delta}\cdot\langle\mathbf{V},\boldsymbol{m}_k\rangle-\sum_{j\in[L]:\rho(k)\notin\mathsf{Att}_j}\widehat{\mathrm{S}}\widehat{\mathrm{B}}_j\widehat{\mathrm{T}}_{j,\rho(k)}
\end{aligned}\tag{20}$$

[16] For example, we have $\widehat{\mathrm{B}}_i\cdot(1/\widehat{\mathrm{B}}_j)=(1/\widehat{\Gamma})\cdot(\widehat{\Gamma}\widehat{\mathrm{B}}_i/\widehat{\mathrm{B}}_j)$.

respectively, where $\langle \boldsymbol{y}_i, \widehat{\mathbf{A}} \rangle$ denotes the inner product between the vector $\boldsymbol{y}_i$ and the vector of variables $\widehat{\mathbf{A}} = (\widehat{\mathrm{A}}_1, \ldots, \widehat{\mathrm{A}}_n)$, that is $\langle \boldsymbol{y}_i, \widehat{\mathbf{A}} \rangle = \sum_{\ell \in [n]} y_{i,\ell} \widehat{\mathrm{A}}_\ell$. Similarly, $\langle \mathbf{V}, \boldsymbol{m}_k \rangle$ denotes the inner product between the vector $\boldsymbol{m}_k$ and the vector of variables $\mathbf{V} = (\widehat{\mathrm{S}}, \widehat{\mathrm{V}}_2, \ldots, \widehat{\mathrm{V}}_N)$. Here, $L_1[h_i]$ is an element in $\mathbb{T}'$ defined as in **Game** 1. Then, $f' \in \mathbb{T}$ is defined as

$$\begin{aligned} &f'(\widehat{\Gamma}, \widehat{\Pi}, \{\widehat{\mathrm{B}}_i\}_{i\in[L]}, \{\widehat{\mathrm{R}}_i\}_{i\in[L]\setminus \mathsf{Cor}}, \{\widehat{\mathrm{T}}_{i,w}\}_{i\in[L],w\in\mathcal{U}_{\mathsf{att}}}, \widehat{\mathrm{S}}, \{\widehat{\mathrm{A}}_\ell\}_{\ell\in[n]}, \{\widehat{\mathrm{V}}_\mu\}_{\mu\in[2,N]}) \coloneqq \\ &\quad f''(\widehat{\Gamma}, \widehat{\Pi}, \{\widehat{\mathrm{B}}_i\}_{i\in[L]}, \{\widehat{\mathrm{R}}_i\}_{i\in[L]\setminus \mathsf{Cor}}, \{r_i\}_{i\in \mathsf{Cor}}, \{\widehat{\mathrm{T}}_{i,w}\}_{i\in[L],w\in\mathcal{U}_{\mathsf{att}}}, \widehat{\mathrm{S}}, \{\widehat{\mathrm{A}}_\ell\}_{\ell\in[n]}, \{\widehat{\mathrm{V}}_\mu\}_{\mu\in[2,N]}). \end{aligned} \tag{21}$$

We emphasize that Cor is the set of corrupted indices at the time when the zero-test query is made. After that, it answers the zero-test query by substituting the formal variables with their values in $\mathbb{Z}_p$. In particular, the challenger returns 1 if it holds that

$$f'(\gamma, \pi, \{\beta_i\}_{i\in[L]}, \{r_i\}_{i\in[L]\setminus \mathsf{Cor}}, \{t_{i,w}\}_{i\in[L],w\in\mathcal{U}_{\mathsf{att}}}, s, \{\alpha_\ell\}_{\ell\in[n]}, \{v_\mu\}_{\mu\in[2,N]}) = 0.$$

We show in Lemma 2 that $\Pr[\mathsf{E}_1] = \Pr[\mathsf{E}_2]$.

Game 3**:** In this game, we switch to the SGM completely and change the way the zero-test queries are answered by the challenger. The challenger no longer uses the integers $\gamma, \pi, \{\beta_i\}_{i\in[L]}, \{r_i\}_{i\in[L]\setminus \mathsf{Cor}}$, $\{t_{i,w}\}_{i\in[L],w\in\mathcal{U}_{\mathsf{att}}}, s, \{\alpha_\ell\}_{\ell\in[n]}, \{v_\mu\}_{\mu\in[2,N]}$ from $\mathbb{Z}_p$ in order to simulate the zero-test queries. Instead, when $\mathcal{A}$ submits a handle corresponding to $f \in \mathbb{T}$, the challenger first computes associated $f' \in \mathbb{T}'$ and returns 1 if

$$f'(\widehat{\Gamma}, \widehat{\Pi}, \{\widehat{\mathrm{B}}_i\}_{i\in[L]}, \{\widehat{\mathrm{R}}_i\}_{i\in[L]\setminus \mathsf{Cor}}, \{\widehat{\mathrm{T}}_{i,w}\}_{i\in[L],w\in\mathcal{U}_{\mathsf{att}}}, \widehat{\mathrm{S}}, \{\widehat{\mathrm{A}}_\ell\}_{\ell\in[n]}, \{\widehat{\mathrm{V}}_\mu\}_{\mu\in[2,N]}) = 0 \tag{22}$$

holds over $\mathbb{T}'$, 0 otherwise. Note that this change in particular means that Eq. (16) and (17) that are checked during the challenge phase is replaced with the following check:

$$L'_1[h^*_i] - L'_2[\widetilde{h}^*_i] \equiv 0 \tag{23}$$

and

$$L'_1[h^*_i]/\widehat{\mathrm{B}}_j - L'_2[h^*_{i,j}]/\widehat{\Gamma} \equiv 0 \quad \forall j \in [L] \setminus \{i\} \tag{24}$$

where $L'_1[h^*_i]$ and $L'_2[h^*_{i,j}]$ are polynomials in $\mathbb{T}'$ that are obtained by replacing $\{\widehat{\mathrm{R}}_i\}_{i\in \mathsf{Cor}}$ with $\{r_i\}_{i\in \mathsf{Cor}}$ that appear in $L_1[h^*_i]$ and $L_2[h^*_{i,j}]$, respectively.[17] These equations are checked over $\mathbb{T}'$ rather than over $\mathbb{Z}_p$. We show in Lemma 10 that

$$|\Pr[\mathsf{E}_2] - \Pr[\mathsf{E}_3]| \leq Q_{\mathsf{zt}}(L+7)^2/p.$$

Game 4**:** In this game, the challenger checks if the adversarially generated public keys are computed honestly before proceeding to computing the challenge ciphertext. More precisely, if $\mathcal{A}$ submits handles $\{(1, h^*_i), (2, \widetilde{h}^*_i), \{(2, h^*_{i,j})\}_{j\in[L]\setminus\{i\}}\}$ with $c_i = \perp$ at the challenge phase, the challenger verifies whether the handles satisfy

$$L'_1[h^*_i] = L'_2[\widetilde{h}^*_i] = r^*_i \widehat{\mathrm{B}}_i + \tau^*_i \widehat{\mathrm{B}}_i \widehat{\mathrm{R}}_i, \quad L'_2[h^*_{i,j}] = r^*_i \widehat{\Gamma} \widehat{\mathrm{B}}_i / \widehat{\mathrm{B}}_j + \tau^*_i \widehat{\Gamma} \widehat{\mathrm{B}}_i \widehat{\mathrm{R}}_i / \widehat{\mathrm{B}}_j, \ \forall j \neq i \tag{25}$$

for some $r^*_i, \tau^*_i \in \mathbb{Z}_p$ and aborts the experiment otherwise. We show in Lemma 11 that $\Pr[\mathsf{E}_3] = \Pr[\mathsf{E}_4]$.

Game 5**:** In this game, we further modify the way the zero-test queries are answered after the challenge phase. When $\mathcal{A}$ submits a handle corresponding to $f \in \mathbb{T}$ for a zero-test query, the challenger represents f as in Eq. (19) and returns 0 if there exists $i \notin \mathsf{Cor} \cup \mathsf{Mal}$ such that $b_i \neq 0$. Otherwise, the challenger proceeds as in the previous game for answering the zero-test queries. We show in Lemma 12 that $\Pr[\mathsf{E}_4] = \Pr[\mathsf{E}_5]$.

[17] $L_1[h^*_i]$ and $L_2[h^*_{i,j}]$ do not contain the terms $\{\widehat{\mathrm{C}}_{1,\ell}\}_{\ell\in[n]}, \widehat{\mathrm{C}}_2, \widehat{\mathrm{C}}_3$ since the challenge ciphertext is not generated at the point when these handles are specified. So, there is no need to consider the step where they are replaced with the corresponding polynomials in $\mathbb{T}'$.

Game 6**:** In this game, we further modify the way the zero-test queries are answered after the challenge phase. When $\mathcal{A}$ submits a handle corresponding to $f \in \mathbb{T}$ for a zero-test query, the challenger represents f as in Eq. (19) and returns 0 if there exists $k \in [K]$ and $\rho(k) \notin \mathsf{Att}_i$ such that $\widetilde{b}_{i,k} \neq 0$. Otherwise, the challenger proceeds as in the previous game for answering the zero-test queries. We show in Lemma 13 that $\Pr[\mathsf{E}_5] = \Pr[\mathsf{E}_6]$.

Game 7**:** In this game, we further modify the way the zero-test queries are answered after the challenge phase. When $\mathcal{A}$ submits a handle corresponding to $f \in \mathbb{T}$ for a zero-test query, the challenger represents f as in Eq. (19) and returns 0 if there exists $(i, \mu) \in [L] \times [2, N]$ such that $\sum_{k\in[K]} \widetilde{b}_{i,k} m_{k,\mu} \neq 0$ where $m_{k,\mu}$ is the (k, μ)-th entry of the LSSS matrix $\mathbf{M}$ of the challenge policy. Otherwise, the challenger proceeds as in the previous game for answering the zero-test queries. We show in Lemma 14 that $\Pr[\mathsf{E}_6] = \Pr[\mathsf{E}_7]$.

Game 8**:** In this game, we further modify the way the zero-test queries are answered after the challenge phase. Let us define a set $\mathcal{I}_{(\mathbf{M},\rho)} = \{i \in [L] : \mathsf{Att}_i \text{ satisfies the policy } (\mathbf{M}, \rho)\}$ which consists of indices in $[L]$ such that the corresponding attribute sets satisfy the challenge policy $(\mathbf{M}, \rho)$. When $\mathcal{A}$ submits a handle corresponding to $f \in \mathbb{T}$ for a zero-test query, the challenger represents f as in Eq. (19) and returns 0 if there exists $i \notin \mathcal{I}_{(\mathbf{M},\rho)}$ such that $b_i \neq 0$. Otherwise, the challenger proceeds as in the previous game for answering the zero-test queries. We show in Lemma 15 that $\Pr[\mathsf{E}_7] = \Pr[\mathsf{E}_8]$.

Game 9**:** In this game, we further modify the way the zero-test queries are answered after the challenge phase. When $\mathcal{A}$ submits a handle corresponding to $f \in \mathbb{T}$ for a zero-test query, the challenger represents f as in Eq. (19) and returns 0 if

$$\boldsymbol{d} = \sum_{i\in(\mathsf{Cor}\cup\mathsf{Mal})\cap\mathcal{I}_{(\mathbf{M},\rho)}} b_i \boldsymbol{y}_i, \quad \text{where} \quad \boldsymbol{d} = (d_1 \ldots, d_n) \tag{26}$$

does not hold. Otherwise, the challenger proceeds as in the previous game for answering the zero-test queries. We show in Lemma 16 that $\Pr[\mathsf{E}_8] = \Pr[\mathsf{E}_9]$.

In Lemma 17, we show $\Pr[\mathsf{E}_9] = 1/2$. Combining the advantage of $\mathcal{A}$ in distinguishing between the adjacent games, we obtain

$$|\Pr[\mathsf{E}_0] - \frac{1}{2}| \leq Q_{\mathsf{zt}}(L+7)^2/p$$

which is negligible in the security parameter.

We now prove Lemma 8 to 17 to complete the security analysis.

Lemma 8 *For any adversary, it holds that* $\Pr[\mathsf{E}_0] = \Pr[\mathsf{E}_1]$.

Proof. The only difference between Game 0 and Game 1 is that the challenger switched to the symbolic group model in Game 1, however, it answers the zero-test queries of $\mathcal{A}$ in the exact same way as it does in Game 0. In particular, in both the games, the zero-test queries are answered using the same distribution of $\gamma, \pi, \{\beta_i\}_{i\in[L]}, \{r_i\}_{i\in[L]}, \{t_{i,w}\}_{i\in[L],w\in\mathcal{U}_{\mathsf{att}}}, \delta, s, \{\alpha_\ell\}_{\ell\in[n]}, \{v_\mu\}_{\mu\in[2,N]}, \{c_{1,\ell}\}_{\ell\in[n]}, c_2, c_3, c_4, \{c_{5,k}\}_{k\in[K]}$ over $\mathbb{Z}_p$. Hence, the view of $\mathcal{A}$ remains the same in both the games. The lemma follows. □

Lemma 9 *For any adversary, it holds that* $\Pr[\mathsf{E}_1] = \Pr[\mathsf{E}_2]$.

Proof. The only difference between Game 1 and Game 2 is that when answering to the zero-test queries corresponding to a polynomial $f \in \mathbb{T}$ after the challenge phase, the challenger replaces the formal variables $\{\widehat{\mathrm{C}}_{1,\ell}\}_{\ell\in[n]}, \widehat{\mathrm{C}}_2, \widehat{\mathrm{C}}_3, \widehat{\mathrm{C}}_4, \{\widehat{\mathrm{C}}_{5,k}\}_{k\in[K]}$ with the polynomials $\{C_{1,\ell}\}_{\ell\in[n]}, C_2, C_3, C_4, \{C_{5,k}\}_{k\in[K]} \in \mathbb{T}'$ defined as Equation (20) and $\{\widehat{\mathrm{R}}_i\}_{i\in\mathsf{Cor}}$ with $\{r_i\}_{i\in\mathsf{Cor}}$, represents f as an element f' in the subring $\mathbb{T}'$, and replaces the formal variables with the $\mathbb{Z}_p$ values to see if the resulting value equals to 0 in Game 2, whereas the challenger directly replaces the formal variables in f with the corresponding $\mathbb{Z}_p$ values without going through the

intermediate polynomial f'. However, this does not change the value that the zero-test oracle returns to the adversary since we have

$$\begin{aligned} & f'(\mathsf{values}') = f''(\mathsf{values}', \mathsf{values}'') = f''(\mathsf{values}) \\ = \;& f(\mathsf{values}, \{C_{1,\ell}(\mathsf{values})\}_{\ell\in[n]}, C_2(\mathsf{values}), C_3(\mathsf{values}), C_4(\mathsf{values}), \{C_{5,k}(\mathsf{values})\}_{k\in[K]}) \\ = \;& f(\mathsf{values}, \{c_{1,\ell}\}_{\ell\in[n]}, c_2, c_3, c_4, \{c_{5,k}\}_{k\in[K]}) \end{aligned}$$

where values, values', and values'' are short-hand expressions for $(\gamma, \pi, \{\beta_i\}_{i\in[L]}, \{r_i\}_{i\in[L]}, \{t_{i,w}\}_{i\in[L],w\in\mathcal{U}_{\mathsf{att}}}, \delta, s, \{\alpha_\ell\}_{\ell\in[n]}, \{v_\mu\}_{\mu\in[2,N]})$, $(\gamma, \pi, \{\beta_i\}_{i\in[L]}, \{r_i\}_{i\in[L]\setminus\mathsf{Cor}}, \{t_{i,w}\}_{i\in[L],w\in\mathcal{U}_{\mathsf{att}}}, \delta, s, \{\alpha_\ell\}_{\ell\in[n]}, \{v_\mu\}_{\mu\in[2,N]})$, and $\{r_i\}_{i\in\mathsf{Cor}}$, respectively. Hence, the view of $\mathcal{A}$ remains the same in both the games. The lemma follows. □

Lemma 10 *For any adversary, it holds that* $|\Pr[\mathsf{E}_2] - \Pr[\mathsf{E}_3]| \leq Q_{\mathsf{zt}}(L+7)^2/p$.

Proof. The only change of Game 3 from Game 2 is that the challenger uses the formal variables $\widehat{\Gamma}, \widehat{\Pi}$, $\{\widehat{\mathrm{B}}_i\}_{i\in[L]}, \{\widehat{\mathrm{R}}_i\}_{i\in[L]\setminus\mathsf{Cor}}, \{\widehat{\mathrm{T}}_{i,w}\}_{i\in[L],w\in\mathcal{U}_{\mathsf{att}}}, \widehat{\Delta}, \widehat{\mathrm{S}}, \{\widehat{\mathrm{A}}_\ell\}_{\ell\in[n]}, \{\widehat{\mathrm{V}}_\mu\}_{\mu\in[2,N]}$ instead of the $\mathbb{Z}_p$ values $\gamma, \pi, \{\beta_i\}_{i\in[L]}$, $\{r_i\}_{i\in[L]\setminus\mathsf{Cor}}, \{t_{i,w}\}_{i\in[L],w\in\mathcal{U}_{\mathsf{att}}}, \delta, s, \{\alpha_\ell\}_{\ell\in[n]}, \{v_\mu\}_{\mu\in[2,N]}$ to answer the zero-test queries of $\mathcal{A}$, where Cor is the set of corrputed indices at the time when the query is made. Therefore, the only possible case when Game 3 behaves differently from Game 2 in $\mathcal{A}$'s view is the following: $\mathcal{A}$ submits a handle for an element $f \in \mathbb{T}$ to the zero-test oracle satisfying

$$\begin{aligned} & f'(\gamma, \pi, \{\beta_i\}_{i\in[L]}, \{r_i\}_{i\in[L]}, \{t_{i,w}\}_{i\in[L],w\in\mathcal{U}_{\mathsf{att}}}, \delta, s, \{\alpha_\ell\}_{\ell\in[n]}, \{v_\mu\}_{\mu\in[2,N]}) = 0 \quad \bigwedge \\ & f'(\widehat{\Gamma}, \widehat{\Pi}, \{\widehat{\mathrm{B}}_i\}_{i\in[L]}, \{\widehat{\mathrm{R}}_i\}_{i\in[L]\setminus\mathsf{Cor}}, \{\widehat{\mathrm{T}}_{i,w}\}_{i\in[L],w\in\mathcal{U}_{\mathsf{att}}}, \widehat{\Delta}, \widehat{\mathrm{S}}, \{\widehat{\mathrm{A}}_\ell\}_{\ell\in[n]}, \{\widehat{\mathrm{V}}_\mu\}_{\mu\in[2,N]}) \neq 0 \end{aligned}$$

where $f' \in \mathbb{T}'$ is defined from associated $f \in \mathbb{T}$ as in Eq. (21). Let us denote the event as $\mathsf{E}_{\mathrm{bad}}$. It is sufficient to bound the probability of $\mathsf{E}_{\mathrm{bad}}$ in Game 2.

We now fix an element $f' \in \mathbb{T}'$ and define a *polynomial*

$$\begin{aligned} & g(\widehat{\Gamma}, \widehat{\Pi}, \{\widehat{\mathrm{B}}_i\}_{i\in[L]}, \{\widehat{\mathrm{R}}_i\}_{i\in[L]\setminus\mathsf{Cor}}, \{\widehat{\mathrm{T}}_{i,w}\}_{i\in[L],w\in\mathcal{U}_{\mathsf{att}}}, \widehat{\Delta}, \widehat{\mathrm{S}}, \{\widehat{\mathrm{A}}_\ell\}_{\ell\in[n]}, \{\widehat{\mathrm{V}}_\mu\}_{\mu\in[2,N]}) \\ & \in \mathbb{Z}_p[\widehat{\Gamma}, \widehat{\Pi}, \{\widehat{\mathrm{B}}_i\}_{i\in[L]}, \{\widehat{\mathrm{R}}_i\}_{i\in[L]\setminus\mathsf{Cor}}, \{\widehat{\mathrm{T}}_{i,w}\}_{i\in[L],w\in\mathcal{U}_{\mathsf{att}}}, \widehat{\Delta}, \widehat{\mathrm{S}}, \{\widehat{\mathrm{A}}_\ell\}_{\ell\in[n]}, \{\widehat{\mathrm{V}}_\mu\}_{\mu\in[2,N]}] \end{aligned}$$

as

$$\begin{aligned} & g(\widehat{\Gamma}, \widehat{\Pi}, \{\widehat{\mathrm{B}}_i\}_{i\in[L]}, \{\widehat{\mathrm{R}}_i\}_{i\in[L]\setminus\mathsf{Cor}}, \{\widehat{\mathrm{T}}_{i,w}\}_{i\in[L],w\in\mathcal{U}_{\mathsf{att}}}, \widehat{\Delta}, \widehat{\mathrm{S}}, \{\widehat{\mathrm{A}}_\ell\}_{\ell\in[n]}, \{\widehat{\mathrm{V}}_\mu\}_{\mu\in[2,N]}) \\ & = \widehat{\Gamma} \cdot \textstyle\prod_{i\in[L]} \widehat{\mathrm{B}}_i \cdot f'(\widehat{\Gamma}, \widehat{\Pi}, \{\widehat{\mathrm{B}}_i\}_{i\in[L]}, \{\widehat{\mathrm{R}}_i\}_{i\in[L]\setminus\mathsf{Cor}}, \{\widehat{\mathrm{T}}_{i,w}\}_{i\in[L],w\in\mathcal{U}_{\mathsf{att}}}, \widehat{\Delta}, \widehat{\mathrm{S}}, \{\widehat{\mathrm{A}}_\ell\}_{\ell\in[n]}, \{\widehat{\mathrm{V}}_\mu\}_{\mu\in[2,N]}). \end{aligned}$$

The product $\widehat{\Gamma} \cdot \prod_{i\in[L]} \widehat{\mathrm{B}}_i$ is introduced to cancel out any denominator of $f' \in \mathbb{T}'$. Note that g is a polynomial in the ring $\mathbb{Z}_p[\widehat{\Gamma}, \widehat{\Pi}, \{\widehat{\mathrm{B}}_i\}_{i\in[L]}, \{\widehat{\mathrm{R}}_i\}_{i\in[L]\setminus\mathsf{Cor}}, \{\widehat{\mathrm{T}}_{i,w}\}_{i\in[L],w\in\mathcal{U}_{\mathsf{att}}}, \widehat{\Delta}, \widehat{\mathrm{S}}, \{\widehat{\mathrm{A}}_\ell\}_{\ell\in[n]}, \{\widehat{\mathrm{V}}_\mu\}_{\mu\in[2,N]}]$ rather than in $\mathbb{T}$. Therefore, the event $\mathsf{E}_{\mathrm{bad}}$ occurs if and only if

$$\begin{aligned} & g(\gamma, \pi, \{\beta_i\}_{i\in[L]}, \{r_i\}_{i\in[L]\setminus\mathsf{Cor}}, \{t_{i,w}\}_{i\in[L],w\in\mathcal{U}_{\mathsf{att}}}, \delta, s, \{\alpha_\ell\}_{\ell\in[n]}, \{v_\mu\}_{\mu\in[2,N]}) = 0 \quad \bigwedge \\ & g(\widehat{\Gamma}, \widehat{\Pi}, \{\widehat{\mathrm{B}}_i\}_{i\in[L]}, \{\widehat{\mathrm{R}}_i\}_{i\in[L]\setminus\mathsf{Cor}}, \{\widehat{\mathrm{T}}_{i,w}\}_{i\in[L],w\in\mathcal{U}_{\mathsf{att}}}, \widehat{\Delta}, \widehat{\mathrm{S}}, \{\widehat{\mathrm{A}}_\ell\}_{\ell\in[n]}, \{\widehat{\mathrm{V}}_\mu\}_{\mu\in[2,N]}) \neq 0 \end{aligned}$$

since γ and $\{\beta_i\}_{i\in[L]}$ are non-zero over $\mathbb{Z}_p$. Using Schwartz-Zippel lemma (Lemma 1), we can bound the probability of $\mathsf{E}_{\mathrm{bad}}$ by $(L+7)^2/p$ since the maximum degree of g is $(L+7)$ over the ring $\mathbb{Z}_p[\widehat{\Gamma}, \widehat{\Pi}, \{\widehat{\mathrm{B}}_i\}_{i\in[L]}$, $\{\widehat{\mathrm{R}}_i\}_{i\in[L]\setminus\mathsf{Cor}}, \{\widehat{\mathrm{T}}_{i,w}\}_{i\in[L],w\in\mathcal{U}_{\mathsf{att}}}, \widehat{\Delta}, \widehat{\mathrm{S}}, \{\widehat{\mathrm{A}}_\ell\}_{\ell\in[n]}, \{\widehat{\mathrm{V}}_\mu\}_{\mu\in[2,N]}]$. The maximum degree is calculated by representing f as in Eq. (19) and noting that the degree of C_2 or $C_{5,k}$ as a polynomial in $\mathbb{T}'$ does not exceed 4, since each $L_1[h_i]$ should be the linear combination of element in $V_1 \setminus \{\widehat{\mathrm{C}}_2, \widehat{\mathrm{C}}_3, \widehat{\mathrm{C}}_4, \{\widehat{\mathrm{C}}_{5,k}\}_{k\in[K]}\}$ (See Eq. (20).). Since there are a total of Q_{zt} number of zero-test queries, the lemma follows. □

Lemma 11 *For any adversary, it holds that* $\Pr[\mathsf{E}_3] = \Pr[\mathsf{E}_4]$.

Proof. The only difference between Game 3 and Game 4 is that the challenger aborts the experiment in Game 4 at the challenge phase if the adversarially sampled public keys do not match with a particular format. In particular, if $\mathcal{A}$ submits handles $\{(1, h_i^*), (2, \widetilde{h}_i^*), \{(2, h_{i,j}^*)\}_{j\in[L]\setminus\{i\}}\}$ representing the handles for (an adversarially generated) public key for the slot i at the challenge phase, then the challenger verifies whether the handles satisfy Eq. (25) for some $r_i^*, \tau_i^* \in \mathbb{Z}_p$. Note that in both the games, the challenger runs $\mathsf{IsValid}$ before proceeding to computing the challenge ciphertext, otherwise the challenger aborts the experiment. Thus, the two games only differ from $\mathcal{A}$'s view if $\mathsf{IsValid}$ outputs 1 on those handles, but the handles do not map to the particular monomials as shown above. Here, we show that this cannot happen.

For proving this, we first define the set of monomials V_0 as $V_1\setminus\{\widehat{\mathrm{C}}_2, \widehat{\mathrm{C}}_3, \widehat{\mathrm{C}}_4, \{\widehat{\mathrm{C}}_{5,k}\}_{k\in[K]}\}$. We can see that $L_1[h_i^*]$ should be an element in $\mathbb{T}'$ that can be represented as a linear combination of monomials in V_0 with coefficients in $\mathbb{Z}_p$.[18] This implies that $L_1'[h_i^*]$ can be represented as a linear combination of monomials in V_0' with coefficients in $\mathbb{Z}_p$, where V_0' is defined as

$$V_0' \coloneqq \left\{1, 1/\widehat{\Gamma}, 1/\widehat{\Pi}, \{\widehat{\mathrm{B}}_\kappa, \{\widehat{\mathrm{B}}_\kappa\widehat{\mathrm{A}}_\ell\}_{\ell\in[L]}\}_{\kappa\in[L]}, \{\widehat{\mathrm{B}}_\kappa\widehat{\mathrm{T}}_{\kappa,w}\}_{\kappa\in[L],w\in\mathcal{U}_{\mathsf{att}}}, \{\widehat{\mathrm{B}}_\kappa\widehat{\mathrm{R}}_\kappa\}_{\kappa\in[L]\setminus\mathsf{Cor}}\right\}$$

Similarly, we can also see that $L_2[\widetilde{h}_i^*]$ and $L_2[h_{i,j}^*]$ for $j \in [L]\setminus\{i\}$ should correspond to elements in $\mathbb{T}'$ that can be represented as linear combinations of monomials in V_2 with coefficients in $\mathbb{Z}_p$. This implies that $L_2'[h_i^*]$ and $L_2'[h_{i,j}^*]$ for all $j \in [L]\setminus\{i\}$ can be represented as linear combinations of monomials in V_2' with coefficients in $\mathbb{Z}_p$, where V_2' is defined as

$$V_2' = \left\{1, \{\widehat{\mathrm{B}}_\kappa, 1/\widehat{\mathrm{B}}_\kappa\}_{\kappa\in[L]}, \{\widehat{\Gamma}\widehat{\mathrm{B}}_\kappa/\widehat{\mathrm{B}}_j, \{\widehat{\Gamma}\widehat{\mathrm{B}}_\kappa\widehat{\mathrm{A}}_\ell/\widehat{\mathrm{B}}_j\}_{\ell\in[n]}, \{\widehat{\Pi}\widehat{\mathrm{B}}_\kappa\widehat{\mathrm{T}}_{\kappa,w}/\widehat{\mathrm{B}}_j\}_{w\in\mathcal{U}_{\mathsf{att}}}\}_{\kappa,j\in[L],\kappa\neq j}, \{\widehat{\mathrm{B}}_\kappa\widehat{\mathrm{R}}_\kappa, \{\widehat{\Gamma}\widehat{\mathrm{B}}_\kappa\widehat{\mathrm{R}}_\kappa/\widehat{\mathrm{B}}_j\}_{j\neq\kappa}\}_{\kappa\in[L]\setminus\mathsf{Cor}}\right\}.$$

If the handles pass the verification, it in particular means that the handles satisfy Eq. (23) . This means that $L_1'[h_i^*]$ (and thus $L_2'[\widetilde{h}_i^*]$) can be represented as a linear combination of monomials in $V_0' \cap V_2' = \{\widehat{\mathrm{B}}_\kappa\}_{\kappa\in[L]} \cup \{\widehat{\mathrm{B}}_\kappa\widehat{\mathrm{R}}_\kappa\}_{\kappa\in[L]\setminus\mathsf{Cor}}$ with coefficients in $\mathbb{Z}_p$. Namely, we can write

$$L_1'[h_i^*] = e_0 + \sum_{\kappa\in[L]} e_\kappa\widehat{\mathrm{B}}_\kappa + \sum_{\kappa\in[L]\setminus\mathsf{Cor}} e'_\kappa\widehat{\mathrm{B}}_\kappa\widehat{\mathrm{R}}_\kappa$$

using $e_\kappa \in \mathbb{Z}_p$ for $\kappa \in [0, L]$ and $e'_\kappa \in \mathbb{Z}_p$ for $\kappa \in [L]\setminus\mathsf{Cor}$. We then show that $e_\kappa = 0$ and $e'_\kappa = 0$ hold for all $\kappa \neq i$ if the handles pass the verification shown in Eq. (24).

- We first show that $e_0 = 0$. For the sake of contradiction, let us assume that $e_0 \neq 0$ and take some index $\kappa^* \neq [L]\setminus\{i\}$. Then, $L_2'[h_{i,\kappa^*}^*]$ should contain the term $\widehat{\Gamma}/\widehat{\mathrm{B}}_{\kappa^*}$, since Eq. (24) with $\kappa^* \neq i$ implies $L_2'[h_{i,\kappa^*}^*] = \widehat{\Gamma}\cdot L_1'[h_i^*]/\widehat{\mathrm{B}}_{\kappa^*}$. However, this contradicts the fact that $L_2'[h_{i,\kappa^*}^*]$ can be represented as a linear combination of the monomials in V_2' with coefficients in $\mathbb{Z}_p$, since $\widehat{\Gamma}/\widehat{\mathrm{B}}_{\kappa^*} \notin V_2'$.
- We first show that $e_\kappa = 0$ for all $\kappa \neq i$. For the sake of contradiction, let us assume that $e_{\kappa^*} \neq 0$ holds for some $\kappa^* \neq i$. Then, $L_2'[h_{i,\kappa^*}^*]$ should contain the term $\widehat{\Gamma}$, since Eq. (24) with $j = \kappa^*$ implies $L_2'[h_{i,\kappa^*}^*] = \widehat{\Gamma}\cdot L_1'[h_i^*]/\widehat{\mathrm{B}}_{\kappa^*}$. However, this contradicts the fact that $L_2'[h_{i,\kappa^*}^*]$ can be represented as a linear combination of the monomials in V_2' with coefficients in $\mathbb{Z}_p$, since $\widehat{\Gamma} \notin V_2'$.
- We next show that $e'_\kappa = 0$ for all $\kappa \neq i$. For the sake of contradiction, let us assume that $e'_{\kappa^*} \neq 0$ holds for some $\kappa^* \neq i$. Then, $L_2'[h_{i,\kappa^*}^*]$ should contain the term $\widehat{\Gamma}\widehat{\mathrm{R}}_{\kappa^*}$, since Eq. (24) with $j = \kappa^*$ implies $L_2'[h_{i,\kappa^*}^*] = \widehat{\Gamma}\cdot L_1'[h_i^*]/\widehat{\mathrm{B}}_{\kappa^*}$. However, this contradicts the fact that $L_2'[h_{i,\kappa^*}^*]$ can be represented as a linear combination of the monomials in V_2' with coefficients in $\mathbb{Z}_p$, since $\widehat{\Gamma}\widehat{\mathrm{R}}_{\kappa^*} \notin V_2'$.

The above shows that $L_1'[h_i^*]$ can be written as $L_1'[h_i^*] = r_i^*\widehat{\mathrm{B}}_i + \tau_i^*\widehat{\mathrm{B}}_i\widehat{\mathrm{R}}_i$ using $r_i^* \in \mathbb{Z}_p$ and $\tau_i^* \in \mathbb{Z}_p$, if we rename e_i and e'_i as r_i^* and τ_i^*, respectively. Then, $L_2'[\widetilde{h}_i^*] = r_i^*\widehat{\mathrm{B}}_i + \tau_i^*\widehat{\mathrm{B}}_i\widehat{\mathrm{R}}_i$ follows from Eq. (16) and $L_2'[h_{i,j}^*] = r_i^*\widehat{\Gamma}\widehat{\mathrm{B}}_i/\widehat{\mathrm{B}}_j + \tau_i^*\widehat{\Gamma}\widehat{\mathrm{B}}_i\widehat{\mathrm{R}}_i/\widehat{\mathrm{B}}_j$ for all $j \neq i$ follows from Eq.(24).

[18] Note that the linear combination should not include $\widehat{\mathrm{C}}_2$ and $\widehat{\mathrm{C}}_3$, since the handle is submitted before the challenge phase.

Lemma 12 *For any adversary, it holds that* $\Pr[\mathsf{E}_4] = \Pr[\mathsf{E}_5]$.

Proof. The only difference between Game 4 and Game 5 is in the way zero-test queries are answered. The only possible case when Game 5 behaves differently from Game 4 in $\mathcal{A}$'s view is the case where the following event occurs: $\mathcal{A}$ submits a handle for a polynomial $f \in \mathbb{T}$ to the zero-test oracle such that $b_{i^*} \neq 0$ for some $i^* \in [L] \backslash (\mathsf{Mal} \cup \mathsf{Cor})$ and the associated polynomial $f' \in \mathbb{T}'$ equals to 0 in $\mathbb{T}'$. Here, we prove that f' cannot be 0 for such f.

Recall that f' is obtained by replacing $\{\widehat{\mathrm{C}}_{1,\ell}\}_{\ell\in[n]}$, $\widehat{\mathrm{C}}_2$, $\widehat{\mathrm{C}}_3$, $\widehat{\mathrm{C}}_4$ and $\{\widehat{\mathrm{C}}_{5,k}\}_{k\in[N]}$ in Eq. (19) with polynomials $\{C_{1,\ell}\}_{\ell\in[n]}$, C_2, C_3, C_4 and $\{C_{5,k}\}_{k\in[K]}$ defined as in Eq. (20) to obtain $f'' \in \mathbb{T}'$ and then replacing $\{\widehat{\mathrm{R}}_i\}_{i\in\mathsf{Cor}}$ that appear in f'' with $\{r_i\}_{i\in\mathsf{Cor}}$. We divide f into f_1, f_2, f_3 and f_4 as in Eq. (19) and define the corresponding polynomials f_1'', f_2'', f_3'', f_4'', and f_1', f_2', f_3' and f_4' similarly. First observe that we can write f_2'' as follows:

$$f_2'' = \sum_{i\in[L]} b_i \cdot C_2 \cdot (1/\widehat{\mathrm{B}}_i) = -\sum_{i\in[L]} b_i \cdot \widehat{\mathrm{S}}\widehat{\Delta}/\widehat{\mathrm{B}}_i - \sum_{i\in[L],j\in[L]} b_i \cdot \widehat{\mathrm{S}}(L_1[h_j] + \widehat{\mathrm{B}}_j \cdot \langle \boldsymbol{y}_j, \widehat{\mathbf{A}}\rangle)(1/\widehat{\mathrm{B}}_i).$$

If we replace $\{\widehat{\mathrm{R}}_i\}_{i\in\mathsf{Cor}}$ that (implicitly) appear in the above sum with $\{r_i\}_{i\in\mathsf{Cor}}$, we obtain the following:

$$\begin{aligned}
f_2' &= -\sum_{i\in[L]} b_i \cdot \widehat{\mathrm{S}}\widehat{\Delta}/\widehat{\mathrm{B}}_i - \sum_{i\in[L],j\in[L]} b_i \cdot \widehat{\mathrm{S}}(L_1'[h_j] + \widehat{\mathrm{B}}_j \cdot \langle \boldsymbol{y}_j, \widehat{\mathbf{A}}\rangle)(1/\widehat{\mathrm{B}}_i) \\
&= -\sum_{i\in[L]} b_i \cdot \widehat{\mathrm{S}}\widehat{\Delta}/\widehat{\mathrm{B}}_i - \sum_{i\in[L],j\in[L]\backslash(\mathsf{Mal}\cup\mathsf{Cor})} b_i \cdot \widehat{\mathrm{S}}\widehat{\mathrm{B}}_j\widehat{\mathrm{R}}_j/\widehat{\mathrm{B}}_i - \sum_{i\in[L],j\in\mathsf{Cor}\backslash\mathsf{Mal}} b_i r_j \cdot \widehat{\mathrm{S}}\widehat{\mathrm{B}}_j/\widehat{\mathrm{B}}_i \\
&\qquad - \sum_{i\in[L],j\in\mathsf{Mal}} b_i \cdot \widehat{\mathrm{S}}\widehat{\mathrm{B}}_j(r_j^* + \tau_j^*\widehat{\mathrm{R}}_j)/\widehat{\mathrm{B}}_i - \sum_{i,j\in[L]} b_i \cdot \langle \boldsymbol{y}_j, \widehat{\mathbf{A}}\rangle \cdot \widehat{\mathrm{S}}\widehat{\mathrm{B}}_j/\widehat{\mathrm{B}}_i \\
&= -\sum_{i\in[L]} b_i \cdot \widehat{\mathrm{S}}\widehat{\Delta}/\widehat{\mathrm{B}}_i - \sum_{i\in[L]\backslash(\mathsf{Mal}\cup\mathsf{Cor})} b_i \cdot \widehat{\mathrm{S}}\widehat{\mathrm{R}}_i - \sum_{i\in\mathsf{Cor}\backslash\mathsf{Mal}} b_i r_i \cdot \widehat{\mathrm{S}} - \sum_{i\in\mathsf{Mal}} b_i \cdot \widehat{\mathrm{S}}(r_i^* + \tau_i^*\widehat{\mathrm{R}}_i) \\
&\qquad - \sum_{i\in[L]} b_i \cdot \langle \boldsymbol{y}_i, \widehat{\mathbf{A}}\rangle \cdot \widehat{\mathrm{S}} - \Phi
\end{aligned} \tag{27}$$

where we define Φ as

$$\begin{aligned}
\Phi &= \sum_{i\in[L],j\in[L]\backslash(\mathsf{Mal}\cup\mathsf{Cor}),i\neq j} b_i \cdot \widehat{\mathrm{S}}\widehat{\mathrm{B}}_j\widehat{\mathrm{R}}_j/\widehat{\mathrm{B}}_i + \sum_{i\in[L],j\in\mathsf{Cor}\backslash\mathsf{Mal},i\neq j} b_i r_j \cdot \widehat{\mathrm{S}}\widehat{\mathrm{B}}_j/\widehat{\mathrm{B}}_i \\
&\qquad + \sum_{i\in[L],j\in\mathsf{Mal},i\neq j} b_i \cdot \widehat{\mathrm{S}}\widehat{\mathrm{B}}_j(r_j^* + \tau_j^*\widehat{\mathrm{R}}_j)/\widehat{\mathrm{B}}_i + \sum_{i,j\in[L],i\neq j} b_i \cdot \langle \boldsymbol{y}_j, \widehat{\mathbf{A}}\rangle \cdot \widehat{\mathrm{S}}\widehat{\mathrm{B}}_j/\widehat{\mathrm{B}}_i.
\end{aligned} \tag{28}$$

In the second line of Eq. (27), we use the fact that $L_1'[h_j]$ can be represented as $L_1'[h_j] = r_j^*\widehat{\mathrm{B}}_j + \tau_j^*\widehat{\mathrm{B}}_j\widehat{\mathrm{R}}_j$ for $j \in \mathsf{Mal}$ due to the change introduced in **Game 4** and in the last line, we single out the terms with $i = j$ from each of the summations in the second line and put other terms into Φ. In particular, f_2' contains the term $\widehat{\mathrm{S}}\widehat{\mathrm{R}}_{i^*}$ with non-zero coefficient b_{i^*}, which appears in the second summation (from left) in the last line of Eq. (27). It can be seen by inspection that the term cannot be cancelled inside the above sum. We then show that not a single polynomial among f_1', f_3' and f_4' contains the monomial $\widehat{\mathrm{S}}\widehat{\mathrm{R}}_{i^*}$.

- It can be observed that f_3' can be represented as a linear combination of $\widehat{\Delta}\widehat{\mathrm{S}}/\widehat{\mathrm{B}}_j$, $\{\widehat{\Delta}\widehat{\mathrm{V}}_\mu/\widehat{\mathrm{B}}_i\}_{\mu\in[2,N],j\in[L]}$ and $\{\widehat{\mathrm{S}}\widehat{\mathrm{B}}_j\widehat{\mathrm{T}}_{j,\rho(k)}/\widehat{\mathrm{B}}_i\}_{i\in[L],j\in[L]:\rho(k)\notin\mathsf{Att}_j}$. Therefore, f_3' does not contain the monomial $\widehat{\mathrm{S}}\widehat{\mathrm{R}}_{i^*}$.
- It is easy to observe that f_4' can be represented as a linear combination of $\widehat{\mathrm{S}}$, 1, and $\{\widehat{\mathrm{S}}\widehat{\mathrm{A}}_\ell\}_\ell$. Therefore, it does not contain the term $\widehat{\mathrm{S}}\widehat{\mathrm{R}}_{i^*}$.
- We then show that f_1' cannot contain the term either. To show this, we consider each combination of $(X, Y) \in (V_1 \times V_2)\backslash(\widetilde{V}_1 \times \widetilde{V}_2)$ where $\widetilde{V}_1 = \{\widehat{\mathrm{C}}_2, \{\widehat{\mathrm{C}}_{5,k}\}_{k\in\in[K]}\}$, $\widetilde{V}_2 = \{1/\widehat{\mathrm{B}}_i\}_{i\in[L]}$ and show that the

resulting element that is obtained by replacing $\widehat{\mathrm{C}}_2, \widehat{\mathrm{C}}_3, \widehat{\mathrm{C}}_4, \{\widehat{\mathrm{C}}_{5,k}\}_{k\in[K]}$ that appear in $X \cdot Y$ with the polynomials C_2, C_3, C_4 and $C_{5,k}$, and then replacing $\{\widehat{\mathrm{R}}_i\}_{i\in\mathsf{Cor}}$ with $\{r_i\}_{i\in\mathsf{Cor}}$ does not contain the desired term $\widehat{\mathrm{S}}\widehat{\mathrm{R}}_{i^*}$. We first observe that X should be one of $\widehat{\mathrm{C}}_2$, $\widehat{\mathrm{C}}_3$, $\widehat{\mathrm{C}}_4$ and $\{\widehat{\mathrm{C}}_{5,k}\}_{k\in[K]}$ since otherwise the resulting element does not contain the multiplicative factor $\widehat{\mathrm{S}}$.

- We first consider the case of $X = \widehat{\mathrm{C}}_2$. In this case, Y should be chosen from $V_2 \backslash \widetilde{V}_2$. Note that, in this case, the term XY is not a monomial, but a sum of monomials. However, the resulting terms are multiplied by $\widehat{\mathrm{B}}_i$ for some i and thus does not yield the term $\widehat{\mathrm{S}}\widehat{\mathrm{R}}_{i^*}$.
- We then consider the case of $X = \widehat{\mathrm{C}}_3$. In this case, Y is chosen from V_2. Then, the resulting element is either multiplied by $1/\widehat{\Gamma}$ or $\widehat{\mathrm{B}}_i/\widehat{\mathrm{B}}_j$ with $i \neq j$. Therefore, it does not yield the term either.
- We then consider the case of $X = \widehat{\mathrm{C}}_4$. In this case, Y is chosen from V_2. Then, the resulting element is either multiplied by $1/\widehat{\Pi}$ or $\widehat{\mathrm{B}}_i/\widehat{\mathrm{B}}_j$ with $i \neq j$. Therefore, it does not yield the term either.
- We next consider the case of $X = \widehat{\mathrm{C}}_{5,k}$. In this case, Y should be chosen from $V_2 \backslash \widetilde{V}_2$. Note that, in this case, the term XY is not a monomial, but a sum of monomials. However, the resulting terms are either multiplied by $\widehat{\Delta}$ or $\widehat{\mathrm{T}}_{j,\rho(k)}$. Therefore, it does not yield the term $\widehat{\mathrm{S}}\widehat{\mathrm{R}}_{i^*}$.

Therefore, the monomial $\widehat{\mathrm{S}}\widehat{\mathrm{R}}_{i^*}$ that appears in f'_1 cannot be cancelled by any term in f'_2, f'_3 and f'_4. This implies $f' = f'_1 + f'_2 + f'_3 + f'_4 \neq 0$ over $\mathbb{T}'$ as desired. This completes the proof. □

Lemma 13 *For any adversary, it holds that* $\Pr[\mathsf{E}_5] = \Pr[\mathsf{E}_6]$.

Proof. The only difference between Game 5 and Game 6 is in the way zero-test queries are answered. The only possible case when Game 6 behaves differently from Game 5 in $\mathcal{A}$'s view is the case where the following event occurs: $\mathcal{A}$ submits a handle for a polynomial $f \in \mathbb{T}$ to the zero-test oracle such that $\widetilde{b}_{i,k} \neq 0$ for some $(i,k) \in [L] \times [K]$ such that $\rho(k) \notin \mathsf{Att}_i$ and the associated polynomial $f' \in \mathbb{T}'$ equals to 0 in $\mathbb{T}'$. Here, we prove that f' cannot be 0 for such f.

Recall that f' is obtained by replacing $\{\widehat{\mathrm{C}}_{1,\ell}\}_{\ell\in[n]}$, $\widehat{\mathrm{C}}_2$, $\widehat{\mathrm{C}}_3$, $\widehat{\mathrm{C}}_4$ and $\{\widehat{\mathrm{C}}_{5,k}\}_{k\in[N]}$ in Eq. (19) with polynomials $\{C_{1,\ell}\}_{\ell\in[n]}$, C_2, C_3, C_4 and $\{C_{5,k}\}_{k\in[K]}$ defined as in Eq. (20) to obtain $f'' \in \mathbb{T}'$ and then replacing $\{\widehat{\mathrm{R}}_i\}_{i\in\mathsf{Cor}}$ that appear in f'' with $\{r_i\}_{i\in\mathsf{Cor}}$. We divide f into f_1, f_2, f_3 and f_4 as in Eq. (19) and define the corresponding polynomials f''_1, f''_2, f''_3, f''_4, and f'_1, f'_2, f'_3 and f'_4 similarly. First observe that we can write $f''_3 = f'_3$ since it does not contain $\widehat{\mathrm{R}}_i$ and it can be written as follows:

$$
\begin{aligned}
f'_3 &= \sum_{i\in[L],k\in[K]} \widetilde{b}_{i,k} \cdot C_{5,k} \cdot (1/\widehat{\mathrm{B}}_i) = \sum_{i\in[L],k\in[K]} \widetilde{b}_{i,k} \cdot \left(\widehat{\Delta} \cdot \langle \mathbf{V}, \boldsymbol{m}_k \rangle - \sum_{j\in[L]:\rho(k)\notin\mathsf{Att}_j} \widehat{\mathrm{S}}\widehat{\mathrm{B}}_j\widehat{\mathrm{T}}_{j,\rho(k)} \right) \cdot (1/\widehat{\mathrm{B}}_i) \\
&= - \sum_{\substack{i\in[L],k\in[K],\\ j\in[L]:\rho(k)\notin\mathsf{Att}_j}} \widetilde{b}_{i,k} \cdot \widehat{\mathrm{S}}\widehat{\mathrm{B}}_j\widehat{\mathrm{T}}_{j,\rho(k)}/\widehat{\mathrm{B}}_i + \sum_{i\in[L],k\in[K]} \widetilde{b}_{i,k} m_{k,1} \cdot \widehat{\Delta}\widehat{\mathrm{S}}/\widehat{\mathrm{B}}_i + \sum_{i\in[L],k\in[K],\mu\in[2,N]} \widetilde{b}_{i,k} m_{k,\mu} \cdot \widehat{\Delta}\widehat{\mathrm{V}}_\mu/\widehat{\mathrm{B}}_i \\
&= - \sum_{i\in[L],k\in[K]} \widetilde{b}_{i,k} \cdot \widehat{\mathrm{S}}\widehat{\mathrm{T}}_{i,\rho(k)} - \sum_{\substack{i\in[L],k\in[K],\\ j\in[L]\backslash\{i\}:\rho(k)\notin\mathsf{Att}_j}} \widetilde{b}_{i,k} \cdot \widehat{\mathrm{S}}\widehat{\mathrm{B}}_j\widehat{\mathrm{T}}_{j,\rho(k)}/\widehat{\mathrm{B}}_i \\
&\qquad + \sum_{i\in[L],k\in[K]} \widetilde{b}_{i,k} m_{k,1} \cdot \widehat{\Delta}\widehat{\mathrm{S}}/\widehat{\mathrm{B}}_i + \sum_{i\in[L],k\in[K],\mu\in[2,N]} \widetilde{b}_{i,k} m_{k,\mu} \cdot \widehat{\Delta}\widehat{\mathrm{V}}_\mu/\widehat{\mathrm{B}}_i \qquad (29)
\end{aligned}
$$

In the last line of Eq. (29), we single out the terms in the first summation of the first line with $\rho(k) \notin \mathsf{Att}_i$, (*i.e.* taking $j = i$) for each summations. We observe that f'_3 contains the term $\widehat{\mathrm{S}}\widehat{\mathrm{T}}_{i,\rho(k)}$ with no-zero coefficient $\widetilde{b}_{i,k}$ which appears in the left most summation in the last line of Eq. (29). It can be seen that the term cannot be cancelled inside the above summation since ρ is *injective* and thus $\widehat{\mathrm{S}}\widehat{\mathrm{T}}_{i,\rho(k)}$ are distinct monomials for $k \in [K]$. We claim that the term does not appear in f'_1, f'_2 and f'_4.

- It can be observed from Eq. (27) that no monomial of f_2' contains $\widehat{\mathrm{T}}_{i,\rho(k)}$. Therefore, f_2' does not contain the monomial $\widehat{\mathrm{S}}\widehat{\mathrm{T}}_{i,\rho(k)}$.
- It is easy to observe that f_4' can be represented as a linear combination of $\widehat{\mathrm{S}}$, 1, and $\{\widehat{\mathrm{S}}\widehat{\mathrm{A}}_\ell\}_\ell$. Therefore, it does not contain the term $\widehat{\mathrm{S}}\widehat{\mathrm{T}}_{i,\rho(k)}$.
- We then show that f_1' cannot contain the term either. To show this, we consider each combination of $(X, Y) \in (V_1 \times V_2) \backslash (\widetilde{V}_1 \times \widetilde{V}_2)$ where $\widetilde{V}_1 = \{\widehat{\mathrm{C}}_2, \{\widehat{\mathrm{C}}_{5,k}\}_{k\in[K]}\}, \widetilde{V}_2 = \{1/\widehat{\mathrm{B}}_i\}_{i\in[L]}$ and show that the resulting element that is obtained by replacing $\widehat{\mathrm{C}}_2, \widehat{\mathrm{C}}_3, \widehat{\mathrm{C}}_4, \{\widehat{\mathrm{C}}_{5,k}\}_{k\in[K]}$ that appear in $X \cdot Y$ with the polynomials C_2, C_3, C_4 and $C_{5,k}$, and then replacing $\{\widehat{\mathrm{R}}_i\}_{i\in\mathsf{Cor}}$ with $\{r_i\}_{i\in\mathsf{Cor}}$ does not contain the desired term $\widehat{\mathrm{S}}\widehat{\mathrm{T}}_{i,\rho(k)}$. We first observe that X should be one of $\widehat{\mathrm{C}}_2$, $\widehat{\mathrm{C}}_3$, $\widehat{\mathrm{C}}_4$ and $\{\widehat{\mathrm{C}}_{5,k}\}_{k\in[K]}$ since otherwise the resulting element does not contain the multiplicative factor $\widehat{\mathrm{S}}$.
  - We first consider the case of $X = \widehat{\mathrm{C}}_2$. In this case, Y should be chosen from $V_2 \backslash \widetilde{V}_2$. Recall that $L_1'[h_j]$ in $\widehat{\mathrm{C}}_2$ takes the form $L_1'[h_j] = r_j^* \widehat{\mathrm{B}}_j + \tau_j^* \widehat{\mathrm{B}}_j \widehat{\mathrm{R}}_j$ for $j \in \mathsf{Mal}$ due to **Game 4**. Note that, in this case, the term XY is not a monomial, but a sum of monomials. Therefore, the resulting terms of $X \cdot Y$ are multiplied by $\widehat{\mathrm{B}}_i$ for some i and thus does not yield the term $\widehat{\mathrm{S}}\widehat{\mathrm{T}}_{i,\rho(k)}$.
  - We then consider the case of $X = \widehat{\mathrm{C}}_3$. In this case, Y is chosen from V_2. Then, the resulting element is either multiplied by $1/\widehat{\Gamma}$ or $\widehat{\mathrm{B}}_i/\widehat{\mathrm{B}}_j$ with $i \neq j$. Therefore, it does not yield the term either.
  - We then consider the case of $X = \widehat{\mathrm{C}}_4$. In this case, Y is chosen from V_2. Then, the resulting element is either multiplied by $1/\widehat{\Pi}$ or $\widehat{\mathrm{B}}_i/\widehat{\mathrm{B}}_j$ with $i \neq j$. Therefore, it does not yield the term $\widehat{\mathrm{S}}\widehat{\mathrm{T}}_{i,\rho(k)}$.
  - We next consider the case of $X = \widehat{\mathrm{C}}_{5,k}$. In this case, Y should be chosen from $V_2 \backslash \widetilde{V}_2$. Note that, in this case, the term XY is not a monomial, but a sum of monomials. However, the resulting element is either multiplied by $\widehat{\Delta}$ or $\widehat{\mathrm{B}}_j$ for some $j \in [L]$. Therefore, it does not yield the term $\widehat{\mathrm{S}}\widehat{\mathrm{T}}_{i,\rho(k)}$.

Therefore, the monomial $\widehat{\mathrm{S}}\widehat{\mathrm{T}}_{i,\rho(k)}$ that appears in f_3' cannot be cancelled by any term from f_1', f_2' or f_4'. This implies $f' = f_1' + f_2' + f_3' + f_4' \neq 0$ over $\mathbb{T}'$ as desired. Therefore, if $f \in \mathbb{T}$ passes the zero-test then it must hold that $\widetilde{b}_{i,k} = 0$ for all $i \in [L], k \in [K]$ with $\rho(k) \notin \mathsf{Att}_i$. This completes the proof. □

Lemma 14 *For any adversary, it holds that* $\Pr[\mathsf{E}_6] = \Pr[\mathsf{E}_7]$.

Proof. The only difference between Game 6 and Game 7 is in the way zero-test queries are answered. The only possible case when Game 7 behaves differently from Game 6 in $\mathcal{A}$'s view is the case where the following event occurs: $\mathcal{A}$ submits a handle for a polynomial $f \in \mathbb{T}$ to the zero-test oracle such that there exists $(i, \mu) \in [L] \times [2, N]$ satisfying $\sum_{k\in[K]} \widetilde{b}_{i,k} m_{k,\mu} \neq 0$ where $m_{k,\mu}$ is the (k, μ)-th entry of the LSSS matrix $\mathbf{M}$ of the challenge policy and the associated polynomial $f' \in \mathbb{T}'$ equals to 0 in $\mathbb{T}'$. Here, we prove that f' cannot be 0 for such f.

We divide f into f_1, f_2, f_3, and f_4 as in Eq. (19) and define the corresponding polynomials f_1', f_2', f_3', and f_4' as in the proof of Lemma 13. We can write f_3' as

$$
\begin{aligned}
f_3' &= \sum_{i\in[L],k\in[K]} \widetilde{b}_{i,k} \cdot C_{5,k} \cdot (1/\widehat{\mathrm{B}}_i) = \sum_{\substack{i\in[L],k\in[K]:\\ \rho(k)\in\mathsf{Att}_i}} \widetilde{b}_{i,k} \cdot C_{5,k} \cdot (1/\widehat{\mathrm{B}}_i) \\
&= - \sum_{\substack{i\in[L],k\in[K],\\ j\in[L]\backslash\{i\}:\rho(k)\notin\mathsf{Att}_j}} \widetilde{b}_{i,k} \cdot \widehat{\mathrm{S}}\widehat{\mathrm{B}}_j \widehat{\mathrm{T}}_{j,\rho(k)}/\widehat{\mathrm{B}}_i + \sum_{\substack{i\in[L],k\in[K]:\\ \rho(k)\in\mathsf{Att}_i}} \widetilde{b}_{i,k} m_{k,1} \cdot \widehat{\mathrm{S}}\widehat{\Delta}/\widehat{\mathrm{B}}_i \\
&\qquad + \sum_{\substack{i\in[L],k\in[K],\\ \mu\in[2,N]:\rho(k)\in\mathsf{Att}_i}} \widetilde{b}_{i,k} m_{k,\mu} \cdot \widehat{\Delta}\widehat{\mathrm{V}}_\mu/\widehat{\mathrm{B}}_i \qquad (30)
\end{aligned}
$$

Firstly, note that in the first line of Eq. (30), we use the fact that $\widetilde{b}_{i,k} = 0$ for all (i,k) such that $\rho(k) \notin \mathsf{Att}_i$ due to **Game 6**. Now, for a fixed $i \in [L]$ and $\mu \in [2,N]$ if $\sum_{k\in[K]:\rho(k)\in\mathsf{Att}_i} \widetilde{b}_{i,k} m_{k,\mu} \neq 0$ then the term $\widehat{\Delta}\widehat{\mathrm{V}}_\mu/\widehat{\mathrm{B}}_i$ will present with non-zero coefficient in f_3'. By inspection, it is easy to see that the term cannot be cancelled inside the above summation. Moreover, we observe that the term cannot be cancelled by any term of f_1', f_2' and f_4' since no element of these polynomials contains $\widehat{\mathrm{V}}_\mu$. This implies $f' = f_1' + f_2' + f_3' + f_4' \neq 0$ over $\mathbb{T}'$ as desired. This completes the proof. Therefore, if $f \in \mathbb{T}$ passes the zero-test query then f_3' must take the form

$$f_3' = - \sum_{\substack{i\in[L],k\in[K],\\ j\in[L]\setminus\{i\}:\rho(k)\notin\mathsf{Att}_j}} \widetilde{b}_{i,k} \cdot \widehat{\mathrm{S}}\widehat{\mathrm{B}}_j \widehat{\mathrm{T}}_{j,\rho(k)}/\widehat{\mathrm{B}}_i + \sum_{\substack{i\in[L],k\in[K]:\\ \rho(k)\in\mathsf{Att}_i}} \widetilde{b}_{i,k} m_{k,1} \cdot \widehat{\mathrm{S}}\widehat{\Delta}/\widehat{\mathrm{B}}_i \tag{31}$$

Lemma 15 *For any adversary, it holds that* $\Pr[\mathsf{E}_7] = \Pr[\mathsf{E}_8]$.

Proof. The only difference between Game 7 and Game 8 is in the way zero-test queries are answered. The only possible case when Game 8 behaves differently from Game 7 in $\mathcal{A}$'s view is the case where the following event occurs: $\mathcal{A}$ submits a handle for a polynomial $f \in \mathbb{T}$ to the zero-test oracle such that $b_i \neq 0$ for any $i \notin \mathcal{I}_{(\mathbf{M},\rho)}$ and the associated polynomial $f' \in \mathbb{T}'$ equals to 0 in $\mathbb{T}'$. Here, we prove that f' cannot be 0 for such f.

We divide f into f_1, f_2, f_3, and f_4 as in Eq. (19) and define the corresponding polynomials f_1', f_2', f_3', and f_4' as in the proof of Lemma 13. We first recall that f_2' can be expressed as Eq. (27). Noting that $b_i = 0$ for all $i \notin \mathsf{Cor} \cup \mathsf{Mal}$, Eq. (27) can be (slightly) simplified as follows:

$$f_2' = - \sum_{i\in\mathsf{Cor}\cup\mathsf{Mal}} b_i \cdot \widehat{\mathrm{S}}\widehat{\Delta}/\widehat{\mathrm{B}}_i - \sum_{i\in\mathsf{Cor}\setminus\mathsf{Mal}} b_i r_i \cdot \widehat{\mathrm{S}} - \sum_{i\in\mathsf{Mal}} b_i \cdot \widehat{\mathrm{S}}(r_i^* + \tau_i^* \widehat{\mathrm{R}}_i) - \sum_{i\in[L]} b_i \cdot \langle \boldsymbol{y}_i, \widehat{\mathbf{A}} \rangle \cdot \widehat{\mathrm{S}} - \Phi$$

where Φ is defined as in Eq. (28). We then show that if $b_i \neq 0$ for any $i \in (\mathsf{Cor} \cup \mathsf{Mal}) \setminus \mathcal{I}_{(\mathbf{M},\rho)}$ then the term $\widehat{\mathrm{S}}\widehat{\Delta}/\widehat{\mathrm{B}}_i$ must appear with non-zero coefficient in f'. Note that the term cannot be cancelled inside f_2' since no other term except $\widehat{\mathrm{S}}\widehat{\Delta}/\widehat{\mathrm{B}}_i$ contains $\widehat{\Delta}$. We now show that the term does not appear in f_1', f_3' and f_4'.

- It is easy to see that f_4' can be represented as a linear combination of $\widehat{\mathrm{S}}, 1$, and $\{\widehat{\mathrm{S}}\widehat{\mathrm{A}}_\ell\}_\ell$. There fore, it does not contain the term $\widehat{\mathrm{S}}\widehat{\Delta}/\widehat{\mathrm{B}}_i$.
- We claim that f_3' does not contain the term $\widehat{\mathrm{S}}\widehat{\Delta}/\widehat{\mathrm{B}}_i$ if $i \notin \mathcal{I}_{(\mathbf{M},\rho)}$. Recall that f_3' takes the form as in Eq. 31 which contains $\widehat{\mathrm{S}}\widehat{\Delta}/\widehat{\mathrm{B}}_i$ with coefficient $\sum_{k\in[K]:\rho(k)\in\mathsf{Att}_i} \widetilde{b}_{i,k} m_{k,1}$. By the property of LSSS, it is known that $(1,0,\ldots,0) \notin \mathsf{Span}\{\mathbf{M}_{\rho(k)} : \rho(k) \in \mathsf{Att}_i \text{ and } i \notin \mathcal{I}_{(\mathbf{M},\rho)}\}$ where $\mathbf{M}_{\rho(k)}$ denotes the $\rho(k)$-th row of the matrix $\mathbf{M}$. Also, we observe from Eq. 31 that f_3' does not contain any monomial involving $\widehat{\mathrm{V}}_\mu$ for $\mu \in [2,N]$. Therefore, if $\sum_{k\in[K]:\rho(k)\in\mathsf{Att}_i} \widetilde{b}_{i,k} m_{k,1} \neq 0$ for some $i \notin \mathcal{I}_{(\mathbf{M},\rho)}$ then it means that $(\widetilde{b}_{i,k}, 0, \ldots, 0) \in \mathsf{Span}\{\mathbf{M}_{\rho(k)} : \rho(k) \in \mathsf{Att}_i \text{ and } i \notin \mathcal{I}_{(\mathbf{M},\rho)}\}$ which contradicts the fact that $(1,0,\ldots,0) \notin \mathsf{Span}\{\mathbf{M}_{\rho(k)} : \rho(k) \in \mathsf{Att}_i \text{ and } i \notin \mathcal{I}_{(\mathbf{M},\rho)}\}$. This implies that $\sum_{k\in[K]:\rho(k)\in\mathsf{Att}_i} \widetilde{b}_{i,k} m_{k,1} = 0$ for all $i \notin \mathcal{I}_{(\mathbf{M},\rho)}$. In particular, Eq. 31 is slightly simplified as

$$f_3' = - \sum_{\substack{i\in[L],k\in[K],\\ j\in[L]\setminus\{i\}:\rho(k)\notin\mathsf{Att}_j}} \widetilde{b}_{i,k} \cdot \widehat{\mathrm{S}}\widehat{\mathrm{B}}_j \widehat{\mathrm{T}}_{j,\rho(k)}/\widehat{\mathrm{B}}_i + \sum_{\substack{i\in\mathcal{I}_{(\mathbf{M},\rho)},k\in[K]:\\ \rho(k)\in\mathsf{Att}_i}} \widetilde{b}_{i,k} m_{k,1} \cdot \widehat{\mathrm{S}}\widehat{\Delta}/\widehat{\mathrm{B}}_i \tag{32}$$

  Therefore, f_3' cannot contain $\widehat{\mathrm{S}}\widehat{\Delta}/\widehat{\mathrm{B}}_i$ if $i \notin \mathcal{I}_{(\mathbf{M},\rho)}$.
- We then show that f_1' cannot contain the term either. To show this, we consider each combination of $(X,Y) \in (V_1 \times V_2) \setminus (\widetilde{V}_1 \times \widetilde{V}_2)$ where $\widetilde{V}_1 = \{\widehat{\mathrm{C}}_2, \{\widehat{\mathrm{C}}_{5,k}\}_{k\in[K]}\}, \widetilde{V}_2 = \{1/\widehat{\mathrm{B}}_i\}_{i\in[L]}$ and show that the resulting element that is obtained by replacing $\widehat{\mathrm{C}}_2, \widehat{\mathrm{C}}_3, \widehat{\mathrm{C}}_4, \{\widehat{\mathrm{C}}_{5,k}\}_{k\in[K]}$ that appear in $X \cdot Y$ with the polynomials C_2, C_3, C_4 and $C_{5,k}$, and then replacing $\{\widehat{\mathrm{R}}_i\}_{i\in\mathsf{Cor}}$ with $\{r_i\}_{i\in\mathsf{Cor}}$ does not contain the desired term $\widehat{\mathrm{S}}\widehat{\Delta}/\widehat{\mathrm{B}}_i$ if $i \notin \mathcal{I}_{(\mathbf{M},\rho)}$. We first observe that X should be one of $\widehat{\mathrm{C}}_2$ and $\{\widehat{\mathrm{C}}_{5,k}\}_{k\in[K]}$ since otherwise the resulting element does not contain the multiplicative factor $\widehat{\Delta}$.

– We first consider the case of $X = \widehat{\mathrm{C}}_2$. In this case, Y should be chosen from $V_2 \backslash \widetilde{V}_2$. Recall that $L'_1[h_j]$ in $\widehat{\mathrm{C}}_2$ does not contain $\widehat{\Delta}$ since it takes the form $L'_1[h_j] = r_j^* \widehat{\mathrm{B}}_j + \tau_j^* \widehat{\mathrm{B}}_j \widehat{\mathrm{R}}_j$ for $j \in \mathsf{Mal}$ due to **Game 4**. Therefore, the resulting terms of $X \cdot Y$ are either multiplied by $\widehat{\mathrm{B}}_i$ for some i or $\widehat{\mathrm{B}}_i/\widehat{\mathrm{B}}_j$ with $i \neq j$, and thus does not yield the term $\widehat{\mathrm{S}}\widehat{\Delta}/\widehat{\mathrm{B}}_i$.
– We next consider the case of $X = \widehat{\mathrm{C}}_{5,k}$. In this case, Y should be chosen from $V_2 \backslash \widetilde{V}_2$. Then, the resulting element is either multiplied by $\widehat{\mathrm{B}}_i$ for some $j \in [L]$ or $\widehat{\mathrm{B}}_i/\widehat{\mathrm{B}}_j$ with $i \neq j$. Therefore, it does not yield the term either.

Therefore, the term $\widehat{\mathrm{S}}\widehat{\Delta}/\widehat{\mathrm{B}}_i$ with $i \notin \mathcal{I}_{(\mathbf{M},\rho)}$ that appears in f'_2 cannot be cancelled by any term from f'_1, f'_3 or f'_4. This implies $f' = f'_1 + f'_2 + f'_3 + f'_4 \neq 0$ over $\mathbb{T}'$ as desired. This completes the proof.

Lemma 16 *For any adversary, it holds that* $\Pr[\mathsf{E}_8] = \Pr[\mathsf{E}_9]$.

Proof. The only difference between Game 8 and Game 9 is in the way zero-test queries are answered. The only possible case when Game 9 behaves differently from Game 8 in $\mathcal{A}$'s view is the case where the following event occurs: $\mathcal{A}$ submits a handle for a polynomial $f \in \mathbb{T}$ to the zero-test oracle such that $b_i = 0$ for all $i \notin (\mathsf{Cor} \cup \mathsf{Mal}) \cap \mathcal{I}_{(\mathbf{M},\rho)}$ and $\boldsymbol{d} \neq \sum_{i \in (\mathsf{Cor} \cup \mathsf{Mal}) \cap \mathcal{I}_{(\mathbf{M},\rho)}} b_i \boldsymbol{y}_i$, but the associated polynomial $f' \in \mathbb{T}'$ equals to 0 in $\mathbb{T}'$. Here, we prove that this cannot happen.

We divide f into f_1, f_2, f_3, and f_4 as in Eq. (19) and define the corresponding polynomials f'_1, f'_2, f'_3, and f'_4 as in the proof of Lemma 13. Recall that f'_2 can be expressed as Eq. (27). Noting that $b_i = 0$ for all $i \notin (\mathsf{Cor} \cup \mathsf{Mal}) \cap \mathcal{I}_{(\mathbf{M},\rho)}$, Eq. (27) can be (slightly) simplified as follows:

$$f'_2 = \underbrace{- \sum_{i \in (\mathsf{Cor} \cup \mathsf{Mal}) \cap \mathcal{I}_{(\mathbf{M},\rho)}} b_i \cdot \widehat{\mathrm{S}}\widehat{\Delta}/\widehat{\mathrm{B}}_i - \sum_{i \in (\mathsf{Cor} \cup \mathsf{Mal}) \cap \mathcal{I}_{(\mathbf{M},\rho)}} b_i \cdot \langle \boldsymbol{y}_i, \widehat{\mathbf{A}} \rangle \cdot \widehat{\mathrm{S}} - \sum_{i \in (\mathsf{Cor} \setminus \mathsf{Mal}) \cap \mathcal{I}_{(\mathbf{M},\rho)}} b_i r_i \cdot \widehat{\mathrm{S}} - \sum_{i \in \mathsf{Mal} \cap \mathcal{I}_{(\mathbf{M},\rho)}} b_i \cdot \widehat{\mathrm{S}}(r_i^* + \tau_i^* \widehat{\mathrm{R}}_i) - \Phi}_{:=\Psi}$$

where Φ is defined as in Eq. (28). We also have

$$f'_4 = d_0 \widehat{\mathrm{S}} + \langle \boldsymbol{d}, \widehat{\mathbf{A}} \rangle \cdot \widehat{\mathrm{S}} + \langle \boldsymbol{x}^*_{\mathsf{coin}}, \boldsymbol{d} \rangle,$$

which is easy to observe. First, we see that f'_3 does not contain the term $\widehat{\mathrm{S}}\widehat{\mathrm{A}}_\ell$ which is easy to observe from Eq. 32. We then show that f'_1 also does not contain the term of the form $\widehat{\mathrm{S}}\widehat{\mathrm{A}}_\ell$ for any ℓ. This is sufficient to complete the proof, since

$$f' = f'_1 + f'_2 + f'_3 + f'_4 = \left\langle \boldsymbol{d} - \sum_{i \in (\mathsf{Cor} \cup \mathsf{Mal}) \cap \mathcal{I}_{(\mathbf{M},\rho)}} b_i \cdot \boldsymbol{y}_i, \widehat{\mathbf{A}} \right\rangle \cdot \widehat{\mathrm{S}} + f'_1 + \Psi + d_0 \widehat{\mathrm{S}} + \langle \boldsymbol{x}^*_{\mathsf{coin}}, \boldsymbol{d} \rangle,$$

does not equal to 0 unless $\boldsymbol{d} - \sum_{i \in \mathsf{Cor} \cup \mathsf{Mal}} b_i \cdot \boldsymbol{y}_i = 0$, which can be seen by observing that $f'_1 + \Psi + d_0 \widehat{\mathrm{S}} + \langle \boldsymbol{x}^*_{\mathsf{coin}}, \boldsymbol{d} \rangle$ does not contain any term of the form $\widehat{\mathrm{S}}\widehat{\mathrm{A}}_\ell$.

We then move to show that f'_1 does not contain the term of the form $\widehat{\mathrm{S}}\widehat{\mathrm{A}}_\ell$ for any ℓ. To show this, we consider each combination of $(X, Y) \in (V_1 \times V_2) \backslash (\widetilde{V}_1 \times \widetilde{V}_2)$ where $\widetilde{V}_1 = \{\widehat{\mathrm{C}}_2, \{\widehat{\mathrm{C}}_{5,k}\}_{k \in \in [K]}\}, \widetilde{V}_2 = \{1/\widehat{\mathrm{B}}_i\}_{i \in [L]}$ and show that the resulting element that is obtained by replacing $\widehat{\mathrm{C}}_2, \widehat{\mathrm{C}}_3, \widehat{\mathrm{C}}_4, \{\widehat{\mathrm{C}}_{5,k}\}_{k \in [K]}$ that appear in $X \cdot Y$ with the polynomials C_2, C_3, C_4 and $C_{5,k}$, and then replacing $\{\widehat{\mathrm{R}}_i\}_{i \in \mathsf{Cor}}$ with $\{r_i\}_{i \in \mathsf{Cor}}$ does not contain the desired term $\widehat{\mathrm{S}}\widehat{\mathrm{A}}_\ell$. We first observe that X should be among $\widehat{\mathrm{C}}_2$, $\widehat{\mathrm{C}}_3, \widehat{\mathrm{C}}_4$ and $\widehat{\mathrm{C}}_{5,k}$, since otherwise the resulting element does not contain the multiplicative factor $\widehat{\mathrm{S}}$.

- We first consider the case of $X = \widehat{\mathrm{C}}_2$. In this case, Y should be chosen from $V_2 \backslash \widetilde{V}_2$. Note that, in this case, the term XY is not a monomial, but a sum of monomials. However, then the resulting terms are multiplied by $\widehat{\mathrm{B}}_i$ for some i and thus does not yield the term.

- We consider the case of $X = \widehat{\mathrm{C}}_3$. If we multiply Y from V_2, the resulting element either carries the multiplicative factor $1/\widehat{\Gamma}$ or $\widehat{\mathrm{B}}_i/\widehat{\mathrm{B}}_j$ with $i \neq j$. In any case, it does not yield the term $\widehat{\mathrm{S}}\widehat{\mathrm{A}}_\ell$.
- Next, we consider the case $X = \widehat{\mathrm{C}}_4$. If we multiply Y from V_2, the resulting element either carries the multiplicative factor $1/\widehat{\Pi}$ or $\widehat{\mathrm{B}}_i/\widehat{\mathrm{B}}_j$ with $i \neq j$. In any case, it does not yield the desired term.
- We then consider the case of $X = \widehat{\mathrm{C}}_{5,k}$. In this case, Y should be chosen from $V_2 \backslash \widetilde{V}_2$. Note that, in this case, the term XY is not a monomial, but a sum of monomials. The resulting element is either multiplied by $\widehat{\Delta}$ or $\widehat{\mathrm{B}}_i$ for some $i \in [L]$. Therefore, it does not yield the term $\widehat{\mathrm{S}}\widehat{\mathrm{A}}_\ell$.

As discussed above, this completes the proof. □

Lemma 17 *For any adversary, it holds that* $\Pr[\mathsf{E}_9] = 1/2$.

Proof. We show that answer to any zero-test query made by $\mathcal{A}$ does not depend on the challenge bit coin and therefore the view of $\mathcal{A}$ is independent from the value of coin. To see this, let us consider $f \in \mathbb{T}$ that corresponds to a zero-test query made by $\mathcal{A}$ during the game and prove that f' corresponds to the same element in $\mathbb{T}'$ regardless of the value of coin if f satisfies the restrictions that are introduced in **Game 5 − 9**. For the analysis, we divide f into f_1, f_2, f_3, and f_4 as in Eq. (19) and define the corresponding polynomials f'_1, f'_2, f'_3, and f'_4 as in the proof of Lemma 16.

We first claim that regardless of the value of coin, f'_1, f'_2 and f'_3 correspond to the same elements in $\mathbb{T}'$. This can be easily seen by observing that f_1, f_2 and f_3 do not include the formal variables $\{\widehat{\mathrm{C}}_{1,\ell}\}_\ell$, which are the only variables that depend on the value of coin when they are replaced with the polynomials in $\mathbb{T}'$.

We then prove the same statement for f'_4. We observe:

$$f'_4 = d_0\widehat{\mathrm{S}} + \langle \boldsymbol{d}, \widehat{\mathbf{A}} \rangle \cdot \widehat{\mathrm{S}} + \langle \boldsymbol{x}^*_{\mathsf{coin}}, \boldsymbol{d} \rangle = d_0\widehat{\mathrm{S}} + \langle \boldsymbol{d}, \widehat{\mathbf{A}} \rangle \cdot \widehat{\mathrm{S}} + \sum_{i \in (\mathsf{Cor} \cup \mathsf{Mal}) \cap \mathcal{I}_{(\mathbf{M},\rho)}} b_i \cdot \langle \boldsymbol{x}^*_{\mathsf{coin}}, \boldsymbol{y}_i \rangle.$$

Since $\langle \boldsymbol{x}^*_0, \boldsymbol{y}_i \rangle = \langle \boldsymbol{x}^*_1, \boldsymbol{y}_i \rangle$ for all $i \in (\mathsf{Cor} \cup \mathsf{Mal}) \cap \mathcal{I}_{(\mathbf{M},\rho)}$ by the admissibility condition, the above refers to the same element in $\mathbb{T}'$ regardless of whether $\mathsf{coin} = 0$ or $\mathsf{coin} = 1$.

Since each of f'_1, f'_2, f'_3, and f'_4 corresponds to the same element in $\mathbb{T}'$ regardless of whether $\mathsf{coin} = 0$ or $\mathsf{coin} = 1$, the same holds for $f' = f'_1 + f'_2 + f'_3 + f'_4$. This completes the proof. □

Finally, the proof of Theorem 2 follows by combining the proofs of Lemma 8 to 17. □

Registered ABIPFE from pairings: Plugging our slotted registered ABIPFE into the transformation described in Section 9, we achieve the following corollary:

Corollary 2 (Bounded Registered ABIPFE) Let λ be a security parameter. Let $\mathcal{U}_{\mathsf{att}}$ be any (polynomial-size) attribute space, and let $\mathcal{P}$ be a set of policies that can be described by a one-use LSSS over $\mathcal{U}_{\mathsf{att}}$. Then, under generic bilinear group model, for every polynomial $L = L(\lambda)$ and an integer $n \in \mathbb{N}$, there exists a bounded registered ABIPFE scheme with function space $\mathcal{U}_F = \mathsf{PSet}(\mathcal{U}_{\mathsf{att}}) \times \mathbb{Z}^n$, message space $\mathcal{M} = \mathcal{P} \times \mathbb{Z}^n$, and supporting up to L users with the following properties:

- The size of the common reference string and the size of the auxiliary data maintained by the key curator is $L^2 \cdot \mathsf{poly}(\lambda, n, |\mathcal{U}_{\mathsf{att}}|, \log L)$.
- The running times of key-generation and registration are $L \cdot \mathsf{poly}(\lambda, |\mathcal{U}_{\mathsf{att}}|, \log L)$ and $L \cdot \mathsf{poly}(\lambda, n, |\mathcal{U}_{\mathsf{att}}|, \log L)$ respectively.
- The size of the master public key and the helper decryption keys are both $|\mathcal{U}_{\mathsf{att}}| \cdot \mathsf{poly}(\lambda, n, \log L)$.
- The size of a ciphertext is $K \cdot \mathsf{poly}(\lambda, n, \log L)$, where K denotes the number of rows in the LSSS matrix $\mathbf{M}$ associated with the access policy.

8 Slotted Registered FE from Indistinguishability Obfuscation

This section is devoted to present the slotted registered FE scheme that can be lifted to a registered FE scheme with *no* a-priori bound on the number of users in the system. The construction transforms the IO-based registered ABE of Hohenberger et al. [HLWW23] into the setting of functional encryption. Before we proceed, let us discuss the cryptographic building blocks required for the construction.

8.1 Cryptographic Tools

Definition 7 (Pseudorandom Generator [HG90, Lev85]) A pseudorandom generator (PRG) $\mathsf{PRG} : \{0,1\}^\lambda \to \{0,1\}^{\lambda+\ell(\lambda)}$ with stretch $\ell(\lambda)$ (ℓ is some polynomial function) is a polynomial-time computable function that satisfies the following. For any PPT adversary $\mathcal{A}$, it holds that

$$|\Pr[\mathcal{A}(\mathsf{PRG}(s)) = 1 : s \leftarrow \{0,1\}^\lambda] - \Pr[\mathcal{A}(r) : r \leftarrow \{0,1\}^{\lambda+\ell(\lambda)}] \leq \mathsf{negl}(\lambda)$$

.

Definition 8 (Secret Key Encryption) Let λ be a security parameter and let p, q, r and s be some polynomials. A secret key encryption scheme is a tuple of algorithms $\mathsf{SKE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ with plaintext space $\mathcal{M} := \{0,1\}^n$, ciphertext space $\mathcal{C} := \{0,1\}^{\ell_c(\lambda)}$, and secret key space $\mathcal{SK} := \{0,1\}^{\ell_k(\lambda)}$.

Setup$(1^\lambda) \to \mathsf{sk}$: The setup algorithm takes the security parameter 1^λ as input and outputs a secret key $\mathsf{sk} \in \mathcal{SK}$.

Enc$(\mathsf{sk}, m) \to \mathsf{ct}$: The encryption algorithm takes sk and a plaintext $m \in \mathcal{M}$ as input, and outputs a ciphertext $\mathsf{ct} \in \mathcal{C}$.

Dec$(\mathsf{sk}, \mathsf{ct}) \in \mathcal{M} \cup \{\perp\}$: The decryption algorithm takes sk and ct as input, and outputs a plaintext $m' \in \mathcal{M}$ or $\perp$.

The algorithms must satisfy the following properties:

Correctness: There exists a negligible function negl such that for any $\lambda \in \mathbb{N}$ and $m \in \mathcal{M}$,

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \neq m \ : \ \begin{array}{l} \mathsf{sk} \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{sk}, m) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

Security: Let $\mathsf{SKE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ be a SKE scheme. We consider the following security experiment $\mathsf{Expt}^{\mathsf{SKE}}_{\mathcal{A}}(\lambda, b)$ against a PPT adversary $\mathcal{A}$.

1. The challenger computes $\mathsf{sk} \leftarrow \mathsf{Setup}(1^\lambda)$.
2. $\mathcal{A}$ sends an encryption query m to the challenger. The challenger computes $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{sk}, m)$ and returns ct to $\mathcal{A}$. $\mathcal{A}$ can repeat this process polynomially many times.
3. $\mathcal{A}$ sends $(m_0, m_1) \in \mathcal{M}^2$ to the challenger.
4. The challenger computes $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{sk}, m_b)$ and sends ct to $\mathcal{A}$.
5. $\mathcal{A}$ sends an encryption query m to the challenger. The challenger computes $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{sk}, m)$ and returns ct to $\mathcal{A}$. $\mathcal{A}$ can repeat this process polynomially many times.
6. $\mathcal{A}$ outputs $b' \in \{0,1\}$. This is the output of the experiment.

We say that SKE is IND-CPA secure if, for any PPT $\mathcal{A}$, it holds that

$$|\Pr[\mathsf{Expt}^{\mathsf{SKE}}_{\mathcal{A}}(\lambda, 0) = 1] - \Pr[\mathsf{Expt}^{\mathsf{SKE}}_{\mathcal{A}}(\lambda, 1) = 1]| \leq \mathsf{negl}(\lambda).$$

Definition 9 (Indistinguishability Obfuscation (IO) [BGI+01, GGH+13]) A PPT algorithm $i\mathcal{O}$ is a secure IO for a class of circuits $\{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ if it satisfies the following two conditions.

- **Functionality:** For any security parameter $\lambda \in \mathbb{N}$, circuit $C \in \mathcal{C}_\lambda$, and input x, we have that

$$\Pr[C'(x) = C(x) \mid C' \leftarrow i\mathcal{O}(1^\lambda, C)] = 1 \ .$$

- **Indistinguishability:** For any pair of circuits $C_0, C_1 \in \mathcal{C}_\lambda$ satisfying $C_0(x) = C_1(x), \forall x$ and any PPT distinguisher $\mathcal{D}$, the following holds:

$$\left|\Pr\left[\mathcal{D}(i\mathcal{O}(1^\lambda, C_0)) = 1\right] - \Pr\left[\mathcal{D}(i\mathcal{O}(1^\lambda, C_1)) = 1\right]\right| \leq \mathsf{negl}(\lambda).$$

Theorem 3 ([JLS22]) *Assume sub-exponential security of the following assumptions:*

- *the Learning Parity with Noise (LPN) assumption over general prime fields $\mathbb{F}_p$ with polynomially many LPN samples and error rate $1/k^\delta$, where k is the dimension of the LPN secret, and $\delta > 0$ is any constant;*
- *the existence of a Boolean Pseudo-Random Generator (PRG) in NC^0 with stretch $n^{1+\tau}$, where n is the length of the PRG seed, and $\tau > 0$ is any constant;*
- *the Decision Linear (DLIN) assumption on symmetric bilinear groups of prime order.*

Then, (subexponentially secure) indistinguishability obfuscation for all polynomial-size circuits exists. Assuming only polynomial security of the assumptions above yields polynomially secure functional encryption for all polynomial-size circuits.

Definition 10 (Somewhere Statistically Binding (SSB) Hash Function [HW15, OPWW15]) Let λ be a security parameter. A somewhere statistically binding (SSB) hash function with block length $\ell_{\mathsf{blk}} = \ell_{\mathsf{blk}}(\lambda)$, output length $\ell_{\mathsf{hash}} = \ell_{\mathsf{hash}}(\lambda)$, and opening length $\ell_{\mathsf{open}} = \ell_{\mathsf{open}}(\lambda)$ is a tuple of efficient algorithms $\mathsf{SSB} = (\mathsf{Setup}, \mathsf{Hash}, \mathsf{Open}, \mathsf{Vrfy})$ with the following properties:

Setup$(1^\lambda, 1^{\ell_{\mathsf{blk}}}, N, i^*) \to \mathsf{hk}$: The setup algorithm takes as input a security parameter λ, a block size ℓ_{blk}, the message length $N \leq 2^\lambda$, and an index $i^* \in [N]$, and outputs a hash key hk. Both N and i^* are encoded in *binary*; in particular, this means that $|\mathsf{hk}| = \mathsf{poly}(\lambda, \ell_{\mathsf{blk}}, \log N)$. We let $\Sigma = \{0,1\}^{\ell_{\mathsf{blk}}}$ denote the block alphabet.

Hash$(\mathsf{hk}, x) \to h$: the hash algorithm takes as input a hash key hk and input x, and outputs a hash value $h \in \{0,1\}^{\ell_{\mathsf{hash}}}$.

Open$(\mathsf{hk}, \mathbf{x}, i) \to \pi_i$: The open algorithm takes as input a hash key hk, an input $\mathbf{x} \in \Sigma^N$ and an index i, and outputs an opening $\pi_i \in \{0,1\}^{\ell_{\mathsf{open}}}$.

Vrfy$(\mathsf{hk}, h, i, x_i, \pi_i) \in \{0,1\}$: The verify algorithm takes as input a hash key hk, a hash value h, an index i, a value $x_i \in \Sigma$, and an opening $\pi_i \in \{0,1\}^{\ell_{\mathsf{open}}}$, and outputs a bit $b \in \{0,1\}$ indicating whether it accepts or rejects.

The algorithm must satisfy the following properties:

Correctness: For all security parameter $\lambda \in \mathbb{N}$, all block sizes $\ell_{\mathsf{blk}} = \ell_{\mathsf{blk}}(\lambda)$, all integers $N \leq 2^\lambda$, all indices $i, i^* \in [N]$, and any $\mathbf{x} \in \Sigma^N$,

$$\Pr\left[\mathsf{Vrfy}(\mathsf{hk}, h, i, x_i, \pi_i) = 1 : \begin{array}{c}\mathsf{hk} \leftarrow \mathsf{Setup}(1^\lambda, 1^{\ell_{\mathsf{blk}}}, N, i^*) \\ h \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathbf{x}); \pi_i \leftarrow \mathsf{Open}(\mathsf{hk}, \mathbf{x}, i)\end{array}\right] = 1.$$

Index hiding: For a bit $b \in \{0,1\}$ and an adversary $\mathcal{A}$, define the index hiding experiment $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{indexSSB}}(1^\lambda, b)$ as follows:

1. $\mathcal{A}$ chooses an integer N and two indices $i_0, i_1 \in [N]$.
2. The challenger samples $\mathsf{hk} \leftarrow \mathsf{Setup}(1^\lambda, 1^{\ell_{\mathsf{blk}}}, N, i_b)$ and gives hk to $\mathcal{A}$.
3. $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$, which is also the output of the experiment.

We require that for all polynomials $\ell_{\mathsf{blk}} = \ell_{\mathsf{blk}}(\lambda)$ and for all efficient adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

$$|\Pr[\mathsf{Expt}_{\mathcal{A}}^{\mathsf{indexSSB}}(1^\lambda, 0) = 1] - \Pr[\mathsf{Expt}_{\mathcal{A}}^{\mathsf{indexSSB}}(1^\lambda, 1) = 1]| = \mathsf{negl}(\lambda).$$

Somewhere statistically binding: We say that a hash key hk is statistically binding for an index $i^* \in [N]$ if there does not exists $h \in \{0,1\}^{\ell_{\mathsf{hash}}}$, $x \neq x' \in \Sigma$, and π, π' where $\mathsf{Vrfy}(\mathsf{hk}, h, i^*, x, \pi) = 1 = \mathsf{Vrfy}(\mathsf{hk}, h, i^*, x', \pi')$. We require that for all polynomial $\ell_{\mathsf{blk}} = \ell_{\mathsf{blk}}(\lambda)$ and for all $N \leq 2^\lambda$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$ and all $i \in [N]$,

$$\Pr[\mathsf{hk} \text{ is statistically binding for index } i : \mathsf{hk} \leftarrow \mathsf{Setup}(1^\lambda, 1^{\ell_{\mathsf{blk}}}, N, i)] = 1 - \mathsf{negl}(\lambda).$$

Succinctness: The hash length ℓ_{hash}, and opening length ℓ_{open} are all fixed polynomials in the security parameter λ and block size ℓ_{blk} (and independent of N).

Theorem 4 ([KLW15, OPWW15]) *Assuming indistinguishability obfuscation for polynomial-size circuits and one-way function, then for any polynomial block size $\ell_{\mathsf{blk}} = \ell_{\mathsf{blk}}(\lambda)$, there exists a somewhere statistically binding hash function for alphabet $\Sigma = \{0,1\}^{\ell_{\mathsf{blk}}}$.*

Subsequently, Okamoto et al. [OPWW15] designed SSB hash functions from various well-studied assumptions such as DDH, DCR, LWE, and others.

8.2 Construction

We use the following cryptographic tools as building blocks:

- A length doubling PRG $\mathsf{PRG} : \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$.
- A secret key encryption scheme $\mathsf{SKE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$.
- A somewhere statistically binding hash function $\mathsf{SSB} = (\mathsf{Setup}, \mathsf{Hash}, \mathsf{Open}, \mathsf{Vrfy})$.
- An indistinguishability obfuscation $i\mathcal{O}$ for $\mathsf{P/poly}$.

The slotted registered functional encryption $\mathsf{SlotRFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{IsValid}, \mathsf{Aggregate}, \mathsf{Enc}, \mathsf{Dec})$ for a function universe $\mathcal{U}_F = \{0,1\}^{\ell_f}$, and message space $\mathcal{M}$ works as follows:

Setup$(1^\lambda, 1^{\ell_f}, L)$: The setup algorithm takes the security parameter λ, the bit-length ℓ_f of a function in $\mathcal{U}_F$ (in unary) and the number of users L (in binary) as inputs and sets $\ell_{\mathsf{blk}} = \ell_f + 2\lambda$, computes $\mathsf{hk} \leftarrow \mathsf{SSB.Setup}(1^\lambda, 1^{\ell_{\mathsf{blk}}}, L, 1)$ and sets $\mathsf{crs} := \mathsf{hk}$. It outputs crs.

KeyGen(crs, i) : The key generation algorithm takes the common reference string crs, and a slot index $i \in [L]$ as inputs and samples $s_i \leftarrow \{0,1\}^\lambda$. It outputs the public key as $\mathsf{pk}_i := \mathsf{PRG}(s_i)$ and the secret key as $\mathsf{sk}_i := s_i$.

IsValid$(\mathsf{crs}, i, \mathsf{pk}_i)$: The key-validation algorithm takes a common reference string crs, a slot index $i \in [L]$ and a public key pk_i as inputs and outputs 1 if $\mathsf{pk}_i \in \{0,1\}^{2\lambda}$; otherwise outputs 0.

Aggregate$(\mathsf{crs}, (\mathsf{pk}_1, f_1), \ldots, (\mathsf{pk}_L, f_L))$: The aggregate algorithm takes a common reference string crs, a list of L public key-function pairs $(\mathsf{pk}_1, f_1), \ldots, (\mathsf{pk}_L, f_L)$ as inputs such that $f_i \in \mathcal{U}_F$ for all $i \in [L]$. It computes

$$h \leftarrow \mathsf{SSB.Hash}(\mathsf{hk}, (\mathsf{pk}_1, f_1), \ldots, (\mathsf{pk}_L, f_L))$$

and sets $\mathsf{MPK} \coloneqq (\mathsf{hk}, h)$. For each user $i \in [L]$, the aggregate algorithm computes

$$\pi_i \leftarrow \mathsf{SSB.Open}(\mathsf{hk}, ((\mathsf{pk}_1, f_1), \ldots, (\mathsf{pk}_L, f_L)), i)$$

where we treat each pair $(\mathsf{pk}_i, f_i) \in \{0,1\}^{\ell_{\mathsf{blk}}}$ as one SSB hash-block. It sets the helper decryption key as $\mathsf{hsk}_i \coloneqq (i, \mathsf{pk}_i, f_i, \pi_i)$ and outputs $\mathsf{MPK}, \mathsf{hsk}_1, \ldots, \mathsf{hsk}_L$.

Enc(MPK, m) : The encryption algorithm takes MPK, and a message $m \in \mathcal{M}$ as inputs and samples $\mathsf{SK}_0, \mathsf{SK}_1 \leftarrow \mathsf{SKE.Setup}(1^\lambda)$, computes

$$\mathsf{CT}_0 \leftarrow \mathsf{SKE.Enc}(\mathsf{SK}_0, m) \text{ and } \mathsf{CT}_1 \leftarrow \mathsf{SKE.Enc}(\mathsf{SK}_1, \mathbf{0}_{|m|}).$$

It writes $(\mathsf{CT}_0, \mathsf{CT}_1) = (\beta_1, \ldots, \beta_{\ell_c}, \beta_{\ell_c+1}, \ldots, \beta_{2\ell_c}) \in \{0,1\}^{2\ell_c}$. The algorithm samples $u_{k,\beta} \leftarrow \{0,1\}^\lambda$ for all $k \in [2\ell_c], \beta \in \{0,1\}$. It computes $V = (v_{k,\beta} \coloneqq \mathsf{PRG}(u_{k,\beta}))_{k\in[2\ell_c],\beta\in\{0,1\}}$. It constructs the circuit $C_0 = C[\mathsf{MPK}, \mathsf{SK}_0, V]$ as defined in Figure 1 and computes $\widetilde{C}_0 \leftarrow i\mathcal{O}(1^\lambda, C_0)$. It outputs the ciphertext $\mathsf{ct} \coloneqq (\mathsf{CT}_0, \mathsf{CT}_1, \widetilde{C}_0, \sigma_{\mathsf{CT}} \coloneqq (u_{k,\beta_k})_{k\in[2\ell_c]})$.

Dec$(\mathsf{sk}_i, \mathsf{hsk}_i, \mathsf{ct})$: The decryption algorithm takes a secret key sk_i, a helper decryption key $\mathsf{hsk}_i = (i, \mathsf{pk}_i, f_i, \pi_i)$ and ciphertext $\mathsf{ct} = (\mathsf{CT}_0, \mathsf{CT}_1, \widetilde{C}_0, \sigma_{\mathsf{CT}})$ as inputs and outputs $\widetilde{C}_0(\mathsf{sk}_i, i, \mathsf{pk}_i, f_i, \pi_i, \mathsf{CT}_0, \mathsf{CT}_1, \sigma_{\mathsf{CT}})$.

Constants: $\mathsf{MPK} = (\mathsf{hk}, h), \mathsf{SK}_j, V = (v_{k,\beta})_{k\in[2\ell_c],\beta\in\{0,1\}}$
Inputs: $\mathsf{sk}_i \in \{0,1\}^\lambda, i \in [L], \mathsf{pk}_i \in \{0,1\}^{2\lambda}, f_i \in \{0,1\}^{\ell_f}, \pi_i \in \{0,1\}^{\ell_{\mathsf{open}}}$, SKE ciphertexts $\{\mathsf{CT}_j\}_{j\in\{0,1\}}$ and $\sigma_{\mathsf{CT}} = (u_k)_{k\in[2\ell_c]}$

1. Parse $(\mathsf{CT}_0, \mathsf{CT}_1) = (\beta_1, \ldots, \beta_{\ell_c}, \beta_{\ell_c+1}, \ldots, \beta_{2\ell_c}) \in \{0,1\}^{2\ell_c}$.
2. If $\mathsf{SSB.Vrfy}(\mathsf{hk}, h, i, (\mathsf{pk}_i, f_i), \pi_i) = 1 \wedge \mathsf{PRG}(\mathsf{sk}_i) = \mathsf{pk}_i \wedge (\mathsf{PRG}(u_k) = v_{k,\beta_k})_{k\in[2\ell_c]}$
   a. Compute $\widehat{m} \leftarrow \mathsf{SKE.Dec}(\mathsf{SK}_j, \mathsf{CT}_j)$
   b. Output $f_i(\widehat{m})$
3. Else, output $\perp$

Figure 1: The circuit $C_j = C_j[\mathsf{MPK}, \mathsf{SK}_j, V]$ for $j \in \{0,1\}$

Completeness: The scheme satisfies completeness since the $\mathsf{IsValid}$ algorithm outputs 1 only if $\mathsf{pk}_i \in \{0,1\}^{2\lambda}$ and the KeyGen algorithm computes $\mathsf{pk}_i = \mathsf{PRG}(s_i)$ which belongs to $\{0,1\}^{2\lambda}$.

Correctness: Consider a secret key $\mathsf{sk}_i = s_i$, a helper decryption key $\mathsf{hsk}_i = (i, \mathsf{pk}_i, f_i, \pi_i)$, and a ciphertext $\mathsf{ct} = (\mathsf{CT}_0, \mathsf{CT}_1, \widetilde{C}_0, \sigma_{\mathsf{CT}})$ generated as above. By definition, $\mathsf{pk}_i = \mathsf{PRG}(\mathsf{sk}_i)$ and $\mathsf{MPK} = (\mathsf{hk}, h)$ where

$$h \leftarrow \mathsf{SSB.Hash}(\mathsf{hk}, (\mathsf{pk}_1, f_1), \ldots, (\mathsf{pk}_L, f_L))$$
$$\pi_i \leftarrow \mathsf{SSB.Open}(\mathsf{hk}, ((\mathsf{pk}_1, f_1), \ldots, (\mathsf{pk}_L, f_L)), i).$$

Therefore, the check of the circuit C_0 passes, *i.e.*, $\mathsf{SSB.Vrfy}(\mathsf{hk}, h, i, (\mathsf{pk}_i, f_i), \pi_i) = 1$ and $\mathsf{PRG}(\mathsf{sk}_i) = \mathsf{pk}_i$ by the correctness of SSB and PRG. Also, by definition $\mathsf{PRG}(u_k) = v_{k,\beta_k}$ holds for all $k \in [2\ell_c]$ where $\sigma_{\mathsf{CT}} = (u_k)_{k\in[2\ell_c]}$. Then, the SKE decryption $\mathsf{SKE.Dec}(\mathsf{SK}_0, \mathsf{CT}_0)$ returns m, and hence the circuit C_0 on input $(\mathsf{sk}_i, \mathsf{hsk}_i, \mathsf{CT}_0, \mathsf{CT}_1, \sigma_{\mathsf{CT}})$ returns $f_i(m)$. Therefore, by the correctness of $i\mathcal{O}$, we get $\widetilde{C}_0(\mathsf{sk}_i, \mathsf{hsk}_i, \mathsf{CT}_0, \mathsf{CT}_1, \sigma_{\mathsf{CT}}) = f_i(m)$.

Compactness: Consider the master public key $\mathsf{MPK} = (\mathsf{hk}, h)$ and the helper decryption key $\mathsf{hsk}_i = (i, \mathsf{pk}_i, f_i, \pi_i)$ output by the Aggregate algorithm. Since SSB.Setup is an efficient algorithm we have $|\mathsf{hk}| = \mathsf{poly}(\lambda, \ell_{\mathsf{blk}}, \log L)$ and due to the succinctness of SSB.Hash we have $|h|, |\pi_i| = \mathsf{poly}(\lambda, \ell_{\mathsf{blk}})$. The maximum length of any function in the universe $\mathcal{U}_F$ is ℓ_f and $\ell_{\mathsf{blk}} = \ell_f + 2\lambda = \log(|\mathcal{U}_F|) + 2\lambda$. Therefore, it must hold that $|\mathsf{MPK}|, |\mathsf{hsk}_i|$ are bounded by $\mathsf{poly}(\lambda, \log(|\mathcal{U}_F|), \log L)$.

8.3 Security Analysis

We prove the following theorem to show that the SlotRFE is secure.

Theorem 5 *Assuming that the* PRG *is secure,* SKE *is IND-CPA secure,* SSB *is correct and secure, and $i\mathcal{O}$ is secure then our* SlotRFE *is secure.*

Proof. We prove the theorem using a sequence of hybrid experiments. We start with a real experiment which is $\mathsf{Expt}^{\mathsf{SlotRFE}}_{\mathcal{A}}(1^\lambda, 0)$ and end up in $\mathsf{Expt}^{\mathsf{SlotRFE}}_{\mathcal{A}}(1^\lambda, 1)$. The computational indistinguishability between the consecutive hybrids will be argued based on the assumptions stated in the theorem.

Game 0 : This is the real experiment with $\mathsf{coin} = 0$. More precisely, it works as follows:

- **Setup phase:** The adversary sends a slot count 1^L to the challenger. The challenger samples $\mathsf{hk} \leftarrow \mathsf{SSB.Setup}(1^\lambda, 1^{\ell_{\mathsf{blk}}}, L, 1)$ and sends $\mathsf{crs} := \mathsf{hk}$ to $\mathcal{A}$. The challenger initializes a counter $\mathsf{ctr} \leftarrow 0$, a dictionary D and a set of slot indices Cor.
- **Pre-challenge query phase:** The adversary $\mathcal{A}$ is allowed to query the following queries:
  - **Key-generation query:** In a key-generation query, $\mathcal{A}$ specifies a slot index $i \in [L]$. The challenger samples $s \leftarrow \{0,1\}^\lambda$ and increments $\mathsf{ctr} \leftarrow \mathsf{ctr}+1$. Then, it sends $(\mathsf{ctr}, \mathsf{pk}_{\mathsf{ctr}} := \mathsf{PRG}s)$ to $\mathcal{A}$. The challenger adds the mapping $\mathsf{ctr} \mapsto (i, \mathsf{pk}_{\mathsf{ctr}}, \mathsf{sk}_{\mathsf{ctr}} := s)$ to D.
  - **Corruption query:** In a corruption query, $\mathcal{A}$ specifies an index $1 \le c \le \mathsf{ctr}$. The challenger looks up the tuple $(i, \mathsf{pk}, s) \leftarrow \mathsf{D}[c]$ and sends s to $\mathcal{A}$.
- **Challenge phase:** For each slot $i \in [L]$, $\mathcal{A}$ specifies a tuple $(c_i, f_i, \mathsf{pk}^*_i)$, and two challenge messages m^*_0, m^*_1. The challenger does the following:
  - If $c_i \in \{1, \ldots, \mathsf{ctr}\}$, then the challenger looks up the entry $\mathsf{D}[c_i] = (i', \mathsf{pk}', \mathsf{sk}')$. If $i = i'$, then the challenger sets $\mathsf{pk}_i \leftarrow \mathsf{pk}'$. Moreover, if $\mathcal{A}$ previously issues a *corruption query* on the index c_i, then the challenger adds the slot index i to Cor. Otherwise, if $i \neq i'$, then the experiment halts.
  - If $c_i = \perp$, then the challenger checks $\mathsf{pk}^*_i \in \{0,1\}^{2\lambda}$. If not, the experiment halts. Otherwise, the challenger sets $\mathsf{pk}_i \leftarrow \mathsf{pk}^*_i$ and adds the slot index i to Cor.

  The challenger computes $h \leftarrow \mathsf{SSB.Hash}(\mathsf{hk}, (\mathsf{pk}_1, f_1), \ldots, (\mathsf{pk}_L, f_L))$ and samples $\mathsf{SK}_0, \mathsf{SK}_1 \leftarrow \mathsf{SKE.Setup}(1^\lambda)$. Then, it computes $\mathsf{CT}_0 \leftarrow \mathsf{SKE.Enc}(\mathsf{SK}_0, m^*_0)$, $\mathsf{CT}_1 \leftarrow \mathsf{SKE.Enc}(\mathsf{SK}_1, \mathbf{0}_{|m^*_0|})$ and $V = (v^*_{k,\beta} := \mathsf{PRG}(u^*_{k,\beta}))_{k\in[2\ell_c],\beta\in\{0,1\}}$ where $u^*_{k,\beta} \leftarrow \{0,1\}^\lambda$. Then, it computes $\widetilde{C}_0 \leftarrow i\mathcal{O}(1^\lambda, C_0)$ and sets $\sigma^*_{\mathsf{CT}} = (u^*_{k,\beta_k})_{k\in[2\ell_c]}$ where $C_0 = C[\mathsf{MPK}, \mathsf{SK}_0, V^*]$ (as defined in Figure 1) and β_k represents the k-th bit of $(\mathsf{CT}_0, \mathsf{CT}_1)$. Finally, it sends $\mathsf{ct}^* := (\mathsf{CT}_0, \mathsf{CT}_1, \widetilde{C}_0, \sigma^*_{\mathsf{CT}})$ to $\mathcal{A}$.
- **Output phase:** At the end of the experiment, $\mathcal{A}$ outputs a bit $\mathsf{coin}' \in \{0,1\}$, which is the output of the experiment.

Constants: $\mathsf{MPK} = (\mathsf{hk}, h), \mathsf{SK}_0, \mathsf{SK}_1, V^* = (v^*_{k,\beta})_{k\in[2\ell_c],\beta\in\{0,1\}}, j \in [0, L]$
Inputs: $\mathsf{sk}_i \in \{0,1\}^\lambda, i \in [L], \mathsf{pk}_i \in \{0,1\}^{2\lambda}, f_i \in \{0,1\}^{\ell_f}, \pi_i \in \{0,1\}^{\ell_{\mathsf{open}}}$, SKE ciphertexts $\{\mathsf{CT}_j\}_{j\in\{0,1\}}$ and $\sigma_{\mathsf{CT}} = (u_k)_{k\in[2\ell_c]}$

1. Parse $(\mathsf{CT}_0, \mathsf{CT}_1) = (\beta_1, \ldots, \beta_{\ell_c}, \beta_{\ell_c+1}, \ldots, \beta_{2\ell_c}) \in \{0,1\}^{2\ell_c}$.
2. If $\mathsf{SSB.Vrfy}(\mathsf{hk}, h, i, (\mathsf{pk}_i, P_i), \pi_i) = 1 \wedge \mathsf{PRG}(\mathsf{sk}_i) = \mathsf{pk}_i \wedge (\mathsf{PRG}(u_k) = v^*_{k,\beta_k})_{k\in[2\ell_c]}$
3. Compute $\widehat{m} \leftarrow \begin{cases} \mathsf{SKE.Dec}(\mathsf{SK}_0, \mathsf{CT}_0) \text{ if } i > j \\ \mathsf{SKE.Dec}(\mathsf{SK}_1, \mathsf{CT}_1) \text{ if } i \le j \end{cases}$
4. Output $F(P_i, \widehat{m})$
5. Otherwise, output $\perp$

Figure 2: The circuit $C^{\mathsf{slot}}_j = C^{\mathsf{slot}}_j[\mathsf{MPK}, \mathsf{SK}_0, \mathsf{SK}_1, V^*, j]$ for $j \in [0, L]$

Game 1 : It is the same as Game 0 except the challenger sets $\mathsf{CT}_1 \leftarrow \mathsf{SKE.Enc}(\mathsf{SK}_1, m^*_1)$ and computes $\mathsf{CT}_0, \widetilde{C}_0 \leftarrow i\mathcal{O}(1^\lambda, C_0), \sigma^*_{\mathsf{CT}}$ as before.

Game 2 : It is the same as Game 1 except the computation of $V^* = (v^*_{k,\beta})_{k\in[2\ell_c],\beta\in\{0,1\}}$. Let $\mathsf{ct}^* = (\mathsf{CT}_0, \mathsf{CT}_1, \widetilde{C}_0, \sigma^*_{\mathsf{CT}})$ be the challenge ciphertext where $\mathsf{CT}_0 \leftarrow \mathsf{SKE.Enc}(\mathsf{SK}_0, m^*_0)$, $\mathsf{CT}_1 \leftarrow \mathsf{SKE.Enc}(\mathsf{SK}_1, m^*_1)$ and $(\mathsf{CT}_0, \mathsf{CT}_1) = (\beta_1, \ldots, \beta_{\ell_c}, \beta_{\ell_c+1}, \ldots, \beta_{2\ell_c}) \in \{0,1\}^{2\ell_c}$. Then, the challenger computes $v^*_{k,\beta}$ as follows:

$$v^*_{k,\beta} \leftarrow \begin{cases} \mathsf{PRG}(u^*_{k,\beta_k}) \text{ for } u^*_{k,\beta_k} \leftarrow \{0,1\}^\lambda, & \text{if } \beta = \beta_k \\ \{0,1\}^{2\lambda}, & \text{if } \beta = 1 - \beta_k \end{cases}$$

for all $k \in [2\ell_c]$. Note that, the challenger defines $C_0 \coloneqq C_0[\mathsf{MPK}, \mathsf{SK}_0, V^*]$ and sets $\sigma^*_{\mathsf{CT}} \coloneqq (u^*_{k,\beta})_{k\in[2\ell_c]}$ as in the previous hybrid.

Game 3 : It is the same as Game 2 except the challenger computes $\widetilde{C}^{\mathsf{slot}}_0 \leftarrow i\mathcal{O}(1^\lambda, C^{\mathsf{slot}}_0)$ instead of $\widetilde{C}_0$ where the circuit $C^{\mathsf{slot}}_0 = C^{\mathsf{slot}}_0[\mathsf{MPK}, \mathsf{SK}_0, \mathsf{SK}_1, V^*, 0]$ is defined in Figure 2. The other components of the challenge ciphertext, *i.e.*, $\mathsf{CT}_0, \mathsf{CT}_1$ remain the same as in the previous hybrid.

Game $3 + j$ $(j \in [L])$: It is the same as Game $2+(j-1)$ except the challenger computes $\widetilde{C}^{\mathsf{slot}}_j \leftarrow i\mathcal{O}(1^\lambda, C^{\mathsf{slot}}_j)$ instead of $\widetilde{C}^{\mathsf{slot}}_{j-1}$ where the circuit $C^{\mathsf{slot}}_j = C^{\mathsf{slot}}_j[\mathsf{MPK}, \mathsf{SK}_0, \mathsf{SK}_1, F, j]$ is defined in Figure 2.

Game $4 + L$: It is the same as Game $3 + L$ except the challenger computes $\widetilde{C}_1 \leftarrow i\mathcal{O}(1^\lambda, C_1)$ instead of $\widetilde{C}^{\mathsf{slot}}_L$ where the circuit $C_1 = C_1[\mathsf{MPK}, \mathsf{SK}_1, V^*]$ is defined in Figure 1. That is, the challenge ciphertext becomes $\mathsf{ct}^* = (\mathsf{CT}_0, \mathsf{CT}_1, \widetilde{C}_1, \sigma^*_{\mathsf{CT}})$.

Game $5 + L$: It is the same as Game $4 + L$ except the challenger sets $\mathsf{CT}_0 \leftarrow \mathsf{SKE.Enc}(\mathsf{SK}_0, m^*_1)$ and computes $\mathsf{CT}_1, \widetilde{C}_1 \leftarrow i\mathcal{O}(1^\lambda, C_0), \sigma^*_{\mathsf{CT}}$ as before.

Game $6 + L$: It is the same as Game $5 + L$ except the challenger computes $\widetilde{C}_0 \leftarrow i\mathcal{O}(1^\lambda, C_0)$ instead of $\widetilde{C}_1$ where the circuit $C_0 = C_0[\mathsf{MPK}, \mathsf{SK}_0, V^*]$ is defined in Figure 1.

Game $7 + L$: It is the same as Game $6 + L$ except the computation of $V^* = (v^*_{k,\beta})_{k\in[2\ell_c],\beta\in\{0,1\}}$. Let $\mathsf{ct}^* = (\mathsf{CT}_0, \mathsf{CT}_1, \widetilde{C}_0, \sigma^*_{\mathsf{CT}})$ be the challenge ciphertext where $\mathsf{CT}_0 \leftarrow \mathsf{SKE.Enc}(\mathsf{SK}_0, m^*_1)$, $\mathsf{CT}_1 \leftarrow$

$\mathsf{SKE.Enc}(\mathsf{SK}_1, m_1^*)$ and $(\mathsf{CT}_0, \mathsf{CT}_1) = (\beta_1, \ldots, \beta_{\ell_c}, \beta_{\ell_c+1}, \ldots, \beta_{2\ell_c}) \in \{0,1\}^{2\ell_c}$. Then, the challenger computes $v^*_{k,\beta}$ as follows:

$$v^*_{k,\beta} \leftarrow \begin{cases} \mathsf{PRG}(u^*_{k,\beta_k}) & \text{for } u^*_{k,\beta_k} \leftarrow \{0,1\}^\lambda, \text{ if } \beta = \beta_k \\ \mathsf{PRG}(u^*_{k,1-\beta_k}) & \text{for } u^*_{k,1-\beta_k} \leftarrow \{0,1\}^\lambda, \text{ if } \beta = 1-\beta_k \end{cases}$$

for all $k \in [2\ell_c]$. Note that, the challenger defines $C_0 \coloneqq C_0[\mathsf{MPK}, \mathsf{SK}_0, V^*]$ and sets $\sigma^*_{\mathsf{CT}} \coloneqq (u^*_{k,\beta})_{k\in[2\ell_c]}$ as in the previous hybrid.

Game $8 + L$: It is the same as Game $7+L$ except the challenger sets $\mathsf{CT}_1 \leftarrow \mathsf{SKE.Enc}(\mathsf{SK}_1, \mathbf{0}_{|m_1^*|})$. Observe that this hybrid is the same as $\mathsf{Expt}^{\mathsf{SlotRFE}}_{\mathcal{A}}(1^\lambda, 1)$.

Let E_i be the event of $\mathcal{A}$ outputting the correct bit coin in Game i. We show that the each pair of consecutive games are indistinguishable from $\mathcal{A}$'s view in the following lemmas.

Lemma 18 *If* SKE *is IND-CPA secure then for all efficient and admissible adversaries* $\mathcal{A}$*, for all* $\lambda \in \mathbb{N}$ *there exists a negligible function* negl *such that*

$$|\Pr[\mathsf{E}_0] - \Pr[\mathsf{E}_1]| = \mathsf{negl}(\lambda).$$

Proof. The only difference between games 0 and 1 is that the challenge ciphertext component CT_1 is an SKE encryption of m_1^* instead of $\mathbf{0}_{|m_0^*|}$. We show that if $\mathcal{A}$ distinguishes between the hybrids with a non-negligible advantage $\epsilon(\lambda)$ then there exists an adversary $\mathcal{B}$ who breaks the IND-CPA security of SKE with at least an advantage of $\epsilon(\lambda)$. The adversary $\mathcal{B}$ works as follows:

1. $\mathcal{B}$ receives the slot count L from $\mathcal{A}$ and then plays the role of the challenger as in the hybrid 0 for the *setup* and *pre-challenge query* phase.
2. When $\mathcal{B}$ receives the challenge query $(\{(c_i, f_i, \mathsf{pk}_i^*)\}_{i\in[L]}, m_0^*, m_1^*)$ from $\mathcal{A}$, it works exactly the same as the challenger in hybrid 0 except it uses the SKE-challenger to compute CT_1. In particular, $\mathcal{B}$ sends the challenge message pair $(\mathbf{0}_{|m_0^*|}, m_1^*)$ to the SKE-challenger and gets back a ciphertext CT_1^*. Finally, $\mathcal{B}$ sends the challenge ciphertext $\mathsf{ct}^* = (\mathsf{CT}_0, \mathsf{CT}_1 \coloneqq \mathsf{CT}_1^*, \widetilde{C}_0, \sigma^*_{\mathsf{CT}})$ to $\mathcal{A}$. Note that, $\mathcal{B}$ does not require the secret key SK_1 to compute the components CT_0, $\widetilde{C}_0, \sigma^*_{\mathsf{CT}}$ of ct^*.
3. At the end of the experiment, $\mathcal{A}$ outputs a guess $\mathsf{coin}' \in \{0,1\}$ which is also the output of $\mathcal{B}$.

If the SKE-challenger computes $\mathsf{CT}_1^* \leftarrow \mathsf{SKE.Enc}(\mathsf{SK}_1, \mathbf{0}_{|m_0^*|})$ then $\mathcal{B}$ perfectly simulates hybrid 0. On the other hand, if the SKE-challenger computes $\mathsf{CT}_1^* \leftarrow \mathsf{SKE.Enc}(\mathsf{SK}_1, m_1^*)$ then $\mathcal{B}$ perfectly simulates hybrid 1. Therefore, $\mathcal{B}$ breaks the IND-CPA security of SKE with advantage at least $\epsilon(\lambda)$ if $\mathcal{A}$ distinguishes between the hybrids advantage $\epsilon(\lambda)$. Hence, the lemma follows. □

Lemma 19 *If* PRG *is secure then for all efficient and admissible adversaries* $\mathcal{A}$*, for all* $\lambda \in \mathbb{N}$*, and* $j \in [L]$ *there exists a negligible function* negl *such that*

$$|\Pr[\mathsf{E}_1] - \Pr[\mathsf{E}_2]| = \mathsf{negl}(\lambda).$$

Proof. We prove the lemma using a sequence of $2\ell_c$ games $\mathsf{G}_{1,k}$ for $k \in [2\ell_c + 1]$ where we sample $v^*_{t,1-\beta_t} \leftarrow \{0,1\}^{2\lambda}$ for all $t < k$ in $\mathsf{G}_{1,k}$ and β_t represents the t-th bit of $(\mathsf{CT}_0, \mathsf{CT}_1)$. Note that, $\mathsf{G}_{1,1}$ is identical to Game 1 and $\mathsf{G}_{1,2\ell_c+1}$ is identical to Game 2. We only show that the distinguishing advantage of $\mathcal{A}$ between the games $\mathsf{G}_{1,k}$ and $\mathsf{G}_{1,k+1}$ is negligible for each $k \in [2\ell_c]$. In particular, we show that if $\mathcal{A}$ distinguishes between the hybrids with a non-negligible advantage $\epsilon(\lambda)$ then there exists an adversary $\mathcal{B}$ that breaks the security of PRG with at least an advantage of $\epsilon(\lambda)$. The adversary $\mathcal{B}$ works as follows:

1. $\mathcal{B}$ receives a string $v^* \in \{0,1\}^{2\lambda}$ from the PRG-challenger.

2. $\mathcal{B}$ plays the role of the challenger as in the game $\mathsf{G}_{1,k}$ or $\mathsf{G}_{1,k+1}$. It receives the slot count L from $\mathcal{A}$ and runs the *setup* phase and sends crs to $\mathcal{A}$.

3. After that, $\mathcal{B}$ simulates the *pre-challenge query* phases as in $\mathsf{G}_{1,k}$ or $\mathsf{G}_{1,k+1}$.

4. At the *challenge query* phase, $\mathcal{B}$ computes $\mathsf{CT}_0 \leftarrow \mathsf{SKE.Enc}(\mathsf{SK}_0, m_0^*)$, $\mathsf{CT}_1 \leftarrow \mathsf{SKE.Enc}(\mathsf{SK}_1, m_1^*)$ and writes $(\mathsf{CT}_0, \mathsf{CT}_1) = (\beta_1, \ldots, \beta_{\ell_c}, \beta_{\ell_c+1}, \ldots, \beta_{2\ell_c}) \in \{0,1\}^{2\ell_c}$. Then, it computes $v^*_{t,\beta}$ as follows:

$$v^*_{t,\beta} \leftarrow \begin{cases} \mathsf{PRG}(u^*_{t,\beta_t}) \text{ for } u^*_{t,\beta_t} \leftarrow \{0,1\}^\lambda, & \text{if } \beta = \beta_t \\ \{0,1\}^{2\lambda}, & \text{if } \beta = 1-\beta_t \text{ and } t < k \\ v^* & \text{if } \beta = 1-\beta_k \\ \mathsf{PRG}(u^*_{t,1-\beta_t}) \text{ for } u^*_{t,1-\beta_t} \leftarrow \{0,1\}^\lambda, & \text{if } \beta = 1-\beta_t \text{ and } t > k \end{cases}$$

   for all $t \in [2\ell_c]$. Finally, it sets $V^* = (v^*_{t,\beta})_{t\in[2\ell_c],\beta\in\{0,1\}}$, $\sigma^*_{\mathsf{CT}} = (u^*_{t,\beta_t})_{t\in[2\ell_c]}$, $\widetilde{C}_0 \leftarrow i\mathcal{O}(1^\lambda, C_0)$ where $C_0 = C_0[\mathsf{MPK}, \mathsf{sk}_j, V]$ and sends $\mathsf{ct}^* = (\mathsf{CT}_0, \mathsf{CT}_1, \widetilde{C}_0, \sigma^*_{\mathsf{CT}})$ to $\mathcal{A}$ as in the previous hybrid.

5. At the end of the experiment, $\mathcal{A}$ outputs a guess $\mathsf{coin}' \in \{0,1\}$ which is also the output of $\mathcal{B}$.

If the PRG-challenger computes $v^* \leftarrow \mathsf{PRG}(u_{k,1-\beta_k})$ for some $u_{k,1-\beta_k} \leftarrow \{0,1\}^\lambda$ then $\mathcal{B}$ perfectly simulates $\mathsf{G}_{1,k}$. On the other hand, if the PRG-challenger samples $v^* \leftarrow \{0,1\}^{2\lambda}$ then $\mathcal{B}$ perfectly simulates $\mathsf{G}_{1,k+1}$. Therefore, $\mathcal{B}$ breaks the security of SSB with advantage at least $\epsilon(\lambda)$ if $\mathcal{A}$ distinguishes between the games with advantage $\epsilon(\lambda)$. Hence, the lemma follows. □

Lemma 20 *If $i\mathcal{O}$ is secure then for all efficient and admissible adversaries $\mathcal{A}$, for all $\lambda \in \mathbb{N}$ there exists a negligible function negl such that*

$$|\Pr[\mathsf{E}_2] - \Pr[\mathsf{E}_3]| = \mathsf{negl}(\lambda).$$

Proof. The only difference between the games 2 and 3 is that the ciphertext component $\widetilde{C}_0$ is replaced by $\widetilde{C}_0^{\mathsf{slot}}$. Since $i\mathcal{O}$ is secure it is sufficient to show that the two circuits C_0 and C_0^{slot} are equivalent. Let $(\mathsf{sk}_i, i, \mathsf{pk}_i, f_i, \pi_i, \mathsf{CT}_0, \mathsf{CT}_1, \sigma_{\mathsf{CT}})$ be an arbitrary input to the circuits. The programming of the circuits differ only in *step* 2 where $\widehat{m}$ is computed via SKE decryption algorithm. The circuit C_0 always decrypts CT_0 using SK_0 whereas the circuit C_0^{slot} decrypts CT_0 using SK_0 if $i > 0$, otherwise it CT_1 using SK_1. Since $i \in [L]$ and $i > 0$ holds for all possible inputs $(\mathsf{sk}_i, i, \mathsf{pk}_i, f_i, \pi_i, \mathsf{CT}_0, \mathsf{CT}_1, \sigma_{\mathsf{CT}})$, the circuit C_0^{slot} always decrypts CT_0 using SK_0 in step 2. Thus, the circuits C_0 and C_0^{slot} are equivalent. By the security of $i\mathcal{O}$, the distinguishing advantage of $\mathcal{A}$ is negligible in λ. □

Lemma 21 *If the PRG is secure, SSB is correct and secure, and $i\mathcal{O}$ is secure then for all efficient and admissible adversaries $\mathcal{A}$, for all $\lambda \in \mathbb{N}$ there exists a negligible function negl such that*

$$|\Pr[\mathsf{E}_{3+j}] - \Pr[\mathsf{E}_{3+(j-1)}]| = \mathsf{negl}(\lambda).$$

Proof. We first introduce a new set of intermediate hybrids jG_{3+j} defined as follows:

$\underline{\mathsf{jG}_{3+j}}$**:** It works exactly the same as Game $3+j$ except the challenger samples $\mathsf{hk} \leftarrow \mathsf{SSB.Setup}(1^\lambda, 1^{\ell_{\mathsf{blk}}}, L, j+1)$ in the setup phase. The hash key hk binds with index $(j+1)$ instead of 1.

We now show that the distinguishing advantage of $\mathcal{A}$ between Game $3+j$ and jG_{3+j} is negligible for each $j \in [L]$. Let jE_{3+j} be the event of $\mathcal{A}$ outputting the correct bit coin in jG_{3+j}.

Claim 1 If SSB satisfies index hiding then for all efficient and admissible adversaries $\mathcal{A}$, for all $\lambda \in \mathbb{N}$ there exists a negligible function negl such that

$$|\Pr[\mathsf{E}_{3+j}] - \Pr[\mathsf{jE}_{3+j}]| = \mathsf{negl}(\lambda).$$

Proof. We show that if $\mathcal{A}$ distinguishes between the hybrids with a non-negligible advantage $\epsilon(\lambda)$ then there exists an adversary $\mathcal{B}$ who breaks the index hiding security of SSB with at least an advantage of $\epsilon(\lambda)$. The adversary $\mathcal{B}$ works as follows:

1. $\mathcal{B}$ receives the slot count L from $\mathcal{A}$. Then, it sends L and $(1, j+1)$ to the SSB-challenger.
2. $\mathcal{B}$ receives a hash key hk^* from it's challenger and sets $\mathsf{hk} \coloneqq \mathsf{hk}^*$. Then, it sends $\mathsf{crs} \coloneqq \mathsf{hk}$ to $\mathcal{A}$.
3. After that, $\mathcal{B}$ plays the role of the challenger exactly similar to Game $3+j$ for simulating the *pre-challenge query* and *challenge* phases.
4. At the end of the experiment, $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$ which is also the output of $\mathcal{B}$.

If the SSB-challenger computes $\mathsf{hk}^* \leftarrow \mathsf{SSB.Setup}(1^\lambda, 1^{\ell_{\mathsf{blk}}}, L, 1)$ then $\mathcal{B}$ perfectly simulates Game $3+j$. On the other hand, if the SSB-challenger computes $\mathsf{hk}^* \leftarrow \mathsf{SSB.Setup}(1^\lambda, 1^{\ell_{\mathsf{blk}}}, L, j+1)$ then $\mathcal{B}$ perfectly simulates jG_{3+j}. Therefore, $\mathcal{B}$ breaks the index hiding security of SSB with advantage at least $\epsilon(\lambda)$ if $\mathcal{A}$ distinguishes between the games advantage $\epsilon(\lambda)$. Hence, the lemma follows. □

By Claim 1, proving Lemma 21 is equivalent ot prove the following claim.

Claim 2 If SSB is somewhere statistically binding then for all efficient and admissible adversaries $\mathcal{A}$, for all $\lambda \in \mathbb{N}$ there exists a negligible function negl such that

$$|\Pr[\mathsf{jE}_{3+j}] - \Pr[\mathsf{jE}_{3+(j-1)}]| = \mathsf{negl}(\lambda).$$

Proof. The only difference between jG_{3+j} and $\mathsf{jG}_{3+(j-1)}$ is in the second last component of the challenge ciphertext where it is $\widetilde{C}_j^{\mathsf{slot}}$ in game jG_{3+j}. The circuits C_j^{slot} and C_{j-1}^{slot} behaves differently only for an input of the form $(\mathsf{sk}_j, j, \mathsf{pk}_j, f_j, \pi_j, \mathsf{CT}_0, \mathsf{CT}_1, \sigma_{\mathsf{CT}})$. The analysis of the claim depends on whether the j-th user is corrupted or not. Let $(c_j, f_j, \mathsf{pk}_j^*)$ be the tuple specified by $\mathcal{A}$ during the challenge query phase. We define an event $\mathsf{NonCorrupt}$ as follows:

1. The index c_j satisfies $\{1, \ldots, \mathsf{ctr}\}$ meaning that pk_j was generated by the challenge on the c_j-th key generation query.
2. $\mathcal{A}$ never make a corruption query on index c_j.

We also denote $\overline{\mathsf{NonCorrupt}}$ by the event which is complement of $\mathsf{NonCorrupt}$. By definition, we can write

$$\begin{aligned}\Pr[\mathsf{jE}_{3+(j-1)}] &= \Pr[\mathsf{jE}_{3+(j-1)} \wedge \mathsf{NonCorrupt}] + \Pr[\mathsf{jE}_{3+(j-1)} \wedge \overline{\mathsf{NonCorrupt}}]\\ \Pr[\mathsf{jE}_{3+j}] &= \Pr[\mathsf{jE}_{3+j} \wedge \mathsf{NonCorrupt}] + \Pr[\mathsf{jE}_{3+j} \wedge \overline{\mathsf{NonCorrupt}}]\end{aligned}$$

Thus, it is sufficient to show that

$$|\Pr[\mathsf{jE}_{3+(j-1)} \wedge \mathsf{NonCorrupt}] - \Pr[\mathsf{jE}_{3+j} \wedge \mathsf{NonCorrupt}]| = \mathsf{negl}(\lambda) \tag{33}$$

$$|\Pr[\mathsf{jE}_{3+(j-1)} \wedge \overline{\mathsf{NonCorrupt}}] - \Pr[\mathsf{jE}_{3+j} \wedge \overline{\mathsf{NonCorrupt}}]| = \mathsf{negl}(\lambda) \tag{34}$$

We show that the equations 33 and 34 hold in claims 3 and 4 respectively.

Claim 3 If the PRG is secure, SSB is correct and secure, and $i\mathcal{O}$ is secure then for all efficient and admissible adversaries $\mathcal{A}$, for all $\lambda \in \mathbb{N}$ there exists a negligible function negl such that

$$|\Pr[\mathsf{jE}_{3+(j-1)} \wedge \mathsf{NonCorrupt}] - \Pr[\mathsf{jE}_{3+j} \wedge \mathsf{NonCorrupt}]| = \mathsf{negl}(\lambda).$$

Proof. The main intuition for proving the claim is that the adversary $\mathcal{A}$ does not have sk_j and hence the associated public key pk_j can be chosen uniformly at random depending on the security of PRG. Then, with the help of SSB and $i\mathcal{O}$ we show that it is possible to change the obfuscated circuit from $\widetilde{C}_{j-1}^{\mathsf{slot}}$ to $\widetilde{C}_j^{\mathsf{slot}}$. More precisely, we use the following sequence of hybrids:

$\mathsf{ncG}_{3+(j-1),1}$: It is the same as $\mathsf{jG}_{3+(j-1)}$ except at the beginning of the experiment, the challenger samples $q \leftarrow [Q]$ where $Q = Q(\lambda)$ denotes the total number of key generation queries $\mathcal{A}$ makes during the query phase. Let pk_q be the public key sampled by the challenger on the q-th key query (if there is one). The challenger aborts with output 0 if either of the following events occurs:

- $\mathcal{A}$ sends the tuple $(c_j, f_j, \mathsf{pk}_j^*)$ for registering the j-th user during the challenge query phase where $c_j \neq q$.
- $\mathcal{A}$ makes a corruption query with a index q.

Otherwise, the experiment proceeds exactly similar to $\mathsf{jG}_{3+(j-1)}$.

$\mathsf{ncG}_{3+(j-1),2}$: It is the same as $\mathsf{ncG}_{3+(j-1),1}$ except the challenger samples $\mathsf{pk}_q \leftarrow \{0,1\}^{2\lambda}$ during the q-th key generation query. In this hybrid, the challenger is not required to answer for a corruption query on index q since it immediately aborts with output 0 as soon as it gets such a query.

$\mathsf{ncG}_{3+(j-1),3}$: It is the same as $\mathsf{ncG}_{3+(j-1),2}$ except the challenger obfuscates the circuit C_j^{slot} instead of C_{j-1}^{slot} while computing the challenge ciphertext.

$\mathsf{ncG}_{3+(j-1),4}$: It is the same as $\mathsf{ncG}_{3+(j-1),3}$ except the challenger samples $\mathsf{sk}_q \leftarrow \{0,1\}^{\lambda}$ and computes $\mathsf{pk}_q \leftarrow \mathsf{PRG}(\mathsf{sk}_q)$ during the q-th key generation query.

$\mathsf{ncG}_{3+(j-1),5}$: It is the same as $\mathsf{ncG}_{3+(j-1),4}$ except the challenger ignores the *abort condition* as defined in $\mathsf{ncG}_{3+(j-1),1}$.

As before, we denote ncE_k be the event of $\mathcal{A}$ outputting the correct bit coin in the game $\mathsf{ncG}_{3+(j-1),k}$ for each $k \in [5]$. Next, we show the indistinguishability between any two consecutive games in the following lemmas.

Lemma 22 *For all efficient and admissible adversaries $\mathcal{A}$, for all $\lambda \in \mathbb{N}$, and $j \in [L]$ there exists a negligible function* negl *such that*

$$\Pr[\mathsf{jE}_{3+(j-1)} \wedge \mathsf{NonCorrupt}] = Q \cdot \Pr[\mathsf{ncE}_{3+(j-1),1}].$$

Proof. By definition, the games $\mathsf{jG}_{3+(j-1)}$ and $\mathsf{ncG}_{3+(j-1),1}$ proceeds exactly in the same way except the challenger aborts with output 0 if $c_j = q$ or $\mathcal{A}$ makes a corruption query for the index q. This means that both the experiments output 1 with the same probability if the event $\mathsf{NonCorrupt}$ occurs and $c_j = q$ holds. Thus, we can write

$$\begin{aligned}
&\Pr[\mathsf{ncE}_{3+(j-1),1}] \\
&= \Pr[\mathsf{jE}_{3+(j-1)} \wedge \mathsf{NonCorrupt} \wedge c_j = q] \\
&= \Pr[c_j = q \mid \mathsf{jE}_{3+(j-1)} \wedge \mathsf{NonCorrupt}] \cdot \Pr[\mathsf{jE}_{3+(j-1)} \wedge \mathsf{NonCorrupt}] \\
&= 1/Q \cdot \Pr[\mathsf{jE}_{3+(j-1)} \wedge \mathsf{NonCorrupt}]
\end{aligned}$$

since the probability that $c_j = q$ holds where $q \leftarrow [Q]$ and $c_j \in \{1, \ldots, \mathsf{ctr}\} \subseteq [Q]$ is $1/Q$ given the event $\mathsf{NonCorrupt}$ has occurred. □

Lemma 23 *If* PRG *is secure then for all efficient and admissible adversaries $\mathcal{A}$, for all $\lambda \in \mathbb{N}$, and $j \in [L]$ there exists a negligible function* negl *such that*

$$|\Pr[\mathsf{ncE}_{3+(j-1),1}] - \Pr[\mathsf{ncE}_{3+(j-1),2}]| = \mathsf{negl}(\lambda).$$

Proof. We show that if $\mathcal{A}$ distinguishes between the hybrids with a non-negligible advantage $\epsilon(\lambda)$ then there exists an adversary $\mathcal{B}$ that breaks the security of PRG with at least an advantage of $\epsilon(\lambda)$. The adversary $\mathcal{B}$ works as follows:

1. $\mathcal{B}$ starts by sampling $q \leftarrow [Q]$ and receives a string $\mathsf{pk}^* \in \{0,1\}^{2\lambda}$ from the PRG-challenger.

2. $\mathcal{B}$ plays the role of the challenger as in the experiment $\mathsf{ncG}_{3+(j-1),1}$ or $\mathsf{ncG}_{3+(j-1),2}$. It receives the slot count L from $\mathcal{A}$ and runs the *setup* phase and sends crs to $\mathcal{A}$.

3. After that, $\mathcal{B}$ simulates the *pre-challenge query* phases as in $\mathsf{ncG}_{3+(j-1),1}$ or $\mathsf{ncG}_{3+(j-1),2}$. $\mathcal{B}$ returns pk^* when it receives a key generation query for the index q and adds $[\mathsf{ctr}] \mapsto (q, \mathsf{pk}^*, \perp)$ to D. If $\mathcal{A}$ makes a corruption query for the index q then $\mathcal{B}$ aborts with output 0.

4. At the *challenge query* phase, $\mathcal{B}$ checks if the tuple $(c_j, f_j, \mathsf{pk}^*_j)$ received from $\mathcal{A}$ satisfies $c_j = q$ and then it proceeds with the role of the challenger. If not, $\mathcal{B}$ aborts with output 0 as in $\mathsf{ncG}_{3+(j-1),1}$ or $\mathsf{ncG}_{3+(j-1),2}$.

5. At the end of the experiment, $\mathcal{A}$ outputs a guess $\mathsf{coin}' \in \{0,1\}$ which is also the output of $\mathcal{B}$.

If the PRG-challenger computes $\mathsf{pk}^* \leftarrow \mathsf{PRG}(s)$ for some $s \leftarrow \{0,1\}^\lambda$ then $\mathcal{B}$ perfectly simulates $\mathsf{ncG}_{3+(j-1),1}$. On the other hand, if the PRG-challenger samples $\mathsf{pk}^* \leftarrow \{0,1\}^{2\lambda}$ then $\mathcal{B}$ perfectly simulates $\mathsf{ncG}_{3+(j-1),2}$. Therefore, $\mathcal{B}$ breaks the security of SSB with advantage at least $\epsilon(\lambda)$ if $\mathcal{A}$ distinguishes between the games with advantage $\epsilon(\lambda)$. Hence, the lemma follows. □

Lemma 24 *If* SSB *is somewhere statistically binding and iO is secure then for all efficient and admissible adversaries* $\mathcal{A}$*, for all* $\lambda \in \mathbb{N}$*, and* $j \in [L]$ *there exists a negligible function* negl *such that*

$$|\Pr[\mathsf{ncE}_{3+(j-1),2}] - \Pr[\mathsf{ncE}_{3+(j-1),3}]| = \mathsf{negl}(\lambda).$$

Proof. The only difference between the games is in the circuit which the challenger obfuscated during the challenge query phase: C^{slot}_{j-1} in $\mathsf{ncG}_{3+(j-1),2}$ and C^{slot}_j in $\mathsf{ncG}_{3+(j-1),3}$. We show that with overwhelming probability over the choice of hk and pk_q the circuits C^{slot}_j and C^{slot}_{j-1} are equivalent. Let us consider an arbitrary input $(\mathsf{sk}_x, x, \mathsf{pk}_x, f_x, \pi_x, \mathsf{CT}'_0, \mathsf{CT}'_1, \sigma_{\mathsf{CT}'})$ to the circuits. Note that the programming of the two circuits is different only in *step 2* (see Figure 2) where SKE decryption algorithm is performed. We consider the following cases:

Case 1: If $x \neq j$, then both the circuits either decrypt CT_0 when $x > j$ or CT_1 when $x < j$ in *step 2*. Hence, output of both the circuits is the same.

Case 2: If $x = j$ and $(\mathsf{pk}_x, f_x) \neq (\mathsf{pk}_q, f_q)$, then we use the somewhere statistically binding property of SSB to argue that both the circuits return $\perp$. Note that, the challenger hardwires $\mathsf{MPK} = (\mathsf{hk}, h)$ in both the circuits computed as

$$\begin{aligned}\mathsf{hk} &\leftarrow \mathsf{SSB.Setup}(1^\lambda, 1^{\ell_{\mathsf{blk}}}, L, j)\\ h &\leftarrow \mathsf{SSB.Hash}(\mathsf{hk}, (\mathsf{pk}_1, f_1), \ldots, (\mathsf{pk}_q, f_q), \ldots (\mathsf{pk}_L, f_L))\end{aligned}$$

in $\mathsf{ncG}_{3+(j-1),2}$ or $\mathsf{ncG}_{3+(j-1),3}$. By the somewhere statistically binding property of SSB, with overwhelming probability over the choice of hk (which binds index j), there does not exist any $(\mathsf{pk}^*, f^*) \neq (\mathsf{pk}_q, f_q)$ and π^* such that $\mathsf{SSB.Vrfy}(\mathsf{hk}, j, (\mathsf{pk}^*, f^*), \pi^*) = 1$. Therefore, if $(\mathsf{pk}_x, f_x) \neq (\mathsf{pk}_q, f_q)$ then the circuits C^{slot}_j and C^{slot}_{j-1} output $\perp$ due to *step 1*.

Case 3: If $x = j$ and $(\mathsf{pk}_x, f_x) = (\mathsf{pk}_q, f_q)$, then the we use the fact that pk_q *is uniformly chosen* to argue that the circuits returns the same value. Let us assume the challenger does not abort in both games $\mathsf{ncG}_{3+(j-1),2}$, $\mathsf{ncG}_{3+(j-1),3}$. This means that $\mathsf{pk}_x = \mathsf{pk}_j = \mathsf{pk}_q$ where $\mathsf{pk}_q \leftarrow \{0,1\}^{2\lambda}$ is the q-th public key. Since pk_q is chosen uniformly at random from $\{0,1\}^{2\lambda}$ then the probability that there exists some $s \in \{0,1\}^\lambda$ such that $\mathsf{PRG}(s) = \mathsf{pk}_q$ is at most $1/2^\lambda$ which is negligible in the security parameter. Therefore, with overwhelming probability it holds that $\mathsf{PRG}(\mathsf{sk}_x) \neq \mathsf{pk}_x = \mathsf{pk}_q$. Consequently, the check in *step 2* of both the circuits does not pass and as a result the circuits C^{slot}_j and C^{slot}_{j-1} output $\perp$.

Hence, for all possible inputs, the circuits C_j^{slot} and C_{j-1}^{slot} output the same value with overwhelming probability over the choice of $\mathsf{hk}, \mathsf{pk}_q$. Therefore, by the security of $i\mathcal{O}$, the lemma follows. □

Lemma 25 *If* PRG *is secure then for all efficient and admissible adversaries* $\mathcal{A}$*, for all* $\lambda \in \mathbb{N}$*, and* $j \in [L]$ *there exists a negligible function* negl *such that*

$$|\Pr[\mathsf{ncE}_{3+(j-1),3}] - \Pr[\mathsf{ncE}_{3+(j-1),4}]| = \mathsf{negl}(\lambda).$$

The proof follows similar to that of Lemma 23.

Lemma 26 *For all efficient and admissible adversaries* $\mathcal{A}$*, for all* $\lambda \in \mathbb{N}$*, and* $j \in [L]$ *there exists a negligible function* negl *such that*

$$\Pr[\mathsf{jE}_{3+j} \wedge \mathsf{NonCorrupt}] = Q \cdot \Pr[\mathsf{ncE}_{3+(j-1),1}].$$

The proof follows similar to that of Lemma 22.
Finally, the proof of Claim 3 follows by combining the Lemmas 22 to 26. □

Claim 4 If SSB is correct and secure, and $i\mathcal{O}$ is secure then for all efficient and admissible adversaries $\mathcal{A}$, for all $\lambda \in \mathbb{N}$ there exists a negligible function negl such that

$$|\Pr[\mathsf{jE}_{3+(j-1)} \wedge \overline{\mathsf{NonCorrupt}}] - \Pr[\mathsf{jE}_{3+j} \wedge \overline{\mathsf{NonCorrupt}}]| = \mathsf{negl}(\lambda).$$

Proof. We prove the claim using the following two hybrid experiments:

$\underline{\mathsf{cG}_{3+(j-1),1}}$**:** It is the same as $\mathsf{jG}_{3+(j-1)}$ except the challenger aborts with output 0 if the event NonCorrupt occurs. This means that the output of the experiment can be 1 only if the public key pk_j is either adversarially generated or $\mathcal{A}$ makes a corruption query for the index j. Since $\mathcal{A}$ is admissible, in either cases, it must hold that $f_j(m_0^*) = f_j(m_1^*)$ where f_j is the associated function with pk_j.

$\underline{\mathsf{cG}_{3+(j-1),2}}$**:** It is the same as $\mathsf{cG}_{3+(j-1),1}$ except that the challenger obfuscates the circuit C_j^{slot} instead of C_{j-1}^{slot} while computing the challenge ciphertext.

As before, we denote $\mathsf{cE}_{3+(j-1),k}$ by the event of $\mathcal{A}$ outputting the correct bit coin in the game $\mathsf{ncG}_{3+(j-1),k}$ for each $k \in \{1, 2\}$. By definition, we have that

$$\Pr[\mathsf{jE}_{3+(j-1)} \wedge \overline{\mathsf{NonCorrupt}}] = \Pr[\mathsf{cE}_{3+(j-1),1}]$$
$$\Pr[\mathsf{jE}_{3+j} \wedge \overline{\mathsf{NonCorrupt}}] = \Pr[\mathsf{cE}_{3+(j-1),2}]$$

Therefore, it is sufficient to prove that

$$|\Pr[\mathsf{cE}_{3+(j-1),1}] - \Pr[\mathsf{cE}_{3+(j-1),2}]| = \mathsf{negl}(\lambda).$$

We prove the indistinguishability between the hybrids in the following lemma.

Lemma 27 *If* SSB *is somewhere statistically binding and* $i\mathcal{O}$ *is secure then for all efficient and admissible adversaries* $\mathcal{A}$*, for all* $\lambda \in \mathbb{N}$*, and* $j \in [L]$ *there exists a negligible function* negl *such that*

$$|\Pr[\mathsf{cE}_{3+(j-1),1}] - \Pr[\mathsf{cE}_{3+(j-1),2}]| = \mathsf{negl}(\lambda).$$

Proof. The only difference between the games is in the circuit which the challenger obfuscated during the challenge query phase: C_{j-1}^{slot} in $\mathsf{cG}_{3+(j-1),1}$ and C_j^{slot} in $\mathsf{cG}_{3+(j-1),2}$. We show that with overwhelming probability over the choice of hk the circuits C_j^{slot} and C_{j-1}^{slot} are equivalent. Let us consider an arbitrary input $(\mathsf{sk}_x, x, \mathsf{pk}_x, f_x, \pi_x, \mathsf{CT}'_0, \mathsf{CT}'_1, \sigma_{\mathsf{CT}'})$ to the circuits. Note that the programming of the two circuits is different only in *step 2* (see Figure 2) where SKE decryption algorithm is performed. We consider the following cases:

Case 1: If $x \neq j$, then both the circuits either decrypt CT_0 when $x > j$ or CT_1 when $x < j$ in *step 2*. Hence, output of both the circuits is the same.

Case 2: If $x = j$ and $(\mathsf{pk}_x, f_x) \neq (\mathsf{pk}_j, f_j)$, then we use the somewhere statistically binding property of SSB to argue that both the circuits return $\perp$. Note that, the challenger hardwires $\mathsf{MPK} = (\mathsf{hk}, h)$ in both the circuits computed as

$$\mathsf{hk} \leftarrow \mathsf{SSB.Setup}(1^\lambda, 1^{\ell_{\mathsf{blk}}}, L, j)$$
$$h \leftarrow \mathsf{SSB.Hash}(\mathsf{hk}, (\mathsf{pk}_1, f_1), \ldots, (\mathsf{pk}_j, f_j), \ldots (\mathsf{pk}_L, f_L))$$

in $\mathsf{cG}_{3+(j-1),1}$ or $\mathsf{cG}_{3+(j-1),2}$. By the somewhere statistically binding property of SSB, with overwhelming probability over the choice of hk (which binds index j), there does not exist any $(\mathsf{pk}^*, f^*) \neq (\mathsf{pk}_j, f_j)$ and π^* such that $\mathsf{SSB.Vrfy}(\mathsf{hk}, j, (\mathsf{pk}^*, f^*), \pi^*) = 1$. Therefore, if $(\mathsf{pk}_x, f_x) \neq (\mathsf{pk}_j, f_j)$ then the circuits C_j^{slot} and C_{j-1}^{slot} output $\perp$ due to *step 1*.

Case 3: If $x = j$ and $(\mathsf{pk}_x, f_x) = (\mathsf{pk}_j, f_j) \wedge (\mathsf{CT}'_0 \neq \mathsf{CT}_0 \vee \mathsf{CT}'_1 \neq \mathsf{CT}_1)$, then we use the fact that $v^*_{k,1-\beta_k}$'s are chosen uniformly from $\{0,1\}^{2\lambda}$ for all $k \in [2\ell_c]$ to argue that both the circuits return $\perp$. Suppose $(\mathsf{CT}_0, \mathsf{CT}_1) = (\beta_1, \ldots, \beta_{\ell_c}, \beta_{\ell_c+1}, \ldots, \beta_{2\ell_c})$ and $(\mathsf{CT}'_0, \mathsf{CT}'_1) = (\beta'_1, \ldots, \beta'_{\ell_c}, \beta'_{\ell_c+1}, \ldots, \beta'_{2\ell_c})$. Since $\mathsf{CT}'_0 \neq \mathsf{CT}_0$ or $\mathsf{CT}'_1 \neq \mathsf{CT}_1$ there exists $t \in [2\ell_c]$ such that $\beta'_t = 1 - \beta_t$. Let us assume $\sigma_{\mathsf{CT}'} = (u'_k)_{k\in[2\ell_c]}$ where $u'_k \in \{0,1\}^\lambda$ for all $k \in [2\ell_c]$. In order to pass the check of *step 2* in both the circuits, it should hold that $\mathsf{PRG}(u'_t) = v^*_{t,\beta'_t}$. Since $v^*_{t,\beta'_t} = v^*_{t,1-\beta_t}$ is chosen uniformly at random from $\{0,1\}^{2\lambda}$ then the probability that there exists $u' \in \{0,1\}^\lambda$ such that $\mathsf{PRG}(u') = v^*_{t,1-\beta_t}$ is at most $1/2^\lambda$ which is negligible in the security parameter. Therefore, with overwhelming probability it holds that $\mathsf{PRG}(u'_t) \neq v^*_{t,\beta'_t}$. Consequently, the check in *step 2* of both the circuits does not pass and as a result the circuits C_j^{slot} and C_{j-1}^{slot} output $\perp$.

Case 4: If $x = j$ and $(\mathsf{pk}_x, f_x) = (\mathsf{pk}_j, f_j) \wedge (\mathsf{CT}'_0 = \mathsf{CT}_0 \wedge \mathsf{CT}'_1 = \mathsf{CT}_1)$, then we use the fact that pk_j is corrupted and $\mathcal{A}$ is admissible. Let us assume the challenger does not abort in both games $\mathsf{cG}_{3+(j-1),1}$, $\mathsf{cG}_{3+(j-1),2}$. This means that $\mathsf{pk}_x = \mathsf{pk}_j$ where pk_j is either adversarially generated or it is corrupted. Assuming that the check of *step 2* passes in both the circuits, the SKE decryption algorithm of *step 2* recovers m^*_0 from CT_0 in C_{j-1}^{slot} whereas it recovers m^*_1 from CT_1 in C_j^{slot}. Consequently, on input $(\mathsf{sk}_j, j, \mathsf{pk}_j, f_j, \pi_j, \mathsf{CT}_0, \mathsf{CT}_1, \sigma_{\mathsf{CT}})$ the circuit C_{j-1}^{slot} outputs $f_j(m^*_0)$ and the circuit C_j^{slot} outputs $f_j(m^*_1)$. Since $\mathcal{A}$ is admissible, we have $f_j(m^*_0) = f_j(m^*_1)$. In other words, both the circuits C_j^{slot} and C_{j-1}^{slot} output the same value.

Hence, for all possible inputs, the circuits C_j^{slot} and C_{j-1}^{slot} output the same value with overwhelming probability over the choice of $\mathsf{hk}, \mathsf{pk}_q$. Therefore, by the security of $i\mathcal{O}$, the lemma follows. □

Combining Lemma 19 and 27, the Claim 4 holds. □

Therefore, the proof of Claim 2 follows from Claims 3 and 3. □

Finally, the proof of Lemma 21 follows from the Claim 2. □

Lemma 28 *If $i\mathcal{O}$ is secure then for all efficient and admissible adversaries $\mathcal{A}$, for all $\lambda \in \mathbb{N}$ there exists a negligible function negl such that*

$$|\Pr[\mathsf{E}_{3+L}] - \Pr[\mathsf{E}_{4+L}]| = \mathsf{negl}(\lambda).$$

The proof of Lemma 28 follows from a similar argument as in Lemma 20.

Lemma 29 *If SKE is IND-CPA secure then for all efficient and admissible adversaries $\mathcal{A}$, for all $\lambda \in \mathbb{N}$ there exists a negligible function negl such that*

$$|\Pr[\mathsf{E}_{4+L}] - \Pr[\mathsf{E}_{5+L}]| = \mathsf{negl}(\lambda).$$

Proof. The only difference between games $4+L$ and $5+L$ is that the challenge ciphertext component CT_0 is an SKE encryption of m_1^* instead of m_0^*. We show that if $\mathcal{A}$ distinguishes between the hybrids with a non-negligible advantage $\epsilon(\lambda)$ then there exists an adversary $\mathcal{B}$ who breaks the IND-CPA security of SKE with at least an advantage of $\epsilon(\lambda)$. The adversary $\mathcal{B}$ works as follows:

1. $\mathcal{B}$ receives the slot count L from $\mathcal{A}$ and then plays the role of the challenger as in the hybrid $3+L$ for the *setup* and *pre-challenge query* phase.
2. When $\mathcal{B}$ receives the challenge query $(\{(c_i, f_i, \mathsf{pk}_i^*)\}_{i\in[L]}, m_0^*, m_1^*)$ from $\mathcal{A}$, it works exactly the same as the challenger in hybrid $3+L$ except it uses the SKE-challenger to compute CT_0. In particular, $\mathcal{B}$ sends the challenge message pair (m_0^*, m_1^*) to the SKE-challenger and gets back a ciphertext CT_0^*. Finally, $\mathcal{B}$ sends the challenge ciphertext $\mathsf{ct}^* = (\mathsf{CT}_0 \coloneqq \mathsf{CT}_0^*, \mathsf{CT}_1, \widetilde{C}_1, \sigma_{\mathsf{CT}}^*)$ to $\mathcal{A}$. Note that, $\mathcal{B}$ does not require the secret key SK_0 to compute the components CT_1, $\widetilde{C}_1, \sigma_{\mathsf{CT}}^*$ of ct^*.
3. At the end of the experiment, $\mathcal{A}$ outputs a guess $\mathsf{coin}' \in \{0,1\}$ which is also the output of $\mathcal{B}$.

If the SKE-challenger computes $\mathsf{CT}_0^* \leftarrow \mathsf{SKE.Enc}(\mathsf{SK}_1, m_0^*)$ then $\mathcal{B}$ perfectly simulates Game $4+L$. On the other hand, if the SKE-challenger computes $\mathsf{CT}_0^* \leftarrow \mathsf{SKE.Enc}(\mathsf{SK}_1, m_1^*)$ then $\mathcal{B}$ perfectly simulates Game $5+L$. Therefore, $\mathcal{B}$ breaks the IND-CPA security of SKE with advantage at least $\epsilon(\lambda)$ if $\mathcal{A}$ distinguishes between the games with advantage $\epsilon(\lambda)$. Hence, the lemma follows. □

Lemma 30 *If $i\mathcal{O}$ is secure then for all efficient and admissible adversaries $\mathcal{A}$, for all $\lambda \in \mathbb{N}$ there exists a negligible function negl such that*

$$|\Pr[\mathsf{E}_{5+L}] - \Pr[\mathsf{E}_{6+L}]| = \mathsf{negl}(\lambda).$$

Proof. The only difference between the games $5+L$ and $6+L$ is that the ciphertext component $\widetilde{C}_1$ is replaced by $\widetilde{C}_1$. Since $i\mathcal{O}$ is secure it is sufficient to show that the two circuits C_0 and C_1 are equivalent. Let $(\mathsf{sk}_x, x, \mathsf{pk}_x, f_x, \pi_x, \mathsf{CT}_0', \mathsf{CT}_1', \sigma_{\mathsf{CT}}')$ be an arbitrary input to the circuits. The programming of the circuits differ only in *step 3* where $\widehat{m}$ is computed via SKE decryption algorithm. The circuit C_0 always decrypts CT_0 using SK_0 whereas the circuit C_1 decrypts CT_1 using SK_1.

Let $(\mathsf{CT}_0, \mathsf{CT}_1)$ be the part of the challenge ciphertext of Game $5+L$ or $6+L$. Suppose $(\mathsf{CT}_0, \mathsf{CT}_1) = (\beta_1, \ldots, \beta_{\ell_c}, \beta_{\ell_c+1}, \ldots, \beta_{2\ell_c})$ and $(\mathsf{CT}_0', \mathsf{CT}_1') = (\beta_1', \ldots, \beta_{\ell_c}', \beta_{\ell_c+1}', \ldots, \beta_{2\ell_c}')$. We have two cases:

Case 1: If $(\mathsf{CT}_0', \mathsf{CT}_1') = (\mathsf{CT}_0, \mathsf{CT}_1)$, then both the circuits compute $m_1^* \leftarrow \mathsf{SKE.Dec}(\mathsf{SK}_j, \mathsf{CT}_j)$ for $j \in \{0,1\}$ (assuming that the check of *step 2* passes for both the circuits). Therefore, output of the circuits C_0 and C_1 are the same for an input of the from $(\mathsf{sk}_x, x, \mathsf{pk}_x, f_x, \pi_x, \mathsf{CT}_0, \mathsf{CT}_1, \sigma_{\mathsf{CT}}')$.

Case 2: If $(\mathsf{CT}_0', \mathsf{CT}_1') \neq (\mathsf{CT}_0, \mathsf{CT}_1)$, then we rely on the formation of V^* to argue that the circuits C_0 and C_1 output $\perp$. Since $\mathsf{CT}_0' \neq \mathsf{CT}_0$ or $\mathsf{CT}_1' \neq \mathsf{CT}_1$ there exists $t \in [2\ell_c]$ such that $\beta_t' = 1 - \beta_t$. Let us assume $\sigma_{\mathsf{CT}'} = (u_k')_{k\in[2\ell_c]}$ where $u_k' \in \{0,1\}^\lambda$ for all $k \in [2\ell_c]$. In order to pass the check of *step 2* in both the circuits, it should hold that $\mathsf{PRG}(u_t') = v^*_{t,\beta_t'}$. Since $v^*_{t,\beta_t'} = v^*_{t,1-\beta_t}$ is chosen uniformly at random from $\{0,1\}^{2\lambda}$ then the probability that there exists $u' \in \{0,1\}^\lambda$ such that $\mathsf{PRG}(u') = v^*_{t,1-\beta_t}$ is at most $1/2^\lambda$ which is negligible in the security parameter. Therefore, with overwhelming probability it holds that $\mathsf{PRG}(u_t') \neq v^*_{t,\beta_t'}$. Consequently, the check in *step 2* of both the circuits does not pass and as a result the circuits C_0 and C_1 output $\perp$.

Thus, the circuits C_0 and C_1 are equivalent over the choice of V^*. By the security of $i\mathcal{O}$, the distinguishing advantage of $\mathcal{A}$ is negligible in λ. □

Lemma 31 *If PRG is secure then for all efficient and admissible adversaries $\mathcal{A}$, for all $\lambda \in \mathbb{N}$, and $j \in [L]$ there exists a negligible function negl such that*

$$|\Pr[\mathsf{E}_{6+L}] - \Pr[\mathsf{E}_{7+L}]| = \mathsf{negl}(\lambda).$$

The proof of Lemma 31 follows from a similar argument as in Lemma 19.

Lemma 32 *If* SKE *is IND-CPA secure then for all efficient and admissible adversaries* $\mathcal{A}$*, for all* $\lambda \in \mathbb{N}$ *there exists a negligible function* negl *such that*

$$|\Pr[\mathsf{E}_{7+L}] - \Pr[\mathsf{E}_{8+L}]| = \mathsf{negl}(\lambda).$$

The proof of Lemma 32 follows from a similar argument as in Lemma 18.

Finally, the proof the Theorem 5 follows from combining the proofs of the Lemmas 18 to 21 and Lemmas 28 to 32. □

Registered FE from indistinguishability obfuscation: Plugging our slotted registered FE into the transformation described in Section 9, we achieve the following corollary:

Corollary 3 (Registered FE) Let λ be a security parameter. Then, under the existence of one-way function and indistinguishability obfuscation, there exists a registered FE scheme for all polynomial-size circuits (i.e., $|\mathcal{U}_F| = \mathsf{poly}(\lambda)$), and supporting any arbitrary number of users, say L, with the following properties:

- The size of the common reference string and the size of the auxiliary data maintained by the key curator is $\mathsf{poly}(\lambda, \log L)$.
- The running time of key-generation and registration is $\mathsf{poly}(\lambda, \log L)$.
- The size of the master public key and the helper decryption keys are both $\mathsf{poly}(\lambda, \log L)$.
- The size of a ciphertext is $\mathsf{poly}(\lambda, |m|, \log L)$, where $|m|$ denotes the bit-length of the message m.

A direct combination of Theorems 3 to 5 and Corollary 3 implies the following result.

Corollary 4 (Registered FE from Standard Assumptions) Assume sub-exponential security of the following assumptions:

- the Learning Parity with Noise (LPN) assumption over general prime fields $\mathbb{F}_p$ with polynomially many LPN samples and error rate $1/k^\delta$, where k is the dimension of the LPN secret, and $\delta > 0$ is any constant;
- the existence of a Boolean Pseudo-Random Generator (PRG) in NC^0 with stretch $n^{1+\tau}$, where n is the length of the PRG seed, and $\tau > 0$ is any constant;
- the Decision Linear (DLIN) assumption on symmetric bilinear groups of prime order.

Then, registered functional encryption for all polynomial-size circuits supporting an arbitrary number of users exists.

9 From Slotted Registered FE to Registered FE

Hohenberger et al. [HLWW23] showed a transformation from slotted registered ABE to registered ABE. We obtain a similar transformation for upgrading the slotted registered FE to the full-fledged registered FE by tweaking the transformation of [HLWW23]. Roughly, they use a simple "powers-of-two" approach for the conversion. If we want to support $L = 2^\ell$ users in the system then the transformation utilizes $(\ell + 1)$ copies of slotted registered FE to achieve a registered FE.

Let $\mathsf{SlotRFE} = \mathsf{SlotRFE}.(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{IsValid}, \mathsf{Aggregate}, \mathsf{Enc}, \mathsf{Dec})$ be a slotted registered FE scheme with a function universe $\mathcal{U}_F$ and message space $\mathcal{M}$. We construct a registered FE scheme $\mathsf{RFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{RegPK}, \mathsf{Enc}, \mathsf{Update}, \mathsf{Dec})$ that supports a bounded number of users and the same function universe $\mathcal{U}_F$ and the message space $\mathcal{M}$. The transformation works with the following conventions:

- We assume that the bound on the number of users supported by the scheme is $L = 2^\ell$ is a power of two. Rounding the bound to the next power of two incurs at most a factor of 2 overhead.
- The registered FE scheme will be built upon $\ell + 1$ slotted RFE schemes where the k-th scheme is a slotted scheme with 2^k slots for $k \in [0, \ell]$.
- The auxiliary data $\mathsf{aux} = (\mathsf{ctr}, \mathsf{D}_1, \mathsf{D}_2, \mathsf{MPK})$ consists of the following components:
  - A counter ctr that keeps track of the number of registered users in the system.
  - A dictionary D_1 that maps a scheme index $k \in [0, \ell]$ and a slot index $i \in [2^k]$ to a pair (pk, f) which specifies the public key and the function currently assigned to slot i of the scheme k.
  - A dictionary D_2 that maps a scheme index $k \in [0, \ell]$ and a user index $i \in [L]$ to the helper decryption key associated with scheme k and user i.
  - The current master public key $\mathsf{MPK} = (\mathsf{ctr}, \mathsf{MPK}_0, \ldots, \mathsf{MPK}_\ell)$.

  If $\mathsf{aux} = \bot$, we parse it as $(\mathsf{ctr}, \mathsf{D}_1, \mathsf{D}_2, \mathsf{MPK})$ where $\mathsf{ctr} = 0$, $\mathsf{D}_1, \mathsf{D}_2 = \emptyset$, and $\mathsf{MPK} = (0, \bot, \ldots, \bot)$. This corresponds to a fresh scheme with no registered users.

Setup$(1^\lambda, 1^{|\mathcal{U}_F|}, 1^L)$: The setup algorithm takes input as the security parameter λ, the (maximum) size $|\mathcal{U}_F|$ of the functions in $\mathcal{U}_F$ and a bound on the number of users $L = 2^\ell$. It runs the setup algorithm of the slotted RFE scheme for $\ell + 1$ times. In particular, for each $k \in [0, \ell]$, it samples $\mathsf{crs}_k \leftarrow \mathsf{SlotRFE.Setup}(1^{secp}, 1^{|\mathcal{U}_F|}, 1^{2^k})$ and outputs $\mathsf{crs} \coloneqq (\mathsf{crs}_0, \ldots, \mathsf{crs}_\ell)$.

KeyGen$(\mathsf{crs}, \mathsf{aux})$: The key generation algorithm takes input as the common reference string $\mathsf{crs} = (\mathsf{crs}_0, \ldots, \mathsf{crs}_\ell)$ and the auxiliary data $\mathsf{aux} = (\mathsf{ctr}, \mathsf{D}_1, \mathsf{D}_2, \mathsf{MPK})$. For each $k \in [0, \ell]$, the key generation algorithm generates a public/secret key-pair $(\mathsf{pk}_k, \mathsf{sk}_k) \leftarrow \mathsf{SlotRFE.KeyGen}(\mathsf{crs}_k, i_k)$ where $i_k \leftarrow (\mathsf{ctr} \bmod 2^k) + 1 \in [2^k]$ be a slot index for the k-th scheme.

RegPK$(\mathsf{crs}, \mathsf{aux}, \mathsf{pk}, f_{\mathsf{pk}})$: The registration algorithm takes input as the common reference string $\mathsf{crs} = (\mathsf{crs}_0, \ldots, \mathsf{crs}_\ell)$, the auxiliary data $\mathsf{aux} = (\mathsf{ctr}_{\mathsf{aux}}, \mathsf{D}_1, \mathsf{D}_2, \mathsf{MPK})$, where $\mathsf{MPK} = (\mathsf{ctr}_{\mathsf{aux}}, \mathsf{MPK}_0, \ldots, \mathsf{MPK}_\ell)$, a public key pk, and a function $f_{\mathsf{pk}} \in \mathcal{U}_F$. The registration algorithm proceeds as follows:

- For each $k \in [0, \ell]$, let $i_k \coloneqq (\mathsf{ctr}_{\mathsf{aux}} \bmod 2^k) + 1 \in [2^k]$ be the slot index for the k-th scheme.
- For each $k \in [0, \ell]$, check that $\mathsf{SlotRFE.IsValid}(\mathsf{crs}_k, i_k, \mathsf{pk}_k) \stackrel{?}{=} 1$ and $\mathsf{ctr}_{\mathsf{aux}} \stackrel{?}{=} \mathsf{ctr}_{\mathsf{pk}}$. If any of the checks fail then the algorithm halts and outputs the current auxiliary data aux and master public key MPK.
- For each $k \in [0, \ell]$, the registration algorithm updates $\mathsf{D}_1[k, i_k] \leftarrow (\mathsf{pk}, f_{\mathsf{pk}})$. Now we consider two cases.

  Case 1 — all of the slots in the scheme k are filled—If $i_k = 2^k$, the registration algorithm additionally does the following:

  * Compute

    $$(\mathsf{MPK}'_k, \mathsf{hsk}'_{k,1}, \ldots, \mathsf{hsk}'_{k,2^k}) \leftarrow \mathsf{SlotRFE.Aggregate}(\mathsf{crs}_k, \mathsf{D}_1[k, 1], \ldots, \mathsf{D}_1[k, 2^k]).$$

  * Update $\mathsf{D}_2[\mathsf{ctr} + 1 - 2^k + i, k] \leftarrow \mathsf{hsk}'_{k,i}$ for each $i \in [2^k]$.

  Case 2 — all the slots of the k-th scheme are not filled—If $i_k \neq 2^k$, the registration algorithm sets $\mathsf{MPK}'_k \coloneqq \mathsf{MPK}_k$. That is, the master public key is unchanged.
- Define the new master public key $\mathsf{MPK}' = (\mathsf{ctr}_{\mathsf{aux}} + 1, \mathsf{MPK}'_0, \ldots, \mathsf{MPK}'_\ell)$.
- Finally, the registration algorithm outputs the new master public key MPK' and auxiliary data $\mathsf{aux}' = (\mathsf{ctr}_{\mathsf{aux}} + 1, \mathsf{D}_1, \mathsf{D}_2, \mathsf{MPK}')$.

Enc(MPK, m) : The encryption algorithm takes input as the master public key $\mathsf{MPK} = (\mathsf{ctr}, \mathsf{MPK}_0, \ldots, \mathsf{MPK}_\ell)$ and a message $m \in \mathcal{M}$. It computes $\mathsf{ct}_k \leftarrow \mathsf{SlotRFE.Enc}(\mathsf{MPK}_k, m)$ for each $k \in [0, \ell]$; if $\mathsf{MPK}_k = \perp$, then it sets $\mathsf{ct}_k \leftarrow \perp$. Then it outputs $\mathsf{ct} = (\mathsf{ctr}, \mathsf{ct}_0, \ldots, \mathsf{ct}_\ell)$.

Update$(\mathsf{crs}, \mathsf{aux}, \mathsf{pk})$: The update algorithm takes input as the common reference string $\mathsf{crs} = (\mathsf{crs}_0, \ldots, \mathsf{crs}_\ell)$, the auxiliary data $\mathsf{aux} = (\mathsf{ctr}_{\mathsf{aux}}, \mathsf{D}_1, \mathsf{D}_2, \mathsf{MPK})$, and a public key $\mathsf{pk} = (\mathsf{ctr}_{\mathsf{pk}}, \mathsf{pk}_0, \ldots, \mathsf{pk}_\ell)$. It outputs $\perp$ if $\mathsf{ctr}_{\mathsf{pk}} \geq \mathsf{ctr}_{\mathsf{aux}}$. Otherwise, for each $k \in [0, \ell]$, it sets $\mathsf{hsk}_k \leftarrow \mathsf{D}_2[\mathsf{ctr}_{\mathsf{pk}} + 1, k]$ and outputs $\mathsf{hsk} = (\mathsf{hsk}_0, \ldots, \mathsf{hsk}_\ell)$.

Dec$(\mathsf{sk}, \mathsf{hsk}, \mathsf{ct})$: The decryption algorithm takes input as a user secret key $\mathsf{sk} = (\mathsf{ctr}_{\mathsf{sk}}, \mathsf{sk}_0, \ldots, \mathsf{sk}_\ell)$, a helper decryption key $\mathsf{hsk} = (\mathsf{hsk}_0, \ldots, \mathsf{hsk}_\ell)$ and a ciphertext $\mathsf{ct} = (\mathsf{ctr}_{\mathsf{ct}}, \mathsf{ct}_0, \ldots, \mathsf{ct}_\ell)$. It outputs $\perp$ if $\mathsf{ctr}_{\mathsf{ct}} \leq \mathsf{ctr}_{\mathsf{sk}}$. Otherwise, it computes the largest index k on which ctr and ctr' differ (where bits are 0-indexed starting from the least significant bit). If $\mathsf{hsk}_k = \perp$, then the decryption algorithm outputs $\mathsf{GetUpdate}$. Otherwise, it outputs $\mathsf{SlotRFE.Dec}(\mathsf{sk}_k, \mathsf{hsk}_k, \mathsf{ct}_k)$.

We now discuss the correctness, compactness and update efficiency of the construction.

Correctness: Let use assume that the underlying $\mathsf{SlotRFE}$ is perfectly correct. Let $\mathsf{crs} = (\mathsf{crs}_0, \ldots, \mathsf{crs}_\ell) \leftarrow \mathsf{Setup}(1^\lambda, 1^{|\mathcal{U}_F|}, 1^L)$ be the common reference string and $\mathsf{aux} = (\mathsf{ctr}_{\mathsf{aux}}, \mathsf{D}_1, \mathsf{D}_2, \mathsf{MPK})$ be the auxiliary data maintained by the challenger in the correctness game. Note that, by construction, $\mathsf{MPK} = (\mathsf{ctr}_{\mathsf{MPK}}, \mathsf{MPK}_0, \ldots, \mathsf{MPK}_\ell)$ and $\mathsf{ctr}_{\mathsf{MPK}}$ associated with the master public key always coincides with the counter $\mathsf{ctr}_{\mathsf{aux}}$ embedded in aux. Let $(\mathsf{pk}^*, \mathsf{sk}^*)$ be the target key sampled by the challenger in response to a target-key registration query. We use the following lemma from [HLWW23].

Lemma 33 (Hohenberger et al. [HLWW23]) *Let* $\mathsf{aux} = (\mathsf{ctr}_{\mathsf{aux}}, \mathsf{D}_1, \mathsf{D}_2, \mathsf{MPK})$ *be the auxiliary data (maintained by the challenger) during the experiment of correctness after the adversary has made a target-key registration query. Let* $\mathsf{MPK} = (\mathsf{ctr}_{\mathsf{aux}}, \mathsf{MPK}_0, \ldots, \mathsf{MPK}_\ell)$ *be the master public key and* $\mathsf{pk}^* = (\mathsf{ctr}^*, \mathsf{pk}_0^*, \ldots, \mathsf{pk}_\ell^*)$ *be the target key the challenger sampled when responding to the target-key registration query. Let* $k' \in [0, \ell]$ *be the most significant bit on which the binary representations of* ctr^* *and* $\mathsf{ctr}_{\mathsf{aux}}$ *differ (indexed as in Dec algorithm). Then the master public key* MPK_{k^*} *was the output of a call to* $\mathsf{SlotRFE.}\mathit{Aggregate}(\mathsf{crs}_{k'}, \cdot)$ *on a tuple of keys and functions that included the target key* (pk_{k^*}, f^*).

The proof is exactly similar to that of Lemma 6.3 in [HLWW23] except we use functions instead of attributes. Given the lemma holds, we now conclude the correctness of our RFE. Let (i_j, m_j) be the j-th encryption query made by $\mathcal{A}$, and let $\mathsf{ct}_j \leftarrow \mathsf{Enc}(\mathsf{MPK}_{i_j}, m_j)$ be the ciphertext produced by the challenger. Suppose $\mathcal{A}$ made a decryption query on such an index j. According to the construction, the challenger computes $\mathsf{Dec}(\mathsf{sk}^*, \mathsf{hsk}^*, \mathsf{ct}_j)$ as follows:

- It parses $\mathsf{sk}^* = (\mathsf{ctr}^*, \mathsf{sk}_0^*, \ldots, \mathsf{sk}_\ell^*)$, $\mathsf{ct}_j = (\mathsf{ctr}_{\mathsf{ct}}, \mathsf{ct}_{j,0}, \ldots, \mathsf{ct}_{j,\ell})$, $\mathsf{hsk}^* = (\mathsf{hsk}_0^*, \ldots, \mathsf{hsk}_\ell^*)$, and $\mathsf{aux} = (\mathsf{ctr}_{\mathsf{aux}}, \mathsf{D}_1, \mathsf{D}_2, \mathsf{MPK})$ where $\mathsf{MPK} = (\mathsf{ctr}_{\mathsf{aux}}, \mathsf{MPK}_0, \ldots, \mathsf{MPK}_\ell)$.
- Let $k' \in [0, \ell]$ be the index of the most significant bit on which $\mathsf{ctr}_{\mathsf{ct}}$ and ctr^* differ.
- If $\mathsf{hsk}_{k'}^* \neq \perp$, the challenger replies with $m_j' \leftarrow \mathsf{SlotRFE.Dec}(\mathsf{sk}_{k'}^*, \mathsf{hsk}_{k'}^*, \mathsf{ct}_{j,k'})$.
- If $\mathsf{hsk}_{k'}^* = \perp$, the challenger first computes $\mathsf{hsk}^* \leftarrow \mathsf{Update}(\mathsf{crs}, \mathsf{aux}, \mathsf{pk}^*)$ and by construction, this sets $\mathsf{hsk}_{k'}^* \leftarrow \mathsf{D}_2[\mathsf{ctr}^* + 1, k']$. The challenger then replies with $m_j' \leftarrow \mathsf{SlotRFE.Dec}(\mathsf{sk}_{k'}^*, \mathsf{hsk}_{k'}^*, \mathsf{ct}_{j,k'})$.

Since ct_j is the output of $\mathsf{Enc}(\mathsf{MPK}_{i_j}, m_j)$ where $\mathsf{MPK}_{i_j} = (\mathsf{ctr}_{i_j}, \mathsf{MPK}_{i_j,0}, \ldots, \mathsf{MPK}_{i_j,\ell})$, it must be the case that $\mathsf{ctr}_{i_j} = \mathsf{ctr}_{\mathsf{ct}}$. By definition of the correctness game, the i_j-th master public key MPK_{i_j} is constructed after the target key pk^* is registered, so $\mathsf{ctr}_{\mathsf{ct}} \geq \mathsf{ctr}^*$. Now Lemma 33 ensures that the target key-function pair (pk^*, f^*) was aggregated in $\mathsf{MPK}_{i_j,k'}$ which was the output of $\mathsf{SlotRFE.Aggregate}(\mathsf{crs}_{k'}, \cdot)$. Therefore, we must have $\mathsf{D}_2[\mathsf{ctr}^* + 1, k'] = \mathsf{hsk}_{k'}^*$. Moreover, by construction of RegPK, the value of $\mathsf{D}_2[\mathsf{ctr}^* + 1, k']$ will never be updated after the first time it is assigned in a call to RegPK (since the counter $\mathsf{ctr}_{\mathsf{aux}}$ is monotonically increasing). Finally, the correctness of $\mathsf{SlotRFE}$, we have $\mathsf{SlotRFE.Dec}(\mathsf{sk}_{k'}^*, \mathsf{hsk}_{k'}^*, \mathsf{ct}_{j,k'}) = f^*(m_j)$. In other words, the output of $\mathsf{Dec}(\mathsf{sk}^*, \mathsf{hsk}^*, \mathsf{ct}_j)$ must be $m_j' = f^*(m_j)$.

Compactness: The master public key MPK of the RFE scheme consists of a ℓ-bit counter ctr indicating the current number of registered users in the system and a set of $\ell+1$ master public keys $\mathsf{MPK}_0, \ldots, \mathsf{MPK}_\ell$ of the underlying $\mathsf{SlotRFE}$ scheme. Note that each MPK_i is a public key of the slotted scheme with at most $L = 2^\ell$ slots. Therefore, each MPK_i is bounded by $\mathsf{poly}(\lambda, |\mathcal{U}_F|, \log L)$ by the compactness of $\mathsf{SlotRFE}$. Hence, the overall size of MPK is bounded by $\mathsf{poly}(\lambda, |\mathcal{U}_F|, \log L)$. Next, we observe that the helper decryption key hsk of RFE consists of $\ell+1$ helper decryption keys $\mathsf{hsk}_0, \ldots, \mathsf{hsk}_\ell$ of the slotted scheme. By the compactness of the $\mathsf{SlotRFE}$ scheme, each hsk_i is bounded by $\mathsf{poly}(\lambda, |\mathcal{U}_F|, \log L)$. Thus, the overall size of the helper decryption key is bounded by $\mathsf{poly}(\lambda, |\mathcal{U}_F|, \log L)$.

Update efficiency: By construction, the helper decryption key hsk of $\mathsf{SlotRFE}$ consists of $\ell+1$ slotted helper decryption keys $\mathsf{hsk}_0, \ldots, \mathsf{hsk}_\ell$. In the course of the correctness game, the challenger call the Update algorithm at most $\ell + 1 = O(\log L)$ times (one for each hsk_i) since the Update algorithm is only invoked when one of the underlying helper decryption keys hsk_i is $\perp$. After the update of hsk_i, it is no longer updated. Since the auxiliary data maintains a dictionary D_2 mapping each pair of (scheme index, slot index) $(k, i) \in [0, \ell] \times [2^k]$ to its helper decryption key, the update operation can be implemented in $\mathsf{poly}(\lambda, |\mathcal{U}_F|, \log L)$ time in the RAM model of computation.

Security: Next, in the following theorem, we show that the construction of RFE is secure.

Theorem 6 *Assuming* $\mathsf{SlotRFE}$ *is a secure slotted registered functional encryption scheme then the above construction of* RFE *is secure as per Definition 2.*

Proof. We prove this theorem using a sequence of hybrid games. Let $L = 2^\ell$ be the total number of users in the system. Then there are $\ell+1$ slotted registered RFE in the construction. We consider $\ell+2$ many hybrid sequences where each hybrid game is parametrized by $k^* \in [0, \ell+1]$.

Game k^* : This is the registered RFE security game except the challenger sets the challenge ciphertext $\mathsf{ct}^* = (\mathsf{ctr}_{\mathsf{ct}^*}, \mathsf{ct}^*_0, \ldots, \mathsf{ct}^*_\ell)$ by computing $\mathsf{ct}^*_j \leftarrow \mathsf{SlotRFE.Enc}(\mathsf{MPK}_j, m^*_1)$ for $j \in [0, k^*-1]$ and $\mathsf{ct}^*_j \leftarrow \mathsf{SlotRFE.Enc}(\mathsf{MPK}_j, m^*_0)$ for $j \in [k^*, \ell]$. Let $\mathcal{A}$ be an adversary against the security of RFE. More precisely, the game proceeds as follows:

- **Setup phase:** The adversary $\mathcal{A}$ submits the number of users 1^L to the challenger and the challenger samples $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^{|\mathcal{U}_F|}, 1^L)$. The challenger initializes the auxiliary input $\mathsf{aux} \leftarrow \perp$, the initial master public key $\mathsf{MPK} \leftarrow \perp$, a counter $t \leftarrow 0$, an empty set of keys $\mathsf{Cor} \leftarrow \emptyset$ and $\mathsf{Mal} \leftarrow \emptyset$, an empty dictionary $\mathsf{D} \leftarrow \emptyset$. The challenger gives the crs to $\mathcal{A}$.
- **Query phase:** The adversary $\mathcal{A}$ is allowed to query the following queries:
  - **Registered malicious key query:** In a corrupted key query, $\mathcal{A}$ specifies a public key pk and a function $f \in \mathcal{U}_F$. The challenger registers the key by computing $(\mathsf{MPK}', \mathsf{aux}') \leftarrow \mathsf{RegPK}(\mathsf{crs}, \mathsf{aux}, \mathsf{pk}, f)$. The challenger updates its copy of the public key $\mathsf{MPK} \leftarrow \mathsf{MPK}'$, its auxiliary data $\mathsf{aux} \leftarrow \mathsf{aux}'$, adds pk to Mal, and updates $\mathsf{D}[\mathsf{pk}] \leftarrow \mathsf{D}[\mathsf{pk}] \cup \{f\}$. It replies to $\mathcal{A}$ with $(\mathsf{MPK}', \mathsf{aux}')$.
  - **Registered honest key query:** In an honest key query, $\mathcal{A}$ specifies a function $f \in \mathcal{U}_F$. The challenger increments $t \leftarrow t+1$ and samples $(\mathsf{pk}_t, \mathsf{sk}_t) \leftarrow \mathsf{KeyGen}(\mathsf{crs}, \mathsf{aux})$, and registers the key by computing $(\mathsf{MPK}', \mathsf{aux}') \leftarrow \mathsf{RegPK}(\mathsf{crs}, \mathsf{aux}, \mathsf{pk}_t, f)$. The challenger updates its public key $\mathsf{MPK} \leftarrow \mathsf{MPK}'$, its auxiliary data $\mathsf{aux} \leftarrow \mathsf{aux}'$, adds $\mathsf{D}[\mathsf{pk}_t] \leftarrow \mathsf{D}[\mathsf{pk}_t] \cup \{f\}$. It replies to $\mathcal{A}$ with $(t, \mathsf{MPK}', \mathsf{aux}', \mathsf{pk}_t)$.
  - **Corrupt honest key query:** In a corrupt-honest key query, $\mathcal{A}$ specifies an index $i \in [t]$. Let $(\mathsf{pk}_i, \mathsf{sk}_i)$ be the i-th public/secret key the challenger samples when responding to the i-th honest-key-registration query. The challenger adds pk_i to Cor and replies to $\mathcal{A}$ with sk_i.
- **Challenge phase:** In the challenge phase, the adversary $\mathcal{A}$ chooses two messages $m^*_0, m^*_1 \in \mathcal{M}$. Then the challenger computes the challenge ciphertext ct^* as follows:

– Let the current auxiliary data be $\mathsf{aux} = (\mathsf{ctr}_{\mathsf{aux}}, \mathsf{D}_1, \mathsf{D}_2, \mathsf{MPK})$ where $\mathsf{MPK} = (\mathsf{ctr}_{\mathsf{aux}}, \mathsf{MPK}_0, \ldots, \mathsf{MPK}_\ell)$.
– For each $j \in [0, \ell]$, if $\mathsf{MPK}_j = \bot$, then it sets $\mathsf{ct}_j \leftarrow \bot$. Otherwise, if $j < k^*$, it computes $\mathsf{ct}_j^* \leftarrow \mathsf{SlotRFE.Enc}(\mathsf{MPK}_j, m_1^*)$, and if $j \geq k^*$, it computes $\mathsf{ct}_j \leftarrow \mathsf{SlotRFE.Enc}(\mathsf{MPK}_j, m_0^*)$.
– The challenger sends $\mathsf{ct}^* = (\mathsf{ctr}_{\mathsf{aux}}, \mathsf{ct}_0, \ldots, \mathsf{ct}_\ell)$ to $\mathcal{A}$.

Let E_{k^*} be the event of $\mathcal{A}$ outputting the correct bit coin in Game k^*.

Lemma 34 *Assuming that the* SlotRFE *is secure then for any PPT adversary $\mathcal{A}$, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all $\lambda \in \mathbb{N}$, all $k^* \in [0, \ell]$, it holds that* $|\Pr[\mathsf{E}_{k^*}] - \Pr[\mathsf{E}_{k^*+1}]| = \mathsf{negl}(\lambda)$.

Proof. Let us assume that $\mathcal{A}$ is a PPT adversary such that $|\Pr[\mathsf{E}_{k^*}] - \Pr[\mathsf{E}_{k^*+1}]| = \epsilon$ where $\epsilon = \epsilon(\lambda)$ is non-negligible. We use $\mathcal{A}$ to construction a PPT adversary $\mathcal{B}$ against the security of the underlying SlotRFE. The adversary $\mathcal{B}$ works as follows:

- **Setup phase:** $\mathcal{B}$ starts by running $\mathcal{A}$ who outputs the number of slots 1^L. Then the adversary $\mathcal{B}$ proceeds as follows:
  – $\mathcal{B}$ sends $1^{2^{k^*}}$ to its challenger who replies with a common reference string crs_{k^*}. We observe that $1^{2^{k^*}}$ is polynomially bounded as $2^{k^*} \leq L$.
  Note: If the setup algorithm of the underlying SlotRFE runs polylogarithmically in the number of users or slots L, then we can allow $\mathcal{A}$ to submit L in binary instead of encoding it in unary. In this case, $\mathcal{B}$ also outputs 2^{k^*} in binary. In other words, if the underlying SlotRFE supports an arbitrary polynomial number of users, similar to our IO-based construction in Section 8, then the transformed RFE scheme also does.
  – $\mathcal{B}$ internally initializes the auxiliary input $\mathsf{aux} = \bot$, the master public key $\mathsf{MPK} = \bot$, and a dictionary $\mathsf{D} \leftarrow \emptyset$ to keep track of the secret keys associated to the honest key generation queries. In addition, $\mathcal{B}$ maintains two ordered lists S_{cur} and S_{new} which will track the public keys and the associated functions aggregated as part of MPK_{k^*}. Initially, $S_{\mathsf{cur}} \leftarrow \bot$ and $S_{\mathsf{new}} \leftarrow (\bot, \ldots, \bot)$ is an uninitialized list of length 2^{k^*}. For an index $i \in [2^{k^*}]$, we write $S_{\mathsf{new}}[i]$ to denote the i-th element of S_{new}.
  – For each $j \in [0, \ell] \setminus \{k^*\}$, $\mathcal{B}$ samples a common reference string $\mathsf{crs}_j \leftarrow \mathsf{Setup}(1^\lambda, 1^{|\mathcal{U}_F|}, 1^{2^j})$.
  – $\mathcal{B}$ sets and sends $\mathsf{crs} \coloneqq (\mathsf{crs}_0, \ldots, \mathsf{crs}_\ell)$ to $\mathcal{A}$.
- **Query phase:** In this phase, $\mathcal{B}$ simulates the key queries made by $\mathcal{A}$ as follows:
  – **Registered malicious key query:** Suppose $\mathcal{A}$ issues a malicious key-generation-query on public key pk and a function $f \in \mathcal{U}_F$. Let ctr be the current counter associated with aux and $i_{k^*} = (\mathsf{ctr} \bmod 2^{k^*}) + 1$. The adversary $\mathcal{B}$ first runs $(\mathsf{MPK}', \mathsf{aux}') \leftarrow \mathsf{RegPK}(\mathsf{aux}, \mathsf{MPK}, \mathsf{pk}, f)$ and replies to $\mathcal{A}$ with MPK' and aux'. In addition, if $\mathsf{aux} \neq \mathsf{aux}'$ (*i.e.*, the registration process updated the auxiliary input), then $\mathcal{B}$ updates $S_{\mathsf{new}}[i_{k^*}] \leftarrow (\bot, f, \mathsf{pk})$. Moreover, if $i_{k^*} = 2^{k^*}$, then $\mathcal{B}$ sets $S_{\mathsf{cur}} \leftarrow S_{\mathsf{new}}$. Finally, $\mathcal{B}$ updates its local state by assigning $\mathsf{MPK}' \leftarrow \mathsf{MPK}$ and $\mathsf{aux} \leftarrow \mathsf{aux}'$.
  – **Registered honest key query:** Suppose $\mathcal{A}$ makes an honest-key-generation query on a function $f \in \mathcal{U}_F$. The adversary $\mathcal{B}$ proceeds as follows:
    ⋆ Let ctr be the current counter in aux. For each $j \in [0, \ell]$, let $i_j = (\mathsf{ctr} \bmod 2^j) + 1$.
    ⋆ For each $j \neq k^*$, sample $(\mathsf{pk}_j, \mathsf{sk}_j) \leftarrow \mathsf{SlotRFE.KeyGen}(\mathsf{crs}_j, i_j)$.
    ⋆ It makes a honest-key-generation query on slot i_{k^*} to its challenger who replies with (t, pk_{k^*}). It sets the public key $\mathsf{pk} = (\mathsf{ctr}, \mathsf{pk}_0, \ldots, \mathsf{pk}_\ell)$ and adds the mapping $t \mapsto (\mathsf{ctr}, \mathsf{sk}_0, \ldots, \mathsf{sk}_{k^*-1}, \mathsf{sk}_{k^*+1}, \ldots, \mathsf{sk}_\ell)$ to the dictionary D. Here, t is the counter on the number of honest-key-generation queries maintained by the challenger.

Next, $\mathcal{B}$ runs $(\mathsf{MPK}', \mathsf{aux}') \leftarrow \mathsf{RegPK}(\mathsf{crs}, \mathsf{aux}, \mathsf{pk}, f)$ and replies with $(t, \mathsf{MPK}', \mathsf{aux}', \mathsf{pk})$ to $\mathcal{A}$. In addition, $\mathcal{B}$ updates $S_{\mathsf{new}}[i_{k^*}] \leftarrow (t, f, \perp)$. Moreover, if $i_{k^*} = 2^{k^*}$, then $\mathcal{B}$ sets $S_{\mathsf{cur}} \leftarrow S_{\mathsf{new}}$. Finally, $\mathcal{B}$ updates its local state by assigning $\mathsf{MPK} \leftarrow \mathsf{MPK}'$ and $\mathsf{aux} \leftarrow \mathsf{aux}'$.

- **Corrupt honest key query:** Suppose $\mathcal{A}$ makes a corruption query on index i. Then, $\mathcal{B}$ first looks up $(\mathsf{ctr}_{\mathsf{sk}}, \mathsf{sk}_0, \ldots, \mathsf{sk}_{k^*-1}, \mathsf{sk}_{k^*+1}, \ldots, \mathsf{sk}_\ell) \leftarrow \mathsf{D}[i]$. Next, $\mathcal{B}$ makes a corruption query on index i to its challenger who replies with sk_{k^*}. Then, $\mathcal{B}$ sets and sends the secret key $\mathsf{sk} \coloneqq (\mathsf{ctr}_{\mathsf{sk}}, \mathsf{sk}_0, \ldots, \mathsf{sk}_\ell)$.

- **Challenge phase:** When $\mathcal{A}$ outputs a pair of challenge messages $m_0^*, m_1^* \in \mathcal{M}$ the adversary $\mathcal{B}$ computes the the challenge ciphertext ct^* as follows. Let $\mathsf{MPK} = (\mathsf{ctr}, \mathsf{MPK}_0, \ldots, \mathsf{MPK}_\ell)$ be the current master public key. The adversary $\mathcal{B}$ gives the challenge ciphertext $\mathsf{ct}^* = (\mathsf{ctr}_{\mathsf{aux}}, \mathsf{ct}_0^*, \ldots, \mathsf{ct}_\ell^*)$ to $\mathcal{A}$ where ct_j^* are computed as follows:
  - If $\mathsf{MPK}_j = \perp$, then $\mathsf{ct}_j^* \leftarrow \perp$.
  - If $\mathsf{MPK}_j \neq \perp$ and $j < k^*$, let $\mathsf{ct}_j^* \leftarrow \mathsf{SlotRFE.Enc}(\mathsf{MPK}_j, m_1^*)$.
  - If $\mathsf{MPK}_j \neq \perp$ and $j > k^*$, let $\mathsf{ct}_j^* \leftarrow \mathsf{SlotRFE.Enc}(\mathsf{MPK}_j, m_0^*)$.
  - If $\mathsf{MPK}_j \neq \perp$ and $j = k^*$, the adversary $\mathcal{B}$ makes a challenge query using the components S_{cur} as the public key-function pair for the slots in the k^*-th $\mathsf{SlotRFE}$, and m_0^*, m_1^* as the pair of challenge messages to its challenger. In a reply, $\mathcal{B}$ receives a ciphertext $\mathsf{ct}_{k^*}^*$ from its challenger.

- **Output phase:** At the end of the experiment, the adversary $\mathcal{A}$ outputs a bit $\mathsf{coin}' \in \{0, 1\}$ which is also the output of $\mathcal{B}$.

We now show that if $\mathcal{A}$ is an admissible adversary of RFE then $\mathcal{B}$ is also an admissible adversary of $\mathsf{SlotRFE}$. By construction, the set S_{cur} exactly tracks the public keys currently aggregated in MPK_{k^*}. If $\mathsf{MPK}_{k^*} = \perp$, then $\mathcal{B}$ does not make a challenge query and hence, by definition, $\mathcal{B}$ is admissible. Suppose, $\mathsf{MPK}_{k^*} \neq \perp$. In this case, $S_{\mathsf{cur}} \neq \perp$. We now consider each component of S_{cur} in the challenge phase:

- $S_{\mathsf{cur}}[i] = (i_j, f, \perp)$: In this case, the adversary $\mathcal{A}$ made a honest-key-registration query with the function f. Since $\mathcal{A}$ is admissible, either f satisfies the equation $f(m_0^*) = f(m_1^*)$ or $\mathcal{A}$ did not make a corruption query on index i_j. Correspondingly, this means that either f satisfies the equation $f(m_0^*) = f(m_1^*)$ or $\mathcal{B}$ does not make a corruption query on index i_j. In both the cases, $\mathcal{B}$ obeys the rule of admissibility.

- $S_{\mathsf{cur}}[i] = (\perp, f, \mathsf{pk})$: In this case, $\mathcal{A}$ made a malicious-key-registration query with public key pk and function f. Since $\mathcal{A}$ is admissible it must hold that $f(m_0^*) = f(m_1^*)$. Therefore, $\mathcal{B}$ is also admissible in this case.

Next, it is easy to observe that $\mathcal{B}$ perfectly simulates the security experiment of the RFE protocol for $\mathcal{A}$. Moreover, if $\mathsf{ct}_{k^*}^* \leftarrow \mathsf{SlotRFE.Enc}(\mathsf{MPK}_{k^*}, m_0^*)$, the adversary $\mathcal{B}$ simulates Game k^* whereas if $\mathsf{ct}_{k^*}^* \leftarrow \mathsf{SlotRFE.Enc}(\mathsf{MPK}_{k^*}, m_1^*)$, the adversary $\mathcal{B}$ simulates Game $k^* + 1$. Consequently, the adversary $\mathcal{B}$ wins the $\mathsf{SlotRFE}$ security game with the same non-negligible advantage ϵ which contradicts the fact that the underlying $\mathsf{SlotRFE}$ is secure. Hence, the lemma follows. □

Finally, we observe that in Game 0, the challenge ciphertext is an encryption of m_0^* and it coincides with the security experiment of RFE with $\mathsf{coin} = 0$. On the other hand, in Game $\ell + 1$, the challenge ciphertext is an encryption of m_1^* and it coincides with the security experiment of RFE with $\mathsf{coin} = 1$. Since ℓ is polynomial in λ, the proof the theorem follows. □

References

[ABB10a] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 553–572. Springer, Heidelberg, May / June 2010. doi:10.1007/978-3-642-13190-5_28.

[ABB10b] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical IBE. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 98–115. Springer, Heidelberg, August 2010. doi:10.1007/978-3-642-14623-7_6.

[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. Springer, Heidelberg, March / April 2015. doi:10.1007/978-3-662-46447-2_33.

[ABKW19] Michel Abdalla, Fabrice Benhamouda, Markulf Kohlweiss, and Hendrik Waldner. Decentralizing inner-product functional encryption. 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 128–157. Springer, Heidelberg, April 2019. doi:10.1007/978-3-030-17259-6_5.

[ACGU20a] Michel Abdalla, Dario Catalano, Romain Gay, and Bogdan Ursu. Inner-product functional encryption with fine-grained access control. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2020, Part III*, volume 12493 of *Lecture Notes in Computer Science*, pages 467–497. Springer, Heidelberg, December 2020. doi:10.1007/978-3-030-64840-4_16.

[ACGU20b] Michel Abdalla, Dario Catalano, Romain Gay, and Bogdan Ursu. Inner-product functional encryption with fine-grained access control. *IACR Cryptology ePrint Archive, Report 2020/577*, 2020.

[Agr17] Shweta Agrawal. Stronger security for reusable garbled circuits, general definitions and attacks. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 3–35. Springer, Heidelberg, August 2017. doi:10.1007/978-3-319-63688-7_1.

[AGW20] Michel Abdalla, Junqing Gong, and Hoeteck Wee. Functional encryption for attribute-weighted sums from k-Lin. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020, Part I*, volume 12170 of *Lecture Notes in Computer Science*, pages 685–716. Springer, Heidelberg, August 2020. doi:10.1007/978-3-030-56784-2_23.

[AJ15] Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 308–326. Springer, Heidelberg, August 2015. doi:10.1007/978-3-662-47989-6_15.

[AJS15] Prabhanjan Ananth, Abhishek Jain, and Amit Sahai. Indistinguishability obfuscation from functional encryption for simple functions. Cryptology ePrint Archive, Report 2015/730, 2015. https://eprint.iacr.org/2015/730.

[AKM+22] Shweta Agrawal, Fuyuki Kitagawa, Anuja Modi, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Bounded functional encryption for turing machines: Adaptive security from general assumptions. In Eike Kiltz and Vinod Vaikuntanathan, editors, *TCC 2022: 20th Theory of Cryptography Conference, Part I*, volume 13747 of *Lecture Notes in Computer Science*, pages 618–647. Springer, Heidelberg, November 2022. doi:10.1007/978-3-031-22318-1_22.

[ALMT20] Shweta Agrawal, Benoît Libert, Monosij Maitra, and Radu Titiu. Adaptive simulation security for inner product functional encryption. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020: 23rd International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 12110 of *Lecture Notes in Computer Science*, pages 34–64. Springer, Heidelberg, May 2020. doi:10.1007/978-3-030-45374-9_2.

[ALS15] Shweta Agrawal, Benoît Libert, and Damien Stehlé. Fully secure functional encryption for inner products, from standard assumptions. Cryptology ePrint Archive, Report 2015/608, 2015. https://eprint.iacr.org/2015/608.

[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. Springer, Heidelberg, August 2016. doi:10.1007/978-3-662-53015-3_12.

[AMVY21] Shweta Agrawal, Monosij Maitra, Narasimha Sai Vempati, and Shota Yamada. Functional encryption for turing machines with dynamic bounded collusion from LWE. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part IV*, volume 12828 of *Lecture Notes in Computer Science*, pages 239–269, Virtual Event, August 2021. Springer, Heidelberg. doi:10.1007/978-3-030-84259-8_9.

[AS16] Prabhanjan Vijendra Ananth and Amit Sahai. Functional encryption for turing machines. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A: 13th Theory of Cryptography Conference, Part I*, volume 9562 of *Lecture Notes in Computer Science*, pages 125–153. Springer, Heidelberg, January 2016. doi:10.1007/978-3-662-49096-9_6.

[Att14] Nuttapong Attrapadung. Dual system encryption via doubly selective security: Framework, fully secure functional encryption for regular languages, and more. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 557–577. Springer, Heidelberg, May 2014. doi:10.1007/978-3-642-55220-5_31.

[AV19] Prabhanjan Ananth and Vinod Vaikuntanathan. Optimal bounded-collusion secure functional encryption. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019: 17th Theory of Cryptography Conference, Part I*, volume 11891 of *Lecture Notes in Computer Science*, pages 174–198. Springer, Heidelberg, December 2019. doi:10.1007/978-3-030-36030-6_8.

[AY20] Shweta Agrawal and Shota Yamada. Optimal broadcast encryption from pairings and LWE. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020, Part I*, volume 12105 of *Lecture Notes in Computer Science*, pages 13–43. Springer, Heidelberg, May 2020. doi:10.1007/978-3-030-45721-1_2.

[BB04] Dan Boneh and Xavier Boyen. Secure identity based encryption without random oracles. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 443–459. Springer, Heidelberg, August 2004. doi:10.1007/978-3-540-28628-8_27.

[BBL17] Fabrice Benhamouda, Florian Bourse, and Helger Lipmaa. CCA-secure inner-product functional encryption from projective hash functions. In Serge Fehr, editor, *PKC 2017: 20th International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 10175 of *Lecture Notes in Computer Science*, pages 36–66. Springer, Heidelberg, March 2017. doi:10.1007/978-3-662-54388-7_2.

[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. Springer, Heidelberg, August 2017. doi:10.1007/978-3-319-63688-7_3.

[Bei96] Amos Beimel. *Secure schemes for secret sharing and key distribution*. PhD thesis, Technion - Israel Institute of Technology, Israel, 1996. URL https://technion.primo.exlibrisgroup.com/permalink/972TEC_INST/q1jq5o/alma990021768270203971.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, Heidelberg, August 2001. doi:10.1007/3-540-44647-8_13.

[BGG+14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 533–556. Springer, Heidelberg, May 2014. doi:10.1007/978-3-642-55220-5_30.

[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. Springer, Heidelberg, August 2001. doi:10.1007/3-540-44647-8_1.

[BGJS17] Saikrishna Badrinarayanan, Vipul Goyal, Aayush Jain, and Amit Sahai. A note on VRFs from verifiable functional encryption. Cryptology ePrint Archive, Report 2017/051, 2017. https://eprint.iacr.org/2017/051.

[BGJT14] Razvan Barbulescu, Pierrick Gaudry, Antoine Joux, and Emmanuel Thomé. A heuristic quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 1–16. Springer, Heidelberg, May 2014. doi:10.1007/978-3-642-55220-5_1.

[BGW05] Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 258–275. Springer, Heidelberg, August 2005. doi:10.1007/11535218_16.

[Bit17] Nir Bitansky. Verifiable random functions from non-interactive witness-indistinguishable proofs. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017: 15th Theory of Cryptography Conference, Part II*, volume 10678 of *Lecture Notes in Computer Science*, pages 567–594. Springer, Heidelberg, November 2017. doi:10.1007/978-3-319-70503-3_19.

[BJK15] Allison Bishop, Abhishek Jain, and Lucas Kowalczyk. Function-hiding inner product encryption. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology – ASIACRYPT 2015, Part I*, volume 9452 of *Lecture Notes in Computer Science*, pages 470–491. Springer, Heidelberg, November / December 2015. doi:10.1007/978-3-662-48797-6_20.

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Yuval Ishai, editor, *TCC 2011: 8th Theory of Cryptography Conference*, volume 6597 of *Lecture Notes in Computer Science*, pages 253–273. Springer, Heidelberg, March 2011. doi:10.1007/978-3-642-19571-6_16.

[BV15] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. In Venkatesan Guruswami, editor, *56th Annual Symposium on Foundations of Computer Science*, pages 171–190. IEEE Computer Society Press, October 2015. doi:10.1109/FOCS.2015.20.

[CDSG+20] Jérémy Chotard, Edouard Dufour-Sans, Romain Gay, Duong Hieu Phan, and David Pointcheval. Dynamic decentralized functional encryption. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020, Part I*, volume 12170 of *Lecture Notes in Computer Science*, pages 747–775. Springer, Heidelberg, August 2020. doi:10.1007/978-3-030-56784-2_25.

[CES21] Kelong Cong, Karim Eldefrawy, and Nigel P. Smart. Optimizing registration based encryption. Cryptology ePrint Archive, Report 2021/499, 2021. https://eprint.iacr.org/2021/499.

[CGKW18] Jie Chen, Junqing Gong, Lucas Kowalczyk, and Hoeteck Wee. Unbounded ABE via bilinear entropy expansion, revisited. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part I*, volume 10820 of *Lecture Notes in Computer Science*, pages 503–534. Springer, Heidelberg, April / May 2018. doi:10.1007/978-3-319-78381-9_19.

[CGW15] Jie Chen, Romain Gay, and Hoeteck Wee. Improved dual system ABE in prime-order groups via predicate encodings. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 595–624. Springer, Heidelberg, April 2015. doi:10.1007/978-3-662-46803-6_20.

[CLT18] Guilhem Castagnos, Fabien Laguillaumie, and Ida Tucker. Practical fully secure unrestricted inner product functional encryption modulo p. In Thomas Peyrin and Steven Galbraith, editors, *Advances in Cryptology – ASIACRYPT 2018, Part II*, volume 11273 of *Lecture Notes in Computer Science*, pages 733–764. Springer, Heidelberg, December 2018. doi:10.1007/978-3-030-03329-3_25.

[CS19] R. Joseph Connor and Max Schuchard. Blind bernoulli trials: A noninteractive protocol for hidden-weight coin flips. In Nadia Heninger and Patrick Traynor, editors, *USENIX Security 2019: 28th USENIX Security Symposium*, pages 1483–1500. USENIX Association, August 2019.

[DDM16] Pratish Datta, Ratna Dutta, and Sourav Mukhopadhyay. Functional encryption for inner product with full function privacy. In Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang, editors, *PKC 2016: 19th International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 9614 of *Lecture Notes in Computer Science*, pages 164–195. Springer, Heidelberg, March 2016. doi:10.1007/978-3-662-49384-7_7.

[DDM+23] Uddipana Dowerah, Subhranil Dutta, Aikaterini Mitrokotsa, Sayantan Mukherjee, and Tapas Pal. Unbounded predicate inner product functional encryption from pairings. *Journal of Cryptology*, 36, 2023. doi:10.1007/s00145-023-09458-2.

[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, Part III*, volume 14006 of *Lecture Notes in Computer Science*, pages 417–446. Springer, Heidelberg, April 2023. doi:10.1007/978-3-031-30620-4_14.

[DOT18] Pratish Datta, Tatsuaki Okamoto, and Katsuyuki Takashima. Adaptively simulation-secure attribute-hiding predicate encryption. In Thomas Peyrin and Steven Galbraith, editors, *Advances in Cryptology – ASIACRYPT 2018, Part II*, volume 11273 of *Lecture Notes in Computer Science*, pages 640–672. Springer, Heidelberg, December 2018. doi:10.1007/978-3-030-03329-3_22.

[DP19] Edouard Dufour Sans and David Pointcheval. Unbounded inner-product functional encryption with succinct keys. In Robert H. Deng, Valérie Gauthier-Umaña, Martín Ochoa, and Moti Yung, editors, *ACNS 19: 17th International Conference on Applied Cryptography and Network Security*, volume 11464 of *Lecture Notes in Computer Science*, pages 426–441. Springer, Heidelberg, June 2019. doi:10.1007/978-3-030-21568-2_21.

[DP21] Pratish Datta and Tapas Pal. (Compact) adaptively secure FE for attribute-weighted sums from k-lin. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2021, Part IV*, volume 13093 of *Lecture Notes in Computer Science*, pages 434–467. Springer, Heidelberg, December 2021. doi:10.1007/978-3-030-92068-5_15.

[DP23] Pratish Datta and Tapas Pal. Decentralized multi-authority attribute-based inner-product FE: Large universe and unbounded. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *PKC 2023: 26th International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 13940 of *Lecture Notes in Computer Science*, pages 587–621. Springer, Heidelberg, May 2023. doi:10.1007/978-3-031-31368-4_21.

[DPT22] Pratish Datta, Tapas Pal, and Katsuyuki Takashima. Compact FE for unbounded attribute-weighted sums for logspace from SXDH. In Shweta Agrawal and Dongdai Lin, editors, *Advances in Cryptology – ASIACRYPT 2022, Part I*, volume 13791 of *Lecture Notes in Computer Science*, pages 126–159. Springer, Heidelberg, December 2022. doi:10.1007/978-3-031-22963-3_5.

[EHK+13] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. An algebraic framework for Diffie-Hellman assumptions. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 129–147. Springer, Heidelberg, August 2013. doi:10.1007/978-3-642-40084-1_8.

[FFM+23] Danilo Francati, Daniele Friolo, Monosij Maitra, Giulio Malavolta, Ahmadreza Rahimi, and Daniele Venturi. Registered (inner-product) functional encryption. Springer-Verlag, 2023.

[FKdP23] Dario Fiore, Dimitris Kolonelos, and Paola de Perthuis. Cuckoo commitments: Registration-based encryption and key-value map commitments for large spaces. Springer-Verlag, 2023.

[FWW23] Cody Freitag, Brent Waters, and David J. Wu. How to use (plain) witness encryption: Registered abe, flexible broadcast, and more. Springer-Verlag, 2023. doi:10.1007/978-3-031-38551-3_16.

[Gay20] Romain Gay. A new paradigm for public-key functional encryption for degree-2 polynomials. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020: 23rd International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 12110 of *Lecture Notes in Computer Science*, pages 95–120. Springer, Heidelberg, May 2020. doi:10.1007/978-3-030-45374-9_4.

[GGH+13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th Annual Symposium on Foundations of Computer Science*, pages 40–49. IEEE Computer Society Press, October 2013. doi:10.1109/FOCS.2013.13.

[GGLW22] Rachit Garg, Rishab Goyal, George Lu, and Brent Waters. Dynamic collusion bounded functional encryption from identity-based encryption. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology – EUROCRYPT 2022, Part II*, volume 13276 of *Lecture Notes in Computer Science*, pages 736–763. Springer, Heidelberg, May / June 2022. doi:10.1007/978-3-031-07085-3_25.

[GGMZ13] Faruk Göloglu, Robert Granger, Gary McGuire, and Jens Zumbrägel. On the function field sieve and the impact of higher splitting probabilities — application to discrete logarithms in $\mathbb{F}_{2^{1971}}$ and $\mathbb{F}_{2^{3164}}$. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 109–128. Springer, Heidelberg, August 2013. doi:10.1007/978-3-642-40084-1_7.

[GHKW17] Rishab Goyal, Susan Hohenberger, Venkata Koppula, and Brent Waters. A generic approach to constructing and proving verifiable random functions. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017: 15th Theory of Cryptography Conference, Part II*, volume 10678 of *Lecture Notes in Computer Science*, pages 537–566. Springer, Heidelberg, November 2017. doi:10.1007/978-3-319-70503-3_18.

[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. Springer, Heidelberg, April 2019. doi:10.1007/978-3-030-17259-6_3.

[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. Springer, Heidelberg, November 2018. doi:10.1007/978-3-030-03807-6_25.

[GKP+13] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th Annual ACM Symposium on Theory of Computing*, pages 555–564. ACM Press, June 2013. doi:10.1145/2488608.2488678.

[GKW17] Rishab Goyal, Venkata Koppula, and Brent Waters. Lockable obfuscation. In Chris Umans, editor, *58th Annual Symposium on Foundations of Computer Science*, pages 612–621. IEEE Computer Society Press, October 2017. doi:10.1109/FOCS.2017.62.

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006: 13th Conference on Computer and Communications Security*, pages 89–98. ACM Press, October / November 2006. doi:10.1145/1180405.1180418. Available as Cryptology ePrint Archive Report 2006/309.

[GPSZ17] Sanjam Garg, Omkant Pandey, Akshayaram Srinivasan, and Mark Zhandry. Breaking the subexponential barrier in obfustopia. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017, Part III*, volume 10212 of *Lecture Notes in Computer Science*, pages 156–181. Springer, Heidelberg, April / May 2017. doi:10.1007/978-3-319-56617-7_6.

[GV20] Rishab Goyal and Satyanarayana Vusirikala. Verifiable registration-based encryption. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020, Part I*, volume 12170 of *Lecture Notes in Computer Science*, pages 621–651. Springer, Heidelberg, August 2020. doi:10.1007/978-3-030-56784-2_21.

[GVW12] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 162–179. Springer, Heidelberg, August 2012. doi:10.1007/978-3-642-32009-5_11.

[GVW13] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th Annual ACM Symposium on Theory of Computing*, pages 545–554. ACM Press, June 2013. doi:10.1145/2488608.2488677.

[GVW15] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate encryption for circuits from LWE. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 503–523. Springer, Heidelberg, August 2015. doi:10.1007/978-3-662-48000-7_25.

[GW20] Junqing Gong and Hoeteck Wee. Adaptively secure ABE for DFA from k-Lin and more. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020, Part III*,

volume 12107 of *Lecture Notes in Computer Science*, pages 278–308. Springer, Heidelberg, May 2020. doi:10.1007/978-3-030-45727-3_10.

[GWW19] Junqing Gong, Brent Waters, and Hoeteck Wee. ABE for DFA from k-Lin. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part II*, volume 11693 of *Lecture Notes in Computer Science*, pages 732–764. Springer, Heidelberg, August 2019. doi:10.1007/978-3-030-26951-7_25.

[HG90] Johan Håstad and Mikael Goldmann. On the power of small-depth threshold circuits. In *31st Annual Symposium on Foundations of Computer Science*, pages 610–618. IEEE Computer Society Press, October 1990. doi:10.1109/FSCS.1990.89582.

[HJO+16] Brett Hemenway, Zahra Jafargholi, Rafail Ostrovsky, Alessandra Scafuro, and Daniel Wichs. Adaptively secure garbled circuits from one-way functions. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part III*, volume 9816 of *Lecture Notes in Computer Science*, pages 149–178. Springer, Heidelberg, August 2016. doi:10.1007/978-3-662-53015-3_6.

[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, Part III*, volume 14006 of *Lecture Notes in Computer Science*, pages 511–542. Springer, Heidelberg, April 2023. doi:10.1007/978-3-031-30620-4_17.

[HW15] Pavel Hubacek and Daniel Wichs. On the communication complexity of secure function evaluation with long output. In Tim Roughgarden, editor, *ITCS 2015: 6th Conference on Innovations in Theoretical Computer Science*, pages 163–172. Association for Computing Machinery, January 2015. doi:10.1145/2688073.2688105.

[JLS21] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In Samir Khuller and Virginia Vassilevska Williams, editors, *53rd Annual ACM Symposium on Theory of Computing*, pages 60–73. ACM Press, June 2021. doi:10.1145/3406325.3451093.

[JLS22] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from LPN over $\mathbb{F}_p$, DLIN, and PRGs in NC^0. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology – EUROCRYPT 2022, Part I*, volume 13275 of *Lecture Notes in Computer Science*, pages 670–699. Springer, Heidelberg, May / June 2022. doi:10.1007/978-3-031-06944-4_23.

[Jou13] Antoine Joux. Faster index calculus for the medium prime case application to 1175-bit and 1425-bit finite fields. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 177–193. Springer, Heidelberg, May 2013. doi:10.1007/978-3-642-38348-9_11.

[Jou14] Antoine Joux. A new index calculus algorithm with complexity $L(1/4 + o(1))$ in small characteristic. In Tanja Lange, Kristin Lauter, and Petr Lisonek, editors, *SAC 2013: 20th Annual International Workshop on Selected Areas in Cryptography*, volume 8282 of *Lecture Notes in Computer Science*, pages 355–379. Springer, Heidelberg, August 2014. doi:10.1007/978-3-662-43414-7_18.

[KLM+18] Sam Kim, Kevin Lewi, Avradip Mandal, Hart Montgomery, Arnab Roy, and David J. Wu. Function-hiding inner product encryption is practical. In Dario Catalano and Roberto De Prisco, editors, *SCN 18: 11th International Conference on Security in Communication Networks*, volume 11035 of *Lecture Notes in Computer Science*, pages 544–562. Springer, Heidelberg, September 2018. doi:10.1007/978-3-319-98113-0_29.

[KLW15] Venkata Koppula, Allison Bishop Lewko, and Brent Waters. Indistinguishability obfuscation for turing machines with unbounded memory. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th Annual ACM Symposium on Theory of Computing*, pages 419–428. ACM Press, June 2015. doi:10.1145/2746539.2746614.

[KMW23] Dimitris Kolonelos, Giulio Malavolta, and Hoeteck Wee. Distributed broadcast encryption from bilinear groups. Springer-Verlag, 2023.

[KNT21] Fuyuki Kitagawa, Ryo Nishimaki, and Keisuke Tanaka. Simple and generic constructions of succinct functional encryption. *Journal of Cryptology*, 34(3):25, July 2021. doi:10.1007/s00145-021-09396-x.

[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. Springer, Heidelberg, April 2008. doi:10.1007/978-3-540-78967-3_9.

[KW19] Lucas Kowalczyk and Hoeteck Wee. Compact adaptively secure ABE for NC^1 from k-Lin. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019, Part I*, volume 11476 of *Lecture Notes in Computer Science*, pages 3–33. Springer, Heidelberg, May 2019. doi:10.1007/978-3-030-17653-2_1.

[KW20] Lucas Kowalczyk and Hoeteck Wee. Compact adaptively secure ABE for NC^1 from k-Lin. *Journal of Cryptology*, 33(3):954–1002, July 2020. doi:10.1007/s00145-019-09335-x.

[Lam79] Leslie Lamport. Constructing digital signatures from a one way function. Technical Report CSL-98, October 1979. URL https://www.microsoft.com/en-us/research/publication/constructing-digital-signatures-one-way-function/. This paper was published by IEEE in the Proceedings of HICSS-43 in January, 2010.

[Lev85] Leonid A. Levin. One-way functions and pseudorandom generators. In *17th Annual ACM Symposium on Theory of Computing*, pages 363–365. ACM Press, May 1985. doi:10.1145/22145.22185.

[LL20a] Huijia Lin and Ji Luo. Compact adaptively secure ABE from k-Lin: Beyond NC^1 and towards NL. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020, Part III*, volume 12107 of *Lecture Notes in Computer Science*, pages 247–277. Springer, Heidelberg, May 2020. doi:10.1007/978-3-030-45727-3_9.

[LL20b] Huijia Lin and Ji Luo. Succinct and adaptively secure ABE for ABP from k-lin. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2020, Part III*, volume 12493 of *Lecture Notes in Computer Science*, pages 437–466. Springer, Heidelberg, December 2020. doi:10.1007/978-3-030-64840-4_15.

[LLW21] Qiqi Lai, Feng-Hao Liu, and Zhedong Wang. New lattice two-stage sampling technique and its applications to functional encryption - stronger security and smaller ciphertexts. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021, Part I*, volume 12696 of *Lecture Notes in Computer Science*, pages 498–527. Springer, Heidelberg, October 2021. doi:10.1007/978-3-030-77870-5_18.

[LOS+10a] Allison Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In *EUROCRYPT 2010*, pages 62–91. Springer, 2010.

[LOS+10b] Allison B. Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 62–91. Springer, Heidelberg, May / June 2010. doi:10.1007/978-3-642-13190-5_4.

[LV16] Huijia Lin and Vinod Vaikuntanathan. Indistinguishability obfuscation from DDH-like assumptions on constant-degree graded encodings. In Irit Dinur, editor, *57th Annual Symposium on Foundations of Computer Science*, pages 11–20. IEEE Computer Society Press, October 2016. doi:10.1109/FOCS.2016.11.

[LW10] Allison B. Lewko and Brent Waters. New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In Daniele Micciancio, editor, *TCC 2010: 7th Theory of Cryptography Conference*, volume 5978 of *Lecture Notes in Computer Science*, pages 455–479. Springer, Heidelberg, February 2010. doi:10.1007/978-3-642-11799-2_27.

[LW11a] Allison B. Lewko and Brent Waters. Decentralizing attribute-based encryption. In Kenneth G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 568–588. Springer, Heidelberg, May 2011. doi:10.1007/978-3-642-20465-4_31.

[LW11b] Allison B. Lewko and Brent Waters. Unbounded HIBE and attribute-based encryption. In Kenneth G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 547–567. Springer, Heidelberg, May 2011. doi:10.1007/978-3-642-20465-4_30.

[LW12] Allison B. Lewko and Brent Waters. New proof methods for attribute-based encryption: Achieving full security through selective techniques. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 180–198. Springer, Heidelberg, August 2012. doi:10.1007/978-3-642-32009-5_12.

[MKMS21] Jose Maria Bermudo Mera, Angshuman Karmakar, Tilen Marc, and Azam Soleimanian. Efficient lattice-based inner-product functional encryption. Cryptology ePrint Archive, Report 2021/046, 2021. https://eprint.iacr.org/2021/046.

[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. Springer, Heidelberg, May 2016. doi:10.1007/978-3-662-49896-5_14.

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd Annual ACM Symposium on Theory of Computing*, pages 427–437. ACM Press, May 1990. doi:10.1145/100216.100273.

[O'N10] Adam O'Neill. Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556, 2010. https://eprint.iacr.org/2010/556.

[OPWW15] Tatsuaki Okamoto, Krzysztof Pietrzak, Brent Waters, and Daniel Wichs. New realizations of somewhere statistically binding hashing and positional accumulators. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology – ASIACRYPT 2015, Part I*, volume 9452 of *Lecture Notes in Computer Science*, pages 121–145. Springer, Heidelberg, November / December 2015. doi:10.1007/978-3-662-48797-6_6.

[OT10] Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 191–208. Springer, Heidelberg, August 2010. doi:10.1007/978-3-642-14623-7_11.

[OT12] Tatsuaki Okamoto and Katsuyuki Takashima. Adaptively attribute-hiding (hierarchical) inner product encryption. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 591–608. Springer, Heidelberg, April 2012. doi:10.1007/978-3-642-29011-4_35.

[PD21a] Tapas Pal and Ratna Dutta. Attribute-based access control for inner product functional encryption from LWE. In Patrick Longa and Carla Ràfols, editors, *Progress in Cryptology - LATINCRYPT 2021: 7th International Conference on Cryptology and Information Security in Latin America*, volume 12912 of *Lecture Notes in Computer Science*, pages 127–148, Bogotá, Colombia, October 2021. Springer, Heidelberg. doi:10.1007/978-3-030-88238-9_7.

[PD21b] Tapas Pal and Ratna Dutta. Attribute-based access control for inner product functional encryption from LWE. In Patrick Longa and Carla Ràfols, editors, *Progress in Cryptology - LATINCRYPT 2021 - 7th International Conference on Cryptology and Information Security in Latin America, Bogotá, Colombia, October 6-8, 2021, Proceedings*, volume 12912 of *Lecture Notes in Computer Science*, pages 127–148. Springer, 2021. doi:10.1007/978-3-030-88238-9_7.

[Sho97a] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266. Springer, Heidelberg, May 1997. doi:10.1007/3-540-69053-0_18.

[Sho97b] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266. Springer, 1997. doi:10.1007/3-540-69053-0_18.

[SW05] Amit Sahai and Brent R. Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 457–473. Springer, Heidelberg, May 2005. doi:10.1007/11426639_27.

[TAO16] Junichi Tomida, Masayuki Abe, and Tatsuaki Okamoto. Efficient functional encryption for inner-product values with full-hiding security. In Matt Bishop and Anderson C. A. Nascimento, editors, *ISC 2016: 19th International Conference on Information Security*, volume 9866 of *Lecture Notes in Computer Science*, pages 408–425. Springer, Heidelberg, September 2016. doi:10.1007/978-3-319-45871-7_24.

[Tom19] Junichi Tomida. Tightly secure inner product functional encryption: Multi-input and function-hiding constructions. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology – ASIACRYPT 2019, Part III*, volume 11923 of *Lecture Notes in Computer Science*, pages 459–488. Springer, Heidelberg, December 2019. doi:10.1007/978-3-030-34618-8_16.

[Tom23] Junichi Tomida. Unbounded quadratic functional encryption and more from pairings. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023, Part III*, volume 14006 of *Lecture Notes in Computer Science*, pages 543–572. Springer, Heidelberg, April 2023. doi:10.1007/978-3-031-30620-4_18.

[TT18] Junichi Tomida and Katsuyuki Takashima. Unbounded inner product functional encryption from bilinear maps. In Thomas Peyrin and Steven Galbraith, editors, *Advances in Cryptology – ASIACRYPT 2018, Part II*, volume 11273 of *Lecture Notes in Computer Science*, pages 609–639. Springer, Heidelberg, December 2018. doi:10.1007/978-3-030-03329-3_21.

[Wat09] Brent Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 619–636. Springer, Heidelberg, August 2009. doi:10.1007/978-3-642-03356-8_36.

[Wat11] Brent Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *PKC 2011: 14th International Conference on Theory and Practice of Public Key Cryptography*, volume 6571 of *Lecture Notes in Computer Science*, pages 53–70. Springer, Heidelberg, March 2011. doi:10.1007/978-3-642-19379-8_4.

[Wat12] Brent Waters. Functional encryption for regular languages. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 218–235. Springer, Heidelberg, August 2012. doi:10.1007/978-3-642-32009-5_14.

[Wee14] Hoeteck Wee. Dual system encryption via predicate encodings. In Yehuda Lindell, editor, *TCC 2014: 11th Theory of Cryptography Conference*, volume 8349 of *Lecture Notes in Computer Science*, pages 616–637. Springer, Heidelberg, February 2014. doi:10.1007/978-3-642-54242-8_26.

[Wee17] Hoeteck Wee. Attribute-hiding predicate encryption in bilinear groups, revisited. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017: 15th Theory of Cryptography Conference, Part I*, volume 10677 of *Lecture Notes in Computer Science*, pages 206–233. Springer, Heidelberg, November 2017. doi:10.1007/978-3-319-70500-2_8.

[Wee20] Hoeteck Wee. Functional encryption for quadratic functions from k-lin, revisited. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020: 18th Theory of Cryptography Conference, Part I*, volume 12550 of *Lecture Notes in Computer Science*, pages 210–228. Springer, Heidelberg, November 2020. doi:10.1007/978-3-030-64375-1_8.

[Wee21] Hoeteck Wee. Broadcast encryption with size $N^{1/3}$ and more from k-lin. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part IV*, volume 12828 of *Lecture Notes in Computer Science*, pages 155–178, Virtual Event, August 2021. Springer, Heidelberg. doi:10.1007/978-3-030-84259-8_6.

[Wee22] Hoeteck Wee. Optimal broadcast encryption and CP-ABE from evasive lattice assumptions. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology – EUROCRYPT 2022, Part II*, volume 13276 of *Lecture Notes in Computer Science*, pages 217–241. Springer, Heidelberg, May / June 2022. doi:10.1007/978-3-031-07085-3_8.

[WFL19] Zhedong Wang, Xiong Fan, and Feng-Hao Liu. FE for inner products and its application to decentralized ABE. 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 97–127. Springer, Heidelberg, April 2019. doi:10.1007/978-3-030-17259-6_4.

[Zip79] Richard Zippel. Probabilistic algorithms for sparse polynomials. In Edward W. Ng, editor, *Symbolic and Algebraic Computation, EUROSAM '79, An International Symposiumon Symbolic and Algebraic Computation, Marseille, France, June 1979, Proceedings*, volume 72 of *Lecture Notes in Computer Science*, pages 216–226. Springer, 1979. doi:10.1007/3-540-09519-5_73.

[ZZGQ23] Ziqi Zhu, Kai Zhang, Junqing Gong, and Haifeng Qian. Registered abe via predicate encodings. Springer-Verlag, 2023.