Dynamic Collusion Functional Encryption and Multi-Authority Attribute-Based Encryption[*]

Rachit Garg UT Austin[†]   Rishab Goyal UW-Madison[‡]   George Lu UT Austin[§]

Abstract

Functional Encryption (FE) is a powerful notion of encryption which enables computations and partial message recovery of encrypted data. In FE, each decryption key is associated with a function f such that decryption recovers the function evaluation $f(m)$ from an encryption of m. Informally, security states that a user with access to function keys $\mathsf{sk}_{f_1}, \mathsf{sk}_{f_2}, \ldots$ (and so on) can only learn $f_1(m), f_2(m), \ldots$ (and so on) but nothing more about the message. The system is said to be q-bounded collusion resistant if the security holds as long as an adversary gets access to at most $q = q(\lambda)$ decryption keys.

However, until very recently, all these works studied bounded collusion resistance in a *static model*, where the collusion bound q was a global system parameter. While the static model has led to many great applications, it has major drawbacks. Recently, Agrawal et al. (Crypto 2021) and Garg et al. (Eurocrypt 2022) introduced the *dynamic model* for bounded collusion resistance, where the collusion bound q was a fluid parameter, not globally set, but chosen by each encryptor. The dynamic model enabled harnessing many virtues of the static model, while avoiding its various drawbacks. In this work, we provide a generic compiler to upgrade any FE scheme from the static model to the dynamic model.

We also extend our techniques to multi-authority attribute-based encryption (MA-ABE). We show that bounded collusion MA-ABE supporting predicates that can be represented as an efficient computational secret sharing (CSS) scheme can be built from minimal assumptions. Efficient CSS schemes are known for access structures whose characteristic function can be computed by a polynomial-size monotone circuit under the existence of one-way functions [Yao89 (unpublished)]. Thus, our MA-ABE construction is the first MA-ABE scheme from standard assumptions for predicates beyond polynomial-size monotone boolean formula. Our construction also satisfies full adaptive security in the Random Oracle Model.

[*]This paper subsumes the results from an earlier unpublished work by the authors [GGL22]. This is the full version of the paper that appeared at PKC 2024.

[†]Email: `rachg96@cs.utexas.edu`.

[‡]Email: `rishab@cs.wisc.edu`. Research supported by Wisconsin Alumni Research Foundation.

[§]Email: `gclu@cs.utexas.edu`.

1 Introduction

Functional Encryption (FE) [SW05, BSW11] is a powerful generalization of public-key encryption [DH76] which enables fine-grained access over encrypted data. In such systems, a setup algorithm produces a master public-secret key pair $(\mathsf{mpk}, \mathsf{msk})$, where the master public key is made public and the master secret key is retained by an authority. Using mpk, any user can encrypt data m to produce a ciphertext ct. On the other hand, the authority can use the master secret key to generate partial decryption keys for authorized users, where the decryption key sk_f for a function f enables the key holder to compute the function output $f(m)$ from the ciphertext encrypting data m while learning nothing else about m.

The notion of "learning nothing else about m" is formalized in one of two ways – indistinguishability (IND) or simulation (SIM) based security. The intuition behind them can be jointly understood as follows. The attacker receives a challenge ciphertext ct encrypting a message m along with a polynomial number (say q) of decryption keys $\mathsf{sk}_{f_1}, \ldots, \mathsf{sk}_{f_q}$ for functions $f_1, \ldots, f_q$. In the case of SIM-security, an attacker cannot distinguish this from a 'simulated' distribution of keys and ciphertext where only the function evaluation $f_1(m), \ldots, f_q(m)$ are available to the simulator, and *not* the entire message. While in IND-security, an attacker cannot distinguish it from when the ciphertext ct encrypts another message m' where $f_i(m) = f_i(m')$ for all q functions.

Over the last several years, functional encryption has been extensively studied and shown to remarkably useful (irrespective of the underlying security model, IND/SIM) for numerous applications across cryptography and beyond. Depending upon the message space and function space supported, it defines a new encryption system with appropriate fine-grained access control. For example, public-key encryption can be viewed as FE with only the identity function, identity-based encryption (IBE) [Sha84, Coc01, BF01] can be viewed as FE with *point* functions, etc.

Bounded collusions. In this work, we study functional encryption in the widely popular bounded collusion model [SS10, GVW12]. The bounded collusion model states that the FE system is guaranteed to be secure so long as the attacker does not receive more than q decryption keys. The parameter q is referred to as the collusion bound. The bounded collusion model has been highly successful and very well-studied leading to numerous positive as well as negative results (see [DKXY02, SS10, GLW12, GVW12, AR17, Agr17, ISV+17, AS17, GKW18, CVW+18, AV19, WFL19] and references therein), and improving understanding of the complexity of designing FE more generally. Moreover, it captures the essence of security needed for many applications, and can be built from *low tech assumptions* such as public-key encryption in most cases. Thus, FE in the bounded collusion model overcomes the extremely high cost needed for full collusion resistance [GGH+13, JLS21, JLS22] (both in terms of cryptographic assumptions and concrete practical efficiency), and bypasses known impossibilities [AGVW13]. Furthermore, it readily gives solutions that are provably post-quantum secure, unlike fully collusion resistant FE where current solutions are either known to be post-quantum insecure [JLS21, JLS22] or have only conjectured security [WW21, GP21, BDGM20, DQV+21].

Until very recently, FE in the bounded collusion model was formalized as the collusion bound q being declared at the time of system setup, restricting it to be fixed for the entire lifetime of the FE system. This introduces an undesirable inflexibility as whenever more than q corruptions occur, the system becomes useless since no security is guaranteed! Moreover, the sizes of all system parameters grew polynomially with q, including all the public and secret keys.

In recent works by Agrawal et al. [AMVY21] and Garg et al. [GGLW22], this limitation in the existing formalization of bounded collusion model was explored. They observed that this inflexibility is simply an artefact of existing formalizations of the bounded collusion model, and not a true barrier. They referred to the existing models as *static bounded collusion model* (henceforth called the static model), and introduced a stronger yet more natural corruption model called the *dynamic bounded collusion model* (henceforth dynamic model).

In the dynamic model, the setup algorithm no longer needs a maximum collusion bound q! It removes this inflexibility by letting an encryptor *dynamically* decide the number of colluding users against which it desires protection. That is, an encryptor selects the collusion bound q for each ciphertext ct at encryption time. And, crucially, a secure FE system in the dynamic model provides a fine-grained "per-ciphertext guarantee", which says a ciphertext ct generated for collusion level q is guaranteed to be secure so long as the attacker does not receive more than q decryption keys. In other words, an encryptor can select different collusion levels while a decryption key holder can decrypt it all the same. As a consequence, in the dynamic model, only the size of the ciphertexts grows with q, while everything else is independent of q.

This new formalization preserves all the desirable features (such as low tech assumptions, concrete efficiency, post-quantum security etc) while giving much better flexibility for applications. [AMVY21, GGLW22] designed FE schemes for circuits in the dynamic model assuming the minimum and necessary[1] assumption of identity-based encryption (IBE). And, [AMVY21] also extended them to succinct FE schemes for circuits, and FE schemes for Turing Machines (TMs) and Nondeterministic Logspace (NL) by leveraging the hardness of learning with errors (LWE) assumption [Reg05].

A recurring theme in their constructions is to perform careful surgery of existing FE schemes secure in the static model to upgrade to the dynamic model. In a few words, the core idea in these works is to open up existing (static) FE constructions and exploit the key space compression offered by IBE to encode the collusion bound more efficiently in each ciphertext rather than the master public key. On a technical level, the existing FE constructions and their proofs in the dynamic model are not more challenging compared to the static model, but they definitely are far more complicated to describe due to the instrinsic adaptivity offered by dynamic model.[2]

This leads to an interesting predicament – static model is well understood and weaker, but constructions are comparatively easier to design; while dynamic model is newer and stronger, but constructions are technically more cumbersome to design. Ideally, we want to design FE in the dynamic model but as simply as have been designing them in the static model so far. Thus, it likely suggests that we have two pathways:

A. Stick with static model for an initial feasibility result, and later circle back to the (static) construction to make it dynamic à la the surgery approach [AMVY21, GGLW22].

[1] See [AMVY21, §7] for more details about minimality of IBE.

[2] Briefly, this is because existing constructions [AMVY21, GGLW22, AKM$^+$22] have to redo all the work done for proving security in the static model and then patch it to make it secure in the dynamic model. And, there is no compiler that can generically upgrade static model to dynamic model.

B. Adopt the dynamic model as the primary corruption model, and develop a holistic approach.

While B would be a better choice scientifically, it might be too much to ask for. Even historically, in similar situations, we have observed pathway A is more likely to be taken. E.g., CPA vs CCA security, generic group vs standard model, composite vs prime order bilinear groups, random oracle vs standard model etc. In all the aforementioned situations, the community more often adopted the easier to design security model as a first step, and later strengthened it to the stronger model.

Static to dynamic generically. This leads to the first question we ask:

Q- *Can we generically upgrade any FE in the static model to dynamic model without changing the function/message class?*

A bit more philosophically, the question we ask is whether the two aforementioned pathways be unified? Somewhat surprisingly, we answer affirmatively! In this work, we show:

Informal Theorem 1. *Assuming IBE, an FE scheme secure in the static model can be transformed into a secure FE scheme in the dynamic model.*

A bit formally, we show that a statically secure FE scheme, protecting against collusions of size $\leq \lambda$, can be lifted to an FE scheme that protects from dynamic collusions. Given IBE is a necessary assumption (see [AMVY21, §7]), thus the above theorem is the best one can hope for. We view our result to follow along the lines of similar (pseudo-)generic transformations known for other comparable models in the literature (such as for CPA-to-CCA [NY90, KW19], composite-to-prime-order-pairings [Fre10, Lew12, OT10, Att14], etc). This reduces the task of building an FE scheme in the general dynamic model to a comparatively easier target of building a λ-bounded collusion FE scheme. In fact our procedure works for both uniform and non-uniform models of computation, which in turn lends to new FE results in the dynamc model as mere corollaries [GSW21, Wee21] (further discussed in the main body).

While in the future, there might be scenarios when a direct construction for a particular function class might be more advantageous, our transformation would remain a good starting point. More importantly, it serves as a good foundation and clear indication that FE in the dynamic model is as easy/hard as in the static model for that function-message space.

1.1 Multi-Authority Attribute-Based Encryption

While FE has been a great abstraction to study encryption over the last few decades, it has its own limitations. One well-known limitation is the need of a central trusted authority necessary for generating and issuing decryption keys. There has been a long line of works focussed on mitigating this centralization issue starting with the works of Chase, Chow, Lewko, and Waters [Cha07, CC09, LW11]. They introduced the notion of decentralization for attribute-based encryption (ABE) systems, commonly referred to as “multi-authority" attribute-based encryption (MA-ABE).

Recall in ABE, each ciphertext ct encrypts the payload m under an access policy ϕ, and a decryption key sk_x is associated with an attribute $\mathbf{x}$.[3] The functionality ensures, given such a ciphertext-key pair, one can learn payload m so long as $\phi(\mathbf{x}) = 1$. While security states that the payload is hidden so long as an attacker receives decryption keys for only unsatisfying attributes.

In an MA-ABE system, anyone can become a key issuing authority. There is no longer one master authority, but multiple individual authorities each controlling only a portion of the attribute space. Without loss of generality, consider attributes to be n-bit strings with n key authorities where the i^{th} authority controls the i^{th} bit of the attribute and gives a partial decryption key for just that attribute bit (say $\mathsf{sk}_b^{(i)}$ denotes partial key from authority i for bit b).

During decryption, a user combines partial decryption keys $\mathsf{sk}_{x_1}^{(1)}, \ldots, \mathsf{sk}_{x_n}^{(n)}$ associated with the same global user identifier GID (necessary for avoiding a *mix&match* attack[4]) to decrypt the ciphertext so long as $\phi(\mathbf{x} = (x_1, \ldots, x_n)) = 1$. Security is still required to hold against users who possess an arbitrary number of unauthorized secret keys, with an additional challenge that some subset of the authorities could now be corrupted as well. What makes MA-ABE so desirable in practice is this significantly stronger corruption model where key authorities can now be corrupted too. Over the last decade, there have been numerous attempts with varying levels of success (see [Cha07, CC09, LW11, WFL19, OT20, AGT21, DKW21, DKW23] to name a few and references therein for complete history) in building MA-ABE schemes, with mostly investigating in the highly popular (by now standardized) GID-model.

In this work, we design MA-ABE with bounded collusion resistance and, following the theme of our work, we formally define it in the dynamic (as well as static) model. Very briefly, we say an MA-ABE scheme is (statically) q-bounded collusion secure: if an attacker that can corrupt an arbitrary set of key authorities in addition to receiving partial decryption keys from all honest key authorities on at most q unique GIDs, cannot guess the message (so long as the partial decryption keys for any particular GID does not satisfy the challenge predicate ϕ). In a few words, the restriction is not on corrupting the key authorities, but only on the number of honest partial keys an attacker can see. The dynamic model can be similarly defined by delaying the choice of collusion bound to the encryptor instead.

We show that, in the bounded collusion setting, MA-ABE supporting any predicate that can be represented as an efficient computational secret sharing (CSS) scheme can be built from general low tech assumptions. A bit formally, we show:

Informal Theorem 2. *Assuming public-key encryption, there exists an MA-ABE scheme for efficient CSS predicates secure in the static model, in the Random Oracle Model (ROM).*

In an unpublished work (mentioned in [Bei11], see also Vinod et al. [VNS⁺03]), Yao showed an efficient computational secret-sharing scheme for access structures whose characteristic function

[3]Note that we are sticking with the ciphertext-policy variant of attribute-based encryption, as is the norm while defining multi-authority ABE.

[4]It is well understood that since the key authorities are completely decentralized and work independently, multiple partial decryption keys for the same attribute bit can be combined arbitrarily with partial keys from other key authorities. Briefly, the issue is partial decryption keys for different users from two independent authorities can be used together. To get around this, Chase [Cha07] introduced the concept of using unique public global identifiers for each distinct user as a "linchpin" for tying partial keys together.

can be computed by a polynomial-size monotone circuit under the existence of one-way functions. Thus, by combining with Yao's secret sharing scheme, we obtain an MA-ABE scheme for monotone circuits in the static model from plain public-key encryption.

Corollary 1. *Assuming public-key encryption, there exists an MA-ABE scheme for polynomial-depth monotone circuits secure in the static model, in the Random Oracle Model (ROM).*

The most expressive state of the art MA-ABE scheme in the fully collusion resistant model [LW11] only supports monotone formulae (i.e., log-depth circuits), thus we can support a much more general class at the cost of relatively weaker security model. Moreover, our construction satisfies full adaptive security where the attacker can corrupt authorities and keys adaptively in any arbitrary order. Prior to our work, there were only two MA-ABE constructions [WFL19, DKW23] that achieved full adaptive security. Both these schemes support encrypting access policies computable using an $\mathbf{NC}^1$ circuit, where Datta et al. [DKW23] gave a fully collusion resistant scheme under standard pairing-based assumptions, while Wang et al. [WFL19] gave a bounded collusion scheme under DDH/LWE assumption. Our scheme supports encrypting polynomial-depth monotone circuits, while relying on a minimal assumption of public-key encryption. Somewhat interestingly (though not surprisingly), we can prove security of our construction in the standard model (without ROM) if we stick to a super selective model. Since we do not formally prove it in the current version, we simply state it as an interesting observation.

Observation 1. *Assuming public-key encryption, there exists an MA-ABE scheme for polynomial-depth monotone circuits secure in the super-selective static model.*

Finally, following the theme of our work, we also construct and prove an MA-ABE scheme in the dynamic collusion model. Here, we show:

Informal Theorem 3. *Assuming identity-based encryption, there exists an MA-ABE scheme for polynomial-depth monotone circuits in the* dynamic *collusion model (in ROM).*

We can make the same observation as above about proving security in the standard model (without ROM). It is an interesting problem to remove the dependence on ROM for proving full adaptive security. We quickly remind the reader that IBE is a necessary assumption (see [AMVY21, §7] for more details).

Lastly, we remark that due to the minimal nature of assumptions needed for our MA-ABE results, they can be instantiated from post-quantum assumptions such as LWE [Reg05, GPV08] or even LPN [Ale03, DGHM18]. This gives us the first post-quantum MA-ABE scheme (though in the bounded collusion model) for predicates beyond DNF, and the very first MA-ABE scheme from the hardness of learning parity with noise (LPN) assumption.

2 Technical Overview

The overview is split into two parts. First, we discuss our design of a generic accumulator for functional encryption systems. We show that such an accumulator is the main technical barrier that separates static and dynamic collusion models. In the second part, we look at how to build a multi-authority attribute-based encryption system for polynomial-sized monotone circuits.

Part I: <u>The FE Accumulator</u>

Consider the problem of succinctly representing 2^λ pairs of FE master keys $(\mathsf{mpk}_i, \mathsf{msk}_i)$ for $i \leq 2^\lambda$ using only polynomial space, i.e. $\mathsf{poly}(\lambda)$ space to store all 2^λ key pairs. That is, you are given a functional encryption system $\mathsf{FE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ for some message-function space $\mathcal{M}, \mathcal{F}$. Without loss of generality, suppose the FE.Setup algorithm takes λ bits of randomness. Thus, FE.Setup outputs 2^λ possible master public-secret key pairs, one for each randomness value. The problem is to come up with a compressed representation, say $(\mathsf{MPK}, \mathsf{MSK})$, of these 2^λ key pairs $(\mathsf{mpk}_i, \mathsf{msk}_i)$ with the following properties:

Succinctness. $|\mathsf{MPK}|, |\mathsf{MSK}| = \mathsf{poly}(\lambda)$, i.e. they have fixed polynomial size.

Functionality. There exist efficient algorithms $\mathsf{ENC}, \mathsf{KEYGEN}, \mathsf{DEC}$ such that $\mathsf{ENC}(\mathsf{MPK}, i, \cdot)$, $\mathsf{KEYGEN}(\mathsf{MSK}, i, \cdot)$, $\mathsf{DEC}(\mathsf{SK}_f, i, \cdot)$ gave similar functionality as $\mathsf{Enc}(\mathsf{mpk}_i, \cdot)$, $\mathsf{KeyGen}(\mathsf{msk}_i, \cdot)$, $\mathsf{Dec}(\mathsf{sk}_{f,i}, \cdot)$.

(Here $\mathsf{SK}_f, \mathsf{sk}_{f,i}$ corresponding to a function decryption key.)

Here the second property can be defined formally, but intuitively it just says that $(\mathsf{MPK}, \mathsf{MSK})$ along with these algorithms simulate the same functionality/behavior of actually using the original versions of the FE keys.

Interestingly, designing such an accumulatable version of FE is the main technical hurdle for closing the gap between bounded collusion FE schemes in the static vs dynamic models. All prior works in the dynamic model [AMVY21, GGLW22, AKM+22] essentially design their own versions of such an accumulatable FE system and use it, either explicitly or implicitly, to build an FE scheme in the dynamic model. Garg et al. [GGLW22] make it most explicit, and define a new object called Tagged Functional Encryption (Tagged FE).

Formally, a tagged FE system contains the same four algorithms – Setup, Enc, KeyGen, Dec. That is, as in regular static model FE scheme, Setup takes the collusion bound q as an input. The main difference as as follows: each ciphertext $\mathsf{ct}_{m,\mathsf{tag}_1}$ and secret key $\mathsf{sk}_{f,\mathsf{tag}_2}$ are now additionally associated with a tag value, tag_1 and tag_2 (respectively), in addition to a message m and a function f. The decryption correctness states that $\mathsf{Dec}(\mathsf{sk}_{f,\mathsf{tag}_2}, \mathsf{ct}_{m,\mathsf{tag}_1})$ is equal to $f(m)$ *iff* $\mathsf{tag}_1 = \mathsf{tag}_2$, otherwise it outputs $\perp$. For security, it should be the case that regular FE security holds for all tag values where an attacker make less than q key queries. It is crucial that the attacker is allowed to make an unrestricted number of key queries on as many tag values that it wants. Basically, for tag values where the collusion bound is not exceeded, FE security should still hold even if collusion bound is exceeded on other tag values.

It is straightforward to see that tagged FE is simply an accumulated version of an FE scheme in the static model. The point is tagged FE enables a succinct representation of an exponential number of static model FE key pairs, and we mentioned before, this captures the main technical challenge to design FE in the dynamic model. Note that the setup of a tagged FE system still takes the collusion bound as input. Later we discuss how tagged FE can be upgraded genericaly in a black-box way to dynamic model FE. For now, we focus on building tagged FE from any static model FE scheme.

Our Accumulation Compiler. The main technical component of our first result is a generic compiler to succinctly represent an exponential number of FE keys in polynomial space. That is, a generic compiler to go from static model FE to tagged FE.

Our first main observation is that a tagged FE is simply an identity-based static FE scheme. That is, one could visualize tagged FE as a generalization of static model FE similar to how IBE is a generalization of plain PKE. Recall in IBE, each ciphertext and function key is associated with an identity (or simply a 'tag' value). Thus, it seems like all we need to do is find a generic mechanism to embed identities/tags in both the secret keys and ciphertexts of a tagged FE scheme.

Coincidentally, a similar problem was solved in an unrelated context of building registration-based encryption [GHMR18, GHM$^+$19, GV20]. Although the motivation behind registration-based encryption was to solve the key-escrow problem, one could very easily visualize it as a mechanism to accumulate a large number of independently sampled PKE public keys into a short commitment where each public key is associated with an identity (and, so is the corresponding secret key as well). And, an encryptor only needs the short commitment to encrypt the message for any particular identity. Internally, these encryption schemes rely on the beautiful line of works studying non-black-box IBE constructions from simpler assumptions [DG17b, DG17a].

Our idea is to use these non-black-box techniques for compressing static model FE schemes. Looking within all these existing constructions, we notice that the usage of PKE in these prior works ([DG17b, DG17a, GHMR18, GHM$^+$19] to name a few) is not essential. Rather we view this overall line of work as providing a beautiful mechanism to embed identities in any type of FE, and not just in plain PKE. While embedding identities in fully collusion resistant FE schemes does not seem to be useful for any new applications, we find that embedding identities is very useful for bounded collusion FE. For us, this gives a simple and generic approach to dynamic model FE. We believe that our re-visualization of Döttling-Garg [DG17b, DG17a] garbling-based tree compression techniques to beyond PKE will find more applications in the future.

Focussing on static model FE as the base encryption system and combining it with the garbling-based compression techniques, we prove the following:

Informal Theorem 4. *Assuming IBE*[5]*, there exists a generic compiler for building tagged FE from static model FE.*

From Static Model to Tagged FE via Garbling. The core technical idea behind the non-black-box garbling techniques [DG17b, DG17a] was to delegate computation of actual ciphertext computation to the decryption algorithm via a sequence of cascaded garbled circuits [Yao82]. Abstractly, there is a sequence of λ garbled circuits where the i^{th} garbled circuit gives as output the wire labels for the $(i+1)^{th}$ garbled circuit. The last garbled circuit is special as the it performs the actual delegated computation of the real ciphertext. This basic abstract representation is not enough on its own as the wire keys for the $(i+1)^{th}$ garbled circuit (the ones that are part of

[5]Looking ahead, we actually use One-Time Signature with Encryption (OTSE) introduced in [DGHM18].

the i^{th} circuit's output) have to be encrypted such that only half of them are revealed during the actual decryption. This last part is crucial for security as it ensures all the garbled circuits can be simulated properly.

In this work, we use one-time signature with encryption (OTSE) for hiding wire labels in the above approach. They were introduced for building IBE generically from plain PKE via garbling [DGHM18]. In short, an OTSE scheme is a one-time signature scheme that is equipped with an encryption and decryption algorithm. Encryption is performed w.r.t. a verification key vk and a pair of message bit and index (i, b). The resulting ciphertext can be decrypted using any signature σ for some string x so long as the bit b matches corresponding bit in x (i.e., $m[i] = b$). Formally, a ciphertext ct is associated with an index-bit pair (i, b), and decryption works given a signature σ for any x such that $x[i] = b$. Combining OTSE with garbled circuits is sufficient for designing a generic accumulation compiler.

To give a quick overview, we start with a simpler goal. Suppose we only want *compress* and *embed* N independent instantiations of a static model FE scheme (referred to as base FE scheme for ease of exposition) into a tagged FE scheme. That is, all the parameter sizes should grow only as $\mathsf{poly}(\lambda, \log N)$, but now the running time of the setup and key generation algorithms can be large (say $\mathsf{poly}(\lambda, N)$). The core idea is as follows:

Setup: Sample N random key pairs $(\mathsf{mpk}_i, \mathsf{msk}_i)$ for base FE (using a PRF key K for generating randomness). Hash all N public keys to a short digest h using a Merkle tree. Set digest h as the short master public key mpk and PRF key K as the short master secret key msk.

Encrypt: Use the garbling technique describe above. It defers the base FE encryption to the decryption phase. It does this by generating a sequence of garbled circuits $C_1, \ldots, C_\lambda$. The final circuit C_λ outputs the base FE encryption under the correct key $\mathsf{mpk}_{\mathsf{tag}}$, while the intermediate circuits perform step-by-step verification of the root-to-leaf path where at each step only one bit of the tag value tag is read and verified.

Now as discussed above, all the garbled wire keys have to be encrypted (using OTSE). This enforces the decryptor to run the final garbled circuit only on $\mathsf{mpk}_{\mathsf{tag}}$. (Clearly, the setup has to be modified to ensure the hash tree is compatible with OTSE, and msk needs to contain OTSE signing keys.)

Key generation: It generates a decryption key for base FE w.r.t. master key $\mathsf{msk}_{\mathsf{tag}}$, and a sequence of OTSE signatures $\sigma_1, \ldots, \sigma_\lambda$ such that they enable evaluation of the intermediate garbled circuits along the appropriate root-to-leaf path.

Decrypt: Use the root-to-leaf path (in the form of OTSE signatures) to iteratively run garbled circuits from the ciphertext. It obtains the wire keys corresponding to the key $\mathsf{mpk}_{\mathsf{tag}}$, and then computes the base FE ciphertext w.r.t. $\mathsf{mpk}_{\mathsf{tag}}$ using the last garbled circuit. Decrypted the base FE ciphertext by using the corresponding base FE decryption key.

The above overview omits a lot of details, but the main intuition is that the root value of the Merkle tree serves as a short commitment to the sequence of N master public keys, and each leaf node can be succinctly opened w.r.t. the root node. Now, to encrypt a message for the tag^{th} base FE system (i.e., under key $\mathsf{mpk}_{\mathsf{tag}}$), the encryptor needs to search the entire Merkle tree to obtain the corresponding public key from the root node. But it cannot perform this search operation given only the root value, thus it defers it to the decryption phase by generating a sequence of garbled circuits. We suggest the reader to the main body for a formal description.

Note that this readily gives the following lemma:

Informal Lemma 1. *Assuming OTSE and garbled circuits, for any $N, q > 0$, there exists an explicit compiler from a static q-bounded collusion FE scheme to a q-bounded tagged FE scheme with tag space $[N]$.*

The above approach relies on the setup algorithm being able to explicitly hash down all the N master public keys, thus seems useful for compression of a polynomial sized sequence of master public keys only. However, prior works in this space [GKW16, DG17b, DG17a, GHMR18] have shown that a slightly more intricate design can be used to compress an exponential number of master keys too. The core trick is to use a simple lazy sampling technique where the setup algorithm uses a PRF key to deterministically generate the hash tree at the time of key generation instead. This enables deferring the setup algorithm's work to the key generator which only needs to open the tree along a particular path at a time. The usage of PRFs is essential for consistency (in turn security). More details are provided later in Section 5.

From Tagged FE to Dynamic Model via Combinatorics. It turns out that the main technical challenge in building FE in the dynamic model is captured in the tagged FE framework. And, tagged FE is powerful enough to build dynamic model FE via black-box transformations that only need simple combinatorial ideas used numerous times over the last decade in FE research. Briefly, Garg et al. [GGLW22] showed that the load balancing trick by Ananth and Vaikuntanathan [AV19] and powers-of-two technique by Goldwasser et al. [GKP+13] are sufficient to get dynamic model FE from tagged FE. Due to the non-cryptographic (combinatorial) nature of the techniques, this gives a simple black-box transformation that is oblivious to the underlying cryptography (and even the model of computation). Let us look at them one-by-one.

STEP 1. Consider the static model FE scheme to satisfy a special property called 'fast key generation'. The property says the running times of the setup and key generation algorithms grow as $\mathsf{poly}(\lambda, \log q)$. The collusion bound q is still a global system parameter, but it does not affect algorithms generating keys too much. It turns out the "powers-of-two trick" [GKP+13] can lift such static model FE schemes to the dynamic model. The core idea is to set up λ parallel static model FE systems with geometrically increasing collusion bounds from $q = 2, 4, \ldots, 2^\lambda$. The master keys and decryption keys are generated for each of the λ parallel systems. The point is the encryptor actually selects exactly one of these λ systems to encrypt. The choice depends on the desired level of collusion security q (i.e., select the $\lceil \log q \rceil^{th}$ static FE system for encryption). The security and correctness follow by design, while the 'fast key generation' property ensures efficiency.

STEP 2. The last missing piece is to translate a tagged FE scheme into a static FE scheme with *fast key generation*. The intuition here is (inspired from [AV19]) to visualize the collusion bound as a security "load" on the system. And, distribute the security "load" of q users into $O(q)$ buckets, each with a maximum security load of just λ users. The idea is to view each bucket as a separate tag value with collusion bound λ, and each decryption key gets tossed in one of those 'tag' buckets at random. Since tagged FE succinctly represents all $O(q)$ buckets within a single FE system, thus tagged FE gives a static FE scheme with desired poly-log dependence property via load balancing.

The above combinatorial ideas have appeared throughout the literature, especially in the bounded collusion regime. Prior works in the dynamic model [AMVY21, GGLW22, AKM$^+$22] have made them even more explicit to separate the cryptographic part from the combinatorial part. In this work, we show that the cryptographic component can be generically executed.

An Alternate and Simpler Approach for Accumulating FE.

In the above overview so far, we have explained our main technical idea for accumulating FE schemes. The technical centerpiece of our accumulator is to use a sequence of λ garbled circuits to generate a "tagged" ciphertext during decryption by using delegation of computation.

It turns out one can perform this "tagging" operation using just a single garbled circuit rather than a sequence of λ garbled circuits. The idea is to simply use the deferred encryption technique developed by Goyal et al. [GKW16]. While in the main body we still stick to the above OTSE approach for instantiating our accumulator, we describe the simpler alternate idea below for completeness and improved future designs.

During setup, we sample an IBE key pair $(\mathsf{mpk}, \mathsf{msk})$ as well as a PRF seed s. The master public key consists of mpk, and rest everything is kept secret. To encrypt a message m under a tag tag, the encryptor constructs a circuit that takes as input an FE master public key X and outputs $\mathsf{BFE.Enc}(X, m)$ – an encryption of m using X as the FE master public key.[6] The encryptor garbles this circuit and obtains a garbled circuit G and 2ℓ garbled circuit wires $\{w_{i,0}, w_{i,1}\}_{i\in[\ell]}$ where ℓ denotes the length of the FE public key. Finally, the encryptor outputs the garbled circuit G and 2ℓ IBE encryptions $\{\mathsf{ct}_{i,b} = \mathsf{IBE.Enc}(\mathsf{mpk}, (\mathsf{tag}, i, b), w_{i,b})\}_{i\in[\ell], b\in\{0,1\}}$.

To generate a secret key for function f corresponding to the tag tag, the key generator first samples an FE key pair as $(\mathsf{mpk}_{\mathsf{tag}}, \mathsf{msk}_{\mathsf{tag}}) \leftarrow \mathsf{BFE.Setup}(1^\lambda; \mathsf{PRF}(s, \mathsf{tag}))$. Next, it generates ℓ IBE secret keys as

$$\{\mathsf{IBE.sk}_{\mathsf{tag},i} \leftarrow \mathsf{IBE.KeyGen}(\mathsf{msk}, (\mathsf{tag}, i, \mathsf{mpk}_{\mathsf{tag}}[i]))\}_{i\in[\ell]}$$

That is, it generates one IBE secret key for each identity in $(\mathsf{tag}, 1, \mathsf{mpk}_{\mathsf{tag}}[1]), \ldots, (\mathsf{tag}, \ell, \mathsf{mpk}_{\mathsf{tag}}[\ell])$. In other words, it encodes the FE master public key inside the IBE keys. Further, the key generator samples an FE key as $\mathsf{BFE.KeyGen}(\mathsf{msk}_{\mathsf{tag}}, f)$, and include this along with the ℓ IBE keys as part of the final tagged secret key. In order to decrypt the ciphertext, a user proceeds in two steps. First, it recovers the wire labels as $\mathsf{IBE.Dec}(\mathsf{IBE.sk}_{\mathsf{tag},i}, \mathsf{ct}_{i,\mathsf{mpk}_{\mathsf{tag}}[i]})$, and runs the garbled circuit G on these wire labels to recover the actual function output. The correctness follows directly from correctness of garbling, untagged FE scheme, and IBE.

The proof of security also follows via a simple sequence of garbled circuits. First, one can use PRF security to argue that the PRF output is indistinguishable from a truly random function. Once we replace all PRF outputs with truly random values, then in the next hybrid we can use IBE security to replace half of the IBE ciphertexts from encryptions of real wire keys to IBE

[6] We ignore the random coins used during encryption for simplicity, however they can be easily hardwired inside such a circuit.

encryptions of zero strings. This can be done because for each unique tag string tag, only one FE master key $\mathsf{mpk}_{\mathsf{tag}}$ is sampled. Thus, at most half of the IBE ciphertexts in the challeneg ciphertexts can be decrypted using available IBE secret keys. Next, we can use simulation security of the garbling scheme to replace the garbled circuit G with a simulated circuit that is generated using a randomly generated FE encryption of challenge message m. Finally, we can use bounded collusion FE security of the underlying untagged FE scheme to finish the proof, since the adversary only learns a single FE ciphertext (inside the simulated garbled circuit) and a bounded number of FE secret keys for the corresponding tagged FE instance.

As mentioned earlier, in the main body, we present our FE accumulator using OTSE as a starting point. The above direct scheme is a simpler alternate approach.

Part II: MA-ABE from simple assumptions.

In multi-authority ABE, there are n key authorities where the i^{th} authority generates its local keys $\mathsf{pk}_u, \mathsf{sk}_u$. An encryptor picks a subset U of authorities under whose public keys $\{\mathsf{pk}_u\}_{u \in U}$ it encrypts a message μ along with policy ϕ. For our MA-ABE construction, we consider the class of all predicates that can be represented as an efficient computational secret sharing (CSS) scheme.

Secret sharing. Recall in an efficient CSS for n parties, there is a dealer algorithm that given a secret s and a description of an access structures $\mathcal{A}$, runs in polynomial time, outputs a set of n shares $\mathsf{sh}_1, \ldots, \mathsf{sh}_n$.[7] More importantly, there is a reconstruction algorithm that given a subset of shares $\{\mathsf{sh}_i\}_{i \in T}$, for some set $T \subseteq [n]$, outputs the reconstructed secret s so long as $T \in \mathcal{A}$ (i.e., T is an authorized set). The secret sharing scheme is secure if the secret s is computationally hidden from every group of *unauthorized* parties.

Getting back to MA-ABE, we view our encryptor to receive the policy ϕ as the description of an access structure $\mathcal{A}$ (for which efficient CSS exists) along with a mapping $\rho : [n] \to U$ from the party index to authority index (recall U is the set of authorities used by encryptor). That is, $\rho(i)$ tells the index of authority from the set $U := \{u_1, \ldots, u_\ell\}$, for some $\ell > 0$, that controls the attribute for the i^{th} party. For simplicity, consider that the number of authorities and parties (as defined by $\mathcal{A}$) is the same (say n), and ρ is simply the identity function.

During key generation, each authority receives a global identifier GID and it outputs a partial predicate key $\mathsf{sk}_{\mathsf{GID},u}$. One should read this as if a user receives a predicate key from authority u, then it is authorized to receive the u^{th} share from the shares generated by the CSS. Lastly, during decryption, a user combines all the partial predicate keys it obtained $\{\mathsf{sk}_{\mathsf{GID},u}\}_{u \in T}$, for some set $T \subseteq [n]$, to decrypt the ciphertext, and outputs the message μ as long as T corresponds to an authorized set w.r.t. $\mathcal{A}$.

MA-ABE for CSS. Our starting point is the Sahai-Seyalioglu [SS10] construction for 1-bounded collusion FE. Clearly, the same construction can be directly extended to a 1-bounded collusion ABE scheme. Briefly, the idea is:

[7]In some formalisms, the dealer outputs a public share sh_0 as well which is said to be available to all parties. For simplicity, we consider it to be a part of each party's share since it does not affect the asymptotic efficiency too much.

Setup: Sample $2n$ PKE public-secret key pairs $\mathsf{pk}_{i,b}, \mathsf{sk}_{i,b}$. Keep all public and all secret PKE keys as the master public and secret key, respectively.

Encrypt: To encrypt a message μ for predicate C, garble the circuit $\mathsf{Test}_{C,\mu}$ which on input an attribute $\mathbf{x}$ outputs μ *iff* $C(\mathbf{x}) = 1$. Encrypt each wire key $w_{i,b}$ under key $\mathsf{pk}_{i,b}$, and release garbled circuit with encrypted wire keys as the ciphertext.

Key generation: For attribute $\mathbf{x}$, the predicate key simply contains n PKE secret keys sk_{i,x_i}.

Decrypt: Decrypt the encrypt wire keys, and use it to evaluate the garbled circuit.

It might seem that Sahai-Seyalioglu scheme readily gives a multi-authority ABE scheme. The idea might be to have the i^{th} authority generate two PKE key pairs $\mathsf{pk}_{i,b}, \mathsf{sk}_{i,b}$ for $b \in \{0, 1\}$. Note that this way the predicate key generation can also be distributed very easily. *Unfortunately, this is not the case!* The problem is the garbled circuit security crucially relies on only half of the wire labels to ever be revealed, but in MA-ABE setting, an attacker can corrupt key authorities. This completes breaks down extending this to the multi-authority setting.

Our idea is to make this basic approach compliant with the multi-authority model. The problem was if an attacker can corrupt key authorities, then garbled circuits does not provide the right protection. However, there is a simple fix – efficient CSS. In a few words, our idea is to replace garbled circuits with CSS, and rely on secrecy of CSS instead of garbling security. Formally, we do as follows:

Authority setup: Sample a $\underline{\text{single}}$ PKE public-secret key pair $\mathsf{pk}_u, \mathsf{sk}_u$.

Encrypt: To encrypt a message μ for access structure $\mathcal{A}$, $\underline{\text{compute a CSS of } \mu \text{ for } \mathcal{A}}$ as $\mathsf{sh}_1, \ldots, \mathsf{sh}_n$. Simply encrypt share sh_u under key pk_u.

Key generation: The predicate key is simply sk_u.

Decrypt: Decrypt the encrypted shares and use them to reconstruct the message.

Amazingly, this simple construction gives us a 1-bounded MA-ABE for efficient CSS from just public key encryption. The idea is simply to use semantic security to hide all unauthorized shares, and then use secrecy of CSS to hide the message. The remaining goal is to upgrade to q-bounded security (i.e., static model), and dynamic model eventually.

To that end, we revisit the simple strategy to go from 1-bounded collusion to q-bounded collusion for plain ABE. The idea there is to use repetition and add enough redundancy in the system. Concretely, one runs a large polynomial number of 1-bounded ABE systems in parallel, and a predicate key contains 1-bounded ABE predicate keys for a 'small' subset of them. The encryptor then threshold secret shares the message μ where each share is independently encrypted under each 1-bounded ABE system. By setting up the parameters carefully, this gives us security in the static model for ABE.

The question is whether we can replay a similar strategy in the multi-authority setting to mirror a similar collusion bound amplification. While this might seem natural, it is not immediately clear. The concern is two-fold:

1. In multi-authority setting, each authority works fully independently and asynchronously. Forget security, just for correctness it is essential that each authority selects the same subset of 1-bounded MA-ABE for a particular user. How can such a subset be computed?

2. Moreover, a core idea in the collusion bound amplification is to use a standard probabilistic argument [GVW12] that says randomly chosen subsets of small size will not have a large combined pairwise intersection. How can we use a probabilistic argument when some of the key authorities can get corrupted? (That is, their choices of subsets might not be truly random.)

Both these issues have simple solutions, but they are conflicting. The problem is to solve the first issue, we need to make the subset selection process deterministic. But, having the subsets be deterministically selected seems problematic for using a standard probabilistic argument such as cover-freeness.

We notice that if the adversary commits to all its corruptions at the beginning, then we could use an (almost) perfect hash function to deterministically select these subsets. Simply set the subset to be the hash of GID. (Recall an almost perfect hash function from $\{0,1\}^* \to [\mathsf{poly}(n)]$ maps n inputs to $\mathsf{poly}(n)$ numbers such that there are no collisions.) The construction now is very simple – each authority selects the subset of 1-bounded systems to use depending upon $H(\mathsf{GID})$. This solves both the above issues since $H(\mathsf{GID})$ is deterministic (and out of adversary's hand) as well as it has nice pairwise intersection property due to our assumption of it being a perfect hash function. It turns out a careful analysis of above approach could be used to prove super-selective security of our MA-ABE scheme in the static model. This is due to the fact that PRFs can be mostly used to design such near-perfect hash functions in the super-selective setting.

However, perfect hashes do not exist if the attacker gets to see the hash key! Moreover, even collision resistant hash functions are not good enough as the digest size needs to be very small so that the encryptor can enumerate over all possible digest values. This is due to the fact that the encryptor in the above strategy has to encrypt a secret share for each possible digest value. While this might seem like a tricky problem to bypass, it turns out modeling the hash function as a random oracle gives us the desired property. A (non-programmable) ROM is sufficient to show that an attacker cannot find q distinct GIDs where a special combinatorial property does not hold. We discuss this further in Section 6.

The above ideas can be formalized to design an MA-ABE for CSS in the static model, and it can be very easily extended to the dynamic model. We discuss this in Section 6.4, but briefly remark here. Our approach is to first extend the above construction to a tagged version of MA-ABE. This can be done quite simply by relying on IBE as the only change would be that an encryptor uses IBE encryption to encrypt the secret shares where the identity is set to be the corresponding tag value. Next, by relying on the combinatorial ideas, discussed previously for the FE accumulator, we can upgrade its security to hold in the dynamic collusion model too.

3 Preliminaries

Notations. Let PPT denote probabilistic polynomial-time. For any integer $q \geq 2$, we let $\mathbb{Z}_q$ denote the ring of integers modulo q. We denote the set of all positive integers upto n as

$[n] := \{1, \ldots, n\}$. For any finite set S, $x \leftarrow S$ denotes a uniformly random element x from the set S. Similarly, for any distribution $\mathcal{D}$, $x \leftarrow \mathcal{D}$ denotes an element x drawn from distribution $\mathcal{D}$. The distribution $\mathcal{D}^n$ is used to represent a distribution over vectors of n components, where each component is drawn independently from the distribution $\mathcal{D}$. Two distributions $\mathcal{D}_1$ and $\mathcal{D}_2$, parameterized by security parameter λ, are said to be computationally indistinguishable, represented by $\mathcal{D}_1 \approx_c \mathcal{D}_2$, if for all PPT adversaries $\mathcal{A}$, $\Pr[\mathcal{A}(x) = 1 : x \leftarrow \mathcal{D}_1] - \Pr[\mathcal{A}(x) = 1 : x \leftarrow \mathcal{D}_2] \leq \mathsf{negl}(\lambda)$.

3.1 Pseudorandom Functions

A pseudorandom function (PRF) is a function that takes inputs in domain $\mathcal{D}_\lambda$, samples a PRF seed of λ bits and computes an output in the range, $\mathcal{R}_\lambda$. The evaluation function runs polynomially in λ and satisfies the following pseudorandomness property.

Definition 3.1 (Pseudorandomness). A PRF scheme is said to be secure if for every stateful PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $\mathcal{D}_\lambda, \mathcal{R}_\lambda$ the following holds:

$$\Pr\left[\mathcal{A}^{\mathsf{PRF}(s,\cdot)}(r_b) = b : \begin{array}{c} s \leftarrow \{0,1\}^\lambda,\ b \leftarrow \{0,1\} \\ x^* \in \mathcal{D}_\lambda \leftarrow \mathcal{A}^{\mathsf{PRF}(s,\cdot)}(1^\lambda) \\ r_0 \leftarrow \mathcal{R}_\lambda,\ r_1 = \mathsf{PRF}(s, x^*) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

where $\mathcal{A}$ must not query the challenge input x^* to the evaluation oracle $\mathsf{PRF}(s, \cdot)$.

3.2 Garbled Circuits

Our definition of garbled circuits [Yao86] is based upon the work of Bellare et al. [BHR12]. Let $\{\mathcal{C}_n\}_n$ be a family of circuits where each circuit in $\mathcal{C}_n$ takes n bit inputs. A garbling scheme GC for circuit family $\{\mathcal{C}_n\}_n$ consists of polynomial-time algorithms Garble and Eval with the following syntax.

$\mathsf{Garble}(1^\lambda, C \in \mathcal{C}_n)$: The garbling algorithm takes as input the security parameter λ and a circuit $C \in \mathcal{C}_n$. It outputs a garbled circuit $\widetilde{C}$, together with $2n$ wire keys $\{w_{i,b}\}_{i \leq n, b \in \{0,1\}}$.

$\mathsf{Eval}(\widetilde{C}, \{w_i\}_{i \leq n})$: The evaluation algorithm takes as input a garbled circuit $\widetilde{C}$ and n wire keys $\{w_i\}_{i \leq n}$ and outputs $y \in \{0, 1\}$.

Correctness. A garbling scheme GC for circuit family $\{\mathcal{C}_n\}_n$ is said to be correct if for all λ, n, $x \in \{0,1\}^n$ and $C \in \mathcal{C}_n$, $\mathsf{Eval}(\widetilde{C}, \{w_{i,x_i}\}_{i \leq n}) = C(x)$, where $(\widetilde{C}, \{w_{i,b}\}_{i \leq n, b \in \{0,1\}}) \leftarrow \mathsf{Garble}(1^\lambda, C)$.

Security. Informally, a garbling scheme is said to be secure if for every circuit C and input x, the garbled circuit $\widetilde{C}$ together with input wires $\{w_{i,x_i}\}_{i \leq n}$ corresponding to some input x reveals only the output of the circuit $C(x)$, and nothing else about the circuit C or input x.

Definition 3.2. A garbling scheme $\mathsf{GC} = (\mathsf{Garble}, \mathsf{Eval})$ for a class of circuits $\mathcal{C} = \{\mathcal{C}_n\}_n$ is said to be a secure garbling scheme if there exists a polynomial-time simulator Sim such that for all n, $C \in \mathcal{C}_n$ and $x \in \{0,1\}^n$, the following distributions are computationally indistinguishable:

$$\left\{\mathsf{Sim}\left(1^\lambda, 1^n, 1^{|C|}, C(x)\right)\right\}_\lambda \approx_c \left\{\left(\widetilde{C}, \{w_{i,x_i}\}_{i\le n}\right) \ : \ \left(\widetilde{C}, \{w_{i,b}\}_{i\le n, b\in\{0,1\}}\right) \leftarrow \mathsf{Garble}(1^\lambda, C)\right\}_\lambda .$$

The following corollary follows from the definition.

Corollary 3.3. If GC is a secure garbling scheme for a class of circuits $\mathcal{C} = \{\mathcal{C}_n\}_n$, then for all n, $C \in \mathcal{C}_n$ and $x \in \{0,1\}^n$, the following distributions are computationally indistinguishable:

$$\left\{\mathsf{Sim}\left(1^\lambda, 1^n, 1^{|C|}\right)\right\}_\lambda \approx_c \left\{\widetilde{C} \ : \ \left(\widetilde{C}, \{w_{i,b}\}_{i\le n, b\in\{0,1\}}\right) \leftarrow \mathsf{Garble}(1^\lambda, C)\right\}_\lambda .$$

While this definition is not as general as the definition in [BHR12], it suffices for our construction.

3.3 Identity-Based Encryption

An Identity-Based Encryption (IBE) scheme IBE for set of identity spaces $\mathcal{I} = \{\mathcal{I}_n\}_{n\in\mathbb{N}}$ and message spaces $\mathcal{M}$ consists of four polynomial time algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ with the following syntax:

$\mathsf{Setup}(1^\lambda, 1^n) \to (\mathsf{mpk}, \mathsf{msk})$. The setup algorithm takes as input the security parameter λ and identity space index n. It outputs the public parameters mpk and the master secret key msk.

$\mathsf{KeyGen}(\mathsf{msk}, \mathsf{id}) \to \mathsf{sk}_{\mathsf{id}}$. The key generation algorithm takes as input the master secret key msk and an identity $\mathsf{id} \in \mathcal{I}_n$. It outputs a secret key $\mathsf{sk}_{\mathsf{id}}$.

$\mathsf{Enc}(\mathsf{mpk}, \mathsf{id}, m) \to \mathsf{ct}$. The encryption algorithm takes as input the public parameters mpk, a message $m \in \mathcal{M}$, and an identity $\mathsf{id} \in \mathcal{I}_n$. It outputs a ciphertext ct.

$\mathsf{Dec}(\mathsf{sk}_{\mathsf{id}}, \mathsf{ct}) \to m/\bot$. The decryption algorithm takes as input a secret key $\mathsf{sk}_{\mathsf{id}}$ and a ciphertext ct. It outputs either $m \in \mathcal{M}$ or a special symbol $\bot$.

Correctness. We say an IBE scheme $\mathsf{IBE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ satisfies correctness if for all $\lambda, n \in \mathbb{N}$, $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$, $\mathsf{id} \in \mathcal{I}_n$, $m \in \mathcal{M}$, $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id})$, and $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{id}, m)$, we have that $\mathsf{Dec}(\mathsf{sk}_{\mathsf{id}}, \mathsf{ct}) = m$.

Definition 3.4. We say an IBE scheme $\mathsf{IBE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is secure if for any stateful PPT adversary $\mathcal{A}$ there exists a negligible function $\mathsf{negl}(\cdot)$, such that for all $\lambda, n \in \mathbb{N}$, the following holds

$$\Pr\left[\mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{st}, \mathsf{ct}) \ = b : \begin{array}{c} (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n); \quad b \leftarrow \{0,1\} \\ (m_0, m_1, \mathsf{id}^*) \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(1^\lambda, 1^n, \mathsf{mpk}) \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{id}^*, m_b) \end{array}\right] \le \frac{1}{2} + \mathsf{negl}(\lambda),$$

where all identities id queried by $\mathcal{A}$ satisfy $\mathsf{id} \neq \mathsf{id}^*$.

3.4 One-Time Signature with Encryption

A One-Time Signature with Encryption (OTSE) scheme, defined by [DG17a], is a one-time signature scheme with an additional encryption and decryption functionality. An OTSE scheme, consists of the following five algorithms, (SSetup, SGen, SSign, SEnc, SDec)[8].

$\mathsf{SSetup}(1^\lambda, \ell) \to \mathsf{pp}$. The setup algorithm takes as input the security parameter λ and a message length parameter ℓ. It outputs a public parameter pp.

$\mathsf{SGen}(\mathsf{pp}) \to (\mathsf{vk}, \mathsf{sk})$. Given parameters pp as input, this outputs a verification and signing key.

$\mathsf{SSign}(\mathsf{pp}, \mathsf{sk}, x) \to \sigma$. On input the public parameters, a signing key and a message $x \in \{0,1\}^\ell$, it outputs a signature σ.

$\mathsf{SEnc}(\mathsf{pp}, (\mathsf{vk}, i, b), m) \to \mathsf{ct}$. On input the public parameters, and a verification key vk, a position $i \in [\ell]$, a bit $b \in \{0,1\}$ and a plaintext m, it outputs a ciphertext ct.

$\mathsf{SDec}(\mathsf{pp}, (\mathsf{vk}, \sigma, x), \mathsf{ct}) \to m'$. On input the public parameters, a verification key vk, a signature σ, a message x and ciphertext ct, it outputs a plaintext m'.

Correctness. For all $\lambda \in \mathbb{N}, \ell \in \mathbb{N}, x \in \{0,1\}^\ell, i \in [\ell]$ and plaintext m, if $\mathsf{pp} \leftarrow \mathsf{SSetup}(1^\lambda, \ell)$, $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{SGen}(\mathsf{pp})$, $\sigma \leftarrow \mathsf{SSign}(\mathsf{pp}, \mathsf{sk}, x)$, and $\mathsf{ct} \leftarrow \mathsf{SEnc}(\mathsf{pp}, (\mathsf{vk}, i, x_i), m)$, the following holds

$$\mathsf{SDec}(\mathsf{pp}, (\mathsf{vk}, \sigma, x), \mathsf{ct}) = m.$$

Succinctness. For $\mathsf{pp} \leftarrow \mathsf{SSetup}(1^\lambda, \ell)$, $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{SGen}(\mathsf{pp})$, it holds that vk is a polynomial function in λ and independent of ℓ.

Selective Security. For every PPT Adversary $\mathcal{A}$, there exists a negligible function negl such that for all $\lambda \in \mathbb{N}, \ell \in \mathbb{N}$, the following holds.

$$\Pr\left[\mathcal{A}(\mathsf{pp}, \mathsf{vk}, \sigma, \mathsf{ct}^*) = b : \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{SSetup}(1^\lambda, \ell), \quad x \leftarrow \mathcal{A}(\mathsf{pp}) \\ (\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{SGen}(\mathsf{pp}), \quad \sigma \leftarrow \mathsf{SSign}(\mathsf{pp}, \mathsf{sk}, x) \\ (i, m_0^*, m_1^*) \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{vk}, \sigma), \quad b \leftarrow \{0,1\} \\ \mathsf{ct}^* \leftarrow \mathsf{SEnc}(\mathsf{pp}, (\mathsf{vk}, i, 1 - x_i), m_b^*) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

where the probability depends on the randomness of all OTSE algorithms as well as the attacker. In our proof we use the multi-message security, wich is implied by a single message security via a standard hybrid argument.

[8]There is no need for a verification algorithm for the signature as one can run the encryption and decryption algorithm to check validity of the signature.

4 Functional Encryption: Definitions

In this section, we revisit the notion of functional encryption (FE) in the bounded setting [SS10, GVW12]. Recent works of [AMVY21, GGLW22] proposed a collusion bound in the dynamic setting where the scheme's setup and key generation algorithms are independent of the collusion bound and instead, we can specify the collusion bound during encryption. We follow the formal security definitions from [GGLW22] almost verbatim and describe them below.

4.1 Static Collusion Model

Syntax. Let $\mathcal{M} = \{\mathcal{M}_n\}_{n\in\mathbb{N}}$, $\mathcal{R} = \{\mathcal{R}_n\}_{n\in\mathbb{N}}$ be families of sets, and $\mathcal{F} = \{\mathcal{F}_n\}$ a family of functions, where for all $n \in \mathbb{N}$ and $f \in \mathcal{F}_n$, $f : \mathcal{M}_n \to \mathcal{R}_n$. We will also assume that for all $n \in \mathbb{N}$, the set $\mathcal{F}_n$ contains an *empty function* $\epsilon_n : \mathcal{M}_n \to \mathcal{R}_n$. As in [BSW11], the empty function is used to capture information that intentionally leaks from the ciphertext.

A bounded functional encryption scheme FE for a family of function classes $\{\mathcal{F}_n\}_{n\in\mathbb{N}}$, message spaces $\{\mathcal{M}_n\}_{n\in\mathbb{N}}$ and collusion bound $q(\lambda)$ consists of four polynomial-time algorithms $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ with following semantics:

$\mathsf{Setup}(1^\lambda, 1^n, 1^q) \to (\mathsf{mpk}, \mathsf{msk})$. The setup algorithm takes as input the security parameter λ, the functionality index n[9] and the collusion bound 1^q. It outputs the master public-secret key pair $(\mathsf{mpk}, \mathsf{msk})$.

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

$\mathsf{KeyGen}(\mathsf{msk}, f \in \mathcal{F}_n) \to \mathsf{sk}_f$. The key generation algorithm takes as input the master secret key msk and a function $f \in \mathcal{F}_n$ and outputs sk_f.

$\mathsf{Dec}(\mathsf{sk}_f, \mathsf{ct}) \to y \in \mathcal{R}_n$. The decryption algorithm takes as input a ciphertext ct and a secret key sk_f and outputs a value $y \in \mathcal{R}_n$.

Correctness and Efficiency. A functional encryption scheme FE is said to be correct if for all $\lambda, n, q \in \mathbb{N}$, functions $f \in \mathcal{F}_n$, messages $m \in \mathcal{M}_n$ and $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, 1^q)$, we have that

$$\Pr\left[\mathsf{Dec}(\mathsf{KeyGen}(\mathsf{msk}, f), \mathsf{Enc}(\mathsf{mpk}, m)) = f(m)\right] = 1.$$

And, it is said to be efficient if the running time of the algorithms is a fixed polynomial in the parameters λ, n and q.

Static bounded collusion security. This is formally captured via the following 'simulation based' security definition as follows.

[9]One could additionally consider the setup algorithm to take as input a sequence of functionality indices where the function class and message space are characterized by all such indices (e.g., having input length and circuit depth as functionality indices). For ease of notation, we keep a single functionality index in the above definition.

Definition 4.1 (static-bounded-collusion simulation-security)**.** A functional encryption scheme FE is said to be *statically-bounded-collusion* simulation-secure if there exists a stateful PPT simulator $\mathsf{Sim} = (\mathsf{S}_0, \mathsf{S}_1, \mathsf{S}_2, \mathsf{S}_3)$ such that for every stateful PPT adversary $\mathcal{A}$, the following distributions are computationally indistinguishable:

$$\left\{ \mathcal{A}^{O(\cdot)}(\mathsf{ct}) : \begin{array}{c} (1^n, 1^q) \leftarrow \mathcal{A}(1^\lambda) \\ (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, 1^q) \\ m \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk}, \cdot)}(\mathsf{mpk}) \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, m) \\ O(\cdot) = \mathsf{KeyGen}(\mathsf{msk}, \cdot) \end{array} \right\}_{\lambda \in \mathbb{N}} \approx_c \left\{ \mathcal{A}^{O(\cdot)}(\mathsf{ct}) : \begin{array}{c} (1^n, 1^q) \leftarrow \mathcal{A}(1^\lambda) \\ (\mathsf{mpk}, \mathsf{st}_0) \leftarrow \mathsf{S}_0(1^\lambda, 1^n, 1^q) \\ m \leftarrow \mathcal{A}^{\mathsf{S}_1(\mathsf{st}_0, \cdot)}(\mathsf{mpk}) \\ (\mathsf{ct}, \mathsf{st}_2) \leftarrow \mathsf{S}_2(\mathsf{st}_1, \Pi^m) \\ O(\cdot) = \mathsf{S}_3^{U_m(\cdot)}(\mathsf{st}_2, \cdot) \end{array} \right\}_{\lambda \in \mathbb{N}}$$

whenever the following admissibility constraints and properties are satisfied:

- S_1 and S_3 are stateful in that after each invocation, they updates their states st_1 and st_3 (respectively) which is carried over to its next invocation.
- Π^m contains a list of functions f_i queried by $\mathcal{A}$ in the pre-challenge phase along with the their output on the challenge message m. That is, if f_i is the i-th function queried by $\mathcal{A}$ to oracle S_1 and q_{pre} be the number of queries $\mathcal{A}$ makes before outputting m, then $\Pi^m = \big((f_1, f_1(m)), \ldots, (f_{q_{\mathsf{pre}}}, f_{q_{\mathsf{pre}}}(m))\big)$.
- $\mathcal{A}$ makes at most q total key generation queries.
- S_3, for each queried function f_i, makes a single query to its message oracle U_m on the same f_i, and gets output as $f_i(m)$.

4.2 Dynamic Collusion Model

In the "dynamic" bounded collusion model [AMVY21, GGLW22], the scheme is no longer tied to a single collusion bound q fixed a-priori at the system setup, but instead the encryptor could choose the amount of collusion resilience it wants. Thus, this changes the syntax of the setup and encryption algorithm when compared to the static setting from above:

$\mathsf{Setup}(1^\lambda, 1^n) \to (\mathsf{mpk}, \mathsf{msk})$. The setup algorithm takes as input the security parameter λ and the functionality index n (in unary). It outputs the master public-secret key pair $(\mathsf{mpk}, \mathsf{msk})$.

$\mathsf{Enc}(\mathsf{mpk}, m \in \mathcal{M}_n, 1^q) \to \mathsf{ct}$. The encryption algorithm takes as input the master public key mpk, a message $m \in \mathcal{M}_n$, and it takes the desired collusion bound q as an input. It outputs a ciphertext ct.

Efficiency. The runtime of $\mathsf{Setup}, \mathsf{KeyGen}$ is polynomial in λ, n. While rest of the algorithms can run in time polynomial in λ, n, q.

Dynamic bounded collusion security. We define a 'simulation based' security notion similar to the static security definition (Definition 4.1).

Definition 4.2 (dynamic-bounded-collusion simulation-security)**.** A functional encryption scheme $\mathsf{FE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ is said to be *dynamically-bounded-collusion* simulation-secure if there exists a stateful PPT simulator Sim such that for every stateful PPT adversary $\mathcal{A}$, the following distributions are computationally indistinguishable:

$$\left\{ \mathcal{A}^{O(\cdot)}(\mathsf{ct}) : \begin{array}{c} 1^n \leftarrow \mathcal{A}(1^\lambda) \\ (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n) \\ (m, 1^q) \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk}, \cdot)}(\mathsf{mpk}) \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, m, 1^q) \\ O(\cdot) = \mathsf{KeyGen}(\mathsf{msk}, \cdot) \end{array} \right\}_{\lambda \in \mathbb{N}} \approx_c \left\{ \mathcal{A}^{O(\cdot)}(\mathsf{ct}) : \begin{array}{c} 1^n \leftarrow \mathcal{A}(1^\lambda) \\ \mathsf{mpk} \leftarrow \mathsf{Sim}(1^\lambda, 1^n) \\ (m, 1^q) \leftarrow \mathcal{A}^{\mathsf{Sim}(\cdot)}(\mathsf{mpk}) \\ \mathsf{ct} \leftarrow \mathsf{Sim}(\Pi^m, 1^q) \\ O(\cdot) = \mathsf{Sim}^{U_m(\cdot)}(\cdot) \end{array} \right\}_{\lambda \in \mathbb{N}}$$

whenever the admissibility constraints and properties, as defined in Definition 4.1, are satisfied.

4.3 Tagged Functional Encryption

Next, we recall the notion of tagged FE from [GGLW22]. Tagged FE intuitively represents a succinct collection of an exponential number of instances of a FE scheme, where each instance is denoted by a tag $\mathsf{tag} \in \mathcal{I}_z$ ($|\mathcal{I}_z|$ is the total number of FE instances bundled together). A tagged bounded functional encryption scheme FE for a family of function classes $\{\mathcal{F}_n\}_{n \in \mathbb{N}}$, message spaces $\{\mathcal{M}_n\}_{n \in \mathbb{N}}$ and tag spaces $\mathcal{I} = \{\mathcal{I}_z\}_{z \in \mathbb{N}}$ consists of four polynomial-time algorithms $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ with the following semantics.

$\mathsf{Setup}(1^\lambda, 1^n, 1^z, 1^q) \to (\mathsf{mpk}, \mathsf{msk})$. In addition to the normal inputs taken by a static-bounded FE scheme, the setup also takes in a tag space index z, which fixes a tag space $\mathcal{I}_z$.

$\mathsf{Enc}(\mathsf{mpk}, \mathsf{tag} \in \mathcal{I}_z, m \in \mathcal{M}_n) \to \mathsf{ct}$. The encryption also takes in a tag $\mathsf{tag} \in \mathcal{I}_z$ to bind to the ciphertext.

$\mathsf{KeyGen}(\mathsf{msk}, \mathsf{tag} \in \mathcal{I}_z, f \in \mathcal{F}_n) \to \mathsf{sk}_{\mathsf{tag},f}$. The key generation also binds the secret keys to a fixed tag $\mathsf{tag} \in \mathcal{I}_z$.

$\mathsf{Dec}(\mathsf{sk}_{\mathsf{tag},f}, \mathsf{ct}) \to \mathcal{R}_n$. The decryption algorithm has syntax identical to a non-tagged scheme.

Definition 4.3 (Correctness)**.** We say the scheme is correct if for all $\lambda, n \in \mathbb{N}, z, q \in \mathsf{poly}(\lambda)$, functions $f \in \mathcal{F}_n$, messages $m \in \mathcal{M}_n$ and $\mathsf{tag} \in \mathcal{I}_z$, we have that for $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, 1^z, 1^q)$, the following holds true,

$$\Pr\left[\mathsf{Dec}(\mathsf{KeyGen}(\mathsf{msk}, \mathsf{tag}, f), \mathsf{Enc}(\mathsf{mpk}, \mathsf{tag}, m)) = f(m)\right] = 1.$$

where the probability is taken over the coins of setup, key generation and encryption algorithms.

Definition 4.4 (tagged-static-bounded-collusion simulation-security)**.** For any choice of parameters $\lambda, n, q, z \in \mathbb{N}$, consider the following list of stateful oracles $\mathsf{S}_0, \mathsf{S}_1, \mathsf{S}_2$ where these oracles simulate the FE setup, key generation, and encryption algorithms respectively, and all three algorithms share and update the same global state of the simulator. Here the attacker interacts with the execution environment $\mathcal{E}$, and the environment makes queries to the simulator oracles. Formally, the simulator oracles and the environment are defined below:

$\mathsf{S}_0(1^\lambda, 1^n, 1^z, 1^q)$ generates the simulated master public key mpk of the system, and initializes the global state st of the simulator which is used by the next two oracles.

$\mathsf{S}_1(\cdot,\cdot,\cdot)$, upon a call to generate secret key on a function-tag-value tuple $(f_i, \mathsf{tag}_i, \mu_i)$, where the function value is either $\mu_i = \bot$ (signalling that the adversary has not yet made any encryption query on tag tag_i), or $(m^{\mathsf{tag}_i}, \mathsf{tag}_i)$ has already been queried for encryption (for some message m^{tag_i}), and $\mu_i = f_i(m^{\mathsf{tag}_i})$, the oracle outputs a simulated key $\mathsf{sk}_{f_i,\mathsf{tag}_i}$.

$\mathsf{S}_2(\cdot,\cdot)$, upon a call to generate ciphertext on a tag-list tuple $(\mathsf{tag}_i, \Pi^{m^{\mathsf{tag}_i}})$, where the list $\Pi^{m^{\mathsf{tag}_i}}$ is a possibly empty list of the form $\Pi^{m^{\mathsf{tag}_i}} = (f_1^{\mathsf{tag}_i}, f_1^{\mathsf{tag}_i}(m^{\mathsf{tag}_i})),$
$\ldots, (f_{q_{\mathsf{pre}}}^{\mathsf{tag}_i}, f_{q_{\mathsf{pre}}}^{\mathsf{tag}_i}(m^{\mathsf{tag}_i}))$ (that is, contains the list of function-value pairs for which the adversary has already received a secret key for), the oracle outputs a simulated ciphertext $\mathsf{ct}_{\mathsf{tag}_i}$.

$\mathcal{E}^{\mathsf{S}_1,\mathsf{S}_2}(\cdot,\cdot)$, receives two types of queries – secret key query and encryption query. Upon a secret key query on a function-tag pair (f_i, tag_i), if $(m^{\mathsf{tag}_i}, \mathsf{tag}_i)$ has already been queried for encryption (for some message m^{tag_i}) then $\mathcal{E}$ queries key oracle S_1 on tuple $(f_i, \mathsf{tag}_i, \mu_i = f_i(m^{\mathsf{tag}_i}))$, otherwise it adds (f_i, tag_i) to the its local state, and queries S_1 on tuple $(f_i, \mathsf{tag}_i, \mu_i = \bot)$. And, it simply forwards oracle's simulated key $\mathsf{sk}_{f_i,\mathsf{tag}_i}$ to the adversary.

Upon a ciphertext query on a message-tag pair (m_i, tag_i), if the adversary made an encryption query on the same tag tag_i previously, then the query is disallowed (that is, at most one message query per every unique tag is permitted). Otherwise, it computes a (possibly empty) list of function-value pairs of the form $\Pi^{m_i} = \left((f_1^{\mathsf{tag}_i}, f_1^{\mathsf{tag}_i}(m^{\mathsf{tag}_i})), \ldots, (f_{q_{\mathsf{pre}}}^{\mathsf{tag}_i}, f_{q_{\mathsf{pre}}}^{\mathsf{tag}_i}(m^{\mathsf{tag}_i}))\right)$ where $(f_j^{\mathsf{tag}_i}, \mathsf{tag}_i)$ are stored in $\mathcal{E}$'s local state, and removes all such pairs $(f_j^{\mathsf{tag}_i}, \mathsf{tag}_i)$ from its local state. $\mathcal{E}$ then queries ciphertext oracle S_2 on tuple $(\mathsf{tag}_i, \Pi^{m_i})$, and simply forwards oracle's simulated ciphertext $\mathsf{ct}_{\mathsf{tag}_i}$ to the adversary.

A tagged functional encryption scheme $\mathsf{FE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ is said to be tagged-statically-bounded-collusion simulation-secure if there exists a stateful PPT simulator $\mathsf{Sim} = (\mathsf{S}_0, \mathsf{S}_1, \mathsf{S}_2)$ such that for every stateful *admissible* PPT adversary $\mathcal{A}$, the following distributions are computationally indistinguishable:

$$\left\{ \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot),\mathsf{Enc}(\mathsf{mpk},\cdot,\cdot)}(\mathsf{mpk}) : \begin{array}{c} (1^n, 1^q, 1^z) \leftarrow \mathcal{A}(1^\lambda) \\ (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, 1^z, 1^q) \end{array} \right\}_{\lambda \in \mathbb{N}}$$
$$\approx_c$$
$$\left\{ \mathcal{A}^{\mathcal{E}^{\mathsf{S}_1,\mathsf{S}_2}(\cdot,\cdot)}(\mathsf{mpk}) : \begin{array}{c} (1^n, 1^q, 1^z) \leftarrow \mathcal{A}(1^\lambda) \\ \mathsf{mpk} \leftarrow \mathsf{S}_0(1^\lambda, 1^n, 1^z, 1^q) \end{array} \right\}_{\lambda \in \mathbb{N}}$$

where $\mathcal{A}$ is an admissible adversary if:

– $\mathcal{A}$ makes at most one encryption query per unique tag (that is, if the adversary made an encryption query on some tag tag_i previously, then making another encryption query for the same tag is disallowed)

– $\mathcal{A}$ makes at most q queries combined to the key generation oracles in the above experiments for all tags tag_i such that it also submitted an encryption query for tag tag_i.

Tagged FE to Dynamic Collusion Resistance. Garg et al. [GGLW22] proved that the notion of tagged FE captures the essence behind dynamic collusion resistance. Below we restate one of their main results.

Theorem 4.5 ([GGLW22, Paraphrased, Theorems 3.1 and 5.1])**.** If tgfe is a tagged statically λ-bounded collusion simulation-secure FE scheme (as per Definition 4.4), then it can be upgraded into a dynamic bounded collusion simulation-secure FE scheme (as per Definition 4.2) via a black-box transformation.

It turns out the same black-box transformation also works for multi-authority attribute-based encryption. We get the following theorem as a simple extension.

Theorem 4.6. If tgMAABE is a tagged statically λ-bounded collusion-secure multi-authority attribute-based encryption scheme, then it can be upgraded into a dynamic bounded collusion-secure multi-authority attribute-based encryption scheme via the GGLW black-box transformation.

5 From Static to Dynamic Collusion Model Generically

In this section, we show how to upgrade any statically bounded collusion FE scheme for function class $\mathcal{F}$ to dynamically bounded collusion FE scheme for the same function class $\mathcal{F}$. Our transformation is fully generic, but it is not black-box since we rely on the use of garbled circuits where we garble the encryption circuit for the underlying statically secure FE scheme.

High level sketch. We start with the simple observation from the previous section that to design a dynamically bounded collusion FE scheme for function class $\mathbb{F}$, it is sufficient to design a *tagged* FE scheme for function class $\mathcal{F}$. This is due to the black-box GGLW compiler that upgrades a tagged FE scheme to be dynamically secure. So, we design a generic compiler that builds a *tagged* FE scheme for function class $\mathcal{F}$ from any statically secure FE scheme for function class $\mathbb{F}$. Combining the generic compiler with the GGLW black-box compiler, we obtain a dynamically secure FE scheme for function class $\mathcal{F}$.

A bit more concretely, we design a tagged FE accumulator which accumulates 2^z many instances of a statically secure FE scheme to a tagged FE scheme for tag space $\mathcal{I}_z = \{0,1\}^z$. Our compiler preserves the bound on the number of key generation queries that can be made on a particular tag. That is, if we start with a q-bounded static FE scheme, we obtain a q-bounded tagged FE scheme. Our accumulator is insipred from the recent success in usage of garbling techniques [DG17a] for designing identity-based encryption, registration-based encryption and more [DG17b, DG17a, BLSV18, GHMR18, GHM$^+$19, GV20].

In the aforementioned list of works, a central component of the design was to combine garbled circuits with a nearly equivalent set of core primitive such as chameleon hash functions, one time signatures with encryption (OTSE), batch encryption, etc. A key idea was to use this combination to conceptually enable delegation of encryption keys to the leaves of a Merkle tree. Our starting point for the tagged FE accumulator is to use a similar delegation trick to delegate an exponential number of FE keys to a unique leaf of the Merkle tree. We build our tagged FE by relying on compact one-time signature with encryption (OTSE), which were initially introduced in [DG17a] to build IBE.

Below we provide our construction for tagged FE accumulator. We remark that our accumulator supports accumulating static FE for uniform models of computation as well.

5.1 Tagged FE Accumulator

Ingredients. Let $\mathsf{BFE} = (\mathsf{BFE.Setup}, \mathsf{BFE.Enc}, \mathsf{BFE.KeyGen}, \mathsf{BFE.Dec})$ be a q-bounded FE scheme for function space $\mathcal{F}$ and message space $\mathcal{M}$, and let $\mathsf{OTSE} = (\mathsf{SSetup}, \mathsf{SGen}, \mathsf{SSign}, \mathsf{SEnc}, \mathsf{SDec})$ be an OTSE scheme. Let PRF be a pseudorandom function with key size λ, inputs to be any bit string of length $\leq z$ bits, and outputs 2λ bits of output. We use PRF_1 and PRF_2 to denote the first λ and last λ bits of the output, respectively.[10] That is, $\mathsf{PRF}(s \in \{0,1\}^\lambda, v \in \{0,1\}^{\leq z}) = \mathsf{PRF}_1(s,v)||\mathsf{PRF}_2(s,v)$.

Below we provide our tagged FE scheme for function space $\mathcal{F}$ and message space $\mathcal{M}$ with tag space $\mathcal{I}_z = \{0,1\}^z$. We remark that $\mathcal{F}$ could be specifying any uniform/non-uniform model of computation.

$\mathsf{Setup}(1^\lambda, 1^n, 1^z, 1^q) \to (\mathsf{mpk}, \mathsf{msk})$. It samples a PRF seed $s \leftarrow \{0,1\}^\lambda$. Let ℓ be the length of $\mathsf{BFE.mpk}$ corresponding to parameters λ, n, q. It samples parameters $\mathsf{pp} \leftarrow \mathsf{SSetup}(1^\lambda, \ell)$.[11] Compute the root parameters as $(\mathsf{vk}_\epsilon, \sigma_\epsilon, x_\epsilon) \leftarrow \mathsf{NodeGen}(\mathsf{pp}, \epsilon, s)$ (routine NodeGen described in Fig. 1).

It outputs the master key pair as $\mathsf{mpk} = (\mathsf{pp}, \mathsf{vk}_\epsilon), \mathsf{msk} = s$.

```
                    NodeGen(pp, v, s)

Input: OTSE parameter pp, node index $v \in \{0,1\}^{z' \leq z}$, seed $s \in \{0,1\}^\lambda$
Output: OTSE verification key $\mathsf{vk}_v$, signature $\sigma_v$, auxiliary value $x_v$

  1. Let $(\mathsf{vk}_w, \mathsf{sk}_w) \leftarrow \mathsf{SGen}(\mathsf{pp}; \mathsf{PRF}_1(s,w))$ for $w \in \{v, v||0, v||1\}$.
  2. Set $x_v = \mathsf{vk}_{v||0}||vk_{v||1}$ and $\sigma_v = \mathsf{SSign}(\mathsf{pp}, \mathsf{sk}_v, x_v)$.
  3. Output $(\mathsf{vk}_v, \sigma_v, x_v)$.
```
Figure 1: Routine NodeGen

$\mathsf{KeyGen}(\mathsf{msk} = s, \mathsf{tag} \in \{0,1\}^z, f \in \mathcal{F}_n) \to \mathsf{sk}_{f,\mathsf{tag}}$. Let v_j denote the j-bit prefix of tag, i.e. $v_j = \mathsf{tag}[1:j]$ for $j \in [0,z]$. Note that $v_0 = \epsilon$ and $v_z = \mathsf{tag}$.

The key generator first computes $(\mathsf{vk}_{v_j}, \sigma_{v_j}, x_{v_j}) \leftarrow \mathsf{NodeGen}(\mathsf{pp}, v_j, s)$ for $j \in [0, z-1]$. It computes $(\mathsf{vk}_{\mathsf{tag}}, \sigma_{\mathsf{tag}}, \mathsf{BFE.mpk}_{\mathsf{tag}}, \mathsf{BFE.sk}_{\mathsf{tag},f}) \leftarrow \mathsf{LeafGen}(\mathsf{pp}, \mathsf{tag}, s, f)$ (routine LeafGen described in Fig. 2).

It outputs $\mathsf{sk}_{f,\mathsf{tag}} = \left(\{(\sigma_{v_j}, x_{v_j})\}_{j \in [0,z-1]}, \sigma_{\mathsf{tag}}, \mathsf{BFE.mpk}_{\mathsf{tag}}, \mathsf{BFE.sk}_{\mathsf{tag},f}\right)$.

[10] We assume, w.l.o.g., that the setup algorithms for BFE and OTSE take λ-bits as input.

[11] Recall from the succinctness property of OTSE, the length of the verification key $|\mathsf{vk}|$ is some polynomial in λ and independent of ℓ.

LeafGen(pp, v, s, f)

Input: OTSE parameters pp, leaf index $v \in \{0,1\}^z$, seed $s \in \{0,1\}^\lambda$, function $f \in \mathcal{F}_n$
Output: OTSE verification key vk_v, signature σ_v, public key of v^{th} instance $\mathsf{BFE.mpk}_v$, secret key $\mathsf{BFE.sk}_{v,f}$

1. Let $(\mathsf{vk}_v, \mathsf{sk}_v) \leftarrow \mathsf{SGen}(\mathsf{pp}; \mathsf{PRF}_1(s, v))$.
2. Compute $(\mathsf{BFE.mpk}_v, \mathsf{BFE.msk}_v) \leftarrow \mathsf{BFE.Setup}(1^\lambda; \mathsf{PRF}_2(s, v))$ (i.e., v^{th} instance of base FE scheme with fixed randomness $\mathsf{PRF}_2(s, v)$).
3. Sample $\mathsf{BFE.sk}_{v,f} \leftarrow \mathsf{BFE.KeyGen}(\mathsf{BFE.msk}_v, f)$, and compute $\sigma_v = \mathsf{SSign}(\mathsf{pp}, \mathsf{sk}_v, \mathsf{BFE.mpk}_v)$.
4. Output $(\mathsf{vk}_v, \sigma_v, \mathsf{BFE.mpk}_v, \mathsf{BFE.sk}_{v,f})$.

Figure 2: Routine LeafGen

$\mathsf{Enc}(\mathsf{mpk}, \mathsf{tag}, m) \rightarrow \mathsf{ct}$. It parses mpk as above. Let v_j denote the j-bit prefix of tag for $j \in [z]$, and $\ell = |\mathsf{BFE.mpk}|$. Let $\mathsf{T}[m, r]$ be a circuit as described in Fig. 3.

The encryptor first garbles $\mathsf{T}[m, r]$ as $(\tilde{\mathsf{T}}, \mathsf{e_T}) \leftarrow \mathsf{GC.Garble}(1^\lambda, \mathsf{T}[m, r])$ for uniform randomness r, and next it garbles $\mathsf{Q}[\mathsf{pp}, 0, \ell, \mathsf{e_T}, \mathsf{r_T}]$ for uniform randomness $\mathsf{r_T}$ as follows (see Fig. 4 for details) – $(\tilde{\mathsf{Q}}^{(z)}, \mathsf{e}_\mathsf{Q}^{(z)}) \leftarrow \mathsf{GC.Garble}(1^\lambda, \mathsf{Q}[\mathsf{pp}, 0, \ell, \mathsf{e_T}, \mathsf{r_T}])$.

Next, for $j = z-1, \ldots, 0$, it garbles a sequence of circuits as

$$(\tilde{\mathsf{Q}}^{(j)}, \mathsf{e}_\mathsf{Q}^{(j)}) \leftarrow \mathsf{GC.Garble}(1^\lambda, \mathsf{Q}[\mathsf{pp}, \mathsf{tag}_{j+1}, \ell', \mathsf{e}_\mathsf{Q}^{(j+1)}, \mathsf{r}_\mathsf{Q}^{(j+1)}]),$$

where ℓ' be $|\mathsf{vk}_\epsilon|$, $\mathsf{r}_\mathsf{Q}^{(j+1)}$ is uniform randomness. Finally, it parses $\mathsf{e}_\mathsf{Q}^{(0)} = \{Y_{\iota,0}, Y_{\iota,1}\}_{\iota \in [\ell']}$, and outputs $\mathsf{ct} = \left(\tilde{\mathsf{y}}^{(0)}, \tilde{\mathsf{Q}}^{(0)}, \ldots, \tilde{\mathsf{Q}}^{(z)}, \tilde{\mathsf{T}}\right)$, where $\tilde{\mathsf{y}}^{(0)} = \{Y_{\iota, \mathsf{y}_\iota}\}_{\iota \in [\ell']}$ and y_ι denote the ι^{th} bit of vk_ϵ.

Leaf Encryption Circuit $\mathsf{T}[m, r](\mathsf{BFE.mpk})$

Input: Leaf public key BFE.mpk, **Output:** Ciphertext ct.
Constant: message m, randomness r.

1. Compute and output $\mathsf{ct} = \mathsf{BFE.Enc}(\mathsf{BFE.mpk}, m; r)$.

Figure 3: Circuit T

Internal Encryption Circuit $\mathsf{Q}[\mathsf{pp}, \beta, \ell', \mathsf{e}, \mathsf{r}](\mathsf{vk})$

Input: OTSE verification key vk, **Output:** Encrypted labels $\hat{\mathsf{e}}^\beta$.
Constants: OTSE parameters pp, bit β, no of labels ℓ', wire labels $\mathsf{e} = \{(Y_{\iota,0}, Y_{\iota,1})\}_{\iota \in [\ell']}$, randomness $\mathsf{r} = \{(r_{\iota,0}, r_{\iota,1})\}_{\iota \in [\ell']}$.

1. Output $\{\mathsf{SEnc}(\mathsf{pp}, (\mathsf{vk}, \beta \cdot \ell' + \iota, b), Y_{\iota,b}; r_{\iota,b})\}_{\iota \in [\ell'], b \in \{0,1\}}$.

Figure 4: Circuit Q

$\mathsf{Dec}(\mathsf{sk}_{\mathsf{tag},f}, \mathsf{ct})$. It parses the key and ciphertext as above. Recall that we use v_j to denote the j-bit prefix of tag, $\ell = |\mathsf{BFE.mpk}|$ and ℓ' as length of OTSE verification key. It simply runs the following iterative procedure for decryption where it first runs the garbled circuit to recover encrypted wire labels, and then decrypt half of the wire labels, and then continue this until it recovers an FE ciphertext under the public key associated for the leaf node corresponding to tag. Concretely, it does as follows:

1. For $j = 0$ to $z - 1$:
   (a) $\{\hat{\mathsf{e}}^{(j)}_{\iota,b}\}_{\iota\in[\ell'],b\in\{0,1\}} \leftarrow \mathsf{GC.Eval}(\tilde{\mathsf{Q}}^{(j)}, \tilde{\mathsf{y}}^{(j)})$.
   (b) $\tilde{\mathsf{y}}^{(j+1)} \leftarrow \left\{\mathsf{SDec}(\mathsf{pp}, (\mathsf{vk}_{v_j}, \sigma_{v_j}, x_{v_j}), \hat{\mathsf{e}}^{(j)}_{\iota,(x_{v_j})_\iota})\right\}_{\iota\in[\ell']}$.
2. Evaluate $\{\hat{\mathsf{e}}^{(z)}_{\iota,b}\}_{\iota\in[\ell],b\in\{0,1\}} \leftarrow \mathsf{GC.Eval}(\tilde{\mathsf{Q}}^{(z)}, \tilde{\mathsf{y}}^{(z)})$, and let pk_ι denote the ι^{th} bit of $\mathsf{BFE.mpk}_{\mathsf{tag}}$.
3. Decrypt $\mathsf{y}^{\mathsf{T}} = \left\{\mathsf{SDec}\left(\mathsf{pp}, \left(\mathsf{vk}_{\mathsf{tag}}, \sigma_{\mathsf{tag}}, \mathsf{BFE.mpk}_{\mathsf{tag}}\right), \hat{\mathsf{e}}^{(z)}_{\iota,\mathsf{pk}_\iota}\right)\right\}_{\iota\in[\ell]}$.
4. Finally, output $\mathsf{BFE.Dec}(\mathsf{BFE.sk}_{\mathsf{tag},f}, \mathsf{ct}_{\mathsf{BFE}})$ where $\mathsf{ct}_{\mathsf{BFE}} = \mathsf{Eval}(\tilde{\mathsf{T}}, \mathsf{y}^{\mathsf{T}})$.

Correctness and efficiency. We say the scheme is correct if it satisfies Definition 4.3. By correctness of the garbling scheme, we have that $\tilde{\mathsf{Q}}^{(0)}$, when run on garbled input $\tilde{\mathsf{y}}^{(0)}$ computes $Q[\mathsf{pp}, 0, \ell, \cdot](\mathsf{vk}_\epsilon)$ and outputs encrypted labels to $\tilde{\mathsf{Q}}^{(1)}$. By correctness of the OTSE encryption scheme, we can decrypt the labels corresponding to tag_1 to compute $\tilde{y}^{(1)}$. Similarly, iteratively calling the correctness of garbling and the signature scheme, we compute the garbled inputs to circuit $\tilde{\mathsf{T}}$ corresponding to $\mathsf{BFE.mpk}_{\mathsf{tag}}$. Thus, eventually it computes $\mathsf{ct}_{\mathsf{BFE}} \leftarrow \mathsf{T}[m](\mathsf{BFE.mpk}_{\mathsf{tag}})$ by correctness of garbling, and by the correctness of the tag^{th} instance of BFE scheme, we get $\mathsf{BFE.Dec}(\mathsf{BFE.sk}_{\mathsf{tag},f}, \mathsf{BFE.Enc}(\mathsf{BFE.mpk}_{\mathsf{tag}}, m)) = f(m)$.

The different algorithms Setup, KeyGen run polynomial in λ, z, n, q and this can be seen easily from the construction. Encryption algorihm runs polynomial in λ, z, n, q and the message m that is chosen during encryption time. Crucially, we observe here that our tagged FE accumulator is agnostic to the model of computation of the base BFE scheme. Thus if the base scheme BFE can support uniform models of computation, so can our transformation.

5.2 Security

Theorem 5.1. If PRF is a secure pseudorandom function, GC is a secure garbling scheme, OTSE is a secure one-time signature with encryption scheme, BFE = (BFE.Setup, BFE.Enc, BFE.KeyGen, BFE.Dec) is a bounded-collusion simulation-secure FE scheme (as per Definition 4.1), then the above scheme is a tagged-statically-bounded-collusion simulation-secure FE scheme (as per Definition 4.4).

Proof. The proof strategy is inspired from the proof of adaptive security of identity-based encryption from [DG17a]. Let us start by discussing the similarities that we can use in our proof of security. The initial idea used in the IBE setting was to successively simulate garbled circuits one-by-one inside the challenge ciphertext using a sequence of hybrid proof steps, where first they switch the use of an actual PRF with a truly random function by making the challenger store state. And, next they simulate the garbled circuits since once half of the wire keys are ever decryptable for any garbled circuit in the sequence, and this is guaranteed by the security of the one-time signature with encryption scheme since the signatures created binds to the entire sequence of tree nodes which can be opened on any path. This strategy enables simulation of every garbled circuit until the challenger wants to simulate the last garbled circuit $T[m,r]$ which contains the actual message. Using the same strategy as above, this garbled circuit can also be simulated using just the encryption of message m under randomness r.

At this point, our proof and the [DG17a] proof diverge since in the case of IBE, the attacker never receives the secret key correspoding to the final ciphertext, thus security follows from security of public-key encryption. However, in our case an attacker can get a bounded number of function keys that enable decryption for the corresponding FE ciphertext. But this is only a minor technical issue and we can get around this using simulation security of the base FE scheme. Thus, by using simulation security of base (untagged) FE, we can simulate this ciphertext and simulate the garbled circuit using the simulated FE ciphertext instead. Below we provide the full proof.

Let $\mathsf{BFE.Sim} = (\mathsf{BFE.S}_0, \mathsf{BFE.S}_1, \mathsf{BFE.S}_2)$ be the simulators satisfying Definition 4.1. We will construct simulators $\mathsf{Sim} = (\mathsf{S}_0, \mathsf{S}_1, \mathsf{S}_2, \mathsf{S}_3)$ that share a global state st and use the simulated routines Figure 5 and Figure 6 that satisfy Definition 4.4 as follows. We point out that for ease of exposition, we drop the uniform randomness from the garbled circuits whenever it is clear from context.

$\mathsf{Sim.LeafGen}(\mathsf{pp}, v, \mathsf{st}, f, \mu)$

Inputs: OTSE public parameters pp; Leaf Index $v \in \{0,1\}^z$; Simulator State $\mathsf{st} \in \{0,1\}^*$; Function $f \in \mathcal{F}_n$; Function Values μ

Output: OTSE verification key vk_v; OTSE signature σ_v; Public Key of v^{th} instance $\mathsf{BFE.mpk}_v$; Secret Key $\mathsf{BFE.sk}_{v,f}$

1. If $(v, \mathsf{vk}_v, \mathsf{sk}_v)$ does not exist in st, then $(\mathsf{vk}_v, \mathsf{sk}_v) \leftarrow \mathsf{SGen}(\mathsf{pp})$. Add $(v, \mathsf{vk}_v, \mathsf{sk}_v)$ to st.
2. Let $(\mathsf{BFE.mpk}_v, \mathsf{BFE}^v.\mathsf{st}_0) \leftarrow \mathsf{BFE.S}_0(1^\lambda, 1^n, q)$.
3. If $\mu = \perp$, run $\mathsf{BFE.sk}_{v,f} \leftarrow \mathsf{BFE.S}_1(\mathsf{BFE}^v.\mathsf{st}_0, f)$ and update state $\mathsf{BFE}^v.\mathsf{st}_1$ in st. Else, get $\mathsf{BFE.st}_2^v$ from st and run $\mathsf{BFE.sk}_{v,f} \leftarrow \mathsf{BFE.S}_3^{\mu}(\mathsf{BFE}^v.\mathsf{st}_2, f)$ where μ contains the list of all previous function evaluations.
4. $\sigma_v = \mathsf{SSign}(\mathsf{pp}, \mathsf{sk}_v, \mathsf{BFE.mpk}_v)$.
5. Output $(\mathsf{vk}_v, \sigma_v, \mathsf{BFE.mpk}_v, \mathsf{BFE.sk}_{v,f})$.

Figure 5: Routine Simulated LeafGen

$\mathsf{S}_0(1^\lambda, 1^n, 1^z, 1^q)$

Sample $\mathsf{pp} \leftarrow \mathsf{SSetup}(1^\lambda, \ell)$. Run $(\mathsf{vk}_\epsilon, \mathsf{sk}_\epsilon, x_\epsilon) \leftarrow \mathsf{Sim.NodeGen}(\mathsf{pp}, \epsilon, \mathsf{st})$. Output $\mathsf{mpk} = (\mathsf{pp}, \mathsf{vk}_\epsilon)$.

$\mathsf{S}_1(\mathsf{tag}, f, \mu)$

$\mathsf{Sim.NodeGen}(\mathsf{pp}, v, \mathsf{st})$

Inputs: OTSE public parameters pp; Node Index $v \in \{0,1\}^{z'}$ where $z' < z$;
Simulator state $\mathsf{st} \in \{0,1\}^*$

Output: OTSE verification key vk_v; OTSE signature σ_v; Auxiliary value x_v

1. If $(v, \mathsf{vk}_v, \mathsf{sk}_v)$ does not exist in st, then $(\mathsf{vk}_v, \mathsf{sk}_v) \leftarrow \mathsf{SGen}(\mathsf{pp})$. Add $(v, \mathsf{vk}_v, \mathsf{sk}_v)$ to st. Similarly, set/check the parameters in st for $v||0$ and $v||1$.
2. Let $x_v = \mathsf{vk}_{v||0}||vk_{v||1}$.
3. $\sigma_v = \mathsf{SSign}(\mathsf{pp}, \mathsf{sk}_v, x_v)$.
4. Output $(\mathsf{vk}_v, \sigma_v, x_v)$.

Figure 6: Routine Simulated NodeGen

Simulated Garbled Label Encryption Circuit $\mathsf{Sim.Q}[\mathsf{pp}, \beta, \ell', \mathsf{y}](\mathsf{vk})$

Inputs: OTSE verification key vk

Constants: OTSE public parameters pp; Bit $\beta \in \{0,1\}$; Number of circuit labels $\ell' \in \mathbb{N}$; Labels of a garbled circuit $\mathsf{y} = \{(Y_\iota)\}_{\iota \in [\ell']}$

Output: Encrypted Labels, $\hat{\mathsf{e}}^\beta$

1. Compute and output $\{\mathsf{SEnc}(\mathsf{pp}, (\mathsf{vk}, \beta \cdot \ell' + \iota, b), Y_\iota)\}_{\iota \in [\ell'], b \in \{0,1\}}$.

Figure 7: Simulation Circuit Sim.Q

1. For $j \in [0, z]$, let v_j denote the prefix of tag, i.e. first j bits of tag. Note that v_0 is ϵ and $v_z = \mathsf{tag}$.
2. For $j \in [0, z-1]$, compute $(\mathsf{vk}_{v_j}, \sigma_{v_j}, x_{v_j}) \leftarrow \mathsf{Sim.NodeGen}(\mathsf{pp}, v_j, \mathsf{st})$.
3. Let $(\mathsf{vk}_{\mathsf{tag}}, \sigma_{\mathsf{tag}}, \mathsf{BFE.mpk}_{\mathsf{tag}}, \mathsf{BFE.sk}_{\mathsf{tag},f}) \leftarrow \mathsf{Sim.LeafGen}(\mathsf{pp}, \mathsf{tag}, \mathsf{st}, f, \mu)$.
4. Output $\left(\{(\sigma_{v_j}, x_{v_j})\}_{j \in [0,z-1]}, \sigma_{\mathsf{tag}}, \mathsf{BFE.mpk}_{\mathsf{tag}}, \mathsf{BFE.sk}_{\mathsf{tag},f}\right)$.

$\mathsf{S}_2(\mathsf{tag}^*, \Pi^{m^{\mathsf{tag}^*}})$

1. For $j \in [z]$, let v_j denote the prefix of tag and ℓ be the length of BFE.mpk.
2. If $\mathsf{BFE}^{\mathsf{tag}^*}.\mathsf{st}_1$ is not in st, setup $(\mathsf{BFE.mpk}_{\mathsf{tag}^*}, \mathsf{BFE}^{\mathsf{tag}^*}.\mathsf{st}_0) \leftarrow \mathsf{BFE.S}_0(1^\lambda, 1^n, q)$. Let $\mathsf{BFE}^{\mathsf{tag}^*}.\mathsf{st}_1 = \mathsf{BFE}^{\mathsf{tag}^*}.\mathsf{st}_0$. Run $\mathsf{ct}^*_{\mathsf{BFE}} \leftarrow \mathsf{BFE.S}_2(\mathsf{BFE}^{\mathsf{tag}^*}.\mathsf{st}_1, \Pi^{m^{\mathsf{tag}^*}})$. Store $\mathsf{BFE}^{\mathsf{tag}^*}.\mathsf{st}_2$ in st. ($\Pi^{m^{\mathsf{tag}^*}}$ contains the functions and their evaluations at m^{tag^*} for the tag tag*).
3. $(\tilde{\mathsf{T}}, \tilde{\mathsf{y}}^{\mathsf{T}}) \leftarrow \mathsf{GC.Sim}(1^\lambda, 1^\ell, 1^{|\mathsf{T}[0]|}, \mathsf{ct}^*_{\mathsf{BFE}})$ (Figure 3)[12].

[12] Any hardcoded public message $\in \mathcal{M}_n$ will work as input to T. Here we set it to 0 for simplicity. In future simulation algorithms, for notational simplicity we omit the hardcoded values and just mention the size as $1^{|\mathsf{T}|}$ or $1^{|\mathsf{Q}|}$. Assume that the size can be set appropriately as a polynomial in λ, ℓ, ℓ'.

4. For $j \in [0, z]$, let v_j denote the prefix, i.e. first j bits of tag. If $(v_j, \mathsf{vk}_{v_j}, \mathsf{sk}_{v_j})$ does not exist in st, then $(\mathsf{vk}_{v_j}, \mathsf{sk}_{v_j}) \leftarrow \mathsf{SGen}(\mathsf{pp})$. Add $(v_j, \mathsf{vk}_{v_j}, \mathsf{sk}_{v_j})$ to st.
5. Compute $\hat{\mathsf{y}}^{\mathsf{T}} \leftarrow \mathsf{Sim.Q}[\mathsf{pp}, 0, \ell, \tilde{\mathsf{y}}^{\mathsf{T}}](\mathsf{vk}_{\mathsf{tag}^*})$.
6. Let $(\tilde{\mathsf{Q}}^{(z)}, \tilde{\mathsf{y}}_{\mathsf{Q}}^{(z)}) \leftarrow \mathsf{GC.Sim}(1^\lambda, 1^\ell, 1^{|\mathsf{Q}|}, \hat{\mathsf{y}}^{\mathsf{T}})$ (Figure 4).
7. For $j = z-1, \ldots, 0$, let ℓ' be $|\mathsf{vk}_\epsilon|$,
   (a) Compute $\hat{\mathsf{y}}_{\mathsf{Q}}^{(j)} \leftarrow \mathsf{Sim.Q}[\mathsf{pp}, \mathsf{tag}_{j+1}, \ell', \tilde{\mathsf{y}}_{\mathsf{Q}}^{(j+1)}](\mathsf{vk}_{v_j})$.
   (b) Let $(\tilde{\mathsf{Q}}^{(j)}, \tilde{\mathsf{y}}_{\mathsf{Q}}^{(j)}) \leftarrow \mathsf{GC.Sim}(1^\lambda, 1^{\ell'}, 1^{|\mathsf{Q}|}, \hat{\mathsf{y}}_{\mathsf{Q}}^{(j)})$ (Figure 4).
8. Output $\left(\tilde{\mathsf{y}}^{(0)}, \tilde{\mathsf{Q}}^{(0)}, \ldots, \tilde{\mathsf{Q}}^{(z)}, \tilde{\mathsf{T}}\right)$.

We will show through a sequence of experiments that the real and simulated games of Theorem 5.1 are computationally indistinguishable. We only note down the steps that are different from the previous experiment in red, rest of the operations and notations remain the same.

Experiment 0. This is the experiment with adversary $\mathcal{A}$ and tagged-statically-bounded-collusion FE scheme. The experiment is parameterized by $\lambda \in \mathbb{N}$.

$$\left\{ \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot),\mathsf{Enc}(\mathsf{mpk},\cdot,\cdot)}(\mathsf{mpk}) : \begin{array}{c} (1^n, 1^q, 1^z) \leftarrow \mathcal{A}(1^\lambda) \\ (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, 1^z, 1^q) \end{array} \right\}_{\lambda \in \mathbb{N}}$$

- **Setup:** $(1^n, 1^q, 1^z) \leftarrow \mathcal{A}(1^\lambda)$.
  1. Let $\mathsf{pp} \leftarrow \mathsf{SSetup}(1^\lambda, \ell)$ (ℓ is length of $\mathsf{BFE.mpk}$ called on securtiy parameter λ, functionality index n and collusion bound q).
  2. Sample $s \leftarrow \{0,1\}^\lambda$ as PRF seed.
  3. Compute $(\mathsf{vk}_\epsilon, \sigma_\epsilon, x_\epsilon) \leftarrow \mathsf{NodeGen}(\mathsf{pp}, \epsilon, s)$.
  4. Set $\mathsf{mpk} = (\mathsf{pp}, \mathsf{vk}_\epsilon)$, $\mathsf{msk} = s$.

  Send mpk to $\mathcal{A}$. $\mathcal{A}$ is given access to two oracles, KeyGen and Enc.
- **Key Queries:** Let $\mathcal{A}$ query KeyGen on tag $\mathsf{tag} \in \{0,1\}^z$ and function f.
  1. For $j \in [0, z]$, let v_j denote the prefix of tag, i.e. first j bits of tag. Note that v_0 is ϵ and $v_z = \mathsf{tag}$.
  2. For $j \in [0, z-1]$, compute $(\mathsf{vk}_{v_j}, \sigma_{v_j}, x_{v_j}) \leftarrow \mathsf{NodeGen}(\mathsf{pp}, v_j, s)$.
  3. Let $(\mathsf{vk}_{\mathsf{tag}}, \sigma_{\mathsf{tag}}, \mathsf{BFE.mpk}_{\mathsf{tag}}, \mathsf{BFE.sk}_{\mathsf{tag},f}) \leftarrow \mathsf{LeafGen}(\mathsf{pp}, \mathsf{tag}, s)$.
  4. Output $\left(\{(\sigma_{v_j}, x_{v_j})\}_{j \in [0,z-1]}, \sigma_{\mathsf{tag}}, \mathsf{BFE.mpk}_{\mathsf{tag}}, \mathsf{BFE.sk}_{\mathsf{tag},f}\right)$.
- **Ciphertext Queries:** Let $\mathcal{A}$ query Enc on tag $\mathsf{tag}^* \in \mathcal{I}_z$ and message $m^{\mathsf{tag}^*} \in \mathcal{M}_n$.
  1. $(\tilde{\mathsf{T}}, \mathsf{e}_{\mathsf{T}}) \leftarrow \mathsf{GC.Garble}(1^\lambda, \mathsf{T}[m])$ (Figure 3).
  2. Let $(\tilde{\mathsf{Q}}^{(z)}, \mathsf{e}_{\mathsf{Q}}^{(z)}) \leftarrow \mathsf{GC.Garble}(1^\lambda, \mathsf{Q}[\mathsf{pp}, 0, \ell, \mathsf{e}_{\mathsf{T}}])$ (Figure 4).

3. For $j = z-1, \dots, 0$, let ℓ' be $|\mathsf{vk}_\epsilon|$,
   (a) $(\tilde{\mathsf{Q}}^{(j)}, \mathsf{e}_{\mathsf{Q}}^{(j)}) \leftarrow \mathsf{GC.Garble}(1^\lambda, \mathsf{Q}[\mathsf{pp}, \mathsf{tag}^*_{j+1}, \ell', \mathsf{e}_{\mathsf{Q}}^{(j+1)}])$ (note that $\mathsf{Q}^{(j+1)}$ consists of labels corresponding to the verification key).
4. Parse $\mathsf{e}_{\mathsf{Q}}^{(0)} = \{Y_{\iota,0}, Y_{\iota,1}\}_{\iota\in[\ell']}$.
5. Let y_ι denote the ι^{th} bit of vk_ϵ. Let $\tilde{\mathsf{y}}^{(0)} \leftarrow \{Y_{\iota,\mathsf{y}_\iota}\}_{\iota\in[\ell']}$.
6. Output $\left(\tilde{\mathsf{y}}^{(0)}, \tilde{\mathsf{Q}}^{(0)}, \dots, \tilde{\mathsf{Q}}^{(z)}, \tilde{\mathsf{T}}\right)$.

- $\mathcal{A}$ outputs a bit b.

We've included additional notational indexing to help with our hybrids.

Experiment 1. In this experiment, we replace the PRF function with a truly random function. We additionally describe a routine that partially simulates LeafGen.

```
PartSim.LeafGen(pp, $v$, st, $f$)

Inputs: OTSE public parameters pp; Leaf Index $v \in \{0,1\}^z$; Simulator State st $\in \{0,1\}^*$;
        Function $f \in \mathcal{F}_n$
Output: OTSE verification key $\mathsf{vk}_v$; OTSE signature $\sigma_v$; Public Key of $v^{th}$ instance
        $\mathsf{BFE.mpk}_v$; Secret Key $\mathsf{BFE.sk}_f$

1. If $(v, \mathsf{vk}_v, \mathsf{sk}_v)$ does not exist in st, then $(\mathsf{vk}_v, \mathsf{sk}_v) \leftarrow \mathsf{SGen}(\mathsf{pp})$. Add $(v, \mathsf{vk}_v, \mathsf{sk}_v)$ to st.
2. Let $(\mathsf{BFE.mpk}_v, \mathsf{BFE.msk}_v) \leftarrow \mathsf{BFE.Setup}(1^\lambda, 1^n, q)$. Store $(v, \mathsf{BFE.mpk}_v, \mathsf{BFE.msk}_v)$ in the
   state st.
3. $\mathsf{BFE.sk}_f \leftarrow \mathsf{BFE.KeyGen}(\mathsf{BFE.msk}_v, f)$
4. $\sigma_v = \mathsf{SSign}(\mathsf{pp}, \mathsf{sk}_v, \mathsf{BFE.mpk}_v)$
5. Output $(\mathsf{vk}_v, \sigma_v, \mathsf{BFE.mpk}_v, \mathsf{BFE.sk}_f)$
```

Figure 8: Routine Partially Simulated LeafGen

- **Setup:** $(1^n, 1^q, 1^z) \leftarrow \mathcal{A}(1^\lambda)$.
  ~~Sample $s \leftarrow \{0,1\}^\lambda$ as PRF seed.~~
  Let st be the global state. Compute $(\mathsf{vk}_\epsilon, \sigma_\epsilon, x_\epsilon) \leftarrow \mathsf{Sim.NodeGen}(\mathsf{pp}, \epsilon, \mathsf{st})$.

- **Key Queries:** Let $\mathcal{A}$ query KeyGen on tag tag $\in \{0,1\}^z$ and function f.
  For $j \in [0, z-1]$, compute $(\mathsf{vk}_{v_j}, \sigma_{v_j}, x_{v_j}) \leftarrow \mathsf{Sim.NodeGen}(\mathsf{pp}, v_j, \mathsf{st})$.
  Let $(\mathsf{vk}_{\mathsf{tag}}, \sigma_{\mathsf{tag}}, \mathsf{BFE.mpk}_{\mathsf{tag}}, \mathsf{BFE.sk}_{\mathsf{tag},f}) \leftarrow \mathsf{PartSim.LeafGen}(\mathsf{pp}, \mathsf{tag}, \mathsf{st})$.

Experiment $2k$**.** Here k varies from 1 to z. In this experiment, we change how $\tilde{\mathsf{Q}}^{(k-1)}$ and the garbled labels $\tilde{\mathsf{y}}^{(k-1)}$ are computed by calling the garbled circuit simulator.
Ciphertext Queries: Let $\mathcal{A}$ query Enc on tag $\mathsf{tag}^* \in \mathcal{I}_z$ and message $m^{\mathsf{tag}^*} \in \mathcal{M}_n$.

- For $j = z-1, \ldots, k$, let ℓ' be $|\mathsf{vk}_\epsilon|$,
  1. $(\tilde{\mathsf{Q}}^{(j)}, \mathsf{e}_{\mathsf{Q}}^{(j)}) \leftarrow \mathsf{GC.Garble}(1^\lambda, \mathsf{Q}[\mathsf{pp}, \mathsf{tag}^*_{j+1}, \ell', \mathsf{e}_{\mathsf{Q}}^{(j+1)}])$ (note that $\mathsf{Q}^{(j+1)}$ consists of labels corresponding to the verification key).
- For $j = (k-1)$,
  1. Compute $\hat{\mathsf{e}}_{\mathsf{Q}}^{(j)} \leftarrow \mathsf{Q}[\mathsf{pp}, \mathsf{tag}_{j+1}, \ell', \mathsf{e}_{\mathsf{Q}}^{(j+1)}](\mathsf{vk}_{v_j})$.
  2. Let $(\tilde{\mathsf{Q}}^{(j)}, \tilde{\mathsf{y}}_{\mathsf{Q}}^{(j)}) \leftarrow \mathsf{GC.Sim}(1^\lambda, 1^{\ell'}, 1^{|\mathsf{Q}|}, \hat{\mathsf{e}}_{\mathsf{Q}}^{(j)})$.
- For $j = (k-2), \ldots, 0$,
  1. Compute $\hat{\mathsf{y}}_{\mathsf{Q}}^{(j)} \leftarrow \mathsf{Sim.Q}[\mathsf{pp}, \mathsf{tag}_{j+1}, \ell', \tilde{\mathsf{y}}_{\mathsf{Q}}^{(j+1)}](\mathsf{vk}_{v_j})$.
  2. Let $(\tilde{\mathsf{Q}}^{(j)}, \tilde{\mathsf{y}}_{\mathsf{Q}}^{(j)}) \leftarrow \mathsf{GC.Sim}(1^\lambda, 1^{\ell'}, 1^{|\mathsf{Q}|}, \hat{\mathsf{y}}_{\mathsf{Q}}^{(j)})$.

Experiment $2k+1$**.** Here k varies from 1 to z. In this experiment, instead of using Q for encrypting the labels $\mathsf{e}_{\mathsf{Q}}^{(k)}$, instead we use Sim.Q to compute $\hat{\mathsf{y}}^{(k-1)}$ and use that inside GC.Sim. In the **Ciphertext Queries** phase, we make the following change,
For $j = (k-1)$,

1. Let $\mathsf{e}_{\mathsf{Q}}^{(j+1)}$ be denoted by the set of labels $\{(Y_{\iota,0}, Y_{\iota,1})\}_{\iota \in [\ell']}$. Let y_ι be the ι^{th} bit of $\mathsf{vk}_{v_{j+1}}$. Use $\mathsf{y}_{\mathsf{Q}}^{(j+1)} = \{Y_{\iota,\mathsf{y}_\iota}\}_{\iota \in [\ell']}$ as the labels.
2. Compute $\hat{\mathsf{y}}_{\mathsf{Q}}^{(j)} \leftarrow \mathsf{Sim.Q}[\mathsf{pp}, \mathsf{tag}_{j+1}, \ell', \mathsf{y}_{\mathsf{Q}}^{(j+1)}](\mathsf{vk}_{v_j})$.
3. Let $(\tilde{\mathsf{Q}}^{(j)}, \tilde{\mathsf{y}_{\mathsf{Q}}^{(j)}}) \leftarrow \mathsf{GC.Sim}(1^\lambda, 1^{\ell'}, 1^{|\mathsf{Q}|}, \hat{\mathsf{y}}_{\mathsf{Q}}^{(j)})$.

Experiment $2z+2$**.** In this experiment, we change how $\tilde{\mathsf{Q}}^{(z)}$ and the garbled labels $\tilde{\mathsf{y}}^{(z)}$ are computed by calling the garbled circuit simulator. In the **Ciphertext Queries** phase, we make the following change -
Let $(\tilde{\mathsf{Q}}^{(z)}, \tilde{\mathsf{y}}_{\mathsf{Q}}^{(z)}) \leftarrow \mathsf{GC.Sim}(1^\lambda, 1^{\ell'}, 1^{|\mathsf{Q}|}, \mathsf{e}^{\mathsf{T}})$ and use $\tilde{\mathsf{y}}_{\mathsf{Q}}^{(z)}$ instead of $\mathsf{e}_{\mathsf{Q}}^{(z)}$ inside Sim.Q.

Experiment $2z+3$**.** In this experiment, instead of using Q for encrypting the labels y^{T}, instead we use Sim.Q to compute $\hat{\mathsf{y}}^{\mathsf{T}}$. In the **Ciphertext Queries** phase, we make the following change - Let e_{T} be denoted by the set of labels $\{(Y_{\iota,0}, Y_{\iota,1})\}_{\iota \in \ell}$. Let y_ι be the ι^{th} bit of $\mathsf{BFE.mpk}_{\mathsf{tag}^*}$. Use $\mathsf{y}_{\mathsf{T}} = \{Y_{\iota,\mathsf{y}_\iota}\}_{\iota \in \ell}$ as the labels. Compute $\hat{\mathsf{y}}^{\mathsf{T}} \leftarrow \mathsf{Sim.Q}[\mathsf{pp}, 0, \ell, \mathsf{y}_{\mathsf{T}}](\mathsf{vk}_{\mathsf{tag}^*})$. Additionally, use $\hat{\mathsf{y}}^{\mathsf{T}}$ instead of e^{T} as the input to the garble simulator algorithm.

Experiment $2z+4$**.** In this experiment, we simulate T using the garbled circuit simulator hard-coded with the BFE.ct. In the **Ciphertext Queries** phase, we make the following change -
Let $\mathsf{ct}^*_{\mathsf{BFE}} \leftarrow \mathsf{BFE.Enc}(\mathsf{BFE.mpk}_{\mathsf{tag}^*}, m)$. Compute $(\tilde{\mathsf{T}}, \tilde{\mathsf{y}}^{\mathsf{T}}) \leftarrow \mathsf{GC.Sim}(1^\lambda, 1^\ell, 1^{|\mathsf{T}[m]|}, \mathsf{ct}^*_{\mathsf{BFE}})$.

Experiment $2z+5$. In this experiment, we simulate $\mathsf{BFE.Enc}$ and $\mathsf{BFE.KeyGen}$ to remove all information about the message m.
In the **Key Queries** phase, use routine $\mathsf{Sim.LeafGen}$ instead of $\mathsf{PartSim.LeafGen}$.
In the **Ciphertext Queries** phase, change how we sample $\mathsf{ct}^*_{\mathsf{BFE}}$ and set it as
$\mathsf{ct}^*_{\mathsf{BFE}} \leftarrow \mathsf{BFE.S}_2(\mathsf{BFE}^{\mathsf{tag}^*}.\mathsf{st}_1, \Pi^{m^{\mathsf{tag}^*}})$. ($\Pi^{m^{\mathsf{tag}^*}}$ contains the functions and their evaluations at m^{tag^*} for the tag tag^*).

Let $\mathcal{P}^i_{\mathcal{A}}(\lambda)$ be the probability that adversary $\mathcal{A}$ outputs 1 on Experiment i run on security parameter λ.

Lemma 5.2. If PRF is a secure pseudorandom functions, then for every adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $|\mathcal{P}^0_{\mathcal{A}}(\lambda) - \mathcal{P}^1_{\mathcal{A}}(\lambda)| = \mathsf{negl}(\lambda)$.

Proof. Suppose there existed an adversary $\mathcal{A}$ that distinguishes with some non-negligible probability, then we can construct an adversary $\mathcal{B}$ that distinguishes between the a truly random function and a PRF with non-negligible probability. Experiment 0 consists of computation corresponding to the PRF and Experiment 1 consists of computation corresponding to the truly random function. Thus we can argue the lemma from PRF security. □

Lemma 5.3. If GC is a secure garbling scheme, then for k from 1 to z, for every adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $|\mathcal{P}^{2k-1}_{\mathcal{A}}(\lambda) - \mathcal{P}^{2k}_{\mathcal{A}}(\lambda)| = \mathsf{negl}(\lambda)$.

Proof. Suppose there existed an adversary $\mathcal{A}$ that distinguishes with some non-negligible probability, then we can construct an adversary $\mathcal{B}$ that distinguishes between the garbling security game with non-negligible probability. Consider the circuit $\tilde{\mathsf{Q}}^{(k-1)}$. In experiment $2k-1$, we compute the circuit using the algorithm $(\tilde{\mathsf{Q}}^{(k-1)}, \mathsf{e}^{(k-1)}_{\mathsf{Q}}) \leftarrow \mathsf{GC.Garble}(1^\lambda, \mathsf{Q}[\mathsf{pp}, v_k, \ell', \mathsf{e}^{(k)}_{\mathsf{Q}}])$. In experiment $2k$, we compute the same circuit from a garbled circuit simulator, where we compute $(\tilde{\mathsf{Q}}^{(k-1)}, \tilde{\mathsf{y}}^{(k-1)}_{\mathsf{Q}}) \leftarrow \mathsf{GC.Sim}(1^\lambda, 1^{\ell'}, 1^{|\mathsf{Q}|}, \hat{\mathsf{e}}^{(k-1)}_{\mathsf{Q}})$ where the labels $\tilde{\mathsf{y}}^{(k-1)}_{\mathsf{Q}}$ correspond to the labels $\mathsf{vk}_{v_{k-1}}$. Observe that in both games, we only proceed with computation on $\tilde{\mathsf{y}}^{(k-1)}_{\mathsf{Q}}$, specifically in experiment $2k-1$, we compute $\mathsf{e}^{(k-1)}_{\mathsf{Q}}$ and only compute on labels corresponding to $\mathsf{vk}_{v_{k-1}}$. Thus by garbled security on $\mathsf{Q}[\mathsf{pp}, v_k, \ell', \mathsf{e}^{(k)}_{\mathsf{Q}}]$ and input $\mathsf{vk}_{v_{k-1}}$, the advantage of $\mathcal{B}$ is the same as advantage of $\mathcal{A}$. □

Lemma 5.4. If OTSE is a secure one time encryption scheme, then for k from 1 to z, for every adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $|\mathcal{P}^{2k}_{\mathcal{A}}(\lambda) - \mathcal{P}^{2k+1}_{\mathcal{A}}(\lambda)| = \mathsf{negl}(\lambda)$.

Proof. We can assume that our proof proceeds through a sequence of sub-experiments where we switch one challenge ciphertext at a time. The loss in the security experiment will be bounded by the number of queries we can make and thus will be q times the loss between one of the sub-experiments. Assume that we switch the ciphertext on challenge tag tag^*.

Suppose there existed an adversary $\mathcal{A}$ that distinguishes with some non-negligible probability, then we can construct an adversary $\mathcal{B}$ that distinguishes between the OTSE security game with non-negligible probability. Consider the routines $\mathsf{Q}[\mathsf{pp}, \mathsf{tag}_k, \ell', \mathsf{e}^{(k)}_{\mathsf{Q}}](\mathsf{vk}_{v_{k-1}})$ and $\mathsf{Sim.Q}[\mathsf{pp}, \mathsf{tag}_k, \ell',$

$\mathsf{y}_{\mathsf{Q}}^{(k)}](\mathsf{vk}_{v_{k-1}})$ used to compute $\hat{\mathsf{y}}_{\mathsf{Q}}^{(k)}$. Let $\mathsf{e}_{\mathsf{Q}}^{(k)}$ be denoted by the set of labels $\{(Y_{\iota,0}, Y_{\iota,1})\}_{\iota\in[\ell']}$. Let y_ι be the ι^{th} bit of vk_{v_k}. Let $\mathsf{y}_{\mathsf{Q}}^{(j+1)} = \{Y_{\iota,\mathsf{y}_\iota}\}_{\iota\in[\ell']}$ as the labels.

The adversary $\mathcal{B}$ functions as follows, it gets the public parameter pp. It then outputs x^* as the challenge message as $\mathsf{vk}_{v_{k-1}||0}||\mathsf{vk}_{v_{k-1}||1}$. Challenger samples $(\mathsf{vk}_{v_{k-1}}, \mathsf{sk}_{v_{k-1}})$ and sends $(\mathsf{vk}_{v_{k-1}})$ to $\mathcal{B}$. $\mathcal{B}$ obtains the signature $\sigma \leftarrow \mathsf{SSign}(\mathsf{pp}, \mathsf{sk}_{v_{k-1}}, x^*)$. It uses σ in keygen procedure as it does not know $\mathsf{sk}_{v_{k-1}}$. Finally, it outputs $i^* = \{\mathsf{tag}_k\ell' + \iota\}_{\iota\in[\ell']}$ as the challenge locations and sets the multi-message challenge ciphertexts as $M_0^* = \{Y_{\iota,1-\mathsf{y}_\iota}\}_{\iota\in[\ell']}$ and $M_1^* = \{Y_{\iota,\mathsf{y}_\iota}\}_{\iota\in[\ell']}$. The ciphertext set $C_0^* = \{\mathsf{SEnc}(\mathsf{pp}, (\mathsf{vk}_{v_{k-1}}, \mathsf{tag}_k\ell' + \iota, 1-\mathsf{y}_\iota), Y_{\iota,1-\mathsf{y}_\iota})\}_{\iota\in[\ell']}$. Consider the ciphertexts $C_1^* = \{\mathsf{SEnc}(\mathsf{pp}, (\mathsf{vk}_{v_{k-1}}, \mathsf{tag}_k\ell' + \iota, 1-\mathsf{y}_\iota), Y_{\iota,\mathsf{y}_\iota})\}_{\iota\in[\ell']}$. Note that when we switch from using Q to Sim.Q, this is precisely the change, and thus we can rely on selective OTSE security to make our claim. □

Lemma 5.5. If GC is a secure garbling scheme, for every adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $|\mathcal{P}_{\mathcal{A}}^{2z+1}(\lambda) - \mathcal{P}_{\mathcal{A}}^{2z+2}(\lambda)| = \mathsf{negl}(\lambda)$.

Proof. This is very similar to the proof of Lemma 5.3. □

Lemma 5.6. If OTSE is a secure one time encryption scheme, for every adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $|\mathcal{P}_{\mathcal{A}}^{2z+2}(\lambda) - \mathcal{P}_{\mathcal{A}}^{2z+3}(\lambda)| = \mathsf{negl}(\lambda)$.

Proof. This is very similar to the proof of Lemma 5.4. □

Lemma 5.7. If GC is a secure garbling scheme, for every adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $|\mathcal{P}_{\mathcal{A}}^{2z+3}(\lambda) - \mathcal{P}_{\mathcal{A}}^{2z+4}(\lambda)| = \mathsf{negl}(\lambda)$.

Proof. This is very similar to the proof of Lemma 5.3. □

Lemma 5.8. If BFE is a secure bounded-collusion simulation-secure FE scheme scheme, for every adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $|\mathcal{P}_{\mathcal{A}}^{2z+4}(\lambda) - \mathcal{P}_{\mathcal{A}}^{2z+5}(\lambda)| = \mathsf{negl}(\lambda)$.

Proof. We rely on the security for each instance of the BFE scheme where a keygen or ciphertext query is made to the tagged scheme. Let q' be the total number of BFE instances on which we make any kind of query. We can go through a sequence of q' sub-experiments where we change each scheme to it's simulated counterpart. The formal details are routine, and if the advantage in breaking the BFE scheme is $\mathsf{negl}'(\lambda)$, then the advantage between the two experiments is atmost $q'\mathsf{negl}'(\lambda)$. Since $q = \mathsf{poly}(\lambda)$, the winning probability is bounded by a negligible function $\mathsf{negl}(\lambda)$. □

□

5.3 Central Theorem

Finally, by combining the above theorem (Theorem 5.1) with Theorem 4.5, we get our central theorem as follows.

Theorem 5.9. If IBE is a secure IBE scheme and BFE is a λ-bounded collusion simulation-secure FE scheme (as per Definition 4.1), then there exists a dynamic bounded collusion simulation-secure FE scheme (as per Definition 4.2). And, the dynamic FE scheme can be obtained via a non-black-box transformation from the static FE scheme.

As discussed in [AMVY21], dynamic bounded collusion FE schemes imply IBE for most basic function classes, thus the above theorem is unconditional, and we could simplify it as follows.

Corollary 5.10. If BFE is a λ-bounded collusion simulation-secure FE scheme (as per Definition 4.1), then there exists a dynamic bounded collusion simulation-secure FE scheme (as per Definition 4.2) obtained via a non-black-box transformation.

This immediately leads to new results by combining with [GSW21, Wee21].

Corollary 5.11. If IBE is a secure IBE scheme, then there exists a dynamic-bounded collusion simulation-secure ABE scheme for Turing Machines.

Corollary 5.12. If Learning with Errors assumption is hard, then there exists a dynamic-bounded collusion simulation-secure ABE scheme for DFAs in the secret-key-selective setting.

6 Multi-Authority ABE: Tagged and Dynamic Collusion

The second result in our paper is a multi-authority attribute-based encryption scheme (MA-ABE) in the bounded collusion model for efficient computational secret sharing schemes (CSS). We show that our construction achieves the desired dynamic collusion property. We obtain our result via the tagged FE framework that we discussed in Section 4.3. We start by recalling the notion of access structures and MA-ABE.

6.1 Definition and Preliminaries

Access structures and computational secret-sharing. We recall the concepts of access structures and computational secret-sharing schemes (CSSS). We follow the notation from prior works [GPSW06, LW11].

Definition 6.1 (Access Structures). Let $\{P_i\}_{i\in[n]}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{\{P_1,\ldots,P_n\}}$ is monotone if $\forall B, C$: if $B \in \mathbb{A}$ and $B \subseteq C$, then $C \in \mathbb{A}$. An access structure (respectively, monotone access structure) is a collection (respectively, monotone collection) $\mathbb{A}$ of non-empty subsets of $\{P_i\}_{i\in[n]}$. The sets in $\mathbb{A}$ are called the authorized sets, and the sets not in $\mathbb{A}$ are called the unauthorized sets.

As in prior works, attributes will play the role of parties and we will only consider monotone access structures. We observe that more general access structures can be (inefficiently) realized with our techniques by letting the negation of an attribute be a separate attribute (this doubles the total number of attributes).

Definition 6.2 (Computational Secret-Sharing Schemes (CSSS)). A computational secret sharing scheme Π over a set of parties $\mathcal{P}$ contains two polynomial time algorithms:

$\mathsf{Share}(1^\lambda, 1^\ell, \mathbb{A}, \rho, s) \to \{\mathsf{sh}_i\}_{i \le n}$. The dealer algorithm takes as input the access structure $\mathbb{A}$ and share mapping function $\rho : [n] \to [\ell]$ along with the secret $s \in \{0,1\}^\lambda$ and number of parties ℓ. It outputs n shares.

$\mathsf{Recon}(\{\mathsf{sh}_i\}_{i \in T}) \to s$. The reconstruction algorithm takes as input a subset of shares $\{sh_i\}_{i \in T}$ for some subset $T \subseteq [n]$, and outputs a reconstructed share s if the set of corresponding parties make up an authorized set.

Correctness. A CSS scheme is said to be correct if for every $\lambda \in \mathbb{N}$, every supported access structure $(\mathbb{A}, \rho)$ and number of corresponding parties ℓ, every *authorized* set of users $U \in \mathbb{A}$, every secret $s \in \{0,1\}^\lambda$, the following holds:

$$\Pr\left[\mathsf{Recon}(\{\mathsf{sh}_i\}_{i:\rho(i) \in U}) = s \ : \ \{\mathsf{sh}_i\}_i \leftarrow \mathsf{Share}(1^\lambda, 1^\ell, \mathbb{A}, \rho, s)\right] = 1.$$

Security. In terms of security, we say CSS satisfy secrecy if the any set of unauthorized shares hide the secret.

Definition 6.3 (CSS secrecy). A CSS scheme for access structure $(\mathbb{A}, \rho)$ satisfies secrecy if there exists a polynomial-time simulator Sim such that for every supported access structure $(\mathbb{A}, \rho)$ and number of corresponding parties ℓ, every unauthorized subset $U \notin \mathbb{A}$, every secret $s \in \{0,1\}^\lambda$, the following distributions are computationally indistinguishable:

$$\left\{\mathsf{Sim}\left(1^\lambda, 1^\ell, \mathbb{A}, U\right)\right\}_\lambda \approx_c \left\{\{\mathsf{sh}_i\}_{i:\rho(i) \in U} \ : \ \{\mathsf{sh}_i\}_{i \in [n]} \leftarrow \mathsf{Share}(1^\lambda, 1^\ell, \mathbb{A}, \rho, s)\right\}_\lambda .$$

Syntax of MA-ABE. A MA-ABE scheme for a CSS schemes consists of the following PPT algorithms.

$\mathsf{GSetup}(1^\lambda) \to \mathsf{crs}$. The setup algorithm takes as input the security parameter λ, and outputs common reference string crs. (We assume that crs includes the space of attribute authorities $\mathcal{AU}$ and the space of global identifiers of users $\mathcal{GID}$, and every algorithm receives crs as an implicit input.)

$\mathsf{ASetup}(\mathsf{crs}, u) \to (\mathsf{pk}_u, \mathsf{sk}_u)$. The authority setup algorithm takes as input crs and authority $u \in \mathcal{AU}$, and outputs an authority key pair.

$\mathsf{KeyGen}(\mathsf{GID}, \mathsf{sk}_u) \to \mathsf{sk}_{\mathsf{GID},u}$. The key generation algorithm takes as input a global identifier $\mathsf{GID} \in \mathcal{GID}$ and sk_u for an authority $u \in \mathcal{AU}$. It outputs the corresponding secret key.

$\mathsf{Enc}((\mathbb{A},\rho),\{\mathsf{pk}_u\},\mu) \to \mathsf{ct}$. The encryption algorithm takes in a message μ, a CSSS access structure $(\mathbb{A},\rho)$. Here, the set $\{\mathsf{pk}_u\}$ denotes all public keys corresponding to the authorities which are specified by the access structure $\mathbb{A}$. It outputs a ciphertext ct. (We assume that the ciphertext implicitly contains $(\mathbb{A},\rho)$. We consider any access structure that is imposed by some polynomial-sized monotone circuit and ρ is its share-labeling function.)

$\mathsf{Dec}(\mathsf{ct},\{\mathsf{sk}_{\mathsf{GID},u}\}) \to \mu \cup \perp$. The decryption algorithm takes as input a ciphertext ct and secret keys issued for different attributes by the respective authorities. It outputs a message μ, or $\perp$ if decryption fails.

Correctness. A MA-ABE scheme is said to be correct if for every $\lambda \in \mathbb{N}$, any set U of attribute authorities, any CSSS access structure $(\mathbb{A},\rho)$ defined over set U, $\mathsf{GID} \in \mathcal{GID}$, message μ, and a set of authorized parties $S \subset U$ which satisfy $\mathbb{A}$, the following holds:

$$\Pr\left[\mathsf{Dec}(\mathsf{ct},\{\mathsf{sk}_{\mathsf{GID},u}\}_{u\in S}) = \mu : \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{GSetup}(1^\lambda) \\ (\mathsf{pk}_u,\mathsf{sk}_u) \leftarrow \mathsf{ASetup}(\mathsf{crs},u)\ \forall u \in U \\ \mathsf{sk}_{\mathsf{GID},u} \leftarrow \mathsf{KeyGen}(\mathsf{GID},\mathsf{sk}_u)\ \forall u \in U \\ \mathsf{ct} \leftarrow \mathsf{Enc}((\mathbb{A},\rho),\{\mathsf{pk}_u\}_u,\mu)) \end{array}\right] = 1.$$

Security. In terms of security, we say MA-ABE is fully secure if the IND-CPA security holds even if the attacker corrupts some of the authorities as well as corrupts secret keys generated by honest authorities *as long as* there is no combination of secret keys and corrupt authorities that are authorized to decrypt the challenge ciphertext.

Definition 6.4 (MA-ABE full security)**.** A MA-ABE scheme is fully secure if for every stateful admissible PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, the following holds

$$\Pr\left[\mathcal{A}^{O(\cdot,\cdot)}(\mathsf{ct}) = b : \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{GSetup}(1^\lambda), \mathcal{C} = \emptyset, \mathcal{N} = \emptyset, b \leftarrow \{0,1\} \\ (U,(\mathbb{A},\rho),(\mu_0,\mu_1)) \leftarrow \mathcal{A}^{O(\cdot,\cdot)}(1^\lambda,\mathsf{crs}) \\ \mathsf{ct} \leftarrow \mathsf{Enc}((\mathbb{A},\rho),\{\mathsf{pk}_u\}_{u\in U},\mu_b)) \end{array}\right] \le \frac{1}{2} + \mathsf{negl}(\lambda),$$

where $O(\cdot,\cdot)$ is a *stateful* oracle that receives four types of queries and responds as follows:

$(\mathsf{AuthGen},\mathsf{u})$. $\mathcal{A}$ submits an authority $u \notin \mathcal{C} \cup \mathcal{N}$. O samples $(\mathsf{pk}_u,\mathsf{sk}_u) \leftarrow \mathsf{ASetup}(\mathsf{crs},u)$, sets $\mathcal{N} := \mathcal{N} \cup \{u\}$, stores $(u,\mathsf{pk}_u,\mathsf{sk}_u)$ in its state, and outputs pk_u to $\mathcal{A}$.

$(\mathsf{Corrupt},\mathsf{u})$. $\mathcal{A}$ submits an authority $u \in \mathcal{N}$. O sets $\mathcal{N} := \mathcal{N} \setminus \{u\}$, $\mathcal{C} := \mathcal{C} \cup \{u\}$, and sends sk_u from its state to $\mathcal{A}$.

$(\mathsf{Register},(\mathsf{u},\mathsf{pk_u}))$. $\mathcal{A}$ submits an authority $u \notin \mathcal{C} \cup \mathcal{N}$ with key pk_u. O sets $\mathcal{C} := \mathcal{C} \cup \{u\}$, stores $(u,\mathsf{pk}_u,\perp)$ in its state.

$(\mathsf{KeyGen},(\mathsf{u},\mathsf{GID}))$. $\mathcal{A}$ submits an authority $u \in \mathcal{N}$ with identifier GID. O samples $\mathsf{sk}_{\mathsf{GID},u} \leftarrow \mathsf{KeyGen}(\mathsf{GID},\mathsf{sk}_u)$ where $(u,\mathsf{pk}_u,\mathsf{sk}_u)$ is in its state, stores (u,GID) as well in its internal state, and sends $\mathsf{sk}_{\mathsf{GID},u}$ to $\mathcal{A}$.

And, the adversary $\mathcal{A}$ is admissible as long as for each unique identifier GID queried by $\mathcal{A}$, the set of authorities $\mathcal{C} \cup \{u : (u, \mathsf{GID}) \text{ was queried}\}$ is not an authorized set.

Definition 6.5 (MA-ABE static collusion security)**.** An MA-ABE scheme is static collusion-bounded secure if the GSetup algorithm takes in an additional parameter q (the collusion bound) and in the security game, $\mathcal{A}$ specifies 1^q at the beginning, and is admissible if it correctly guesses b *and* the number of unique identifiers GID for which $\mathcal{A}$ submits KeyGen queries is $\leq q$.

Definition 6.6 (MA-ABE dynamic collusion security)**.** An MA-ABE scheme is dynamic collusion-bounded secure if the Enc algorithm takes in an additional parameter q (the collusion bound) and in the security game, $\mathcal{A}$ specifies 1^q during the "challenge" phase, and is admissible if it correctly guesses b *and* the number of unique identifiers GID for which $\mathcal{A}$ submits KeyGen queries is $\leq q$.

6.2 Statically Secure MA-ABE for CSS schemes

Ingredients. Let $\mathsf{IBE} = (\mathsf{IBE.Setup}, \mathsf{IBE.KeyGen}, \mathsf{IBE.Enc}, \mathsf{IBE.Dec})$ be an identity based encryption scheme and $\Pi = (\mathsf{Share}, \mathsf{Recon})$ be a secure CSS scheme, and $\mathcal{H}$ be a hash function modelled as a random oracle. Below we provide our construction for a statically secure MA-ABE scheme.

We want to point out that our construction can be instantiated from the minimal assumption of public key encryption. The reason we use IBE in our construction instead is for an easier exposition and, as we discuss later, because it results in a tagged MA-ABE scheme quite easily.

$\mathsf{GSetup}(1^\lambda, 1^q) \to \mathsf{crs}$. Our global parameters simply specify the domain and range of a random oracle, which grows with collusion bound q.

- Let $\mathcal{H}$ be a hash function from $\mathcal{GID} \times [q\lambda] \to [q^2]$.
- Output global parameters $\mathsf{crs} = (\lambda, q, \mathcal{H})$.

$\mathsf{ASetup}(\mathsf{crs}, u) \to (\mathsf{pk}_u, \mathsf{sk}_u)$. Each authority independently generates an IBE key pair, which encryptors later use to generate ciphertext components.

- $(\mathsf{ibe.mpk}_u, \mathsf{ibe.msk}_u) \leftarrow \mathsf{IBE.Setup}(1^\lambda, \mathcal{ID} = ([q\lambda] \times [q^2]))$.
- Output $\mathsf{pk}_u = \mathsf{ibe.mpk}_u$, $\mathsf{sk}_u = \mathsf{ibe.msk}_u$.

$\mathsf{KeyGen}(\mathsf{GID}, \mathsf{sk}_u) \to \mathsf{sk}_{\mathsf{GID},u}$. A secret key consists of $q \cdot \lambda$ IBE keys, where each IBE key is randomly drawn from disjoint space of size q^2. Effectively, our keyspace is partitioned into $q\lambda$ intervals of size $[q^2]$. For each interval, it deterministically samples a random identity from $[q^2]$ for each $i \in [q\lambda]$ according to the random oracle on the GID.

- For $i \in [q\lambda]$, compute $\mathsf{ibe.sk}_{u,\mathsf{GID},i} \leftarrow \mathsf{IBE.KeyGen}(\mathsf{sk}_u, \mathsf{ID}_{\mathsf{GID},i})$ where $\mathsf{ID}_{\mathsf{GID},i} = (i, \mathcal{H}(\mathsf{GID}, i))$.
- Output $\mathsf{sk}_{\mathsf{GID},u} = \{\mathsf{ibe.sk}_{u,\mathsf{GID},i}\}_{i \in [q\lambda]}$.

$\mathsf{Enc}((\mathbb{A}, \rho), \{\mathsf{pk}_u\}_{u \in U}, \mu) \to \mathsf{ct}$. To encrypt, we simply *additively* secret share our message into $q\lambda$ shares $\mu_1, \mu_2, \ldots \mu_{q\lambda}$ (as the keyspace is partitioned). Then each secret share is itself secret shared via the CSS access structure $(\mathbb{A}, \rho)$ q^2 times, and encrypted resulting share under the identity corresponding to the authority and slot.

- Let $\mu_1, \mu_2, \ldots \mu_{q\lambda}$ be an additive N-of-N secret sharing of μ. (That is, $\mu = \oplus_i \mu_i$.)
- For all $i \in [q\lambda]$ and $\mathsf{id} \in [q^2]$,
  - Compute shares $\{\mathsf{sh}_{i,\mathsf{id},j}\}_j \leftarrow \mathsf{Share}(1^\lambda, 1^{|U|}, \mathbb{A}, \rho, \mu_i)$
  - For all $u \in U$, compute $\mathsf{ct}_{u,i,\mathsf{id}} \leftarrow \mathsf{IBE.Enc}(\mathsf{pk}_u, (i, \mathsf{id}), \{\mathsf{sh}_{i,\mathsf{id},j}\}_{j:\rho(j)=u})$.[13]
- Output $\mathsf{ct} = ((\mathbb{A}, \rho), \{\mathsf{ct}_{u,i,\mathsf{id}}\}_{u,i,\mathsf{id}})$.

$\mathsf{Dec}(\mathsf{ct}, \{\mathsf{sk}_{\mathsf{GID},u}\}_{u \in S}) \rightarrow \mu$. To decrypt, for each of the $q\lambda$ intervals, we simply recover the corresponding secret share μ_i for exactly one of the q^2 CSS schemes as determined by the random oracle on the GID. To recover the final message, we simply add our message shares together.

- Parse $\mathsf{ct} = ((\mathbb{A}, \rho), \{\mathsf{ct}_{u,i,\mathsf{id}}\}_{u,i,\mathsf{id}})$, and $\mathsf{sk}_{\mathsf{GID},u} = \{\mathsf{ibe.sk}_{u,\mathsf{GID},i}\}_i$.
- For each $i \in [q\lambda]$,
  - For each $u \in S$, recover $\{\mathsf{sh}_{i,\mathcal{H}(\mathsf{GID},i),j}\}_{j:\rho(j)=u} = \mathsf{IBE.Dec}((\mathsf{sk}_{u,\mathsf{GID},i}, \mathsf{ct}_{u,i,\mathcal{H}(\mathsf{GID},i)})$.
  - Recover $\mu_i = \mathsf{Recon}(\{\mathsf{sh}_{i,\mathcal{H}(\mathsf{GID},i),j}\}_{j:\rho(j)\in S})$.
- Output $\mu = \oplus_{i \in [q\lambda]} \mu_i$.

Correctness and Efficiency. The correctness of the scheme follows from the correctness of IBE scheme and the reconstruction property of CSS. Note that by reconstruction property of CSS, we have that in every honestly computed ciphertext, for each additive share μ_i and its corresponding CSS shares $\{\mathsf{sh}_{i,\mathsf{id},j}\}_j$, we have that $\mathsf{Recon}(\{\mathsf{sh}_{i,\mathcal{H}(\mathsf{GID},i),j}\}_{\rho(j)\in S})$ for every set S that is authorized for CSS access structure $(\mathcal{A}, \rho)$. Combining this with the fact that the IBE decryption part of the above decryption procedure recovers $\{\mathsf{sh}_{i,\mathcal{H}(\mathsf{GID},i),j}\}_{\rho(j)=u}$ whenever $\mathsf{id} = \mathcal{H}(\mathsf{GID}, i)$ using the IBE secret key $\mathsf{ibe.sk}_{u,\mathsf{GID},i}$. Since a set of authorized secret keys for a particular GID contains all such IBE keys for $i \in [q\lambda]$ and $u \in S$ where S is the set of authorized attributes, thus the decryption correctness follows by combining above facts.

Next, the efficiency of the scheme follows directly from the efficiency of the IBE system. Each user's attribute key contains a fixed number $(q\lambda)$ of IBE secret keys, while the ciphertext contains $q^3 \cdot \lambda \cdot |U|$ IBE ciphertexts. Thus, the secret keys and ciphertexts are fixed polynomial in the collusion bound q. Since the goal is to design a statically secure MA-ABE scheme, thus it satisfies the required efficiency condition.

Security. The main intuition behind the security proof can be explained in two steps.

1. First, observe that $\mathcal{H}$ is modelled as a random oracle and an admissible adversary does not make key generation queries for honest authorities on more than q distinct global identifiers. Thus, for each index $i \in [q\lambda]$, the number of IBE identities for which a secret key will get generated is at most q. Note that there are q^2 possible identities for index i. Now, information-theoretically, we can show that there will exist at least one index i^* such that for all q queried $\mathsf{GID}_1, \ldots, \mathsf{GID}_q$ identifiers, their corresponding hash values $\mathcal{H}(\mathsf{GID}_1, i^*), \ldots, \mathcal{H}(\mathsf{GID}_q, i^*)$ are all pairwise distinct (i.e., their are no collisions). This no-collision property for queried identifiers is crucial in the next step of the proof.

[13]Recall that IBE supports encryptions of unbounded length messages via hybrid encryption.

2. Second, note that we secret share the message μ using a N-of-N secret sharing scheme into $q\lambda$ shares $\{\mu_i\}_i$. Thus, to prove IND-CPA security, it is sufficient to show that one of these shares is computationally hidden. Once we prove this, then the security follows directly from the secrecy of secret sharing.

   At this point, we can use the no-collision property of the queried identifiers. That is, we know that there exists an i^* where all q queried $\mathsf{GID}_1, \ldots, \mathsf{GID}_q$ identifiers are uniquely hashed. Now recall that for each i, we do a second level of secret sharing for each share μ_i where we use the CSS to secret share each μ_i independently q^2 times. That is, for each possible hash value $\mathsf{id} \in [q^2]$, we do a fresh secret sharing of μ_i for each i. By the admissibility constraint on the attacker, we have that for every queried GID the attacker does not have a set of authorized keys. This, combined with the no-collision property, gives us that there cannot be a hash value $\mathsf{id} \in [q^2]$ where the attacker has enough CSS shares to reconstruct μ_i. Thus, by using CSS secrecy property for each sharing (for $\mathsf{id} \in [q^2]$) and security of IBE scheme (which is applied for hiding all unauthorized CSS shares), we get that μ_{i^*} is computationally hidden. This gives us our result.

Below we formalize the above intuition in a full security proof. Although, we only prove security when the attacker makes all corruption queries in the pre-challenge phase. It can be easily extended to handle post-challenge corruption queries by making a small modification to the construction. We will use a non-committing encryption scheme to encrypt the inner share and the corresponding secret key will be available to the appropriate key holder. This can be easily executed in the random oracle model by simply using random oracle programmability. However, for ease of exposition, we stick to the proof in the simpler case of zero post-challenge queries. Formally, we show the following.

Theorem 6.7. If IBE is a secure IBE scheme and Π is a secure CSS scheme, then the above scheme is a statically secure bounded-collusion MA-ABE scheme as per Definition 6.5 in the random oracle model where the adversary makes all queries in the pre-challenge phase.

Proof. To begin, we prove a simple useful information-theoretic lemma.

Lemma 6.8. Let $p = p(\lambda), q = q(\lambda)$ be any polynomials in λ, and $\mathbf{R} \in [q^2]^{p \times (q\lambda)}$ be a random matrix with entries uniformly random in $[q^2]$. With all but negligible probability, for all subset of rows $\mathbf{R}^* \in [q^2]^{q \times (q\lambda)}$ there exists a column of $\mathbf{R}^*$ with unique entries.

Proof. First, we can examine the probability that for a uniformly random matrix $\mathbf{R}' \in [q^2]^{q \times (q\lambda)}$ has no columns with unique entries. Observe that the probability any fixed column has unique entries can be computed by consider elements sequentially as

$$\frac{q^2}{q^2} \cdot \frac{q^2-1}{q^2} \cdots \frac{q^2-q+1}{q^2} \geq \left(\frac{q^2-q}{q^2}\right)^q \geq \left(1 - \frac{1}{q}\right)^q \geq \frac{1}{e^2}$$

The last inequality follows from the fact that for $x \leq 1/2$ we have that $1 - x \geq e^{-2x}$. Since $q \geq 2$, thus $1 - 1/q \geq e^{-2/q}$ which implies the above.

Now, since all $q\lambda$ columns are independently sampled, thus the probability that no column (out of the $q\lambda$ columns) has unique entries can be upper bounded by $\left(1 - \frac{1}{e^2}\right)^{q\lambda} \leq 2^{-q\lambda/2}$. In words, this is a very significantly negligible function. To complete the proof, we simply need to show that for a randomly sampled $\mathbf{R}$, of dimensions as mentioned above, there does not exists any submatrix $\mathbf{R}^*$ containing q distinct rows of $\mathbf{R}$ such that no column of $\mathbf{R}^*$ has unique entries. This follows via a simple union bound over all possible submatrices of $\mathbf{R}$. Note that the total number of such unique submatrices is at most p^q (since all size q subsets of $[p]$ define a unique submatrix). Thus, using union bound, we get that the probability *any* such submatrix has no column of unique entries is at most

$$p^q \cdot 2^{-q\lambda/2} = 2^{q(\log p - \lambda/2)} \leq 2^{-q\cdot\omega(\log\lambda)} = \mathsf{negl}(\lambda).$$

Thus, the lemma follows. □

Next, we will show through a sequence of experiments that our construction is a secure static-bounded MA-ABE scheme. Without loss of generality, we assume $\mathcal{A}$ always submits secret key queries to exactly q unique GID's (if an adversary queries any less, we can produce another adversary $\mathcal{A}'$ which simply runs $\mathcal{A}$ and queries arbitrary GID's until q unique GID's are queried).

Experiment 0: This is the original security game.

1. The challenger initializes sets $\mathcal{C}, \mathcal{N} = \emptyset$ and empty dictionary D.
2. **Setup Phase**:
   - $\mathcal{A}$ sends 1^q to challenger and receives $\mathcal{H} : \mathcal{GID} \times [q\lambda] \to [q^2]$.
3. **Pre-Challenge Phase**: The adversary can make the following queries as many times as desired–
   - **Generate honest public key:**
     - Adversary submits an authority $u \notin \mathcal{C} \cup \mathcal{N}$.
     - Challenger computes $(\mathsf{ibe.mpk}_u, \mathsf{ibe.msk}_u) \leftarrow \mathsf{IBE.Setup}(1^\lambda, ([q^2] \times [q\lambda]))$, adding u to the $\mathcal{N}$ and $u \mapsto \mathsf{ibe.mpk}_u$ to D, and sends $\mathsf{ibe.mpk}_u$ to the adversary.
   - **Corrupt honest public key:**
     - Adversary submits an authority $u \in \mathcal{N}$.
     - Challenger sets $\mathcal{N} := \mathcal{N} - \{u\}$, $\mathcal{C} := \mathcal{C} \cup \{u\}$, and returns sk_u to the adversary.
   - **Generate corrupt public key:**
     - Adversary submits an authority $u \notin \mathcal{C} \cup \mathcal{N}$ and public key $\mathsf{ibe.mpk}_u$.
     - Challenger adds u to $\mathcal{C}$ and $u \mapsto \mathsf{ibe.mpk}_u$ to D.
   - **Secret key query:**
     - Adversary submits a (u, GID) where authority $u \in \mathcal{N}$ and $\mathsf{GID} \in \mathcal{GID}$.
     - For $i \in [q\lambda]$, challenger computes $\mathsf{ibe.sk}_{u,\mathsf{GID},i} \leftarrow \mathsf{IBE.KeyGen}(\mathsf{ibe.msk}_u, \mathsf{ID}_{\mathsf{GID},i})$.
     - Challenger returns $\{\mathsf{ibe.sk}_{u,\mathsf{GID},i}\}_{i\in[q\lambda]}$ to the adversary.
4. **Challenge Phase**:

- Adversary sends a pair of messages m_0, m_1, an access policy $(\mathbb{A}, \rho)$, and a list of authorities $U = \{u_1, \ldots, u_\ell\}$, where the $U \subseteq \mathcal{C} \cup \mathcal{N}$ and ℓ denotes the number of authorities in U.
- Challenger samples a random bit $b \leftarrow \{0, 1\}$ and generates $\mu_1, \mu_2, \ldots \mu_{q\lambda}$ as an additive secret sharing of m_b.
- For all $i \in [q\lambda]$ and $\mathsf{id} \in [q^2]$:
  - Compute shares $\{\mathsf{sh}_{i,\mathsf{id},j}\}_j \leftarrow \mathsf{Share}(1^\lambda, 1^\ell, \mathbb{A}, \rho, \mu_i)$.
  - For all $u \in U$, set $\mathsf{ct}_{u,i,\mathsf{id}} \leftarrow \mathsf{IBE.Enc}(\mathsf{ibe.mpk}_u, (i, \mathsf{id}), \{\mathsf{sh}_{i,\mathsf{id},k}\}_{k:\rho(k)=u})$.
- Challenger returns $\{\mathsf{ct}_{u,i,\mathsf{id}}\}_{u,i,\mathsf{id}}$ to the adversary.

5. **Post-Challenge Phase**: The adversary can make the same types of queries as it did before as long it does not violate the admissibility requirements.

6. Adversary outputs a bit b'.

Experiment 1: Since $\mathcal{A}$ is efficient, there exists some polynomial $p = p_{\mathcal{A}}(\lambda)$ which upper bounds the number of unique random oracle queries $\mathcal{A}$ makes. In this experiment, we simply substitute outputs of the explicit hash function with random values. That is, we rely on the random oracle heuristic.

1. The challenger initializes sets $\mathcal{C}, \mathcal{N} = \emptyset$, empty dictionaries D, H and counter $j' = 1$.

2. **Setup Phase:**

   - $\mathcal{A}$ sends 1^q to challenger, and gets oracle access to the hash function $\mathcal{H}$ modeled as a random oracle.
   - Challenger uniformly samples $\mathbf{R} \in [q^2]^{p \times (q\lambda)}$, where p denotes the maximum number of queries $\mathcal{A}$ makes. (We point out it is not essential for proving security that the reduction knows such an upper bound, but it enables a cleaner exposition, thus we stick with the above formalization.)
   - When $\mathcal{A}$ queries $\mathcal{H}$ on (GID, i), if $H[\mathsf{GID}] = \perp$ (i.e., GID is queried for the first time), it sets $H[\mathsf{GID}] = j'$ and increments $j' := j' + 1$. It returns $\mathbf{R}_{H[\mathsf{GID}],i}$.

   ... (rest of the game is as before.)

Experiment 2: This is same as previous game, except the challenger aborts in case the matrix $\mathbf{R}$ is such that its i^*-th column vector does not have unique entries for every unique GID queried by the adversary. Here i^* is randomly sampled at the beginning of the game.

In further detail, let $\mathbf{R}^*$ denote the subset of $\mathbf{R}$ such that $\mathbf{R}^*$ contains all those rows of $\mathbf{R}$ where the adversary makes at least one secret key query for the corresponding GID. Next, let $\mathbf{r}^* \in [q^2]^q$ denote the i^{*th} column vector of $\mathbf{R}^*$. Note that since the adversary can make at most q unique GID queries, thus the length of $\mathbf{r}^*$ is at most q (and we assume it to be exactly q without loss of generality). Now, in this game, we say the challenger aborts if $\mathbf{r}^*$ does not have unique entries. This can be checked as soon as the adversary makes q GID queries.

Experiment $3.j$ **for** $j \in [q^2]$**:** In this game, the challenger secret shares a random value $v_{i^*,\mathsf{id},1}$ instead of μ_{i^*} for all indices id $\leq j$.

4. **Challenge Phase:**

   ...

   - For $i \in [q\lambda]$ and id $\in [q^2]$:
     For every id $\in [q^2]$, let S_{id} denote the subset of the challenge authority set U such that, for each authority $u \in S_{\mathsf{id}}$, either $u \in \mathcal{C}$ (i.e., u has been corrupted), or the adversary made a secret key query to honest authority u for an identifier GID such that $\mathcal{H}(\mathsf{GID}, i^*) = \mathbf{R}[H[\mathsf{GID}], i^*] = \mathsf{id}$ (i.e., the adversary has an IBE secret key for identity (i^*, id) corresponding to authority u).
     In this hybrid, we are assuming a non-adaptive adversary for simplicity. That is, the adversary does not make any corruptions or key queries after challenge phase. However, by using non-committing encryption techniques, this can be easily solved, thus we avoid discussing it for simplicity.
     - If $i = i^*$ and id $\leq j$:
       * It runs the CSS simulator on $(1^\lambda, 1^\ell, \mathbb{A}, S_{\mathsf{id}})$ to obtain secret shares $\{\mathsf{sh}_{i^*,\mathsf{id},j}\}_{j:\rho(j)\in S_{\mathsf{id}}}$. That is, for each id, it generates fresh simulated CSS shares for all corrupted key slots.
       * Next, for all $u \in U$, it encrypts the shares as $\mathsf{ct}_{u,i^*,\mathsf{id}} \leftarrow \mathsf{IBE.Enc}(\mathsf{ibe.mpk}_u, (i^*, \mathsf{id}), \mathsf{sh}'_{i^*,\mathsf{id},j})$ where

$$\mathsf{sh}'_{i^*,\mathsf{id},j} = \begin{cases} \{\mathsf{sh}_{i^*,\mathsf{id},j}\}_{j:\rho(j)=u} & \text{if } u \in S_{\mathsf{id}}, \\ \mathbf{0} & \text{otherwise.} \end{cases}$$

     - Otherwise, compute IBE ciphertexts as before.
   - Challenger returns $\{\mathsf{ct}_{u,i,\mathsf{id}}\}_{u\in U, i\in[q\lambda], \mathsf{id}\in[q^2]}$ to the adversary.
     ... (rest of the game is as before.)

Experiment 4: In this game, the challenger no longer shares the challenger message, but encrypts a random message instead.

4. **Challenge Phase:**

   ...

   - Challenger samples $\mu_1, \mu_2, \ldots \mu_{q\lambda}$ as uniformly random elements
     ... (rest of the game is as before.)

Analysis. Let $\mathsf{Exp}_i(\mathcal{A})$ denote the advantage of $\mathcal{A}$ in experiment i. That is, $\Pr[\mathcal{A} \text{ wins Expt } i] - \frac{1}{2}$.

Lemma 6.9. For all PPT adversaries $\mathcal{A}$, $\mathsf{Exp}_0(\mathcal{A}) = \mathsf{Exp}_1(\mathcal{A})$ in the random oracle model.

Proof. Since $\mathbf{R}$ has independent uniformly random entries in $[q^2]$, the output of $\mathcal{H}$ will be independent and uniformly random on distinct inputs as well, so these experiments are identical. This relies on the random oracle heuristic. □

Lemma 6.10. For all PPT adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $\mathsf{Exp}_2(\mathcal{A}) \geq \frac{1}{q\lambda} \cdot (\mathsf{Exp}_1(\mathcal{A}) - \mathsf{negl}(\lambda))$.

Proof. From Lemma 6.8, $\mathbf{R}^*$ contains a column with unique entries with all but negligible probability. Since there are $q\lambda$ total columns, the probability a random i^* is such a column occurs with probability at least $\frac{1}{q\lambda}$. □

Lemma 6.11. For all PPT adversaries $\mathcal{A}$, $\mathsf{Exp}_2(\mathcal{A}) = \mathsf{Exp}_{3.0}(\mathcal{A})$.

Proof. Since $\mathsf{id} \in [q^2]$ is always > 0, the added conditional is never checked and these experiments proceed identically. □

Lemma 6.12. Assuming IBE is a secure identity-based encryption scheme and Π is a secure CSS scheme, for all PPT adversaries $\mathcal{A}$ and $j \in [q^2]$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $\mathsf{Exp}_{3.j-1}(\mathcal{A}) - \mathsf{Exp}_{3.j}(\mathcal{A}) = \mathsf{negl}(\lambda)$.

Proof. The proof follows from a sequence of subhybrids. However, before diving into the full proof, we would like to remind the reader that for simplicity we are assuming a non-adaptive adversary. That is, all corruptions happen in the pre-challenge phase. To handle adaptive corruptions, we can use random oracle based non-committing encryption trick to use each post-challenge secret key/authority corruption query as a mechanism to program appropriate secret share in the corresponding IBE ciphertext. Below we sketch the proof for a non-adaptive attacker.

To define these subhybrids, recall S_j as defined in the experiments $3.j$. Recall that whenever our challenger does not abort (starting from experiment 2), we know that if only exists a unique GID for each index j such that $\mathcal{H}(\mathsf{GID}, i^*) = \mathbf{R}[H[\mathsf{GID}], i^*] = j$. However, from the MA-ABE security game for $\mathcal{A}$ to win, we know the set of attributes from corresponding to users in S_j cannot satisfy the challenge access policy $\mathbb{A}$ for *any* GID. First, by using IBE security, we can switch each IBE ciphertext that cannot be decrypted by the adversary. Next, by CSS security, we can simulate the secret shares instead of computing them honestly.

Experiment 3.j-1.k for $k \in [0, \ell]$: In this game, the challenger replaces the secret shares for any IBE ciphertext it hasn't given a key for with an encryption of 0.

4. **Challenge Phase:**
   ...
   - For $i \in [q\lambda]$ and $\mathsf{id} \in [q^2]$:
     - Run as Experiment 3.j-1, except if $i = i^*$ and $\mathsf{id} = j$:
       * Compute shares $\{\mathsf{sh}_{i^*,\mathsf{id},j}\}_j \leftarrow \mathsf{Share}(1^\lambda, 1^\ell, \mathbb{A}, \rho, \mu_{i^*})$
       * Next, for all $u \in U$, it encrypts the shares as $\mathsf{ct}_{u,i^*,\mathsf{id}} \leftarrow \mathsf{IBE.Enc}(\mathsf{ibe.mpk}_u, (i^*, \mathsf{id}), \mathsf{sh}'_{i^*,\mathsf{id},j})$ where

$$\mathsf{sh}'_{i^*,\mathsf{id},j} = \begin{cases} \{\mathsf{sh}_{i^*,\mathsf{id},j}\}_{j:\rho(j)=u} & \text{if } u \in S_{\mathsf{id}} \cup \{u_{k+1}, \ldots u_\ell\}, \\ \mathbf{0} & \text{otherwise.} \end{cases}$$

  – Otherwise, compute IBE ciphertexts as before.

- Challenger returns $\{\mathsf{ct}_{u,i,\mathsf{id}}\}_{u\in U, i\in[q\lambda], \mathsf{id}\in[q^2]}$ to the adversary.
  ... (rest of the game is as before.)

Claim 6.13. For all adversaries $\mathcal{A}$, $\mathsf{Exp}_{3.j-1}(\mathcal{A}) = \mathsf{Exp}_{3.j-1.0}(\mathcal{A})$.

Proof. Since $u \in \{u_1, \dots u_\ell\} = U$, on index j the challenger will always encrypt CSS shares in $\mathsf{Exp}_{3.j-1.0}(\mathcal{A})$, just as in the real game. □

Claim 6.14. Assuming IBE is a secure identity-based encryption scheme for all PPT adversaries $\mathcal{A}$ and $j \in [q^2], k \in [\ell]$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $\mathsf{Exp}_{3.j-1.k-1}(\mathcal{A}) - \mathsf{Exp}_{3.j-1.k}(\mathcal{A}) = \mathsf{negl}(\lambda)$.

Proof. Observe that these hybrids are identical *except* when $u_k \notin S_{\mathsf{id}}$, where ciphertext $\mathsf{ct}_{u,i^*,\mathsf{id}}$ is switched to an encryption of 0 instead of an encryption of $\{\mathsf{sh}_{i^*,\mathsf{id},j}\}_{j:\rho(j)=u_k}$. Suppose there exists an adversary $\mathcal{A}$ that can distinguish these games with some non-negligible probability. We can define adversary $\mathcal{B}$ against the security of IBE as follows. Reduction $\mathcal{B}$ receives an IBE master public key $\mathsf{IBE.mpk}^*$, and acts as the role of the challenger for $\mathcal{A}$ in $\mathsf{Exp}_{3.j-1.k}(\mathcal{A})$. On a random **Generate honest public key** query, $\mathcal{B}$ will instead forward the challenge key $\mathsf{IBE.mpk}^*$ (which will be u_k with inverse polynomial probability). Now, $\mathcal{B}$ will answer any **Secret key** queries by simply using their KeyGen oracle. Finally, in the challenge phase $\mathcal{B}$ will forward $m_0 = \{\mathsf{sh}_{i^*,\mathsf{id},j}\}_{j:\rho(j)=u_k}$, $m_1 = 0$, and $\mathsf{id}^* = (i^*, j)$, and receive ct^*. It will then set $\mathsf{ct}_{u,i^*,\mathsf{id}} = \mathsf{ct}^*$ and continue simulating $\mathcal{A}$, forwarding the bit it returns. We can see that if ct^* is an encryption of m_0 this is $\mathsf{Exp}_{3.j-1.k-1}(\mathcal{A})$, and similarly for m_1, $\mathsf{Exp}_{3.j-1.k}$. Recall that since $u_k \notin S_{\mathsf{id}}$, $\mathcal{A}$ does not request a **Secret key** query on (i^*, j), and hence this is an admissible IBE adversary. □

Claim 6.15. Assuming Π is a secure CSS scheme, for all PPT adversaries $\mathcal{A}$ and $j \in [q^2]$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $\mathsf{Exp}_{3.j-1.\ell}(\mathcal{A}) - \mathsf{Exp}_{3.j}(\mathcal{A}) = \mathsf{negl}(\lambda)$.

Proof. Observe that since $|U| = \ell$, $\mathsf{Exp}_{3.j-1.\ell}$ and $\mathsf{Exp}_{3.j}$ generate IBE encryptions of 0 at the exact same values. Thus, the only difference is for the shares $\{\mathsf{sh}_{i^*,\mathsf{id},j}\}_{j:\rho(j)\in U}$, which are generated as $\mathsf{Share}(1^\lambda, 1^\ell, \mathbb{A}, \rho, \mu_{i^*})$ in $\mathsf{Exp}_{3.j-1.\ell}$ and as $\mathsf{Sim}(1^\lambda, 1^\ell, \mathbb{A}, S_{\mathsf{id}})$ in $\mathsf{Exp}_{3.j}$ Since $S_{\mathsf{id}} \notin \mathbb{A}$, these are exactly indistinguishable by CSS secrecy. □

Taking together Claim 6.13, Claim 6.14 and Claim 6.15 gives us our lemma.

□

Lemma 6.16. For all adversaries $\mathcal{A}$, $\mathsf{Exp}_{3.q^2}(\mathcal{A}) = \mathsf{Exp}_4(\mathcal{A})$.

Proof. Observe that by Experiment $3.q^2$, all shares for $i = i^*$ are simulated, thus the distribution received by the adversary is independent of μ_{i^*}. By the fact that $\{\mu_i\}_{i\in[q\lambda]}$ is an additive n-of-n secret sharing scheme, the distribution of $\{\mu_i\}_{i\in[q\lambda]\setminus i^*}$ is uniformly random. □

Lemma 6.17. For all adversaries $\mathcal{A}$, $\mathsf{Exp}_4(\mathcal{A}) = 0$.

Proof. The final distribution of encrypted messages are independent of the bit b. □

Taking together Lemmas 6.9 to 6.12, 6.16 and 6.17, we conclude that the adversary's advantage in Experiment 0 is negligible, and so our construction is a secure bounded collusion MA-ABE scheme. □

6.3 Achieving adaptive security

As disccused before, to handle adaptive corruptions, we can use standard non-committing encryption tricks in the bounded collusion setting [GVW12, GSW21, GGL22] to program each post-challenge key generation query appropriately. This allows the security proof to program simulated secret shares into keys (even after fixing the challenge ciphertext). In addition, unlike in the non-adaptive setting, the CSS simulator does not have complete knowledge of the challenge set S_{id}, as we receive adaptive queries one-by-one. Thus we modify our definition our CSS security to allow for adaptive specification of the challenge set S_{id} so long as it does satisfy the challenge policy. Fortunately, the construction by Yao (mentioned in [Bei11], see also Vinod et al. [VNS+03]) satisfies the adaptive security.

Definition 6.18 (CSS adaptive secrecy)**.** A CSS scheme for access structure $(\mathbb{A}, \rho)$ satisfies adaptive secrecy if there exists a polynomial-time simulator Sim such that for every supported access structure $(\mathbb{A}, \rho)$ and number of corresponding parties ℓ, every secret $s \in \{0,1\}^\lambda$, there exists a negligible function $\mathsf{negl}(\cdot)$, such that the advantage of an admissable adversary in the game below is $\mathsf{negl}(\lambda)$.

- The challenger samples a bit $b \in \{0,1\}$. It computes the real shares $\{\mathsf{sh}_i\}_i \leftarrow \mathsf{Share}(1^\lambda, 1^\ell, \mathbb{A}, \rho, s)$ and samples an empty state st for the simulator. It initializes the simulator state by running $\mathsf{Sim}(\mathsf{st}, 1^\lambda, 1^\ell, \mathbb{A}, \rho)$.

- Adversary can make multiple requests for shares for different parties as desired. It requests a share for party $u \in [\ell]$. If $b = 0$, the challenger returns $\{\mathsf{sh}_i\}_{i:\rho(i)=u}$. If $b = 1$, the adversary runs the simulator on its current state and current share and returns $\mathsf{Sim}(\mathsf{st}, u)$. It updates the simualtor state st.

- Adversary outputs a bit $b' \in \{0,1\}$.

Adversary's advantage is defined by, $|\Pr[b = b'] - 1/2|$. An adversary is considered admissable if the collection of the shares requested do not satisfy the access policy $(\mathbb{A}, \rho)$.

The construction by Yao (and [Bei11, VNS+03]) actually satisfies a stronger adaptive security notion because the simulator can initialize it's state by running the real algorithm on secret 0, i.e. $\{\mathsf{sh}_i\}_i \leftarrow \mathsf{Share}(1^\lambda, 1^\ell, \mathbb{A}, \rho, 0)$, where the simulator's state is $\rho, \{\mathsf{sh}_i\}_{i\in[n]}$. When querying $u \in [\ell]$, the simulator can respond by giving out $\{\mathsf{sh}_i\}_{i:\rho(i)=u}$. The adaptive notion is enough to showcase our result, hence we don't define the stronger notion. We do not present the fully formal proof in our writeup.

We follow the syntax of weak committing encryption (wNCE) from [GSW21] on message space $\{0,1\}$ and algorithms ($\mathsf{wNCE.Setup}, \mathsf{wNCE.Enc}, \mathsf{wNCE.Dec}, \mathsf{wNCE.SimSetup}, \mathsf{wNCE.SimOpen}$), which can be constructed from any public key encryption scheme. For considering bigger message space, [GSW21] show an extension of their construction in the Random Oracle Model [BR93]. The algorithms $\mathsf{wNCE.Setup}, \mathsf{wNCE.Enc}, \mathsf{wNCE.Dec}$ operate similar to any public key encryption scheme, while $\mathsf{wNCE.SimSetup}, \mathsf{wNCE.SimOpen}$ present an indistinguishable setup mode which produces ciphertexts which can be equivocated to any value determined at simulation-time. We use them to sketch our construction below. For ease of exposition, we use multiple copies of wNCE, but in practice, the same IBE scheme can be used to instantiate the multiple copies of wNCE.

$\mathsf{GSetup}(1^\lambda, 1^q) \to \mathsf{crs}$.

....

$\mathsf{ASetup}(\mathsf{crs}, u) \to (\mathsf{pk}_u, \mathsf{sk}_u)$.

- ...
- Let ℓ' be the length of the share when the CSS is initialized on security parameter λ, attribute universe of size ℓ[14].
  For each $i \in [q\lambda]$, $\mathsf{id} \in [q^2]$, $u \in U$, $k \in [\ell']$, sample a wNCE public and secret key, i.e. $(\mathsf{wNCE.pk}_{i,\mathsf{id},u,k}, \mathsf{wNCE.sk}_{i,\mathsf{id},u,k}) \leftarrow \mathsf{wNCE.Setup}(1^\lambda)$.
- Add the wNCE keys to the authorities public and secret keys respectively.

$\mathsf{KeyGen}(\mathsf{GID}, \mathsf{sk}_u) \to \mathsf{sk}_{\mathsf{GID},u}$.

- For $i \in [q\lambda]$, compute $\mathsf{ibe.sk}_{u,\mathsf{GID},i} \leftarrow \mathsf{IBE.KeyGen}(\mathsf{sk}_u, \mathsf{ID}_{\mathsf{GID},i})$ where $\mathsf{ID}_{\mathsf{GID},i} = (i, \mathcal{H}(\mathsf{GID}, i))$. For each $k \in [\ell']$, output $\mathsf{wNCE.sk}_{i,\mathcal{H}(\mathsf{GID},i),u,k}$.
- Output $\mathsf{sk}_{\mathsf{GID},u} = \{\mathsf{ibe.sk}_{u,\mathsf{GID},i}\}_{i\in[q\lambda]}, \{\mathsf{wNCE.sk}_{i,\mathcal{H}(\mathsf{GID},i),u,k}\}_{i\in[q\lambda],k\in[\ell']}$.

$\mathsf{Enc}((\mathbb{A}, \rho), \{\mathsf{pk}_u\}_{u\in U}, \mu) \to \mathsf{ct}$.

- Let $\mu_1, \mu_2, \dots \mu_{q\lambda}$ be an additive N-of-N secret sharing of μ. (That is, $\mu = \oplus_i \mu_i$.)
- For all $i \in [q\lambda]$ and $\mathsf{id} \in [q^2]$,
  - Compute shares $\{\mathsf{sh}_{i,\mathsf{id},j}\}_j \leftarrow \mathsf{Share}(1^\lambda, 1^{|U|}, \mathbb{A}, \rho, \mu_i)$
  - Compute encrypted shares, for each $k \in [\ell']$, for each $j \in [n]$, compute $\mathsf{shEnc}_{i,\mathsf{id},j,k} \leftarrow \mathsf{wNCE.Enc}(\mathsf{pk}_{i,\mathsf{id},u_{\rho(j)},k}, \mathsf{sh}_{i,\mathsf{id},j,k})$ where $\mathsf{sh}_{i,\mathsf{id},j,k}$ denotes the k^{th} bit of share $\mathsf{sh}_{i,\mathsf{id},j}$.
  - For all $u \in U$, compute $\mathsf{ct}_{u,i,\mathsf{id}} \leftarrow \mathsf{IBE.Enc}(\mathsf{pk}_u, (i, \mathsf{id}), \{\mathsf{shEnc}_{i,\mathsf{id},j,k}\}_{j:\rho(j)=u,k\in[\ell']})$.
- Output $\mathsf{ct} = ((\mathbb{A}, \rho), \{\mathsf{ct}_{u,i,\mathsf{id}}\}_{u,i,\mathsf{id}})$.

$\mathsf{Dec}(\mathsf{ct}, \{\mathsf{sk}_{\mathsf{GID},u}\}_{u\in S}) \to \mu$.

- Parse $\mathsf{ct} = ((\mathbb{A}, \rho), \{\mathsf{ct}_{u,i,\mathsf{id}}\}_{u,i,\mathsf{id}})$, and $\mathsf{sk}_{\mathsf{GID},u} = \{\mathsf{ibe.sk}_{u,\mathsf{GID},i}\}_i$.
- For each $i \in [q\lambda]$,
  - For each $u \in S$, recover $\{\mathsf{shEnc}_{i,\mathcal{H}(\mathsf{GID},i),j,k}\}_{j:\rho(j)=u,k\in[\ell']} = \mathsf{IBE.Dec}((\mathsf{sk}_{u,\mathsf{GID},i}, \mathsf{ct}_{u,i,\mathcal{H}(\mathsf{GID},i)})$.

[14]We are assuming that all shares are of the same length for ease of exposition.

  - For each $u \in S$, $k \in [\ell']$, for all j such that $\rho(j) = u$, recover

$$\mathsf{sh}_{i,\mathcal{H}(\mathsf{GID},i),j,k} \leftarrow \mathsf{wNCE.Dec}(\mathsf{wNCE.sk}_{i,\mathcal{H}(\mathsf{GID},i),u,k}, \mathsf{shEnc}_{i,\mathcal{H}(\mathsf{GID},i),j,k}),$$

    where $\mathsf{sh}_{i,\mathcal{H}(\mathsf{GID},i),j,k}$ denotes the k^{th} bit of share $\mathsf{sh}_{i,\mathcal{H}(\mathsf{GID},i),j}$.
  - Recover $\mu_i = \mathsf{Recon}(\{\mathsf{sh}_{i,\mathcal{H}(\mathsf{GID},i),j}\}_{j:\rho(j)\in S})$.
- Output $\mu = \oplus_{i\in[q\lambda]}\mu_i$.

At a high level, this construction achieves adaptive security by allowing the simulator to not output real shares during the challenge phase, rather only open the shares when adaptively querying. Informally, in the **Pre-Challenge Phase**, real wNCE secret keys are revealed. In the **Challenge Phase**, for each of the non-adaptively queried encryption shares, it outputs the IBE encryptions of wNCE encryptions of the simulated CSS shares. For all the other non-queried shares it outputs IBE encryptions of a ciphertext which have been setup using the SimSetup mode. During the **Post-Challenge Phase**, the simulator programs the wNCE secret keys to output the the simulated CSS shares using the SimOpen algorithm. By mode indistinguishability of wNCE, the simulated and real wNCE keys as well as ciphertexts are indistinguishable. Since this follows the same trick as [GVW12, GSW21, GGL22], we do not go into the details of the security proof. See [GSW21] on how to do this when using wNCE, and [GGL22] to see how to use IBE instead of PKE to bootstrap a non-adaptive construction to an adaptive one.

6.4 Making it Tagged and Handling Dynamic Collusion

Tagged MA-ABE. While the above MA-ABE construction is a statically secure scheme, it can be be made a tagged MA-ABE scheme quite easily. The central idea is to simply use the tag as an additional component of the identity space. Concretely, the encryption and key generation algorithms can be updated as follows.

$\mathsf{KeyGen}(\mathsf{tag}, \mathsf{GID}, \mathsf{sk}_u) \rightarrow \mathsf{sk}_{\mathsf{GID},u}$. A secret key consists of $q \cdot \lambda$ IBE keys computed as follows:

- For $i \in [q\lambda]$, compute $\mathsf{ibe.sk}_{u,\mathsf{tag},\mathsf{GID},i} \leftarrow \mathsf{IBE.KeyGen}(\mathsf{sk}_u, \mathsf{ID}_{\mathsf{tag},\mathsf{GID},i})$ where $\mathsf{ID}_{\mathsf{tag},\mathsf{GID},i} = (\mathsf{tag}, i, \mathcal{H}(\mathsf{GID}, i))$.
- Output $\mathsf{sk}_{\mathsf{tag},\mathsf{GID},u} = \{\mathsf{ibe.sk}_{u,\mathsf{tag},\mathsf{GID},i}\}_{i\in[q\lambda]}$.

$\mathsf{Enc}(\mathsf{tag}, (\mathbb{A}, \rho), \{\mathsf{pk}_u\}_{u\in U}, \mu) \rightarrow \mathsf{ct}$. It is identical to the encryption algorithm as before, except the IBE encryption algorithm specifies tag as an additional part of the identity. Below we highlight the main change.

- For all $i \in [q\lambda]$ and $\mathsf{id} \in [q^2]$,
  - For all $u \in U$, compute $\mathsf{ct}_{u,i,\mathsf{id}} \leftarrow \mathsf{IBE.Enc}(\mathsf{ibe.mpk}_u, (\mathsf{tag}, i, \mathsf{id}), \{\mathsf{sh}_{i,\mathsf{id},j}\}_{\rho(j)=u})$.

The correctness, efficiency, and proof of security are similar to that of the untagged scheme.

Fully Dynamically Bounded Collusion Secure. Finally, combining it with Theorem 4.6, we get our final result.

Theorem 6.19. If IBE is a secure IBE scheme and Π is a secure CSS scheme, then there exists a dynamically secure bounded-collusion MA-ABE scheme in the random oracle model.

References

[Agr17] Shweta Agrawal. Stronger security for reusable garbled circuits, general definitions and attacks. In *CRYPTO*, 2017.

[AGT21] Shweta Agrawal, Rishab Goyal, and Junichi Tomida. Multi-party functional encryption. In *Theory of Cryptography: 19th International Conference, TCC 2021, Raleigh, NC, USA, November 8–11, 2021, Proceedings, Part II*, pages 224–255. Springer, 2021.

[AGVW13] Shweta Agrawal, Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption: New perspectives and lower bounds. In *CRYPTO*, 2013.

[AKM$^+$22] Shweta Agrawal, Fuyuki Kitagawa, Anuja Modi, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Bounded functional encryption for turing machines: Adaptive security from general assumptions. In *Theory of Cryptography: 20th International Conference, TCC 2022, Chicago, IL, USA, November 7–10, 2022, Proceedings, Part I*, pages 618–647. Springer, 2022.

[Ale03] Michael Alekhnovich. More on average case vs approximation complexity. In *Foundations of Computer Science, 2003. Proceedings. 44th Annual IEEE Symposium on*, 2003.

[AMVY21] Shweta Agrawal, Monosij Maitra, Narasimha Sai Vempati, and Shota Yamada. Functional encryption for turing machines with dynamic bounded collusion from lwe. In *CRYPTO*, 2021.

[AR17] Shweta Agrawal and Alon Rosen. Functional encryption for bounded collusions, revisited. In *Theory of Cryptography Conference*, 2017.

[AS17] Shweta Agrawal and Ishaan Preet Singh. Reusable garbled deterministic finite automata from learning with errors. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *ICALP*, 2017.

[Att14] Nuttapong Attrapadung. Dual system encryption via doubly selective security: Framework, fully secure functional encryption for regular languages, and more. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, volume 8441 of *Lecture Notes in Computer Science*, pages 557–577. Springer, 2014.

[AV19] Prabhanjan Ananth and Vinod Vaikuntanathan. Optimal bounded-collusion secure functional encryption. In Dennis Hofheinz and Alon Rosen, editors, *TCC*, 2019.

[BDGM20] Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Factoring and pairings are not necessary for io: Circular-secure lwe suffices. *Cryptology ePrint Archive*, 2020.

[Bei11] Amos Beimel. Secret-sharing schemes: A survey. In *IWCC*, pages 11–46, 2011.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. In *CRYPTO*, 2001.

[BHR12] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In *CCS '12*, 2012.

[BLSV18] Zvika Brakerski, Alex Lombardi, Gil Segev, and Vinod Vaikuntanathan. Anonymous ibe, leakage resilience and circular security from new assumptions. In *EUROCRYPT*, 2018.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: definitions and challenges. In *TCC*, 2011.

[CC09] Melissa Chase and Sherman S. M. Chow. Improving privacy and security in multi-authority attribute-based encryption. In *ACM Conference on Computer and Communications Security*, pages 121–130, 2009.

[Cha07] Melissa Chase. Multi-authority attribute based encryption. In *TCC*, pages 515–534, 2007.

[Coc01] Clifford Cocks. An identity based encryption scheme based on Quadratic Residues. In *Cryptography and Coding, IMA International Conference*, volume 2260 of LNCS, pages 360–363, 2001.

[CVW$^+$18] Yilei Chen, Vinod Vaikuntanathan, Brent Waters, Hoeteck Wee, and Daniel Wichs. Traitor-tracing from lwe made simple and attribute-based. In *TCC*, 2018.

[DG17a] Nico Döttling and Sanjam Garg. From selective ibe to full ibe and selective hibe. TCC, 2017.

[DG17b] Nico Döttling and Sanjam Garg. Identity-based encryption from the diffie-hellman assumption. In *CRYPTO*, 2017.

[DGHM18] Nico Döttling, Sanjam Garg, Mohammad Hajiabadi, and Daniel Masny. New constructions of identity-based and key-dependent message secure encryption schemes. In

IACR International Workshop on Public Key Cryptography, pages 3–31. Springer, 2018.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography, 1976.

[DKW21] Pratish Datta, Ilan Komargodski, and Brent Waters. Decentralized multi-authority abe for dnfs from lwe. In *Advances in Cryptology–EUROCRYPT 2021: 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17–21, 2021, Proceedings, Part I*, pages 177–209. Springer, 2021.

[DKW23] Pratish Datta, Ilan Komargodski, and Brent Waters. Fully adaptive decentralized multi-authority abe. In *Advances in Cryptology–EUROCRYPT 2023: 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part III*, pages 447–478. Springer, 2023.

[DKXY02] Yevgeniy Dodis, Jonathan Katz, Shouhuai Xu, and Moti Yung. Key-insulated public key cryptosystems. In *International Conference on the Theory and Applications of Cryptographic Techniques*, 2002.

[DQV$^+$21] Lalita Devadas, Willy Quach, Vinod Vaikuntanathan, Hoeteck Wee, and Daniel Wichs. Succinct lwe sampling, random polynomials, and obfuscation. In *Theory of Cryptography: 19th International Conference, TCC 2021, Raleigh, NC, USA, November 8–11, 2021, Proceedings, Part II 19*, pages 256–287. Springer, 2021.

[Fre10] David Mandell Freeman. Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In *Advances in Cryptology–EUROCRYPT 2010: 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30–June 3, 2010. Proceedings 29*, pages 44–61. Springer, 2010.

[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 *FOCS*, 2013.

[GGL22] Rachit Garg, Rishab Goyal, and George Lu. A simple and generic approach to dynamic collusion model. Cryptology ePrint Archive, Paper 2022/330, 2022. https://eprint.iacr.org/2022/330.

[GGLW22] Rachit Garg, Rishab Goyal, George Lu, and Brent Waters. Dynamic collusion bounded functional encryption from identity-based encryption. In *EUROCRYPT*, 2022.

[GHM$^+$19] Sanjam Garg, Mohammad Hajiabadi, Mohammad Mahmoody, Ahmadreza Rahimi, and Sruthi Sekar. Registration-based encryption from standard assumptions. In *IACR international workshop on public key cryptography*, pages 63–93. Springer, 2019.

[GHMR18] Sanjam Garg, Mohammad Hajiabadi, Mohammad Mahmoody, and Ahmadreza Rahimi. Registration-based encryption: Removing private-key generator from IBE. In *TCC*, 2018.

[GKP$^+$13] Shafi Goldwasser, Yael Tauman Kalai, Raluca Ada Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. How to run turing machines on encrypted data. In *Annual Cryptology Conference*, pages 536–553. Springer, 2013.

[GKW16] Rishab Goyal, Venkata Koppula, and Brent Waters. Semi-adaptive security and bundling functionalities made generic and easy. In *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part II*, 2016.

[GKW18] Rishab Goyal, Venkata Koppula, and Brent Waters. Collusion resistant traitor tracing from learning with errors. In *STOC*, 2018.

[GLW12] Shafi Goldwasser, Allison Lewko, and David A Wilson. Bounded-collusion ibe from key homomorphism. In *Theory of Cryptography Conference*, 2012.

[GP21] Romain Gay and Rafael Pass. Indistinguishability obfuscation from circular security. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 736–749, 2021.

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, Ioctober 30 - November 3, 2006*, pages 89–98. ACM, 2006.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206, 2008.

[GSW21] Rishab Goyal, Ridwan Syed, and Brent Waters. Bounded collusion abe for tms from ibe. In *ASIACRYPT*, 2021.

[GV20] Rishab Goyal and Satyanarayana Vusirikala. Verifiable registration-based encryption. In *CRYPTO*, 2020.

[GVW12] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In *CRYPTO*, 2012.

[ISV$^+$17] Gene Itkis, Emily Shen, Mayank Varia, David Wilson, and Arkady Yerukhimovich. Bounded-collusion attribute-based encryption from minimal assumptions. In *PKC*, 2017.

[JLS21] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In *STOC*, 2021.

[JLS22] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from lpn over f p, dlin, and prgs in nc 0. In *Advances in Cryptology–EUROCRYPT 2022: 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Trondheim, Norway, May 30–June 3, 2022, Proceedings, Part I*, pages 670–699. Springer, 2022.

[KW19] Venkata Koppula and Brent Waters. Realizing chosen ciphertext security generically in attribute-based encryption and predicate encryption. In *CRYPTO 2019*, 2019.

[Lew12] Allison Lewko. Tools for simulating features of composite order bilinear groups in the prime order setting. In *Advances in Cryptology–EUROCRYPT 2012: 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings 31*, pages 318–335. Springer, 2012.

[LW11] Allison B. Lewko and Brent Waters. Decentralizing attribute-based encryption. In *EUROCRYPT*, pages 568–588, 2011.

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *STOC*, pages 427–437, 1990.

[OT10] Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In *CRYPTO*, pages 191–208, 2010.

[OT20] Tatsuaki Okamoto and Katsuyuki Takashima. Decentralized attribute-based encryption and signatures. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 103(1):41–73, 2020.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, 2005.

[Sha84] Adi Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO*, 1984.

[SS10] Amit Sahai and Hakan Seyalioglu. Worry-free encryption: functional encryption with public keys. In *CCS*, 2010.

[SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, pages 457–473, 2005.

[VNS$^+$03] V Vinod, Arvind Narayanan, K Srinathan, C Pandu Rangan, and Kwangjo Kim. On the power of computational secret sharing. In *Progress in Cryptology-INDOCRYPT 2003: 4th International Conference on Cryptology in India, New Delhi, India, December 8-10, 2003. Proceedings 4*, pages 162–176. Springer, 2003.

[Wee21] Hoeteck Wee. Abe for dfa from lwe against bounded collusions, revisited. In *TCC*, 2021.

[WFL19] Zhedong Wang, Xiong Fan, and Feng-Hao Liu. Fe for inner products and its application to decentralized abe. In *IACR international workshop on public key cryptography*, pages 97–127. Springer, 2019.

[WW21] Hoeteck Wee and Daniel Wichs. Candidate obfuscation via oblivious lwe sampling. In *Advances in Cryptology–EUROCRYPT 2021: 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17–21, 2021, Proceedings, Part III*, pages 127–156. Springer, 2021.

[Yao82] Andrew C Yao. Protocols for secure computations. In *FOCS*, 1982.

[Yao86] Andrew Yao. How to generate and exchange secrets. In *FOCS*, 1986.