A Framework for Witness Encryption from Linearly Verifiable SNARKs and Applications

Sanjam Garg[1], Mohammad Hajiabadi[2], Dimitris Kolonelos[1], Abhiram Kothapalli[1], and Guru-Vamsi Policharla[1]

[1]UC Berkeley, {sanjamg,dimitris.kolonelos,akothapalli,guruvamsip}@berkeley.edu
[2]University of Waterloo, mdhajiabadi@uwaterloo.ca

Abstract

Witness Encryption (WE) is a powerful cryptographic primitive, enabling applications that would otherwise appear infeasible. While general-purpose WE requires strong cryptographic assumptions, and is highly inefficient, recent works have demonstrated that it is possible to design *special-purpose* WE schemes for targeted applications that can be built from weaker assumptions and can also be concretely efficient. Despite the plethora of constructions in the literature that (implicitly) use witness encryption schemes, there has been no systematic study of special purpose witness encryption schemes.

In this work we make progress towards this goal by designing a modular and extensible *framework*, which allows us to better understand existing schemes and further enables us to construct new witness encryption schemes. The framework is designed around simple but powerful building blocks that we refer to as "gadgets". Gadgets can be thought of as witness encryption schemes for small targeted relations (induced by linearly verifiable arguments) but they can be composed with each other to build larger, more expressive relations that are useful in applications. To highlight the power of our framework we methodically recover past results, improve upon them and even provide new feasibility results.

The first application of our framework is a Registered Attribute-Based Encryption Scheme [Hohenberger et al. (Eurocrypt 23)] with linear sized common reference string (CRS). Numerous Registered Attribute-Based Encryption (R-ABE) constructions have introduced though a black-box R-ABE construction with a linear–in the number of users–CRS has been a persistent open problem, with the state-of-the-art concretely being $\approx N^{1.58}$ (Garg et al. [GLWW, CRYPTO 24]). Empowered by our Witness Encryption framework we provide the first construction of black-box R-ABE with linear-sized CRS. Our construction is based on a novel realization of encryption for DNF formulas that leverages encryption for set membership.

Our second application is a feasibility result for Registered Threshold Encryption (RTE) with succinct ciphertexts. RTE (Branco et al. [ASIACRYPT 2024] is an analogue of the recently introduced Silent Threshold Encryption (Garg et al. [GKPW, CRYPTO 24]) in the Registered Setting. We revisit Registered Threshold Encryption and provide an efficient construction, with constant-sized encryption key and ciphertexts, that makes use of our WE framework.

Contents

1 Introduction

A public-key encryption scheme allows us to encrypt a message to a *public key* such that a receiver in possession of the corresponding secret key can learn the message. Witness Encryption [GGSW13] generalizes this notion by allowing us to encrypt a message to a statement u in some NP relation $\mathcal{R}$. Any receiver in possession of a valid witness w of the statement u can learn the message. Extractable Witness Encryption [GKP$^+$13] further guarantees that for any receiver that breaks semantic security of ciphertexts encrypted to a statement u, there exists an efficient extractor that can extract a valid witness w of u.

Witness Encryption (and its variations) for all NP relations is a very powerful primitive that can be used to build a wide range of cryptographic primitives including public-key encryption, identity-based encryption, attribute-based encryption (ABE) for circuits [GGSW13] and Turing Machines [GKP$^+$13], multi-party non-interactive key exchange [Zha16], distributed broadcast encryption [FWW23], registered attribute-based encryption (R-ABE) [FWW23], oblivious transfer (OT) [BGI$^+$17], and laconic arguments [FNV17, BISW18].

Initial works building *general purpose* Witness Encryption relied either on multilinear maps [GGSW13, GLW14] or on general purpose Indistinguishability Obfuscation [BGI$^+$01, GGH$^+$13]. More recently, there has been a line of work [CVW18, Tsa22, VWW22] that leverages lattice based assumptions [Wee22, Tsa22] to give a direct construction of Witness Encryption that is arguably simpler than prior approaches. While promising, these assumptions remain non-standard and in some cases have shown to be insecure (see for example [BÜW24, DJM$^+$25, AMYY25, HJL25]) and the constructions still quite inefficient. Therefore, general purpose Witness Encryption is to-date widely regarded as a heavy cryptographic primitive, both in terms of assumptions and efficiency.

Alongside the above efforts, we have also seen the emergence of *special purpose* Witness Encryption-like schemes that target *specific* relations. The main advantage of these schemes is that they are typically simpler, black-box constructions and often orders of magnitude more efficient – to the point where they can be implemented and deployed in real world systems. Moreover, they can be built from *weaker* assumptions in the sense that they are not known to imply general purpose Witness Encryption.

One of the earliest examples is universal hash proofs or (as widely known) hash proof systems [CS02]. At a high level, hash proof systems allows us to sample a pair of keys $(\mathsf{hk}, \mathsf{hp})$ such that hk can be used to "hash" any statement u to a digest (bit string). We are also guaranteed that given hp, anyone with a valid witness w for u can recover the digest. But if the statement is false, then the digest is statistically indistinguishable from random.[1] While the initial motivation for hash proof systems was to provide a modular framework for building chosen-ciphertext secure encryption schemes [CS98], it has since been extended to pairing-based relations [BC16], and found various applications the most prominent of which in building password authenticated key exchange [GL03, ACP09, KV11, BBC$^+$13, BC16], and oblivious transfer [Kal05, ABB$^+$13, BC16].

Another canonical example is the Boneh-Franklin identity-based encryption (IBE) scheme [BF01] which can be viewed as being built from an (extractable) witness encryption scheme amounting to: *"You can decrypt my ciphertext if and only if you know a signature [BLS01, BLS04] under a public key* pk*, on a message* m*"*,[2] where the public key and message are part of the statement and chosen by the encryptor. To build IBE from such a witness encryption scheme, one can simply set pk to be the master public key of the IBE system and m to be the identity of the user. This in turn implies that signatures on identities are the keys to decrypt messages that a user receives.

In the area of secure multi-party computation, Garg and Srinivasan [GS17] built a witness encryption

[1]Of course, the digest can in turn be used to encrypt messages by extracting a hardcore predicate bit via the Goldreich-Levin technique [GL89].

[2]While not originally formalized as such, the extractability can be proved in the generic group model (GGM) [Sho97, Mau05], in which we can extract a signature from any PPT algorithm that breaks semantic security by simply observing its GGM queries.

scheme where one can encrypt with respect to a commitment $\overline{b}$, and a bit b such that an opening proof of $\overline{b}$ to the bit b is the witness required for decryption. This was a core primitive used to build the first two-round multi-party computation protocol from pairings [GS17]. This also inspired constructions of two-round multi-party computation protocols from oblivious transfer [GS18, BL18]. A few years later, Benhamouda and Lin [BL20], took this one step further such that we can now encrypt to a commitment $\overline{w}$ and any predicate f (in P) such that the witness required for decryption is a proof that the commitment $\overline{w}$ opens to a value w such that $f(w) = 1$. Again, this was a core primitive used in building the first *reusable* two round multi-party computation protocol from pairing based assumptions [BL20].

We refer to the witness encryption schemes discussed above as *first-generation* witness encryption schemes. In the *second generation* of witness encryption-like schemes, we begin to observe a notion of *succinctness* in the statement. In more detail, they construct a mechanism to *compress* a very large statement into a succinct digest such that the encryption algorithm only needs access to this digest, and not the entire statement. The encryption time, ciphertext size, and decryption time are still allowed to grow with the size of the statement. A particularly instructive example is the chameleon hash encryption [CDG+17, DG17b], where a message is encrypted to the *succinct* digest $h = \mathsf{Hash}(u)$ of a statement u, a position $i \in \{1, \ldots, |u|\}$, and a bit b. This comes with the guarantee that (1) the message can be recovered using the pre-image u if the i-th bit of the statement $u_i = b$ and (2) the ciphertext reveals no information about the message (to a computationally bounded adversary) if $u_i \neq b$. This seemingly simple primitive kicked off an entire area of *laconic* cryptography [CDG+17, GOS18, QWW18, DGGM19, DGI+19, PCFT20, ABD+21, ALOS22, Ros22, HLL23, DKL+23, FHAS24, DGM23, GHMM24, DHMW24, DHM+24, BDHL24] which has many interesting applications in the field of secure computation. It also led to breakthrough results including the first constructions of identity-based encryption from CDH [DG17b, DG17a, BLSV18], and trapdoor functions from CDH [GH18].

Finally, we come to the *third generation* of witness encryption-like schemes that achieve succinctness in *both* the statement, *and* the witness. Let $\mathcal{R}$ be the specific NP relation we are interested in building a witness encryption scheme for. As in the previous generation, a large statement u is first hashed to a succinct digest $h = \mathsf{Hash}(u)$, but now the witness w can also be compressed to a short proof π such that checking whether $(u, w) \in \mathcal{R}$ can be reduced to checking whether $(h, \pi) \in \mathcal{R}'$ for some smaller, *related* relation $\mathcal{R}'$. This allows the witness encryption schemes to be constructed with respect to $\mathcal{R}'$ which in turn leads to succinct ciphertexts and encryption/decryption time. Indeed, this is reminiscent of the pre-processing techniques used in the succinct proof systems literature, and as we will see later, many of these witness encryption schemes borrow techniques from this area.[3,4]

One of the first examples of such a construction dates back to the celebrated work of Boneh et al. [BGW05] who build broadcast encryption using bilinear pairings.[5] Although it was not formalized as such, one can extrapolate a witness encryption scheme in their construction amounting to: *"You can decrypt my ciphertext if and only if you know a secret key* sk*, corresponding to some public key* pk *in the set* $S = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\}$*."* Note that the statement size grows with the size of the set S, and naively using "first generation" techniques would result in a ciphertext size that grows with the statement. But the key insight in [BGW05] is that by exploiting the structure of the public keys, one can *accumulate* S into a constant sized digest $h = \mathsf{Hash}(S)$. Furthermore, one can compute a proof π to prove that some public key pk was accumulated into h, and this proof can be verified in constant time. This allows the witness encryption scheme to be

[3]We note that similar characteristics were achieved by bootstrapping hash encryption with garbled circuits in [CDG+17, DG17b], but this was a non-blackbox technique that incurred a large overhead due to cryptographic primitives being evaluated the garbled circuits. The schemes we refer to as the "third generation" are fully black box and aim to be concretely efficient.

[4]Due to the *compression* of the relation, it is typically the case that all statements are true and hence witness encryption (without extractability) does not offer any meaningful guarantees.

[5]In broadcast encryption, we can encrypt a message to any subset S of users in the system with ciphertext size *sub-linear* in $|S|$.

constructed with respect to a new relation that amounts to: *"You can decrypt my ciphertext if and only if you know a secret key* sk*, corresponding to some public key* pk*, and a proof* π *that* pk *was accumulated into* h*."* Since the new relation can be checked in constant time, it is now possible to build a witness encryption scheme with $O(1)$ encryption/decryption time and ciphertext size. Building broadcast encryption given such a witness encryption scheme is now straightforward.

Several recent works, implicitly, use third generation witness encryption schemes to build primitives that were previously only thought to be possible from general purpose witness encryption or $i\mathcal{O}$. These include registration based encryption (RBE) [GKMR23, FKdP23], distributed broadcast encryption (DBE) [WQZD10, KMW23], laconic private set intersection [CDG$^+$17, ALOS22, DKL$^+$23], set membership encryption [GJV23], registered functional-encryption [FFM$^+$23], registered attribute-based encryption (R-ABE) [HLWW23, ZZGQ23, GLWW24, AT24], batched-threshold encryption [CGPP24, CGPW24, BFOQ24, AFP24], and silent-threshold encryption [GKPW24]. One step further, the recent works of Campanelli et al. [CFK24] and Fleischhacker et al. [FHAS24] explicitly construct special purpose witness encryption schemes for their applications.

Despite these techniques being very widespread in the literature, there has been no systematic study of the techniques used to build these schemes. Even though there is an underlying common theme of first reducing the relation of interest to a *smaller* relation, and then building a witness encryption scheme with respect to the smaller relation, most of the above works do not present their constructions in this light and there is no formal framework to guide the construction of new schemes.

We note that some of the works [BC16, BL20, CFK24, GKPW24] do make the observation that when relations can be checked using pairing product equations with a *linear verifier* (see for instance [GKPW24, Section 2.1]), then one can build a witness encryption scheme for the same relation. But this characterization falls short on two key points:

- First, it is not useful by itself to build new witness encryption schemes. It specifies a sufficient condition to compile the verifier of a relation into a witness encryption, but it offers little help in constructing the verifier in the first place.
- Second, building third generation witness encryption schemes requires an intricate understanding of the specialized area of succinct proof systems. This is key in order to guarantee soundness after compressing the relation.

1.1 Our Contributions

Given the plethora of relations for which the community has built special purpose witness encryption schemes, it is highly desirable to have a framework that allows us to not only easily understand existing constructions, but also to provide a systematic way of designing new witness encryption schemes.

A Witness Encryption Framework. To overcome the above shortcomings, we build an *extensible* and *modular* framework for witness encryption. The core building blocks of the framework are what we refer to as "gadgets" which are *simple* but powerful relations. These gadgets can be *composed* with each other to build a larger, more expressive relation of interest, and our framework automatically constructs a witness encryption scheme, where the succinctness is fully captured within the gadgets. Importantly, using this framework to build third generation witness encryption schemes does not need expertise in succinct proof systems. This is similar to the architecture in modern zkSNARK implementations, where ZK experts implement gadgets for proving commonly used primitives like hash functions or signature verification, and non-expert developers can then use these gadgets in a black-box fashion to build their applications. We demonstrate the power of our framework by methodically recovering previously known results, improving upon them, and providing new feasibility results for advanced encryption schemes.

Registered Attribute-Based Encryption with a linear CRS. Attribute-based encryption (ABE) generalizes public-key encryption and enables fine-grained control to encrypted data but there is a central trusted authority that issues decryption keys to users corresponding to their attributes. [HLWW23] introduced registered ABE (R-ABE) which removes the need for a trusted authority, and instead users generate their own keys and register them with a key curator, who is only trusted for integrity and can be audited. The initial pairing-based R-ABE schemes [HLWW23, ZZGQ23] required a common reference string (CRS) of size $O(|\mathbb{U}|M^2)$, for M users in the system with $\mathbb{U}$ attribute space. The work of Garg et al. [GLWW24] uses a combinatorial technique to reduce the size of the CRS, and was able to bring it down asymptotically to almost linear $O(M^{1+o(1)})$. However, the concrete constants in the exponent turn out to be high; as a matter of fact for $M = 100,000$ users the CRS has a size of $\approx M^{1.58}$. To date [GLWW24] is the state-of-the-art in CRS size of black-box constructions of Ciphertext-Policy R-ABE.[6]

On the other hand, other pairing-based advanced encryption schemes in the registered setting [GKMR23, FKdP23, KMW23, GKPW24] only needing a linear-sized CRS. It has been an intriguing open problem in the area to bring down the CRS of R-ABE to $O(M)$, with both a theoretical and practical interest.

In this work, we close this gap by showing a construction of a registered ABE scheme for DNF ciphertext-policies, that for M users only requires an $O(M)$ sized CRS. Equipped with our framework, we were able to use the power of the KZG polynomial commitment [KZG10], that admits only a linear CRS. We compare our scheme with the existing (Ciphertext-Policy) R-ABE schemes in the literature, in Table 1.

Most of the works in the literature on pairing-based R-ABE constructions are focusing on translating “traditional” ABE schemes in the trusted setting to the registered setting. On the contrary, we started from scratch with a new approach using our witness encryption framework as a core tool. We carefully designed the R-ABE relation of the witness encryption and then composed KZG-based witness encryption gadgets to implement it. Our framework provided us both the gadgets and the power of composition in order to build a (special-purpose) witness encryption scheme for the delicate statement of R-ABE. Our technique is purely algebraic and our resulting scheme black-box. Following our framework, our construction is proved secure in the Generic Group Model (GGM) [Sho97, Mau05]. We postpone a more insightful description of our techniques to Section 2.2.

We note that our scheme is for DNF formula policies; Although, less expressive than the policies of prior works (e.g. Monotone Span Programs), it pushes the expressivity of prior decryption policies (e.g. identity policy for RBE, set membership policy for DBE) that could be realized with linear-sized CRS. Using our WE framework to incorporate more general policies while maintaining a linear-sized CRS is an interesting future direction.

Registered Threshold Encryption. Silent threshold encryption [GKPW24] allows us to encrypt a message to an arbitrary set of users given their public keys with a constant-sized ciphertext, such that any threshold t (chosen at the time of encryption) number of parties can *non-interactively* decrypt the ciphertext any PPT adversary corrupting up to $t-1$ parties learns no information about the underlying message. Silent threshold encryption assumes that, similarly to public-key encryption, the encryptor knows the public keys of all users.

Branco et al. [BLM$^+$24] introduced the notion of Registered Threshold Encryption, in which a key curator aggregates the users' public keys into a short master public key. The encryptors by having only the concise master public key can generate a threshold ciphertext for any arbitrary set of users. However, unlike Silent Threshold Encryption, the definition of [BLM$^+$24] fixes the threshold t before setup (cannot be chosen during encryption) and the ciphertext is not succinct (has $O(t)$ size). Furthermore, in [BLM$^+$24]

[6] The concurrent work of Zhu et al. [ZZC$^+$25] gives a construction of constant-size CRS for the orthogonal notion of Key-Policy R-ABE from Evasive LWE. However, this assumption already implies general purpose witness encryption and therefore constant-size CRS.

	$\lvert\mathsf{crs}\rvert$	$\lvert\mathsf{mpk}\rvert$	$\lvert\mathsf{pk}\rvert$	$\lvert\mathsf{hsk}\rvert$	$\lvert\mathbb{U}\rvert$	Policy	Setting	Security
[HLWW23, §5]	1	1	1	1	∞	Circuit	iO	Adaptive
[FWW23]	1	$\lvert\mathcal{P}\rvert$	$\lvert\mathcal{P}\rvert$	$\lvert\mathcal{P}\rvert$	∞	Circuit	WE	Static
[HLWW23, §7]	$\lvert\mathbb{U}\rvert M^2$	$\lvert\mathbb{U}\rvert$	M	$\lvert\mathbb{U}\rvert$	$\mathsf{poly}(\lambda)$	MSP	Composite, static	Adaptive
[ZZGQ23]	$\lvert\mathbb{U}\rvert M^2$	$\lvert\mathbb{U}\rvert$	M	$\lvert\mathbb{U}\rvert$	$\mathsf{poly}(\lambda)$	ABP	Prime, static	Adaptive
[AT24]	M^2	$(\max\lvert\mathcal{U}_i\rvert)^2$	M	$(\max\lvert\mathcal{U}_i\rvert)^2$	∞	SP	Prime, static	Adaptive
[GLWW24, §4]	$M^{1+o(1)}$	$\lvert\mathbb{U}\rvert$	M	$\lvert\mathbb{U}\rvert$	$\mathsf{poly}(\lambda)$	MSP	Prime, q-type	Static
[GLWW24, §5]	$\lvert\mathbb{U}\rvert M^{1+o(1)}$	$\lvert\mathbb{U}\rvert$	M	$\lvert\mathbb{U}\rvert$	$\mathsf{poly}(\lambda)$	MSP	Composite, static	Adaptive
Our Scheme	M	$\lvert\mathcal{U}_{\mathsf{eff}}\rvert$	M	$\lvert\mathcal{U}_{\mathsf{eff}}\rvert$	∞	DNF	Prime, GGM	Adaptive

Table 1: Comparison of existing Ciphertext-Policy R-ABE schemes. crs stand for common reference string, mpk for the master public key, pk for the individual users keys, hsk for the helper decryption information and $\mathbb{U}$ for the attribute space. M is an upper bound on the number of users in the system. By $\mathcal{U}_i$ we denote the ith user's attribute set and by $\mathcal{U}_{\mathsf{eff}}$ the set of 'effective attributes' currently registered in the system. 'MSP' stands for monotone span programs, 'ABP' arithmetic branching programs, 'SP' general span programs and 'DNF' DNF formulas. By 'composite'/'prime' we refer the order of the bilinear group, by 'static'/'q-type' the type of the assumption and by 'GGM' the generic group model. For all the schemes $\lvert\mathsf{ct}\rvert = O(\lvert\mathcal{P}\rvert)$ and $\lvert\mathsf{sk}\rvert = O(1)$, thus we omit these metrics from the table.

each user should be attached to a index $i \in \{1, \dots, M\}$ (where M is the number of registered users).

We revisit the definition of Registered Threshold Encryption. Users in the system will first register their public keys and corresponding identities with a key curator who is only trusted for integrity. The key curator will then *aggregate* public keys of users into a succinct *master public key* which can then be used to encrypt messages to any arbitrary set of identities, such that any subset of t parties can non-interactively decrypt the ciphertext. The decrypting identities and the threshold t are chosen during encryption time. Importantly, we will demand constant sized ciphertexts, and that the runtime of the encryptor only grows with the size of the set they encrypt to. Our construction is proved secure in the GGM, and assumes a structured common reference string, as is typical for pairing-based advanced encryption schemes (including [GKPW24]), and grows linearly with the number of users registered in the system.

2 Technical Overview

2.1 Overview of Our Framework

Our framework is centered around the observation that we can construct a witness encryption scheme for a relation $\mathcal{R}$ so long as we can reduce $\mathcal{R}$ to a set of *linearly verifiable* relations $\mathcal{R}_1, \dots, \mathcal{R}_n$, which we refer to as gadget relations. In particular, a relation $\mathcal{R}$ is linearly verifiable if there exists a non-interactive argument of knowledge Π for $\mathcal{R}$ such that the verifier for Π can be expressed as checking a linear system of equations on the proof terms (Definition 5). Moreover, if the arguments $\Pi_1, \dots, \Pi_n$ have succinct proofs (i.e., the proof size is sublinear in the size of the witness and statement) then we get a witness encryption scheme with succinct ciphertexts.

Given the above, if we had a succinct linearly-verifiable argument for all of NP we would immediately get witness encryption for any relation $\mathcal{R}$. However, Groth [Gro16] demonstrates that such arguments are infeasible in the pairing-based setting. Instead, we show that a number of simple but powerful relations such as checking inner-products, BLS signatures, degree-checks and so on have succinct, linearly verifiable proofs. By stitching together this simpler gadgets we design witness encryption schemes for complex relations. If needed, our framework can be extended by additional user-defined gadgets. Theorem 1 summarizes the interface of our framework.

Witness Encryption from Linearly Verifiable SNARKs. The high-level flow of the construction is that a witness for relation $\mathcal{R}$ is reduced to a set of witnesses in linearly verifiable relations $\mathcal{R}_1, \ldots, \mathcal{R}_n$. Using the corresponding proof systems $\Pi_1, \ldots, \Pi_n$ a corresponding set of proofs of knowledge $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_n)$ are produced. We observe that so long as proofs $\pi_1, \ldots, \pi_n$ are linearly verifiable then $\boldsymbol{\pi}$ is linearly verifiable. That is, there exists a matrix A and vector b (dependent on the instances) such that π is satisfying if and only if $A \circ \boldsymbol{\pi} = b$. Then, following prior work, an encryptor encrypts message msg (even without knowledge of π) by first sampling a random vector $\boldsymbol{s}$ and computing the ciphertext $c = (c_1, c_2)$ where $c_1 \leftarrow \boldsymbol{s} \cdot A$, and $c_2 \leftarrow \boldsymbol{s} \cdot b + \mathsf{msg}$. Given a witness w for a prescribed instance in $\mathcal{R}$, a decryptor can first compute a proof $\boldsymbol{\pi}$ of w using $\Pi_1, \ldots, \Pi_n$ and then compute $\mathsf{msg} \leftarrow c_1 \cdot \boldsymbol{\pi} - c_2$.

Challenges. There are two key challenges that the above sketch glosses over: As we will see almost always we will want to check the same witness across different relations in $\mathcal{R}_1, \ldots, \mathcal{R}_n$. A common approach to enforce this is to have different proofs reference the same witness commitment in the instance. In our context, however, we must ensure that we only compose gadgets that work over compatible commitment schemes. A contribution of our framework is designing the appropriate language for specifying when gadgets are intercompatible. Second, some gadgets require additional preprocessed material that are correlated with (but not dependent on) the witness to be efficient. We show how to properly bubble up this requirement for auxiliary material to the interface of the final witness encryption scheme. That is, we provide a variant of witness encryption that is only efficient when additional auxiliary material is provided in a preprocessing step.

Constructing DBE with Our Framework. To demonstrate exactly why these challenges arise, and how we resolve them, we construct a simple witness encryption scheme that implies distributed broadcast encryption (DBE) through our framework. Recall that, for a fixed set of parties with independent public keys, DBE allows an encryptor to compute a ciphertext that can be decrypted by any party while having a size that is independent of the total number of parties.

Formally, consider symmetric pairing friendly groups $(\mathbb{G}, \mathbb{G}_T)$ over scalar field $\mathbb{F}$ with pairing operation $\circ : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. Consider group generator $g \in \mathbb{G}$, public key $\mathsf{pk} = [\mathsf{sk}] \in \mathbb{G}$, and a tag $\mathsf{tg} \in \mathbb{G}$ where $[\mathsf{sk}]$ denotes $\mathsf{sk} \cdot g$. Recall that a BLS signature $\sigma = \mathsf{sk} \cdot \mathsf{tg}$ on tg can be checked by checking

$$g \circ \sigma = \mathsf{tg} \circ \mathsf{pk}_i$$

We define the BLS distributed broadcast encryption relation DBE over public parameter, index, instance, witness tuples signatures as follows.

$$\mathsf{DBE} = \left\{ \begin{array}{l|l} \text{parameters } g, & \\ \text{index } (\mathsf{pk}_1, \ldots, \mathsf{pk}_n), & g \circ \sigma = \mathsf{tg} \circ \mathsf{pk}_i \\ \text{instance } \mathsf{tg}, & \\ \text{witness } (\sigma, i) & \end{array} \right\}.$$

Crucially we split the statement into public parameter, index, and instance pairs to demarcate to our eventual witness encryption scheme and corresponding security definitions that we expect soundness to hold so long as g is sampled honestly, and that the ciphertext should be designed to be short specifically with respect to the index $(\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$.

Our goal now is to reduce DBE into a set of gadget relations. Indeed, our framework comes equipped with gadgets for the indexed inner-product relation (with empty public parameters and instance)

$$\mathsf{IIP} = \left\{ \begin{array}{l|l} \text{parameters } \bot, & \\ \text{index } u, & (u, w, v) \in (\mathbb{G}^n, \mathbb{F}^n, \mathbb{G}), \\ \text{instance } \bot, & u \cdot w = v \\ \text{witness } (w, v) & \end{array} \right\}$$

and the non-zero relation

$$\mathsf{NonZero} = \left\{ \begin{array}{l|c} \text{parameters } \bot, & \\ \text{index } \bot, & \\ \text{instance } \bot, & \|B\| \geqslant 1 \\ \text{witness } B & \end{array} \right\}$$

where $\|B\|$ denotes the number of non-zero entries in B. More formally, this means that we provide linearly-verifiable succinct non-interactive arguments of knowledge for relations *closely related* to IIP and NonZero, which we explain further below. As a minor technicality, we formally realize the NonZero check by utilizing yet another IIP gadget and a degree check gadget.

Next, our framework provides a gadget for the BLS signature relation

$$\mathsf{Sig} = \left\{ \begin{array}{l|c} \text{parameters } g, & \\ \text{index } \bot, & \\ \text{instance } \mathsf{tg}, & g \circ \sigma = \mathsf{tg} \circ \mathsf{pk} \ . \\ \text{witness } (\mathsf{pk}, \sigma) & \end{array} \right\}$$

Essentially, a verifier for Sig can perform a linear check on the proof $\pi \leftarrow (\mathsf{pk}, \sigma)$. While this gadget is technically not succinct, ultimately this will not be an issue as the proof is constant-sized.

At this point, we observe that given a BLS distributed broadcast encryption witness (highlighted) and statement

$$(g, (\mathsf{pk}_1, \ldots, \mathsf{pk}_n), \mathsf{tg}, \ (\sigma, i)\ ) \in \mathsf{DBE}$$

we can first compute the point vector $B_i \in \{0,1\}^n$ that is 0 at all points except index i and reduce to checking the statement and witness pairs

$$\begin{array}{llllll} (\bot, & (\mathsf{pk}_1, \ldots, \mathsf{pk}_n), & \bot, & (B_i, \mathsf{pk}_i)\ ) & \in \mathsf{IIP} \\ (g, & \bot, & \mathsf{tg}, & (\mathsf{pk}_i, \sigma)\ ) & \in \mathsf{Sig} \\ (\bot, & \bot, & \bot, & B_i\ ) & \in \mathsf{NonZero}. \end{array}$$

Arguing the soundness of this reduction falls outside of the realm of our framework, but still requires care. A subtle point that is leveraged is that a cheating decryptor could compute an arbitrary B_i that is not a point vector in $\{0,1\}^n$, however by the aggregability and unforgeability property of BLS, providing a valid signature for some linear combination of the public keys necessarily means that the adversary knows secret keys for all the public keys scaled by non-zero terms. This brings out an even subtler point: Technically, the cheating decryptor can produce a trivially satisfying witness by setting B_i to the zero vector and $\mathsf{pk}_i = 0$. To circumvent this, the final check enforces that B_i has at least one non-zero value.

Ensuring Correlated Witnesses with Verifiable Routing. Seemingly, we have reduced DBE to the product relation $\mathsf{IIP} \times \mathsf{Sig} \times \mathsf{NonZero}$ however a key caveat is that the encryptor must also enforce that the same B_i is checked in the inner-product relation and the non-zero relation and that the same pk_i is checked in the inner-product relation and the signature relation.[7] We use the special notation $\boxed{\mathsf{IIP} \times \mathsf{Sig} \times \mathsf{NonZero}}_{\mathcal{C}}$ which we refer to as the routed product relation to denote a product relation where portions of the aggregate public parameters, indexes, instances, and witness are routed to the constituent relations according to the routing description $\mathcal{C}$.

[7]Formally $((\mathsf{crs}_1, \mathsf{crs}_2), (\mathsf{idx}_1, \mathsf{idx}_2), (u_1, u_2), (w_1, w_2)) \in \mathcal{R}_1 \times \mathcal{R}_2$ if and only if $(\mathsf{crs}_1, \mathsf{idx}_1, u_1, w_1) \in \mathcal{R}_1$ and $(\mathsf{crs}_2, \mathsf{idx}_2, u_2, w_2) \in \mathcal{R}_2$.

To enforce that the same witnesses are checked across multiple relations, we actually provide linearly verifiable arguments for *committed* variants of the original relation where the witness is additionally checked against an additional commitment in the instance. This commitment is replicated across gadgets in the instance to ensure consistency. Given a relation $\mathcal{R}$, we denote the corresponding committed relation with respect to a commitment scheme Com as $[\mathcal{R}]_{\mathsf{Com}}$. For simpler gadgets such as the linearly-verifiable argument for Sig we utilize the identity commitment where the witness is copied into the instance. Concretely, we provide linearly-verifiable arguments for relations $[\mathsf{IIP}]_{(\mathsf{SCS},\mathsf{ID})}$, $[\mathsf{Sig}]_{(\mathsf{ID},\mathsf{ID})}$, and $[\mathsf{NonZero}]_{\mathsf{SCS}}$ where SCS is a structured commitment scheme for vectors (Definition 1) and ID is the identity commitment scheme.

An important consideration now is that gadgets to which the same witnesses are routed must also work with respect to the same commitment schemes. This holds in our particular context because $[\mathsf{IIP}]_{(\mathsf{SCS},\mathsf{ID})}$ and $[\mathsf{NonZero}]_{\mathsf{SCS}}$ both expect SCS commitments to B_i and $[\mathsf{IIP}]_{(\mathsf{SCS},\mathsf{ID})}$ and $[\mathsf{Sig}]_{(\mathsf{ID},\mathsf{ID})}$ both expect the identity commitment to pk_i. In general the intercompatibility of gadgets depends on the user specified routing description $\mathcal{C}$. We define this property as $\mathcal{C}$-intercompatibility (Definition 10) and require that the user route terms to $\mathcal{C}$-intercompatible gadgets to produce a linearly verifiable proof for $\boxed{\mathcal{R}_1 \times \cdots \times \mathcal{R}_n}_{\mathcal{C}}$.

Now to design a linearly verifiable proof for $\boxed{\mathsf{IIP} \times \mathsf{Sig} \times \mathsf{NonZero}}_{\mathcal{C}}$, the aggregate prover first produces commitments $(\overline{B}_i, \mathsf{pk}_i, \sigma)$ to the aggregate witnesses $(B_i, \mathsf{pk}_i, \sigma)$. These commitments are routed to the provers of each of the linearly verifiable arguments alongside the routed public parameters, indices, instances, and witnesses. These provers produce the corresponding linearly verifiable proofs π_{IIP}, π_{Sig}, and π_{NonZero}. The final aggregated proof to be used for decryption is set to be $\boldsymbol{\pi} = (\overline{B}_i, \mathsf{pk}_i, \sigma, \pi_{\mathsf{IIP}}, \pi_{\mathsf{Sig}}, \pi_{\mathsf{NonZero}})$. Critically, the witnesses are not committed at encryption time but rather are generated during the decryption stage.

We must still show that the aggregated proof $\boldsymbol{\pi}$ is still linearly verifiable. This follows from the observation that we can first construct a linear routing matrix (dependent on $\mathcal{C}$) that appropriately duplicates proof terms in $\boldsymbol{\pi}$. Then by the linear verifiability of each of the segments we have that the overall proof is linearly verifiable.

Achieving Efficiency with Correlated Auxiliary Inputs. As discussed, some gadgets require additional preprocessed material that are correlated with the witness to be efficient. This generalizes the notion of "hints" as described by Garg et al. [GJM+24]. Concretely given index $\mathsf{pk} = ([\mathsf{sk}_1], \ldots, [\mathsf{sk}_n])$ the IIP argument is only succinct if the prover, given a structured crs $([\tau], [\tau^2], \ldots, [\tau^n])$, is additionally provided preprocessed auxiliary material $([\mathsf{sk}_i \cdot \tau], [\mathsf{sk}_i \cdot \tau^2], \ldots, [\mathsf{sk}_i \cdot \tau^n])$ for all $i \in \{1, \ldots, n\}$ (we use implicit notation: given a group generator g, $[\mathsf{sk}_1]$ denotes $\mathsf{sk}_1 \cdot g$). In practice, we can imagine each party posting the auxiliary material associated with their secret key ahead of time on a bulletin board. The framework ensures that the auxiliary material required for each of the gadgets gets bubbled up in an intercompatible manner as a requirement for the final witness encryption scheme.

2.2 Our Registered Attribute-Based Encryption

In (Ciphertext-Policy) Registered Attribute Based Encryption [HLWW23], users register in the system with some attributes (e.g. "Student", "Biology"). Then an encryptor encrypts a message with respect to a policy over the attributes (e.g. "Professor" AND "Physics" OR "Dean") so that only the users that satisfy the policy can decrypt.

Basic Syntax. As is usual in the literature, we are working on the Slotted Registered Attribute Encryption model [HLWW23],[8] that goes as follows: On input a number of slots-users, a one-time setup is run

[8][HLWW23] showed a generic transformation that boosts any Slotted R-ABE to a fully-fledged R-ABE.

to produce a common reference string $\mathsf{crs} \leftarrow \mathtt{Setup}(1^\lambda, m)$. Then each user samples locally a pair of secret and public keys $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathtt{KGen}(\mathsf{crs}, i)$. Users submit their public keys together with a set of attributes. Afterwards, a deterministic aggregator compresses the m public keys of the users, together with a set of attributes $\mathcal{U}_i$ from each user, into a succinct master public key $(\mathsf{mpk}, \mathsf{hsk}_1, \ldots, \mathsf{hsk}_m) \leftarrow \mathtt{Aggregate}(\mathsf{crs}, (\mathsf{pk}_1, \mathcal{U}_1), \ldots, (\mathsf{pk}_m, \mathcal{U}_m))$. Anyone with the master public key can choose a policy (a boolean function) over the attribute space and a encrypt a message w.r.t. it $\mathsf{ct} \leftarrow \mathtt{Enc}(\mathsf{mpk}, \mathcal{P}, \mathsf{msg})$.[9] Finally, any user $i \in [\![m]\!]$ ($[\![m]\!] = \{1, \ldots, m\}$ in our notation) can decrypt ct if their attributes satisfy the policy of the ciphertext, i.e. $\mathsf{msg} \leftarrow \mathtt{Dec}(\mathsf{sk}, \mathsf{hsk}, \mathsf{ct})$ if $\mathcal{P}(\mathcal{U}_i) = 1$.

Our Starting Observation. Our main observation is that policy satisfaction is closely related to set membership. Assume for example a trivial ciphertext policy of just one attribute $\mathcal{P} = u$. Each slot i can just either possess the attribute or not. Therefore this policy is equivalent to a set membership in the following manner: Let $\mathcal{I}_u$ be the set of indices-users (in $[\![m]\!]$) possessing the attribute u, then

$$\mathcal{P}(\mathcal{U}_i) = 1 \Leftrightarrow i \in \mathcal{I}_u.$$

However, we have evidence from the literature that set membership is a simpler problem to deal with than arbitrary formulas. In fact, unlike R-ABE, we already have pairing-based advanced encryption schemes for set membership policies with linear-sized CRS, which is essentially Distributed Broadcast Encryption (DBE) Schemes [KMW23, GKPW24].[10] Therefore we put forth the following question:

Can we leverage Witness Encryption for Set Membership to build R-ABE for formulas with linear CRS?

AND Clauses as Set Membership tests. Now we model AND policies as a system of membership tests. To ease the presentation, we guide the reader through an example. Assume that we have $m = 8$ users-indices with corresponding attribute sets $\mathcal{U}_1, \ldots, \mathcal{U}_8$ from an unbounded attribute space $u \in \{0,1\}^*$ as follows:

$$\mathcal{U}_1 = \{\mathsf{Crypto}, \mathsf{PL}, \mathsf{CS}\},\ \mathcal{U}_2 = \{\mathsf{Crypto}, \mathsf{PhD}\},\ \mathcal{U}_3 = \{\mathsf{CS}\},\ \mathcal{U}_4 = \{\mathsf{CS}\}$$
$$\mathcal{U}_5 = \{\mathsf{ML}, \mathsf{PL}, \mathsf{Prof}, \mathsf{CS}, \mathsf{PhD}\},\ \mathcal{U}_6 = \{\mathsf{ML}, \mathsf{PhD}\},\ \mathcal{U}_7 = \{\mathsf{CS}\},\ \mathcal{U}_8 = \{\mathsf{ML}, \mathsf{PL}, \mathsf{CS}\}.$$

The attributes of the users form a set that we call the set of effective attributes $\mathcal{U}_{\mathsf{eff}} = \mathcal{U}_1 \cup \ldots \cup \mathcal{U}_8 = \{\mathsf{Crypto}, \mathsf{ML}, \mathsf{PL}, \mathsf{Prof}, \mathsf{CS}, \mathsf{PhD}\}$. We further define a set for each effective attribute as:

$$\begin{aligned} \mathcal{I}_{\mathsf{Crypto}} &= \{1, 2, 5\} \\ \mathcal{I}_{\mathsf{ML}} &= \{5, 6, 8\} \\ \mathcal{I}_{\mathsf{PL}} &= \{1, 5, 8\} \\ \mathcal{I}_{\mathsf{Prof}} &= \{5\} \\ \mathcal{I}_{\mathsf{CS}} &= \{1, 3, 4, 5, 7, 8\} \\ \mathcal{I}_{\mathsf{PhD}} &= \{2, 5, 6\} \end{aligned}$$

Now let the AND clause policy $\mathcal{P} = \mathsf{ML} \wedge \mathsf{CS} \wedge \mathsf{PhD}$. We have that:

$$\mathcal{P}(\mathcal{U}_i) = 1 \Leftrightarrow (i \in \mathcal{I}_{\mathsf{ML}}) \wedge (i \in \mathcal{I}_{\mathsf{CS}}) \wedge (i \in \mathcal{I}_{\mathsf{PhD}})$$

Therefore, in order to encrypt for the policy $\mathcal{P}$ we need an encryption scheme that composes three membership tests.

For our R-ABE the aggregator creates a succinct commitment to each set $\mathcal{I}_u$, for every effective attribute u. These commitments are stored in the master public key.

[9] Typically the policy is part of the ciphertext ct.

[10] It is well known that Broadcast Encryption is virtually ciphertext succinct CP-ABE for the set membership policy [AY20, BV22, Wee22]

Witness Encryption for Set Membership over Multiple Sets. Earlier we saw that, building on our framework, we can get a Distributed Broadcast Encryption, which is essentially a set-membership. We recall that this consisted of

$$\begin{array}{lllll}(\bot, & (\mathsf{pk}_1, \ldots, \mathsf{pk}_m), & \bot, & (B_i, \mathsf{pk}_i)\,) & \in \mathsf{IIP} \\ (g, & \bot, & \mathsf{tg}, & (\mathsf{pk}_i, \sigma)\,) & \in \mathsf{Sig} \\ (\bot, & \bot, & \bot, & B_i\,) & \in \mathsf{NonZero}.\end{array}$$

This shows that anyone in the set $1, \ldots, m$ can decrypt the ciphertext. For a set $\mathcal{I}$ we add the constraint that B_i should be zero in $[\![m]\!] \setminus \mathcal{I}$. For this we build an additional Witness Encryption gadget Zero.

Then a ciphertext for $\mathcal{P} = \mathsf{ML} \wedge \mathsf{CS} \wedge \mathsf{PhD}$ is witness encryption for

$$\begin{array}{lllll}(\bot, & (\mathsf{pk}_1, \ldots, \mathsf{pk}_m), & \bot, & (B_i, \mathsf{pk}_i)\,) & \in \mathsf{IIP} \\ (g, & \bot, & \mathsf{tg}, & (\mathsf{pk}_i, \sigma)\,) & \in \mathsf{Sig} \\ (\bot, & \bot, & \bot, & B_i\,) & \in \mathsf{NonZero}. \\ (\bot, & [\![m]\!] \setminus \mathcal{I}_{\mathsf{ML}}, & \bot, & B_i\,) & \in \mathsf{Zero}. \\ (\bot, & [\![m]\!] \setminus \mathcal{I}_{\mathsf{CS}}, & \bot, & B_i\,) & \in \mathsf{Zero}. \\ (\bot, & [\![m]\!] \setminus \mathcal{I}_{\mathsf{PhD}}, & \bot, & B_i\,) & \in \mathsf{Zero}.\end{array}$$

Since the gadgets for all these relations are based on the KZG polynomial commitment they are work over the same CRS, of size $O(m)$. Therefore, the overall size of the CRS is linear and, notably, also independent from the size of effective attribute set.

General DNF Formulas. Until now we have shown how to encrypt for AND clauses. We add negation in a straightforward way, adding a commitment of the complementary sets in the master public key, for each effective attribute. That is, now we have both $\mathcal{I}_u$ and $[\![m]\!] \setminus \mathcal{I}_u$, and $\neg u$ translates to membership in $[\![m]\!] \setminus \mathcal{I}_u$. Therefore, with this we can encrypt for general AND clauses (that also include negations).

To support general DNF formulas (OR of AND clauses) we are left with resolving the OR gates. For this we simply give a different ciphertext for each term of OR. In more detail, let the DNF policy

$$\mathcal{P} := (\mathsf{ML} \wedge \mathsf{CS} \wedge \mathsf{PhD}) \vee (\neg\mathsf{Prof}) \vee (\neg\mathsf{Crypto} \wedge \mathsf{PhD} \wedge \mathsf{Prof}) \vee (\mathsf{PL}).$$

For this we compute one ciphertext per AND clause $\mathsf{ML} \wedge \mathsf{CS} \wedge \mathsf{PhD}$, $\neg\mathsf{Prof}$, etc. and the final ciphertext is just a concatenation of 4 ciphertexts. This captures the nature of OR gate; Even being able to decrypt one of the four ciphertexts will allow one to retrieve the message.

2.3 Registered Threshold Encryption

In Registered Threshold Encryption [BLM+24] users that wish to decrypt–namely decryptors–locally sample their secret and public key, $(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathtt{KGen}(\mathsf{crs})$. Then, similarly to Registered Attribute-Based Encryption, 'register' their public key with a deterministic aggregator that comperesses all the public keys into a master public key mpk. The latter should be succinct in the number of decryptors (i.e. much smaller than $(\mathsf{pk}_1, \ldots, \mathsf{pk}_m)$). Anyone can encrypt a message msg for a committee of decryptors $\mathcal{S}$ and a threshold t, $\mathsf{ct} \leftarrow \mathtt{Enc}(\mathsf{mpk}, \mathcal{S}, t, \mathsf{msg})$, so that any subset of $\mathcal{D} \subseteq \mathcal{S}$ can decrypt the ciphertext as long as it is above the threshold t, $|\mathcal{D}| \geq t$. That is, each decryptor using their secret key produces a partial decryption $\sigma \leftarrow \mathtt{PartDec}(\mathsf{sk}, \mathsf{ct})$ and then anyone having at least t valid partial decryptions can retrieve the message $\mathsf{msg} \leftarrow \mathtt{DecAggr}(\mathsf{crs}, \mathsf{aux}, \mathcal{S}, \{\sigma_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{D}}, \mathsf{ct})$. We crucially require that the ciphertext and the partial decryptions are succinct in $\mathcal{S}$ and t.

For our construction, our starting point is the Silent Threshold Encryption (STE) construction of Garg et al. [GKPW24]. STE is a closely relevant primitive that works similarly to RTE with the distinctive point that the encryption key may not be concise. In fact, in the STE construction of [GKPW24] the encryptor holds all the public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_m$. In order to achieve RTE we should be able to 'compress' the encryptor's information.

We achieve this as follows. In [GKPW24] the encryptor needs to know all the individual public keys $(\mathsf{pk}_1, \ldots, \mathsf{pk}_m)$ and then form a concise commitment of the vector consisting only of the public keys in the desired set $\mathcal{S}$, $(\mathsf{pk}_i)_{i\in\mathcal{S}}$. The ciphertext of [GKPW24] (implicitly) contains an IIP gadget for an inner product between the vector $(\mathsf{pk}_{i_1}, \ldots, \mathsf{pk}_{i_{|\mathcal{S}|}})$ and a 'selector' vector B representing the parties that provided a partial decryption; B is computed during decryption accroding to $\mathcal{D}$. To overcome this, we observe that one can hold from the beginning a concise commitment to the vector of *all keys* and then ensure that only valid keys (i.e. keys in $\mathcal{S}$) are allowed to participate in the decryption, in other words, that only parties in the set $\mathcal{S}$ can be part of the selector vector B. We enforce this condition with the Zero gadget making sure that all positions $j \notin \mathcal{S}$ of B are zero. Then the concise vector is computed by the aggregator and plays the role of the master public key.

3 Preliminaries

Notation. With λ we will denote the security parameter, with negl a negligible function, (i.e. $\mathsf{negl}(x) < 1/f(x)$ for *any* polynomial f), and with poly a polynomial function in its variables (they can be more than 1). For events A and B, we let $\Pr[A] \approx \Pr[B]$ denote that $|\Pr[A] - \Pr[B]| = \mathsf{negl}(\lambda)$. For brevity, we use $[\![n]\!]$ for the set $\{1, \ldots, n\}$, where $n > 0$ is a positive integer. We denote sets with calligraphic capital letters, e.g. $\mathcal{S}, \mathcal{U}$. Given a tuple U and a set $\mathcal{S}$ we let $U_{\mathcal{S}}$ denote $\{U_i\}_{i\in\mathcal{S}}$. Vectors will be written in small bold font, e.g. $\boldsymbol{x} = (x_1, \ldots, x_n)$ and matrices in capital bold, e.g. $\boldsymbol{A} = (a_{ij})_{i,j}$. For functions f and g, we define function (f, g) as $(f, g)(x_1, x_2) = (f(x_1), g(x_2))$. We let $\hom(U, V)$ denote the set of linear transformations from vector space U to vector space V.

Bilinear Groups. A Bilinear Group $\mathcal{BG}$, generated as $(\mathbb{F}, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e) \leftarrow \mathcal{BG}(1^\lambda)$, is specified by a field $\mathbb{F}$ of prime order $p = 2^{\Theta(\lambda)}$, three groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ (the first two we call "source groups" and the third "target group"), a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ that we call "pairing" and one random generator g_1, g_2 for each group. We use the *implicit notation*, i.e., $[x]_s := x \cdot g_s$ and more generally $[\boldsymbol{A}]_s$ represents a matrix of the corresponding group elements, for $s \in \{1, 2, T\}$. Also, we denote the group operation additively, $[x]_s + [y]_s = [x + y]_s$, for $s \in \{1, 2, t\}$. By $[x]_1 \circ [y]_2 = [y]_2 \circ [x]_1 = [x \cdot y]_T$, we denote the pairing $e([x]_1, [y]_2)$. We note that the way it is defined $\circ$ is commutative, for instance $([a]_1, [b]_2)^\intercal \circ ([c]_2, [d]_1)$ is well-defined and gives the outcome $[ac + bd]_T$. In our constructions, we will sometimes omit writing explicitly the Bilinear Group in the algorithms' inputs, even though it is implicitly taken as input.

3.1 Polynomials

Throughout the paper, we make use of univariate polynomials over the $\mathbb{F}_p$, defined by the bilinear group. In more detail, we rely on the Lagrange interpolation form: Let $\mathcal{S} = \{x_1, \ldots, x_M\} \subseteq \mathbb{F}$ be a set of field points, we denote by $L_1^{\mathcal{S}}(X), L_2^{\mathcal{S}}(X), \ldots, L_M^{\mathcal{S}}(X)$ the Lagrange basis polynomials over S, where:

$$L_i^{\mathcal{S}}(X) := \prod_{j\in 1, j\neq i}^{m} \frac{X - \mathcal{S}[j]}{\mathcal{S}[i] - \mathcal{S}[j]} = \prod_{j\in 1, j\neq i}^{m} \frac{X - x_j}{x_i - x_j}.$$

Each L_i^S has the property that $L_i^S(x_i) = 1$ and $L_i^S(x_j) = 0$ for each $j \in \mathcal{S} \setminus \{i\}$. The polynomial $f(X) = y_1 L_1^S(X) + \ldots + y_M L_M^S(X)$ interpolates the points $\{(x_1, y_1), \ldots, (x_M, y_M)\}$. In this work will mostly focus on two types of sets S, either the set of M-th roots of unity: $\Omega = \{\omega^1, \ldots, \omega^M\}$ or arbitrary subsets $\mathcal{U} \subseteq \{0,1\}^{2\lambda}$.

The vanishing polynomial over a set $\mathcal{V} = \{v_1, \ldots, v_k\} \subseteq \mathbb{F}$ is defined as:

$$Z_{\mathcal{V}}(X) := \prod_{i=1}^{k}(X - \mathcal{V}[i]) = \prod_{i=1}^{k}(X - v_i).$$

We recall a useful property, if $f(X)$ is divisible by $Z_{\mathcal{V}}(X)$, i.e. there is a quotient polynomial $Q(X)$ such that $f(X)/Z_{\mathcal{V}}(X) = Q(X)$, then $f(v_1) = \ldots = f(v_k) = 0$.

Generalized Univariate Sumcheck. [RZ21] constructed a generalized variant of the univariate sumcheck [BCR+19] that works over arbitrary domains. We recall their construction but generalize to support inner product arguments instead of just a sumcheck.

Lemma 1. *Let $\mathcal{D} = \{d_1, \ldots, d_n\} \subset \mathbb{F}$ be an arbitrary domain of size n and $Z_{\mathcal{D}}(X)$ be the vanishing polynomial over $\mathcal{D}$. Let $A(X) \in \mathbb{F}[X]$, be a polynomial of arbitrary degree, and $b = (b_1, \ldots, b_n) \in \mathbb{F}^n$. For any $x^* \in \mathbb{F} \setminus \mathcal{D}$, we define the polynomial $B(X) := \sum_{i=1}^{n} b_i \cdot L_i^{\mathcal{D}}(x^*)^{-1} \cdot L_i^{\mathcal{D}}(X)$. We then have that for all $x^* \in \mathbb{F} \setminus \mathcal{D}$, $\sum_{i=1}^{n} b_i \cdot A(d_i) = s$ iff there exist polynomials $Q_Z(X), Q_X(X)$ such that $\deg(Q_X(X)) \leqslant n-2$, and*

$$A(X) \cdot B(X) = s + Q_X(X) \cdot (X - x^*) + Q_Z(X) \cdot Z(X)$$

When $\mathcal{D}$ is a smooth multiplicative domain such as the roots of unity, we can set $x^* = 0$ to recover the univariate sumcheck.

3.2 Generic Group Model

Our security proof is based on the Generic (Bilinear) Group Model [Sho97, Mau05]. A 'generic' adversary does not have concrete representations of the elements of the group and can only use generic group operations. This model captures the possible 'algebraic' attacks that an adversary can perform.

In particular, we follow the Maurer's GGM [Mau05], which is extended to Bilinear Groups by [BBG05]. The adversary in GGM makes oracle queries for each generic group operation she wishes to perform and receives a handle for the resulting group element, instead of the actual element itself. We call the party that answers the queries the 'challenger'. The challenger keeps three lists $\boldsymbol{L}_1, \boldsymbol{L}_2, \boldsymbol{L}_T$ of all group elements resulted from the queries of the adversary together with their corresponding handles.

3.3 Commitment Schemes

Below we define commitment schemes and committed relations.

Definition 1 (Commitment Scheme)**.** *A commitment scheme is defined by polynomial-time algorithm* $\mathtt{CGen} : \mathbb{N}^2 \times \mathsf{td} \to \mathbb{CK}$ *that produces a commitment key given the security parameter and size parameter, a deterministic polynomial-time algorithm* $\mathtt{Commit} : \mathbb{CK} \times \mathbb{M} \times \mathbb{R} \to \mathbb{C}$ *that produces a commitment in* $\mathbb{C}$ *given a commitment key, message, and randomness tuple such that binding holds. That is, for any* PPT *adversary* $\mathcal{A}$*, given* $\mathsf{ck} \leftarrow \mathtt{CGen}(\lambda, n, \mathsf{td})$ *for randomly sampled trapdoor* td*, and given* $((m_1, r_1), (m_2, r_2)) \leftarrow \mathcal{A}(\mathsf{ck})$ *we have that*

$$\Pr[(m_1, r_1) \neq (m_2, r_2) \wedge \mathtt{Commit}(\mathsf{ck}, m_1, r_1) = \mathtt{Commit}(\mathsf{ck}, m_2, r_2)] = \mathsf{negl}(\lambda).$$

The commitment scheme is deterministic if $\mathtt{Commit}$ *does not use its randomness.*

Construction 1 (Structured Commitment Scheme)**.** *Consider group $\mathbb{G}$ over scalar field $\mathbb{F}$. We construct a structured commitment scheme* $\mathsf{SCS}(\mathbb{G})$ *characterized by group $\mathbb{G}$ over message space $\mathbb{F}^n$ and commitment space $\mathbb{G}$ as follows.*

- $\texttt{CGen}_{\mathsf{SCS}}(\lambda, n, \tau \in \mathbb{F})$*: Output* $\mathsf{ck} = ([1], [\tau^1], \ldots, [\tau^{n-1}])$*.*
- $\texttt{Commit}_{\mathsf{SCS}}(\mathsf{ck}, w)$*: Output* $\overline{w} \leftarrow [w_1 \cdot L_1^{\mathcal{D}}(\tau) + \ldots + w_n \cdot L_n^{\mathcal{D}}(\tau)]$*, where* $L_i^{\mathcal{D}}(X)$ *are the Lagrange polynomials corresponding to any domain* $\mathcal{D} \subset \mathbb{F}$ *of size* n*.*

Definition 2 (Committed Relation)**.** *For an indexed relation $\mathcal{R}$ for witnesses in space $\mathbb{W}$ and a commitment scheme over messages in $\mathbb{M} = \mathbb{W}$ we define the committed relation $[\mathcal{R}]$ characterized by commitment scheme* $\mathsf{Com} = (\texttt{CGen}, \texttt{Commit})$ *as follows*

$$[\mathcal{R}]_{\mathsf{Com}} = \left\{ \begin{array}{l|l} \textit{parameters}\ (\mathsf{pp}, \mathsf{ck}), & \\ \textit{index}\ \mathsf{idx}, & (\mathsf{pp}, \mathsf{idx}, u, w) \in \mathcal{R}, \\ \textit{instance}\ \ (\overline{w}, u), & \overline{w} = \texttt{Commit}(\mathsf{ck}, w) \\ \textit{witness}\ \ w & \end{array} \right\}.$$

3.4 Arguments of Knowledge

Below we define arguments of knowledge (and related notions such as linear verifiability) over indexed relations represented as predicates over parameter, index, instance, and witness tuples. We augment the standard notion of arguments of knowledge with an `AuxGen` function to capture the ability to preprocess auxiliary material and the index in a correlated fashion from some secret material. The common reference string generator `Gen` produces both the parameters of the relation pp and additional auxiliary material crs that is dependent on the trapdoor but is purely a proof artifact (used to generate prover and verifier keys) as opposed to material that is checked in the final relation.

To enable composition, we consider a black-box straight-line notion of extractability. Although black-box extraction is in principle incompatible with succinctness in the standard model it is still meaningful in idealized models (e.g., Generic Group Model, or Random Oracle Model), where the extractor is additionally provided the prover's oracle queries. We leave our definitions generic to the particular oracle model, which we later specify in our constructions.

Definition 3 (Non-Interactive Argument of Knowledge)**.** *Consider an indexed relation relations $\mathcal{R}$. An argument of knowledge in oracle model ρ for $\mathcal{R}$ is defined by PPT algorithms* $(\texttt{Gen}, \texttt{Prove}, \texttt{Verify})$ *called the generator, the prover, the verifier respectively and deterministic algorithms* `AuxGen` *and* `Digest`*, all with access to ρ with the following interface.*

- $\texttt{Gen}(\lambda, N, \mathsf{td}) \to (\mathsf{pp}, \mathsf{crs})$*: Takes as input security parameter λ, trapdoor* td*, and size parameters N. Deterministically outputs parameters* pp *and the argument's common reference string* crs*.*
- $\texttt{AuxGen}((\mathsf{pp}, \mathsf{crs}), \mathsf{s}_i) \to (\mathsf{idx}_i, \mathsf{aux}_i)$*: Takes as input parameters* pp*, common reference string* crs*, and (private) value* s_i*. Deterministically produces a portion of the index* idx_i *and (public) auxiliary value* aux_i*.*
- $\texttt{Digest}((\mathsf{pp}, \mathsf{crs}), \mathsf{idx}, \mathsf{aux}) \to (\mathsf{pk}, \mathsf{vk})$*: Takes as input parameters* pp *and common reference string* crs*, index* idx*, and auxiliary values* aux*. Deterministically outputs prover key* pk *and verifier key* vk*.*
- $\texttt{Prove}(\mathsf{pk}, u, w) \to \pi$*: Takes as input a prover key* pk *and an instance-witness pair* (u, w)*. Produces a proof* π*.*

- $\texttt{Verify}(\mathsf{vk}, u, \pi) \to \{0,1\}$*: Takes as input the verifier key* vk *an instance* u*, and a proof* π*. Outputs* 1 *if and only if* π *is valid.*

An argument of knowledge $(\texttt{Gen}, \texttt{AuxGen}, \texttt{Digest}, \texttt{Prove}, \texttt{Verify})$ *with trapdoor space* $\mathcal{T}$ *satisfies the following conditions.*

1. **Completeness***: For all* $(u, w, (\mathsf{s}_1, \ldots, \mathsf{s}_m))$*, given randomly sampled* $\mathsf{td} \in \mathcal{T}$*,* $(\mathsf{pp}, \mathsf{crs}) \leftarrow \texttt{Gen}(\lambda, N, \mathsf{td})$*,* $\{(\mathsf{idx}_i, \mathsf{aux}_i) \leftarrow \texttt{AuxGen}((\mathsf{pp}, \mathsf{crs}), \mathsf{s}_i)\}_{i\in[m]}$*, such that* $(\mathsf{pp}, \mathbf{idx}, u, w) \in \mathcal{R}$ *for* $\mathbf{idx} \leftarrow (\mathsf{idx}_1, \ldots, \mathsf{idx}_m)$*,* $(\mathsf{pk}, \mathsf{vk}) \leftarrow \texttt{Digest}((\mathsf{pp}, \mathsf{crs}), \mathbf{idx}, \mathbf{aux})$ *for* $\mathbf{aux} \leftarrow (\mathsf{aux}_1, \ldots, \mathsf{aux}_m)$*, and* $\pi \leftarrow \texttt{Prove}(\mathsf{pk}, u, w)$ *we have that*
$$\texttt{Verify}(\mathsf{vk}, u, \pi) = 1.$$

2. **Knowledge Soundness***: There exists a probabilistic polynomial-time extractor* $\mathcal{E}$ *such that for any probabilistic polynomial-time adversary* $\texttt{Prove}^*$*, given a randomly sampled trapdoor* $\mathsf{td} \in \mathcal{T}$*,* $(\mathsf{pp}, \mathsf{crs}) \leftarrow \texttt{Gen}(\lambda, N, \mathsf{td})$*,* $((\mathsf{s}_1, \ldots, \mathsf{s}_m), u, \pi, \mathsf{tr}) \leftarrow \texttt{Prove}^*(\mathsf{pp}, \mathsf{crs})$*,* $\{(\mathsf{idx}_i, \mathsf{aux}_i) \leftarrow \texttt{AuxGen}((\mathsf{pp}, \mathsf{crs}), \mathsf{s}_i)\}_{i\in[m]}$ *and* $(\mathsf{pk}, \mathsf{vk}) \leftarrow \texttt{Digest}((\mathsf{pp}, \mathsf{crs}), \mathbf{idx}, \mathbf{aux})$ *we have that*
$$\Pr[(\mathsf{pp}, \mathbf{idx}, u, \mathcal{E}((\mathsf{pp}, \mathsf{crs}), \mathbf{idx}, \mathbf{aux}, u, \pi, \mathsf{tr})) \in \mathcal{R} \mid \texttt{Verify}(\mathsf{vk}, u, \pi) = 1] \approx 1$$
where $\mathbf{idx} \leftarrow (\mathsf{idx}_1, \ldots, \mathsf{idx}_m)$*,* $\mathbf{aux} \leftarrow (\mathsf{aux}_1, \ldots, \mathsf{aux}_m)$*, and* tr *is the list of oracle queries and responses made by* $\texttt{Prove}^*$*.*

Definition 4 (Succinctness)**.** *An argument of knowledge is succinct if the proof is sublinear in the size of the index and witness.*

Definition 5 (Linear Verifiability)**.** *A non-interactive argument of knowledge* Π *for a committed relation* $\mathcal{R}$ *is linearly verifiable over* $\hom(W, V)$ *if there exists some linear transformation* $A_{\mathsf{vk},u} \in \hom(W, V)$ *and* $b_{\mathsf{vk},u} \in V$ *such that*
$$\Pi.\texttt{Verify}(\mathsf{vk}, (\overline{w}, u), \pi) = 1 \Leftrightarrow A_{\mathsf{vk},u}(\overline{w}, \pi) = b_{\mathsf{vk},u}.$$
Likewise, a committed relation $\mathcal{R}$ *is linearly verifiable over* $\hom(W, V)$ *if there exists an argument of knowledge* Π *that is linearly verifiable over* $\hom(W, V)$ *for* $\mathcal{R}$*.*

Definition 6 (Reduction)**.** *For relations* $\mathcal{R}_1$ *and* $\mathcal{R}_2$ *in* NP*, a reduction* Π *from* $\mathcal{R}_1$ *to* $\mathcal{R}_2$ *(denoted* $\Pi : \mathcal{R}_1 \to \mathcal{R}_2$*) consists of two polynomial-time, deterministic, invertible functions, the instance map* φ_{inst} *and the witness map* φ_{wit}*, such that for* $(u_2, w_2) \leftarrow \varphi_{\mathsf{wit}}(\mathsf{pp}, \mathsf{idx}, u_1, w_1)$ *and* $u_2' \leftarrow \varphi_{\mathsf{inst}}(u_1)$ *we have that* $u_2 = u_2'$ *and* $(\mathsf{pp}, \mathsf{idx}, u_2, w_2) \in \mathcal{R}_2$ *if and only if* $(\mathsf{pp}, \mathsf{idx}, u_1, w_1) \in \mathcal{R}_1$*.*

4 A Framework for Witness Encryption

In this section, we formally construct our framework for witness encryption, which given a reduction from a relation $\mathcal{R}$ to intercompatible gadget relations $\mathcal{R}_1, \ldots, \mathcal{R}_n$ with succinct linear verification (Definition 5) produces a succinct witness encryption scheme for $\mathcal{R}$.

We begin by formally defining witness encryption augmented with auxiliary material correlated with satisfying witnesses. As with the definition of arguments of knowledge, we consider a black-box straight-line notion of extractability with respect to a generic oracle.

Definition 7 (Witness Encryption)**.** *A witness encryption scheme in oracle model* ρ *for an indexed relation* $\mathcal{R}$ *consists of the following algorithms with access to* ρ*.*

- $\texttt{Gen}(\lambda, N, \mathsf{td}) \to (\mathsf{pp}, \mathsf{crs})$*: Takes as input a security parameter* λ*, length parameter* N*, and trapdoor* td*. Deterministically outputs the (structured) public parameters* pp *and common reference string* crs*.*

- $\texttt{AuxGen}((\mathsf{pp}, \mathsf{crs}), \mathsf{s}_i) \to (\mathsf{idx}_i, \mathsf{aux}_i)$*: Takes as input public parameters* pp *and common reference string* crs *and (private) value* s_i*. Deterministically produces a portion of the index* idx_i *and (public) auxiliary value* aux_i*.*

- $\texttt{Digest}((\mathsf{pp}, \mathsf{crs}), \mathsf{idx}, \mathsf{aux}) \to (\mathsf{ed}, \mathsf{dd})$*: Takes as input public parameters* pp*, and common reference string* crs*, index* idx*, and auxiliary values* aux*. Deterministically produces a public encryption digest* ed *and decryption digest* dd*.*

- $\texttt{Enc}(\mathsf{ed}, u, \mathsf{msg}) \to \mathsf{ct}$*: Takes as input an encryption digest* ed*, instance* u*, and message* msg*. Outputs a ciphertext* ct*.*

- $\texttt{Prove}(\mathsf{dd}, u, w) \to \pi$*: Takes as input a decryption digest* dd*, instance* u*, and a witness* w*. Produces a proof* π*.*

- $\texttt{Dec}(\mathsf{ct}, \pi) \to \mathsf{msg}$*: Takes as input a ciphertext* ct*, and a proof* π*. Outputs* msg *if* π *is valid.*

A witness encryption scheme $(\texttt{Gen}, \texttt{AuxGen}, \texttt{Digest}, \texttt{Enc}, \texttt{Prove}, \texttt{Dec})$ *over trapdoor space* $\mathcal{T}$ *for relation* $\mathcal{R}$ *satisfies the following conditions.*

1. **Perfect Correctness***: For all* (u, w, s)*, given randomly sampled* $\mathsf{td} \in \mathcal{T}$*,* $(\mathsf{pp}, \mathsf{crs}) \leftarrow \texttt{Gen}(\lambda, N, \mathsf{td})$*,* $\{(\mathsf{idx}_i, \mathsf{aux}_i) \leftarrow \texttt{AuxGen}((\mathsf{pp}, \mathsf{crs}), \mathsf{s}_i)\}_{i \in [m]}$ *such that* $(\mathsf{pp}, \mathbf{idx}, u, w) \in \mathcal{R}$ *for* $\mathbf{idx} \leftarrow (\mathsf{idx}_1, \ldots, \mathsf{idx}_m)$*,* $(\mathsf{ed}, \mathsf{dd}) \leftarrow \texttt{Digest}((\mathsf{pp}, \mathsf{crs}), \mathbf{idx}, \mathbf{aux})$ *for* $\mathbf{aux} \leftarrow (\mathsf{aux}_1, \ldots, \mathsf{aux}_m)$*, and* $\pi \leftarrow \texttt{Prove}(\mathsf{dd}, u, w)$ *we have that*
$$\texttt{Dec}(\mathsf{dd}, \texttt{Enc}(\mathsf{ed}, u, \mathsf{msg}), \pi) = \mathsf{msg}.$$

2. **Extractable Security***: There exists a probabilistic polynomial-time extractor* $\mathcal{E}$ *such that for any probabilistic polynomial-time adversary* $\mathcal{A}$*, given a randomly sampled trapdoor* $\mathsf{td} \in \mathcal{T}$*,* $(\mathsf{pp}, \mathsf{crs}) \leftarrow \texttt{Gen}(\lambda, N, \mathsf{td})$*,* $((\mathsf{s}_1, \ldots, \mathsf{s}_m), u, (m_0, m_1), \mathsf{tr}_1) \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{crs})$*,* $\{(\mathsf{idx}_i, \mathsf{aux}_i) \leftarrow \texttt{AuxGen}((\mathsf{pp}, \mathsf{crs}), \mathsf{s}_i)\}_{i \in [m]}$*,* $(\mathsf{ed}, \mathsf{dd}) \leftarrow \texttt{Digest}((\mathsf{pp}, \mathsf{crs}), \mathbf{idx}, \mathbf{aux})$*,* $b \xleftarrow{\$} \{0, 1\}$*,* $c_b \leftarrow \texttt{Enc}(\mathsf{ed}, u, m_b)$*,* $(b', \mathsf{tr}_2) \leftarrow \mathcal{A}(c_b)$*, we have that given*
$$\Pr[b' = b] = \frac{1}{2} + \epsilon$$
we have that
$$\Pr[(\mathsf{pp}, \mathbf{idx}, u, \mathcal{E}((\mathsf{pp}, \mathsf{crs}), \mathbf{idx}, \mathbf{aux}, u, (m_0, m_1), (\mathsf{tr}_1, \mathsf{tr}_2))) \in \mathcal{R}] \geqslant \epsilon - \mathsf{negl}(\lambda).$$
where $\mathbf{idx} \leftarrow (\mathsf{idx}_1, \ldots, \mathsf{idx}_m)$*,* $\mathbf{aux} \leftarrow (\mathsf{aux}_1, \ldots, \mathsf{aux}_m)$*, and* $(\mathsf{tr}_1, \mathsf{tr}_2)$ *are the list of oracle queries and responses made by* $\mathcal{A}$*.*

Definition 8 (Succinctness)**.** *A witness encryption scheme is succinct if the ciphertext size is sublinear in the size of the index and witness.*

Below we formally define the routed product relation, which routes aggregate public parameters, indices, instances and witnesses into a set of sub-relations according to a routing description $\mathcal{C}$.

Definition 9 (Routed Product Relation)**.** *Consider relations $\mathcal{R}_1, \ldots, \mathcal{R}_n$ and consider routing description $\mathcal{C} = (\mathcal{C}_{\mathsf{CRS}}, \mathcal{C}_{\mathsf{IDX}}, \mathcal{C}_{\mathsf{INS}}, \mathcal{C}_{\mathsf{WIT}})$ where $\mathcal{C}_{\mathsf{CRS}}$, $\mathcal{C}_{\mathsf{IDX}}$, $\mathcal{C}_{\mathsf{INS}}$, and $\mathcal{C}_{\mathsf{WIT}}$ are each tuples of sets. We define routed product relation over $\mathcal{R}_1 \times \cdots \times \mathcal{R}_n$ characterized by $\mathcal{C}$ as follows.*

$$\boxed{\mathcal{R}_1 \times \cdots \times \mathcal{R}_n}_{\mathcal{C}} = \left\{ \begin{array}{l|l} \textit{parameters } \mathbf{pp}, & (\mathbf{pp}_{\mathcal{C}_{\mathsf{CRS},1}}, \mathbf{idx}_{\mathcal{C}_{\mathsf{IDX},1}}, \boldsymbol{u}_{\mathcal{C}_{\mathsf{INS},1}}, \boldsymbol{w}_{\mathcal{C}_{\mathsf{WIT},1}}) \in \mathcal{R}_1, \\ \textit{index } \mathbf{idx}, & \vdots \\ \textit{instance } \boldsymbol{u}, & \\ \textit{witness } \boldsymbol{w} & (\mathbf{pp}_{\mathcal{C}_{\mathsf{CRS},n}}, \mathbf{idx}_{\mathcal{C}_{\mathsf{IDX},n}}, \boldsymbol{u}_{\mathcal{C}_{\mathsf{INS},n}}, \boldsymbol{w}_{\mathcal{C}_{\mathsf{WIT},n}}) \in \mathcal{R}_n \end{array} \right\}.$$

We will omit $\mathcal{C}$ when clear from context.

To ensure that the set of gadgets used are intercompatible with respect to a routing description $\mathcal{C}$ (e.g. support the same witness commitments) we define the notion of $\mathcal{C}$-compatibility. Because each gadget separately specifies `Gen` and `AuxGen` algorithms, we must additionally ensure that gadgets have compatible generator algorithms (with respect to $\mathcal{C}$) so that we can sample the crs and auxiliary information *once* with respect to the same trapdoor and secret information and route accordingly.

Definition 10 (Intercompatibility)**.** *For $i \in [\![n]\!]$ we say arguments of knowledge $\Pi_i = (\mathtt{Gen}_i, \mathtt{AuxGen}_i, \mathtt{Digest}_i, \mathtt{Prove}_i, \mathtt{Verify}_i)$ for $[\mathcal{R}_i]_{(\mathtt{CGen}_i, \mathtt{Commit}_i)}$ over trapdoor space $\mathcal{T}_i$ are $\mathcal{C}$-intercompatible for $\mathcal{C} = (\mathcal{C}_{\mathsf{CRS}}, \mathcal{C}_{\mathsf{IDX}}, \mathcal{C}_{\mathsf{INS}}, \mathcal{C}_{\mathsf{WIT}})$ if there exists tuples of functions $\mathtt{Gen}'$ where $\mathtt{Gen}'_i$ is over trapdoor space $\mathcal{T}'_i$, $\mathtt{AuxGen}'$, $\mathtt{CGen}'$ and $\mathtt{Commit}'$ such that $\mathtt{Gen}_i = \mathtt{Gen}'_{\mathcal{C}_{\mathsf{CRS},i}}$, $\mathcal{T}_i = \mathcal{T}'_{\mathcal{C}_{\mathsf{CRS},i}}$, $\mathtt{AuxGen}_i = \mathtt{AuxGen}'_{\mathcal{C}_{\mathsf{IDX},i}}$, $\mathtt{CGen}_i = \mathtt{CGen}'_{\mathcal{C}_{\mathsf{WIT},i}}$ and $\mathtt{Commit}_i = \mathtt{Commit}'_{\mathcal{C}_{\mathsf{WIT},i}}$.*

We now proceed to the actual construction of our framework.

Construction 2 (Witness Encryption from Linearly Verifiable Proofs)**.** *Consider a reduction Π from $\mathcal{R}$ to $\boxed{\mathcal{R}_1 \times \cdots \times \mathcal{R}_n}_{\mathcal{C}}$ for $\mathcal{C} = (\mathcal{C}_{\mathsf{CRS}}, \mathcal{C}_{\mathsf{IDX}}, \mathcal{C}_{\mathsf{INS}}, \mathcal{C}_{\mathsf{WIT}})$ such that for $i \in [\![n]\!]$ there exist a (succinct) non-interactive, argument of knowledge $\Pi_i = (\mathtt{Gen}_i, \mathtt{AuxGen}_i, \mathtt{Digest}, \mathtt{Prove}_i, \mathtt{Verify}_i)$ for $[\mathcal{R}_i]_{(\mathtt{CGen}_i, \mathtt{Commit}_i)}$ over trapdoor space $\mathcal{T}_i$ that is linearly verifiable over $\hom(U_i^{\ell_i}, V^{\ell'_i})$. Consider the n'-tuple of functions $\mathtt{Gen}'$ where $\mathtt{Gen}'_i$ is over trapdoor space $\mathcal{T}'_i$ such that $\mathcal{T}_i = \mathcal{T}'_{\mathcal{C}_{\mathsf{CRS},i}}$ guaranteed by the intercompatibility property. We construct a succinct witness encryption scheme for $\mathcal{R}$ over trapdoor space $\mathcal{T}' = \mathcal{T}_1 \times \cdots \times \mathcal{T}_{n'}$ as follows.*

- $\mathtt{Gen}(\lambda, N, \mathbf{td} \in \mathcal{T}')$: *Consider a tuple of generators $\mathtt{Gen}'$ such that $\mathtt{Gen}_i = \mathtt{Gen}'_{\mathcal{C}_{\mathsf{CRS},i}}$ guaranteed by the intercompatibility property. Compute the public parameters and common reference string*

  $$(\mathsf{pp}'_j, \mathsf{crs}'_j) \leftarrow \mathtt{Gen}'_j(\lambda, N, \mathsf{td}_j)$$

  and output public parameters $\mathbf{pp'}$ and common reference string $\mathbf{crs'}$.

- $\mathtt{AuxGen}((\mathbf{pp'}, \mathbf{crs'}), \mathsf{s}_k)$: *Consider the tuple of auxiliary generators $\mathtt{AuxGen}'$ such that $\mathtt{AuxGen}_i = \mathtt{AuxGen}'_{\mathcal{C}_{\mathsf{IDX},i}}$. Compute*

  $$(\mathsf{idx}'_{k,j}, \mathsf{aux}'_{k,j}) \leftarrow \mathtt{AuxGen}'_j((\mathbf{pp'}, \mathbf{crs'}), \mathsf{s}_k)$$

  and output index $\mathbf{idx'_k}$ and auxiliary index $\mathbf{aux'_k}$.

- $\mathtt{Digest}((\mathbf{pp'}, \mathbf{crs'}), (\mathbf{idx'_1}, \ldots, \mathbf{idx'_m}), (\mathbf{aux'_1}, \ldots, \mathbf{aux'_m}))$: *Compute the prover and verifier keys*

  $$(\mathsf{pk}_i, \mathsf{vk}_i) \leftarrow \mathtt{Digest}_i((\mathbf{pp}'_{\mathcal{C}_{\mathsf{IDX},i}}, \mathbf{crs}'_{\mathcal{C}_{\mathsf{IDX},i}}), (\mathbf{idx'_1}_{\mathcal{C}_{\mathsf{IDX},i}}, \ldots, \mathbf{idx'_m}_{\mathcal{C}_{\mathsf{IDX},i}}), (\mathbf{aux'_1}_{\mathcal{C}_{\mathsf{IDX},i}}, \ldots, \mathbf{aux'_m}_{\mathcal{C}_{\mathsf{IDX},i}}))$$

  for $i \in [\![n]\!]$ and output the encryption digest $\mathbf{vk}$ and the decryption digest $(\mathbf{pk}, \mathbf{crs'}, \mathbf{pp'}, (\mathbf{idx'_1}, \ldots, \mathbf{idx'_m}))$.

- $\texttt{Enc}(\textsf{vk}, u, \textsf{msg})$: *Let $\varphi_{\textsf{inst}}$ be the instance map guaranteed by Π. Compute the reduced instance $\boldsymbol{u'} \leftarrow \varphi_{\textsf{inst}}(u)$ and the corresponding routed instance $u_i = \boldsymbol{u'}_{\mathcal{C}_{\textsf{INS},i}}$ for $i \in [\![n]\!]$. Sample random s and let $\boldsymbol{s} = (s^1, \ldots, s^{\sum_{i \in n} \ell'_i})$. Parse $(\textsf{vk}_1, \ldots, \textsf{vk}_n) \leftarrow \textsf{ed}$. Consider the linear transformation $A_{i,u_i,\textsf{vk}_i}$ and the target $b_{i,u_i,\textsf{vk}_i}$ guaranteed by the linear verifiability property. Additionally, let M_i be the routing matrix such that $M_i \circ (\overline{\boldsymbol{w}}', \boldsymbol{\pi}) = (\overline{\boldsymbol{w}}'_{\mathcal{C}_{\textsf{WIT},i}}, \pi_i)$. Given random oracle H, output $\textsf{ct} \leftarrow (\textsf{ct}_1, \textsf{ct}_2)$ where*

$$\textsf{ct}_1 \leftarrow \boldsymbol{s} \cdot \begin{pmatrix} A_{1,u_1,\textsf{vk}_1} M_1 \\ \vdots \\ A_{n,u_n,\textsf{vk}_n} M_n \end{pmatrix} \qquad \textsf{ct}_2 \leftarrow H\left(\boldsymbol{s} \cdot \begin{pmatrix} b_{1,u_1,\textsf{vk}_1} \\ \vdots \\ b_{n,u_n,\textsf{vk}_n} \end{pmatrix} \right) + \textsf{msg}.$$

- $\texttt{Prove}((\textsf{pk}, \textsf{crs}', \textsf{pp}', (\textsf{idx}'_1, \ldots, \textsf{idx}'_m)), u, w)$: *Let $\varphi_{\textsf{wit}}$ be the witness map guaranteed by Π. Compute the reduced instance-witness pair*

$$(\boldsymbol{u'}, \boldsymbol{w'}) \leftarrow \varphi_{\textsf{wit}}(\textsf{crs}', (\textsf{idx}'_1, \ldots, \textsf{idx}'_m), u, w).$$

Consider the tuple of functions $\texttt{Commit}'$ be such that $\texttt{Commit}_i = \texttt{Commit}'_{\mathcal{C}_{\textsf{WIT},i}}$ guaranteed by the intercompatibility property. Compute the witness commitments

$$\overline{w}'_j \leftarrow \texttt{Commit}'_j(\textsf{crs}', w'_j).$$

Compute the routed instances $u_i \leftarrow u'_{\mathcal{C}_{\textsf{INS},i}}$, witnesses $w_i \leftarrow w'_{\mathcal{C}_{\textsf{WIT},i}}$, and witness commitments $\overline{w}_i \leftarrow \overline{\boldsymbol{w}}'_{\mathcal{C}_{\textsf{WIT},i}}$ for $i \in [\![n]\!]$. Compute the proofs for each reduced relation

$$\pi_i \leftarrow \Pi_i.\texttt{Prove}(\textsf{pk}_i, (\overline{w}_i, u_i), w_i)$$

Compute and output the aggregate proof $(\overline{\boldsymbol{w}}', \boldsymbol{\pi})$.

- $\texttt{Dec}(\textsf{ct}, (\overline{\boldsymbol{w}}', \boldsymbol{\pi}))$: *Parse $(\textsf{ct}_1, \textsf{ct}_2) \leftarrow \textsf{ct}$. For random oracle H, output the decrypted message $\textsf{msg} \leftarrow H(\textsf{ct}_1 \circ (\overline{\boldsymbol{w}}', \boldsymbol{\pi})) - \textsf{ct}_2$.*

Theorem 1 (Witness Encryption from Linearly Verifiable Proofs)**.** *Consider pairing-friendly groups $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$. Given a reduction from $\mathcal{R}$ to relation $\boxed{\mathcal{R}_1 \times \cdots \times \mathcal{R}_n}_{\mathcal{C}}$, such that there exists $\mathcal{C}$-intercompatible (succinct) non-interactive arguments of knowledge Π_i for $[\mathcal{R}_i]_{\textsf{Com}_i}$ that are linearly verifiable over $\hom(U_i^{\ell_i}, \mathbb{G}_t^{\ell'_i})$ for $U_i \in \{\mathbb{G}_1, \mathbb{G}_2, \mathbb{F}\}$ and $\ell_i, \ell'_i \in \mathbb{N}$, Construction 2 produces a (succinct) witness encryption scheme for $\mathcal{R}$ in the GGM and ROM.*

Proof. Correctness follows by inspection. We prove extractable security in the generic group model. Consider arbitrary probabilistic polynomial-time adversary $\mathcal{A}$. Suppose that for randomly sampled trapdoor $\textsf{td} \in \mathcal{T}' = \mathcal{T}'_1 \times \cdots \times \mathcal{T}'_{n'}$, $(\textsf{pp}', \textsf{crs}') \leftarrow \texttt{Gen}(\lambda, N, \textsf{td})$, $(\textsf{s}, u, (m_0, m_1), \textsf{tr}_1) \leftarrow \mathcal{A}(\textsf{pp}', \textsf{crs}')$, $(\textsf{idx}', \textsf{aux}') \leftarrow \texttt{AuxGen}((\textsf{pp}', \textsf{crs}'), \textsf{s})$, $(\textsf{ed}, \textsf{dd}) \leftarrow \texttt{Digest}((\textsf{pp}', \textsf{crs}'), \textsf{idx}', \textsf{aux}')$, $b \xleftarrow{\$} \{0,1\}$, $c_b \leftarrow \texttt{Enc}(\textsf{ed}, u, m_b)$, $(b', \textsf{tr}_2) \leftarrow \mathcal{A}(c_b)$, we have that

$$\Pr[b' = b] = \frac{1}{2} + \epsilon$$

We must construct a corresponding extractor $\mathcal{E}$ such that

$$\Pr[(\textsf{pp}', \textsf{idx}', u, \mathcal{E}((\textsf{pp}', \textsf{crs}'), \textsf{idx}', \textsf{aux}', u, (m_0, m_1), (\textsf{tr}_1, \textsf{tr}_2))) \in \mathcal{R}] \geqslant \epsilon - \textsf{negl}(\lambda).$$

Indeed, given extractor $\mathcal{E}_i$ for each of the gadget arguments Π_i, we construct extractor $\mathcal{E}$ as follows

- $\mathcal{E}((\mathbf{pp}', \mathbf{crs}'), \mathbf{idx}', \mathbf{aux}', u, (m_0, m_1), (\mathsf{tr}_1, \mathsf{tr}_2))$:

  1. Observe the oracle queries in $(\mathsf{tr}_1, \mathsf{tr}_2)$ to compute π. For simplicity, we write the ciphertext from Construction 2 as $\mathsf{ct}_1 = \boldsymbol{s} \cdot A$, $\mathsf{ct}_2 = H(s \cdot b) + \mathsf{msg}$, where $A \in \mathbb{G}_1^{m \times n}$,[11] $\boldsymbol{s} \in \mathbb{F}^m$ and $b \in \mathbb{G}_T^m$. Then we construct an extractor $\mathcal{E}'$ which outputs π such that $A \cdot \pi = b$. Let E be the event that the list of random oracle queries made by $\mathcal{A}$ contains $\boldsymbol{s} \cdot b$. Then, we have that $\Pr[\mathcal{E}' \to \pi] \geqslant \Pr[E] - \mathsf{negl}(\lambda)$. This can be seen by viewing $\boldsymbol{s}$ as formal variables in the GGM. Since $\boldsymbol{s} \cdot b$ was in the list of random oracle queries, it also means that by observing the GGM queries we can obtain a representation in terms of ct_1:

  $$\boldsymbol{s} \cdot b = \sum_{i \in [n]} \mathsf{ct}_{1,i} \circ w_i$$

  Note that there can be no other types of terms can appear in the representation because the degree of the formal variables $\boldsymbol{s}$ must be exactly one on both sides of the equation. All remaining combinations of terms either do not depend on the formal variables $\boldsymbol{s}$ or have degree 2 in $\boldsymbol{s}$. The small loss comes because of we switch from the 'real' world to the GGM. Thus, if the event E occurs, the extract succeeds with overwhelming probability. We now show that the event E occurs with non-negligible probability. We are given an adversary such that

  $$\Pr[\mathcal{A}(\mathsf{ct}_b) \to b] \geqslant \frac{1}{2} + \epsilon,$$

  for some non-negligible ϵ. This can be written as

  $$\begin{aligned} \Pr[\mathcal{A}(\mathsf{ct}_b) \to b] =& \frac{1}{2} \cdot (\Pr[\mathcal{A}(\mathsf{ct}_0) \to 0] + \Pr[\mathcal{A}(\mathsf{ct}_1) \to 1]) \\ =& \frac{1}{2} \cdot (\Pr[\mathcal{A}(\mathsf{ct}_0) \to 0 \mid E] \cdot \Pr[E] + \Pr[\mathcal{A}(\mathsf{ct}_0) \to 0 \mid \neg E] \cdot \Pr[\neg E]) \\ &+ \frac{1}{2} \cdot (\Pr[\mathcal{A}(\mathsf{ct}_1) \to 1 \mid E] \cdot \Pr[E] + \Pr[\mathcal{A}(\mathsf{ct}_1) \to 1 \mid \neg E] \cdot \Pr[\neg E]) \end{aligned}$$

  We now have that $\Pr[\mathcal{A}(\mathsf{ct}_b) \to b \mid \neg E] = 1/2$ because the output of the random oracle is uniformly random, and hence the adversary learns no information about the message unless it queries $\boldsymbol{s} \cdot b$. The expression then simplifies to

  $$\begin{aligned} \Pr[\mathcal{A}(\mathsf{ct}_b) \to b] =& \frac{1}{2} \cdot \Pr[E] \cdot (\Pr[\mathcal{A}(\mathsf{ct}_0) \to 0 \mid E] + \Pr[\mathcal{A}(\mathsf{ct}_1) \to 1 \mid E]) + \frac{1}{2} \cdot (1 - \Pr[E]) \\ \leqslant& \frac{1 + \Pr[E]}{2} \end{aligned}$$

  where the inequality arises from $\Pr[\mathcal{A}(\mathsf{ct}_b) \to b \mid E] \leqslant 1$. Thus, $\Pr[E] \geqslant 2 \cdot \epsilon$, and hence $\mathcal{E}'$ succeeds with non-negligible probability.

  2. Parse $(\overline{\boldsymbol{w}}', (\pi_1, \ldots, \pi_n)) \leftarrow \pi$
  3. Compute $\boldsymbol{u}' \leftarrow \varphi_{\mathsf{inst}}(u)$
  4. For $i \in [\![n]\!]$ compute

  $$w_i \leftarrow \mathcal{E}_i((\mathbf{pp}'_{\mathcal{C}_{\mathsf{CRS},i}}, \mathbf{crs}'_{\mathcal{C}_{\mathsf{CRS},i}}), \mathbf{idx}'_{\mathcal{C}_{\mathsf{IDX},i}}, \mathbf{aux}'_{\mathcal{C}_{\mathsf{IDX},i}}, (\overline{\boldsymbol{w}}'_{\mathcal{C}_{\mathsf{WIT},i}}, \boldsymbol{u}'_{\mathcal{C}_{\mathsf{INS},i}}), \pi_i, (\mathsf{tr}_1, \mathsf{tr}_2))$$

  5. Compute $\boldsymbol{w}'$ such that $\boldsymbol{w}'_{\mathcal{C}_{\mathsf{WIT},i}} = w_i$.

[11]Note that this argument can easily be extended to the more general case where A has entries from both $\mathbb{G}_1$ and $\mathbb{G}_2$.

6. Compute and output $w \leftarrow \varphi_{\mathsf{wit}}^{-1}(\mathbf{pp'}, \mathbf{idx'}, \boldsymbol{u'}, \boldsymbol{w'})$.

We now argue that the extractor produces a satisfying witness with probability $\epsilon - \mathsf{negl}(\lambda)$. Indeed, by the above reasoning, we have with probability $\epsilon - \mathsf{negl}(\lambda)$ that $\mathcal{E}$ produces $(\overline{\boldsymbol{w}}', (\pi_1, \ldots, \pi_n)) \leftarrow \pi$ such that

$$\mathtt{Verify}_i(\mathsf{vk}_i, (\overline{\boldsymbol{w}}'_{\mathcal{C}_{\mathsf{WIT},i}}, \boldsymbol{u}'_{\mathcal{C}_{\mathsf{INS},i}}), \pi_i) = 1$$

for $\mathsf{vk}_i \leftarrow \mathsf{dd}_i$ in the first step.

Then, by the independence of trapdoors for each of the $\mathtt{Gen}'$ algorithms, and by the knowledge soundness of Π_i, if $\mathcal{E}$ successfully produces an accepting proof π, for

$$w_i \leftarrow \mathcal{E}_i((\mathbf{pp}'_{\mathcal{C}_{\mathsf{CRS},i}}, \mathbf{crs}'_{\mathcal{C}_{\mathsf{CRS},i}}), \mathbf{idx}'_{\mathcal{C}_{\mathsf{IDX},i}}, \mathbf{aux}'_{\mathcal{C}_{\mathsf{IDX},i}}, (\overline{\boldsymbol{w}}'_{\mathcal{C}_{\mathsf{WIT},i}}, \boldsymbol{u}'_{\mathcal{C}_{\mathsf{INS},i}}), \pi_i, (\mathsf{tr}_1, \mathsf{tr}_2)) \tag{1}$$

we have that

$$\Pr[(\mathbf{pp}'_{\mathcal{C}_{\mathsf{CRS},i}}, \mathbf{idx}'_{\mathcal{C}_{\mathsf{IDX},i}}, (\overline{\boldsymbol{w}}'_{\mathcal{C}_{\mathsf{WIT},i}}, \boldsymbol{u}'_{\mathcal{C}_{\mathsf{INS},i}}), w_i) \in [\mathcal{R}_i]_{(\mathtt{CGen}_i, \mathtt{Commit}_i)}] \approx 1 \tag{2}$$

for all $i \in [\![n]\!]$.

Next, we must ensure that there exists $\boldsymbol{w}'$ such that $\boldsymbol{w}'_{\mathcal{C}_{\mathsf{WIT},i}} = w_i$ for all $i \in [\![n]\!]$. By the construction of the routing matrices $M_1, \ldots, M_n$ and intercompatibility property we have that

$$\begin{aligned}\overline{\boldsymbol{w}}'_{\mathcal{C}_{\mathsf{WIT},i}} &= \mathtt{Commit}_i(\mathbf{crs}', w_i) \\ &= \mathtt{Commit}'_{\mathcal{C}_{\mathsf{WIT},i}}(\mathbf{crs}', w_i)\end{aligned}$$

By definition, we must have that for $j \in \mathcal{C}_{\mathsf{WIT},i}$

$$\overline{\boldsymbol{w}}'_j = \mathtt{Commit}'_j(\mathsf{crs}', w_{i,j})$$

By the binding property of $\overline{\boldsymbol{w}}'$ we must have that $w_{i,j} = w_{i',j}$ for all $i, i' \in [\![n]\!]$. We refer to this value as w'_j. Then, by definition, we have that $\boldsymbol{w}'_{\mathcal{C}_{\mathsf{WIT},i}} = w_i$ for all $i \in [\![n]\!]$.

Next, by Equation (2) and the definition of a committed relation, we have that for all $i \in [\![n]\!]$

$$(\mathbf{pp}'_{\mathcal{C}_{\mathsf{CRS},i}}, \mathbf{idx}'_{\mathcal{C}_{\mathsf{IDX},i}}, \boldsymbol{u}'_{\mathcal{C}_{\mathsf{INS},i}}, \boldsymbol{w}'_{\mathcal{C}_{\mathsf{WIT},i}}) \in \mathcal{R}_i$$

with probability $1 - \mathsf{negl}(\lambda)$. Then, by definition of a routed relation this means that

$$(\mathbf{pp}', \mathbf{idx}', \boldsymbol{u}', \boldsymbol{w}') \in \boxed{\mathcal{R}_1 \times \cdots \mathcal{R}_n}_{\mathcal{C}}.$$

Then by the invertibility of Π we have that for $(u, w) \leftarrow \varphi_{\mathsf{wit}}(\mathbf{pp}', \mathbf{idx}', \boldsymbol{u}', \boldsymbol{w}')$ that $(\mathbf{pp}', \mathbf{idx}', u, w) \in \mathcal{R}$. As such $\mathcal{E}$ succeeds with probability $\epsilon - \mathsf{negl}(\lambda)$. □

5 Linearly Verifiable Relations

To aid in the design of witness encryption schemes from linearly verifiable arguments (Definition 5), we construct *gadgets*, which are *succinct* linearly-verifiable arguments secure in the GGM, for various useful relations. The succinctness is obtained by first committing to the witness using a succinct commitment scheme and then constructing a linearly verifiable argument on top of the commitments, which can in turn be reduced to the original relation. These gadgets can be *composed* with each other to build a succinct, linearly verifiable argument for a larger relation that would enable interesting applications such as the advanced encryption schemes that we consider in Section 6 and 7.

Common Setup. All gadgets that we construct use an identical setup – the powers-of-tau CRS:

$$\texttt{Gen}(\lambda, N, \tau) \to ([1]_1, [\tau^1]_1, \dots, [\tau^N]_1), ([1]_2, [\tau^1]_2, \dots, [\tau^N]_2)$$

where $\tau \leftarrow\!\!\$\ \mathbb{F}$ and N is chosen to be an upper-bound on the size of the largest relation. We will show that all gadgets can use the same CRS that is setup once. Importantly, this means the size of the CRS only grows with the size of the largest relation and not the number of gadgets that are used.

Extensions. In all gadgets, any subset of the witness can be trivially made a part of the statement as the resulting relation will still have linear verification. Furthermore, the groups $\mathbb{G}_1, \mathbb{G}_2$ can always be swapped as needed for efficiency/intercompatibility with other gadgets.

5.1 Signature Relation

We start with a simple example of the signature relation. Note here that we make use of a versino where the public key pk is defined to be a part of the witness; It could also be made a part of the statement.

Definition 11 (Signature Relation). *We define the signature relation* Sig *as:*

$$\mathsf{Sig} = \left\{ \begin{array}{l|l} \textit{parameters } \bot, & \\ \textit{index } \bot, & \\ \textit{instance } \mathsf{tg}, & [1]_1 \circ \sigma = \mathsf{tg} \circ \mathsf{pk} \;\; . \\ \textit{witness } (\mathsf{pk}, \sigma) & \end{array} \right\}$$

Observe that the Sig relation corresponds to a proof of knowledge of a BLS signature [BLS01]. The signature gadget will then, in fact, be the Boneh-Franklin IBE scheme [BF01]. Note that succinctness is not interesting here as the relation itself is constant-sized.

Construction 3 (Signature Gadget). *Let* $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ *be pairing friendly groups. We construct a linearly verifiable non-interactive argument of knowledge for* $[\mathsf{Sig}]_{\mathsf{ID}\times\mathsf{ID}}$.

- $\texttt{AuxGen}(\cdot) \to \bot$
- $\texttt{Digest}(\mathsf{crs}, \bot, \bot)$: *Output* $\mathsf{vk} \leftarrow \mathsf{crs}$*, and* $\mathsf{pk} \leftarrow \mathsf{crs}$.
- $\texttt{Prove}(\mathsf{pk}, (\mathsf{tg}, (\mathsf{pk}, \sigma)))$*: Given* σ *such that* $[1]_1 \circ \sigma = \mathsf{tg} \cdot \mathsf{pk}$*, output* $\pi = (\mathsf{pk}, \sigma)$.
- $\texttt{Verify}(\mathsf{vk}, \mathsf{tg}, \pi)$*: Parse* $\pi = (\mathsf{pk}, \sigma)$*. Check that* $[1]_1 \circ \boxed{\sigma} = \mathsf{tg} \circ \boxed{\mathsf{pk}}$.

Lemma 2 (Signature Gadget). *Construction 3 is a linearly verifiable non-interactive argument for* $[\mathsf{Sig}]_{\mathsf{ID}}$ *in the generic group model.*

Proof. Completeness, and linear verification follows by inspection. For knowledge soundness, we can simply output π to be the signature σ. □

5.2 Max Degree Relation

We now move on to the first relation with an interesting notion of succinctness that we refer to as Max Degree. The witness value is a vector of *evaluations* of some polynomial (say $f(X)$) on some domain D specific as part of the statement. The relation is satisfied if $\deg(f) \leqslant d$, where d is again specified in the statement. The gadget uses a KZG-like [KZG10] commitment scheme using a powers-of-tau common reference string with powers up to, say, N and then demanding that the prover creates a commitment to the polynomial $f'(X) = X^{N-d} \cdot f(X)$. If $\deg(f(X)) > d$, then the prover cannot create such a commitment because $\deg(f'(X)) > N$, but the CRS only has powers up to N.

Definition 12 (Max Degree Relation)**.** *We define the max degree relation* MaxDeg *over instance, witness tuples as follows*

$$\mathsf{MaxDeg} = \left\{ \begin{array}{l|l} \textit{parameters } \bot, & D \subset \mathbb{F}, |D| = n, \\ \textit{index } \bot, & (d, w) \in \mathbb{F} \times \mathbb{F}^n, \\ \textit{instance } (D, d), & \deg(\sum_i w_i \cdot L_i^D(X)) \leqslant d \\ \textit{witness } w & \end{array} \right\},$$

where $\{L_i^D(X)\}_{i \in [\![n]\!]}$ *denote Lagrange polynomials corresponding to domain* D.

Construction 4 (Max Degree Gadget)**.** *Let* $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ *be pairing friendly groups. We construct a linearly verifiable succinct non-interactive argument of knowledge for* $[\mathsf{MaxDeg}]_{\mathsf{SCS}(\mathbb{G}_1)}$*, with the domain of* $\mathsf{SCS}(\mathbb{G}_1)$ *set to the* D *in Definition 12.*

- $\texttt{AuxGen}(\cdot) \to \bot$
- $\texttt{Digest}(\mathsf{crs}, \bot, \bot)$: *Output* $\mathsf{vk} \leftarrow \mathsf{crs}$*, and* $\mathsf{pk} \leftarrow \mathsf{crs}$.
- $\texttt{Prove}(\mathsf{pk}, ((d, \overline{w}), w))$*: Given* w *such that* $\overline{w} = \mathsf{SCS}(\mathbb{G}_2).\texttt{Commit}(w) = \left[\sum_{i \in [\![n]\!]} w_i \cdot L_i(\tau)\right]_1$*, compute* $\hat{w}(X) = X^{N-d} \cdot \left(\sum_{i \in [\![n]\!]} w_i \cdot L_i(X)\right)$*. Using* crs*, homomorphically compute, and output* $\pi \leftarrow [\hat{w}(\tau)]_1$.
- $\texttt{Verify}(\mathsf{vk}, (d, \overline{w}), \pi)$*: Parse* $\pi = [\hat{w}(\tau)]_1$*. Check that* $\overline{w} \circ [\tau^{N-d}]_2 = [\hat{w}(\tau)]_1 \circ [1]_2$.

Lemma 3 (Max Degree Gadget)**.** *Construction 4 is a linearly verifiable succinct non-interactive argument for* $[\mathsf{MaxDeg}]_{\mathsf{SCS}(\mathbb{G}_1)}$ *in the generic group model.*

Proof. Completeness, linear verification and succinctness follows by inspection. For knowledge soundness, we can extract w by observing the GGM queries made by the prover to compute $\overline{w}$. Let $(\overline{w}, [\hat{w}(\tau)]_1)$ be expressed as:

$$\overline{w} = \sum_{i=0}^{N} f_i \cdot [\tau^i]_1 + \sum_i \overline{z}_i \cdot [R_i]_1, \quad \hat{w} = \sum_{i=0}^{N} \hat{f}_i \cdot [\tau^i]_1 + \sum_i \hat{z}_i \cdot [R_i]_1,$$

where $\{[R_i]_1\}_i$ denotes all other terms that are not in the pk that $\texttt{Prove}^*$ receives. By treating τ as a formal variable X, it must be the case that:

$$\left(\sum_{i=0}^{N} f_i \cdot X^i + \sum_i \overline{z}_i \cdot R_i\right) \cdot X^{N-d} = \sum_{i=0}^{N} \hat{f}_i \cdot X^i + \sum_i \hat{z}_i \cdot R_i.$$

This implies that:

- $\{\hat{f}_i = 0\}_{i \in [0, N-d-1]}$, and $\{\hat{z}_i = 0\}$ as the left hand side has no terms of degree $< N - d$.
- $\{f_i = 0\}_{i \in [d+1, N]}$ as the right hand side has no terms of degree $> N$.

Simplifying the above equation, we have:

$$\sum_{i=0}^{d} f_i \cdot X^i + \sum_i \overline{z}_i \cdot R_i = \sum_{i=0}^{d} \hat{f}_{i+(N-d)} \cdot X^i.$$

Thus, $\overline{w} = \sum_{i=0}^{d} \hat{f}_{i+(N-d)} \cdot [\tau^i]_1$, and we can extract w as the evaluations of the polynomial $\overline{f}(X) = \sum_{i=0}^{d} \hat{f}_{i+(N-d)} \cdot X^i$ on the domain D.

□

5.3 Zero Check Relation

Next, we consider the zero check relation which enforces that a witness is zero at some set of positions specified in the statement. The gadget works by enforcing that the polynomial interpolating the witness on some domain, has roots on a corresponding subset of points. Concretely, let $f(X)$ be the polynomial that interpolates the witness vector on the domain D. It is then sufficient to check that $f(X)$ can be written as $f(X) = Z(X) \cdot Q(X)$, where $Z(X)$ is the vanishing polynomial on the subset $S \subseteq D$, where we want to enforce zero values.

Definition 13 (Zero Check Relation). *We define the zero check relation* Zero *over instance, witness tuples as follows*

$$\mathsf{Zero} = \left\{ \begin{array}{l|l} \textit{parameters } \bot, & \\ \textit{index } \bot, & S \subseteq [n], w \in \mathbb{F}^n, \\ \textit{instance } S, & \{w_i = 0\}_{i \in S} \\ \textit{witness } w & \end{array} \right\}.$$

Construction 5 (Zero Check Gadget). *Let* $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ *be pairing friendly groups. We construct a linearly verifiable succinct non-interactive argument of knowledge for* $[\mathsf{Zero}]_{\mathsf{SCS}(\mathbb{G}_1)}$,

- $\texttt{AuxGen}(\cdot) \to \bot$
- $\texttt{Digest}(\mathsf{crs}, \bot, \bot)$: *Output* $\mathsf{vk} \leftarrow \mathsf{crs}$*, and* $\mathsf{pk} \leftarrow \mathsf{crs}$.
- $\texttt{Prove}(\mathsf{pk}, ((S, \overline{w}), w))$*: Given* w *such that* $\overline{w} = \mathsf{SCS}(\mathbb{G}_2).\texttt{Commit}(w) = \sum_{i \in [n]} w_i \cdot L_i(\tau)$*, compute* $Q(X) = \left(\sum_{i \in [n]} w_i \cdot L_i(X)\right) / Z_S(X)$ *where* $Z_S(X) \leftarrow \prod_{i \in S}(X - D[i])$*, and* D *is the domain in* $\mathsf{SCS}(\mathbb{G}_1)$*. Using* pk*, homomorphically compute and output* $\pi \leftarrow [Q(\tau)]_1$.
- $\texttt{Verify}(\mathsf{vk}, (S, \overline{w}), \pi)$*: Parse* $\pi = [Q(\tau)]_1$*. Using* vk*, homomorphically compute* $[Z_S(\tau)]$ *where* $Z_S(X) = \prod_{i \in S}(X - D[i])$*, and* D *is the domain in* $\mathsf{SCS}(\mathbb{G}_1)$*. Check that* $\overline{w} \circ [1]_2 = [Q(\tau)]_1 \circ [Z_S(\tau)]_2$.

Remark 1. $[Z_S(\tau)]_2$ *can be preprocessed and given as part of* vk *when computing* $\texttt{Digest}$*, if the set* S *is fixed/known beforehand.*

Lemma 4 (Zero Check Gadget). *Construction 5 is a linearly verifiable succinct non-interactive argument for* $[\mathsf{Zero}]_{\mathsf{SCS}(\mathbb{G}_1)}$ *in the generic group model.*

Proof. Completeness, linear verification and succinctness follows by inspection. By observing the GGM queries, we can express $\overline{w}$, and $Q[\tau]_1$ as:

$$\overline{w} = \sum_{i=0}^{N} f_i \cdot [\tau^i]_1 + \sum_i z_i \cdot [R_i]_1, \quad Q[\tau]_1 = \sum_{i=0}^{N} q_i \cdot [\tau^i]_1 + \sum_i z'_i \cdot [R_i]_1,$$

where $\{[R_i]_1\}_i$ denotes all other terms that are not in the pk that $\texttt{Prove}^*$ receives. By treating τ as a formal variable X and denoting $z = \sum_i z_i \cdot R_i$, and $z' = \sum_i z'_i \cdot R_i$, we have that:

$$\sum_{i=0}^{N} f_i \cdot X^i + z = \left(\sum_{i=0}^{N} q_i \cdot X^i + z'\right) \cdot Z_S(X).$$

It must now be the case that:

$$\left\{\sum_{i=0}^{N} f_i \cdot D[s]^i + z = 0\right\}_{s \in S}$$

Since $|S| \geqslant 1$, we can now solve for z as:

$$z = -\sum_{i=0}^{N} f_i \cdot D[s]^i$$

for some $s \in S$. We can now extract w as the evaluations of $\overline{f}(X) = \sum_{i=1}^{N} f_i X^i + f_0 + z$ on the domain D in $\mathsf{SCS}(\mathbb{G}_1)$. We are guaranteed that $\{w_i = 0\}_{i \in S}$ because the right hand side of the verification equation evaluates to zero on the same points by definition of $Z_S(X)$. $\square$

5.4 Indexed Inner Product Relation

The last relation we consider is the inner product relation which checks that a group element $v = \sum_i w_i \cdot u_i$ where $u \in \mathbb{G}^n$ and $w \in \mathbb{F}^n$. In this gadget, we make use of an additional method `AuxGen` which will generate auxiliary information that will aid the `Prove` algorithm. At a high level, we will be using the univariate sumcheck [BCR+19, RZ21] which states that for a given polynomial $f(X)$, the inner product of it's evaluations over some smooth multiplicative domain D with a vector w is equal to some claimed value s, i.e. $\sum_{i \in D} f(i) \cdot w_i = s$, if and only if the following equation is satisfied

$$A(X) \cdot B(X) = s + Q_X(X) \cdot X + Q_Z(X) \cdot Z(X),$$

where $\deg(Q_X(X)) \leqslant |D| - 2$, $B(X)$ interpolates w on D, and $Z(X)$ is the vanishing polynomial on the domain D. Again, we can test this identity at a point τ given the powers-of-tau crs. More accurately, we will work with the generalized version which works over arbitrary domains as stated in Lemma 1.

In it's original formulation, the prover needed knowledge of $A(X)$ and $B(X)$ in order to compute $Q_X(X)$ and $Q_Z(X)$ But recently, [GJM+24, DCX+23] showed that this can be extended to prove a multi-scalar multiplication over a committed set of group elements $(u_1, \ldots, u_n)$. But we can no longer compute the quotient polynomials as we do not know the discrete logarithms of these elements. However, one can compute these quotient polynomials "in the exponent" if the prover is given *auxiliary* information which is computed as function of the crs and the discrete logarithm of $\{u_i\}_i$'s. `AuxGen` computes this information which is then given to the prover. We refer the reader to [GJM+24, DCX+23] for a more detailed exposition.

Definition 14 (Indexed Inner-Product Relation). *Consider a group $\mathbb{G}$ with a scalar field $\mathbb{F}$. We define the indexed inner-product relation* IIP *over index, witness tuples as follows*

$$\mathsf{IIP} = \left\{ \begin{array}{l|l} \textit{parameters } \ \bot, & \\ \textit{index } \ u, & (u, w, v) \in (\mathbb{G}^n, \mathbb{F}^n, \mathbb{G}), \\ \textit{instance } \ \bot, & u \cdot w = v \\ \textit{witness } \ (w, v) & \end{array} \right\}.$$

Construction 6 (Indexed Inner-Product Gadget). *Let $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ be pairing friendly groups. We construct a linearly verifiable succinct non-interactive argument of knowledge for $[\mathsf{IIP}]_{\mathsf{SCS}(\mathbb{G}_2) \times \mathsf{ID}}$ as follows.*

- `AuxGen`(crs, s_i): *Using* crs, *homomorphically compute*

$$\begin{aligned} &\mathsf{idx}_i \leftarrow [s_i]_1, \\ &\mathsf{aux}_i \leftarrow \left\{ \left[s_i \cdot \tau^j\right]_1, \left[s_i \cdot \tau^{N-n+j}\right]_1 \right\}_{j \in [\![n]\!]}, \end{aligned}$$

  where n is the size of the domain D used in $\mathsf{SCS}(\mathbb{G}_2)$. *Output* $(\mathsf{idx}_i, \mathsf{aux}_i)$.

- `Digest`(crs, idx, aux)*: Let D be the domain of the* $\mathsf{SCS}(\mathbb{G}_2)$ *commitment scheme. Let x^* be the point such that all Lagrange polynomials $\{L_i(X)\}_{i\in[\![n]\!]}$ over the domain D, evaluate to the same value y^* at the point x^* i.e. $\{L_i(x^*) = y^*\}_{i\in[\![n]\!]}$. Note that such a point always exists and can be computed in polynomial time using Gaussian elimination. Using* $\mathsf{idx} = ([s_1]_1, \ldots, [s_n]_1)$ *and* aux*, homomorphically compute $C = y^* \cdot \left[\sum_{i\in[n]} s_i \cdot L_i(\tau)\right]_1$, and $Z = [Z(\tau)]_2$ and output*

$$\mathsf{vk} \leftarrow \left(x^*, y^*, C, Z, [\tau]_1, [\tau^{N-n}]_1, [\tau^{N-n+2}]_2, [\tau^N]_2\right),$$

$$\mathsf{pk} \leftarrow (\mathsf{crs}, \mathsf{idx}, \mathsf{aux}),$$

  where $Z(X) = \prod_{i\in[\![n]\!]}(X - D[i])$ is the vanishing polynomial over the domain D.

- `Prove`(pk, $(\overline{w}, v), (w, v)$)*: Given w such that $\overline{w} = \mathsf{SCS}(\mathbb{G}_2).\mathtt{Commit}(w) = \sum_{i\in[\![n]\!]} w_i \cdot L_i(\tau)$, and $v = \sum_{i\in[\![n]\!]} w_i \cdot [s_i]_1$, let $B(X) := \sum_{i\in[\![n]\!]} w_i \cdot L_i(X)$. Next, compute the unique polynomials $Q_X(X)$, and $Q_Z(X)$ satisfying*

$$\left(\sum_{i\in[\![n]\!]} s_i \cdot L_i(X)\right) \cdot B(X) = \frac{\sum_{i\in[\![n]\!]} w_i \cdot s_i}{y^*} + Q_X(X)\cdot(X - x^*) + Q_Z(X)\cdot Z(X),$$

  such that $Q_X(X)$ has degree at most $n-2$. Further compute the polynomials

  - $\hat{Q}_X(X) = X^{N-n+2} \cdot Q_X(X)$
  - $\hat{v}(X) = X^N \cdot \left(\sum_{i\in[\![n]\!]} w_i \cdot s_i\right)$

  Using homomorphic operations over pk *compute:*

$$\pi \leftarrow \left([Q_Z(\tau)]_1, [Q_X(\tau)]_1, [\hat{Q}_X(\tau)]_1, [\hat{v}(\tau)]_1, [\hat{w}(\tau)]_2\right)$$

- `Verify`(vk, $(\overline{w}, v), \pi$)*: Check that*

  1. $C \circ \overline{w} = v \circ [y^{*-1}]_2 + [Q_X(\tau)]_1 \circ [\tau - x^*]_2 + [Q_Z(\tau)]_1 \circ Z$
  2. $[Q_X(\tau)]_1 \circ [\tau^{N-n+2}]_2 = [\hat{Q}_X(\tau)]_1 \circ [1]_2$
  3. $v \circ [\tau^N]_2 = [\hat{v}(\tau)]_1 \circ [1]_2$

Lemma 5 (Indexed Inner-Product Gadget)**.** *Construction 6 is a linearly verifiable succinct non-interactive argument for* $[\mathsf{IIP}]_{(\mathsf{SCS}(\mathbb{G}_2),\mathsf{ID})}$.

Proof. Completeness, linear verification and succinctness follows by inspection. For knowledge soundness, we begin by first simplifying pk to be crs, $\{[s_i \cdot \tau^j]_1\}_{i\in[n],j\in[0,n]}$, as all other terms can be computed using homomorphic operations on these terms. By monitoring the GGM queries made by the adversary, $\overline{w}$ can be expressed as:

$$\overline{w} = \sum_{i=0}^{N} f_{w,i} \cdot [\tau^i]_2 + \sum_i b_{w,i} \cdot [R_{2,i}]_2$$

where $\{[R_{2,i}]_2\}_i$ denotes all other terms that are not in the pk that `Prove`* receives. Using a similar argument as in Lemma 3:

- Check 2 implies that we can compute a representation of $Q_X = \sum_{i=0}^{n-2} f_{x,i} \cdot [\tau^i]_1$,

- Check 3 implies that we can compute a representation of $v = f_{v,0} \cdot [1]_1$.

By treating τ as a formal variable X in the symbolic GGM, Check 1 then implies that:

$$\left(\sum_{i \in [\![n]\!]} s_i \cdot L_i(X)\right) \cdot \underbrace{\left(\sum_{i=0}^{N} f_{w,i} \cdot X^i + \sum_i b_{w,i} \cdot R_{2,i}\right)}_{w(X)} = \frac{f_{v,0}}{y^*} + \left(\sum_{i=0}^{n-2} f_{x,i} \cdot X^i\right) \cdot (X - x^*) + Q_Z(X) \cdot Z(X) \tag{3}$$

where $Q_Z(X)$ is a polynomial of degree at most $N-1$ because the left hand side has degree at most $N+n-1$ and $Z(X)$ has degree n.

Note that we do not know $R_{2,i}$, and hence cannot yet extract a witness $w \in \mathbb{F}^n$ such that $\overline{w} = \sum_{i \in [\![n]\!]} w_i \cdot L_i(\tau)$. However, we can extract $w' = f_{w,0} + \sum_i b_{w,i} \cdot R_{2,i}$, which will be sufficient to compute the witness w. From the univariate sumcheck lemma 1 it must be the case that:

$$f_{v,0} = \sum_{x \in D} s_i \cdot w(x).$$

By expressing $w(X) = w' + \sum_{i=1}^{N} f_{w,i} \cdot X^i$, we have that:

$$f_{v,0} = \sum_{x \in D} s_i \cdot \left(w' + \sum_{i=1}^{N} f_{w,i} \cdot X^i\right).$$

We now have two cases:

- If $\sum_{x \in D} s_i \neq 0$, then we can solve for w', and hence extract the corresponding witness w as the evaluations of $w(X)$ on the domain D in $\mathsf{SCS}(\mathbb{G}_2)$.
- If $\sum_{x \in D} s_i = 0$, we start by writing $Q_Z(X) = \sum_{i=0}^{N-1} f_{z,i} \cdot X^i + \sum_i a_{z,i} \cdot R_{1,i}$. Again, we will solve for $z' = f_{z,0} + \sum_i a_{z,i} \cdot R_{1,i}$. Now we evaluate Equation 3 at $X = x^*$ and arrive at:

$$\frac{f_{v,0}}{y^*} + \left(\sum_{i=1}^{N-1} f_{z,i} \cdot {x^*}^i + z'\right) \cdot Z(x^*) = 0$$

because at the point x^* the Lagrange polynomials evaluate to the same value y^*, and hence the left hand size evaluates to zero. Note that $x^* \notin D$ because this would lead to a contradiction that all Lagrange polynomials evaluate to the same value, hence $Z(x^*) \neq 0$. Thus, we can solve for z'. This in turn allows us to solve for w' from Equation 3. Finally, we can extract the corresponding witness w as the evaluations of $w(X)$ on the domain D in $\mathsf{SCS}(\mathbb{G}_2)$.

□

Hard-to-Invert `AuxGen`. We further formalize the property of Hard-to-Invert `AuxGen` algorithm and prove that it holds for the IIP gadget. The essence of the property is that: Given idx and aux outputted by `AuxGen`(crs, s), it is hard to find s. This is a natural property that should hold in applications where `AuxGen` is used as a public-key generation algorithm (with pk = (idx, aux) and sk = s). In particular, we use the `AuxGen` algorithm of IIP for our R-ABE construction (section 6.2), therefore here we prove that it is hard to invert.

Definition 15 (Hardness of `AuxGen`). *Let* WE = (`Gen`, `AuxGen`, `Digest`, `Enc`, `Prove`, `Dec`) *be a witness encryption scheme. Consider the following game between a PPT adversary $\mathcal{A}$ and a challenger $\mathcal{C}$:*

1. **Setup Phase.** *$\mathcal{C}$ samples $\mathsf{crs} \leftarrow \mathtt{Gen}(\lambda, N, \mathsf{td})$ and sends crs to $\mathcal{A}$. $\mathcal{C}$ also initializes a set of inverted indices $\mathsf{Cor} \leftarrow \emptyset$ and a counter $K \leftarrow 0$.*

2. **Query Phase.** *$\mathcal{A}$ chooses one of the following queries.*

   (a) $\mathtt{AuxGen}$ **query.** *$\mathcal{C}$ samples a uniformly random s_i, runs $(\mathsf{idx}_i, \mathsf{aux}_i) \leftarrow \mathtt{AuxGen}(\mathsf{crs}, s_i)$ and sends $(\mathsf{idx}_i, \mathsf{aux}_i)$ to $\mathcal{A}$. $\mathcal{C}$ also updates $K \leftarrow K + 1$*

   (b) **Inversion query.** *$\mathcal{A}$ chooses an index i and sends it to $\mathcal{C}$. If $i \in [\![K]\!]$, $\mathcal{C}$ updates $\mathsf{Cor} \leftarrow \mathsf{Cor} \cup \{i\}$ and sends s_i to $\mathcal{A}$.*

3. **Output.** *The adversary $\mathcal{A}$ outputs (i, s_i^*).*

We say that WE *has a Hard-to-Invert* $\mathtt{AuxGen}$ *if for any PPT adversary* $\Pr[i \in [\![K]\!] \setminus \mathsf{Cor} \land \mathtt{AuxGen}(\mathsf{crs}, s_i^*) = (\mathsf{idx}_i, \mathsf{aux}_i)] = \mathsf{negl}(\lambda)$.

Lemma 6 (Hardness of IIP $\mathtt{AuxGen}$)**.** *Construction 6 has a Hard-to-Invert* $\mathtt{AuxGen}$ *algorithm against generic adversaries.*

Proof. The proof is straightforward in the Generic Group Model. At the end of step 2 the adversary's view in the symbolic GGM is:

$$\left\{ \left[s_i \cdot X^j\right]_1, \left[s_i \cdot X^{N-n+j}\right]_1 \right\}_{j \in [\![n]\!]} \quad \text{for each } i \in \mathsf{Cor}$$
$$\left\{ S_i, \left[S_i \cdot X^j\right]_1, \left[S_i \cdot X^{N-n+j}\right]_1 \right\}_{j \in [\![n]\!]} \quad \text{for each } i \in [\![K]\!] \setminus \mathsf{Cor}$$

where S_i and X are formal variables. Since all S_i's $i \in [\![K]\!] \setminus \mathsf{Cor}$ are formal variables then (after sampling a random point for each formal variable) the probability of the adversary to output an s_i for $i \in [\![K]\!] \setminus \mathsf{Cor}$ is $\mathsf{negl}(\lambda)$. □

6 Registered Attribute-Based Encryption with Linear CRS

In this section, we present our Registered Attribute-Based Encryption (R-ABE) construction, that enjoys linear — in the number of users — common reference string. We begin by recalling the definition of R-ABE and then proceed to our construction. We describe our construction via our witness encryption framework. For completeness, we provide the unrolled description of our protocol in Appendix B.1.

6.1 Slotted Registered Attribute-Based Encryption

Hohenberger et al. [HLWW23] introduced the notion of Registered Attribute-Based Encryption. They, further, introduced the simpler model of Slotted Registered Attribute-Based Encryption (Slotted R-ABE). In Slotted R-ABE, there is a fixed number m of slots and every user is registering to one of the slots. Then, registration is not dynamic, all m users register at once. This greatly simplifies the construction, security and exposition of R-ABE. Furthermore, Hohenberger et al. [HLWW23], inspired by the technique of [GHMR18], provided a generic way to bootstrap any Slotted R-ABE to a full-fledged R-ABE. It has yet become standard to work in the simpler model of Slotted R-ABE, therefore we, also, focus on it.

Below is the formal definition of Slotted R-ABE [HLWW23]. For completeness we, also, provide the definition of full-fledged R-ABE in Appendix A.

Definition 16 (Slotted R-ABE)**.** *A (Ciphertext-Policy) Slotted Registered Attribute-Based Encryption scheme with attribute space $\mathbb{U}$ and policy space $\mathbb{P}$ consists of the following six PPT algorithms* $(\mathtt{Setup}, \mathtt{KGen}, \mathtt{isValid}, \mathtt{Aggregate}, \mathtt{Enc}, \mathtt{Dec})$.

- $\texttt{Setup}(1^\lambda, m) \to \mathsf{crs}$: *On input the security parameter λ and the number of slots m, the randomized setup algorithm samples a common reference string* crs. *The message space $\mathbb{M}$ is also specified here and is (implicitly) contained in the* crs.
- $\texttt{KGen}(\mathsf{crs}, i) \to (\mathsf{pk}_i, \mathsf{sk}_i)$: *On input the common reference string* crs *and a slot index $i \in [\![m]\!]$, the randomized key generation algorithm outputs a public and secret key* $(\mathsf{pk}_i, \mathsf{sk}_i)$.
- $\texttt{isValid}(\mathsf{crs}, i, \mathsf{pk}_i) \to \{0, 1\}$: *On input the common reference string, a slot index $i \in [\![m]\!]$ and the corresponding public key* pk_i, *the deterministic public key verification algorithm outputs a bit, which is* 1 *if* pk_i *is well-formed.*
- $\texttt{Aggregate}(\mathsf{crs}, (\mathsf{pk}_1, \mathcal{U}_1), \ldots, (\mathsf{pk}_m, \mathcal{U}_m)) \to (\mathsf{mpk}, \mathsf{hsk}_1, \ldots, \mathsf{hsk}_m)$: *On input the common reference string* crs *and a set of m public keys with their corresponding attribute sets* $(\mathsf{pk}_1, \mathcal{U}_1), \ldots, (\mathsf{pk}_m, \mathcal{U}_m)$, *the deterministic aggregation algorithm outputs the master public key* mpk *and a set of m helper decryption keys* $(\mathsf{hsk}_1, \ldots, \mathsf{hsk}_m)$ *corresponding to the input public keys respectively.*
- $\texttt{Enc}(\mathsf{mpk}, \mathcal{P}, \mathsf{msg}) \to \mathsf{ct}$: *On input the current public parameters* mpk, *a policy function $\mathcal{P} \in \mathbb{P}$ and a message* $\mathsf{msg} \in \mathbb{M}$, *the randomized encryption algorithm outputs a ciphertext* ct.
- $\texttt{Dec}(\mathsf{sk}, \mathsf{hsk}, \mathsf{ct}) \to \mathsf{msg}$: *On input the secret* sk, *the (current) helper decryption key* hsk *and a ciphertext* ct, *the deterministic decryption algorithm outputs a message* $\mathsf{msg} \in \mathbb{M} \cup \{\bot\}$. *The symbol $\bot$ indicates a syntax error.*

Definition 17 (Slotted R-ABE correctness and efficiency). *A slotted R-ABE scheme should satisfy the following correctness and efficiency properties.*

- **Correctness of keys.** *For all λ, $m = \mathsf{poly}(\lambda)$ and for every $i \in [\![m]\!]$:*

$$\Pr\left[\ \texttt{isValid}(\mathsf{crs}, i, \mathsf{pk}_i) = 1 \quad : \quad \begin{array}{c} \mathsf{crs} \leftarrow \texttt{Setup}(1^\lambda, m); \\ (\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \texttt{KGen}(\mathsf{crs}, i) \end{array}\right] = \mathsf{negl}(\lambda)$$

- **Correctness.** *For all λ, $m = \mathsf{poly}(\lambda)$ and for every $i \in [\![m]\!]$, let* $\mathsf{crs} \leftarrow \texttt{Setup}(1^\lambda, m)$ *and* $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \texttt{KGen}(\mathsf{crs}, i)$. *Then for all* $\{\mathsf{pk}_j\}_{j=1, j\neq i}^{m}$ *with* $\texttt{isValid}(\mathsf{crs}, j, \mathsf{pk}_j) = 1$, *all messages* $\mathsf{msg} \in \mathbb{M}$, *all attribute sets $\mathcal{U}_1, \ldots, \mathcal{U}_m \subseteq \mathbb{U}$ and policies $\mathcal{P} \in \mathbb{P}$ with $\mathcal{P}(\mathcal{U}) = 1$ we have:*

$$\Pr\left[\begin{array}{c} \texttt{Dec}(\mathsf{sk}_i, \mathsf{hsk}_i, \mathsf{ct}) = \mathsf{msg} : \\ (\mathsf{mpk}, \mathsf{hsk}_1, \ldots, \mathsf{hsk}_m) \leftarrow \texttt{Aggregate}(\mathsf{crs}, (\mathsf{pk}_1, \mathcal{U}_1), \ldots, (\mathsf{pk}_m, \mathcal{U}_m)); \\ \mathsf{ct} \leftarrow \texttt{Enc}(\mathsf{mpk}, \mathcal{P}, \mathsf{msg}) \end{array}\right] = \mathsf{negl}(\lambda)$$

- **Compactness.** $|\mathsf{mpk}| = \mathsf{poly}(\lambda, n, \log m)$ *and* $|\mathsf{hsk}_i| = \mathsf{poly}(\lambda, n, \log m)$ *for each $i \in [\![m]\!]$, where $n = |\mathcal{U}_1 \cup \ldots \cup \mathcal{U}_m|$.*

Remark 2 (Unbounded Attribute Space). *In the original definition [HLWW23] the compactness property allows* $|\mathsf{mpk}|$ *and* $|\mathsf{hsk}_i|$ *to polynomially depend on the attribute space* ($\mathsf{poly}(|\mathbb{U}|)$). *This implies that $\mathbb{U}$ has polynomial-size. To allow for unbounded attribute spaces, we modify the definition so that* $|\mathsf{mpk}|$ *and* $|\mathsf{hsk}_i|$ *only polynomially depend on the number of effective attributes* ($\mathsf{poly}(n)$ *instead), i.e. the attributes that have been included so far in an attribute set of a registered user.*

Definition 18 (Security of Slotted R-ABE). *Consider the following game between a PPT adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.*

1. **Setup Phase.** *$\mathcal{C}$ samples* $\mathsf{crs} \leftarrow \mathtt{Setup}(1^\lambda, M)$*. It also initializes a public key-secret key map* $\mathcal{D}_k$ *and a public key-slot index map* $\mathcal{D}_s$*, both initialized implicitly to* $\mathcal{D}_k[\mathsf{pk}] = \bot$ *and* $\mathcal{D}_s[\mathsf{pk}] = \bot$ *for every* pk*. It also initializes a set of corrupted keys* $\mathsf{Cor} \leftarrow \emptyset$*.* $\mathcal{C}$ *sends* crs *to* $\mathcal{A}$*.*

2. **Query Phase.** *$\mathcal{A}$ chooses one of the following queries.*

   (a) **Honest key generation query.** *$\mathcal{A}$ sends a slot index i to $\mathcal{C}$. $\mathcal{C}$ samples a public-secret key pair* $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathtt{KGen}(\mathsf{crs}, i)$ *for the slot i. Then, $\mathcal{C}$ sets* $\mathcal{D}_k[\mathsf{pk}_i] \leftarrow \mathsf{sk}_i$ *and* $\mathcal{D}_s[\mathsf{pk}_i] \leftarrow i$*. $\mathcal{C}$ sends* pk_i *to $\mathcal{A}$.*

   (b) **Key corruption query.** *$\mathcal{A}$ chooses a public key* pk *and sends it to $\mathcal{C}$. $\mathcal{C}$ finds* $\mathsf{sk} \leftarrow \mathcal{D}_k[\mathsf{pk}]$ *in the dictionary. If* $\mathsf{sk} = \bot$ *(meaning that* pk *has never been generated in previous queries) then skip this step. Otherwise, $\mathcal{C}$ add* pk *to the corrupted keys,* $\mathsf{Cor} \leftarrow \mathsf{Cor} \cup \{\mathsf{pk}\}$*, and sends* sk *to $\mathcal{A}$.*

3. **Challenge phase.** *$\mathcal{A}$ chooses a set of public-key attribute pairs* $\{(\mathsf{pk}_1^*, \mathcal{U}_1^*), \ldots, (\mathsf{pk}_m^*, \mathcal{U}_m^*)\}$*, a policy* $\mathcal{P}^* \in \mathbb{P}$ *and two messages* $(\mathsf{msg}_0, \mathsf{msg}_1)$ *and sends them to $\mathcal{C}$. $\mathcal{C}$ for each* $i \in [\![m]\!]$ *proceeds as follows:*

   - *If* $\mathcal{D}_k[\mathsf{pk}_i^*] = \bot$ *(public key chosen by the adversary, not queried) and* $\mathtt{isValid}(\mathsf{crs}, i, \mathsf{pk}_i) = 0$ *(not valid key) then aborts the game.*
   - *If* $\mathcal{D}_k[\mathsf{pk}_i^*] \neq \bot$ *(public key queried) and* $\mathcal{D}_s[\mathsf{pk}_i] \neq i$ *(queried with a different slot) then aborts the game.*
   - *If* $\mathcal{D}_k[\mathsf{pk}_i^*] = \bot$ *(public key chosen by the adversary, not queried) and* $\mathtt{isValid}(\mathsf{crs}, i, \mathsf{pk}_i^*) = 1$ *(valid key) then add* pk_i^* *to corrupted keys* $\mathsf{Cor} \leftarrow \mathsf{Cor} \cup \{\mathsf{pk}_i^*\}$*.*

   Then, if for any $i \in [\![m]\!]$*,* $\mathsf{pk}_i^* \in \mathsf{Cor}$ *and* $\mathcal{P}^*(\mathcal{U}_i^*) = 1$ *(any corrupted set of attributes satisfies the challenge policy) then $\mathcal{C}$ aborts the game. Then, $\mathcal{C}$ runs the deterministic aggregation algorithm* $(\mathsf{mpk}, \mathsf{hsk}_1, \ldots, \mathsf{hsk}_m) \leftarrow \mathtt{Aggregate}(\mathsf{crs}, (\mathsf{pk}_1, \mathcal{U}_1), \ldots, (\mathsf{pk}_m, \mathcal{U}_m))$ *and computes the challenge ciphertext* $\mathsf{ct}^* \leftarrow \mathtt{Enc}(\mathsf{mpk}, \mathcal{P}^*, \mathsf{msg}^*)$*.*

4. **Output.** *The adversary $\mathcal{A}$ outputs a bit b'.*

We call an R-ABE scheme secure if for all $m = \mathsf{poly}(\lambda)$ *and for all PPT adversaries $\mathcal{A}$ it holds that* $\Pr[b = b'] \leqslant \frac{1}{2} + \mathsf{negl}(\lambda)$*.*

6.2 Our Slotted R-ABE Construction

We construct a Slotted Registered Attribute-Based Encryption for DNF formulas, denoted as:

$$\mathcal{P} := \bigvee_{\alpha=1}^{|\mathrm{OR}|} \bigwedge_{\beta=1}^{k_\alpha} u_{i_{\alpha,\beta}}^{(b_{\alpha,\beta})}$$

where u_i are attributes and b denotes negation, i.e. $u^{(0)} = u$ and $u^{(1)} = \neg u$.

Let $\mathcal{U}_1, \ldots, \mathcal{U}_m$ be the attribute sets of the users. We define the effective set of attributes as $\mathcal{U}_{\mathsf{eff}} = \mathcal{U}_1 \cup \ldots \cup \mathcal{U}_m := \{u_1, \ldots, u_n\}$ and, further, the sets of indices (i.e. users) that possess each attribute as $\mathcal{I}_{u_1} = \{i \in [\![m]\!] : u_1 \in \mathcal{U}_i\}, \ldots, \mathcal{I}_{u_n} = \{i \in [\![m]\!] : u_n \in \mathcal{U}_i\}$.

Then for each AND clause of a DNF policy we are working over the relation:

$$\mathsf{AND} = \left\{ \begin{array}{l|l} \text{index } (\mathsf{pk}_i)_{i\in[\![m]\!]}, (\mathcal{I}_{u_i})_{i\in[\![n]\!]}, \mathcal{U}_{\mathsf{eff}} & \langle(\mathsf{pk}_1, \ldots, \mathsf{pk}_m), \boldsymbol{w}\rangle = \mathsf{pk}, \\ \text{instance } \bigwedge_{\beta=1}^{k} u_{i_\beta}^{(b_\beta)} & \mathsf{pk} = [\mathsf{sk}]_1, \\ \text{witness } \boldsymbol{w}, \mathsf{pk}, \mathsf{sk} & w_j = 0 \text{ in } j \in \mathcal{J}_{u_{i_\beta}^{(b_\beta)}}, \text{ for } \beta \in [\![k]\!] \end{array} \right\}$$

where in the above $\mathcal{J}$ is defined as:

$$\mathcal{J}_{u_{i_\beta}^{(b_\beta)}} = \begin{cases} [\![m]\!] \setminus \mathcal{I}_{u_{i_\beta}}, & \text{if } b_\beta = 0 \text{ and } u_{i_\beta} \in \mathcal{U}_{\mathsf{eff}} \\ \mathcal{I}_{u_{i_\beta}}, & \text{if } b_\beta = 1 \text{ and } u_{i_\beta} \in \mathcal{U}_{\mathsf{eff}} \\ [\![m]\!], & \text{if } b_\beta = 0 \text{ and } u_{i_\beta} \notin \mathcal{U}_{\mathsf{eff}} \\ \emptyset & \text{if } b_\beta = 1 \text{ and } u_{i_\beta} \notin \mathcal{U}_{\mathsf{eff}} \end{cases}$$

$\mathcal{J}_{u_{i_\beta}^{(b_\beta)}}$ can be roughly seen as the complementary of the $\mathcal{I}_{u_{i_\beta}^{(b_\beta)}}$. An instance of the relation AND

$$\left(((\mathsf{pk}_i)_{i\in[\![m]\!]}, (\mathcal{I}_{u_i})_{i\in[\![n]\!]}, \mathcal{U}_{\mathsf{eff}}),\ \Big(\bigwedge_{\beta=1}^{k} u_{i_\beta}^{(b_\beta)} \Big),\ (\ \boldsymbol{w}\ ) \in \mathbb{F}^m \right) \in \mathsf{AND}$$

decomposes as follows:

$$\begin{aligned} \big((\mathsf{pk}_i)_{i\in[\![m]\!]}, \bot,\ (\boldsymbol{w}, \mathsf{pk})\ \big) &\in \mathsf{IIP} \\ \big(\bot, \bot,\ \boldsymbol{w}\ \big) &\in \mathsf{NonZero} \\ \big(\bot, \bot,\ (\mathsf{pk}, \mathsf{sk})\ \big) &\in \mathsf{DLOG} \\ \big(((\mathcal{I}_{u_i})_{i\in[\![n]\!]}, \mathcal{U}_{\mathsf{eff}}), u_{i_\beta}^{(b_\beta)},\ \boldsymbol{w}\ \big) &\in \widehat{\mathsf{Zero}},\ \text{for each } \beta \in [\![k]\!] \end{aligned}$$

where

$$\widehat{\mathsf{Zero}} = \left\{ \begin{array}{l} \text{index } (\mathcal{I}_{u_i})_{i\in[\![n]\!]}, \mathcal{U}_{\mathsf{eff}} \\ \text{instance } u_i^{(b)} \\ \text{witness } \boldsymbol{w} \end{array} \;\middle|\; \big(\bot, \mathcal{J}_{u_i^{(\beta)}}, \boldsymbol{w}\big) \in \mathsf{Zero} \right\}$$

In conclusion, AND reduces to $\boxed{\mathsf{IIP} \times \mathsf{NonZero} \times \mathsf{DLOG} \times \mathsf{Zero}_1 \times \ldots \times \mathsf{Zero}_k}$. Now let $\mathsf{WE}^{\mathsf{AND}}$ be the witness encryption scheme for AND that is built from WE schemes for the corresponding individual relations in

$$\boxed{\mathsf{IIP} \times \mathsf{NonZero} \times \mathsf{DLOG} \times \mathsf{Zero}_1 \times \ldots \times \mathsf{Zero}_k}$$

according to Theorem 1.

Our R-ABE construction, for the most part, relies in a black-box way on (any) Witness Encryption scheme $\mathsf{WE}^{\mathsf{AND}}$ for AND. In fact, $\mathsf{WE}^{\mathsf{AND}}$ suffices to construct a secure and correct R-ABE scheme for DNF policies. However, there is a caveat in the computation of hsk_i in the aggregation phase. Observe that the formula (which is going to be the policy in our scheme) is part of the statement in AND, but the policy is chosen in the encryption phase. For this, the `Prove` algorithm can only be run after the encryption phase, which means that decryptor needs the large `aux` information from the aggregator in order to run; This violates the efficiency property of succinct hsk information.

In order to overcome this we have to "open the box" of $\mathsf{WE}^{\mathsf{AND}}$. Our solution is the following: We let the aggregator precompute the IIP proof of the user i and all possible $\widehat{\mathsf{Zero}}$ proofs, i.e. $\widehat{\mathsf{Zero}}$ proofs for all the $2n$ literals $u_1^{(0)}, u_1^{(1)}, \ldots, u_n^{(0)}, u_n^{(1)}$. Notice that $\mathsf{WE}^{\mathsf{AND}}$ works individually over each literal of the formula, $u_{i_\beta}^{(b_\beta)}$. Therefore, the user i at decryption time can choose-and-combine the $2n$ precomputed Zero proofs according to the ciphertext-policy and the IIP proof, and is only left to compute the proof for DLOG (which requires constant size information). Intuitively, the Aggregator computes the linear SNARK proofs for each user, which would otherwise take $O(m)$ time for the user to compute.

Construction 7. *Let* $\mathsf{WE}^{\mathsf{AND}}$ *be the witness encryption scheme for the* AND *relation, as described above. Below is the Registered Attribute-Based Encryption scheme.*

- $\texttt{Setup}(1^\lambda, M)$: *Sample* $\mathsf{td} \leftarrow\!\!\$\ \mathbb{F}$*, and compute* $(\widetilde{\mathsf{pp}}, \widetilde{\mathsf{crs}}) \leftarrow \mathsf{WE}^{\mathsf{AND}}.\texttt{Gen}(\lambda, m, \mathsf{td})$. *Output* $\mathsf{crs} = (\widetilde{\mathsf{pp}}, \widetilde{\mathsf{crs}})$.

- $\texttt{KGen}(\mathsf{crs})$ *Sample a secret key* $\mathsf{sk} \leftarrow\!\!\$\ \mathbb{F}$ *and run* $(\widetilde{\mathsf{pk}}, \mathsf{aux}^{\mathsf{IIP}}) \leftarrow \mathsf{WE}^{\mathsf{IIP}}.\texttt{AuxGen}(\mathsf{crs}, \mathsf{sk})$ *Output* sk, $\mathsf{pk} = (\widetilde{\mathsf{pk}}, \mathsf{aux}^{\mathsf{IIP}}, \pi)$*, where* π *is a NIZK proof of correct execution of* $\mathsf{WE}^{\mathsf{IIP}}.\texttt{AuxGen}(\mathsf{crs}, \mathsf{sk})$.

- $\texttt{isValid}(\mathsf{crs}, \mathsf{pk}, \mathsf{hint})$: *Parse* $\mathsf{pk} = (\widetilde{\mathsf{pk}}, \mathsf{aux}^{\mathsf{IIP}}, \pi)$*, and check that the NIZK proof passes verification.*[12]

- $\texttt{Aggregate}(\mathsf{crs}, (\mathsf{pk}_1, \mathcal{U}_1), \ldots, (\mathsf{pk}_m, \mathcal{U}_m))$: *Define the set of effective attributes as* $\mathcal{U}_{\mathsf{eff}} = \mathcal{U}_1 \cup \ldots \cup \mathcal{U}_m := \{u_1, \ldots, u_n\} \subseteq \mathbb{U}$. *Then further define a set of slots for each attribute, representing the corresponding users that possess the attribute:*

$$\mathcal{I}_{u_1} = \{i \in [\![m]\!] : u_1 \in \mathcal{U}_i\}, \ldots, \mathcal{I}_{u_n} = \{i \in [\![m]\!] : u_n \in \mathcal{U}_i\}.$$

For each $i \in [\![m]\!]$*, parse* $\mathsf{pk}_i := (\widetilde{\mathsf{pk}}_i, \mathsf{aux}_i^{\mathsf{IIP}}, \pi_i)$ *and run*

$$(\widetilde{\mathsf{ed}}, \widetilde{\mathsf{dd}}) \leftarrow \mathsf{WE}^{\mathsf{AND}}.\texttt{Digest}(\mathsf{crs}, ((\widetilde{\mathsf{pk}}_i)_{i \in [\![m]\!]}, (\mathcal{I}_{u_i})_{i \in [\![n]\!]}, \mathcal{U}_{\mathsf{eff}}), (\mathsf{aux}_i^{\mathsf{IIP}})_{i \in [\![m]\!]}).$$

For each $i \in [\![m]\!]$ *and* $j \in [\![n]\!]$ *compute:*

$$\begin{aligned}
\mathsf{dd}_i^{\mathsf{IIP}} &\leftarrow \mathsf{WE}^{\mathsf{IIP}}.\texttt{Digest}(\mathsf{crs}, (\mathsf{pk}_i)_{i \in [\![m]\!]}, \mathsf{aux}_i^{\mathsf{IIP}}) \\
\mathsf{hsk}_{i,0} &\leftarrow \mathsf{WE}^{\mathsf{IIP}}.\texttt{Prove}(\mathsf{dd}_i^{\mathsf{IIP}}, \bot, (\boldsymbol{e}_i, \widetilde{\mathsf{pk}}_i)) \\
\mathsf{dd}_i^{\mathsf{NonZero}} &\leftarrow \mathsf{WE}^{\mathsf{NonZero}}.\texttt{Digest}(\mathsf{crs}, \bot, \bot) \\
\mathsf{hsk}'_{i,0} &\leftarrow \mathsf{WE}^{\mathsf{NonZero}}.\texttt{Prove}(\mathsf{dd}_i^{\mathsf{NonZero}}, \bot, \boldsymbol{e}_i) \\
\mathsf{dd}_{i,j}^{\widehat{\mathsf{Zero}}} &\leftarrow \mathsf{WE}^{\widehat{\mathsf{Zero}}}.\texttt{Digest}(\mathsf{crs}, (\mathcal{I}_{u_i})_{i \in [\![n]\!]}), \bot) \\
\mathsf{hsk}_{i,j}^{(0)} &\leftarrow \mathsf{WE}^{\widehat{\mathsf{Zero}}}.\texttt{Prove}(\mathsf{dd}_{i,j}^{\widehat{\mathsf{Zero}}}, u_j^{(0)}, \boldsymbol{e}_i) \\
\mathsf{hsk}_{i,j}^{(1)} &\leftarrow \mathsf{WE}^{\widehat{\mathsf{Zero}}}.\texttt{Prove}(\mathsf{dd}_{i,j}^{\widehat{\mathsf{Zero}}}, u_j^{(1)}, \boldsymbol{e}_i)
\end{aligned}$$

where $\boldsymbol{e}_i = (0, \ldots, 1, \ldots, 0)$ *is the* i*-th unit vector. Output*

$$\mathsf{mpk} = (\widetilde{\mathsf{ed}}, \mathcal{U}_{\mathsf{eff}})$$
$$\left\{ \mathsf{hsk}_i = \left(\mathcal{U}_i, \mathsf{hsk}_{i,0}, \mathsf{hsk}'_{i,0}, \{\mathsf{hsk}_{i,1}^{(b)}, \ldots, \mathsf{hsk}_{i,n}^{(b)}\}_{b \in \{0,1\}} \right) \right\}_{i \in [\![m]\!]}$$

- $\texttt{Enc}(\mathsf{mpk}, \mathcal{P}, \mathsf{msg})$: *Parse* $\mathsf{mpk} := (\widetilde{\mathsf{ed}}, \mathcal{U}_{\mathsf{eff}})$. *Let the policy be the DNF formula* $\mathcal{P} := \bigvee_{\alpha=1}^{|OR|} l_\alpha$ *and* $l_\alpha := \bigwedge_{\beta=1}^{k_\alpha} u_{i_{\alpha,\beta}}^{(b_{\alpha,\beta})}$*, where the superscript bit denotes negation or not, i.e.* $u_{i_{\alpha,\beta}}^{(0)} = u_{i_{\alpha,\beta}}$ *and* $u_{i_{\alpha,\beta}}^{(1)} = \neg u_{i_{\alpha,\beta}}$. *For each* $\alpha \in [\![|OR|]\!]$*, compute* $\widetilde{\mathsf{ct}}_i \leftarrow \mathsf{WE}^{\mathsf{AND}}.\mathsf{Enc}(\widetilde{\mathsf{ed}}, l_\alpha, \mathsf{msg})$. *Output* $\mathsf{ct} = (\mathcal{P}, \widetilde{\mathsf{ct}}_1, \ldots, \widetilde{\mathsf{ct}}_{|OR|})$.

- $\texttt{Dec}(\mathsf{sk}, \mathsf{hsk}, \mathsf{ct})$: *Parse* $\mathsf{ct} := (\mathcal{P}, \widetilde{\mathsf{ct}}_1, \ldots, \widetilde{\mathsf{ct}}_k)$, $\mathcal{P} := \bigvee_{\alpha=1}^{|OR|} \bigwedge_{\beta=1}^{k_\alpha} u_{i_{\alpha,\beta}}^{(b_{\alpha,\beta})}$ *and* $\mathsf{hsk} := \left(\mathcal{U}, \mathsf{hsk}_0, \mathsf{hsk}'_0, \{\mathsf{hsk}_1^{(b)}, \ldots, \mathsf{hsk}_n^{(b)}\}_{b \in \{0,1\}} \right)$. *Compute*

$$\begin{aligned}
(\bot, \mathsf{aux}^{\mathsf{DLOG}}) &\leftarrow \mathsf{WE}^{\mathsf{DLOG}}.\texttt{AuxGen}(\mathsf{crs}, \mathsf{sk}) \\
\mathsf{dd}^{\mathsf{DLOG}} &\leftarrow \mathsf{WE}^{\mathsf{DLOG}}.\texttt{Digest}(\mathsf{crs}, \bot, \mathsf{aux}^{\mathsf{DLOG}}) \\
\pi_{\mathsf{DLOG}} &\leftarrow \mathsf{WE}^{\mathsf{DLOG}}.\texttt{Prove}(\mathsf{dd}^{\mathsf{DLOG}}, \bot, (\mathsf{pk}, \mathsf{sk}))
\end{aligned}$$

[12]In our concrete instantiation, the NIZK proof can be avoided as $\mathsf{aux}^{\mathsf{IIP}}$ is publicly checkable using pairings.

and for each $\alpha \in [\![|OR|]\!]$ *set* $\pi_\alpha = (\pi_{\mathsf{DLOG}}, \mathsf{hsk}_0, \mathsf{hsk}'_0, \{\mathsf{hsk}_{i_{\alpha,\beta}}^{(b_{\alpha,\beta})}\}_{\beta \in [\![k_\alpha]\!]})$.

If there is no clause $\alpha \in [\![k_\alpha]\!]$ *that is satisfied by* $\mathcal{U}$ *then output* $\mathsf{msg} = \bot$. *Otherwise, assume the clause* l_{α^*} *is satisfied by the* $\mathcal{U}$ *attributes then compute:* $\mathsf{msg} \leftarrow \mathsf{WE}^{\mathsf{AND}}.\mathtt{Dec}(\mathsf{ct}, \pi_{\alpha^*})$

Security. For the security of our construction we merely have to show that AND is a suitable relation for R-ABE.

We assume that our underlying witness encryption schemes are secure. We will need the additional property that $\mathsf{WE}^{\mathsf{IIP}}.\mathtt{AuxGen}$ is 'hard-to-invert'. This is, essentially, the most natural property for any public-key generation algorithm. Roughly speaking, it means that no PPT adversary, given a list of $\mathsf{pk}_1 = \mathsf{WE}^{\mathsf{IIP}}.\mathtt{AuxGen}(\mathsf{crs}, \mathsf{sk}_i), \dots, \mathsf{pk}_n = \mathsf{WE}^{\mathsf{IIP}}.\mathtt{AuxGen}(\mathsf{crs}, \mathsf{sk}_n)$ and an (adaptive) inversion oracle for pk's (returning the corresponding sk), can compute an sk' such that $\mathsf{pk}_i = \mathsf{WE}^{\mathsf{IIP}}.\mathtt{AuxGen}(\mathsf{crs}, \mathsf{sk}')$ for any $i \in [\![n]\!]$ (so long as pk_i inversion was not queried). For a formal treatment and proof of the property for $\mathsf{WE}^{\mathsf{IIP}}$ we refer to Section 5.4.

Theorem 2. *Assume that* $\mathsf{WE}^{\mathsf{IIP}}$, $\mathsf{WE}^{\mathsf{NonZero}}$, $\mathsf{WE}^{\mathsf{DLOG}}$ *and* $\mathsf{WE}^{\widehat{\mathsf{Zero}}}$ *are secure Witness Encryption schemes and* $\mathsf{WE}^{\mathsf{IIP}}.\mathtt{AuxGen}$ *is hard to invert, then construction 7 is a secure Slotted Registered Attribute-Based Encryption scheme.*

Proof. Let $\mathcal{A}$ be an admissible adversary that wins the security of R-ABE game with non-negligible probability ϵ. We construct an adversary $\mathcal{B}$ that inverts $\mathsf{WE}^{\mathsf{IIP}}.\mathtt{AuxGen}$ with non-negligible probability.

Adversary $\mathcal{B}$ interacts as a challenger with $\mathcal{A}$ in the R-ABE security game and with a $\mathsf{WE}^{\mathsf{IIP}}.\mathtt{AuxGen}$-inversion challenger $\mathcal{C}$.

1. **Setup Phase.** $\mathcal{B}$ runs the setup phase of R-ABE and sends crs to $\mathcal{A}$.

2. **Query Phase.** $\mathcal{B}$ proceeds as follows:

   (a) For each honest key generation query, for slot i, $\mathcal{B}$ asks $\mathcal{C}$ for a $\mathsf{pk}_i = (\widetilde{\mathsf{pk}}_i, \mathsf{aux}_i^{\mathsf{IIP}}) := \mathsf{WE}^{\mathsf{IIP}}.\mathtt{AuxGen}(\mathsf{crs}, \cdot)$ and sends pk_i to $\mathcal{A}$.

   (b) For each key corruption query, of pk_i, $\mathcal{B}$ makes an inversion query to $\mathcal{C}$ to obtain the corresponding sk_i such that $\mathsf{pk}_i = \mathsf{WE}^{\mathsf{IIP}}.\mathtt{AuxGen}(\mathsf{crs}, \mathsf{sk}_i)$ and sends it to $\mathcal{A}$.

Let $\mathsf{ct}^* := (\mathcal{P}^*, \mathsf{ct}_1^*, \dots, \mathsf{ct}_{|\mathrm{OR}|}^*)$ be the challenge ciphertext. Since $\mathsf{ct}_1^*, \dots, \mathsf{ct}_{|\mathrm{OR}|}^*$ are computed independently there is at least one position α where $\mathcal{A}$ can distinguish. Choose one $\alpha \in [\![|\mathrm{OR}|]\!]$ at random. With probability $1/|\mathrm{OR}|$ it is the one that $\mathcal{A}$ can distinguish with probability ϵ.

Then there is an extractor that with probability $\epsilon/|\mathrm{OR}|$ outputs $\boldsymbol{w}, \mathsf{pk}, \mathsf{sk}$ such that:

1. $\langle (\mathsf{pk}_1, \dots, \mathsf{pk}_m), \boldsymbol{w} \rangle = \mathsf{pk}$

2. $\boldsymbol{w} \neq \mathbf{0}$

3. $\mathsf{pk} = [\mathsf{sk}]_1$

4. $w_j = 0$ in $j \in \mathcal{J}_{u_{i_{\alpha,\beta}}^{(b_{\alpha,\beta})}}$, for each $\beta \in [\![k_\alpha]\!]$

Recall that

$$\mathcal{J}_{u_{i_{\alpha,\beta}}^{(b_{\alpha,\beta})}} = \begin{cases} [\![m]\!] \setminus \mathcal{I}_{u_{i_{\alpha,\beta}}}, & \text{if } b_{\alpha,\beta} = 0 \text{ and } u_{i_{\alpha,\beta}} \in \mathcal{U}_{\mathsf{eff}} \\ \mathcal{I}_{u_{i_{\alpha,\beta}}}, & \text{if } b_{\alpha,\beta} = 1 \text{ and } u_{i_{\alpha,\beta}} \in \mathcal{U}_{\mathsf{eff}} \\ [\![m]\!], & \text{if } b_{\alpha,\beta} = 0 \text{ and } u_{i_{\alpha,\beta}} \notin \mathcal{U}_{\mathsf{eff}} \\ \emptyset & \text{if } b_{\alpha,\beta} = 1 \text{ and } u_{i_{\alpha,\beta}} \notin \mathcal{U}_{\mathsf{eff}} \end{cases}$$

By definition of $\mathcal{J}_{u_{i_{\alpha,\beta}}^{(b_{\alpha,\beta})}}$, $w_j = 0$ exactly in all the positions that the corresponding attribute set $\mathcal{U}_j$ does not satisfy the clause $\bigwedge_{\beta=1}^{k_\alpha} u_{i_{\alpha,\beta}}^{(b_{\alpha,\beta})}$ (or, in other words, $j \notin \mathcal{I}_{u_{i_{\alpha,\beta}}^{(b_{\alpha,\beta})}}$ for some $\beta \in [\![k_a]\!]$). Also, by definition of an admissible adversary in the R-ABE game, none of the corrupted keys $\mathsf{pk}_i \in \mathsf{Cor}$ of the adversary $\mathcal{A}$ satisfy the target policy $\mathcal{P}^*$. Furthermore, $\boldsymbol{w} \neq \mathbf{0}$ excludes the degenerate case of $\mathsf{pk} = [0]_1, \mathsf{sk} = 0$, that could be computed by $\mathcal{B}$ without an inversion query. This means that the extracted pk is a result of an honest key generation query and therefore, sk has not been asked to $\mathcal{C}$ for inversion by $\mathcal{B}$.

Finally, $\mathcal{B}$ outputs sk inverting $\mathsf{WE}^{\mathsf{IIP}}.\mathtt{AuxGen}$ with probability $\epsilon/|\mathrm{OR}|$. □

7 Registered Threshold Encryption

In this section, we formally introduce and provide a construction of Registered Threshold Encryption (RTE) with succinct ciphertexts. First we give the formal definitions of RTE, revisiting the definition of Branco et al. [BLM+24], and then we present our construction, using our WE framework. For completeness we present the unwrapped construction in the Section B.2.

7.1 Definition

Here we provide the formal definitions of Registered Threshold Encryption (RTE). We revisit the definition of RTE of Branco et al. [BLM+24] as follows:

- We allow for dynamic choice of the threshold t, i.e. t is chosen during encryption time,
- we formally require ciphertext compactness, i.e. the ciphertext should grow at most polylogarithmically in t,
- we allow for identity-based decryption committees, i.e. each user is registered with an identity id.

For security of RTE, we provide semantic security definitions. Our definitions can be seen as an adaptation of Silent Threshold Encryption [GKPW24] to the registered setting [GHMR18]. First, we begin with the syntax of RTE.

Definition 19 (Registered Threshold Encryption)**.** *A Registered Threshold Encryption scheme with identity space $\mathbb{ID}$ consists of the following PPT algorithms* ($\mathtt{Setup}, \mathtt{KGen}, \mathtt{isValid}, \mathtt{Reg}, \mathtt{Enc}, \mathtt{PartDec}, \mathtt{PartVerify}, \mathtt{DecAggr}$)*.*

- $\mathtt{Setup}(1^\lambda, M) \to \mathsf{crs}$ *On input the security parameter λ and a maximum number of users $M > 0$, the randomized setup algorithm samples a common reference string* crs*. The message space $\mathbb{M}$ is also specified here and is (implicitly) contained in the* crs*.*
- $\mathtt{KGen}(\mathsf{crs}) \to (\mathsf{hint}, \mathsf{pk}, \mathsf{sk})$: *On input the common reference string* crs*, the randomized key generation algorithm outputs a public and secret key together the corresponding hint,* $(\mathsf{hint}, \mathsf{pk}, \mathsf{sk})$*.*
- $\mathtt{isValid}(\mathsf{crs}, \mathsf{pk}, \mathsf{hint}) \to \{0, 1\}$: *On input the common reference string* crs*, a public key* pk *with a corresponding hint* hint*, the deterministic hint verification algorithm outputs a bit, which is* 1 *if* hint *is well-formed.*
- $\mathtt{Reg}(\mathsf{crs}, \mathsf{aux}, \mathsf{id}, \mathsf{pk}, \mathsf{hint}) \to (\mathsf{mpk}', \mathsf{aux}')$: *On input the common reference string* crs*, the current master public key* mpk*, auxiliary information* aux*, an identity* id*, a public key* pk *with a corresponding hint* hint *the deterministic registration algorithm outputs the new master public key* mpk' *and auxiliary information* aux'*. (The system is initialized with a master public key* mpk_0 *and auxiliary information* $\mathsf{aux} \leftarrow \perp$*).*

- $\texttt{Enc}(\mathsf{mpk}, \mathcal{S}, t, \mathsf{msg}) \to \mathsf{ct}$: *On input the current public parameters* mpk, *a committee of decryptors* $\mathcal{S} \subseteq \mathbb{ID}$, *a threshold* t *and a message* $\mathsf{msg} \in \mathbb{M}$, *the randomized encryption algorithm outputs a ciphertext* ct.

- $\texttt{PartDec}(\mathsf{sk}, \mathsf{ct}) \to \sigma$: *On input a secret* sk *and a ciphertext* ct, *the deterministic partial decryption algorithm outputs a partial decryption* σ.

- $\texttt{PartVerify}(\mathsf{ct}, \sigma, \mathsf{pk}) \to \{0, 1\}$: *On input a ciphertext* ct, *a partial decryption* σ *and a public key* pk, *the deterministic partial decryption verification algorithm outputs a bit, which is* 1 *if the partial decryption is valid.*

- $\texttt{DecAggr}(\mathsf{crs}, \mathsf{aux}, \mathcal{S}, \{\sigma_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{D}}, \mathsf{ct}) \to \mathsf{msg}$: *On input the common reference string* crs, *the auxiliary information* aux, *the committee of decryptors* $\mathcal{S}$ *of the ciphertext, a set* $\{\sigma_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{D}}$ *of partial decryptions together and a ciphertext* ct, *the deterministic decryption algorithm outputs a message* $\mathsf{msg} \in \mathbb{M} \cup \{\bot\}$. *The symbol* $\bot$ *indicates a syntax error.*

We proceed to the definition of correctness and the efficiency properties. For efficiency, we, first, require the threshold encryption ciphertexts and the partial decryption values to be succinct in the number of decryptors. This makes the threshold encryption non-trivial. Furthermore, for the registration procedure to be meaningful we require that the master public key, used for encryption, should be succinct in the number of registered users in the system.

Definition 20 (Correctness/Efficiency of Registered Threshold Encryption)**.** *For any unbounded adversary* $\mathcal{A}$, *consider the following game between an adversary* $\mathcal{A}$ *and a challenger* $\mathcal{C}$, *parameterized by* λ *and a bound* $M = \mathsf{poly}(\lambda)$.

1. **Setup phase.** $\mathcal{C}$ *samples* $\mathsf{crs} \leftarrow \texttt{Setup}(1^\lambda, M)$ *and initializes* $\mathsf{mpk} \leftarrow \mathsf{mpk}_0$, $\mathsf{aux} \leftarrow \bot$. *It also initializes counters* $N \leftarrow 0$, $E \leftarrow 0$ *for the number of registration and encryption queries respectively,* $\mathcal{U} \leftarrow \emptyset$ *for the set of registered identities,* $\mathsf{Cor} \leftarrow \emptyset$ *for the set of corrupted identities and* $\mathcal{S}^* = \bot$, $t^* = \bot$ *for the target committee and threshold respectively. It also keeps a map* $\mathcal{D}$ *of the effective identities during encryption (implicitly initialized to* $\mathcal{D}[i] = \emptyset$ *for every* i*).* $\mathcal{C}$ *sends* crs *to* $\mathcal{A}$.

2. **Query phase.** $\mathcal{A}$ *chooses one of the following queries.*

   (a) **Registering new corrupted identity query.** *If* $N = M$ *skip this step. Otherwise,* $\mathcal{A}$ *chooses an identity* $\mathsf{id} \in \mathbb{ID} \setminus \mathcal{U}$ *together with a public key* pk *and a hint* hint *and sends* $(\mathsf{id}, \mathsf{pk}, \mathsf{hint})$ *to* $\mathcal{C}$ *for registration. If* $\texttt{isValid}(\mathsf{crs}, \mathsf{pk}, \mathsf{hint}) = 0$ *skip this step. Otherwise,* $\mathcal{C}$ *updates the counter* $N \leftarrow N + 1$ *and the sets* $\mathcal{U} \leftarrow \mathcal{U} \cup \{\mathsf{id}\}$ *and* $\mathsf{Cor} \leftarrow \mathsf{Cor} \cup \{\mathsf{id}\}$ *and then registers* $(\mathsf{id}, \mathsf{pk}, \mathsf{hint})$ *computing* $(\mathsf{mpk}', \mathsf{aux}') \leftarrow \texttt{Reg}(\mathsf{crs}, \mathsf{aux}, \mathsf{id}, \mathsf{pk}, \mathsf{hint})$ *and updating* $\mathsf{mpk} \leftarrow \mathsf{mpk}'$, $\mathsf{aux} \leftarrow \mathsf{aux}'$. $\mathcal{C}$ *sends* $(N, \mathsf{mpk}, \mathsf{aux})$ *to* $\mathcal{A}$.

   (b) **Registering new honest identity query.** *If* $N = M$ *skip this step.* $\mathcal{A}$ *chooses an identity* $\mathsf{id} \in \mathbb{ID} \setminus \mathcal{U}$ *and sends it to* $\mathcal{C}$ *for registration.* $\mathcal{C}$ *samples* $(\mathsf{hint}, \mathsf{pk}, \mathsf{sk}) \leftarrow \texttt{KGen}(\mathsf{crs})$, *updates the counter* $N \leftarrow N + 1$ *and the set* $\mathcal{U} \leftarrow \mathcal{U} \cup \{\mathsf{id}\}$ *and then registers* $(\mathsf{id}, \mathsf{pk}, \mathsf{hint})$ *computing* $(\mathsf{mpk}', \mathsf{aux}') \leftarrow \texttt{Reg}(\mathsf{crs}, \mathsf{aux}, \mathsf{id}, \mathsf{pk}, \mathsf{hint})$ *and updating* $\mathsf{mpk} \leftarrow \mathsf{mpk}'$, $\mathsf{aux} \leftarrow \mathsf{aux}'$. $\mathcal{C}$ *sends* $(N, \mathsf{mpk}, \mathsf{aux}, \mathsf{pk}, \mathsf{hint})$ *to* $\mathcal{A}$.

   (c) **Encryption query.** $\mathcal{A}$ *sends to* $\mathcal{C}$ *a target committee set* $\mathcal{S}$, *a target threshold* t *and a message* $\mathsf{msg} \in \mathbb{M}$. *If* $\mathcal{S}^* \neq \bot \wedge \mathcal{S} \neq \mathcal{S}^*$ *or* $t^* \neq \bot \wedge t \neq t^*$ *then skip (*$\mathcal{A}$ *can only choose one target committee and threshold throughout the game), otherwise* $\mathcal{C}$ *sets* $\mathcal{S}^* \leftarrow S$ *and* $t^* \leftarrow t$. *If* $|\mathcal{U} \cap \mathcal{S}| < t$ *(i.e. not at least* t *identities of* $\mathcal{S}$ *are registered) then skip. Otherwise,* $\mathcal{C}$ *updates* *the counter $E \leftarrow E+1$, sets $\mathcal{D}[E] \leftarrow \mathcal{U} \cap \mathcal{S}^*$ (the registered identities of the committee during encryption) and computes $\mathsf{ct}_E \leftarrow \mathtt{Enc}(\mathsf{mpk}, \mathcal{S}^*, t^*, \mathsf{msg})$ and sends back (E, ct_E) to $\mathcal{A}$.*

(d) **Partial decryption query.** *$\mathcal{A}$ send to $\mathcal{C}$ an index of a ciphertext $j \in [\![E]\!]$ and an identity id. If $\mathsf{id} \notin \mathcal{D}[j]$ (id was not registered during encryption) or $\mathsf{id} \in \mathsf{Cor}$ then $\mathcal{C}$ sends $\perp$ to $\mathcal{A}$. Otherwise, $\mathcal{C}$ computes the partial decryption $\sigma \leftarrow \mathtt{PartDec}(\mathsf{sk}, \mathsf{ct}_j)$, where sk is the corresponding secret key of id ($\mathcal{C}$ has it since id was an honest identity registration). $\mathcal{C}$ send σ to $\mathcal{A}$.*

(e) **Decryption query.** *$\mathcal{A}$ sends to $\mathcal{C}$ an index $j \in [\![E]\!]$ of a ciphertext together with a set Σ with partial decryptions $\{\sigma_{\mathsf{id}}\}_{\mathsf{id} \in \Sigma}$. If $|\Sigma| < t^*$ or $\mathsf{id} \notin \mathcal{D}[j]$ (id was not registered during encryption) for any of the $\mathsf{id} \in \Sigma$ skip this step. If there exists $(\mathsf{id}, \sigma_{\mathsf{id}}) \in \Sigma$ such that $\mathtt{PartVerify}(\mathsf{ct}_j, \sigma_{\mathsf{id}}, \mathsf{pk}_{\mathsf{id}}) = 0$ skip this step. $\mathcal{C}$ computes $\widetilde{\mathsf{msg}}_j \leftarrow \mathtt{DecAggr}(\mathsf{crs}, \mathsf{aux}, \mathcal{S}^*, \{\sigma_{\mathsf{id}}\}_{\mathsf{id} \in \Sigma}, \mathsf{ct})$.*

3. *After the query phase, the adversary $\mathcal{A}$ wins the game if there exists $j \in [\![E]\!]$ such that $\widetilde{\mathsf{msg}}_j \neq \mathsf{msg}_j$.*

Let $Q \in \mathsf{poly}(\lambda)$ be an upper bound on the number of queries issued by $\mathcal{A}$. We require the following properties to hold for any adversary $\mathcal{A}$.

- **Correctness.** $\Pr[\mathcal{A}$ *wins*$] = \mathsf{negl}(\lambda)$.
- **Master Public Key Compactness.** *For all queries $q \in [\![Q]\!]$, let mpk_q be the public parameters and N_q the registration counter respectively after the q-th query. Then for all $q \in [\![Q]\!]$, $|\mathsf{mpk}_q| = \mathsf{poly}(\lambda, \log(N_q))$.*
- **Ciphertext Compactness.** *For every $j \in [\![E]\!]$, $|\mathsf{ct}| = \mathsf{poly}(\lambda, \log(t^*))$.*
- **Partial Decryption Compactness.** *For every $j \in [\![E]\!]$ and every $\mathsf{id} \in \mathcal{D}[j]$, $|\sigma_{\mathsf{id}}| = \mathsf{poly}(\lambda, \log(|\mathcal{S}^*|))$.*

We define security for Registered Threshold Encryption with adaptive corruptions.

Definition 21 (Adaptive Security of Registered Threshold Encryption)**.** *Consider the following game between a PPT adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.*

1. **Setup Phase.** *$\mathcal{C}$ initializes $\mathsf{mpk} \leftarrow \mathsf{mpk}_0$, $\mathsf{aux} \leftarrow \perp$ and samples $\mathsf{crs} \leftarrow \mathtt{Setup}(1^\lambda, M)$. It also initializes a counter N for the number overall of registration queries, a set $\mathsf{Cor} \leftarrow \emptyset$ for the corrupted identities and a set $\mathcal{H} \leftarrow \emptyset$ for set of honest identities. $\mathcal{C}$ sends crs to $\mathcal{A}$.*

2. **Query Phase.** *$\mathcal{A}$ chooses one of the following queries.*

(a) **Registering corrupted key query.** *If $N = M$ skip this step. Otherwise, $\mathcal{A}$ chooses a pk together with a hint hint and $\mathsf{id} \in \mathbb{ID}$ and sends $(\mathsf{id}, \mathsf{pk}, \mathsf{hint})$ to $\mathcal{C}$. $\mathcal{C}$ registers $(\mathsf{id}, \mathsf{pk}_{\mathsf{id}}, \mathsf{hint}_{\mathsf{id}})$ computing $(\mathsf{mpk}', \mathsf{aux}') \leftarrow \mathtt{Reg}(\mathsf{crs}, \mathsf{aux}, \mathsf{id}, \mathsf{pk}_{\mathsf{id}}, \mathsf{hint}_{\mathsf{id}})$. Then updates $\mathsf{mpk} \leftarrow \mathsf{mpk}'$, $\mathsf{aux} \leftarrow \mathsf{aux}'$, $\mathsf{Cor} \leftarrow \mathsf{Cor} \cup \{\mathsf{id}\}$ and $N \leftarrow N+1$. $\mathcal{C}$ sends $(\mathsf{mpk}, \mathsf{aux})$ to $\mathcal{A}$.*

(b) **Registering honest key query.** *If $N = M$ skip this step. Otherwise, $\mathcal{A}$ chooses an identity $\mathsf{id} \subseteq \mathbb{ID}$ and sends it to $\mathcal{C}$. $\mathcal{C}$ samples $(\mathsf{hint}_{\mathsf{id}}, \mathsf{pk}_{\mathsf{id}}, \mathsf{sk}_{\mathsf{id}}) \leftarrow \mathtt{KGen}(\mathsf{crs}, \mathsf{aux})$ and registers the identity computing $(\mathsf{mpk}', \mathsf{aux}') \leftarrow \mathtt{Reg}(\mathsf{crs}, \mathsf{aux}, \mathsf{id}, \mathsf{pk}_{\mathsf{id}}, \mathsf{hint}_{\mathsf{id}})$. Then updates $\mathsf{mpk} \leftarrow \mathsf{mpk}'$, $\mathsf{aux} \leftarrow \mathsf{aux}'$, $\mathcal{H} \leftarrow \mathcal{H} \cup \{\mathsf{id}\}$ and $N \leftarrow N+1$. $\mathcal{C}$ sends $(\mathsf{mpk}, \mathsf{aux})$ to $\mathcal{A}$.*

(c) **Corrupt honest key query.** *$\mathcal{A}$ sends an identity id to $\mathcal{C}$. If $\mathsf{id} \notin \mathcal{H}$ then skip this step. $\mathcal{C}$ adds*

id *to the corrupted identities,* $\mathsf{Cor} \leftarrow \mathsf{Cor} \cup \{\mathsf{id}\}$ *and sends back to* $\mathcal{A}$ *the corresponding secret key* $\mathsf{sk}_{\mathsf{id}}$.

3. **Challenge phase.** *$\mathcal{A}$ chooses a set $\mathcal{S}^* \subseteq \mathbb{ID}$, a threshold t^* and two messages* $(\mathsf{msg}_0, \mathsf{msg}_1)$ *and sends them to $\mathcal{C}$. $\mathcal{C}$ generates* $\mathsf{ct}^* \leftarrow \mathtt{Enc}(\mathsf{mpk}, \mathcal{S}^*, t^*, \mathsf{msg}_b)$*, where $b \leftarrow\!\!\$\ \{0,1\}$ is a randomly chosen bit, and sends* ct^* *to $\mathcal{A}$.*

4. **Output.** *The adversary $\mathcal{A}$ outputs a bit b'.*

An adversary is admissible if $|\mathcal{S}^* \cap \mathsf{Cor}| < t^*$. *We call an RTE scheme secure if for all PPT admissible adversaries $\mathcal{A}$ it holds that* $\Pr[b = b'] \leqslant \frac{1}{2} + \mathsf{negl}(\lambda)$.

7.2 Our Registered Threshold Encryption Construction

In this section we present our Registered Threshold Encryption construction. In the following we will denote by:

- M: The maximum number of identities that can register in the system (excluding the dummy $\mathsf{id} = 1$).
- $\mathcal{U}$: The set of identities are registered in the system, $\mathcal{U} = \{1, \mathsf{id}_1, \ldots, \mathsf{id}_N\}$. For notational convenience, we include the dummy party in $\mathcal{U}$ and, further, denote $|\mathcal{U}| := N + 1$.
- $\mathcal{S}$: The committee of identities that can participate in the decryption of a certain ciphertext (excluding the dummy $\mathsf{id} = 1$.
- $\mathcal{D}$: The actual members of the committee $\mathcal{S}$ that provided a partial decryption (excluding the dummy $\mathsf{id} = 1$).
- $\mathcal{D}_V$: The identities that provided a valid partial decryption (excluding the dummy $\mathsf{id} = 1$).

Clearly, $\mathcal{D}_V \subseteq \mathcal{D} \subseteq \mathcal{S} \subseteq \mathcal{U} \setminus \{1\}$.

We first provide a construction of Registered Threshold Encryption assuming a Witness Encryption over the following relation. Later, we will show how to actually build such a witness encryption scheme.

Definition 22 (Aggregate BLS Threshold Signature Relation). *We define the aggregate BLS threshold signature relation* tBLS *over public parameter, index, instance, witness tuples as follows.*

$$\mathsf{tBLS} = \left\{ (([1]_1), (\{\mathsf{pk}_i\}_{i\in\mathcal{U}}), (\mathcal{S}, t, \mathsf{tg}), (\sigma^*, \boldsymbol{w})) \;\middle|\; \begin{array}{l} \{\mathsf{pk}_i \in \mathbb{G}_2\}_{i\in\mathcal{U}}, \boldsymbol{w} \in \mathbb{F}^n, t \in [\![|\mathcal{S}|]\!] \\ \deg\left(\sum_{i\in[\![|\mathcal{U}|]\!]} w_i \cdot L_i(X)\right) \leqslant |\mathcal{U}| - t \\ \boldsymbol{w}_{\mathcal{U}\setminus\mathcal{S}} = 0, w_1 = 1 \\ g \circ \sigma^* = \left(\sum_{i\in\mathcal{U}} w_i \cdot \mathsf{pk}_i\right) \circ H(\mathsf{tg}) \end{array} \right\},$$

where $H : \{0,1\}^* \to \mathbb{G}_2$ is a random oracle, and $\{L_i\}_{i\in\mathcal{U}}$ are the Lagrange polynomials over the domain $\mathcal{U}$.

Before presenting the construction for registered threshold encryption (rTE), we examine the tBLS relation more closely. In plain english, the relation demand that *You know an aggregate signature σ^* on* tg *under some public key* pk^**, where* pk^* *is an aggregation of at least t different public keys from the set* $\{\mathsf{pk}_i\}_{i\in\mathcal{U}}$.

We are now ready to describe the construction. All users sample their secret key and run `AuxGen` before registering their public key, auxiliary information and corresponding identity. After this the public keys of all users are *hashed* down to a short digest by running the `Digest` method which generates encryption and decryption digests ed/dd. In our construction ed will be constant sized and this is the only information required to encrypt a message to any set of users in the system such that any t parties can non-interactively decrypt the ciphertext. The encryption algorithm of RTE will choose a random tg and run the witness encryption's Enc algorithm.

Now coming to partial decryptions, observe that the witness to decrypt is a signature under an aggregated public key of at least t parties. But to produce such a signature, we require signatures from at least t different parties on tg. Thus it is natural to make the signatures themselves the partial decryptions. Given enough of them, they can be publicly aggregated to recover the message by using the Dec algorithm.

Construction 8 (Registered Threshold Encryption)**.** *Let* $\mathsf{WE}^{\mathsf{tBLS}}$ *be a witness encryption scheme for the relation* tBLS *of Definition 22. We construct a Registered Threshold Encryption scheme with identity space* $\mathbb{ID} = \mathbb{F}$ *as follows:*

- $\mathtt{Setup}(1^\lambda, M)$: *Sample* $\mathsf{td} \leftarrow\!\$\ \mathbb{F}$*, and compute* $\widetilde{\mathsf{crs}} \leftarrow \mathsf{WE}^{\mathsf{tBLS}}.\mathtt{Gen}(\lambda, M, \mathsf{td})$*. Output* $\mathsf{crs} = \widetilde{\mathsf{crs}}$*.*
- $\mathtt{KGen}(\mathsf{crs})$ *Sample* $\widetilde{\mathsf{sk}} \leftarrow\!\$\ \mathbb{F}$ *and run* $(\widetilde{\mathsf{idx}}, \widetilde{\mathsf{aux}}) \leftarrow \mathsf{WE}^{\mathsf{tBLS}}.\mathtt{AuxGen}(\mathsf{crs}, \widetilde{\mathsf{sk}})$*. Output* $\mathsf{sk} = \widetilde{\mathsf{sk}}$*,* $\mathsf{pk} = \widetilde{\mathsf{idx}}$*,* $\mathsf{hint} = (\widetilde{\mathsf{aux}}, \pi)$*, where* π *is NIZK proof of correct execution of* $\mathsf{WE}^{\mathsf{tBLS}}.\mathtt{AuxGen}(\mathsf{crs}, \widetilde{\mathsf{sk}})$*.*
- $\mathtt{isValid}(\mathsf{crs}, \mathsf{pk}, \mathsf{hint})$: *Check that the NIZK proof* π *verifies.*[13]
- $\mathtt{Reg}(\mathsf{crs}, \mathsf{aux}, \mathsf{id}, \mathsf{pk}, \mathsf{hint})$: *Parse* $\mathsf{aux} := (\mathcal{U}, \{\mathsf{pk}_i, \mathsf{hint}_i\}_{i\in\mathcal{U}})$*. Run* $(\widetilde{\mathsf{ed}}, \widetilde{\mathsf{dd}}) \leftarrow \mathsf{WE}^{\mathsf{tBLS}}.\mathtt{Digest}(\mathsf{crs}, \{\mathsf{pk}_i\}_{i\in\mathcal{U}} \cup \{\mathsf{pk}\}, \{\mathsf{hint}_i\}_{i\in\mathcal{U}} \cup \{\mathsf{hint}\})$*. Output* $\mathsf{mpk}' = \mathsf{ed}$*,* $\mathsf{aux}' = (\mathcal{U}', \{\mathsf{pk}_i, \mathsf{hint}_i\}_{i\in\mathcal{U}} \cup \{\mathsf{pk}, \mathsf{hint}\})$
- $\mathtt{Enc}(\mathsf{mpk}, \mathcal{S}, t, \mathsf{msg})$: *Sample* $\mathsf{tg} \leftarrow \{0,1\}^\lambda$*, compute* $\widetilde{\mathsf{ct}} \leftarrow \mathsf{WE}^{\mathsf{tBLS}}.\mathtt{Enc}(\mathsf{mpk}, (\mathcal{S}, t, \mathsf{tg}), \mathsf{msg})$ *and output* $\mathsf{ct} = (\widetilde{\mathsf{ct}}, \mathsf{tg})$*.*
- $\mathtt{PartDec}(\mathsf{sk}, \mathsf{ct})$: *Parse* $\mathsf{ct} := (\widetilde{\mathsf{ct}}, \mathsf{tg})$ *and output* $\sigma \leftarrow \mathsf{sk} \cdot H(\mathsf{tg})$
- $\mathtt{PartVerify}(\mathsf{ct}, \sigma, \mathsf{pk})$: *Parse* $\mathsf{ct} := (\widetilde{\mathsf{ct}}, \mathsf{tg})$*. If* $[1]_1 \circ \sigma = \mathsf{pk} \circ H(\mathsf{tg})$ *output* 1*, else output* 0*.*
- $\mathtt{DecAggr}(\mathsf{crs}, \mathsf{aux}, \mathcal{S}, \{\sigma_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{D}}, \mathsf{ct}, \mathsf{tg})$: *Compute the polynomial* $B(X)$ *that interpolates* $B(1) = 1$ *and* $\{B(i) = 0\}_{i\in\mathcal{U}\setminus\mathcal{D}}$*, and set* $\{w_i = B(i)\}_{i\in\mathcal{U}}$ *and* $\sigma^* \leftarrow \sum_{i\in\mathcal{D}} w_i \cdot \sigma_i$*. Run* $\pi \leftarrow \mathsf{WE}^{\mathsf{tBLS}}.\mathtt{Prove}(\mathsf{aux}, (\mathcal{S}, t, \mathsf{tg}), (\sigma^* \boldsymbol{w}))$*. Finally, recover the message* $\widetilde{\mathsf{msg}} \leftarrow \mathsf{WE}^{\mathsf{tBLS}}.\mathtt{Dec}(\mathsf{ct}, \pi)$ *and output* $\mathsf{msg} = \widetilde{\mathsf{msg}}$*.*

Theorem 3. *Let* $\mathsf{WE}^{\mathsf{tBLS}}$ *be a secure witness encryption scheme for the relation* tBLS*, then Construction 8 is a secure Registered Threshold Encryption scheme.*

Proof Sketch. Let $\mathcal{A}$ be an admissible PPT adversary of the RTE security game. We construct an adversary $\mathcal{B}$ that, interacting with $\mathcal{A}$ in the RTE security game, breaks the security of BLS multi-signatures.

$\mathcal{B}$ initializes the crs according to the protocol and sends it to $\mathcal{A}$ and simulates honestly (according to the protocol) all the queries of $\mathcal{A}$. Let $\mathcal{S}^*, t^*$ be the target set and threshold respectively and Cor and $\mathcal{H}$ the final sets of honest and corrupted identities. Let $\mathsf{ct}^* := (\widetilde{\mathsf{ct}}^*, \mathsf{tg}^*)$ be the challenge ciphertext, where $\widetilde{\mathsf{ct}} = \mathsf{WE}^{\mathsf{tBLS}}.\mathtt{Enc}(\mathsf{mpk}, (\mathcal{S}^*, t^*, \mathsf{tg}), \mathsf{msg}_b)$. Since $\mathcal{A}$ is an admissible adversary $|\mathcal{S}^* \cap \mathsf{Cor}| < t^*$.

Let $\mathcal{A}$ have probability ϵ of winning the RTE security game. This means that $\mathcal{A}$ is able to distinguish a ciphertext for msg_0 and msg_1 of $\mathsf{WE}^{\mathsf{tBLS}}$ for $\mathsf{idx} := (\mathsf{pk}_i)_{i\in\mathsf{Cor}} \cup (\mathsf{pk}_i)_{i\in\mathcal{H}}$, $u := (\mathcal{S}^*, t^*, \mathsf{tg}^*)$. Therefore, from the knowledge soundness of $\mathsf{WE}^{\mathsf{tBLS}}$ there is an extractor that with probability ϵ outputs σ^*, w such that:

1. $\deg\left(\sum_{i\in\mathcal{U}} w_i \cdot L_i(X)\right) \leqslant |\mathcal{U}| - t$
2. $\boldsymbol{w}_{\mathcal{U}\setminus\mathcal{S}^*} = 0, w_1 = 1$
3. $g \circ \sigma^* = \left(\sum_{i\in\mathcal{U}} w_i \cdot \mathsf{pk}_i\right) \circ H(\mathsf{tg})$

[13]In our concrete instantiation we do not need a separate NIZK proof as the aux is publicly verifiable for well-formedness.

Let the polynomial $B(X) = \sum_{i \in \mathcal{U}} w_i L_i(X)$, from (2) we get that $B(X) = L_1(X) + \sum_{i \in \mathcal{S}^*} w_i L_i(X)$. Now from (1) we get that at most $|\mathcal{U}| - t$ points of $\mathcal{U}$ were fixed when constructing $B(X)$, therefore unless with negligible probability at most $|\mathcal{U}| - t$ points of the polynomial are 0 (to apply this we take into account that $B(X)$ is not the zero polynomial since $B(1) = 1$). Thus there are at least t non-zero w_i's for $i \in \mathcal{S}^*$, from which recall that $\mathcal{A}$ has at most $t-1$ corrupted. Thus $\mathcal{B}$ using σ^* can break the BLS multi-signature for public key $\left(\sum_{i \in \mathcal{S}^*} w_i \cdot \mathsf{pk}_i\right)$. □

7.2.1 Construction of $\mathsf{WE}^{\mathsf{tBLS}}$

Finally, we now discuss how to actually construct $\mathsf{WE}^{\mathsf{tBLS}}$. It can be decomposed into four different gadgets — IIP, Sig, MaxDeg, and Zero. The IIP allows us to provably *aggregate* public keys but there are three additional checks that need to be carried out. First, we need to prove that *sufficiently* many public keys were aggregated, and this can be done by using a MaxDeg check. Because if the witness w used in aggregation interpolates to a degree $\mathcal{U} - t$ polynomial on $\mathcal{U}$, it must mean that this polynomial has at least t non-zero values on the domain $\mathcal{U}$. An astute reader may have already noticed that the zero polynomial also satisfies the degree check and we wish to prevent this. To do so we simply inset a dummy party with identity 1, and demand that w_1 is always 1. Coming to the final check, we only want to aggregate public keys from the set the user has specified. So we will also demand that the polynomial is zero at all points outside of this set.

Construction 9 (Linearly-Verifiable Aggregate BLS Threshold Signature)**.** *We construct a reduction from*

$$\mathsf{tBLS} \to \boxed{\mathsf{IIP} \times \mathsf{Sig} \times \mathsf{MaxDeg} \times \mathsf{Zero}}.$$

Indeed, consider input instance, witness pair

$$(([1]_1), (\{\mathsf{pk}_i\}_{i \in \mathcal{U}}), (\mathcal{S}, t, \mathsf{tg}), (\boxed{\sigma^*}, \boxed{w})) \in \mathsf{tBLS} \tag{4}$$

Let $\mathsf{apk} \leftarrow \sum_{i \in \mathcal{U}} w_i \cdot \mathsf{pk}_i$. *The* tBLS *relation reduces to checking:*

$$\begin{aligned}
(\{\mathsf{pk}_i\}_{i \in \mathcal{U}}, (\boxed{\mathsf{apk}}, \boxed{w})) &\in \mathsf{IIP} \\
(([1]_1, [0]_1^{|\mathcal{U}|-1}), ([1]_1, \boxed{w})) &\in \mathsf{IIP} \\
(g, h, \boxed{(\mathsf{apk}, \sigma^*)}) &\in \mathsf{Sig} \\
((\mathcal{U}, |\mathcal{U}| - t), \boxed{w}) &\in \mathsf{MaxDeg} \\
((\mathcal{U} \setminus \mathcal{S}), \boxed{w}) &\in \mathsf{Zero}.
\end{aligned}$$

7.3 Registered Broadcast Encryption

As noted in [GKPW24], Distributed Broadcast Encryption [WQZD10, BZ14] is a special case of Silent Threshold Encryption (with succinct ciphertexts), when simply setting the threshold to $t = 1$.

We highlight that, similarly, one can define Registered Broadcast Encryption, where each decryptor is registering her key with a Key Curator. The latter "aggregates" the key in the registration to a succinct master public key mpk. Then any encryptor can encrypt to a set of users, using the succinct master public key, and, unlike Distributed Broadcast Encryption the encryptors do not have store the keys of all the users. One can trivially construct Registered Broadcast Encryption from any Registered Threshold Encryption, by setting $t = 1$. Therefore, we omit the formally defining and constructing Registered Broadcast Encryption. However, we note that Registered Broadcast Encryption is a direct implication of Threshold Broadcast Encryption.

Acknowledgements

This work is supported in part by the AFOSR Award FA9550-24-1-0156 and research grants from the Bakar Fund, J. P. Morgan Faculty Research Award, Supra Inc., Sui Foundation, and the Stellar Development Foundation. Dimitris Kolonelos is also supported in part by a Berkeley Center for Responsible, Decentralized Intelligence (RDI) Fellowship. Abhiram Kothapalli is also supported in part by Anaxi Labs.

References

[ABB$^+$13] Michel Abdalla, Fabrice Benhamouda, Olivier Blazy, Céline Chevalier, and David Pointcheval. SPHF-friendly non-interactive commitments. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 214–234. Springer, Berlin, Heidelberg, December 2013. 3

[ABD$^+$21] Navid Alamati, Pedro Branco, Nico Döttling, Sanjam Garg, Mohammad Hajiabadi, and Sihang Pu. Laconic private set intersection and applications. In Kobbi Nissim and Brent Waters, editors, *TCC 2021, Part III*, volume 13044 of *LNCS*, pages 94–125. Springer, Cham, November 2021. 4

[ACP09] Michel Abdalla, Céline Chevalier, and David Pointcheval. Smooth projective hashing for conditionally extractable commitments. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 671–689. Springer, Berlin, Heidelberg, August 2009. 3

[AFP24] Amit Agarwal, Rex Fernando, and Benny Pinkas. Efficiently-thresholdizable batched identity based encryption, with applications. Cryptology ePrint Archive, Paper 2024/1575, 2024. 5

[ALOS22] Diego F. Aranha, Chuanwei Lin, Claudio Orlandi, and Mark Simkin. Laconic private set-intersection from pairings. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 111–124. ACM Press, November 2022. 4, 5

[AMYY25] Shweta Agrawal, Anuja Modi, Anshu Yadav, and Shota Yamada. Evasive lwe: attacks, variants & obfustopia. *Cryptology ePrint Archive*, 2025. 3

[AT24] Nuttapong Attrapadung and Junichi Tomida. A modular approach to registered ABE for unbounded predicates. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part III*, volume 14922 of *LNCS*, pages 280–316. Springer, Cham, August 2024. 5, 7, 47

[AY20] Shweta Agrawal and Shota Yamada. Optimal broadcast encryption from pairings and LWE. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 13–43. Springer, Cham, May 2020. 11

[BBC$^+$13] Fabrice Ben Hamouda, Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. Efficient UC-secure authenticated key-exchange for algebraic languages. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 272–291. Springer, Berlin, Heidelberg, February / March 2013. 3

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 440–456. Springer, Berlin, Heidelberg, May 2005. 14

[BC16] Olivier Blazy and Céline Chevalier. Structure-preserving smooth projective hashing. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 339–369. Springer, Berlin, Heidelberg, December 2016. 3, 5

[BCR$^+$19] Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. Aurora: Transparent succinct arguments for R1CS. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 103–128. Springer, Cham, May 2019. 14, 25

[BDHL24] Rishabh Bhadauria, Nico Döttling, Carmit Hazay, and Chuanwei Lin. Private laconic oblivious transfer with preprocessing. Cryptology ePrint Archive, Paper 2024/1555, 2024. 4

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer, Berlin, Heidelberg, August 2001. 3, 22

[BFOQ24] Jan Bormet, Sebastian Faust, Hussien Othman, and Ziyan Qu. BEAT-MEV: Epochless approach to batched threshold encryption for MEV prevention. Cryptology ePrint Archive, Paper 2024/1533, 2024. 5

[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, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 1–18. Springer, Berlin, Heidelberg, August 2001. 3

[BGI$^+$17] Saikrishna Badrinarayanan, Sanjam Garg, Yuval Ishai, Amit Sahai, and Akshay Wadia. Two-message witness indistinguishability and secure computation in the plain model from new assumptions. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part III*, volume 10626 of *LNCS*, pages 275–303. Springer, Cham, December 2017. 3

[BGW05] Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 258–275. Springer, Berlin, Heidelberg, August 2005. 4

[BISW18] Dan Boneh, Yuval Ishai, Amit Sahai, and David J. Wu. Quasi-optimal SNARGs via linear multi-prover interactive proofs. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 222–255. Springer, Cham, April / May 2018. 3

[BL18] Fabrice Benhamouda and Huijia Lin. k-round multiparty computation from k-round oblivious transfer via garbled interactive circuits. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 500–532. Springer, Cham, April / May 2018. 4

[BL20] Fabrice Benhamouda and Huijia Lin. Mr NISC: Multiparty reusable non-interactive secure computation. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part II*, volume 12551 of *LNCS*, pages 349–378. Springer, Cham, November 2020. 4, 5

[BLM$^+$24] Pedro Branco, Russell W. F. Lai, Monosij Maitra, Giulio Malavolta, Ahmadreza Rahimi, and Ivy K. Y. Woo. Traitor tracing without trusted authority from registered functional encryption. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part III*, volume 15486 of *LNCS*, pages 33–66. Springer, Singapore, December 2024. 6, 12, 34

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532. Springer, Berlin, Heidelberg, December 2001. 3, 22

[BLS04] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. *Journal of Cryptology*, 17(4):297–319, September 2004. 3

[BLSV18] Zvika Brakerski, Alex Lombardi, Gil Segev, and Vinod Vaikuntanathan. Anonymous IBE, leakage resilience and circular security from new assumptions. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 535–564. Springer, Cham, April / May 2018. 4

[BÜW24] Chris Brzuska, Akin Ünal, and Ivy K. Y. Woo. Evasive LWE assumptions: Definitions, classes, and counterexamples. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part IV*, volume 15487 of *LNCS*, pages 418–449. Springer, Singapore, December 2024. 3

[BV22] Zvika Brakerski and Vinod Vaikuntanathan. Lattice-inspired broadcast encryption and succinct ciphertext-policy ABE. In Mark Braverman, editor, *ITCS 2022*, volume 215, pages 28:1–28:20. LIPIcs, January / February 2022. 11

[BZ14] Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 480–499. Springer, Berlin, Heidelberg, August 2014. 39

[CDG$^+$17] Chongwon Cho, Nico Döttling, Sanjam Garg, Divya Gupta, Peihan Miao, and Antigoni Polychroniadou. Laconic oblivious transfer and its applications. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 33–65. Springer, Cham, August 2017. 4, 5

[CFK24] Matteo Campanelli, Dario Fiore, and Hamidreza Khoshakhlagh. Witness encryption for succinct functional commitments and applications. In Qiang Tang and Vanessa Teague, editors, *PKC 2024, Part II*, volume 14602 of *LNCS*, pages 132–167. Springer, Cham, April 2024. 5

[CGPP24] Arka Rai Choudhuri, Sanjam Garg, Julien Piet, and Guru-Vamsi Policharla. Mempool privacy via batched threshold encryption: Attacks and defenses. In Davide Balzarotti and Wenyuan Xu, editors, *USENIX Security 2024*. USENIX Association, August 2024. 5

[CGPW24] Arka Rai Choudhuri, Sanjam Garg, Guru-Vamsi Policharla, and Mingyuan Wang. Practical mempool privacy via one-time setup batched threshold encryption. Cryptology ePrint Archive, Paper 2024/1516, 2024. 5

[CS98] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Hugo Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 13–25. Springer, Berlin, Heidelberg, August 1998. 3

[CS02] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 45–64. Springer, Berlin, Heidelberg, April / May 2002. 3

[CVW18] Yilei Chen, Vinod Vaikuntanathan, and Hoeteck Wee. GGH15 beyond permutation branching programs: Proofs, attacks, and candidates. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 577–607. Springer, Cham, August 2018. 3

[DCX^{+}23] Sourav Das, Philippe Camacho, Zhuolun Xiang, Javier Nieto, Benedikt Bünz, and Ling Ren. Threshold signatures from inner product argument: Succinct, weighted, and multi-threshold. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *ACM CCS 2023*, pages 356–370. ACM Press, November 2023. 25

[DG17a] Nico Döttling and Sanjam Garg. From selective IBE to full IBE and selective HIBE. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 372–408. Springer, Cham, November 2017. 4

[DG17b] Nico Döttling and Sanjam Garg. Identity-based encryption from the Diffie-Hellman assumption. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 537–569. Springer, Cham, August 2017. 4

[DGGM19] Nico Döttling, Sanjam Garg, Vipul Goyal, and Giulio Malavolta. Laconic conditional disclosure of secrets and applications. In David Zuckerman, editor, *60th FOCS*, pages 661–685. IEEE Computer Society Press, November 2019. 4

[DGI^{+}19] Nico Döttling, Sanjam Garg, Yuval Ishai, Giulio Malavolta, Tamer Mour, and Rafail Ostrovsky. Trapdoor hash functions and their applications. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 3–32. Springer, Cham, August 2019. 4

[DGM23] Nico Döttling, Phillip Gajland, and Giulio Malavolta. Laconic function evaluation for Turing machines. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *PKC 2023, Part II*, volume 13941 of *LNCS*, pages 606–634. Springer, Cham, May 2023. 4

[DHM^{+}24] Fangqi Dong, Zihan Hao, Ethan Mook, Hoeteck Wee, and Daniel Wichs. Laconic function evaluation and ABE for RAMs from (ring-)LWE. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part III*, volume 14922 of *LNCS*, pages 107–142. Springer, Cham, August 2024. 4

[DHMW24] Fangqi Dong, Zihan Hao, Ethan Mook, and Daniel Wichs. Laconic function evaluation, functional encryption and obfuscation for RAMs with sublinear computation. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part II*, volume 14652 of *LNCS*, pages 190–218. Springer, Cham, May 2024. 4

[DJM^{+}25] Nico Döttling, Abhishek Jain, Giulio Malavolta, Surya Mathialagan, and Vinod Vaikuntanathan. Simple and general counterexamples for private-coin evasive lwe. *Cryptology ePrint Archive*, 2025. 3

[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, *EUROCRYPT 2023, Part III*, volume 14006 of *LNCS*, pages 417–446. Springer, Cham, April 2023. 4, 5

[FFM^{+}23] Danilo Francati, Daniele Friolo, Monosij Maitra, Giulio Malavolta, Ahmadreza Rahimi, and Daniele Venturi. Registered (inner-product) functional encryption. In Jian Guo and Ron Steinfeld, editors, *ASIACRYPT 2023, Part V*, volume 14442 of *LNCS*, pages 98–133. Springer, Singapore, December 2023. 5

[FHAS24] Nils Fleischhacker, Mathias Hall-Andersen, and Mark Simkin. Extractable witness encryption for KZG commitments and efficient laconic OT. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part II*, volume 15485 of *LNCS*, pages 423–453. Springer, Singapore, December 2024. 4, 5

[FKdP23] Dario Fiore, Dimitris Kolonelos, and Paola de Perthuis. Cuckoo commitments: Registration-based encryption and key-value map commitments for large spaces. In Jian Guo and Ron Steinfeld, editors, *ASIACRYPT 2023, Part V*, volume 14442 of *LNCS*, pages 166–200. Springer, Singapore, December 2023. 5, 6

[FNV17] Antonio Faonio, Jesper Buus Nielsen, and Daniele Venturi. Predictable arguments of knowledge. In Serge Fehr, editor, *PKC 2017, Part I*, volume 10174 of *LNCS*, pages 121–150. Springer, Berlin, Heidelberg, March 2017. 3

[FWW23] Cody Freitag, Brent Waters, and David J. Wu. How to use (plain) witness encryption: Registered ABE, flexible broadcast, and more. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part IV*, volume 14084 of *LNCS*, pages 498–531. Springer, Cham, August 2023. 3, 7

[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 FOCS*, pages 40–49. IEEE Computer Society Press, October 2013. 3

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 467–476. ACM Press, June 2013. 3

[GH18] Sanjam Garg and Mohammad Hajiabadi. Trapdoor functions from the computational Diffie-Hellman assumption. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 362–391. Springer, Cham, August 2018. 4

[GHMM24] Sanjam Garg, Mohammad Hajiabadi, Peihan Miao, and Alice Murphy. Laconic branching programs from the Diffie-Hellman assumption. In Qiang Tang and Vanessa Teague, editors, *PKC 2024, Part II*, volume 14603 of *LNCS*, pages 323–355. Springer, Cham, April 2024. 4

[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, Part I*, volume 11239 of *LNCS*, pages 689–718. Springer, Cham, November 2018. 28, 34

[GJM$^+$24] Sanjam Garg, Abhishek Jain, Pratyay Mukherjee, Rohit Sinha, Mingyuan Wang, and Yinuo Zhang. hinTS: Threshold signatures with silent setup. In *2024 IEEE Symposium on Security and Privacy*, pages 3034–3052. IEEE Computer Society Press, May 2024. 10, 25

[GJV23] Matthew Green, Abhishek Jain, and Gijs Van Laer. Efficient set membership encryption and applications. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *ACM CCS 2023*, pages 1080–1092. ACM Press, November 2023. 5

[GKMR23] Noemi Glaeser, Dimitris Kolonelos, Giulio Malavolta, and Ahmadreza Rahimi. Efficient registration-based encryption. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *ACM CCS 2023*, pages 1065–1079. ACM Press, November 2023. 5, 6

[GKP$^+$13] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. How to run Turing machines on encrypted data. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 536–553. Springer, Berlin, Heidelberg, August 2013. 3

[GKPW24] Sanjam Garg, Dimitris Kolonelos, Guru-Vamsi Policharla, and Mingyuan Wang. Threshold encryption with silent setup. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part VII*, volume 14926 of *LNCS*, pages 352–386. Springer, Cham, August 2024. 5, 6, 7, 11, 13, 34, 39

[GL89] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *21st ACM STOC*, pages 25–32. ACM Press, May 1989. 3

[GL03] Rosario Gennaro and Yehuda Lindell. A framework for password-based authenticated key exchange. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 524–543. Springer, Berlin, Heidelberg, May 2003. 3

[GLW14] Craig Gentry, Allison B. Lewko, and Brent Waters. Witness encryption from instance independent assumptions. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 426–443. Springer, Berlin, Heidelberg, August 2014. 3

[GLWW24] Rachit Garg, George Lu, Brent Waters, and David J. Wu. Reducing the CRS size in registered ABE systems. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part III*, volume 14922 of *LNCS*, pages 143–177. Springer, Cham, August 2024. 5, 6, 7

[GOS18] Sanjam Garg, Rafail Ostrovsky, and Akshayaram Srinivasan. Adaptive garbled RAM from laconic oblivious transfer. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 515–544. Springer, Cham, August 2018. 4

[Gro16] Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, Berlin, Heidelberg, May 2016. 7

[GS17] Sanjam Garg and Akshayaram Srinivasan. Garbled protocols and two-round MPC from bilinear maps. In Chris Umans, editor, *58th FOCS*, pages 588–599. IEEE Computer Society Press, October 2017. 3, 4

[GS18] Sanjam Garg and Akshayaram Srinivasan. Two-round multiparty secure computation from minimal assumptions. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 468–499. Springer, Cham, April / May 2018. 4

[HJL25] Yao-Ching Hsieh, Aayush Jain, and Huijia Lin. Lattice-based post-quantum io from circular security with random opening assumption (part ii: zeroizing attacks against private-coin evasive lwe assumptions). *Cryptology ePrint Archive*, 2025. 3

[HLL23] Yao-Ching Hsieh, Huijia Lin, and Ji Luo. Attribute-based encryption for circuits of unbounded depth from lattices. In *64th FOCS*, pages 415–434. IEEE Computer Society Press, November 2023. 4

[HLWW23] Susan Hohenberger, George Lu, Brent Waters, and David J. Wu. Registered attribute-based encryption. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part III*, volume 14006 of *LNCS*, pages 511–542. Springer, Cham, April 2023. 5, 6, 7, 10, 28, 29, 47, 48

[Kal05] Yael Tauman Kalai. Smooth projective hashing and two-message oblivious transfer. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 78–95. Springer, Berlin, Heidelberg, May 2005. 3

[KMW23] Dimitris Kolonelos, Giulio Malavolta, and Hoeteck Wee. Distributed broadcast encryption from bilinear groups. In Jian Guo and Ron Steinfeld, editors, *ASIACRYPT 2023, Part V*, volume 14442 of *LNCS*, pages 407–441. Springer, Singapore, December 2023. 5, 6, 11

[KV11] Jonathan Katz and Vinod Vaikuntanathan. Round-optimal password-based authenticated key exchange. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 293–310. Springer, Berlin, Heidelberg, March 2011. 3

[KZG10] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 177–194. Springer, Berlin, Heidelberg, December 2010. 6, 22

[Mau05] Ueli M. Maurer. Abstract models of computation in cryptography (invited paper). In Nigel P. Smart, editor, *10th IMA International Conference on Cryptography and Coding*, volume 3796 of *LNCS*, pages 1–12. Springer, Berlin, Heidelberg, December 2005. 3, 6, 14

[PCFT20] Bo Pang, Long Chen, Xiong Fan, and Qiang Tang. Multi-input laconic function evaluation. In Joseph K. Liu and Hui Cui, editors, *ACISP 20*, volume 12248 of *LNCS*, pages 369–388. Springer, Cham, November / December 2020. 4

[QWW18] Willy Quach, Hoeteck Wee, and Daniel Wichs. Laconic function evaluation and applications. In Mikkel Thorup, editor, *59th FOCS*, pages 859–870. IEEE Computer Society Press, October 2018. 4

[Ros22] Razvan Rosie. Adaptively secure laconic function evaluation for NC^1. In Steven D. Galbraith, editor, *CT-RSA 2022*, volume 13161 of *LNCS*, pages 427–450. Springer, Cham, March 2022. 4

[RZ21] Carla Ràfols and Arantxa Zapico. An algebraic framework for universal and updatable SNARKs. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 774–804, Virtual Event, August 2021. Springer, Cham. 14, 25

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Berlin, Heidelberg, May 1997. 3, 6, 14

[Tsa22] Rotem Tsabary. Candidate witness encryption from lattice techniques. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part I*, volume 13507 of *LNCS*, pages 535–559. Springer, Cham, August 2022. 3

[VWW22] Vinod Vaikuntanathan, Hoeteck Wee, and Daniel Wichs. Witness encryption and null-IO from evasive LWE. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part I*, volume 13791 of *LNCS*, pages 195–221. Springer, Cham, December 2022. 3

[Wee22] Hoeteck Wee. Optimal broadcast encryption and CP-ABE from evasive lattice assumptions. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 217–241. Springer, Cham, May / June 2022. 3, 11

[WQZD10] Qianhong Wu, Bo Qin, Lei Zhang, and Josep Domingo-Ferrer. Ad hoc broadcast encryption (poster presentation). In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *ACM CCS 2010*, pages 741–743. ACM Press, October 2010. 5, 39

[Zha16] Mark Zhandry. How to avoid obfuscation using witness PRFs. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part II*, volume 9563 of *LNCS*, pages 421–448. Springer, Berlin, Heidelberg, January 2016. 3

[ZZC$^+$25] Ziqi Zhu, Kai Zhang, Zhili Chen, Junqing Gong, and Haifeng Qian. Black-box registered abe from lattices. *Cryptology ePrint Archive*, 2025. 6

[ZZGQ23] Ziqi Zhu, Kai Zhang, Junqing Gong, and Haifeng Qian. Registered ABE via predicate encodings. In Jian Guo and Ron Steinfeld, editors, *ASIACRYPT 2023, Part V*, volume 14442 of *LNCS*, pages 66–97. Springer, Singapore, December 2023. 5, 6, 7

A Registered Attribute-Based Encryption Definition

We recall the definition of Registered Attribute-Based Encryption (R-ABE). We closely follow the definition in [HLWW23], slightly rephrased, with the main difference being that, unlike [HLWW23] (and similarly to [AT24]), we define R-ABE for an unbounded attribute space. Additionally, our definition below is for R-ABE with an a-priori upper bound on the number of users that it can support, namely Bounded Registered Attribute-Based Encryption.

Definition 23 (Bounded Registered Attributed-Based Encryption (R-ABE))**.** *A (Ciphertext-Policy) Bounded Registered Attribute-Based Encryption scheme with attribute space* $\mathbb{U}$ *and policy space* $\mathbb{P}$ *consists of the following six PPT algorithms* $(\texttt{Setup}, \texttt{KGen}, \texttt{Reg}, \texttt{Upd}, \texttt{Enc}, \texttt{Dec})$. $\texttt{Setup}$

- $\texttt{Setup}(1^\lambda, M) \to \mathsf{crs}$: *On input the security parameter* λ *and a maximum number of users* $M > 0$*, the randomized setup algorithm samples a common reference string* crs*. The message space* $\mathbb{M}$ *is also specified here and is (implicitly) contained in the* crs*.*
- $\texttt{KGen}(\mathsf{crs}, \mathsf{aux}) \to (\mathsf{pk}, \mathsf{sk})$: *On input the common reference string* crs *and auxiliary information* aux*, the randomized key generation algorithm outputs a public and secret key* $(\mathsf{pk}, \mathsf{sk})$*.*
- $\texttt{Reg}(\mathsf{crs}, \mathsf{mpk}, \mathsf{aux}, \mathcal{U}, \mathsf{pk}) \to (\mathsf{mpk}', \mathsf{aux}')$: *On input the common reference string* crs*, auxiliary information* aux*, the current master public key* mpk*, a set of user's attributes* $\mathcal{U} \subseteq \mathbb{U}$*, and a public key* pk*, the deterministic registration algorithm outputs the new master public key* mpk' *and auxiliary information* aux'*. (The system is initialized with master public key* $\mathsf{mpk} = \perp$ *and auxiliary information* $\mathsf{aux} = \perp$*.)*
- $\texttt{Upd}(\mathsf{crs}, \mathsf{aux}, \mathsf{pk}) \to \mathsf{hsk}$: *On input the common reference string* crs*, the auxiliary public information* aux *and a public key* pk*, the deterministic update algorithm outputs a helping decryption key* hsk*.*
- $\texttt{Enc}(\mathsf{mpk}, \mathcal{P}, \mathsf{msg}) \to \mathsf{ct}$: *On input the current public parameters* mpk*, a policy function* $\mathcal{P} \in \mathbb{P}$ *and a message* $\mathsf{msg} \in \mathbb{M}$*, the randomized encryption algorithm outputs a ciphertext* ct*.*
- $\texttt{Dec}(\mathsf{sk}, \mathsf{hsk}, \mathsf{ct}) \to \mathsf{msg}$: *On input the secret* sk*, the (current) helper decryption key* hsk *and a ciphertext* ct*, the deterministic decryption algorithm outputs a message* $\mathsf{msg} \in \mathbb{M} \cup \{\perp, \mathsf{GetUpd}\}$*. The symbol* $\perp$ *indicates a syntax error, while* GetUpd *indicates that the current helper decryption key* hsk *is not updated and a new* $\texttt{Upd}$ *needs to be run to obtain an updated one.*

Definition 24 (Correctness and efficiency of R-ABE)**.** *For any interactive PPT adversary* $\mathcal{A}$*, consider the following game between an adversary* $\mathcal{A}$ *and a challenger* $\mathcal{C}$*, parametrized by* λ *and a bound* $M = \mathsf{poly}(\lambda)$*.*

1. **Setup phase.** *$\mathcal{C}$ initializes* $\mathsf{mpk} \leftarrow \perp$, $\mathsf{aux} \leftarrow \perp$ *and samples* $\mathsf{crs} \leftarrow \mathtt{Setup}(1^\lambda, M)$. *It also initializes counters* $N \leftarrow 0$, $E \leftarrow 0$ *for the number of registration and encryption queries respectively and initializes* $i^* = \perp$ *for the target key index. It further initialized a set* $\mathcal{U}_{\mathsf{eff}} \leftarrow \emptyset$ *for the set of effective (registered) attributes.* $\mathcal{C}$ *sends* crs *to* $\mathcal{A}$.

2. **Query phase.** $\mathcal{A}$ *chooses one of the following queries.*

   (a) **Registering new (non-target) key query.** *If* $N = M$ *skip this step. Otherwise,* $\mathcal{A}$ *chooses a public* pk *and set of attributes* $\mathcal{U} \subseteq \mathbb{U}$ *and send them to* $\mathcal{C}$ *for registration.* $\mathcal{C}$ *updates the counter* $N \leftarrow N + 1$ *and the set of effective attributes* $\mathcal{U}_{\mathsf{eff}} \leftarrow \mathcal{U}_{\mathsf{eff}} \cup \mathcal{U}$ *and then registers* $(\mathsf{pk}, \mathcal{U})$ *computing* $(\mathsf{mpk}_N, \mathsf{aux}') \leftarrow \mathtt{Reg}(\mathsf{crs}, \mathsf{mpk}_{N-1}, \mathsf{aux}, \mathcal{U}, \mathsf{pk})$ *and setting* $\mathsf{aux} \leftarrow \mathsf{aux}'$. $\mathcal{C}$ *sends* $(N, \mathsf{mpk}_N, \mathsf{aux})$ *to* $\mathcal{A}$.

   (b) **Registering the target key query.** *If* $i^* \neq \perp$ *or* $N = M$ *, skip this step. Otherwise,* $\mathcal{A}$ *chooses a set of attributes* $\mathcal{U}^*$ *and sends it to* $\mathcal{C}$. $\mathcal{C}$ *then updates the counter* $N \leftarrow N + 1$ *and the set* $\mathcal{U}_{\mathsf{eff}} \leftarrow \mathcal{U}_{\mathsf{eff}} \cup \mathcal{U}^*$, *samples a new key-pair* $(\mathsf{pk}^*, \mathsf{sk}^*) \leftarrow \mathtt{KGen}(\mathsf{crs}, \mathsf{aux})$, *registers the key* $(\mathsf{mpk}_N, \mathsf{aux}') \leftarrow \mathtt{Reg}(\mathsf{crs}, \mathsf{mpk}_{N-1}, \mathsf{aux}, \mathcal{U}^*, \mathsf{pk}^*)$, *computes the helper decryption key* $\mathsf{hsk}^* \leftarrow \mathtt{Upd}(\mathsf{crs}, \mathsf{aux}, \mathsf{pk})$, *updates* $\mathsf{aux} \leftarrow \mathsf{aux}'$ *and sets the target key index to* $i^* \leftarrow N$. $\mathcal{C}$ *sends back* $(N, \mathsf{mpk}_N, \mathsf{aux}, \mathsf{pk}^*, \mathsf{hsk}^*, \mathsf{sk}^*)$ *to* $\mathcal{A}$.

   (c) **Encryption query.** $\mathcal{A}$ *sends to* $\mathcal{C}$ *sends an index* $i \in \{i^*, \ldots, N\}$ *of a master public key (computed after the target key's registration), a message* $\mathsf{msg} \in \mathbb{M}$ *and a policy* $\mathcal{P} \in \mathbb{P}$. *If the target key has not been registered* $i^* = \perp$ *or the target key's attributes do not satisfy the policy* $\mathcal{P}(\mathcal{U}^*) \neq 1$ *then* $\mathcal{C}$ *sends back* $\perp$ *to* $\mathcal{A}$. *Otherwise,* $\mathcal{C}$ *updates the counter* $E \leftarrow E + 1$ *and computes* $\mathsf{ct}_E \leftarrow \mathsf{Enc}(\mathsf{mpk}_{i^*}, \mathcal{P}, \mathsf{msg})$ *and sends back* (E, ct_E) *to* $\mathcal{A}$.

   (d) **Decryption query.** $\mathcal{A}$ *sends to* $\mathcal{C}$ *an index* $j \in [\![E]\!]$ *of a ciphertext.* $\mathcal{C}$ *computes* $\widetilde{\mathsf{msg}}_j \leftarrow \mathsf{Dec}(\mathsf{sk}^*, \mathsf{hsk}^*, \mathsf{ct}_j)$. *If* $\widetilde{\mathsf{msg}}_j = \mathsf{GetUpd}$ *then updates the helper decryption key* $\mathsf{hsk}^* \leftarrow \mathtt{Upd}(\mathsf{crs}, \mathsf{aux}, \mathsf{pk}^*)$ *and recomputes* $\widetilde{\mathsf{msg}}_j \leftarrow \mathsf{Dec}(\mathsf{sk}^*, \mathsf{hsk}^*, \mathsf{ct}_j)$.

3. *After the query phase, the adversary* $\mathcal{A}$ *wins the game if there exists* $j \in [\![E]\!]$ *such that* $\widetilde{\mathsf{msg}}_j \neq \mathsf{msg}_j$.

Let $Q \in \mathsf{poly}(\lambda)$ *be an upper bound on the number of queries issued by* $\mathcal{A}$. *We require the following properties to hold for any adversary* $\mathcal{A}$.

- **Correctness.** $\Pr[\mathcal{A} \textit{ wins}] = \mathsf{negl}(\lambda)$.

- **Compactness** *For all queries* $q \in [\![Q]\!]$, *let* $\mathsf{mpk}^{(q)}$ *be the public parameters,* $N^{(q)}$ *the registration counter and* $n^{(q)} = |\mathcal{U}_{\mathsf{eff}}^{(q)}|$ *the overall number of registered attributes after the* q*-th query. Then for all* $q \in [\![Q]\!]$, $|\mathsf{mpk}_q| = \mathsf{poly}(\lambda, n^{(q)}, \log(N^{(q)}))$. *Moreover, the size of the helper decryption key* hsk^* *is bounded by* $|\mathsf{hsk}^*| = \mathsf{poly}(\lambda, n^{(q)}, \log(N^{(q)}))$.

- **Efficiency of the number of updates.** *The total number of invocations of* $\mathtt{Upd}$ *in Step 2(d) of the game is at most* $\log(N)$.

Remark 3 (Unbounded Attribute Space). *In the original definition [HLWW23] the compactness property allows* $|\mathsf{mpk}^{(q)}|$ *and* $|\mathsf{hsk}^*|$ *to polynomially depend on the attribute space* $(\mathsf{poly}(|\mathbb{U}|))$. *This implies that* $\mathbb{U}$ *has polynomial-size. To allow for unbounded attribute spaces, we modify the definition so that* $|\mathsf{mpk}^{(q)}|$ *and* $|\mathsf{hsk}^*|$ *only polynomially depend on the number of effective attributes* $(\mathsf{poly}(n^{(q)})$ *instead), i.e. the attributes that have been included so far in an attribute set of a registered user.*

Definition 25 (Security of R-ABE). *Consider the following game $\mathsf{Sec}_{\mathcal{A}}(\lambda)$ between a PPT adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.*

1. **Setup Phase.** *$\mathcal{C}$ initializes $\mathsf{mpk} \leftarrow \bot$, $\mathsf{aux} \leftarrow \bot$ and samples $\mathsf{crs} \leftarrow \mathtt{Setup}(1^\lambda, M)$. It also initializes a counter $N_h \leftarrow 0$ for the number of honest key registration queries, a set $\mathsf{Cor} \leftarrow \emptyset$ for the set of corrupted keys and a key-attribute set dictionary $\mathsf{Dict} \leftarrow \emptyset$ for a mapping between keys and attribute sets. $\mathcal{C}$ sends crs to $\mathcal{A}$.*

2. **Query Phase.** *$\mathcal{A}$ chooses one of the following queries.*

   (a) **Registering corrupted key query.** *If $|\mathsf{Dict}| = M$ skip this step. Otherwise, $\mathcal{A}$ chooses a pk and a set of attributes $\mathcal{U} \subseteq \mathbb{U}$ and sends them to $\mathcal{C}$. $\mathcal{C}$ registers $(\mathsf{pk}, \mathcal{U})$ computing $(\mathsf{mpk}', \mathsf{aux}') \leftarrow \mathtt{Reg}(\mathsf{crs}, \mathsf{mpk}, \mathsf{aux}, \mathcal{U}, \mathsf{pk})$. Then updates $\mathsf{mpk} \leftarrow \mathsf{mpk}'$, $\mathsf{aux} \leftarrow \mathsf{aux}'$ and $\mathsf{Dict}[\mathsf{pk}] \leftarrow \mathcal{U}$, $\mathsf{Cor} \leftarrow \mathsf{Cor} \cup \{\mathsf{pk}\}$. $\mathcal{C}$ sends $(\mathsf{mpk}, \mathsf{aux})$ to $\mathcal{A}$.*

   (b) **Registering honest key query.** *If $|\mathsf{Dict}| = M$ skip this step. Otherwise, $\mathcal{A}$ chooses a set of attributes $\mathcal{U} \subseteq \mathbb{U}$ and sends it to $\mathcal{C}$. $\mathcal{C}$ updates $N_h \leftarrow N_h + 1$, samples a key-pair $(\mathsf{pk}_{N_h}, \mathsf{sk}_{N_h}) \leftarrow \mathtt{KGen}(\mathsf{crs}, \mathsf{aux})$ and registers the key with the attribute set $\mathcal{A}$ chose $(\mathsf{pk}_{N_h}, \mathcal{U})$ computing $(\mathsf{mpk}', \mathsf{aux}') \leftarrow \mathtt{Reg}(\mathsf{crs}, \mathsf{mpk}, \mathsf{aux}, \mathcal{U}, \mathsf{pk}_{N_h})$. Then updates $\mathsf{mpk} \leftarrow \mathsf{mpk}'$, $\mathsf{aux} \leftarrow \mathsf{aux}'$ and $\mathsf{Dict}[\mathsf{pk}_{N_h}] \leftarrow \mathcal{U}$. $\mathcal{C}$ sends $(\mathsf{mpk}, \mathsf{aux})$ to $\mathcal{A}$.*

   (c) **Corrupt honest key query.** *$\mathcal{A}$ sends an index $i \in \{1, \ldots, N_h\}$ of a public key to $\mathcal{C}$. $\mathcal{C}$ adds pk_i to the corrupted keys, $\mathsf{Cor} \leftarrow \mathsf{Cor} \cup \{\mathsf{pk}_i\}$ and sends back to $\mathcal{A}$ the correspinding secret key sk_i.*

3. **Challenge phase.** *$\mathcal{A}$ chooses a policy $\mathcal{P}^* \in \mathbb{P}$ and two messages $(\mathsf{msg}_0, \mathsf{msg}_1)$ and sends them to $\mathcal{C}$. $\mathcal{C}$ generates $\mathsf{ct}^* \leftarrow \mathtt{Enc}(\mathsf{mpk}, \mathcal{P}^*, \mathsf{msg}_b)$, where $b \leftarrow\!\!\$\ \{0,1\}$ is a randomly chosen bit, and sends ct^* to $\mathcal{A}$.*

4. **Output.** *The adversary $\mathcal{A}$ outputs a bit b'.*

An adversary is admissible if none of the corrupted keys' ($\mathsf{pk} \in \mathsf{Cor}$) attribute sets satisfy the policy $\mathcal{P}^*$. We call an R-ABE scheme secure if for all $M = \mathsf{poly}(\lambda)$ and for all PPT admissible adversaries $\mathcal{A}$ it holds that $\Pr[b = b'] \leqslant \frac{1}{2} + \mathsf{negl}(\lambda)$.

B Full Descriptions of our protocols

B.1 Our R-ABE Construction Unwrapped

For completeness, below we provide the full description of our slotted R-ABE protocol of 6. When unwrapped, we observe that our protocol can enjoy some concrete efficiency optimizations. We observe that the IIP and DLOG relations, when unwrapped, can be unified to a single, slightly different, IIP relation. Instead of getting the inner product of $([\mathsf{sk}_1]_1, \ldots, [\mathsf{sk}_m]_1)$ with the unit vector $\boldsymbol{w} = \boldsymbol{e}_i$ that gives $[\mathsf{sk}_i]_1 = \mathsf{pk}_i$ as a result and then applying the DLOG gadget to the result, we can equivalently get an inner product between $([\mathsf{sk}_1]_1, \ldots, [\mathsf{sk}_m]_1)$ and $\boldsymbol{w} = \mathsf{sk}_i^{-1}\boldsymbol{e}_i = (0 \ldots, \mathsf{sk}_i^{-1}, \ldots, 0)$ with the requirement that it gives the result $[1]_T$. Intuitively this combines both IIP and DLOG into a single IIP. Moreover, this is implicitly a non-zero check since no zero vector $\boldsymbol{w}$ can give outcome $[1]_T$, which eliminates the need for the NonZero relation. We incorporate these optimizations to the description of the unwrapped protocol.

The WE equations for encryption of an AND gate, $u_1 \wedge u_2 \wedge \ldots \wedge u_k$ are:

$$\begin{array}{rl}
1. & [Z_{\mathcal{I}_{u_1}}(\tau)]_1 \circ [B_i(\tau)]_2 = [Z_\Omega(\tau)]_2 \circ [Q_{u_1}(\tau)]_1 \\
\vdots & \qquad\qquad \vdots \\
\text{k.} & [Z_{\mathcal{I}_{u_n}}(\tau)]_1 \circ [B_i(\tau)]_2 = [Z_\Omega(\tau)]_2 \circ [Q_{u_k}(\tau)]_1 \\
\text{k+1.} & [\sum_{j=1}^{m} \mathsf{sk}_j L_j^\Omega(\tau)]_1 \circ B_i(\tau)]_2 = [1]_1 \circ [1]_2 + [Z_\Omega(\tau)]_2 \circ [Q_Z(\tau)]_1 + [\tau]_2 \circ [Q_x(\tau)]_1 \\
\text{k+2.} & [\tau^2]_2 \circ [Q_x(\tau)]_1 = [1]_2 \circ [\hat{Q}_x(\tau)]_1
\end{array}$$

Here $B(X) = \mathsf{sk}_i^{-1} L_i^\Omega(X)$ is full degree, interpolating 0 everywhere and sk_i^{-1} in ω^i.

Our unwrapped R-ABE construction is below.

- $\texttt{Setup}(1^\lambda, m)$: Sample $\tau \leftarrow\!\$\ \mathbb{Z}_p$ and output:

$$\mathsf{crs} = \left([\tau^1]_1, \ldots, [\tau^m]_1, [\tau^1]_2, \ldots, [\tau^m]_2\right).$$

- $\texttt{KGen}(\mathsf{crs}, i)$: Sample $x \leftarrow\!\$\ \mathbb{Z}_p^*$ and output

$$\mathsf{sk}_i = x, \quad \mathsf{pk}_i = \left([x]_1, [x \cdot \tau]_1, \ldots, [x \cdot \tau^m]_1\right).$$

- $\texttt{isValid}(\mathsf{crs}, i, \mathsf{pk}_i)$: Parse $\mathsf{pk}_i := (\mathsf{pk}_{i,0}, \mathsf{pk}_{i,1}, \ldots, \mathsf{pk}_{i,m})$ and output 1 if for each $j \in [\![m]\!]$:

$$[\mathsf{pk}_{i,0}]_1 \circ [\tau^j]_2 = [\mathsf{pk}_{i,j}]_1 \circ [1]_2.$$

- $\texttt{Aggregate}(\mathsf{crs}, (\mathsf{pk}_1, \mathcal{U}_1), \ldots, (\mathsf{pk}_m, \mathcal{U}_m))$: Define the set of effective attributes as $\mathcal{U}_{\mathsf{eff}} = \mathcal{U}_1 \cup \ldots \cup \mathcal{U}_m := \{u_1, \ldots, u_n\} \subseteq \mathbb{U}$. Then further define a set of slots for each attribute, representing the corresponding users that possess the attribute and its complementary, representing the corresponding users that do not possess the attribute:

$$\begin{aligned}
\mathcal{I}_{u_1}^{(0)} &= \{i \in [\![m]\!] : u_1 \in \mathcal{U}_i\}, & \mathcal{I}_{u_1}^{(1)} &= \{i \in [\![m]\!] : u_1 \notin \mathcal{U}_i\} \\
&\vdots & &\vdots \\
\mathcal{I}_{u_n}^{(0)} &= \{i \in [\![m]\!] : u_n \in \mathcal{U}_i\}, & \mathcal{I}_{u_n}^{(1)} &= \{i \in [\![m]\!] : u_n \notin \mathcal{U}_i\}
\end{aligned}$$

  Also, let $\Omega = \{\omega^1, \ldots, \omega^m\}$ be a group of roots of unity of order m.

  For each user $i \in [\![m]\!]$ parse $\mathsf{pk}_i := (\mathsf{pk}_{i,0}, \mathsf{pk}_{i,1}, \ldots, \mathsf{pk}_{i,m}) = ([\mathsf{sk}_i]_1, [\mathsf{sk}_i\tau]_1, \ldots, [\mathsf{sk}_i\tau^m])$. Using this information compute

$$C = [\sum_{i=1}^{m} \mathsf{sk}_i L_i^{\Omega}(\tau)]_1$$

$$U = [Z_{\Omega}(\tau)]_2 = \left[\prod_{i=1}^{m}(\tau - \omega^i)\right]$$

$$U_1^{(0)} = [Z_{\mathcal{I}_{u_1}^{(0)}}(\tau)]_1 = \left[\prod_{i \in \mathcal{I}_{u_1}^{(0)}} (\tau - \omega^i)\right]_1, \quad U_1^{(1)} = [Z_{\mathcal{I}_{u_1}^{(1)}}(\tau)]_1 = \left[\prod_{i \in \mathcal{I}_{u_1}^{(1)}} (\tau - \omega^i)\right]_1$$

$$\vdots$$

$$U_n^{(0)} = [Z_{\mathcal{I}_{u_n}^{(0)}}(\tau)]_1 = \left[\prod_{i \in \mathcal{I}_{u_n}^{(0)}} (\tau - \omega^i)\right]_1, \quad U_n^{(1)} = [Z_{\mathcal{I}_{u_n}^{(1)}}(\tau)]_1 = \left[\prod_{i \in \mathcal{I}_{u_n}^{(1)}} (\tau - \omega^i)\right]_1$$

and for all $i \in [\![m]\!]$:

$$\mathsf{hsk}_{i,1}^{(0)} = \left[\frac{L_i^{\Omega}(\tau) Z_{\mathcal{I}_{u_1}^{(0)}}(\tau)}{Z_{\Omega}(\tau)}\right]_1, \qquad \mathsf{hsk}_{i,1}^{(1)} = \left[\frac{L_i^{\Omega}(\tau) Z_{\mathcal{I}_{u_1}^{(1)}}(\tau)}{Z_{\Omega}(\tau)}\right]_1$$

$$\vdots$$

$$\mathsf{hsk}_{i,n}^{(0)} = \left[\frac{L_i^{\Omega}(\tau) Z_{\mathcal{I}_{u_n}^{(0)}}(\tau)}{Z_{\Omega}(\tau)}\right]_1, \qquad \mathsf{hsk}_{i,n}^{(1)} = \left[\frac{L_i^{\Omega}(\tau) Z_{\mathcal{I}_{u_n}^{(1)}}(\tau)}{Z_{\Omega}(\tau)}\right]_1$$

$$\mathsf{hsk}_{i,n+1} = [L_i^{\Omega}(\tau)]_2$$

$$\mathsf{hsk}_{i,n+2} = \left[\left(\frac{(L_i^{\Omega}(\tau))^2 - L_i^{\Omega}(\tau)}{Z_{\Omega}(\tau)}\right)\right]_1$$

$$\mathsf{hsk}_{i,n+3} = \left[\left(\sum_{j=1, j\neq i}^{m} \mathsf{sk}_j \frac{L_i^{\Omega}(\tau) L_j^{\Omega}(\tau)}{Z_{\Omega}(\tau)}\right)\right]_1$$

$$\mathsf{hsk}_{i,n+4} = \left[\frac{L_i^{\Omega}(\tau) - L_i^{\Omega}(0)}{\tau}\right]_1$$

$$\mathsf{hsk}_{i,n+5} = \left[L_i^{\Omega}(\tau) - L_i^{\Omega}(0)\right]_1$$

Finally, output

$$\mathsf{mpk} = (C, U, \{U_1^{(b)}, \ldots, U_n^{(b)}\}_{b\in\{0,1\}}, \mathcal{U}_{\mathsf{eff}})$$

$$\left\{\mathsf{hsk}_i = \left(\mathcal{U}_{\mathsf{eff}}, \mathcal{U}_i, \left\{\mathsf{hsk}_{i,1}^{(b)}, \ldots, \mathsf{hsk}_{i,n}^{(b)}\right\}_{b\in\{0,1\}}, \mathsf{hsk}_{i,n+1}, \ldots, \mathsf{hsk}_{i,n+5}\right)\right\}_{i=1}^{m}$$

- $\mathtt{Enc}(\mathsf{mpk}, \mathcal{P}, \mathsf{msg})$: Parse $\mathsf{mpk} := (C, U, \{U_1^{(b)}, \ldots, U_n^{(b)}\}_{b\in\{0,1\}}, \mathcal{U}_{\mathsf{eff}})$. Let the policy be the DNF formula $\mathcal{P} := \bigvee_{\alpha=1}^{|\mathrm{OR}|} l_\alpha$ and $l_\alpha := \bigwedge_{\beta=1}^{k_\alpha} u_{i_{\alpha,\beta}}^{(b_{\alpha,\beta})}$, where the superscript bit denotes negation or not, i.e. $u^{(0)}_{i_{\alpha,\beta}} = u_{i_{\alpha,\beta}}$ and $u^{(1)}_{i_{\alpha,\beta}} = \neg u_{i_{\alpha,\beta}}$. Further, for each $\alpha \in [\![|\mathrm{OR}|]\!], \beta \in [\![k_\alpha]\!]$ set:

$$\hat{U}_{i_{\alpha,\beta}} = \begin{cases} U^{(0)}_{i_{\alpha,\beta}}, & \text{if } b_{\alpha,\beta} = 0 \text{ and } u_{i_{\alpha,\beta}} \in \mathcal{U}_{\mathsf{eff}} \\ U^{(1)}_{i_{\alpha,\beta}}, & \text{if } b_{\alpha,\beta} = 1 \text{ and } u_{i_{\alpha,\beta}} \in \mathcal{U}_{\mathsf{eff}} \\ [1]_2, & \text{if } b_{\alpha,\beta} = 0 \text{ and } u_{i_{\alpha,\beta}} \notin \mathcal{U}_{\mathsf{eff}} \\ U, & \text{if } b_{\alpha,\beta} = 1 \text{ and } u_{i_{\alpha,\beta}} \notin \mathcal{U}_{\mathsf{eff}} \end{cases}$$

Then, for each clause $\alpha \in [\![|\mathrm{OR}|]\!]$ compute the $(k_\alpha + 2) \times (k_\alpha + 4)$ matrix and $k_\alpha + 2$ vector:

$$\boldsymbol{A}_\alpha = \begin{pmatrix} -Z_\Omega & \dots & 0 & \hat{U}_{i_{\alpha,1}} & 0 & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & -Z_\Omega & \hat{U}_{i_{\alpha,k_\alpha}} & 0 & 0 & 0 \\ 0 & \dots & 0 & C & -Z_\Omega & [\tau]_2 & 0 \\ 0 & \dots & 0 & 0 & 0 & [\tau^2]_2 & -[1]_2 \end{pmatrix}, \quad \boldsymbol{b} = \begin{pmatrix} [1]_T \\ [0]_T \\ [0]_T \\ \vdots \\ [0]_T \end{pmatrix},$$

sample a vector $\boldsymbol{s}_\alpha \leftarrow\!\!\$\; \mathbb{F}^{k_\alpha+2}$ and compute $\mathsf{ct}_\alpha = \big(\boldsymbol{s}_\alpha^\top \boldsymbol{A}_\alpha, \boldsymbol{s}_\alpha^\top \boldsymbol{b} + \mathsf{msg}\big)$.

Finally, output the ciphertext:

$$\mathsf{ct} = (\mathcal{P}, \mathsf{ct}_1, \dots, \mathsf{ct}_{|\mathrm{OR}|})$$

- $\mathtt{Dec}(\mathsf{sk}, \mathsf{hsk}, \mathsf{ct})$: Parse $\mathsf{hsk} := \big(\mathcal{U}_{\mathsf{eff}}, \mathcal{U}, \{\mathsf{hsk}^{(b)}_{i,1}, \dots, \mathsf{hsk}^{(b)}_{i,n}\}_{b\in\{0,1\}}, \mathsf{hsk}_{i,n+1}, \dots, \mathsf{hsk}_{i,n+5}\big)$, $\mathsf{ct} := (\mathcal{P}, \mathsf{ct}_1, \dots, \mathsf{ct}_{|\mathrm{OR}|})$ and $\mathcal{P} := \bigvee_{\alpha=1}^{|\mathrm{OR}|} \bigwedge_{\beta=1}^{k_\alpha} u^{(b_{\alpha,\beta})}_{i_{\alpha,\beta}}$. If there is no clause $\alpha \in [\![k_\alpha]\!]$ that is satisfied by $\mathcal{U}$ then output $\mathsf{msg} = \perp$. Otherwise, assume the clause l_{α^*} is satisfied by the $\mathcal{U}$ attributes then compute:

$$\left\{ \pi_{\alpha^*,\beta} = \begin{cases} [Q_{u^{(b_{\alpha^*,1})}_{i_{\alpha^*,\beta}}}(\tau)]_1 = \mathsf{sk}^{-1} \cdot \mathsf{hsk}^{(b_{\alpha^*,\beta})}_{i_{\alpha^*,\beta}}, & \text{if } u^{(b_{\alpha^*,\beta})}_{i_{\alpha^*,\beta}} \in \mathcal{U}_{\mathsf{eff}} \\ [B_i(\tau)]_2 = \mathsf{sk}^{-1} \cdot \mathsf{hsk}_{n+1}, & \text{if } u_{i_{\alpha^*,\beta}} \notin \mathcal{U}_{\mathsf{eff}} \text{ and } (b_{\alpha^*,\beta}) = 1 \\ 0, & \text{otherwise} \end{cases} \right\}_{\beta \in [\![|\mathrm{OR}|]\!]}$$

$$\pi_{\alpha^*,k_{\alpha^*}+1} = [B_i(\tau)]_2 = \mathsf{sk}^{-1} \cdot \mathsf{hsk}_{n+1}$$
$$\pi_{\alpha^*,k_{\alpha^*}+2} = [Q_Z(\tau)]_1 = \mathsf{hsk}_{n+2} + \mathsf{sk}^{-1} \cdot \mathsf{hsk}_{n+3}$$
$$\pi_{\alpha^*,k_{\alpha^*}+3} = [Q_x(\tau)]_1 = \mathsf{hsk}_{n+4}$$
$$\pi_{\alpha^*,k_{\alpha^*}+4} = [\hat{Q}(\tau)]_1 = \mathsf{hsk}_{n+5}$$

and set $\boldsymbol{\pi} = \big(\pi_{\alpha^*,1}, \dots, \pi_{\alpha^*,k_{\alpha^*}}, \pi_{\alpha^*,k_{\alpha^*}+1}, \pi_{\alpha^*,k_{\alpha^*}+2}, \pi_{\alpha^*,k_{\alpha^*}+3}, w_{\alpha^*,k_{\alpha^*}+4}\big)^\top$.

Finally, parse $\mathsf{ct}_{\alpha^*} = (\mathsf{ct}_{\alpha^*,1}, \mathbf{ct}_{\alpha^*,2}, \mathsf{ct}_{\alpha^*,3})$ and compute:

$$\mathsf{msg} = \mathsf{ct}_{\alpha^*,3} - \mathbf{ct}_{\alpha^*,2} \circ \boldsymbol{\pi}.$$

B.2 Our RTE Construction Unwrapped

For completeness, in this section we give the full description of our Registered Threshold Encryption scheme.

The WE equations are the following:

1. $[\sum_{\mathsf{id}\in\mathcal{U}} \mathsf{sk}_{\mathsf{id}} L^{\mathcal{U}}_{\mathsf{id}}(\tau)]_1 \circ [B(\tau)]_2 = [1]_2 \circ \mathsf{aPK} + [Z_{\mathcal{U}}(\tau)]_2 \circ [Q_Z(\tau)]_1 + [\tau]_2 \circ [Q_x(\tau)]_1$

2. $[\tau^2]_2 \circ [Q_x(\tau)]_1 = [1]_2 \circ [\hat{Q}_x(\tau)]_1$

3. $[\gamma]_2 \circ \mathsf{aPK} = [1]_1 \circ \hat{\sigma}$

4. $[\tau^{M-N+t+1}]_1 \circ [B(\tau)]_2 = [1]_2 \circ [\hat{B}(\tau)]_1$

5. $[1]_1 \circ [B(\tau)]_2 = [\tau - 1]_2 \circ [Q_0(\tau)]_1 + [1]_1 \circ [1]_2$

6. $[Z_{\mathcal{S}}(\tau)]_1 \circ [B(\tau)]_2 = [Z_{\mathcal{U}}(\tau)]_2 \circ [Q_{\mathcal{S}}(\tau)]_1$

In matrix form:

$$\begin{pmatrix} C & -[1]_2 & -Z_{\mathcal{U}} & -[\tau]_2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & [\tau^2]_2 & -[1]_2 & 0 & 0 & 0 & 0 \\ 0 & [\gamma]_2 & 0 & 0 & 0 & -[1]_1 & 0 & 0 & 0 \\ [\tau^{M-N+t+1}]_1 & 0 & 0 & 0 & 0 & 0 & -[1]_2 & 0 & 0 \\ [1]_1 & 0 & 0 & 0 & 0 & 0 & 0 & -Z_0 & 0 \\ Z_{\mathcal{S}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -Z_{\mathcal{U}} \end{pmatrix} \circ \begin{pmatrix} [B(\tau)]_2 \\ \mathsf{aPK} \\ [Q_Z(\tau)]_1 \\ [Q_x(\tau)]_1 \\ [\hat{Q}_x(\tau)]_1 \\ \hat{\sigma} \\ [\hat{B}(\tau)]_1 \\ [Q_0(\tau)]_1 \\ [Q_{\mathcal{S}}(\tau)]_1 \end{pmatrix} = \begin{pmatrix} [0]_T \\ [0]_T \\ [0]_T \\ [0]_T \\ [1]_T \\ [0]_T \end{pmatrix}$$

Construction 10. *Below is our ID-STE construction.*

- `Setup`$(1^\lambda, M)$*: Sample* $\tau \leftarrow\!\!\$\ \mathbb{F}_p$ *and output:*

$$\mathsf{crs} = \left([\tau^1]_1, \ldots, [\tau^{M+1}]_1, [\tau^1]_2, \ldots, [\tau^{M+1}]_2\right).$$

- `KGen`(crs)*: Sample* $x \leftarrow\!\!\$\ \mathbb{F}_p$ *and output*

$$\mathsf{sk} = x, \quad \mathsf{pk} = [x]_1, \quad \mathsf{hint} = \left([\mathsf{sk} \cdot \tau]_1, \ldots, [\mathsf{sk} \cdot \tau^{M+1}]_1\right).$$

- `isValid`$(\mathsf{crs}, \mathsf{pk}, \mathsf{hint})$*: Parse* $\mathsf{hint} := (\mathsf{hint}_1, \ldots, \mathsf{hint}_{M+1})$ *and output* 1 *if for each* $j \in [\![M+1]\!]$*:*

$$[\mathsf{pk}]_1 \circ [\tau^j]_2 = [\mathsf{hint}_j]_1 \circ [1]_2$$

- `Reg`$(\mathsf{crs}, \mathsf{aux}, \mathsf{id}, \mathsf{pk}, \mathsf{hint})$ *: Parse* $\mathsf{aux} := (\mathcal{U}, \{\mathsf{pk}_i, \mathsf{hint}_i\}_{i \in \mathcal{U}})$ *and set* $\mathcal{U}' \leftarrow \mathcal{U} \cup \{\mathsf{id}\}$ *and* $|\mathcal{U}'| := N+1$. *Compute the new Lagrange polynomials* $L_i^{\mathcal{U}'}(X) = \prod_{j \in \mathcal{U}'} \frac{X-j}{i-j}$ *for each* $i \in \mathcal{U}'$*, the new vanishing polynomial* $Z_{\mathcal{U}'}(X) = \prod_{i \in \mathcal{U}'}(X - i)$ *and then using the hints compute:*

$$\mathsf{mpk}' = \left(N+1, \left[\sum_{i \in \mathcal{U}'} \mathsf{sk}_i L_i^{\mathcal{U}'}(\tau)\right]_1, [Z_{\mathcal{U}'}(\tau)]_2\right), \quad \mathsf{aux}' = \left(\mathcal{U}', \{\mathsf{pk}_i, \mathsf{hint}_i\}_{i \in \mathcal{U}'}\right).$$

(The system is initialized with $\mathsf{mpk}_0 = (1, [L_1^{\{1\}}(\tau)]_1, [Z_{\{1\}}(\tau)]_2)$ *and* $\mathsf{aux} = \bot$*)*

- $\texttt{Enc}(\mathsf{mpk}, \mathcal{S}, t, \mathsf{msg})$: *Sample* $[\gamma]_2 \leftarrow\!\!\$\ \mathbb{G}_2$. *Parse* $\mathsf{mpk} := (N, C, Z_{\mathcal{U}})$, *compute* $Z_{\mathcal{S}}(X) = \prod_{\mathsf{id} \in \mathcal{S}}(X - \mathsf{id})$, *set* $Z_0 = [\tau - 1]_2$, $Z_{\mathcal{S}} = [Z_{\mathcal{S}}(\tau)]_2$ *and*

$$\boldsymbol{A} = \begin{pmatrix} C & -[1]_2 & -Z_{\mathcal{U}} & -[\tau]_2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & [\tau^2]_2 & -[1]_2 & 0 & 0 & 0 & 0 \\ 0 & [\gamma]_2 & 0 & 0 & 0 & -[1]_1 & 0 & 0 & 0 \\ [\tau^{M-N+t+1}]_1 & 0 & 0 & 0 & 0 & 0 & -[1]_2 & 0 & 0 \\ [1]_1 & 0 & 0 & 0 & 0 & 0 & 0 & -Z_0 & 0 \\ Z_{\mathcal{S}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -Z_{\mathcal{U}} \end{pmatrix}, \boldsymbol{b} = \begin{pmatrix} [0]_T \\ [0]_T \\ [0]_T \\ [0]_T \\ [1]_T \\ [0]_T \end{pmatrix}.$$

Sample a vector $\boldsymbol{s} \leftarrow\!\!\$\ (\mathbb{F}_p)^6$ *and output*

$$\mathsf{ct} = \left([\gamma]_2, \boldsymbol{s}^\top \boldsymbol{A}, \boldsymbol{s}^\top \boldsymbol{b} + \mathsf{msg}\right).$$

- $\texttt{PartDec}(\mathsf{sk}, \mathsf{ct})$: *Parse* $\mathsf{ct} := ([\gamma]_2, \mathbf{ct}_2, \mathsf{ct}_3)$ *and output*

$$\sigma = \mathsf{sk} \cdot [\gamma]_2.$$

- $\texttt{PartVerify}(\mathsf{ct}, \sigma, \mathsf{pk})$: *Parse* $\mathsf{ct} := ([\gamma]_2, \mathbf{ct}_2, \mathsf{ct}_3)$ *and output* 1 *if and only iff*

$$\mathsf{pk} \circ [\gamma]_2 = [1]_1 \circ \sigma.$$

- $\texttt{DecAggr}(\mathsf{crs}, \mathsf{aux}, \mathcal{S}, \{\sigma_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{D}}, \mathsf{ct})$: *Compute the set* $\mathcal{D}_V \subseteq \mathcal{D}$ *of identities with valid partial decryptions:*

$$\mathcal{D}_V := \big\{\mathsf{id} \in \mathcal{D} : \mathsf{id} \in \mathcal{S} \wedge \texttt{PartVerify}(\mathsf{ct}, \sigma_{\mathsf{id}}, \mathsf{pk}_{\mathsf{id}}) = 1\big\}.$$

Parse $\mathsf{aux} := (\mathcal{U}, \{\mathsf{pk}_i, \mathsf{hint}_i\}_{i \in \mathcal{U}})$ *and then, using this information, proceed as follows:*

1. *Compute a polynomial* $B(X)$ *by interpolating* 0 *on all* $\mathsf{id} \notin \mathcal{D}_V$ *and* 1 *on* 1, *i.e., interpolate* $B(X)$ *as* $B(1) = 1, \{B(\mathsf{id}) = 0\}_{\mathsf{id} \in \mathcal{U} \setminus \mathcal{D}_V}$. *Then evaluate* $B(X)$ *on* $\{\mathsf{id}\}_{\mathsf{id} \in \mathcal{D}_V}$ *and set*

$$w_1 = \left[\sum_{\mathsf{id} \in \mathcal{U}} B(\mathsf{id}) L_{\mathsf{id}}(\tau)\right]_2$$

2. *Set*

$$w_2 = \mathsf{aPK} = \frac{1}{|\mathcal{U}|}\Big(\sum_{\mathsf{id} \in \mathcal{D}_V} B(\mathsf{id})\mathsf{pk}_{\mathsf{id}} + [1]_1\Big)$$

3. *Compute*

$$w_3 = [Q_Z(\tau)]_1 = \left[\sum_{\mathsf{id} \in \mathcal{U}} B(\mathsf{id}) \left(\mathsf{sk}_{\mathsf{id}} \frac{L^2_{\mathsf{id}}(\tau) - L_{\mathsf{id}}(\tau)}{Z(\tau)}\right) + \left(\sum_{\mathsf{id} \in \mathcal{U}} B(\omega^i) \sum_{j \in \mathcal{U}, j \neq \mathsf{id}} \mathsf{sk}_j \frac{L_{\mathsf{id}}(\tau) L_j(\tau)}{Z(\tau)}\right)\right]_1$$

4. *Compute*

$$w_4 = [Q_x(\tau)]_1 = \left[\sum_{\mathsf{id} \in \mathcal{U}} B(\mathsf{id}) \left(\mathsf{sk}_{\mathsf{id}} \frac{L_{\mathsf{id}}(\tau) - L_{\mathsf{id}}(0)}{\tau}\right)\right]_1$$

5. *Compute*

$$w_5 = [\hat{Q}_x(\tau)]_1 = \left[\sum_{\mathsf{id}\in\mathcal{U}} B(\mathsf{id}) \cdot \mathsf{sk}_{\mathsf{id}} \left(L_{\mathsf{id}}(\tau) - L_{\mathsf{id}}(0)\right)\right]_1$$

6. *Set*

$$w_6 = \hat{\sigma} = \frac{1}{|\mathcal{U}|}(\sum_{\mathsf{id}\in\mathcal{D}_V} B(\mathsf{id})\sigma_{\mathsf{id}} + 1 \cdot \mathsf{ct}_1)$$

7. *Compute*

$$w_7 = [\hat{B}(\tau)]_1 = [\tau^{M-N+t+1} B(\tau)]_1$$

8. *Compute a KZG evaluation at* 1*, i.e., compute* $Q_0(X)$ *such that* $B(X) - 1 = Q_0(X)(X-1)$ *and set*

$$w_8 = [Q_0(\tau)]_1 = \left[\frac{B(\tau) - 1}{\tau - 1}\right]_1$$

9. *Compute a batch KZG evaluation at* $\mathcal{S}^*$*:*

$$w_9 = [Q_{\mathcal{S}}(\tau)]_1 = \left[\frac{B(\tau) - 0}{Z_{\mathcal{S}}(\tau)}\right]_1$$

Set $\boldsymbol{w} = (w_1, \dots, w_9)^\top$*, parse* $\mathsf{ct} = (\mathsf{ct}_1, \mathbf{ct}_2, \mathsf{ct}_3)$ *and output:*

$$\mathsf{msg}^* = \mathsf{ct}_3 - \mathbf{ct}_2 \circ \boldsymbol{w}$$