Multi-Authority Functional Encryption: Corrupt Authorities, Dynamic Collusion, Lower Bounds, and More

Rishab Goyal
UW-Madison*

Saikumar Yadugiri
UW-Madison†

Abstract

Decentralization is a great enabler for adoption of modern cryptography in real-world systems. Widespread adoption of blockchains and secure multi-party computation protocols are perfect evidentiary examples for dramatic rise in deployment of decentralized cryptographic systems. Much of cryptographic research can be viewed as reducing (or eliminating) the dependence on trusted parties, while shielding from stronger adversarial threats. In this work, we study the problem of multi-authority functional encryption (MAFE), a popular decentralized generalization of functional encryption (FE). Our main contributions are:

1. We design MAFE for all poly-sized circuits, in the bounded collusion model, under the minimal assumption of PKE/OWFs. Prior to our work, this required either sub-exponentially secure obfuscation, or $\log n$-party key exchange, or Random Oracles and sub-exponentially secure PKE. We also extend our constructions to the *dynamic* collusion model under the minimal assumptions of IBE/OWFs. Unlike all prior works, our MAFE systems are truly dynamic and put no restrictions on the maximum number of authorities.
2. Under the hardness of learning with errors (LWE) assumption, we design MAFE for all poly-sized circuits where we allow adversaries to adaptively corrupt local authorities. We allow an adversary to corrupt any k out of n local authorities as long as $\binom{n}{k} = \mathsf{poly}(\lambda)$. Prior to this, such MAFE relied on sub-exponentially secure obfuscation. Additionally, we design a new MAFE compiler for boosting selective authority corruptions to non-adaptive authority corruptions.
3. We prove a tight implication from MAFE to (VBB/indistinguishability) obfuscation. We show that MAFE implies obfuscation only if the number of attribute bits (jointly) controlled by all corrupt local authorities is $\omega(\log \lambda)$. This proves optimality of our second result for a wide range of parameters.
4. Finally, we propose a new MAFE system that we refer to as multi-authority attribute-based functional encryption (MA-ABFE). We view it as an approach to get best of both worlds (fully collusion resistant MA-ABE, and bounded collusion resistant MAFE). By combining our results with prior MA-ABE results, we obtain MA-ABFE for $\mathsf{NC}^1 \circ \mathsf{P/Poly}$ from standard pairing-based assumptions, and for $\mathsf{DNF} \circ \mathsf{P/Poly}$ from LWE, both in the Random Oracle Model. We also describe a simple construction of MA-ABE for general predicates from witness encryption, and combining with known results, we also get MA-ABFE for $\mathsf{P/Poly} \circ \mathsf{P/Poly}$ from *evasive* LWE.

*Email: `rishab@cs.wisc.edu`. Support for this research was provided by OVCRGE at UW–Madison with funding from the Wisconsin Alumni Research Foundation.

†Email: `saikumar@cs.wisc.edu`.

Contents

1 Introduction

In standard FE [SW05, BSW11], there is a single trusted authority that generates the global public parameters pp and a master (secret) key msk. The trusted authority is responsible for securely storing msk as well as generating and distributing *functional* secret keys to authorized users. Each functional key, sk_x, is associated with an attribute x. And, given pp, one can encrypt any efficiently computable predicate ϕ, such that the resulting ciphertext can be decrypted using sk_x to learn $\phi(x)$, and nothing else. FE pushes the envelope of public-key encryption significantly by upgrading ciphertexts from static (all-or-nothing) objects to dynamic objects providing fine-grained control over encrypted data.

Key Escrow and Single Point-of-Failure. Unfortunately, the FE advancement comes at the cost of *establishing a central point of trust*! FE assumes a *single trusted authority* that has a master key msk for the entire system throughout its lifetime. While this is essential for supporting fine-grained access control, it is quite troubling for practical applications since this trust model is too strong. The authority could intercept and decrypt any ciphertext ct using msk. Moreover, it becomes a single point-of-failure. If the authority gets corrupted, then all security is lost! This was beautifully explained in Rogaway's essay [Rog15]. It highlights that FE implicitly embeds such key escrow problems at the cost of promising powerful functionality.

In light of these limitations, it is very important to study FE in decentralized settings reducing the dependence on a central trusted party. Over the last three decades, the problem of decentralizing FE has received a lot of attention. Numerous approaches have been studied to weaken the trust model [BF01, CHSS02, ARP03, LBD+04, CCV04, Goy07, PS08, GHMR18, GHM+19, GV20] in FE and its various specializations [Sha85, Coc01, BF01, GPSW06]. One very popular and well-adopted approach is colloquially referred as the *multi-authority* model [Cha07, CC09, LW11, BCG+17b].

Multi-Authority FE (MAFE) is an established generalization of traditional FE. The goal is to decentralize trust from a single central authority to a group of (non-interactive and mistrusting) local authorities. Each local authority samples its own public-secret key pair $(\mathsf{mpk}_u, \mathsf{msk}_u)$, publishing mpk_u publicly. The point is each local authority has control over only a small subspace of attributes. A sender can select any subset of authorities, U, under which it wants to encrypt its predicate ϕ. The sender only needs $\{\mathsf{mpk}_u\}_{u \in U}$ for those authorities.

Unlike standard FE, the key generation process is fully decentralized. There is no interaction between the local authorities, nor they are aware of each other's existence! Each receiver (annotated with a unique global public identifier[1] GID) can request a partial functional key from an authority u for the portion of the attribute space controlled by u. E.g., Department of Motor Vehicles (DMV) can act as a local authority, and it will only control the attribute bit(s) associated with a user's driving license information. Thus, user GID can ask authority u for a partial functional key for its local attribute x_u, and gets a key $\mathsf{sk}_{u,x_u,\mathsf{GID}}$ annotated with its GID and associated with its authorized attribute. To decrypt, a user combines all partial functional keys it has, $\{\mathsf{sk}_{u,x_u,\mathsf{GID}}\}_u$, and use them as its joint functional key.

Clearly, MAFE has all advantages of FE (such as fine-grained access control) while simultaneously relaxing the underlying trust assumption. It offers significant robustness by resisting attackers that can fully corrupt some of the local authorities. This makes it readily more suitable for applications, since there is no single point-of-failure and does not suffer from key escrow.

[1]As we discuss later, usage of GIDs is a common and well-known approach to prevent mix-and-match attacks in the multi-authority setting.

Current landscape for available constructions of MAFE is unfortunately very limited. We either can achieve all-or-nothing functionality[2] [Cha07, CC09, LW11, MJ18, Kim19, WFL19, OT20, DKW21, AG21, AGT21, DKW23, DP23, GGL24], or rely on very heavy hammers like *sub-exponentially* secure indistinguishability obfuscation (iO) [BCG+17b]. A recent work by Goyal-Yadugiri [GY24] made interesting progress by developing new templates for designing MAFE, but their schemes are insecure even if a single local authority gets compromised, and require sub-exponential hardness assumptions (for any $\omega(1)$ number of authorities). This highlights major gaps and deficiencies in our understanding of MAFE systems.

Our contributions. We design multiple new MAFE systems with varying functionality and security. Most of our MAFE schemes are designed in the standard model, and rely on polynomial hardness of standard cryptographic assumptions. We improve prior lower bounds tightening implications between MAFE and obfuscation [BGI+12]. Overall, our work addresses many existing gaps in the MAFE literature, and builds a better toolkit for multi-authority encryption systems. Our main contributions can be summarized as follows:

1. We design MAFE for all poly-sized circuits, in the bounded collusion model [DKXY02, SS10, GLW12, GVW12], under the minimal assumption of public-key encryption. Our construction extends to the secret-key setting, where our secret-key MAFE construction only relies on the minimal assumption of one-way functions. Prior to our work, MAFE for all poly-sized circuits required either sub-exponentially secure iO [BCG+17b], or $\log n$-party key exchange for n-authority MAFE [GY24], or Random Oracles and sub-exponentially secure PKE [GY24].

   We also design MAFE for all circuits in the *dynamic* collusion model [AMVY21, GGLW22] (i.e., collusion bound is dynamically chosen by an encryptor, and not fixed during setup) under the minimal assumption of identity-based encryption and one-way functions (for secret-key setting). Moreover, we do not put a limit on the number of local authorities, or require any global coordination in any of our MAFE systems. Each local authority can join and generate partial functional keys independent of others. All prior MAFE constructions [BCG+17b, GY24] required an a-priori bound on the number of authorities, and [GY24] also required all authorities to have finished system setup before any functional keys had to be generated.

2. Under the hardness of learning with errors (LWE) assumption, we design MAFE for all poly-sized circuits where we allow adversaries to adaptively corrupt local authorities. We allow an adversary to corrupt any k out of n local authorities as long as $\binom{n}{k} = \mathsf{poly}(\lambda)$, and treat corruptions for the rest of $(n-k)$ authorities as in the dynamic collusion model. Prior to this, the only known MAFE construction providing such security relied on sub-exponentially secure iO [BCG+17b].

   As an additional contribution, we design a new compiler for MAFE, where we show that any MAFE scheme where the attacker declares the set of corrupted authorities *selectively* can be boosted to an MAFE scheme where the attacker can declare these non-adaptively (i.e., before the challenge phase).

3. Brakerski et al. [BCG+17b] showed that any MAFE scheme, that is secure even against a single local authority corruption, implies a (VBB/ indistinguishability) obfuscation scheme [BGI+12]. We prove a tighter formulation of this implication, giving us a better lower bound for MAFE.

[2]By this we mean multi-authority attribute-based encryption and its inner-product extensions.

We show that as long the number of attribute bits (jointly) controlled by all corrupt local authorities is bounded as $O(\log \lambda)$, then it does not imply obfuscation. That is, an MAFE scheme which is secure in presence of $\omega(\log \lambda)$ "corrupted" attribute bits implies (VBB/ indistinguishability) obfuscation. (This proves optimality of our second result for a wide range of parameters.)

4. Finally, we propose a new MAFE system that we refer to as multi-authority attribute-based functional encryption (MA-ABFE). As we discuss later in the overview, this is a non-trivial composition[3] of MA-ABE and MAFE. The goal of the above formulation is to design an MA-ABFE schemes that allows an *unbounded* number of *unsatisfying key queries* and *bounded* number of *satisfying key queries*. We view it as an approach to get best of both worlds (fully collusion resistant MA-ABE, and bounded collusion resistant MAFE). By combining our results with prior MA-ABE constructions [LW11, DKW21, DKW23], we obtain MA-ABFE for $\mathsf{NC}^1 \circ \mathsf{P/Poly}$ from standard pairing-based assumptions, and for $\mathsf{DNF} \circ \mathsf{P/Poly}$ from LWE, both in the Random Oracle Model.

   We also describe a simple construction of MA-ABE for general predicates from witness encryption [GGSW13] without relying on Random Oracles. The only prior work in MA-ABE that does not rely on Random Oracles was by Waters et al. [WWW22], but they could only support the class of subset predicates. We show how to support arbitrary polynomial sized circuits. Instantiating our new MA-ABE with known results [VWW22], we also get MA-ABFE for $\mathsf{P/Poly} \circ \mathsf{P/Poly}$ from *evasive* LWE assumption [Wee22, Tsa22].

Reference	**Authorities**	**Security**	**Corruptions**	**Assumption**	**ROM**
[BCG+17b]	$\mathsf{poly}(\lambda)$	Full	✓	Sub-exp iO + OWFs	No
[GY24]	$O(1)$	Static	✗	(B)DDH	No
[GY24]	$\mathsf{poly}(\lambda)$	Static	✗	Sub-exp PKE	Yes
Theorem 5.3	$\mathsf{poly}(\lambda)$	Static	✗	PKE	No
Theorem 6.5	$\mathsf{poly}(\lambda)$	Dynamic	✗	IBE	No
Theorem 7.4	$\mathsf{poly}(\lambda)$	Dynamic	✓	LWE	No

Table 1: Comparison of our results with prior works on MAFE for $\mathsf{P/Poly}$ circuits with adaptive security. In the table **Security** column refers to the collusion-resistance property. "Fixed" refers to an a-priori bound on partial functional key corruptions set for the whole system. "Dynamic" refers to the bound set for each ciphertext [AMVY21, GGLW22].

Related work. MAFE was proposed nearly a decade ago as generalization of FE by [BCG+17b]. Despite this, the only known constructions for $\mathsf{P/Poly}$ circuits are [BCG+17b, GY24]. The former provided construction of indistinguishability-based MAFE with unbounded collusions where *more than one* secret key per GID can be issued using sub-exponential iO and injective one-way functions. Moreover, this construction can handle arbitrary authority corruptions. [BCG+17b] also provided relations between MAFE and iO/VBB regardless of whether MAFE allows authority corruptions. [GY24] identified that these implications aren't tight in case of MAFE without authority corruptions and that issuing more than one secret key per GID goes against the motivation to use GIDs. [GY24]

[3]A trivial composition would be to first create an MAFE ciphertext, and then encrypt it under MA-ABE. Unfortunately, this is totally insecure as we explain later.

Reference	**Assumption**	$\mathcal{P} \circ \mathcal{C}$	**Query Bound**	**Security**	**ROM**
[AGT21]	Pairings	$\mathsf{LSSS} \circ \mathsf{IP}$	$(\mathsf{poly}, \mathsf{poly})$	Selective	Yes
[DP23]	BDDH	$\mathsf{LSSS} \circ \mathsf{IP}$	$(\mathsf{poly}, \mathsf{poly})$	Static	Yes
Theorem K.6, Corollary 13.3	Evasive LWE	$\mathsf{P/Poly} \circ \mathsf{P/Poly}$	(poly, Q)	Static	No
[DKW23], Theorem 13.2	Sub-exp pairings	$\mathsf{NC}^1 \circ \mathsf{P/Poly}$	(poly, Q)	Adaptive	Yes
[DKW21], Corollary 13.3	LWE	$\mathsf{DNF} \circ \mathsf{P/Poly}$	(poly, Q)	Static	Yes

Table 2: Comparison of our results with prior works on MA-ABFE. Here, $\mathcal{P}$ is the policy class (for ABE part) and $\mathcal{C}$ is the circuit class (for FE part). LSSS is the policy class for linear secret sharing schemes and IP is inner-product functionality. By (poly, Q) we mean unbounded "unsatisfying" queries and Q "satisfying" queries.

provided constructions where a bounded number of GID queries are allowed with static security from (plain) PKE, adaptive security for $\mathsf{poly}(\lambda)$ authorities in ROM [BR93] from sub-exponential PKE and for $O(1)$ authorities from 2/3-party key exchange. All constructions of [GY24] satisfy simulation security.

MA-ABE [Cha07, CC09] is an all-or-nothing specialization of MAFE, where if the policy is satisfied, then one learns the full message. [LW11] constructed decentralized ABE for monotone span programs using paring-based assumptions. Both multi-authority and decentralized ABE saw multitude of results in the past decade (cf. [MJ18, Kim19, WFL19, OT20, AG21] and the references there in). Recently, [DKW23] constructed an MA-ABE scheme for NC^1 policies from computational pairing-based assumptions and [DKW21] constructed MA-ABE for DNF formulae from LWE. Both these works relied on the Random Oracle Model. The only known construction for MA-ABE that does not rely on ROM is by Waters et al. [WWW22], and could support subset predicate under the hardness of evasive LWE. In the bounded collusion setting, [WFL19, GGL24] construct MA-ABE schemes for monotone boolean formulae and circuits respectively from nearly minimal assumptions.

2 Technical Overview

In this section, we provide a high-level overview of our main ideas and techniques. We start by recalling the definition of MAFE.

Reviewing MAFE. An MAFE scheme contains four main algorithms $(\mathsf{ASetup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$. The authority setup algorithm ASetup is used by each authority to sample master public-secret key pair $\mathsf{MPK}, \mathsf{MSK}$ on their own (i.e., without any interaction or synchronization). The key generation algorithm KGen algorithm is used by each authority to compute a partial secret key for a portion of the input x_i using their MSK_i. Enc and Dec are defined similar to FE, where the master public key consists of $\{\mathsf{MPK}_i\}_i$ from all authorities, and the secret key for input $X = (x_i)_i$ is $\{\mathsf{SK}_{i,x_i}\}_i$. To avoid mix-and-match attacks (by combining keys for two users from two different authorities), every partial secret key is embedded with a global user identifier GID. We refer the reader to prior works [LW11, GY24] to get a better understanding of GID. For this high level overview, we mostly ignore GIDs while explaining our main ideas.

To define the security of MAFE, we consider two types of corruptions – (a) secret key corruptions, (b) authority corruptions. A type-(a) attacker can only query for secret keys for various inputs from each authority, but not corrupt their master keys. Type-(a+b) is a stronger attacker, who can additionally corrupt MSK_i of any authority of its choice. In this work, we work in the bounded collusion model [DKXY02, SS10, GLW12], where the attacker can make a fixed number (say Q) of type (a) corruption queries to each authority. In addition, we also consider stronger corruptions, where an attacker can adaptively corrupt any authority's MSK_i as well. That is, we consider both type (a) and (a+b) adversaries.

In Section 2.1, we consider only type-(a) adversaries, and later in Section 2.2, we expand our designs to resist type-(a+b) adversaries as well. Finally, in Section 2.3, we also expand our techniques and combine them with fully-collusion-resistant (multi-authority) attribute-based encryption systems to handle an unbounded number of unsatisfying predicate queries, and a bounded number of satisfying predicate queries.

2.1 MAFE from Minimal Assumptions

Our first goal is to design an MAFE scheme from public-key encryption (PKE). Prior works [BCG+17b, GY24] require far stronger assumptions, or rely on random oracle heuristics (see Table 1). Our core technique is a new compiler for composing n *single-authority* FE systems with a single n-authority MAFE system with *very weak security.* We design such a weak n-authority MAFE from PKE, and plugging this in our new compiler, we design an n-authority MAFE with standard security. Let us describe our new *weak* MAFE notion.

MAFE with trusted setup. The main technical tool we use is a new notion of MAFE which we call *MAFE with trusted setup* (tMAFE). Given the goal of MAFE is to remove the need for a central trusted party, it seems rather meaningless to even define the notion of MAFE with trusted setup. Regardless, as we show later, tMAFE is surprisingly the right abstraction for designing standard decentralized MAFE. Let us first expand on the notion of tMAFE.

In tMAFE, there is a *central trusted authority* that generates master public-secret keys for all authorities and distributes it to every authority. Thus, individual authorities do **not** sample any master public-secret keys. Essentially, instead of the ASetup algorithm, tMAFE has a global setup GSetup algorithm to generate $\{(\mathsf{MPK}_i, \mathsf{MSK}_i)\}_i$ for all authorities. Other than this, tMAFE is defined as its standard (decentralized) counterpart. That is, each authority generates partial secret keys in a decentralized and asynchronous manner as before. For security, we consider a nearly identical game, except the challenger generates all authority keys by running GSetup instead of ASetup n times. Next, we show how to generically bootstrap tMAFE to a standard (non-trusted) MAFE scheme.

tMAFE $\Longrightarrow$ MAFE. Note that tMAFE already captures the desired functionality that we want from general MAFE, except the authorities need to get together to sample their individual master public-secret keys. Our idea is to delegate the computation of tMAFE's global setup to encryption, by relying on another FE layer to handle the trusted setup problem. Suppose we have a single-authority FE scheme, let us call it 1-FE. Our plan is to use 1-FE to generate n independent 1-FE systems, one for each individual authority, and somehow use these independent systems to unlock the tMAFE secret keys. Let us explain further.

In our MAFE systems, we ask the encryptor to run the global setup, GSetup, for tMAFE. This gives the encryptor a set of n master public-secret key pairs for tMAFE. Clearly, it can encrypt the predicate circuit (C) using tMAFE encryption. But, the question is *how to share the tMAFE authority keys with the "real" authorities*, or rather how to ensure an authorized user gets the right tMAFE key without talking to the encryptor? Our insight is that this can be achieved by delegating the tMAFE key generation. That is, we encrypt $\mathsf{tMAFE.KGen}$ circuit, with i-th tMAFE master key hardwired, under the i-th 1-FE system. Concretely, each authority samples 1-FE keys $(\mathsf{1\text{-}fe.mpk}_i, \mathsf{1\text{-}fe.msk}_i)$, and uses $\mathsf{1\text{-}fe.msk}_i$ to answer secret key queries as $\mathsf{SK}_{i,x_i} \leftarrow \mathsf{1\text{-}FE.KGen}(\mathsf{1\text{-}fe.msk}_i, x_i)$. During encryption, we run a fresh GSetup for tMAFE to generate n authority key pairs, $\{(\mathsf{MPK}_i, \mathsf{MSK}_i)\}_i$. Given $\{\mathsf{MPK}_i\}_i$, encrypt C as $\mathsf{tMAFE.Enc}(C)$. Moreover, delegate tMAFE key generation by creating n 1-FE ciphertexts as $\mathsf{1\text{-}FE.Enc}(\mathsf{1\text{-}fe.mpk}_i, F_i)$, where $F_i = \mathsf{tMAFE.KGen}(\mathsf{MSK}_i, \cdot)$. The resulting 1-FE ciphertexts $\{\mathsf{1\text{-}fe.ct}_i\}_i$ and tMAFE ciphertext $\mathsf{tMAFE.ct}$ are jointly set as the full MAFE ciphertext. Clearly, decryption is done in two stages– (i) decrypt $\{\mathsf{1\text{-}fe.ct}_i\}_i$ using $\{\mathsf{SK}_{i,x_i}\}_i$ (resp.) to compute $\mathsf{tMAFE.SK}_{i,x_i}$, and (ii) then decrypt $\mathsf{tMAFE.ct}$ using these to recover $C(x_1, \ldots, x_n)$.

The security of the above template can be quite readily reduced to 1-FE and tMAFE security. The intuition is that as long as none of authorities are corrupted, we can argue that each 1-FE ciphertext only reveals the tMAFE keys for authorized inputs. Once this happens, we can proceed to show the $\mathsf{tMAFE.ct}$ hides C beyond what can be learned by running decryption honestly. Our compiler is described in detail later in Section 5. We highlight that we do not need any synchronization among the authorities, they are even oblivious to the existence of each other. Thus, we do not require an a-priori bound on the number of authorities, and neither we need to pass any public/secret key between authorities before key generation. Thus, our resulting MAFE scheme supports fully dynamic authorities, unlike prior works [BCG+17b, GY24]. Looking ahead, this simplicity and modularity of our template makes it readily extendable to several other corruption models like the dynamic collusion model [AMVY21, GGLW22, GGL24], full authority corruptions [LW11, BCG+17b]. Next, we talk about constructing tMAFE, and later circle back to other improvements of our base template.

Constructing tMAFE. Since we already have constructions for 1-FE for poly-size circuits from minimal assumptions (public/symmetric-key encryption) [GVW12, AV19], thus to instantiate our compiler from minimal assumptions, we just need to design tMAFE under same set of assumptions.

A natural question the reader might have is: *isn't tMAFE already pretty close to a single-authority FE (1-FE) system?* Basically, since there is a trusted setup, then what prevents us from sampling just a 1-FE system during the global setup, GSetup, and later thresholdizing by relying on some threshold homomorphic encryption techniques [BGG+18]. While this will not give us the desired result from minimal assumptions, it does suggest that designing tMAFE might not be so hard after all. Although this is a good first idea, looking a bit carefully shows why such an approach will be flawed.

Recall that a major feature of (t)MAFE is that each authority only generates a partial key for *its portion of the attribute*. That is, authority i only gets to see x_i, and not the full input $(x_1, \ldots)$. Therefore, any approach to just use threshold FHE (or even 2-round MPC-techniques) might just fail! Because, in such approaches, we either need all parties to know the full input, or there needs to be another round of communication. Both of these are not allowed in the multi-authority model. Therefore, we really have to design tMAFE from ground up.

Our strategy is to open up known constructions for 1-FE/MAFE [SS10, GVW12, AV19, GY24], and generalize their frameworks to the trusted setup model, while relying only on minimal assumptions. We start with the beautiful 1-FE construction by Gorbunov-Vaikuntanathan-Wee (GVW) [GVW12]. In a few words, the reason GVW is not already a multi-authority system is because: (i) neither their key generation is distributed, (ii) nor their security reductions can handle attackers that can learn partial secret keys adaptively. Such issues were recently explored by

Goyal-Yadugiri [GY24], who designed (standard) MAFE from simple assumptions (see Table 1). While [GY24] were able to resolve (i) pretty easily, the second issue turned out a lot more challenging. To which, they relied on stronger assumptions like n-party non-interactive key exchange (NIKE), or sub-exponential security and random oracle (RO) heuristics.

Our insight is that if we work in the *trusted* setting, then we could get around issue (ii). Abstractly, the core missing piece that prohibits known 1-FE schemes [GVW12, AV19, GY24] to become multi-authority is we need each authority to take certain (randomized) choices to answer a partial key query for a particular GID *consistently*. Without an extra round of interaction between authorities, this is hard to solve. (This is why [GY24] used NIKE.) Our observation is that, in tMAFE, we have a *global* setup option. Thus, we can fix this issue by secretly selecting a random function to synchronize between all n authorities. Basically, this function would deterministically provide consistent choices to every authority. In the main body, we show that by selecting a vanilla pseudo-random function during GSetup, we can set the function as $F_K(\cdot)$, where K is the globally shared PRF key. Since we only consider type-(a) adversaries, K is never corrupted, and we can use pseudorandomness security of F to prove security of resulting tMAFE. Our tMAFE system directly builds on proof techniques from prior works [GVW12, AV19, GY24], and we provide more details in Section 4.

We remark that our tMAFE construction easily generalizes to the secret-key setting as well. Therefore, we are able to design a tMAFE scheme in the public-key setting under the minimal assumption of PKE, while in the secret-key setting under the minimal assumption of one-way functions. Therefore, combining the above with our base (tMAFE $\implies$ MAFE) template, we can design MAFE for all poly-size circuits secure against all type-(a) adversaries in the bounded collusion model. We also can extend our secret-key MAFE scheme to be function hiding by relying on prior compilers [BS18].

Handling dynamic collusions in MAFE. As a natural progression, we improve our base MAFE template to resist attackers in the dynamic collusion model [AMVY21, GGLW22]. The dynamic collusion model is a non-trivial strengthening of the (static) bounded collusion model, where the collusion bound is selected by an encryptor, and not fixed globally after system setup. That is, the bound Q is declared during Enc, and as such the $\mathsf{ASetup}, \mathsf{KGen}$ algorithms are independent of Q. This allows a more fine-grained control over collusion security for each ciphertext, and we refer to [AMVY21, GGLW22, GGL24] for a detailed discussion on more benefits.

Let us reuse our base template! Our intuition is that if we use a 1-FE secure in the dynamic collusion model, then our original MAFE template 'almost' already gives us an MAFE with desired security. The reason is that a dynamic model 1-FE already guarantees that its setup and key generation are independent of any collusion bound, thus ASetup and KGen for our resulting MAFE will also be independent of the collusion bound. All that is left to check is that the resulting encryption algorithm does support dynamic collusions. And, this also seems relatively straightforward since the encryptor really samples a fresh tMAFE system, thus in the dynamic model, the encryptor can sample the tMAFE for appropriate level of collusion security. Overall, this seems to suggest that we can quite easily extend our MAFE to be secure in the dynamic collusion model.

Unfortunately, there is an intricate bug in the above approach. The reason is that the size of a tMAFE partial key, $|\mathsf{tMAFE.SK}_{i,x_i}|$ grows polynomially with the collusion bound, Q. Thus, the 1-FE system that we want to use must be able to evaluate functions whose output lengths scale with Q. This is too strong, and one could even show that this is almost equivalent to full collusion resistance. While this seems like a major barrier, our observation is that this can be fixed by simply designing '*slightly better*' tMAFE systems. That is, suppose we can design tMAFE where $|\mathsf{tMAFE.SK}_{i,x_i}| = \mathsf{poly}(\lambda, \log Q)$, then the issue goes away, and our original template is enough for MAFE in the dynamic collusion model. Given 1-FE in the dynamic model can be constructed from identity-based encryption (IBE) [AMVY21, GGLW22, GGL24] (or one-way functions in the secret-key setting), this gives us a matching result for dynamic collusion MAFE.

Thus, we have reduced the dynamic collusion problem to designing tMAFE with 'short' secret keys. As it turns out, this problem was also encountered in the single-authority setting [GGLW22]. Garg et al. [GGLW22] referred to this as designing 1-FE with *weakly optimal efficiency*. While their original solution was a very complicated design, in a recent follow-up work Garg et al. [GGL24] presented a simpler generic approach to boost security of any 1-FE to dynamic collusion security. At a very high level, their idea was to use the *"deferred-encryption"* paradigm [GKW16, DG17b, DG17a, GHMR18, GHM+19, GV20] to embed 'tags' in every ciphertext and secret keys. The purpose of these tags was to ensure ciphertexts and keys with only matching tags can be decrypted. Garg et al. [GGLW22, GGL24] proved that such a *tagged* 1-FE is sufficient to achieve weak optimality by using standard combinatorial tricks. We refer the reader to [GGL24] for a detailed overview.

Our approach is to follow a similar template to achieve weak optimality for tMAFE. That is, first we embed 'tags' in every *partial* secret keys as well as ciphertexts, with the same tag functionality as in [GGLW22, GGL24]. And, then using standard combinatorial arguments, we boost this to weak optimality. Since we are in the multi-authority setting, thus we need to be extra careful while embedding tagging à la [GGL24]. The reason is, unlike single-authority setting, there are multiple authorities where each will have to individually embed a consistent tag. While this could be a big hurdle, what saves us is that we are working in the *trusted* model, thus we could rely on the same PRF trick as before. As we shown in Section 6, this is sufficient to design tMAFE with *weakly optimal efficiency* generically, and in turn, this gives us an MAFE in the dynamic collusion model from minimal[4] assumption of IBE (/one-way functions in secret-key setting).

So far, we have addressed the problem of designing MAFE against type-(a) attackers from minimal assumptions. However, such attackers are severely limited in two ways: (1) an attacker cannot corrupt master secret keys, and (2) an attacker cannot make more than a fixed number secret key queries (even if those keys do not provide any meaningful output). In the rest of the overview, we tackle both these problems.

2.2 MAFE with Authority Corruptions

In this section, we provide three main results. First, we design an MAFE scheme that is secure against type-(a+b) attackers. That is, we can prove security in presence of corrupt authorities. Next, we provide new generic compilers for boosting security of MAFE schemes with corrupt authorities. These generic compilers reduce the problem of designing MAFE with authority corruptions to a simpler model. And, lastly, we show improved lower bounds for MAFE with corrupt authorities, highlighting optimality of our MAFE design for a wide range of parameters. Let us dive into our first result.

Corrupt authorities. As a starting point, we set a much simpler goal. We start by designing MAFE that can handle authority corruptions only when all corrupt authorities are declared a-priori. That is, the MAFE system is initialized given a set of corrupted authorities $\mathcal{K} \subset [n]$. While

[4]Refer to [AMVY21] for a discussion on minimality of IBE.

such "declared" corruptions do not capture any real-world attacks, we will show later how these can be used as stepping stones for handling arbitrary corruptions.

Consider $\mathcal{K} = \{1\}$, i.e. only first authority is corrupt. Such an attacker can learn $C(0, x_2, \ldots, x_n)$ and $C(1, x_2, \ldots, x_n)$ from a ciphertext CT and secret keys $\mathsf{sk}_{2,x_2}, \ldots, \mathsf{sk}_{n,x_n}$ by honestly running KGen and Dec. Our intuition is that if $\mathcal{K}$ is available to us during the setup, then we could simply define two parallel MAFE instantiations, $\mathsf{MAFE}^{(0)}$, $\mathsf{MAFE}^{(1)}$ that encrypt $C(0, \cdot)$ and $C(1, \cdot)$. And, each honest authority will generate secret keys for both instantiations. An attacker in possession of $\mathsf{sk}_{2,x_2} = (\mathsf{sk}^{(0)}_{2,x_2}, \mathsf{sk}^{(1)}_{2,x_2}), \ldots$ can learn $C(0, x_2, \ldots, x_n)$ using $\mathsf{MAFE}^{(0)}$ and $C(1, x_2, \ldots, x_n)$ using $\mathsf{MAFE}^{(1)}$ as desired. The above can be easily generalized for any set $\mathcal{K} \subset [n]$ as long as the total number of attribute bits that are associated with $\mathcal{K}$ are at most $O(\log \lambda)$. (Looking ahead, going beyond this is either impossible, or requires very strong assumptions.)

One might wonder whether we could use complexity leveraging to guess $\mathcal{K}$, and then prove security of above approach under sub-exponential hardness assumptions. This would be wrong! Here we require $\mathcal{K}$ to be available even for setting up the MAFE system. Thus, description of the scheme reveals $\mathcal{K}$. This further highlights why the above corruption model is too weak (and somewhat meaningless). Regardless, it does serve as a good springboard as we show next.

Hiding $\mathcal{K}$ via HSS. Our insight is that if, rather than guessing $\mathcal{K}$, we consider all feasible $\mathcal{K}$ and secret share the computation under them, then this could potentially work. To execute this idea, we rely on homomorphic secret sharing schemes [ARS24, DIJL23, CM21, BCG+17a, FGJS17, BGI16]. Let us elaborate.

Let k denote the maximum number of authorities whose corruptions we want to tolerate. Let $\mathcal{C}$ denote the set of all size-k subsets of $[n]$. That is, $\mathcal{C}$ contains all possibilities for authorities who might be corrupt. Clearly, $|\mathcal{C}| = \binom{n}{k}$. We will sample $|\mathcal{C}|$ many MAFE systems with *declared corruptions*, $\mathsf{MAFE}^{(0)}, \ldots, \mathsf{MAFE}^{(|\mathcal{C}|)}$. In $\mathsf{MAFE}^{(i)}$, the corruption set $\mathcal{K}_i$ is the i-th entry in $\mathcal{C}$. During authority setup, the u-th authority samples fresh master public-secret key pairs from $\mathsf{MAFE}^{(i)}$ if and only if $u \notin \mathcal{K}_i$. We will set $\mathsf{MPK}_u = (\mathsf{mpk}^{(i)})_{\{i: u \notin \mathcal{K}_i\}}$ and $\mathsf{MSK}_u = (\mathsf{msk}^{(i)})_{\{i: u \notin \mathcal{K}_i\}}$ Similarly, to generate a partial secret key, we will sample a partial secret key from each $\mathsf{MAFE}^{(i)}$ such that $u \notin \mathcal{K}_i$ and $\mathsf{SK}_{u,x_u} = (\mathsf{sk}^{(i)}_{u,x_u})_{\{i: u \notin \mathcal{K}_i\}}$. To encrypt, we use an HSS to secret share C into $|\mathcal{C}|$ share and encrypt the corresponding share in each $\mathsf{MAFE}^{(i)}$ and $\mathsf{CT} = (\mathsf{ct}^{(i)})_i$. To ensure decryption works correctly, we have to make mild adjustments, but the high level idea is simply that a decryptor will recover MAFE outputs for every $\mathcal{K} \in \mathcal{C}$, and later it combine them by using reconstruction feature of HSS.

To prove security of the above system, we need a very strong simulation property for HSS. In a few words, we require that an attacker can learn evaluated shares corresponding to various honest shares as well. And, even if an attacker corrupts both original shares and evaluated (honest) shares, then it still does not learn anything beyond the final reconstructed value. To the best of our knowledge, this strong simulation property has not been explicitly studied for HSS schemes in the literature. In this work, we design such an HSS scheme by using multi-key fully homomorphic encryption scheme (mkFHE) with threshold decryption property [MW16, DHRW16]. For more details about this construction, please refer Appendix F. We remark that while there exists alternate constructions for HSS (e.g. Dao et al. [DIJL23]), they do not satisfy our strong simulation guarantee. For instance, if the evaluated shares of an honest share are leaked in the HSS of [DIJL23], then that would reveal error-free LPN samples which can be used to break sparse LPN security. We leave it as a very interesting open problem to construct HSS schemes with strong simulation guarantee from other standard assumptions as it readily gives an MAFE scheme with adaptive corruptions.

Lastly, note that for efficiency we require $\binom{n}{k} = \mathsf{poly}(\lambda)$, as otherwise the running time will be super-polynomial. For instance, if $n = \mathsf{poly}(\lambda), k = O(1)$, or $n = O(\log \lambda), k = O(\log \lambda)$, then above construction could be efficiently instantiated. For more details, we refer the reader to Section 7.

Boosting selective to non-adaptive authority corruptions. Next, we describe an interesting approach to be boost security of any MAFE scheme, where the adversary must selectively decide the set of corrupt authorities (i.e., pick $\mathcal{K}$ before setup), to an MAFE where the authority can *non-adaptively* pick $\mathcal{K}$. We remark that the selective corruption model is different than the "declared-corruption" model that we introduced earlier. Here the MAFE scheme is defined independent of $\mathcal{K}$, and only the challenger needs to know $\mathcal{K}$ selectively.

Somewhat surprisingly, our compiler is again just our base MAFE template. That is, recall our MAFE construction based on tMAFE and 1-FE. Our observation is that we replace tMAFE with an MAFE scheme with selective corruptions, then we can essentially construct an MAFE scheme where an attacker can corrupt any authority before declaring C. We have to organize the security arguments carefully as we need to leverage the fact that pre-challenge simulator of 1-FE works exactly as an honest challenger [SS10, GVW12, AV19]. Once we do this, then we can corrupt $\mathsf{1\text{-}fe.msk}_i$ for any i non-adaptively, and continue simulation from the challenge phase naturally. Since we know all corruptions by the challenge phase (in the non-adaptive corruption model), thus we can instantiate and rely on security of MAFE with selective authority corruptions. We describe the proof in Section 8. Our compiler has reduced the task of designing MAFE with non-adaptive corruptions to just selective corruptions, and we leave the problem of designing MAFE with selective corruptions from minimal assumption as an interesting open problem.

Lastly, we complement our positive results regarding authority corruptions with improved/tighter lower bounds. We discuss those briefly below.

Tighter lower bounds. In our construction for MAFE with adaptive corruptions using HSS, we instantiated 2 MAFE schemes per one corrupted bit. Extending this generically, we would need 2^{χ} parallel compositions if we corrupt a maximum of χ bits. So, we ask the natural question — *can we achieve any slightly better efficiency?* In particular, can the parameters of our system grow sub-linearly in 2^{χ}, i.e, $2^{\chi(1-\xi)}$ where $\xi > 0$? We show that an MAFE scheme with such a guarantee can be transformed into an obfuscation scheme.

To provide an intuition for this result, recall [BV15, AJ15]. These results state that in a 1-FE scheme secure against Q secret-key corruptions, $|\mathsf{1\text{-}FE.ct}| \geq Q \cdot \mathsf{poly}(\lambda)$. Any sub-linearity in size i.e, $Q^{1-\xi}, \xi > 0$ implies iO (as long as 1-FE offers indistinguishability-based security). In Section 9 we show how to construct 1-FE with sub-linearity from an MAFE scheme with selective corruptions. At a high level, $\mathsf{1\text{-}FE.Enc}(x) \equiv \mathsf{MAFE.Enc}(F(x, \cdot, \cdot))$ where F is a circuit that uses j, C and outputs j-th bit of $C(x)$. We will corrupt MSK for slot j and set $\mathsf{1\text{-}FE.sk}_C := (\mathsf{MSK}, \mathsf{mafe.sk}_C)$. By sub-linearity of MAFE, $\mathsf{1\text{-}FE.ct}$ only grows in $Q^{1-\xi}$. In addition, when $Q = 1$, this will be a worst-case iO scheme (XiO) with sub-linear size, an object that in conjunction with LWE yields iO [LPST16]. Moreover, we also get that simulation-secure versions of above are impossible in the standard model due to known incompressibility arguments [BSW11, AGVW13].

2.3 Multi-Authority Attribute-Based Functional Encryption

Finally, we define an amalgamation of MAFE with its attribute-based (all-or-nothing) variant. We call it multi-authority attribute-based functional encryption (MA-ABFE). It generalizes MA-AB-

IPFE systems studied in prior works [AGT21, DP23]. Our goal is to study MAFE against attackers that can make an unbounded number of *unsatisfying* secret key queries, but only a bounded number of *satisfying* key queries.

Defining MA-ABFE. An MA-ABFE consists of the same four algorithms — ASetup, KGen, Enc, Dec. The main difference is that the KGen takes two inputs, a predicate attribute x and a circuit input y, while Enc is used to encrypt a pair of predicate and circuit (P, C). Correctness is naturally extended to learn $C(Y) = C(y_1, \ldots, y_n)$ if and only if $P(X) = P(x_1, \ldots, x_n) = 1$. For security, we consider a fused definition where the attacker can issue both, *satisfying* and *unsatisfying* queries. For a challenge ciphertext encrypting (P^*, C^*), a *satisfying* query on $\{(x_i, y_i)\}_i$ is such that $P^*(X) = 1$, while it is *unsatisfying* if $P^*(X) = 0$. And, the scheme is secure if an attacker can only learn $\mathcal{C}^*(Y)$ for every satisfying query $\{(x_i, y_i)\}_i$. In this work, we study attackers that can make an arbitrary polynomial number of *unsatisfying* key queries, but only a bounded number of *satisfying* key queries. We call this (poly, Q)-bounded security.

A simple but flawed construction. Why isn't MA-ABFE just a black-box combination of MA-ABE and MAFE? Consider the following construction. An encryptor computes an MAFE ciphertext for circuit C, and encrypts it using MA-ABE for predicate P. KGen provides secret keys for both MA-ABE and MAFE systems, individually. While correctness follows trivially, such a scheme will be totally insecure. Consider an attacker that makes 1 *satisfying* and 1 *unsatisfying* secret key query. Now using the satisfying MA-ABE key, an attacker can recover the MAFE ciphertext and then it can decrypt it using both MAFE secret keys (one from the *satisfying* and other from *unsatisfying* secret key). This merely suggests that designing MA-ABFE is not a straightforward task.

Starting from scratch: $(\mathsf{poly}, 1)$-security. We start by explaining our design for a non-adaptively secure $(\mathsf{poly}, 1)$-MA-ABFE scheme. At a very high level, the construction draws inspiration from the classical Sahai-Seyalioglu [SS10] 1-FE construction. Our strategy is to first garble the circuit C^*, and encrypt its $2n$ wire labels under $2n$ $\mathsf{MA\text{-}ABE}$ ciphertexts for predicate P^*. Now the i-th authority generates n MA-ABE secret keys for attribute x_i. That is, it gives secrets keys corresponding to wire keys $\{(j, y_j)\}_{j \in [n]}$. Thus, if an attacker makes a single *satisfying* key query can only learn half of the wire labels. Thus, by using garbling simulation security, we can simulate it given just $C^*(Y)$ to generate the ciphertext. We can formalize the above ideas to prove non-adaptive $(\mathsf{poly}, 1)$-security of the above design. Note that this preserves the MA-ABE predicate class, and the MAFE portion is for $\mathsf{P/Poly}$ circuits. We notice that by relying on non-committing techniques (generalized to MA-ABE) [GVW12, HMNY22, HKM$^+$23, GY24], we can also boost its security to adaptive $(\mathsf{poly}, 1)$-security. We provide full details later in Section 11.

Multi-Authority Collusion Amplification Techniques [GY24]. Moving forward, we want to amplify $(\mathsf{poly}, 1)$-secure MA-ABFE to (poly, Q)-security. At a high level, our strategy is to mirror the multi-authority collusion amplification template developed in a very recent work by Goyal-Yadugiri [GY24]. In a few words, we start by strengthening the distributed client-server framework (dCSF) from [GY24] to an attribute-based dCSF. This is provided in Section 12. Given this, we can design a (poly, Q)-secure MA-ABFE scheme in the Random Oracle Model (ROM) by relying on sub-exponential security of $(\mathsf{poly}, 1)$-secure MA-ABE. We discuss our construction in detail later in Section 13. Overall, our result lifts every component of the blueprint laid out in [GY24] to handle unbounded number of *unsatisfying* queries. While there are many subtleties that appear during this process, we are able to design efficient reductions if we work in the ROM. This is crucial because now we have to ensure that each *unsatisfying* query does not reveal any "un-simulatable" information about the circuit C^* as well as the random set selections made for answering each secret key query do not negatively affect the desired statistical lemmas which are essential for proving security. We refer the reader to main body for a more detailed discussion.

We conclude by saying that by plugging our compiler with known results from the MA-ABE literature, we get an MA-ABFE scheme for NC^1 policies and $\mathsf{P/Poly}$ circuits ($\mathsf{NC}^1 \circ \mathsf{P/Poly}$) assuming sub-exponential hardness of standard pairing-based assumptions [LW11, DKW23] in the ROM. Similarly, we also get MA-ABFE for $\mathsf{DNF} \circ \mathsf{P/Poly}$ assuming the hardness of LWE in the ROM, by relying on [DKW21].

As a side contribution, we also show a simple construction for a statically secure MA-ABE for any polynomial-size predicate family from vanilla witness encryption by generalizing the framework provided in [GGSW13]. Our construction *does not* require Random Oracles and also handles arbitrary authority corruptions declared statically (provided along with all key generation queries). [WWW22] constructed an MA-ABE scheme for subset functionalities using evasive LWE but left constructing MA-ABE from vanilla witness encryption as an open question. Our construction resolves this question and also improves on their work by constructing an MA-ABE for any polynomial-size predicate family from *vanilla* witness encryption. We leave constructing an adaptively secure MA-ABE for all polynomial-size predicates from vanilla witness encryption as an interesting open problem. We believe that using similar techniques from [WW24, GY25] might yield some fruitful results.

Both our MA-ABE and [WWW22] can be instantiated under the hardness of evasive LWE assumption [Wee22, Tsa22, VWW22]. Thus, this readily gives us a statically secure MA-ABFE for all polynomial size predicates and circuits *without* Random Oracles from evasive LWE. For more details, please confer Appendix K.

3 Definitions and Preliminaries

We introduce the notation we use throughout the paper below.

Notation. By PPT we denote probabilistic polynomial-time. We denote the security parameter by λ and the set of positive integers by $\mathbb{N}$. For any $a, b \in \{0\} \cup \mathbb{N},\ a \leq b$, we denote by $[a, b]$, the set of all integers from a to b including a and b. In other words, $[a, b] = \{a, \ldots, b\}$. We denote by $[n] := [1, n]$. We denote by $x \xleftarrow{\$} \mathcal{X}$, the process of sampling an element x from the set $\mathcal{X}$, with uniform probability. Similarly, for any PPT algorithm $\mathcal{A}$, $x \leftarrow \mathcal{A}(y)$ denotes the process of sampling x from the output distribution of $\mathcal{A}$ when run on y. By $\mathsf{negl}(\cdot)$, we denote negligible functions. By poly, we denote positive polynomials. All of our logarithms log are in base 2. We denote the power set A by 2^A and the set of all subsets of A of size a by $2^A\big|_a$.

We say that two efficiently samplable probability distributions $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda \in \mathbb{N}}$ and $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_{\lambda \in \mathbb{N}}$ are computationally indistinguishable if for any non-uniform PPT distinguisher $\mathcal{D} = \{\mathcal{D}_\lambda\}_{\lambda \in \mathbb{N}}$, and for large enough $\lambda \in \mathbb{N}$,

$$\left| \Pr_{\alpha \leftarrow \mathcal{X}_\lambda}\left[1 \leftarrow \mathcal{D}(1^\lambda, \alpha)\right] - \Pr_{\alpha \leftarrow \mathcal{Y}_\lambda}\left[1 \leftarrow \mathcal{D}(1^\lambda, \alpha)\right] \right| \leq \mathsf{negl}(\lambda)$$

Now, we provide the definition for an MAFE scheme we use in this work. Specifically, we revise the definition of MAFE from [GY24] and provide a strengthening of this definition where authorities can be created dynamically. In contrast, [GY24] was only able to construct MAFE schemes where the number of authorities are bounded and must be declared for the specification of the scheme.

Syntax. A multi-authority functional encryption (MAFE) scheme MAFE for circuit class $\mathcal{C} = \{\mathcal{C}_\lambda : \mathcal{X}_{\lambda,1} \times \ldots \times \mathcal{X}_{\lambda,n} \to \mathcal{O}_\lambda\}_{\lambda \in \mathbb{N}}$ and global identifier domain $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda \in \mathbb{N}}$ consists of the following polynomial-time algorithms.

$\mathsf{GSetup}(1^\lambda, 1^Q, 1^s) \to \mathsf{CRS}$. The possibly randomized global setup algorithm takes as input the security parameter λ, the query bound Q, the maximum circuit size s, and outputs CRS. The following algorithms take CRS implicitly.

$\mathsf{ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}}) \to (\mathsf{MPK}, \mathsf{MSK})$. The probabilistic authority setup algorithm takes as input the input-size for the id-th authority ℓ_{id}, and outputs master public and secret key pair $(\mathsf{MPK}, \mathsf{MSK})$.

$\mathsf{KGen}(\mathsf{id}, \mathsf{MSK}_{\mathsf{id}}, \mathsf{GID}, x) \to \mathsf{SK}_{\mathsf{id},\mathsf{GID},x}$. The possibly randomized key generation algorithm takes as input the id-th master secret key $\mathsf{MSK}_{\mathsf{id}}$, the global identifier for the user $\mathsf{GID} \in \mathcal{GID}_\lambda$, an input $x \in \mathcal{X}_{\mathsf{id}} = \{0,1\}^{\ell_{\mathsf{id}}}$, and outputs the secret key $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x}$.

Note: We do not require KGen to take $\{\mathsf{MPK}_{\mathsf{idx}}\}_{\mathsf{idx}}$ like [GY24]. This is why our MAFE has truly dynamic functionality as authorities can join at any point.

$\mathsf{Enc}(\{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{I}}, C) \to \mathsf{CT}$. The probabilistic encryption algorithm takes as input the master public keys for authorities in $\mathcal{I}$, $\{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{I}}$, a circuit $C \in \mathcal{C}_\lambda$, and outputs ciphertext CT.

$\mathsf{Dec}(\{\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}\}_{\mathsf{id} \in \mathcal{I}}, \mathsf{CT}) \to y$. The deterministic decryption algorithm takes as input the secret keys from authorities in $\mathcal{I}$, $\{\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}\}_{\mathsf{id} \in \mathcal{I}}$, ciphertext CT, and outputs the value y.

Definition 3.1 (MAFE). An MAFE scheme $(\mathsf{GSetup}, \mathsf{ASetup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ is said to be an MAFE scheme for circuit class $\mathcal{C}$ and $\mathcal{GID}$ if it satisfies the following properties.

Correctness. For any $\lambda \in \mathbb{N}$, $Q = Q(\lambda), n = n(\lambda), s = s(\lambda)$, $C \in \mathcal{C}_\lambda, \mathsf{GID} \in \mathcal{GID}_\lambda$, $\forall\ \mathcal{I}, \ell_{\mathsf{id}} = \ell_{\mathsf{id}}(\lambda), x_{\mathsf{id}} \in \{0,1\}^{\ell_{\mathsf{id}}}$,

$$\Pr\left[\begin{array}{l} C(x_1, \ldots x_n) = \\ \mathsf{Dec}(\{\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}\}_{\mathsf{id}}, \mathsf{CT}) \end{array} : \begin{array}{l} \mathsf{CRS} \leftarrow \mathsf{GSetup}(1^\lambda, 1^Q, 1^s), \forall\ \mathsf{id} \in \mathcal{I}, \\ (\mathsf{MPK}_{\mathsf{id}}, \mathsf{MSK}_{\mathsf{id}}) \leftarrow \mathsf{ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}}), \\ \mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}} \leftarrow \mathsf{KGen}(\mathsf{id}, \mathsf{MSK}_{\mathsf{id}}, \mathsf{GID}, x_{\mathsf{id}}), \\ \mathsf{CT} \leftarrow \mathsf{Enc}(\{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}}, C) \end{array}\right] = 1$$

Security. There exists a stateful simulator Sim such that for any admissible adversary $\mathcal{A}$, $\forall\ \lambda \in \mathbb{N}$,

$$\Pr\left[b \leftarrow \mathcal{A}^{\mathcal{O}_b(\cdot,\cdot)}(\mathsf{CT}) : \begin{array}{l} Q, s \leftarrow \mathcal{A}(1^\lambda), b \xleftarrow{\$} \{0,1\}, \\ \mathsf{CRS} \leftarrow \mathcal{O}_b(1^\lambda, 1^Q, 1^s), \\ (C, \mathcal{I}) \leftarrow \mathcal{A}^{\mathcal{O}_b(\cdot,\cdot)}(\mathsf{CRS}), \\ \mathsf{CT} \leftarrow \mathcal{O}_b(\mathcal{I}, C) \end{array}\right] \le \frac{1}{2} + \mathsf{negl}(\lambda)$$

where both $\mathcal{O}_0, \mathcal{O}_1$ are stateful oracles that respond to queries from $\mathcal{A}$ of the form

$(\mathsf{ASetup}, (\mathsf{id}, 1^{\ell_{\mathsf{id}}}))$ $\mathcal{A}$ queries to setup id-th authority with input size ℓ_{id}. Gets $\mathsf{MPK}_{\mathsf{id}}$ in response.

($\mathsf{KGen}, (\mathsf{id}, \mathsf{GID}, x)$) $\mathcal{A}$ queries to corrupt secret key for id-th authority for GID and input x such that $|x| = \ell_{\mathsf{id}}$. Gets $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x}$ in response.

$\mathcal{O}_0$ uses ($\mathsf{GSetup}, \mathsf{ASetup}, \mathsf{KGen}, \mathsf{Enc}$) to respond to queries from $\mathcal{A}$. However, it maintains state to store public parameters for all authorities and secret keys generated thus far. $\mathcal{O}_1$ uses the simulator Sim to respond to these queries. $\mathcal{O}_1$ also provides Sim with the set $1^{|C|}, \mathcal{V}$ for ciphertext generation. Here, $\mathcal{V} = \{(\mathsf{GID}, X = \{x_{\mathsf{GID},\mathsf{id}}\}_{\mathsf{id}\in\mathcal{I}}, C(X))$: for every GID that was used to query all $\mathsf{id} \in \mathcal{I}$ in pre-challenge query phase$\}$. In addition, $\mathcal{O}_1$ also provides $C(X)$ for every X completely queried in post-challenge phase to Sim.

A stateful PPT machine $\mathcal{A}$ is admissible if it makes polynomially many queries with the restriction that for the set of authorities submitted in challenge query phase, $\mathcal{I}$, it can make adaptively at most Q secret key queries across $|\mathcal{I}|$ authorities such that each authority is queried at most once per GID. In addition, $\mathcal{A}$ will make at most one ciphertext query with circuit $C \in \mathcal{C}_\lambda$ of size at most s adaptively. For simplicity, we assume that authorities will be created before secret key is queried for that authority. Thus, all $\mathsf{id} \in \mathcal{I}$ will be created before challenge circuit is declared.

Definition 3.2 (MAFE with "bounded" authorities)**.** An MAFE scheme ($\mathsf{GSetup}, \mathsf{ASetup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec}$) is said to be an MAFE scheme with bounded authorities if the number of authorities (n) are declared a-priori and GSetup takes 1^n as input and $\mathcal{I} = [n]$. In addition, KGen also takes master public keys from all n authorities. The security definition is also slightly altered where all the authorities are initiated in the setup phase and master public keys for all authorities are provided along with CRS. That is, $\mathcal{O}_b$ is essentially a KGen oracle and $\mathcal{A}$ is only allowed to query it at most Q times.

We provide the definition of single-authority MAFE scheme, also called BFE.

Definition 3.3 (BFE)**.** A BFE scheme is an MAFE scheme for a single authority (i.e, $n = 1$). Hence, we ignore CRS and $\mathcal{GID}$ as all key queries are answered by a single authority. We also alter the syntax where Setup (which also depends on output size ℓ_{out}) is used instead of GSetup. $\mathsf{id}, \mathsf{GID}$ are dropped from KGen. Our constructions require a stronger version of simulation that is implicitly satisfied by BFE schemes from [GVW12, AV19]. We define it as follows.

Strong Simulation Security. For any admissible adversary $\mathcal{A}$, there exists a stateful PPT simulator $\mathsf{Sim} = (\mathsf{S}_0, \mathsf{S}_1, \mathsf{S}_2, \mathsf{S}_3)$ such that,

$$\left\{ \mathcal{A}^{\mathsf{KGen}(\mathsf{MSK},\cdot)}(\mathsf{CT}) : \begin{array}{l} Q, \ell_{\mathsf{in}}, \ell_{\mathsf{out}} \leftarrow \mathcal{A}(1^\lambda), \\ (\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, 1^Q, 1^{\ell_{\mathsf{in}}}, 1^{\ell_{\mathsf{out}}}), \\ C \leftarrow \mathcal{A}^{\mathsf{KGen}(\mathsf{MSK},\cdot)}(\mathsf{MPK}), \\ \mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{MPK}, C) \end{array} \right\}$$

$$\equiv$$

$$\left\{ \mathcal{A}^{\mathsf{KGen}(\mathsf{MSK},\cdot)}(\mathsf{CT}) : \begin{array}{l} Q, \ell_{\mathsf{in}}, \ell_{\mathsf{out}} \leftarrow \mathcal{A}(1^\lambda), \\ (\mathsf{MPK}, \mathsf{MSK}, \mathsf{st}) \leftarrow \mathsf{S}_0(1^\lambda, 1^Q, 1^{\ell_{\mathsf{in}}}, 1^{\ell_{\mathsf{out}}}), \\ C \leftarrow \mathcal{A}^{\mathsf{S}_1(\mathsf{MSK},\mathsf{st},\cdot)}(\mathsf{MPK}), \\ \mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{MPK}, C) \end{array} \right\}$$

$$\approx_c$$

$$\left\{ \mathcal{A}^{\mathsf{S}_3^{C(\cdot)}(\mathsf{st},\cdot)}(\mathsf{CT}) : \begin{array}{l} Q, \ell_{\mathsf{in}}, \ell_{\mathsf{out}} \leftarrow \mathcal{A}(1^\lambda), \\ (\mathsf{MPK}, \mathsf{MSK}, \mathsf{st}) \leftarrow \mathsf{S}_0(1^\lambda, 1^Q, 1^{\ell_{\mathsf{in}}}, 1^{\ell_{\mathsf{out}}}), \\ C \leftarrow \mathcal{A}^{\mathsf{S}_1(\mathsf{MSK},\mathsf{st},\cdot)}(\mathsf{MPK}), \\ \mathsf{CT} \leftarrow \mathsf{S}_2(\mathsf{st}, 1^{|C|}, \mathcal{V}) \end{array} \right\}$$

where $\mathcal{V} = \{(x, C(x)) :$ for each x queried in pre-challenge phase$\}$. A stateful PPT machine $\mathcal{A}$ is said to be admissible if

- It makes at most Q queries to the KGen oracle *and* one challenge circuit query (OR)
- Makes unbounded queries to KGen oracle and *does not* make any challenge query.

Definition 3.4 (Dynamic BFE)**.** A BFE scheme (Setup, KGen, Enc, Dec) is said to be a dynamic query bounded BFE scheme for circuit class $\mathcal{C} = \{\mathcal{C}_\lambda : \{0,1\}^{\ell_{\mathsf{in}}} \to \{0,1\}^{\ell_{\mathsf{out}}}\}_{\lambda \in \mathbb{N}}$ if it satisfies Definition 3.3 and Setup no longer takes the query bound Q as input and instead Enc takes it as input. This also implies that the running times of Setup and KGen are independent of Q and are polynomial in λ, s and running time of Enc is polynomial in λ, Q, s. Security of the scheme is defined accordingly.

Remark 3.5 (Ciphertext-policy BFE, [GVW12, AV19])**.** A BFE scheme for any class of P/Poly circuits can be constructed from the minimal assumption of public-key encryption. In addition, when considering ciphertext-policy variant of BFE as opposed to key-policy BFE in [GVW12, AV19], the Setup algorithm does not need to take the size of the circuit as input. One can use replace Yao's garbling scheme in [SS10] with point-and-permute garbling scheme and only work with the input and output lengths to construct BFE in which the Enc algorithm can handle circuits of unbounded-size.

Remark 3.6 ([AMVY21, GGLW22, GGL24])**.** A dynamic query bounded (ciphertext-policy) BFE scheme for any class of P/Poly circuits can be constructed from the minimal assumption of Identity-Based Encryption.

4 MAFE with Trusted Setup

In this section, we define and show the construction of an important variant of MAFE, *MAFE with trusted setup* (tMAFE). In a tMAFE scheme, there is central authority that initiates the system by generating CRS and master public-secret key pairs for all authorities. So, we forgo the ASetup algorithm in this scheme and consider GSetup to output all the required parameters for setting up the system. In the next section, we show that this weakening of MAFE is enough to construct general MAFE from minimal assumptions. We formally define this object below and provide the construction.

Definition 4.1 (tMAFE)**.** A tMAFE scheme $(\mathsf{GSetup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ is said to be an MAFE scheme with trusted setup (tMAFE) for circuit class $\mathcal{C}$ and $\mathcal{GID}$ if there is a central authority that generates all system parameters (i.e, $\mathsf{CRS}, \mathsf{MPK}_{\mathsf{id}}, \mathsf{MSK}_{\mathsf{id}}$ for all id). To reflect this, we ignore the ASetup algorithm and change GSetup algorithm as follows.

$\mathsf{GSetup}(1^\lambda, 1^Q, 1^n, 1^s, 1^{\ell_1}, \ldots, 1^{\ell_n}) \to (\mathsf{CRS}, \{(\mathsf{MPK}_{\mathsf{id}}, \mathsf{MSK}_{\mathsf{id}})\}_{\mathsf{id}\in[n]})$**.** The possibly randomized global setup takes the number of authorities n and input sizes for all authorities, $\ell_1, \ldots, \ell_n$ as additional input, and outputs CRS, master public-secret key pairs $\{(\mathsf{MPK}_{\mathsf{id}}, \mathsf{MSK}_{\mathsf{id}})\}_{\mathsf{id}\in[n]}$ for all authorities.

The security of the scheme is defined analogously.

4.1 Construction

We provide the construction of a tMAFE scheme for $\mathsf{P/Poly}$ circuits and $\mathcal{GID}$ as follows. At a high level, in [GY24] constructs a *statically* secure MAFE scheme from the minimal assumption of one-way functions using a PRF key embedded as part of the CRS. This is already an optimal construction if we are limited to static security. However, they use additional assumptions to realize an adaptively secure MAFE scheme such as random oracle, non-interactive key exchange, etc.

As explained in overview, our observation, if we restrict ourselves to the trusted setup model, we can embed the same PRF key as part of master secret key for each authorities. Since there is a centralized authority that distributes these keys and we do not consider authority corruptions, this key remains hidden and we can realize adaptive security easily. In Section 5 we show how we can use tMAFE in conjunction with BFE to construct a general MAFE scheme. Parts of the following construction are taken verbatim from [GY24].

Construction 4.2 (tMAFE)**.** We provide the construction of a tMAFE scheme for $\mathsf{P/Poly}$ circuits adn $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda\in\mathbb{N}}$ using an MAFE for $\mathsf{P/Poly}$ circuits for *one* GID. ([GY24]), a dCSF scheme for $\mathsf{P/Poly}$ circuits and $\mathcal{GID}$ (Definition B.1), and $\mathsf{PRF} : \{0,1\}^\lambda \times \mathcal{GID}_\lambda \to 2^{[N]}\big|_D \times 2^{[T]}\big|_v$ (Definition A.3) as follows. Here N, D, T, v are from Lemma B.2 and Lemma B.3.

$\underline{\mathsf{GSetup}(1^\lambda, 1^Q, 1^n, 1^s, 1^{\ell_1}, \ldots, 1^{\ell_n})}$**.** $\forall\ u \in [N]$, sample $\mathsf{mafe.crs}_u \leftarrow \mathsf{1MAFE.GSetup}(1^\lambda, 1^n, 1^s)$. Set $\mathsf{CRS} := (\mathsf{mafe.crs}_u)_{u\in[N]}$. Sample $K \xleftarrow{\$} \{0,1\}^\lambda$. $\forall\ u \in [N], (\mathsf{mafe.mpk}_{u,\mathsf{id}}, \mathsf{mafe.msk}_{u,\mathsf{id}}) \leftarrow \mathsf{1MAFE.ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}})$. Set $\mathsf{MPK}_{\mathsf{id}} := (\mathsf{mafe.mpk}_{u,\mathsf{id}})_{u\in[N]}$, and $\mathsf{MSK}_{\mathsf{id}} := (K, (\mathsf{mafe.msk}_{u,\mathsf{id}})_u)$. Output $\mathsf{CRS}, \{(\mathsf{MPK}_{\mathsf{id}}, \mathsf{MSK}_{\mathsf{id}})\}_{\mathsf{id}\in[n]}$.

$\underline{\mathsf{KGen}(\mathsf{id}, \mathsf{MSK}_{\mathsf{id}}, \{\mathsf{MPK}_{\mathsf{idx}}\}_{\mathsf{idx}\in[n]}, \mathsf{GID}, x)}$**.** Parse $\mathsf{MPK}_{\mathsf{idx}}$ as $(\mathsf{mafe.mpk}_{u,\mathsf{idx}})_{u\in[N]}$ for each $\mathsf{idx} \in [n]$ and $\mathsf{MSK}_{\mathsf{id}}$ as $(K, (\mathsf{mafe.msk}_{u,\mathsf{id}})_{u\in[N]})$. Compute $(\mathbf{S}, \mathbf{\Delta}) = \mathsf{PRF}(K, \mathsf{GID})$.

- $\{\widehat{x}^u_{\mathsf{GID},\mathsf{id}}\}_{u\in[N]} = \mathsf{dCSF.ServEnc}(1^\lambda, 1^Q, 1^n, 1^s, \mathsf{GID}, \mathsf{id}, x, \mathbf{\Delta})$.
- $\forall\ u \in \mathbf{S}, \mathsf{mafe.sk}_u = \mathsf{1MAFE.KGen}(\mathsf{id}, \mathsf{mafe.msk}_{u,\mathsf{id}}, \{\mathsf{mafe.mpk}_{u,\mathsf{idx}}\}_{\mathsf{idx}}, \widehat{x}^u_{\mathsf{GID},\mathsf{id}})$.

Output $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} := (\mathbf{S}, \{\mathsf{mafe.sk}_u\}_{u \in \mathbf{S}})$.

$\underline{\mathsf{Enc}(\{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id} \in [n]}, C)}$. Parse $\mathsf{MPK}_{\mathsf{id}}$ as $(\mathsf{mafe.mpk}_{u,\mathsf{id}})_{u \in [N]}$ for each $\mathsf{id} \in [n]$. Sample $\{\widehat{C}^u\}_{u \in [N]} \leftarrow \mathsf{dCSF.CliEnc}(1^\lambda, 1^Q, 1^n, 1^s, C)$ and with $\widehat{F}^u = \mathsf{dCSF.UserComp}(\cdot, \ldots, \cdot, \widehat{C}^u)$, $\mathsf{mafe.ct}_u \leftarrow \mathsf{1MAFE.Enc}(\{\mathsf{mafe.mpk}_{u,\mathsf{id}}\}_{\mathsf{id} \in [n]}, \widehat{F}^u)$. Output $\mathsf{CT} := (\mathsf{mafe.ct}_u)_{u \in [N]}$.

$\underline{\mathsf{Dec}(\{\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}\}_{\mathsf{id} \in [n]}, \mathsf{CT})}$. Parse $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}$ as $(\mathbf{S}_{\mathsf{id}}, \{\mathsf{mafe.sk}_{u,\mathsf{id}}\}_{u \in \mathbf{S}_{\mathsf{id}}})$ and CT as $(\mathsf{mafe.ct}_u)_{u \in [N]}$. If all $\mathbf{S}_{\mathsf{id}}$'s are not the same, abort and output $\perp$.

Otherwise, let $\mathbf{S} = \mathbf{S}_1$ and for each $u \in \mathbf{S}$, $\widehat{y}^u_{\mathsf{GID}} = \mathsf{1MAFE.Dec}(\{\mathsf{mafe.sk}_{u,\mathsf{id}}\}_{\mathsf{id}}, \mathsf{mafe.ct}_u)$. Output $y = \mathsf{dCSF.Decode}(\{\widehat{y}^u_{\mathsf{GID}}\}_{u \in \mathbf{S}}, \mathbf{S})$.

Theorem 4.3. If 1MAFE is a 1-GID MAFE scheme for P/Poly circuits, dCSF is a secure dCSF scheme for P/Poly circuits and $\mathcal{GID}$ (Definition B.1), and $\mathsf{PRF} : \{0,1\}^\lambda \times \mathcal{GID}_\lambda \to 2^{[N]}\big|_D \times 2^{[T]}\big|_v$ is a secure PRF (Definition A.3), then Construction 4.2 is a tMAFE scheme for P/Poly circuits and $\mathcal{GID}$.

Proof. The proof of this theorem follows closely the structure of proofs from [GY24]. We provide a sketch of the hybrid argument here.

Hybrid 0. This is the real experiment with honest challenger. We use $\mathsf{GSetup}, \mathsf{KGen}, \mathsf{Enc}$ algorithms to respond to adversary's queries.

Hybrid 1. In this hybrid, we will sample random $(\mathbf{S}, \mathbf{\Delta})$ for each of the Q GID queries during setup. As the PRF key is not leaked to the adversary, this change remains unnoticeable by the security of PRF.

Hybrid 2. In this hybrid, we will check if $\{\mathbf{S}_{\mathsf{GID}}\}_{\mathsf{GID}}$ obey Lemma B.2 and $\{\mathbf{\Delta}_{\mathsf{GID}}\}_{\mathsf{GID}}$ obey Lemma B.3. Hybrids 1 and 2 are statistically indistinguishable due to those lemmas. We will also compute the set of corrupted users and set of unique indices in each $\mathbf{\Delta}_{\mathsf{GID}}$.

Hybrid $3, j$. for $j \in [N+1]$. In this hybrid, we will simulate the first $j-1$ 1MAFE instantiations if the corresponding u is not corrupted. The indistinguishability between hybrids can be argued from the security of 1MAFE.

Hybrid 4. In this hybrid, we know the set of corrupted users, unique indices and all the non-corrupted $u \in [N]$ only use outputs from dCSF. Hence, we will simulate dCSF. The indistinguishability of hybrids can be argued by security of the same. This is description of tMAFE.Sim.

The full proof is provided in the full version. □

Corollary 4.4 ([GY24]). Assuming the existence of public-key encryption, there exists a tMAFE scheme for P/Poly circuits and $\mathcal{GID}$.

Proof. As shown in [GY24], we can construct a 1MAFE scheme for P/Poly circuits and dCSF for P/Poly circuits and $\mathcal{GID}$ from PKE and one-way functions respectively. Hence, the proof of this corollary follows immediately. □

Corollary 4.5. Assuming the existence of one-way functions, there exists a secret-key tMAFE scheme for $\mathsf{P/Poly}$ circuits and $\mathcal{GID}$.

Proof. The proof of this corollary is similar to Corollary 4.4 as a secret-key $\mathsf{1MAFE}$ instantiation for $\mathsf{P/Poly}$ circuits can be constructed from any secret-key encryption scheme. □

5 MAFE from Public-Key Encryption

In this section, we provide the construction of an general MAFE scheme for $\mathsf{P/Poly}$ circuits (Definition 3.1) using a tMAFE scheme and a BFE scheme for $\mathsf{P/Poly}$ circuits. As explained in overview is that we initiate tMAFE in Enc of the MAFE scheme and use BFE to encrypt the key generation circuit of tMAFE for various authorities. This way, each authority can independently initialize a BFE instantiation and issue secret keys for a bounded number of input queries. While decrypting we can use the BFE secret keys to decrypt and learn tMAFE secret keys. And as we encrypt the circuit under tMAFE, we can decrypt this and learn the output. Note that the whole construction crucially requires that authorities are not corrupted.

Remark 5.1 (Deterministic KGen)**.** In the construction below, we assume w.l.o.g that $\mathsf{tMAFE.KGen}$ is a deterministic algorithm. We can always de-randomize a probabilistic key generation algorithm by including a PRF key (for appropriate parameters) along with MSK as long as MSK is not corrupted.

We now provide the construction of an MAFE scheme with "dynamic" authorities as follows.

Construction 5.2 (MAFE)**.** We provide the construction of an MAFE scheme for $\mathsf{P/Poly}$ circuits and $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda\in\mathbb{N}}$ using a BFE scheme for $\mathsf{P/Poly}$ circuits (Definition 3.3) and a tMAFE scheme for $\mathsf{P/Poly}$ circuits and $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda\in\mathbb{N}}$ as follows.

$\underline{\mathsf{GSetup}(1^\lambda, 1^Q, 1^s).}$ Output $\mathsf{CRS} := (\lambda, Q, s)$.

$\underline{\mathsf{ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}}).}$ Sample keys $(\mathsf{bfe.mpk}, \mathsf{bfe.msk}) \leftarrow \mathsf{BFE.Setup}(1^\lambda, 1^Q, 1^{\lambda+|\mathsf{GID}|+\ell_{\mathsf{id}}}, 1^{\kappa_{\mathsf{id}}})$. Here $\kappa_{\mathsf{id}} \le \mathsf{poly}(\lambda, Q, |\mathsf{GID}|, \ell_{\mathsf{id}})$ is the maximum size of the secret key of tMAFE for these parameters. Output $\mathsf{MPK} := (\ell_{\mathsf{id}}, \mathsf{bfe.mpk})$ and $\mathsf{MSK} := (\ell_{\mathsf{id}}, \mathsf{bfe.msk})$.

$\underline{\mathsf{KGen}(\mathsf{id}, \mathsf{MSK}_{\mathsf{id}}, \mathsf{GID}, x).}$ Parse $\mathsf{MSK}_{\mathsf{id}}$ as $(\ell_{\mathsf{id}}, \mathsf{bfe.msk}_{\mathsf{id}})$. Sample $\mathsf{bfe.sk}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{BFE.KGen}(\mathsf{bfe.msk}_{\mathsf{id}}, (\mathsf{id}, \mathsf{GID}, x))$. Output $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} := \mathsf{bfe.sk}_{\mathsf{id},\mathsf{GID},x}$.

$\underline{\mathsf{Enc}(\{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{I}}, C).}$ Parse $\mathsf{MPK}_{\mathsf{id}}$ as $(\ell_{\mathsf{id}}, \mathsf{bfe.mpk}_{\mathsf{id}})$ for each $\mathsf{id} \in \mathcal{I}$. Let $n = |\mathcal{I}|$ and $\mathsf{id}_1, \ldots, \mathsf{id}_n$ be the increasing order of elements in $\mathcal{I}$.

- Sample $\mathsf{mafe.crs}, \{(\mathsf{mafe.mpk}_i, \mathsf{mafe.msk}_i)\}_i \leftarrow \mathsf{tMAFE.GSetup}(1^\lambda, 1^Q, 1^n, 1^s, 1^{\ell_{\mathsf{id}_1}}, \ldots, 1^{\ell_{\mathsf{id}_n}})$.
- Compute $\mathsf{mafe.ct} \leftarrow \mathsf{tMAFE.Enc}(\{\mathsf{mafe.mpk}_i\}_i, C)$,
- $\forall i \in [n], \mathsf{bfe.ct}_{\mathsf{id}_i} \leftarrow \mathsf{BFE.Enc}(\mathsf{bfe.mpk}_{\mathsf{id}_i}, \mathsf{tMAFE.KGen}(\cdot, \mathsf{mafe.msk}_i, \{\mathsf{mafe.mpk}_{i'}\}_{i'\in[n]}, \cdot, \cdot))$.

Output $\mathsf{CT} := (\mathsf{mafe.ct}, \mathcal{I}, \{\mathsf{bfe.ct}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{I}})$.

$\underline{\mathsf{Dec}(\{\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}\}_{\mathsf{id}\in\mathcal{I}}, \mathsf{CT}).}$ Parse $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}$ as $\mathsf{bfe.sk}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}$ for each $\mathsf{id} \in \mathcal{I}$ and $\mathsf{CT} := (\mathsf{mafe.ct}, \mathcal{I}', \{\mathsf{bfe.ct}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{I}'})$. If $\mathcal{I} \neq \mathcal{I}'$, abort and output $\perp$.

Otherwise, for each $\mathsf{id} \in \mathcal{I}$, compute $\mathsf{mafe.sk}_{\mathsf{id}} = \mathsf{BFE.Dec}(\mathsf{bfe.sk}_{\mathsf{id},\mathsf{GID},x}, \mathsf{bfe.ct}_{\mathsf{id}})$. Output $y := \mathsf{tMAFE.Dec}(\{\mathsf{mafe.sk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{I}}, \mathsf{mafe.ct})$.

Correctness. The correctness of the scheme follows from the correctness of BFE and tMAFE. Note that $\mathsf{tMAFE.KGen}(\cdot, \mathsf{mafe.msk}_i, \{\mathsf{mafe.mpk}_{i'}\}_{i' \in [n]}, \cdot, \cdot)$ is a valid $\mathsf{P/Poly}$ circuit. Hence, by correctness of BFE, we have that $\mathsf{mafe.sk}_{\mathsf{id}} = \mathsf{tMAFE.KGen}(\mathsf{id}, \mathsf{mafe.msk}_i, \{\mathsf{mafe.mpk}_{i'}\}_{i'}, \mathsf{GID}, x_{\mathsf{id}})$. Thus, $y = C(x_1, \ldots, x_n)$ by correctness of tMAFE.

Theorem 5.3. If tMAFE is a tMAFE scheme (Definition 4.1) for $\mathsf{P/Poly}$ circuits and $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda \in \mathbb{N}}$ and BFE is a BFE scheme for $\mathsf{P/Poly}$ circuits (Definition 3.3), then Construction 5.2 is an MAFE scheme for $\mathsf{P/Poly}$ circuits and $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda \in \mathbb{N}}$ (Definition 3.1).

Proof Sketch. We provide a sketch of the hybrid argument used to prove security of Construction 5.2 as follows.

Hybrid 0. This is the same as the real experiment from Definition 3.1. We use honest and stateless BFE and tMAFE algorithms to respond to $\mathcal{A}$.

Hybrid 1. In this hybrid, we will simulate the pre-challenge queries using $\mathsf{BFE.S}_0, \mathsf{BFE.S}_1$ for all queried authorities. That is, we will use $(\mathsf{BFE.S}_0, \mathsf{BFE.S}_1, \mathsf{BFE.Enc}, \mathsf{BFE.KGen})$ to respond to authority setup and secret key queries made to authorities in the pre-challenge query phase. We will still create authorities honestly in the post-challenge query phase. The output distribution will be identical to hybrid 0 from security of BFE.

Hybrid $2, j$. for $j \in [n+1]$. In this hybrid, we simulate the BFE instantiation for $\mathsf{id}_1, \ldots, \mathsf{id}_{j-1} \in \mathcal{I}$ using $\mathsf{BFE.Sim} = (\mathsf{BFE.S}_0, \mathsf{BFE.S}_1, \mathsf{BFE.S}_2, \mathsf{BFE.S}_3)$. The indistinguishability between these hybrids is argued from the security of BFE instantiation.

Hybrid 3. In this hybrid, we will simulate the instantiation of tMAFE. Now that all $\mathsf{id} \in \mathcal{I}$ are simulated, we will only be using secret keys and not he master secret keys of tMAFE. Hence, indistinguishability can be argued from security of tMAFE. This is the description of Sim from Definition 3.1.

We provide the full proof in Appendix C. □

Corollary 5.4. Assuming the existence of public-key encryption, there exists a MAFE scheme (with unbounded authorities) for $\mathsf{P/Poly}$ circuits and $\mathcal{GID}$.

Proof. The proof of this corollary follows from Corollary 4.4, and construction of BFE for $\mathsf{P/Poly}$ circuits assuming PKE from [AV19]. □

Corollary 5.5. Assuming the existence of one-way functions, there exists a secret-key MAFE scheme (with unbounded authorities) for $\mathsf{P/Poly}$ circuits and $\mathcal{GID}$ that satisfies full security, i.e, function- and message-privacy[5].

Proof. The proof of this corollary follows from Corollary 4.5, secret-key BFE for $\mathsf{P/Poly}$ circuits using one-way functions with full security from [BS18, AV19]. □

[5]As we work in the ciphertext-policy setting, message-privacy means that an adversary cannot distinguish the input used in KGen.

6 MAFE with Dynamic Collusions

Dynamic collusion model as introduced by [AMVY21, GGLW22] is a strengthening of the bounded query model [DKXY02, SS10, GLW12]. In this model encryptor can choose the collusion bound for a ciphertext. This allows the encryptor to set a desired collusion bound and provide more fine-grained control per encryption rather than adhere to strict collusion bound set during the system initiation. We refer the reader to [AMVY21, GGLW22] for an in-depth discussion on why the dynamic collusion model is much stronger and useful for BFE systems.

In this section, we construct an MAFE scheme in which an encryptor can dynamically set the query bound during the encryption and thus usher in all the advantages of the dynamic collusion setting to the multi-authority model for functional encryption systems. Concretely, the idea is that the GSetup, ASetup, and KGen algorithms of an MAFE scheme should be independent of the query bound Q and only the Enc and Dec algorithms should grow polynomially in Q. We formally define an MAFE scheme with dynamic collusions as follows.

Definition 6.1 (MAFE with Dynamic Collusions)**.** MAFE with dynamic collusions is an MAFE scheme where $\mathsf{GSetup}, \mathsf{ASetup}$, and KGen algorithms are independent of query bound Q. Enc takes the query bound 1^Q as input. The security definition is also defined similarly to Definition 3.1 where $\mathcal{A}$ specifies the query bound 1^Q during challenge query phase.

Taking a closer look at Construction 5.2, as we use a BFE scheme in ASetup and KGen. Hence, if we switch to BFE with dynamic collusions, we should realize a dynamic query MAFE scheme readily. As we are initiating tMAFE in Enc, we can depend polynomially in Q and proceed as before. However, there is a technical subtlety that avoids this straight forward construction.

The issue is that the output of BFE scheme is $\kappa_{\mathsf{id}} = \mathsf{poly}(\lambda, Q, |\mathsf{GID}|, \ell)$ and depends on the query bound Q. This is because the output of $\mathsf{tMAFE.KGen}$ grows polynomially in Q. The main idea in this section is that if we can construct a tMAFE scheme where the $\mathsf{GSetup}, \mathsf{KGen}$ and secret keys grow polynomially in $\lceil \log Q \rceil$, we can readily use BFE scheme with dynamic collusions to construct an MAFE scheme with dynamic collusions. More details are provided in Section 6.2. We call this object a "weakly optimal" tMAFE scheme (wotMAFE). This is analogous to weakly optimal BFE constructed in [GGLW22]. Definition of wotMAFE is as follows.

Definition 6.2 (wotMAFE)**.** A $\mathsf{wotMAFE}$ scheme $(\mathsf{GSetup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ is said to be a weakly optimal trusted MAFE scheme (wotMAFE) for circuit class $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ and $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda \in \mathbb{N}}$ if the running time of GSetup and KGen is upper bounded by $\mathsf{poly}(\lambda, s, \ell_1, \ldots, \ell_n, \lceil \log Q \rceil)$. Running time of Enc can grow with Q. Hence, we alter the syntax of GSetup and Enc where GSetup takes the query bound Q in binary whereas Enc takes the query bound in unary, 1^Q.

Our construction depends on garbled circuits, pseudorandom functions, and Identity-Based Encryption (IBE).

6.1 Weakly Optimal tMAFE from IBE and tMAFE

We construct a wotMAFE scheme using IBE and tMAFE below. The techniques used in Construction 6.3 are similar to the construction of weakly optimal BFE from IBE as seen in [GGL24, GGLW22]. In addition, as noted by [AMVY21], IBE is the minimal assumption for wotMAFE.

Construction 6.3 (wotMAFE)**.** We provide the construction of a wotMAFE scheme for $\mathsf{P/Poly}$ circuits and $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda\in\mathbb{N}}$ using a tMAFE scheme for $\mathsf{P/Poly}$ circuits (Definition 6.2), $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda\in\mathbb{N}}$ (Definition 4.1), an IBE scheme for identity space $[Q] \times \{0,1\}^{O(\log\lambda)} \times \{0,1\}$, garbling scheme $(\mathsf{Garble}, \mathsf{Eval})$ for $\mathsf{P/Poly}$ circuits (Definition A.1), and pseudorandom functions $\mathsf{PRF}_0 : \{0,1\}^\lambda \times \mathcal{GID} \to [Q], \mathsf{PRF}_1 : \{0,1\}^\lambda \times [Q] \to \{0,1\}^\lambda$ (Definition A.3) as follows.

$\underline{\mathsf{GSetup}(1^\lambda, 1^n, 1^s, 1^{\ell_1}, \ldots, 1^{\ell_n}, Q).}$ Set $\mathsf{CRS} := (\lambda, Q, n, s)$. Sample $K^{(0)}, K^{(1)} \xleftarrow{\$} \{0,1\}^\lambda$. For each $\mathsf{id} \in [n], (\mathsf{ibe.mpk}_{\mathsf{id}}, \mathsf{ibe.msk}_{\mathsf{id}}) \leftarrow \mathsf{IBE.Setup}(1^\lambda, 1^z)$ where $z = \lceil\log Q\rceil + \lceil\log L_{\mathsf{id}}\rceil + 1$ (L_{id} defined in KGen).

Set $\mathsf{MPK}_{\mathsf{id}} := (\ell_{\mathsf{id}}, \mathsf{ibe.mpk}_{\mathsf{id}})$, $\mathsf{MSK}_{\mathsf{id}} := (\ell_{\mathsf{id}}, \mathsf{ibe.msk}_{\mathsf{id}}, K^{(0)}, K^{(1)})$. Output $(\mathsf{CRS}, \{(\mathsf{MPK}_{\mathsf{id}}, \mathsf{MSK}_{\mathsf{id}})\}_{\mathsf{id}\in[n]})$.

$\underline{\mathsf{KGen}(\mathsf{id}, \mathsf{MSK}_{\mathsf{id}}, \{\mathsf{MPK}_{\mathsf{idx}}\}_{\mathsf{idx}\in[n]}, \mathsf{GID}, x).}$ Parse $\mathsf{MSK}_{\mathsf{id}}$ as $(\ell_{\mathsf{id}}, \mathsf{ibe.msk}_{\mathsf{id}}, K^{(0)}, K^{(1)})$ and $\forall\ \mathsf{idx} \in [n]$, $\mathsf{MPK}_{\mathsf{idx}}$ as $(\ell_{\mathsf{idx}}, \mathsf{ibe.mpk}_{\mathsf{idx}})$. Compute $\mathsf{Tag} = \mathsf{PRF}_0(K^{(0)}, \mathsf{GID})$ and $R = \mathsf{PRF}_1(K^{(1)}, \mathsf{Tag})$.

Using R deterministically sample $\mathsf{mafe.crs}, \{(\mathsf{mafe.mpk}_{\mathsf{idx}}, \mathsf{mafe.msk}_{\mathsf{idx}})\}_{\mathsf{idx}} \leftarrow \mathsf{tMAFE.GSetup}$ $(1^\lambda, 1^q, 1^n, 1^s, 1^{\ell_1}, \ldots, 1^{\ell_n}; R)$ where $q = \lambda$.

Sample secret key, $\mathsf{mafe.sk}_{\mathsf{id},\mathsf{GID},x} = \mathsf{tMAFE.KGen}(\mathsf{id}, \mathsf{mafe.msk}_{\mathsf{id}}, \{\mathsf{mafe.mpk}_{\mathsf{idx}}\}_{\mathsf{idx}}, \mathsf{GID}, x)$.

Let $L_{\mathsf{id}} = |\mathsf{mafe.mpk}_{\mathsf{id}}|$. $\forall\ i \in [L_{\mathsf{id}}]$, $\mathsf{ibe.sk}_{\mathsf{Tag},i} \leftarrow \mathsf{IBE.KGen}(\mathsf{ibe.msk}_{\mathsf{id}}, (\mathsf{Tag}, i, \mathsf{mafe.mpk}_{\mathsf{id}}[i]))$[6]. Output $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} := (\mathsf{Tag}, \mathsf{mafe.mpk}_{\mathsf{id}}, \mathsf{mafe.sk}_{\mathsf{id},\mathsf{GID},x}, \{\mathsf{ibe.sk}_{\mathsf{Tag},i}\}_{i\in[L_{\mathsf{id}}]})$.

$\underline{\mathsf{Enc}(1^Q, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}, C).}$ Parse $\mathsf{MPK}_{\mathsf{id}}$ as $(\ell_{\mathsf{id}}, \mathsf{ibe.mpk}_{\mathsf{id}})$ for each $\mathsf{id} \in [n]$. Let F be circuit that takes as input $\{\mathsf{mafe.mpk}_{\mathsf{idx}}\}_{\mathsf{idx}\in[n]}$ and outputs $\mathsf{tMAFE.Enc}(\{\mathsf{mafe.mpk}_{\mathsf{idx}}\}_{\mathsf{idx}}, C; R)$ for some hardwired randomness R. $\forall\ \mathsf{Tag} \in [Q]$,

- Sample $R^{\mathsf{Tag}} \xleftarrow{\$} \{0,1\}^\lambda$, garble F^{Tag}, $(\widetilde{F}^{\mathsf{Tag}}, \{w^{\mathsf{Tag}}_{\mathsf{id},i,b}\}) \leftarrow \mathsf{Garble}(1^\lambda, F^{\mathsf{Tag}})$ where $\mathsf{id} \in [n], i \in [L_{\mathsf{id}}], b \in \{0,1\}$[7].
- $\forall\ \mathsf{id} \in [n]$, $\mathsf{ibe.ct}^{\mathsf{Tag}}_{\mathsf{id},i,b} \leftarrow \mathsf{IBE.Enc}(\mathsf{ibe.mpk}_{\mathsf{id}}, (\mathsf{Tag}, i, b), w^{\mathsf{Tag}}_{\mathsf{id},i,b})$.

Output $\mathsf{CT} := (\widetilde{F}^{\mathsf{Tag}}, \{\mathsf{ibe.ct}^{\mathsf{Tag}}_{\mathsf{id},i,b}\}_{\mathsf{id},i,b})_{\mathsf{Tag}\in[Q]}$.

$\underline{\mathsf{Dec}(\{\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}\}_{\mathsf{id}\in[n]}, \mathsf{CT}).}$ Parse $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}$ as $(\mathsf{Tag}_{\mathsf{id}}, \mathsf{mafe.mpk}_{\mathsf{id}}, \mathsf{mafe.sk}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}, \{\mathsf{ibe.sk}_{\mathsf{Tag}_{\mathsf{id}},i}\}_i)$ for each $\mathsf{id} \in [n], i \in [L_{\mathsf{id}}]$. If all $\mathsf{Tag}_{\mathsf{id}}$ are not the same, abort and output $\perp$. Otherwise, set $\mathsf{Tag} = \mathsf{Tag}_1$.

Parse CT accordingly and use the Tag-th component, $\widetilde{F}^{\mathsf{Tag}}, \{\mathsf{ibe.ct}^{\mathsf{Tag}}_{\mathsf{id},i,b}\}$.

Perform IBE decryption $w^{\mathsf{Tag}}_{\mathsf{id},i,\mathsf{mafe.mpk}_{\mathsf{id}}[i]} = \mathsf{IBE.Dec}(\mathsf{ibe.sk}_{\mathsf{Tag},i}, \mathsf{ibe.ct}^{\mathsf{Tag}}_{\mathsf{id},i,\mathsf{mafe.mpk}_{\mathsf{id}}[i]})$. Evaluate garbled circuit, $\mathsf{mafe.ct} = \mathsf{Eval}(\widetilde{F}^{\mathsf{Tag}}, \{w^{\mathsf{Tag}}_{\mathsf{id},i,\mathsf{mafe.mpk}_{\mathsf{id}}[i]}\}_{\mathsf{id},i\in[L_{\mathsf{id}}]})$.

Output $y := \mathsf{tMAFE.Dec}(\{\mathsf{mafe.sk}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}\}_{\mathsf{id}\in[n]}, \mathsf{mafe.ct})$.

[6]Here, $\mathsf{mafe.mpk}_{\mathsf{id}}[i]$ is the i-th bit of $\mathsf{mafe.mpk}_{\mathsf{id}}$.
[7]For notational convenience, we assume that wires are pre-partitioned this way.

Correctness. The correctness of the scheme follows from the correctness of tMAFE, IBE, Garb, and deterministic nature of PRF. Concretely, by correctness of IBE, we get the wire labels for the $w^{\mathsf{Tag}}_{\mathsf{id},i,\mathsf{mafe.mpk_{id}}[i]}$ as we have secret keys for the identities $(\mathsf{Tag}, i, \mathsf{mafe.mpk_{id}}[i])$. Thus, by correctness of Garb, we have that $\mathsf{mafe.ct} = \mathsf{tMAFE.Enc}(\{\mathsf{mafe.mpk_{id}}\}_{\mathsf{id}\in[n]}, C; R^{\mathsf{Tag}})$. Finally, by correctness of tMAFE, we have $y = C(x_1, \ldots, x_n)$.

Efficiency. Note that running times of GSetup and KGen are only dependent on $\lceil \log Q \rceil$ as query bound q is set to λ and $[Q] \subseteq \{0,1\}^{\lceil \log Q \rceil}$. The running times of Enc and Dec are polynomial in Q.

Theorem 6.4. If tMAFE is a tMAFE scheme (Definition 4.1) for $\mathsf{P/Poly}$ circuits, $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda\in\mathbb{N}}$, IBE is an IBE scheme (Definition A.4) for identity space $[Q] \times \{0,1\}^{O(\log \lambda)} \times \{0,1\}$, $(\mathsf{Garble}, \mathsf{Eval})$ is a garbling scheme (Definition A.1) for $\mathsf{P/Poly}$ circuits, and $\mathsf{PRF}_0, \mathsf{PRF}_1$ are pseudorandom functions (Definition A.3), then Construction 6.3 is a wotMAFE scheme for $\mathsf{P/Poly}$ circuits, $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda\in\mathbb{N}}$.

Proof Sketch. We provide a sketch of the hybrid argument used to prove security of Construction 6.3 as follows.

Hybrid 0. This is the same as the real experiment from Definition 4.1. We use honest algorithms for $\mathsf{PRF}_0, \mathsf{PRF}_1, \mathsf{IBE}, \mathsf{tMAFE}$, and $(\mathsf{Garb}, \mathsf{Eval})$ to respond to adversary's queries.

Hybrid 1. In this hybrid, we will sample the Tags used in key generation as uniformly random strings early and use a dictionary $\mathcal{T}$ to set Tags consistently. The indistinguishability between these hybrid can be argued using the pseudorandomness of PRF.

Hybrid 2. In this hybrid, we will check if for any $\mathsf{Tag} \in [Q]$, the number of times Tag is sampled is more than λ. Hybrids 2 and 3 are statistically indistinguishable which follows from Lemma 5.2 of [GGLW22] or Lemma 3.4 of [AMVY21] or Claim 1 of [AV19].

Hybrid 3. In this hybrid, we will sample outputs of PRF_1 uniformly at the beginning and use them accordingly based on Tag. Indistinguishability follows from pseudorandomness of PRF_1.

Hybrid 4. In this hybrid, we will sample all tMAFE master public and secret keys early for each $\mathsf{Tag} \in [Q]$ and use them accordingly based on selected random Tag for key generation and encryption. Note that difference between hybrids 3 and 4 is the explicit sampling of randomness for $\mathsf{tMAFE.GSetup}$. Hence, these two hybrids are identically distributed.

Hybrid $5, k$. for $k \in [n+1]$. In this hybrid, we will set IBE encryptions of half the wire labels for the first $k-1$ users to be all zero strings based on $\mathsf{mafe.mpk}^{\mathsf{Tag}}_{\mathsf{id}}[i]$ for every Tag, i and $\mathsf{id} < k$. Indistinguishability can be argued using multi-challenge security of IBE as we never give out secret keys corresponding to $(1 - \mathsf{mafe.mpk}^{\mathsf{Tag}}_{\mathsf{id}}[i])$.

Hybrid $6, t$. for $t \in [Q+1]$. In this hybrid, we will simulate the garbled circuits for the first $t-1$ Tags. Indistinguishability can be argued by the security of garbled circuits as by the end of hybrid $5, n+1$, only half the wire labels are ever given out for all garbled circuits.

Hybrid $7, t$. for $t \in [Q+1]$. In this hybrid, we will simulate the Tag-th instantiation of MAFE for $\mathsf{Tag} < t$. When we simulate all garbled circuits, we are using $\mathsf{mafe.ct}^{\mathsf{Tag}} \leftarrow \mathsf{tMAFE.Enc}(\{\mathsf{mafe.}$

$\mathsf{mpk}_{\mathsf{id}}^{\mathsf{Tag}}\}_{\mathsf{id}}, C)$ in a black-box manner. Hence, we can proceed to simulating the instantiation readily and qrgue security using tMAFE security. $\mathbf{Hyb}_{7,Q+1}^{\mathcal{A}}(1^\lambda)$ is the description of Sim from Definition 6.2.

The full proof is provided in Appendix D. □

6.2 Dynamic MAFE from Minimal Assumptions

As discussed, essentially the same construction as Construction 5.2 when BFE is replaced with a dynamic query bounded BFE scheme for P/Poly circuits (Definition 3.4) and tMAFE by wotMAFE scheme for P/Poly circuits and $\mathcal{GID}$ (Definition 6.2) yields a dynamic query bounded MAFE scheme for P/Poly circuits and $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda\in\mathbb{N}}$. Although the output size of wotMAFE.KGen grows with $\lceil\log Q\rceil$, we can set $Q = 2^\lambda$ while initiating dynamic BFE and pad the output of wotMAFE secret keys to this length. Thus, we can eliminate the dependency of Q on GSetup, ASetup, KGen completely. On the other hand, in Enc, we get Q in unary and can proceed with instantiating wotMAFE as normal. For completeness, we provide the construction in Appendix E. The proof of the following theorem is immediate.

Theorem 6.5 (MAFE with Dynamic Collusions)**.** If there exists a BFE scheme for P/Poly circuits with dynamic collusions (Definition 3.4) and a wotMAFE scheme for P/Poly circuits and $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda\in\mathbb{N}}$ (Definition 6.2), there exists an MAFE scheme with dynamic collusions for P/Poly circuits and $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda\in\mathbb{N}}$ (Definition 6.1).

Corollary 6.6. Assuming the existence of IBE, there exists a general MAFE scheme (with unbounded authorities) for P/Poly circuits and $\mathcal{GID}$.

Proof. As IBE implies PKE and one-way functions, this corollary follows from Theorem 6.4 and Theorem 6.5. □

Corollary 6.7. Assuming the existence of one-way functions, there exists a secret-key general MAFE scheme (with unbounded authorities) for P/Poly circuits and $\mathcal{GID}$ with full security i.e, function- and message-privacy.

Proof. As secret-key IBE can be constructed from one-way functions, this corollary follows from Theorem 6.4 and secret-key message-hiding BFE for P/Poly circuits from one-way functions constructed in [BS18, AV19]. □

7 MAFE with Authority Corruptions

In this section, we provide the definition and construction for an MAFE scheme with authority corruptions. For an MAFE scheme with authority corruptions (cMAFE), the main difference is that it is secure even if some of the master secret keys for authorities are lost. The way the adversary corrupts authorities result in different definitions of (cMAFE). For instance, in an MAFE scheme with selective corruptions (sel-cMAFE), the adversary declares all corrupted authorities at the beginning and receives all master secret keys for these authorities along with the public parameters.

As authorities get corrupted, we need to ensure that the ciphertexts also reveal $C(X)$ for each $X \in \mathbf{X}_1 \times \ldots \mathbf{X}_n$ where $\mathbf{X}_{\mathsf{id}} = \{0,1\}^{\ell_{\mathsf{id}}}$ if id is corrupted. This also induces the restriction that sum of corrupted authorities' bits sum up to $O(\log\lambda)$. We expand more on this in Section 9. We provide the definition MAFE with authority corruptions below.

7.1 Definition

Definition 7.1 (cMAFE)**.** A cMAFE scheme, $(\mathsf{GSetup}, \mathsf{ASetup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ is said to be an MAFE scheme that supports χ corruptible bits for circuit class $\mathcal{C}$, and $\mathcal{GID}$ if it supports adaptive corruptions of authority master secret keys where the cumulative sum of input sizes of corrupted authorities is at most χ. We take the number of authorities n and χ as inputs to GSetup and work in the "bounded" authority setting of MAFE (Definition 3.2). Security of the scheme is defined analogously with the following important changes.

- $\mathcal{O}$ responds to KGen queries and in addition, responds to authority corruption queries where $\mathcal{A}$ sends $(\mathsf{Corr}, \mathsf{id})$ and receives $\mathsf{MSK}_{\mathsf{id}}$.
- Input for Sim in challenge query phase is defined as follows — $\mathcal{V} = \{(\mathsf{GID}, X, C(X)) : X \in \mathbf{X}_{\mathsf{GID},1} \times \ldots \mathbf{X}_{\mathsf{GID},n}$ for every GID that was used to query all authorities. If id-th authority corrupted, then $\mathbf{X}_{\mathsf{GID},\mathsf{id}} = \{0,1\}^{\ell_{\mathsf{id}}}\}$. Similarly, the simulator Sim queries C for every $X \in\in \mathbf{X}_{\mathsf{GID},1} \times \ldots \mathbf{X}_{\mathsf{GID},n}$ where for corrupted id, $\mathbf{X}_{\mathsf{GID},\mathsf{id}} = \{0,1\}^{\ell_{\mathsf{id}}}$ and a singleton $x_{\mathsf{GID},\mathsf{id}}$ otherwise.

A stateful PPT machine is admissible if it is an admissible adversary for MAFE. In addition, the adversary can corrupt authorities adaptively. For all corrupted authorities, $\sum_{\mathsf{id}} \ell_{\mathsf{id}} \leq \chi$ and $\chi(\lambda) = O(\log \lambda)$. Once id is corrupted, $\mathcal{A}$ should not issue for secret key queries to id.

7.2 MAFE with Adaptive Corruptions from HSS

In this section, we construct an MAFE scheme with adaptive authority corruptions using a homomorphic secret sharing scheme for $\mathsf{P/Poly}$ circuits. In particular, we construct a cMAFE scheme in which the number of authorities that can be corrupted, $k < n$ are known a-priori such that $\binom{n}{k} = \mathsf{poly}(\lambda)$. For instance, some choices of k could be $k = O(1), n = \mathsf{poly}(\lambda)$ or $n = O(\log \lambda), k = O(\log \lambda)$.

As mentioned in overview, the idea in this construction is to do a "brute-force" instantiation of MAFE schemes without authority corruptions such that we cover all possible k-size subset of n-authority corruptions. Note that by design as $\binom{n}{k} = \mathsf{poly}(\lambda)$, our algorithms will be efficient. In the i-th instantiation of this MAFE scheme, we will corrupt the i-th subset of $[n]$ of size k.

The idea is that as the valid adversary corrupts authorities, it must corrupt authorities one of these subsets. In the security argument, we will randomly choose a subset of $[n]$ of size k and rely on security of this instantiation. The reduction algorithm will be correct with a probability of $1/\binom{n}{k} = 1/\mathsf{poly}(\lambda)$. Thus, we successfully reduce the security of the scheme to security of one of the underlying MAFE schemes.

However, a subtle issue is that if we encrypt circuit C provided by adversary in every MAFE instantiation, security will not hold as we will not be able to protect C present in other instantiations. Hence, we need to "share" C among all the MAFE instantiation such that even if only one of the shares are honest, we can still argue security. This is where we use homomorphic secret sharing (HSS). Using HSS, we can share C among all the MAFE instantiations and evaluate on it to reveal the output. However, the current HSS schemes from literature only satisfy a weaker security guarantee where evaluations of honest shares cannot be leaked. We require this feature as the adversary can use the secret keys from MAFE to learn output of honest share's evaluation. To this end, we define and construct a new HSS scheme that facilitates just that. We formally define and construct such a HSS scheme from multi-key fully homomorphic encryption in Section F. Below, we formally define and construct (n,k)-cMAFE.

Definition 7.2 ((n,k)-cMAFE)**.** An (n,k)-cMAFE scheme for circuit class $\mathcal{C}$ and $\mathcal{GID}$ is defined similarly to Definition 7.1 and the maximum number of corruptible authorities, k are declared a-priori as an input to the GSetup algorithm. In addition, we require that $\binom{n}{k} = \mathsf{poly}(\lambda)$. We satisfy a the following notion of security in Construction 7.3. For any admissible adversary $\mathcal{A}$ that meets the admissibility criterion in Definition 7.1, there exists a PPT simulator Sim such that $\forall\ \lambda \in \mathbb{N}$,

$$\left\{ \begin{array}{l} Q, n, s, k, \chi, \mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}}, \\ \{\mathsf{MSK}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{K}}, \mathcal{Q}, \mathsf{CT}, b \end{array} : \begin{array}{l} Q, n, k, \chi, s, \ell_1, \dots, \ell_n \leftarrow \mathcal{A}(1^\lambda), \\ \mathsf{CRS} \leftarrow \mathsf{GSetup}(1^\lambda, 1^Q, 1^n, 1^s, 1^k, 1^\chi), \\ \forall\ \mathsf{id}, (\mathsf{MPK}_{\mathsf{id}}, \mathsf{MSK}_{\mathsf{id}}) \leftarrow \mathsf{ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}}), \\ C \leftarrow \mathcal{A}^{\mathcal{O}(\cdot,\cdot)}(\mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}}), \\ \mathsf{CT} \leftarrow \mathsf{Enc}(\{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}}, C), \\ b \leftarrow \mathcal{A}^{\mathcal{O}(\cdot,\cdot)}(\mathsf{CT}) \end{array} \right\}$$

$$\approx_c$$

$$\left\{ \mathsf{Sim}^{\mathcal{A}, C(\cdot)}(1^\lambda) \right\}$$

where $\mathcal{O}$ is a stateful oracle that responds to key generation using KGen and authority corruption queries. $\mathcal{O}$ adds the query and response to $\mathcal{Q}$ for each query made by $\mathcal{A}$. $\mathcal{K}$ is the set of all adaptively corrupted authorities.

Construction 7.3 ((n,k)-cMAFE)**.** We provide the construction of (n,k)-cMAFE for $\mathsf{P/Poly}$ circuits and $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda\in\mathbb{N}}$ (Definition 7.2) using an MAFE scheme for $\mathsf{P/Poly}$ circuits and $\mathcal{GID}$ (Definition 3.2), a HSS scheme for $\mathsf{P/Poly}$ circuits (Definition F.4) as follows.

$\underline{\mathsf{GSetup}(1^\lambda, 1^Q, 1^n, 1^s, 1^k, 1^\chi)}$**.** Let $\mathbb{I} = \left[\binom{n}{k}\right]$ and $\mathbb{J} = [2^\chi]$. We will assume some fixed ordering of $2^{[n]}\big|_k$[8] and say $\mathcal{K}_i \in 2^{[n]}\big|_k$ for $i \in \mathbb{I}$. For each $i \in \mathbb{I}, j \in \mathbb{J}$, instantiate an MAFE,

$$\forall\ i \in \mathbb{I}, j \in \mathbb{J}, \mathsf{crs}^{(i,j)} \leftarrow \mathsf{MAFE.GSetup}(1^\lambda, 1^Q, 1^{n-k}, 1^s)$$

We assume w.l.o.g that for any $i \in \mathbb{I}$, the authority identifiers are elements of the set $[n] \setminus \mathcal{K}_i$[9], i.e, for the (i,j)-th instantiation of MAFE, $\mathsf{id} \notin \mathcal{K}_i$[10]. Output $\mathsf{CRS} := (\mathsf{crs}^{(i,j)})_{i\in\mathbb{I}, j\in\mathbb{J}}$.

$\underline{\mathsf{ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}})}$**.** For each $i \in \mathbb{I}, j \in \mathbb{J}$, generate master public and secret keys for the (i,j)-th instantiation of MAFE if $\mathsf{id} \notin \mathcal{K}_i$, $(\mathsf{mpk}^{(i,j)}, \mathsf{msk}^{(i,j)}) \leftarrow \mathsf{MAFE.ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}})$. Output $\mathsf{MPK} := (\mathsf{mpk}^{(i,j)})_{i,j}$ and $\mathsf{MSK} := (\mathsf{msk}^{(i,j)})_{i,j}$.

$\underline{\mathsf{KGen}(\mathsf{id}, \mathsf{MSK}_{\mathsf{id}}, \{\mathsf{MPK}_{\mathsf{idx}}\}_{\mathsf{idx}\in[n]}, \mathsf{GID}, x)}$**.** Parse $\mathsf{MSK}_{\mathsf{id}}$ as $(\mathsf{mafe.msk}_{\mathsf{id}}^{(i,j)})_{i\in\mathbb{I}, j\in\mathbb{J}}$. For each $\mathsf{idx} \in [n]$, parse $\mathsf{MPK}_{\mathsf{idx}}$ as $(\mathsf{mafe.mpk}_{\mathsf{idx}}^{(i,j)})_{i\in\mathbb{I}, j\in\mathbb{J}}$.

Note that each $\mathsf{mafe.mpk}_{\mathsf{idx}}^{(i,j)}$ implicitly consists of information about ℓ_{idx}. Using $(\ell_1, \dots, \ell_n)$, define a corruptible set $\mathcal{K}' \subseteq [n]$ such that $\ell_{\mathsf{id}} = O(\log\lambda)$ for $\mathsf{id} \in \mathcal{K}'$.

Let $\mathbb{I}' = \{i : i \in \mathbb{I}, \mathcal{K}_i \subseteq \mathcal{K}', \sum_{\mathsf{id}\in\mathcal{K}_i} \ell_{\mathsf{id}} \le \chi\}$. Let $\mathbb{I}'_{\mathsf{id}} = \{i : i \in \mathbb{I}', \mathsf{id} \notin \mathcal{K}_i\}$. For every $i \in \mathbb{I}'_{\mathsf{id}}, j \in \mathbb{J}$,

$$\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j)} \leftarrow \mathsf{MAFE.KGen}(\mathsf{id}, \mathsf{msk}_{\mathsf{id}}^{(i,j)}, \{\mathsf{mpk}_{\mathsf{idx}}^{(i,j)}\}_{\mathsf{idx}}, \mathsf{GID}, x)$$

[8]Recall from Section 2 that $2^{\mathbf{A}}\big|_a$ is the set of all subsets of set $\mathbf{A}$ of size a.

[9]We use $\mathsf{id} \notin \mathcal{K}_i$ and $\mathsf{id} \in [n] \setminus \mathcal{K}_i$ interchangeably.

[10]This is done to avoid complicated notation required in the subsequent algorithms. We use this fact implicitly.

Output $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} := (\mathsf{sk}^{(i,j)}_{\mathsf{id},\mathsf{GID},x})_{i,j}$.

$\underline{\mathsf{Enc}(\{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}, C).}$ Parse $\mathsf{MPK}_{\mathsf{id}}$ as $(\mathsf{mpk}^{(i,j)}_{\mathsf{id}})_{i\in\mathbb{I},j\in\mathbb{J}}$ for each $\mathsf{id} \in [n]$. Parse $\mathcal{K}'$ similar to KGen. Share the circuit C using HSS for all $i \in \mathbb{I}'$, $\{\widetilde{C}_i\}_{i\in\mathbb{I}'} \leftarrow \mathsf{HSS}.\mathsf{Share}(1^\lambda, 1^Q, 1^{|\mathbb{I}'|}, C)$. For each $i \in \mathbb{I}'$, encrypt the circuit F_{ij} where this circuit takes $(\ell_1 + \ldots + \ell_n) - \chi$ bits as input and sets $\{x_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{K}_i} = j$ (truncations done appropriately) and computes $\mathsf{HSS}.\mathsf{Eval}(i, \widetilde{C}_i, x)$.

That is, for every $i \in \mathbb{I}', j \in \mathbb{J}, \mathsf{ct}^{(i,j)} \leftarrow \mathsf{MAFE}.\mathsf{Enc}(\{\mathsf{mpk}^{(i,j)}_{\mathsf{id}}\}_{\mathsf{id}\notin\mathcal{K}_i}, F_{ij})$. Output $\mathsf{CT} := (\mathsf{ct}^{(i,j)})_{i\in\mathbb{I}',j\in\mathbb{J}}$.

$\underline{\mathsf{Dec}(\{\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}\}_{\mathsf{id}\in[n]}, \mathsf{CT}).}$ Parse CT as $(\mathsf{mafe.ct}^{(i,j)})_{i\in\mathbb{I}',j\in\mathbb{J}}$ and $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}$ as $(\mathsf{mafe.sk}^{(i,j)}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}})_{i,j}$ for each $\mathsf{id} \in [n]$. Let $x = (x_1, \ldots, x_n)$. For each $i \in \mathbb{I}'$, partition x as $\{x_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{K}_i}$ and $\{x_{\mathsf{id}}\}_{\mathsf{id}\notin\mathcal{K}_i}$. Observe that we have secret keys for all $\{x_{\mathsf{id}}\}_{\mathsf{id}\notin\mathcal{K}_i}$. Decrypt the $(i, j = \{x_{\mathsf{id}}\}_{\mathsf{id}\notin\mathcal{K}_i})$-th instantiation of MAFE, $y_{i,j} = \mathsf{MAFE}.\mathsf{Dec}(\{\mathsf{sk}^{(i,j)}_{\mathsf{id},\mathsf{GID},x}\}_{\mathsf{id}\notin\mathcal{K}_i}, \mathsf{ct}^{(i,j)})$. Output $y := \mathsf{HSS}.\mathsf{Retrieve}(\{y_{i,j}\}_{i\in\mathbb{I}'})$.

Correctness. The correctness of the scheme follows from the correctness of MAFE and HSS schemes. In particular, $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}} = (\mathsf{sk}^{(i,j)}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}})_{i\in\mathbb{I}'_{\mathsf{id}},j\in\mathbb{J}}$ and $\mathsf{CT} = \{\mathsf{MAFE}.\mathsf{Enc}(\{\mathsf{mpk}^{(i,j)}_{\mathsf{id}}\}_{\mathsf{id}\notin\mathcal{K}_i}, \mathsf{HSS}.\mathsf{Eval}(i, \widetilde{C}_i, \cdot, j))\}_{i\in\mathbb{I}',j\in\mathbb{J}}$. By correctness of $(i, j = \{x_{\mathsf{id}}\}_{\mathsf{id}\notin\mathcal{K}_i})$-th instantiation of MAFE, $y_{i,j} = \mathsf{MAFE}.\mathsf{Dec}(\{\mathsf{sk}^{(i,j)}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}\}_{\mathsf{id}\notin\mathcal{K}_i}, \mathsf{ct}^{(i,j)}) = \mathsf{HSS}.\mathsf{Eval}(i, \widetilde{C}_i, x = (x_1, \ldots, x_n))$. Thus, by correctness of HSS, $y = C(x_1, \ldots, x_n)$.

Theorem 7.4. If HSS is a secure HSS scheme for $\mathsf{P/Poly}$ circuits (Definition F.4), MAFE is a secure MAFE scheme for $\mathsf{P/Poly}$ circuits and $\mathcal{GID}$ (Definition 3.2), then Construction 7.3 is a secure (n, k)-cMAFE scheme for $\mathsf{P/Poly}$ circuits and $\mathcal{GID}$.

Proof Sketch. We provide a sketch of the hybrid argument used to prove security of Construction 7.3 as follows.

Hybrid 0. This is the real experiment from Definition 7.2. We use the stateless MAFE and HSS algorithms to respond to queries from adversary.

Hybrid $1, \kappa$. For $\kappa \in [|\mathbb{J}| + 1]$. In this hybrid, we will randomly guess $i^* \xleftarrow{\$} \mathbb{I}$ and simulate the $(i^*, 1), \ldots, (i^*, \kappa - 1)$ instantiations of MAFE. The indistinguishability of hybrids can be argued using the security of MAFE. In addition, if the adversary corrupts any id that is not in the i^*-th element of $\mathbb{I}$, we abort. The indistinguishability is argued using a two fold argument – if we correctly guess the authority's corruption set, the challenger does not abort with $1/|\mathbb{I}|$ probability which by design is $1/\binom{n}{k} = 1/\mathsf{poly}(\lambda)$. If this happens, we then proceed to argue indistinguishability from MAFE security.

Hybrid 2. In this hybrid, we will simulate the HSS instantiation. Now that all MAFE instantiations for i^* (after correctly guessing at the beginning) are simulated, we would only be using the outputs of share evaluation. Thus, we can readily rely on the strong security of HSS to show indistinguishability. This is the description of Sim from Definition 7.2.

The full proof is in Appendix G. □

Remark 7.5 (Corruptions & dynamic collusions)**.** We remark that Construction 7.3 is agnostic to the query bound Q. That is, even if we provide the query bound during encryption, the same construction works *mutatis mutandis*. This is because of the black-box usage of MAFE. Hence, a similar construction yields an (n, k)-cMAFE scheme with dynamic collusions.

Remark 7.6 (Leakage Resilient FE)**.** We remark that our techniques provide a way to construct a leakage resilient FE scheme in which the master secret key can be corrupted block by block such that the number of bits of input it affects is bounded by χ.

8 Generic Compilers for Authority Corruptions

In this section, we show relations between various notions of authority corruptions in MAFE as defined in Section 7.1. In particular, we show that selective corruptions are good enough to achieve non-adaptive corruptions generically and adaptive corruptions via complexity leveraging. MAFE with selective and non-adaptive authority corruptions are defined as follows.

Definition 8.1 (sel-cMAFE)**.** A MAFE scheme with selectively corruptible authorities (sel-cMAFE) scheme for circuit class $\mathcal{C}$ and $\mathcal{GID}$ is defined similarly to Definition 7.1 except that the adversary will declare all the corruptions a-priori. That is, the adversary will send the set of corrupted authorities, $\mathcal{K}$ at the beginning and receive $\{\mathsf{MSK}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{K}}$ along with the public parameters. The adversary will not issue secret key queries for corrupted authorities.

Definition 8.2 (na-cMAFE)**.** A MAFE scheme with non-adaptive authority corruptions (na-cMAFE) scheme for circuit class $\mathcal{C}$ and $\mathcal{GID}$ is defined similarly to Definition 7.1 except that the adversary is prohibited to corrupt any authorities in the post-challenge query phase. That is, all authority corruptions should be made before sending the challenge circuit.

In order to achieve non-adaptive corruptions, we rely on a similar compiler as Section 5. The idea is that instead of a tMAFE scheme, we use a sel-cMAFE scheme to achieve non-adaptive corruptions[11]. As the pre-challenge simulator BFE $(\mathsf{BFE.S}_0, \mathsf{BFE.S}_1)$ behaves identically to an honest challenger, we can reveal $\mathsf{BFE.MSK}$ at most by the challenge query phase and proceed honestly. Thus, if all the corruptions are made before challenge query phase, we can readily rely on sel-cMAFE security as well.

For adaptive corruptions, we guess the authorities that will be corrupted a-priori and proceed with the security argument from there. The probability that our guess is correct is at most $2^{-O(\log^2 \lambda)}$.

We now construct a non-adaptively corruptible MAFE scheme ($\mathsf{na\text{-}cMAFE}$) using a selectively corruptible MAFE scheme ($\mathsf{sel\text{-}cMAFE}$).

Construction 8.3 ($\mathsf{na\text{-}cMAFE}$)**.** We provide the construction of $\mathsf{na\text{-}cMAFE}$ for $\mathsf{P/Poly}$ circuits and $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda\in\mathbb{N}}$ using a $\mathsf{sel\text{-}cMAFE}$ scheme for $\mathsf{P/Poly}$ circuits and $\mathcal{GID}$ (Definition 8.1) and a BFE scheme for $\mathsf{P/Poly}$ circuits (Definition 3.3) as follows.

$\underline{\mathsf{GSetup}(1^\lambda, 1^Q, 1^n, 1^s, 1^\chi).}$ Output $\mathsf{CRS} := (\lambda, Q, n, s, \chi)$.

$\underline{\mathsf{ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}}).}$ Sample keys $(\mathsf{bfe.mpk}, \mathsf{bfe.msk}) \leftarrow \mathsf{BFE.Setup}(1^\lambda, 1^Q, 1^{\lambda+|\mathsf{GID}|+\ell_{\mathsf{id}}}, 1^{\kappa_{\mathsf{id}}})$ where $\kappa_{\mathsf{id}} \leq \mathsf{poly}(\lambda, Q, |\mathsf{GID}|, \ell_{\mathsf{id}})$. Output $\mathsf{MPK} := (\ell_{\mathsf{id}}, \mathsf{bfe.mpk})$ and $\mathsf{MSK} := (\ell_{\mathsf{id}}, \mathsf{bfe.msk})$.

[11]Note that even in Construction 5.2, we could rely on selectively secure tMAFE to achieve the same result.

$\underline{\mathsf{KGen}(\mathsf{id}, \mathsf{MSK}_{\mathsf{id}}, \{\mathsf{MPK}_{\mathsf{idx}}\}_{\mathsf{idx}\in[n]}, \mathsf{GID}, x)}$**.** Parse $\mathsf{MSK}_{\mathsf{id}}$ as $(\ell_{\mathsf{id}}, \mathsf{bfe.msk}_{\mathsf{id}})$ and for all $\mathsf{idx} \in [n]$, $\mathsf{MPK}_{\mathsf{idx}}$ as $(\ell_{\mathsf{idx}}, \mathsf{bfe.mpk}_{\mathsf{idx}})$. Sample $\mathsf{bfe.sk}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{BFE.KGen}(\mathsf{bfe.msk}_{\mathsf{id}}, (\mathsf{id}, \mathsf{GID}, x))$. Output $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} := \mathsf{bfe.sk}_{\mathsf{id},\mathsf{GID},x}$.

$\underline{\mathsf{Enc}(\{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}, C)}$**.** Parse $\mathsf{MPK}_{\mathsf{id}}$ as $(\ell_{\mathsf{id}}, \mathsf{bfe.mpk}_{\mathsf{id}})$ for each $\mathsf{id} \in [n]$.

- Sample $\mathsf{mafe.crs} \leftarrow \mathsf{sel\text{-}cMAFE.GSetup}(1^\lambda, 1^Q, 1^n, 1^s, 1^\chi)$ and $\forall\, \mathsf{id} \in [n]$, $(\mathsf{mafe.mpk}_{\mathsf{id}}, \mathsf{mafe.msk}_{\mathsf{id}}) \leftarrow \mathsf{sel\text{-}cMAFE.ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}})$.
- Compute $\mathsf{mafe.ct} \leftarrow \mathsf{sel\text{-}cMAFE.Enc}(\{\mathsf{mafe.mpk}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}, C)$.
- $\forall\, \mathsf{id} \in [n]$, $\mathsf{bfe.ct}_{\mathsf{id}} \leftarrow \mathsf{BFE.Enc}(\mathsf{bfe.mpk}_{\mathsf{id}}, \mathsf{sel\text{-}cMAFE.KGen}(\cdot, \mathsf{mafe.msk}_{\mathsf{id}}, \{\mathsf{mafe.mpk}_{\mathsf{idx}}\}_{\mathsf{idx}}, \cdot, \cdot))$.

Output $\mathsf{CT} := (\mathsf{mafe.ct}, \{\mathsf{bfe.ct}_{\mathsf{id}}\}_{\mathsf{id}\in[n]})$.

$\underline{\mathsf{Dec}(\{\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}\}_{\mathsf{id}\in[n]}, \mathsf{CT})}$**.** Parse $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}$ as $\mathsf{bfe.sk}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}$ for each $\mathsf{id} \in [n]$ and $\mathsf{CT} := (\mathsf{mafe.ct}, \{\mathsf{bfe.ct}_{\mathsf{id}}\}_{\mathsf{id}\in[n]})$. For each $\mathsf{id} \in [n]$, compute $\mathsf{mafe.sk}_{\mathsf{id}} = \mathsf{BFE.Dec}(\mathsf{bfe.sk}_{\mathsf{id},\mathsf{GID},x}, \mathsf{bfe.ct}_{\mathsf{id}})$. Output $y := \mathsf{sel\text{-}cMAFE.Dec}(\{\mathsf{mafe.sk}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}, \mathsf{mafe.ct})$.

Correctness. The correctness of the scheme follows from the correctness of BFE and sel-cMAFE.

Theorem 8.4. If sel-cMAFE is a sel-cMAFE scheme (Definition 8.1) for P/Poly circuits and $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda\in\mathbb{N}}$ and BFE is a BFE scheme for P/Poly circuits (Definition 3.3), then Construction 8.3 is an na-cMAFE scheme for P/Poly circuits and $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda\in\mathbb{N}}$ (Definition 8.2).

Proof Sketch. The proof of this theorem is similar to the proof of Theorem 5.3. We provide a sketch of the hybrid argument used to prove security of Construction 8.3 as follows.

Hybrid 0. This is the real experiment from Definition 8.2. We use the stateless sel-cMAFE and BFE algorithms to respond to queries form adversary.

Hybrid 1. In this hybrid, we will simulate the pre-challenge queries using $\mathsf{BFE.S}_0$ and $\mathsf{BFE.S}_1$. That is, we will use $(\mathsf{BFE.S}_0, \mathsf{BFE.S}_1, \mathsf{BFE.Enc}, \mathsf{BFE.KGen})$ to respond to all secret key queries made to n authorities in the pre-challenge secret key query phase. Note that if id is corrupted, we can proceed with revealing its $\mathsf{MSK}_{\mathsf{id}}$ as the output distribution remains the same as hybrid 0 by BFE security.

Hybrid $2, j$. for $j \in [n+1]$. In this hybrid, we will simulate the honest BFE instantiations for the first $j-1$ authorities using BFE.Sim. And, we will use BFE.Enc and BFE.KGen algorithms for corrupted authorities which we will know by challenge query phase. The indistinguishability of hybrids can be argued from security of BFE.

Hybrid 3. In this hybrid, as we know all the corruptions by challenge query phase, we can simulate the sel-cMAFE instantiation. We can use the master secret keys to encrypt the honest sel-cMAFE.KGen algorithm using the corrupted authorities' BFE instantiations. Moreover, as no authority corruptions are allowed in the post-challenge query phase, indistinguishability can be argued using the security of sel-cMAFE. This is the description of Sim from Definition 8.2.

We omit full proof for brevity. □

Remark 8.5 (Adaptive Corruptions)**.** Assuming the quasi-polynomial security of $\mathsf{sel\text{-}cMAFE}$, we can construct a cMAFE scheme simply by guessing the corruption set a-priori. As we have $\sum_{\mathsf{id}} \ell_{\mathsf{id}} = \mathsf{poly}(\lambda)$ and out of which we need to guess $\chi = O(\log \lambda)$ bits that get corrupted. We will be correct with probability $1/\binom{\mathsf{poly}(\lambda)}{O(\log \lambda)} = 2^{-O(\log^2 \lambda)}$. Hence, by relying on sub-exponential security of $\mathsf{sel\text{-}cMAFE}$, we can get a cMAFE scheme generically. Moreover, we can construct a tight reduction using artificial abort techniques [Wat05, GY24] so that the resulting cMAFE scheme is secure against attackers running in times $2^{O(\log^2 \lambda)}$ with advantage at most $2^{-O(\log^2 \lambda)}$.

9 Lower Bounds on MAFE with Authority Corruptions

In this section, we provide lower bounds on the efficiency of MAFE schemes that support authority corruptions. Specifically, we show that if an MAFE scheme supports at most χ-bit corruptions, the size of its ciphertexts and master keys *must* grow with 2^χ. We show that any non-trivial efficiency in this regard, i.e, $2^{\chi(1-\xi)}$ for $\xi > 0$, will imply an indistinguishability obfuscation (iO) scheme[12]. This result implies that Construction 7.3 is optimal in certain regards.

The main idea as explained in overview is that MAFE schemes that satisfy a weaker notion of security such as static security (Definition 9.5) with a non-trivial efficiency guarantee as described above can be bootstrapped to an iO scheme. Here, by static security, we mean that adversary makes secret key and corruption queries before receiving the public parameters of the system.

We show that an MAFE scheme that obeys this definition implies an XiO scheme for $\mathsf{P}^{\log}/\mathsf{Poly}$ [LPST16] if it supports one GID query and a weakly succinct FE scheme for P/Poly [BV15] if it supports at most Q GID queries. Our implications are formally proved in Section 9.3 and Section 9.4. Note that an XiO scheme for $\mathsf{P}^{\log}/\mathsf{Poly}$ in conjunction with LWE implies an iO scheme whereas a weakly succinct FE scheme for P/Poly implies it generically. We provide the definitions of these preliminaries in Section 9.1 and formally define statically secure MAFE schemes in Section 9.2.

9.1 Preliminaries

We provide definitions of some preliminaries we use in this section.

Definition 9.1 (iO)**.** An $i\mathcal{O}$ scheme with polynomial-time algorithms $(\mathsf{Obf}, \mathsf{Eval})$ is said to be an indistinguishability obfuscation (iO) for the circuit class $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ if it satisfies the following properties.

Correctness. For any $\lambda \in \mathbb{N}, x, C \in \mathcal{C}_\lambda$,

$$\Pr[C(x) = \mathsf{Eval}(\widetilde{C}, x) : \widetilde{C} \leftarrow \mathsf{Obf}(1^\lambda, C)] = 1$$

Indistinguishability. For any stateful PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $\forall \lambda \in \mathbb{N}, C_0, C_1 \in \mathcal{C}_\lambda, |C_0| = |C_1|, \forall x, C_0(x) = C_1(x)$,

$$\Pr\left[\ b \leftarrow \mathcal{A}(\widetilde{C}) \quad : \begin{array}{l} b \xleftarrow{\$} \{0,1\}, (C_0, C_1) \leftarrow \mathcal{A}(1^\lambda), \\ \widetilde{C} \leftarrow \mathsf{Obf}(1^\lambda, C_b) \end{array} \right] \le \frac{1}{2} + \mathsf{negl}(\lambda)$$

[12]We consider indistinguishability-based security for our schemes in this section and simulation-based security is impossible for such schemes (cf. Remark 9.13).

Definition 9.2 (XiO)**.** An exponentially-efficient indistinguishability obfuscation (XiO) scheme $\mathsf{X}i\mathcal{O}$ as defined by [LPST16] is an $i\mathcal{O}$ scheme for the circuit class $\mathcal{C} = \left\{\mathcal{C}_\lambda : \{0,1\}^{n(\lambda)} \to \{0,1\}^{m(\lambda)}\right\}_{\lambda\in\mathbb{N}}$ that satisfies Definition 9.1 and the following property.

Non-trivial Efficiency. There exists a constant $\xi > 0$ such that for any $\lambda \in \mathbb{N}$, circuit $C \in \mathcal{C}_\lambda$, the probabilistic algorithm Obf runs in time $\mathsf{poly}(\lambda, |C|, 2^n)$ and output size $|\widetilde{C}| = \mathsf{poly}(\lambda, |C|) \cdot 2^{n(1-\xi)}$.

Definition 9.3 (Circuit class $\mathsf{P}^{\log}/\mathsf{Poly}$, [LPST16])**.** $\mathsf{P}^{\log}/\mathsf{Poly}$ is a subclass of P/Poly circuits that takes $O(\log\lambda)$ many bits as input. The output of any circuit $C \in \mathsf{P}^{\log}/\mathsf{Poly}$ is polynomially bounded. Note that $\mathsf{X}i\mathcal{O}$ is efficient for $\mathsf{P}^{\log}/\mathsf{Poly}$.

Definition 9.4 (sFE, [BV15])**.** An sFE scheme $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ is said to be a weakly size-succinct functional encryption (sFE) scheme for circuit class $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ if it satisfies the following properties.

Correctness. For any $\lambda \in \mathbb{N}, Q = Q(\lambda)$, and any $C_q \in \mathcal{C}_\lambda$ for $q \in [Q]$, any x,

$$\Pr\left[\ y = C_q(x) \ : \ \begin{array}{l} (\mathsf{MPK}, \{\mathsf{SK}_q\}_q) \leftarrow \mathsf{Setup}(1^\lambda, \{C_q\}_q), \\ \mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{MPK}, x), y = \mathsf{Dec}(\mathsf{SK}_q, \mathsf{CT}) \end{array} \right] = 1$$

Weak size-succinctness. The size of the encryption circuit, $|\mathsf{Enc}(\mathsf{MPK}, \cdot)| \le Q^{1-\xi} \cdot \mathsf{poly}(\lambda, |C|)$ where $\xi > 0$ and $|C|$ is the maximum circuit size for $\mathcal{C}_\lambda$.

Security. For any stateful PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $\forall\ \lambda \in \mathbb{N}$,

$$\Pr\left[\ b = b' \ : \ \begin{array}{l} \forall\ q \in [Q], \{C_q\}_q \leftarrow \mathcal{A}(1^\lambda), b \xleftarrow{\$} \{0,1\}, \\ (\mathsf{MPK}, \{\mathsf{SK}_q\}_q) \leftarrow \mathsf{Setup}(1^\lambda, \{C_q\}_q), \\ (x_0, x_1) \leftarrow \mathcal{A}(\mathsf{MPK}, \{\mathsf{SK}_q\}_q), \\ \mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{MPK}, x_b), b' \leftarrow \mathcal{A}(\mathsf{CT}) \end{array} \right] \le \frac{1}{2} + \mathsf{negl}(\lambda)$$

9.2 Definitions

We provide the definitions of primitives with non-trivial efficiency for MAFE with authority corruptions.

Syntax. A static Q-GID MAFE (stMAFE) scheme for the circuit class $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ and $\mathcal{GID} = \{\mathsf{GID}_\lambda\}_{\lambda\in\mathbb{N}}$ that can handle corruption of χ bits with non-trivial efficiency consists of algorithms $(\mathsf{GSetup}, \mathsf{ASetup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ defined similar to Definition 7.1.

Definition 9.5 (stMAFE)**.** An stMAFE scheme $(\mathsf{GSetup}, \mathsf{ASetup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ is said to be an stMAFE scheme if it satisfies the following properties.

Correctness. For any $\lambda \in \mathbb{N}, Q = Q(\lambda), n = n(\lambda), s = s(\lambda), k < n, \chi = \chi(\lambda), C \in \mathcal{C}_\lambda, \mathsf{GID} \in \mathcal{GID}_\lambda, \forall\ \mathsf{id} \in [n], \ell_{\mathsf{id}} = \ell_{\mathsf{id}}(\lambda)$,

$$\Pr\left[\ \begin{array}{l} C(x_1, \ldots, x_n) = \\ \mathsf{Dec}(\{\mathsf{SK}_{\mathsf{id},\mathsf{GID},x}\}_{\mathsf{id}}, \mathsf{CT}) \end{array} \ : \ \begin{array}{l} \mathsf{CRS} \leftarrow \mathsf{GSetup}(1^\lambda, 1^Q, 1^n, 1^s, 1^k, 1^\chi), \\ \forall\ \mathsf{id}, (\mathsf{MPK}_{\mathsf{id}}, \mathsf{MSK}_{\mathsf{id}}) \leftarrow \mathsf{ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}}), \\ \mathsf{CT} \leftarrow \mathsf{Enc}(\{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}}, C) \end{array} \right] = 1$$

where $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{KGen}(\mathsf{id}, \mathsf{MSK}_{\mathsf{id}}, \{\mathsf{MPK}_{\mathsf{idx}}\}_{\mathsf{idx}\in[n]}, \mathsf{GID}, x)$.

Non-trivial efficiency. The size of the encryption circuit, $|\mathsf{Enc}(\{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}}, \cdot)|$ is at most $(Q \cdot 2^{\chi})^{1-\xi} \cdot \mathsf{poly}(\lambda, \ell_1, \ldots, \ell_n, |C|)$.

Static IND **security.** For any stateful PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $\forall\ \lambda \in \mathbb{N}$,

$$\Pr\left[b \leftarrow \mathcal{A}(\mathsf{CT}) \quad : \quad \begin{array}{l} \left(\begin{array}{c} Q, n, s, k, \chi, \mathcal{K}, \{\ell_{\mathsf{id}}\}_{\mathsf{id}\in[n]}, \\ \{\mathsf{GID}_q, \{x_{q,\mathsf{id}}\}_{\mathsf{id}\notin\mathcal{K}}\}_{q\in[Q]} \end{array} \right) \leftarrow \mathcal{A}(1^\lambda), \\ \forall\ \mathsf{id}, q, (\mathsf{MPK}_{\mathsf{id}}, \mathsf{MSK}_{\mathsf{id}}) \leftarrow \mathsf{ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}}), \\ \mathsf{SK}_{\mathsf{id},q} \leftarrow \mathsf{KGen}(\mathsf{id}, \mathsf{MSK}_{\mathsf{id}}, \{\mathsf{MPK}_{\mathsf{idx}}\}_{\mathsf{idx}}, \mathsf{GID}_q, x_{q,\mathsf{id}}), \\ (C_0, C_1) \leftarrow \mathcal{A}\left(\begin{array}{c} \mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}, \\ \{\mathsf{MSK}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{K}}, \{\mathsf{SK}_{\mathsf{id},q}\}_{q,\mathsf{id}} \end{array} \right), \\ b \xleftarrow{\$} \{0,1\}, \mathsf{CT} \leftarrow \mathsf{Enc}(\{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}}, C_b) \end{array} \right]$$
$$\leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where $\mathsf{CRS} \leftarrow \mathsf{GSetup}(1^\lambda, 1^Q, 1^n, 1^s, 1^k, 1^\chi)$ and KGen is defined similarly. (C_0, C_1) are such that for any $\mathsf{GID} \in \{\mathsf{GID}_q\}, x \in \mathbf{X}_{\mathsf{GID},1} \times \ldots \times \mathbf{X}_{\mathsf{GID},n}$, $C_0(x) = C_1(x)$ where $\mathbf{X}_{\mathsf{GID},\mathsf{id}} = \{0,1\}^{\ell_{\mathsf{id}}}$ if $\mathsf{id} \in \mathcal{K}$ and $\mathbf{X}_{\mathsf{GID},\mathsf{id}} = \{x_{\mathsf{GID},\mathsf{id}}\}$ otherwise. In addition, $\sum_{\mathsf{id}\in\mathcal{K}} \ell_{\mathsf{id}} \leq \chi$.

Definition 9.6 (fst1MAFE)**.** A fst1MAFE scheme (GSetup, ASetup, KGen, Enc, Dec) is said to be a fully static fixed-corruptible 1-GID MAFE scheme for circuit class $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ and a single GID if in the above definition, $Q = 1$ (and we ignore it in syntax) and $\mathcal{A}$ sends challenge circuits (C_0, C_1) along with secret key queries. In addition, we make the following change to efficiency.

Non-trivial efficiency. The sizes of CRS, $\mathsf{MSK}_{\mathsf{id}}$, $\mathsf{MPK}_{\mathsf{id}}$ for any $\mathsf{id} \in [n]$, $\mathsf{SK}_{\mathsf{id},x}, x \in \{0,1\}^{\ell_{\mathsf{id}}}$, and CT for the circuit C are at most $\mathsf{poly}(\lambda, \ell_1, \ldots, \ell_n, |C|) \cdot 2^{\chi(1-\xi)}$ where $\xi > 0$. In addition, the running times of GSetup, ASetup, KGen, Enc are at most $\mathsf{poly}(\lambda, \ell_1, \ldots, \ell_n, |C|, 2^\chi)$.

9.3 XiO from fst1MAFE

Construction 9.7 ($\mathsf{X}i\mathcal{O}$)**.** We provide the construction of an XiO scheme (Definition 9.2) for $\mathsf{P}^{\log}/\mathsf{Poly}$ circuits (Definition 9.3) $\mathcal{C} = \{\mathcal{C}_\lambda : \{0,1\}^{n(\lambda)} \to \mathcal{O}_\lambda\}_{\lambda\in\mathbb{N}}$ from an fst1MAFE scheme for P/Poly circuits (Definition 9.6) that can handle n corruptions as follows.

$\underline{\mathsf{Obf}(1^\lambda, C)}$**.** Let $C : \{0,1\}^{n(\lambda)} \to \mathcal{O}_\lambda$ for $n(\lambda) = O(\log\lambda)$. Sample an fst1MAFE instantiation with $N = 2$ authorities, $k = 1$ corrupts where $\ell_1 = n, \ell_2 = \mathsf{poly}(\lambda), \chi = \ell_1$,

crs $\leftarrow$ fst1MAFE.GSetup$(1^\lambda, 1^N, 1^{s'}, 1^n, 1^k, 1^\chi)$. $\forall$ id $\in \{1,2\}$, $(\mathsf{mpk}_{\mathsf{id}}, \mathsf{msk}_{\mathsf{id}}) \leftarrow$ fst1MAFE. ASetup$(\mathsf{id}, 1^{\ell_{\mathsf{id}}})$.

Sample a secret key for a random $\alpha \in \{0,1\}^{\ell_2}$, $\mathsf{sk}_{2,\alpha} \leftarrow$ fst1MAFE.KGen$(2, \mathsf{msk}_2, \alpha)$. Now, generate a ciphertext for the circuit C' of size $s' = \Theta(|C|)$, ct $\leftarrow$ fst1MAFE.Enc$(\mathsf{mpk}_1, \mathsf{mpk}_2, C')$. C' simply ignores the second input.

Output $\widetilde{C} := (\mathsf{crs}, \mathsf{mpk}_1, \mathsf{mpk}_2, \mathsf{msk}_1, \mathsf{sk}_{2,\alpha}, \mathsf{ct})$.

$\underline{\mathsf{Eval}(\widetilde{C}, x)}$**.** Parse $\widetilde{C}$ as $(\mathsf{crs}, \mathsf{mpk}_1, \mathsf{mpk}_2, \mathsf{msk}_1, \mathsf{sk}_{2,\alpha}, \mathsf{ct})$. Run KGen for x, $\mathsf{sk}_{1,x} \leftarrow$ fst1MAFE.KGen$(1, \mathsf{msk}_1, \{\mathsf{mpk}_{\mathsf{id}}\}_{\mathsf{id}}, x)$.

Output $y :=$ fst1MAFE.Dec$(\mathsf{sk}_{1,x}, \mathsf{sk}_{2,\alpha}, \mathsf{ct})$.

Correctness. This follows from the correctness of $\mathsf{fst1MAFE}$.

Non-trivial Efficiency. As all algorithms of $\mathsf{fst1MAFE}$ used in Obf run in $\mathsf{poly}(\lambda, n, \ell_2, |C|, 2^n)$, running time of Obf is $\mathsf{poly}(\lambda, n, \ell_2, |C|, 2^n) \leq \mathsf{poly}(\lambda, |C|, |C|, 2^n) = \mathsf{poly}(\lambda, |C|, 2^n)$. The size of $\widetilde{C}$ can be similarly shown to be $\mathsf{poly}(\lambda, |C|) \cdot 2^{n(1-\xi')}$ where $\xi' > 0$.

Theorem 9.8. If $\mathsf{fst1MAFE}$ is a fully static 1-GID MAFE scheme for $\mathsf{P/Poly}$ circuits that can handle $O(\log \lambda)$ corruptions (Definition 9.6), then Construction 9.7 is an $\mathsf{X}i\mathcal{O}$ scheme for $\mathsf{P}^{\log}/\mathsf{Poly}$ circuits (Definition 9.2).

Proof. We prove the security of Construction 9.7 by showing a reduction from an adversary that breaks $\mathsf{X}i\mathcal{O}$ security to an adversary that breaks $\mathsf{fst1MAFE}$ security. Assume that there exists an adversary $\mathcal{A}$ that runs in time t against $\mathsf{X}i\mathcal{O}$ that can distinguish between $\widetilde{C}_0$ and $\widetilde{C}_1$ with advantage ϵ. Here, C_0 and C_1 are such that $|C_0| = |C_1|$ and $\forall\, x, C_0(x) = C_1(x)$. We prove the security of $\mathsf{X}i\mathcal{O}$ by constructing a reduction $\mathcal{B}$ between $\mathcal{A}$ and a challenger Chal for $\mathsf{fst1MAFE}$ for $\mathsf{P/Poly}$ circuits. The description of $\mathcal{B}$ is as follows.

$\underline{\mathcal{B}^{\mathsf{Chal}}(1^\lambda).}$ $\mathcal{A}$ sends C_0, C_1 to $\mathcal{B}$. Let $C_0, C_1 : \{0,1\}^n \to \mathcal{O}_\lambda$ for $n = O(\log \lambda)$. Set $\ell_2 := \mathsf{poly}(\lambda), N = 2, k = 1, \chi = n$, and sample $\alpha \xleftarrow{\$} \{0,1\}^{\ell_2}$. Send $N, s', k, \chi, \mathcal{K} = \{1\}, \alpha, (C_0', C_1')$ to Chal. Parse the information sent by Chal and set $\widetilde{C} := (\mathsf{crs}, \mathsf{mpk}_1, \mathsf{mpk}_2, \mathsf{msk}_1, \mathsf{sk}_{2,\alpha}, \mathsf{ct})$. Run $\mathcal{A}$ on $\widetilde{C}$. Output whatever $\mathcal{A}$ outputs. Note that the running time of $\mathcal{B}$ is polynomial in the running time of $\mathcal{A}$ and λ. In addition, $\mathcal{B}$ acts as a valid adversary for Chal and a valid challenger for $\mathcal{A}$. Advantage of $\mathcal{B}$ is exactly ϵ. □

Thus, if we have a sub-exponentially secure $\mathsf{fst1MAFE}$ for $\mathsf{P/Poly}$ circuits, we can build a sub-exponentially secure $\mathsf{X}i\mathcal{O}$ scheme for $\mathsf{P}^{\log}/\mathsf{Poly}$ circuits. The following theorem from [LPST16] shows that we can build sub-exponentially secure $i\mathcal{O}$ for $\mathsf{P/Poly}$ circuits from $\mathsf{X}i\mathcal{O}$ for $\mathsf{P}^{\log}/\mathsf{Poly}$.

Theorem 9.9 ([LPST16])**.** Assuming the existence of sub-exponentially secure $\mathsf{X}i\mathcal{O}$ for $\mathsf{P}^{\log}/\mathsf{Poly}$ and sub-exponentially secure LWE, there exists a sub-exponentially secure $i\mathcal{O}$ scheme for $\mathsf{P/Poly}$ circuits (Definition 9.1).

9.4 Weakly Size-Succinct FE from stMAFE

Construction 9.10 (sFE)**.** We provide the construction of a weakly size-succinct FE scheme for $\mathsf{P/Poly}$ circuits (Definition 9.4) from an stMAFE scheme (Definition 9.5) for $\mathsf{P/Poly}$ circuits and $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda\in\mathbb{N}}$ as follows.

$\underline{\mathsf{Setup}(1^\lambda, \{C_q\}_{q\in[Q]})}$. W.l.o.g., assume that all circuits are of size s and output size $m = m(\lambda)$. We will use $\ell_1 = \lceil \log m \rceil, \ell_2 = s, N = 2, k = 1, \chi = \ell_1, \mathcal{K} := \{1\}$ and s' to be the size of the circuit from Enc.

Generate an stMAFE instantiation, $\mathsf{crs} \leftarrow \mathsf{stMAFE.GSetup}(1^\lambda, 1^Q, 1^N, 1^{s'}, 1^k, 1^\chi)$. For each $\mathsf{id} = \{1, 2\}$, $(\mathsf{mpk}_{\mathsf{id}}, \mathsf{msk}_{\mathsf{id}}) \leftarrow \mathsf{stMAFE.ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}})$.

Sample Q GIDs randomly from $\mathcal{GID}_\lambda$. That is $\forall\, q \in [Q], \mathsf{GID}_q \leftarrow \mathcal{GID}_\lambda$. Sample secret keys for the $\mathsf{id} = 2$, for GID_q, C_q, $\mathsf{sk}_{2,q,C_q} \leftarrow \mathsf{KGen}(2, \mathsf{msk}_2, \mathsf{GID}_q, C_q)$.

Output $\mathsf{MPK} := (\mathsf{crs}, \mathsf{mpk}_1, \mathsf{mpk}_2)$ and $\mathsf{SK}_q := (\mathsf{MPK}, \mathsf{msk}_1, \mathsf{GID}_q, C_q, \mathsf{sk}_{2,q,C_q})$.

$\underline{\mathsf{Enc}(\mathsf{MPK}, x)}$. Parse MPK as $(\mathsf{crs}, \{\mathsf{mpk}_{\mathsf{id}}\}_{\mathsf{id}})$. Construct the circuit $F = F(x, \cdot, \cdot)$. Here, $F(x, i, C)$ outputs the i-th bit of $C(x)$ for $i \in [m]$. Note that the size of circuit F is $O(s^2)$. Generate ciphertext for F using stMAFE, $\mathsf{ct} \leftarrow \mathsf{stMAFE.Enc}(\mathsf{mpk}_1, \mathsf{mpk}_2, F)$ and output $\mathsf{CT} := \mathsf{ct}$.

$\underline{\mathsf{Dec}(\mathsf{SK}, \mathsf{CT})}$. Parse CT as ct and SK as $(\mathsf{crs}, \mathsf{mpk}_1, \mathsf{mpk}_2, \mathsf{msk}_1, \mathsf{GID}_q, C_q, \mathsf{sk}_{2,q,C})$. Run stMAFE key generation m times for $\mathsf{id} = 1$, for $\mathsf{sk}_{1,q,i} \leftarrow \mathsf{stMAFE.KGen}(1, \mathsf{msk}_1, \{\mathsf{mpk}_{\mathsf{id}}\}_{\mathsf{id}}, \mathsf{GID}_q, i)$. Output $y := (y_1, \ldots, y_m)$ where $\forall\, i \in [m], y_i = \mathsf{stMAFE.Dec}(\mathsf{sk}_{1,q,i}, \mathsf{sk}_{2,q,C}, \mathsf{ct})$.

Correctness. This follows from the correctness of stMAFE.

Weak size-succinctness. The size of $\mathsf{Enc}(\mathsf{MPK}, \cdot)$ is at most

$$\begin{aligned} |F| + |\mathsf{stMAFE.Enc}(\{\mathsf{mpk}_{\mathsf{id}}\}_{\mathsf{id}}, \cdot)| &\le s^2 + (Q2^{\lceil \log m \rceil})^{1-\xi} \cdot \mathsf{poly}(\lambda, s, \lceil \log m \rceil, s) \\ &\le Q^{1-\xi}\mathsf{poly}(\lambda, s) \end{aligned}$$

Theorem 9.11. If stMAFE is a secure static fixed-corruptible Q-GID MAFE scheme for $\mathsf{P/Poly}$ circuits and $\mathcal{GID}$ (Definition 9.5), then Construction 9.10 is a secure weakly size-succinct FE scheme for $\mathsf{P/Poly}$ circuits.

Proof. We prove the security of Construction 9.10 by showing a reduction between any adversary that breaks the security of sFE and a challenger for stMAFE. Assume that there exists a t-time adversary $\mathcal{A}$ that runs in time t against sFE as per Definition 9.4 with advantage ϵ. We prove the security of sFE by constructing a reduction $\mathcal{B}$ between $\mathcal{A}$ and a challenger Chal for stMAFE for $\mathsf{P/Poly}$ circuits. The description of $\mathcal{B}$ is as follows.

$\underline{\mathcal{B}^{\mathsf{Chal}}(1^\lambda)}$. $\mathcal{A}$ sends size-s circuits with output size m, $\{C_q\}_{q\in[Q]}$. Set $\ell_1 := \lceil \log m \rceil, \ell_2 = s$, $N = 2, k = 1, \chi = \ell_1$, $\mathcal{K} = \{1\}$, and $s' = O(s^2)$. Sample $\mathsf{GID}_q \xleftarrow{\$} \mathcal{GID}_\lambda$ and sets $x_{2,q} := C_q$ for $q \in [Q]$. Send $Q, N, s', k, \chi, \ell_1, \ell_2, \mathcal{K}, \{\mathsf{GID}_q, \{x_{2,q}\}\}_q$ to Chal. Receive $(\mathsf{crs}, \mathsf{mpk}_1, \mathsf{mpk}_2, \mathsf{msk}_1, \{\mathsf{sk}_{2,q,C_q}\}_q)$ from Chal. Set $\mathsf{MPK} := (\mathsf{crs}, \mathsf{mpk}_1, \mathsf{mpk}_2)$, $\mathsf{SK}_q := (\mathsf{MPK}, \mathsf{msk}_1, \mathsf{GID}_q, C_q, \mathsf{sk}_{2,q,C_q})$, and send $(\mathsf{MPK}, \{\mathsf{SK}_q\}_q)$ to $\mathcal{A}$. $\mathcal{A}$ sends (x_0, x_1). Construct $F_0 = F_0(x_0, \cdot, \cdot)$, $F_1 = F_1(x_1, \cdot, \cdot)$ where $F_b(x_b, i, C)$ outputs the i-th bit of $C(x_b)$ for $b \in \{0, 1\}$. Send (F_0, F_1) to Chal. Receive CT from Chal and run $\mathcal{A}$ on CT. Output whatever $\mathcal{A}$ outputs. Note that the running time of $\mathcal{B}$ is polynomial in the running time of $\mathcal{A}$ and λ. In addition, $\mathcal{B}$ acts as a valid adversary for Chal and a valid challenger for $\mathcal{A}$. Advantage of $\mathcal{B}$ is exactly ϵ. $\square$

Thus, if we have a sub-exponentially secure $\mathsf{fst1MAFE}$ for $\mathsf{P/Poly}$ circuits, we can build a sub-exponentially secure sFE for $\mathsf{P/Poly}$ circuits. The following theorem from [BV15] shows that we can build sub-exponentially secure $i\mathcal{O}$ for $\mathsf{P/Poly}$ circuits from sFE for $\mathsf{P/Poly}$ circuits.

Theorem 9.12 ([BV15]). Assuming the existence of sub-exponentially secure sFE for $\mathsf{P/Poly}$ circuits, there exists a sub-exponentially secure $i\mathcal{O}$ scheme for $\mathsf{P/Poly}$ circuits.

Remark 9.13 (Impossibilities for SIM-security). By an incompressibility argument similar to [BSW11, AGVW13], SIM-secure $\mathsf{fst1MAFE}$ and stMAFE are impossible in the standard model.

10 (poly, Q)-MA-ABFE Definition

In this section, we provide the definition of (poly, Q)-MA-ABFE with varying security levels. An MA-ABFE scheme can be thought of as an MAFE scheme for the function class that can be split into a predicate and a circuit. That is, $\mathcal{F} = \{(\mathcal{P}_\lambda, \mathcal{C}_\lambda)\}_{\lambda\in\mathbb{N}}$ comprises of a tuple of predicates in $\mathcal{P} = \{\mathcal{P}_\lambda : \mathcal{X}_{\lambda,1} \times \ldots \times \mathcal{X}_{\lambda,n} \to \{0,1\}\}_{\lambda\in\mathbb{N}}$ and circuits in $\mathcal{C} = \{\mathcal{C}_\lambda : \mathcal{Y}_{\lambda,1} \times \ldots \times \mathcal{Y}_{\lambda,n} \to \mathcal{O}_\lambda\}_{\lambda\in\mathbb{N}}$. The functionality of $\mathcal{F}$ is defined as follows — for any $F = (P \in \mathcal{P}_\lambda, C \in \mathcal{C}_\lambda), (x_1, \ldots, x_n) \in \mathcal{X}_1 \times \ldots \times \mathcal{X}_n$, and $(y_1, \ldots, y_n) \in \mathcal{Y}_1 \times \ldots \times \mathcal{Y}_n$,

$$F((x_1, y_1), \ldots, (x_n, y_n)) = C(y_1, \ldots, y_n) \text{ if and only if } P(x_1, \ldots, x_n) = 1$$

The idea of MA-ABFE as explained in Section 2 is to facilitate unbounded secret key collusion per ciphertext as opposed to the bounded collusion as in MAFE. Because of this, our syntax for MA-ABFE remains similar to an MAFE scheme's syntax. For KGen, we take the input as a tuple (x, y) and for Enc, we take (P, C) as input where P is a predicate and C is a circuit. In addition, GSetup algorithm takes maximum input size $L = |y_{\mathsf{id}}|$ for any id and maximum size of predicate s_{abe} as additional inputs. Similarly correctness is altered to two cases, when $P(x_1, \ldots, x_n) = 0/1$. We summarize these ideas formally in the definition below.

Definition 10.1 (MA-ABFE)**.** A multi-authority attribute-based functional encryption (MA-ABFE) scheme is an MAFE scheme for predicate class $\mathcal{P}$, circuit class $\mathcal{C}$, and $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda\in\mathbb{N}}$. We define a new security definition, namely, (poly, Q)-bounded security which is defined below. We also make a few changes to the "bounded" authority variant of MAFE syntax as follows.

Syntax. The GSetup algorithm takes the maximum length of any of circuit inputs L, size of predicate s_{abe} as additional inputs. Input for KGen is split into x and y for predicate and circuit respectively. Enc takes the predicate $P \in \mathcal{P}_\lambda$ and circuit $C \in \mathcal{C}_\lambda$ as inputs.

Correctness. This is defined for two types of queries, namely a satisfying query $(P(x_1, \ldots, x_n) = 1)$ and unsatisfying query $(P(x_1, \ldots, x_n) = 0)$. Dec outputs $C(y_1, \ldots, y_n)$ for satisfying queries and $\perp$ for unsatisfying queries.

(poly, Q)**-Bounded Security.** For any admissible adversary $\mathcal{A}$, there exists a stateful simulator Sim such that $\forall\ \lambda \in \mathbb{N}$,

$$\left\{ \mathcal{A}^{\mathsf{KGen}(\cdot,\cdot,\cdot,\cdot)}(\mathsf{CT}) : \begin{array}{l} Q, n, L, s_{\mathsf{abe}}, s_{\mathsf{fe}}, \ell_1, \ldots, \ell_n \leftarrow \mathcal{A}(1^\lambda), \\ \mathsf{CRS} \leftarrow \mathsf{GSetup}(1^\lambda, 1^Q, 1^n, 1^{s_{\mathsf{abe}}}, 1^{s_{\mathsf{fe}}}), \\ \forall\ \mathsf{id} \in [n], (\mathsf{MPK}_{\mathsf{id}}, \mathsf{MSK}_{\mathsf{id}}) \leftarrow \mathsf{ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}}), \\ P, C \leftarrow \mathcal{A}^{\mathsf{KGen}(\cdot,\cdot,\cdot,\cdot)}(\mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}}), \\ \mathsf{CT} \leftarrow \mathsf{Enc}(\{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{I}}, P, C) \end{array} \right\}$$

$$\approx_c$$

$$\left\{ \mathcal{A}^{\mathsf{Sim}^{C(\cdot)}(\cdot,\cdot,\cdot,\cdot)}(\mathsf{CT}) : \begin{array}{l} Q, n, L, s_{\mathsf{abe}}, s_{\mathsf{fe}}, \ell_1, \ldots, \ell_n \leftarrow \mathcal{A}(1^\lambda), \\ \mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]} \leftarrow \mathsf{Sim}(1^\lambda, 1^Q, 1^n, 1^{s_{\mathsf{abe}}}, 1^{s_{\mathsf{fe}}}), \\ \forall\ \mathsf{id} \in [n], \mathsf{MPK}_{\mathsf{id}} \leftarrow \mathsf{Sim}(\mathsf{id}, 1^{\ell_{\mathsf{id}}}), \\ P, C \leftarrow \mathcal{A}^{\mathsf{Sim}(\cdot,\cdot,\cdot,\cdot)}(\mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}}), \\ \mathsf{CT} \leftarrow \mathsf{Sim}(P, 1^{|C|}, \mathcal{V}) \end{array} \right\}$$

where $\mathcal{V} = \{(\mathsf{GID}, Y = \{y_{\mathsf{GID},\mathsf{id}}\}_{\mathsf{id} \in [n]}, C(Y)) :$ for every GID that was used to query all $\mathsf{id} \in [n]$ in pre-challenge query phase such that $P(x_{\mathsf{GID},1}, \ldots, x_{\mathsf{GID},n}) = 1\}$. Similarly, Sim queries $C(\cdot)$ with $Y = (\{y_{\mathsf{GID},\mathsf{id}}\}_{\mathsf{id} \in [n]})$ and receives $C(y_{\mathsf{GID},1}, \ldots, y_{\mathsf{GID},|\mathcal{I}|})$ if $P(x_{\mathsf{GID},1}, \ldots, x_{\mathsf{GID},n}) = 1$. KGen is an oracle that responds to the following type of queries — $\mathcal{A}$ queries for secret keys using $(\mathsf{id}, \mathsf{GID}, x, y)$ and KGen responds with $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x,y} \leftarrow \mathsf{KGen}(\mathsf{id}, \mathsf{MSK}_{\mathsf{id}}, \{\mathsf{MPK}_{\mathsf{idx}}\}_{\mathsf{idx} \in [n]}, \mathsf{GID}, x, y)$.

A stateful PPT machine $\mathcal{A}$ is admissible if it makes polynomially many queries to KGen with the restriction that the number of queries such that P is satisfied (satisfying queries) are at most Q. In addition, $\mathcal{A}$ will make at most one ciphertext query with circuit $C \in \mathcal{C}_\lambda$ of size at most s adaptively.

We now provide definitions for a few variants of (poly, Q)-MA-ABFE below.

Definition 10.2 (Non-adaptive (poly, Q)-MA-ABFE)**.** We say that a (poly, Q)-bounded MA-ABFE scheme $(\mathsf{GSetup}, \mathsf{ASetup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ is non-adaptively secure if in the above security game, satisfying queries are not allowed in post-challenge query phase.

Definition 10.3 ($(\mathsf{poly}, 1)$-MA-ABFE)**.** We say that an MA-ABFE scheme $(\mathsf{GSetup}, \mathsf{ASetup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ is a $(\mathsf{poly}, 1)$-bounded MA-ABFE if in the above security game, $Q = 1$. In addition, we ignore Q in syntax.

Definition 10.4 (Non-adaptive $(\mathsf{poly}, 1)$-MA-ABFE)**.** We say that an MA-ABFE scheme $(\mathsf{GSetup}, \mathsf{ASetup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ satisfies non-adaptive $(\mathsf{poly}, 1)$-bounded security if it satisfies both Definition 10.2 and Definition 10.3.

11 (poly, 1)-MA-ABFE from MA-ABE and Garbled Circuits

In this section, we show how to construct an MA-ABFE scheme that is $(\mathsf{poly}, 1)$-bounded secure. That is, the adversary is allowed to query *one* set of queries such that $P(x_1, \ldots, x_n)$ and in that case, learn $C(y_1, \ldots, y_n)$. The main hurdle in this construction is that such a query can be made adaptively or non-adaptively or in a way that half the authorities are queried in the pre-challenge phase and the other half in post-challenge phase. In all of these cases, we need to construct a simulator that only requires $C(y_1, \ldots, y_n)$ to simulate the transcript of communication between a challenger and an adversary.

As explained in overview, we rely on the template put forth by [GY24] and proceed in 3 stages. Firstly, we construct a non-adaptively secure $(\mathsf{poly}, 1)$-bounded MA-ABFE scheme in Section 11.2 where the satisfying query can only be made completely in the pre-challenge query phase. We construct this by relying on garbled circuits. For now, assume that $y_{\mathsf{id}} = 1$ for each id. We proceed somewhat similar to [SS10] where we replace the PKE instantiation by MA-ABE. Let us elaborate. We instantiate $2n$ MA-ABE schemes for n authorities and $b \in \{0, 1\}$. For $i \in [n]$, id-th authority issues a secret key for each $b \in \{0, 1\}$ if $i \neq \mathsf{id}$ and issues secret key for $(i, b) = (\mathsf{id}, y_{\mathsf{id}})$-th MA-ABE scheme. In encryption, we will garble the circuit C and encrypt $w_{i,b}$ under (i, b)-th MA-ABE with predicate P. Note that we provide all secret keys from all authorities for (i, y_i)-th MA-ABE instantiations for $i \in [n]$. Thus, adversary will only learn half the wire labels. The security of this scheme is then straight-forward where if the adversary make a satisfying query, we can simulate the garbled circuit and rely on MA-ABE security to hide the other half wire labels.

Secondly, we construct a non-committing variant of MA-ABE. In a non-committing MA-ABE, apart from the regular MA-ABE schemes, there is a special way to "fake" a ciphertext and later "reveal" it to be an encryption for message m. The idea is that if an adversary makes a satisfying query in the post-challenge phase, it shouldn't be able to guess whether it received a honestly generate $\mathsf{CT}, \{\mathsf{SK}_{\mathsf{id}}\}_{\mathsf{id}}$ pair or a faked pair $\widetilde{\mathsf{CT}}, \{\widetilde{\mathsf{SK}}_{\mathsf{id}}\}_{\mathsf{id}}$. Note that adversary can query $n-1$ authorities for the satisfying GID in the pre-challenge phase and last one in post-challenge phase. Even then, the security should hold. We proceed somewhat similar to [GVW12] where we replace the PKE instantiation by MA-ABE. Let us elaborate. Consider that we are working with single bit messages. We instantiate $2n$ MA-ABE schemes and generate secret key for id-th authority for every $(i \neq \mathsf{id}, b)$-th MA-ABE schemes and $(\mathsf{id}, \rho_{\mathsf{id}})$-th MA-ABE where $b_{\mathsf{id}} \xleftarrow{\$} \{0,1\}$. To encrypt, we sample $R_1, \ldots, R_n \xleftarrow{\$} \{0,1\}$ and R_i in the (i,b)-th MA-ABE system and set $\widetilde{R} = \oplus_{\mathsf{id}} R_{\mathsf{id}} \oplus m$. Correctness of the scheme follows immediately as we can decrypt $R_1, \ldots, R_n$ and learn m from there. This structure allows us to encrypt 0 and 1 randomly in $(i,0)$ and $(i,1)$-th MA-ABE systems. And using the secret keys learn $R_{\mathsf{id}} := \rho_{\mathsf{id}}$. Thus, when the last authority (say n) is queried for a satisfying query, we can set $\rho_n = \oplus_{i \neq n} \rho_i \oplus \widetilde{R} \oplus m$ and reveal $\widetilde{R}$ to be $\sum_i \rho_i \oplus m$. Thus, we can equivocate the ciphertext when the last authority is queried using this trick and the distributions will be computationally indistinguishable by security of MA-ABE. Full construction of this scheme is provided in Section 11.3.

Finally, to construct a $(\mathsf{poly}, 1)$-bounded MA-ABFE in Section 11.4 where we combine both the non-committing MA-ABE and non-adaptive $(\mathsf{poly}, 1)$-bounded MA-ABFE schemes to achieve the desired result. We generate a ciphertext for non-adaptive $(\mathsf{poly}, 1)$-bounded MA-ABFE and encrypt it in the non-committing MA-ABE scheme. This way, we can fake the ciphertext and reveal it only when a satisfying query is made, we can reveal the ciphertext and rely on non-adaptive $(\mathsf{poly}, 1)$-bounded MA-ABFE security to simulate the ciphertext with only $C(y_1, \ldots, y_n)$. The main ingredient we use in our constructions in this section is an MA-ABE scheme which we define in Section 11.1 as a special case of $(\mathsf{poly}, 0)$-bounded MA-ABFE for a special circuit class.

11.1 Preliminaries

We recall the definition of $\mathsf{MA\text{-}ABE}$ but as a special case of $\mathsf{MA\text{-}ABFE}$. This is done in order to maintain consistency between the definitions of MAFE, $\mathsf{MA\text{-}ABFE}$, and $\mathsf{MA\text{-}ABE}$.

Definition 11.1 (MA-ABE)**.** A multi-authority attribute-based encryption is a special case of MA-ABFE (Definition 10.1) that satisfies $(\mathsf{poly}, 0)$-bounded security for predicate class $\mathcal{P}$ and $\mathcal{C}$ contains of circuits that outputs a message $m \in \{0,1\}^*$[13]. In syntax, the GSetup algorithm does not take Q, s_{fe} as inputs and we forgo y as input to KGen as it essentially does not matter. On the other hand Enc takes the message m as input rather than circuit C. We formulate an indistinguishability-based definition as follows.

IND Security. For any admissible adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\lambda)$ such

[13] Recall that $\mathsf{MA\text{-}ABE}$ can encrypt messages of arbitrary sizes using hybrid encryption.

that,

$$\Pr\left[\begin{array}{lcl} b \leftarrow & & (n, s, \ell_1, \ldots, \ell_n) \leftarrow \mathcal{A}(1^\lambda), \\ \mathcal{A}^{\mathcal{E}(\cdot,\cdot)}(\mathsf{CT}) & : & \mathsf{CRS} \leftarrow \mathsf{GSetup}(1^\lambda, 1^n, 1^{s_{\mathsf{abe}}}), \forall\ \mathsf{id} \in [n], \\ & & (\mathsf{MPK}_{\mathsf{id}}, \mathsf{MSK}_{\mathsf{id}}) \leftarrow \mathsf{ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}}), \\ & & (P, m_0, m_1) \leftarrow \mathcal{A}^{\mathcal{E}(\cdot,\cdot)}(\mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}}), \\ & & b \xleftarrow{\$} \{0,1\}, \mathsf{CT} \leftarrow \mathsf{Enc}(\{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}}, P, m_b) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where $\mathcal{E}$ oracle responds to secret key queries as follows — $\mathcal{A}$ sends $(\mathsf{KGen}, (\mathsf{id}, \mathsf{GID}, x))$ and receives $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{KGen}(\mathsf{id}, \mathsf{MSK}_{\mathsf{id}}, \{\mathsf{MPK}_{\mathsf{idx}}\}_{\mathsf{idx}\in[n]}, \mathsf{GID}, x)$ from $\mathcal{E}$. A stateful PPT machine $\mathcal{A}$ is said to be admissible if it makes polynomially many queries to $\mathcal{E}$ and makes one KGen query per GID to an authority and for every set of $X = (\{x_{\mathsf{GID},\mathsf{id}}\}_{\mathsf{id}}), P(X) = 0$.

11.2 Non-Adaptive (poly, 1)-MA-ABFE

Construction 11.2 (na1MA-ABFE). We construct a non-adaptive $(\mathsf{poly}, 1)$-bounded MA-ABFE scheme for predicate class $\mathcal{P} = \{\mathcal{P}_\lambda\}_{\lambda\in\mathbb{N}}$, P/Poly circuits, $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda\in\mathbb{N}}$ (Definition 10.4) using an MA-ABE scheme for predicate class $\mathcal{P} = \{\mathcal{P}_\lambda\}_{\lambda\in\mathbb{N}}$, $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda\in\mathbb{N}}$ with no corruptions (Definition 11.1), and a garbling scheme (Garble, Eval) for P/Poly circuits (Definition A.1) as follows.

$\underline{\mathsf{GSetup}(1^\lambda, 1^n, 1^L, 1^{s_{\mathsf{abe}}}, 1^{s_{\mathsf{fe}}})}$. For each $i \in [n], j \in [L], b \in \{0,1\}$, sample $\mathsf{crs}^{(i,j,b)} \leftarrow \mathsf{MA\text{-}ABE.GSetup}$ $(1^\lambda, 1^n, 1^{s_{\mathsf{abe}}})$. Output $\mathsf{CRS} := (n, L, s_{\mathsf{abe}}, s_{\mathsf{fe}}, (\mathsf{crs}^{(i,j,b)})_{i,j,b})$.

$\underline{\mathsf{ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}})}$. For each $i \in [n], j \in [L], b \in \{0,1\}$, sample $(\mathsf{mpk}_{\mathsf{id}}^{(i,j,b)}, \mathsf{msk}_{\mathsf{id}}^{(i,j,b)}) \leftarrow \mathsf{MA\text{-}ABE.ASetup}$ $(\mathsf{id}, 1^{\ell_{\mathsf{id}}})$. Output $\mathsf{MPK}_{\mathsf{id}} := (\mathsf{mpk}_{\mathsf{id}}^{(i,j,b)})_{i,j,b}$ and $\mathsf{MSK}_{\mathsf{id}} := (\mathsf{msk}_{\mathsf{id}}^{(i,j,b)})_{i,j,b}$.

$\underline{\mathsf{KGen}(\mathsf{id}, \mathsf{MSK}_{\mathsf{id}}, \{\mathsf{MPK}_{\mathsf{idx}}\}_{\mathsf{idx}\in[n]}, \mathsf{GID}, x, y)}$. Parse $\mathsf{MSK}_{\mathsf{id}}$ as $(\mathsf{msk}_{\mathsf{id}}^{(i,j,b)})_{i,j,b}$, $\mathsf{MPK}_{\mathsf{idx}}$ as $(\mathsf{mpk}_{\mathsf{idx}}^{(i,j,b)})_{i,j,b}$ for $\mathsf{idx} \in [n], i \in [n], j \in [L], b \in \{0,1\}$. Pad y such that $|y| = L$. For each $i \in [n]$,

- If $i \neq \mathsf{id}$, for each j, b, $\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j,b)} \leftarrow \mathsf{MA\text{-}ABE.KGen}(\mathsf{id}, \mathsf{msk}_{\mathsf{id}}^{(i,j,b)}, \{\mathsf{mpk}_{\mathsf{idx}}^{(i,j,b)}\}_{\mathsf{idx}}, \mathsf{GID}, x)$.
- If $i = \mathsf{id}$, for each j, $\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j,y[j])} \leftarrow \mathsf{MA\text{-}ABE.KGen}(\mathsf{id}, \mathsf{msk}_{\mathsf{id}}^{(i,j,y[j])}, \{\mathsf{mpk}_{\mathsf{idx}}^{(i,j,y[j])}\}_{\mathsf{idx}}, \mathsf{GID}, x)$[14].

Output $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x,y} = \left(\{\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j,b)}\}_{i\neq\mathsf{id},j,b}, \{\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j,y[j])}\}_{i=\mathsf{id},j}, x, y\right)$

$\underline{\mathsf{Enc}(\{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}, P, C)}$. Parse $\mathsf{MPK}_{\mathsf{id}}$ as $(\mathsf{mpk}_{\mathsf{id}}^{(i,j,b)})_{i,j,b}$ for $\mathsf{id} \in [n], i \in [n], j \in [L], b \in \{0,1\}$.

Garble the circuit C[15], $(\widetilde{C}, \{w_{i,j,b}\}_{i,j,b}) \leftarrow \mathsf{Garble}(1^\lambda, 1^{nL}, C)$.

Compute for each i, j, b, $\mathsf{ct}^{(i,j,b)} \leftarrow \mathsf{MA\text{-}ABE.Enc}(\{\mathsf{mpk}_{\mathsf{id}}^{(i,j,b)}\}, P, w_{i,j,b})$.

Output $\mathsf{CT} := (\widetilde{C}, (\mathsf{ct}^{(i,j,b)})_{i,j,b})$.

$\underline{\mathsf{Dec}(\{\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}},y_{\mathsf{id}}}\}_{\mathsf{id}\in[n]}, \mathsf{CT})}$. Parse CT as $(\widetilde{C}, (\mathsf{ct}^{(i,j,b)})_{i,j,b})$ and $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}},y_{\mathsf{id}}}$ as $\left(\{\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j,b)}\}_{i\neq\mathsf{id},j,b},\right.$ $\left.\{\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j,y[j])}\}_{i=\mathsf{id},j}, x_{\mathsf{id}}, y_{\mathsf{id}}\right)$ for $\mathsf{id} \in [n], i \in [n], j \in [L], b \in \{0,1\}$. For each i, j, $w_{i,j,y_i[j]} =$ $\mathsf{MA\text{-}ABE.Dec}(\{\mathsf{sk}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}^{(i,j,y_i[j])}\}_{\mathsf{id}\in[n]}, \mathsf{ct}^{(i,j,y_i[j])})$. Output $z = \mathsf{Eval}(\widetilde{C}, \{w_{i,j,y_i[j]}\})$.

[14]By $y[j]$ we denote the j-th bit of y

[15]For notational convenience, we assume that wire labels are pre-partitioned this way.

Correctness. The correctness follows from correctness of $\mathsf{MA\text{-}ABE}$ and $(\mathsf{Garble}, \mathsf{Eval})$. In particular, for each $i \in [n], j \in [L]$, we only have a complete set of secret keys for the instances $(i, j, y_i[j])$. Hence, using these and by correctness of $\mathsf{MA\text{-}ABE}$, we get half the wire labels $\{w_{i,j,y_i[j]}\}_{i,j}$. From here, we evaluate the garbled circuit $\widetilde{C}$ to obtain $z = C(y_1, \ldots, y_n)$.

Theorem 11.3. If $\mathsf{MA\text{-}ABE}$ is an MA-ABE scheme for predicate class $\mathcal{P} = \{\mathcal{P}_\lambda\}_{\lambda \in \mathbb{N}}$ and $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda \in \mathbb{N}}$ without corruptions (Definition 11.1), $(\mathsf{Garble}, \mathsf{Eval})$ is a garbling scheme for $\mathsf{P/Poly}$ circuits (Definition A.1), then Construction 11.2 is a non-adaptive $(\mathsf{poly}, 1)$-bounded MA-ABFE scheme for predicate class $\mathcal{P} = \{\mathcal{P}_\lambda\}_{\lambda \in \mathbb{N}}$, $\mathsf{P/Poly}$ circuits, and $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda \in \mathbb{N}}$ (Definition 10.4).

Proof Sketch. We provide a sketch of the hybrid argument used to prove security of Construction 11.2 as follows.

Hybrid 0. This is the real experiment with Chal from Definition 10.4. We use the $\mathsf{MA\text{-}ABE}$ honestly to encrypt all the wire labels.

Hybrid $1, \iota, \gamma$. for $\iota \in [n], \gamma \in [L]$. In this hybrid, if the adversary submits a satisfying query, we replace the wire label $(w_{i,j,1-y_i[j]})$ with all zero string of same length for every $i < \iota, j < \gamma$. Note that we only give $n-1$ secret keys for the $(i, j, 1-y_i[j])$-th instantiation and as such the adversary cannot decrypt even though $P(x_1, \ldots, x_n) = 1$. Thus, we can readily rely on $\mathsf{MA\text{-}ABE}$ security to argue indistinguishability of these hybrids.

Hybrid 2. In this hybrid, half the wire labels are replaced with all zero strings. This is similar to the final hybrid from previous phase. Indistinguishability between hybrids $1, n, L$ and 2 can be argued from security of $\mathsf{MA\text{-}ABE}$.

Hybrid 3. In this hybrid, if the adversary submits a satisfying query (which can only happen in pre-challenge query phase and thus we know the whole input $(y_1, \ldots, y_n)$), we simulate the garbled circuit instantiation. As there are only half the wire labels being used in encryption, security comes from security of $(\mathsf{Garb}, \mathsf{Eval})$. This is the description of Sim from Definition 10.4.

We provide full proof in Appendix H.

□

11.3 Non-Committing MA-ABE

A non-committing MA-ABE scheme can be viewed as an MA-ABFE that satisfies adaptive $(\mathsf{poly}, 1)$-bounded security for the circuit class that outputs a hard-coded message $m \in \{0,1\}^L$ for every input. We present another definition using MA-ABE that can "fake" a ciphertext without using message m and produce a secret key that "reveals" the ciphertext to be encrypting m.

Definition 11.4 (ncMA-ABE)**.** A non-committing MA-ABE scheme is an MA-ABE scheme (Definition 11.1) equipped with two additional algorithms, $\mathsf{Fake}, \mathsf{Reveal}$. Using Fake, we can simulate a ciphertext without the message $m \in \{0,1\}^L$ and when the adversary makes a satisfying query, we can use Reveal to adaptively equivocate the ciphertext to reveal m. In the syntax of the scheme GSetup takes L as an input. Correctness is defined similarly to Definition 11.1. The syntax of $\mathsf{Fake}, \mathsf{Reveal}$ and security are defined as follows.

$\mathsf{Fake}(\{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}, P) \to (\widetilde{\mathsf{CT}}, \mathsf{aux})$. The probabilistic faking algorithm takes as input the set of all master public keys $\{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}$, predicate $P \in \mathcal{P}_\lambda$ and outputs a ciphertext $\widetilde{\mathsf{CT}}$ and auxiliary information aux.

$\mathsf{Reveal}(\mathsf{id}, \mathsf{MPK}_{\mathsf{id}}, \mathsf{MSK}_{\mathsf{id}}, \{(\mathsf{MPK}_{\mathsf{idx}}, \rho_{\mathsf{idx}})\}_{\mathsf{idx}\neq\mathsf{id}}, \mathsf{GID}, x, \mathsf{aux}, m) \to (\widetilde{\mathsf{SK}}_{\mathsf{id},\mathsf{GID},x})$. The possibly randomized revealing algorithm takes as input master secret of id-th authority $\mathsf{MSK}_{\mathsf{id}}$, master public keys from all authorities, $\mathsf{MPK}_{\mathsf{id}}, \{\mathsf{MPK}_{\mathsf{idx}}\}_{\mathsf{idx}\neq\mathsf{id}}$, random strings from secret keys of all authorities $\{\rho_{\mathsf{idx}}\}_{\mathsf{idx}\neq\mathsf{id}}$, user global identifier GID, attribute x, auxiliary information aux, message m, and outputs a secret key $\widetilde{\mathsf{SK}}_{\mathsf{id},\mathsf{GID},x}$.

Security. For any admissible adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that,

$$\Pr\left[b \leftarrow \mathcal{A}^{\mathcal{E}_b(\cdot,\cdot)}(\mathsf{CT}) \quad : \quad \begin{array}{l} (n, s_{\mathsf{abe}}, L, \ell_1, \ldots, \ell_n) \leftarrow \mathcal{A}(1^\lambda), b \xleftarrow{\$} \{0,1\}, \\ \mathsf{CRS} \leftarrow \mathsf{GSetup}(1^\lambda, 1^n, 1^{s_{\mathsf{abe}}}), \forall\ \mathsf{id} \in [n], \\ (\mathsf{MPK}_{\mathsf{id}}, \mathsf{MSK}_{\mathsf{id}}) \leftarrow \mathsf{ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}}), \\ (P, m) \leftarrow \mathcal{A}^{\mathcal{E}_b(\cdot,\cdot)}(\mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}}), \\ \text{If } b = 0, \mathsf{CT} \leftarrow \mathsf{Enc}(\{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}}, P, m), \\ \text{If } b = 1, (\widetilde{\mathsf{CT}}, \mathsf{aux}) \leftarrow \mathsf{Fake}(\{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}}, P) \end{array} \right]$$
$$\leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where $\mathcal{E}_0$ is a stateless oracle responds to secret key queries for $(\mathsf{KGen}, (\mathsf{id}, \mathsf{GID}, x))$ from $\mathcal{A}$ as $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{KGen}(\mathsf{id}, \mathsf{MSK}_{\mathsf{id}}, \{\mathsf{MPK}_{\mathsf{idx}}\}_{\mathsf{idx}\in[n]}, \mathsf{GID}, x)$. $\mathcal{E}_1$ is a stateful algorithm that when a satisfying query is made, for the last id with GID, x responds with $\widetilde{\mathsf{SK}}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{Reveal}(\mathsf{id}, \mathsf{MPK}_{\mathsf{id}}, \mathsf{MSK}_{\mathsf{id}}, \{(\mathsf{MPK}_{\mathsf{idx}}, \rho_{\mathsf{idx},\mathsf{GID}})\}_{\mathsf{idx}\neq\mathsf{id}}, \mathsf{GID}, x, \mathsf{aux}, m)$ A stateful PPT machine $\mathcal{A}$ is said to be admissible if it makes polynomially many queries to $\mathcal{E}_b$ and makes one KGen query per GID to an authority such that there is at most one $X = (\{x_{\mathsf{GID},\mathsf{id}}\}_{\mathsf{id}})$, with $P(X) = 1$.

Construction 11.5 (ncMA-ABE). We construct a non-committing MA-ABE scheme for predicate class $\mathcal{P} = \{\mathcal{P}_\lambda\}_{\lambda\in\mathbb{N}}$ and $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda\in\mathbb{N}}$ (Definition 11.4) using an MA-ABE scheme for predicate class $\mathcal{P} = \{\mathcal{P}_\lambda\}_{\lambda\in\mathbb{N}}$, $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda\in\mathbb{N}}$ with no corruptions (Definition 11.1) as follows.

$\underline{\mathsf{GSetup}(1^\lambda, 1^n, 1^{s_{\mathsf{abe}}}, 1^L)}$. For each $i \in [n], j \in [L], b \in \{0,1\}$, sample $\mathsf{crs}^{(i,j,b)} \leftarrow \mathsf{MA\text{-}ABE.GSetup}(1^\lambda, 1^n, 1^{s_{\mathsf{abe}}})$. Output $\mathsf{CRS} := (n, s_{\mathsf{abe}}, L, (\mathsf{crs}^{(i,j,b)})_{i,j,b})$.

$\underline{\mathsf{ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}})}$. For each $i \in [n], j \in [L], b \in \{0,1\}$, sample $(\mathsf{mpk}_{\mathsf{id}}^{(i,j,b)}, \mathsf{msk}_{\mathsf{id}}^{(i,j,b)}) \leftarrow \mathsf{MA\text{-}ABE.ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}})$. Output $\mathsf{MPK}_{\mathsf{id}} := (\mathsf{mpk}_{\mathsf{id}}^{(i,j,b)})_{i,j,b}$ and $\mathsf{MSK}_{\mathsf{id}} := (\mathsf{msk}_{\mathsf{id}}^{(i,j,b)})_{i,j,b}$.

$\underline{\mathsf{KGen}(\mathsf{id}, \mathsf{MSK}_{\mathsf{id}}, \{\mathsf{MPK}_{\mathsf{idx}}\}_{\mathsf{idx}\in[n]}, \mathsf{GID}, x)}$. Parse $\mathsf{MSK}_{\mathsf{id}}$ as $(\mathsf{msk}_{\mathsf{id}}^{(i,j,b)})_{i,j,b}$, $\mathsf{MPK}_{\mathsf{idx}}$ as $(\mathsf{mpk}_{\mathsf{idx}}^{(i,j,b)})_{i,j,b}$ for $\mathsf{idx} \in [n], i \in [n], j \in [L], b \in \{0,1\}$. Sample $\rho \xleftarrow{\$} \{0,1\}^L$ and for $i \in [n]$,

- If $i \neq \mathsf{id}$, for each j, b, $\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j,b)} \leftarrow \mathsf{MA\text{-}ABE.KGen}(\mathsf{id}, \mathsf{msk}_{\mathsf{id}}^{(i,j,b)}, \{\mathsf{mpk}_{\mathsf{idx}}^{(i,j,b)}\}_{\mathsf{idx}}, \mathsf{GID}, x)$.
- If $i = \mathsf{id}$, for each j, $\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j,\rho[j])} \leftarrow \mathsf{MA\text{-}ABE.KGen}(\mathsf{id}, \mathsf{msk}_{\mathsf{id}}^{(i,j,\rho[j])}, \{\mathsf{mpk}_{\mathsf{idx}}^{(i,j,\rho[j])}\}_{\mathsf{idx}}, \mathsf{GID}, x)$.

Output $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x,y} = \left(\{\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j,b)}\}_{i\neq\mathsf{id},j,b}, \{\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j,\rho[j])}\}_{i=\mathsf{id},j}, x, \rho\right)$

$\underline{\mathsf{Enc}(\{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}, P, m).}$ Parse $\mathsf{MPK}_{\mathsf{id}}$ as $(\mathsf{mpk}_{\mathsf{id}}^{(i,j,b)})_{i,j,b}$ for $\mathsf{id} \in [n], i \in [n], j \in [L], b \in \{0,1\}$. Sample $R_1, \ldots, R_n \xleftarrow{\$} \{0,1\}^L$ and set $\widetilde{R} := R_1 \oplus \ldots \oplus R_n \oplus m$. Compute for each i, j, b, $\mathsf{ct}^{(i,j,b)} \leftarrow \mathsf{MA\text{-}ABE.Enc}(\{\mathsf{mpk}_{\mathsf{id}}^{(i,j,b)}\}, P, R_i[j])$. Output $\widetilde{\mathsf{CT}} := (\widetilde{R}, (\mathsf{ct}^{(i,j,b)})_{i,j,b})$.

$\underline{\mathsf{Dec}(\{\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}},y_{\mathsf{id}}}\}_{\mathsf{id}\in[n]}, \mathsf{CT}).}$ Parse CT as $(\widetilde{R}, (\mathsf{ct}^{(i,j,b)})_{i,j,b})$ and $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}},y_{\mathsf{id}}}$ as $\left(\{\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j,b)}\}_{i\neq\mathsf{id},j,b}, \{\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j,\rho[j])}\}_{i=\mathsf{id},j}, x_{\mathsf{id}}, \rho_{\mathsf{id}}\right)$ for $\mathsf{id} \in [n], i \in [n], j \in [L], b \in \{0,1\}$. For each i, j, $R_i[j] = \mathsf{MA\text{-}ABE.Dec}(\{\mathsf{sk}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}^{(i,j,\rho_i[j])}\}_{\mathsf{id}\in[n]}, \mathsf{ct}^{(i,j,\rho_i[j])})$. Output $m' = R_1 \oplus \ldots R_n \oplus \widetilde{R}$.

$\underline{\mathsf{Fake}(\{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}, P).}$ Parse $\mathsf{MPK}_{\mathsf{id}}$ as $(\mathsf{mpk}_{\mathsf{id}}^{(i,j,b)})_{i,j,b}$ for $\mathsf{id} \in [n], i \in [n], j \in [L], b \in \{0,1\}$. Sample $R_1, \ldots, R_n, \widetilde{R} \xleftarrow{\$} \{0,1\}^L$. Compute for each i, j, $\mathsf{ct}^{(i,j,R_i[j])} \leftarrow \mathsf{MA\text{-}ABE.Enc}(\{\mathsf{mpk}_{\mathsf{id}}^{(i,j,R_i[j])}\}, P, 0)$ and $\mathsf{ct}^{(i,j,1-R_i[j])} \leftarrow \mathsf{MA\text{-}ABE.Enc}(\{\mathsf{mpk}_{\mathsf{id}}^{(i,j,1-R_i[j])}\}, P, 1)$. Output $\mathsf{CT} := (\widetilde{R}, (\mathsf{ct}^{(i,j,b)})_{i,j,b})$ and $\mathsf{aux} = (R_1, \ldots, R_n, \widetilde{R})$.

$\underline{\mathsf{Reveal}(\mathsf{id}, \mathsf{MPK}_{\mathsf{id}}, \mathsf{MSK}_{\mathsf{id}}, \{(\mathsf{MPK}_{\mathsf{idx}}, \rho_{\mathsf{idx}})\}_{\mathsf{idx}\neq\mathsf{id}}, \mathsf{GID}, x, \mathsf{aux}, m).}$ Parse $\mathsf{MSK}_{\mathsf{id}}$ as $(\mathsf{msk}_{\mathsf{id}}^{(i,j,b)})_{i,j,b}$, $\mathsf{MPK}_{\mathsf{id}}$ as $(\mathsf{mpk}_{\mathsf{id}}^{(i,j,b)})_{i,j,b}$, $\mathsf{MPK}_{\mathsf{idx}}$ as $(\mathsf{mpk}_{\mathsf{idx}}^{(i,j,b)})_{i,j,b}$ for $\mathsf{idx} \neq \mathsf{id}, i \in [n], j \in [L], b \in \{0,1\}$ and aux as $(R_1, \ldots, R_n, \widetilde{R})$. Set $\rho := \left(\bigoplus_{\mathsf{idx}\neq\mathsf{id}} \rho_{\mathsf{idx}} \oplus R_{\mathsf{idx}}\right) \oplus R_{\mathsf{id}} \oplus \widetilde{R} \oplus m$. For $i \in [n]$,

- If $i \neq \mathsf{id}$, for each j, b, $\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j,b)} \leftarrow \mathsf{MA\text{-}ABE.KGen}(\mathsf{id}, \mathsf{msk}_{\mathsf{id}}^{(i,j,b)}, \{\mathsf{mpk}_{\mathsf{idx}}^{(i,j,b)}\}_{\mathsf{idx}}, \mathsf{GID}, x)$.
- If $i = \mathsf{id}$, for each j, $\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j,\rho[j])} \leftarrow \mathsf{MA\text{-}ABE.KGen}(\mathsf{id}, \mathsf{msk}_{\mathsf{id}}^{(i,j,\rho[j])}, \{\mathsf{mpk}_{\mathsf{idx}}^{(i,j,\rho[j])}\}_{\mathsf{idx}}, \mathsf{GID}, x)$.

Output $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x,y} = \left(\{\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j,b)}\}_{i\neq\mathsf{id},j,b}, \{\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j,\rho[j])}\}_{i=\mathsf{id},j}, x, \rho\right)$

Correctness. The correctness of the scheme follows from the correctness of MA-ABE. In particular, similar to Construction 11.2, we have a complete set of secret keys for the instances $(i, j, \rho_i[j])$. Using these and by correctness of MA-ABE, we can recover $\{R_i[j]\}_{i,j}$ and thus $m' = m$.

Theorem 11.6. If MA-ABE is an MA-ABE scheme for predicate class $\mathcal{P} = \{\mathcal{P}_\lambda\}_{\lambda\in\mathbb{N}}$ and $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda\in\mathbb{N}}$ with no corruptions (Definition 11.1), then Construction 11.5 is a non-committing MA-ABE scheme for predicate class $\mathcal{P} = \{\mathcal{P}_\lambda\}_{\lambda\in\mathbb{N}}$ and $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda\in\mathbb{N}}$ (Definition 11.4).

Proof Sketch. We provide a sketch of the hybrid argument used to prove security of Construction 11.5 as follows. Note that for this proof, if $\mathcal{A}$ makes a satisfying query in the pre-challenge query phase, all hybrids proceed similar to hybrid 0. Hence, assume that $\mathcal{A}$ makes a satisfying query in the post-challenge query phase.

Hybrid 0. This is the real experiment with $\mathcal{E}_0$ from Definition 11.4. We use the MA-ABE honestly to encrypt $R_i[j]$ in both $(i, j, 0)$ and $(i, j, 1)$ instantiations of MA-ABE.

Hybrid 1. In this hybrid, as $\mathcal{A}$ runs in polynomial time, it queries with at most $q = \mathsf{poly}(\lambda)$ many GIDs. Chal guesses that N^*-th GID, where $N^* \xleftarrow{\$} [q]$, is the satisfying query that $\mathcal{A}$ issues. In addition, Chal will sample $\rho_1^*, \ldots, \rho_n^* \xleftarrow{\$} \{0,1\}^L$ uniformly during setup phase and use these strings for the N^*-th GID's secret key queries. Chal will abort if it guesses incorrectly. Chal does not abort is at most $1/q$ probability and output distribution is exactly like hybrid 0.

Hybrid $2, \iota, \gamma$. for $\iota \in [n], \gamma \in [L]$. In this hybrid, we will encrypt $1 - R_i[j]$ in the $(i, j, 1 - \rho_i^*[j])$-th instantiation of MA-ABE for $i < \iota$ and $j < \gamma$. As only $n - 1$ secret keys for $(i, j, 1 - \rho_i^*[j])$-th instantiation and as such the adversary cannot decrypt even though $P(x_1, \ldots, x_n) = 1$. Thus, we can readily rely on MA-ABE security to argue indistinguishability of these hybrids with a $1/q$ loss in advantage due to the guess made like in hybrid 1.

Hybrid 3. In this hybrid, $(i, j, \rho_i^*[j])$-th instantiation encrypts $R_i[j]$ and $(i, j, 1 - \rho_i^*[j])$-th instantiation encrypts $1 - R_i[j]$ for every i, j. Indistinguishability between hybrids $2, n, L$ and 3 can be argued from security of MA-ABE.

Hybrid 4. In this hybrid, we will use Fake to encrypt and when a satisfying query is made, use Reveal to equivocate m. Output distribution of hybrids 3 and 4 are identical.

We provide full proof in Appendix I. □

11.4 Adaptive (poly, 1)-MA-ABFE

Construction 11.7 (1MA-ABFE)**.** We construct a $(\mathsf{poly}, 1)$-bounded MA-ABFE scheme for predicate class $\mathcal{P} = \{\mathcal{P}_\lambda\}_{\lambda \in \mathbb{N}}$, P/Poly circuits, $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda \in \mathbb{N}}$ (Definition 10.3) using a non-adaptive $(\mathsf{poly}, 1)$-bounded MA-ABFE scheme for predicate class $\mathcal{P} = \{\mathcal{P}_\lambda\}_{\lambda \in \mathbb{N}}$, P/Poly circuits, $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda \in \mathbb{N}}$ (Definition 10.4), a non-committing MA-ABE scheme for predicate class $\mathcal{P} = \{\mathcal{P}_\lambda\}_{\lambda \in \mathbb{N}}$, $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda \in \mathbb{N}}$ (Definition 11.4) as follows.

$\underline{\mathsf{GSetup}(1^\lambda, 1^n, 1^L, 1^{s_{\mathsf{abe}}}, 1^{s_{\mathsf{fe}}})}$**.** Sample $\mathsf{abfe.crs} \leftarrow \mathsf{na1MA\text{-}ABFE.GSetup}(1^\lambda, 1^n, 1^L, 1^{s_{\mathsf{abe}}}, 1^{s_{\mathsf{fe}}})$. Let $G = \mathsf{poly}(\lambda, n, s_{\mathsf{abe}}, s_{\mathsf{fe}}, L)$ be the length of na1MA-ABFE's ciphertext for these parameters. $\mathsf{abe.crs} \leftarrow \mathsf{ncMA\text{-}ABE.GSetup}(1^\lambda, 1^n, 1^{s_{\mathsf{abe}}}, 1^G)$. Output $\mathsf{CRS} := (\mathsf{abfe.crs}, \mathsf{abe.crs})$.

$\underline{\mathsf{ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}})}$**.** Sample $(\mathsf{abfe.mpk}, \mathsf{abfe.msk}) \leftarrow \mathsf{na1MA\text{-}ABFE.ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}})$, $(\mathsf{abe.mpk}, \mathsf{abe.msk}) \leftarrow \mathsf{ncMA\text{-}ABE.ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}})$. Set $\mathsf{MPK} := (\mathsf{abfe.mpk}, \mathsf{abe.mpk})$, $\mathsf{MSK} := (\mathsf{abfe.msk}, \mathsf{abe.msk})$ and output $(\mathsf{MPK}, \mathsf{MSK})$.

$\underline{\mathsf{KGen}(\mathsf{id}, \mathsf{MSK}_{\mathsf{id}}, \{\mathsf{MPK}_{\mathsf{idx}}\}_{\mathsf{idx} \in [n]}, \mathsf{GID}, x, y)}$**.** Parse $\mathsf{MSK}_{\mathsf{id}}$ as $(\mathsf{abfe.msk}_{\mathsf{id}}, \mathsf{abe.msk}_{\mathsf{id}})$, $\mathsf{MPK}_{\mathsf{idx}}$ as $(\mathsf{abfe.mpk}_{\mathsf{idx}}, \mathsf{abe.mpk}_{\mathsf{idx}})$ for each $\mathsf{idx} \in [n]$. Run $\mathsf{abfe.sk}_{\mathsf{id},\mathsf{GID},x,y} \leftarrow \mathsf{na1MA\text{-}ABFE.KGen}(\mathsf{id}, \mathsf{abfe.msk}_{\mathsf{id}}, \{\mathsf{abfe.mpk}_{\mathsf{idx}}\}_{\mathsf{idx}}, \mathsf{GID}, x, y)$ and $\mathsf{abe.sk}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{ncMA\text{-}ABE.KGen}(\mathsf{id}, \mathsf{abe.msk}_{\mathsf{id}}, \{\mathsf{abe.mpk}_{\mathsf{idx}}\}_{\mathsf{idx}}, \mathsf{GID}, x)$. Output $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x,y} := (\mathsf{abfe.sk}_{\mathsf{id},\mathsf{GID},x,y}, \mathsf{abe.sk}_{\mathsf{id},\mathsf{GID},x})$.

$\underline{\mathsf{Enc}(\{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id} \in [n]}, P, C)}$**.** Parse $\mathsf{MPK}_{\mathsf{id}}$ as $(\mathsf{abfe.mpk}_{\mathsf{id}}, \mathsf{abe.mpk}_{\mathsf{id}})$ for each $\mathsf{id} \in [n]$. Sample $\mathsf{abfe.ct} \leftarrow \mathsf{na1MA\text{-}ABFE.Enc}(\{\mathsf{abfe.mpk}_{\mathsf{id}}\}_{\mathsf{id} \in [n]}, P, C)$. Now, run $\mathsf{abe.ct} \leftarrow \mathsf{ncMA\text{-}ABE.Enc}(\{\mathsf{abe.mpk}_{\mathsf{id}}\}_{\mathsf{id}}, P, \mathsf{abfe.ct})$. Output $\mathsf{CT} := \mathsf{abe.ct}$.

$\underline{\mathsf{Dec}(\{\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}},y_{\mathsf{id}}}\}_{\mathsf{id} \in [n]}, \mathsf{CT})}$**.** Parse $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}},y_{\mathsf{id}}}$ as $(\mathsf{abfe.sk}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}},y_{\mathsf{id}}}, \mathsf{abe.sk}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}})$ for each $\mathsf{id} \in [n]$ and CT as $\mathsf{abe.ct}$. Run $\mathsf{abe.ct}' = \mathsf{ncMA\text{-}ABE.Dec}(\{\mathsf{abe.sk}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}\}_{\mathsf{id}}, \mathsf{abe.ct})$. If $\mathsf{abfe.ct}' = \perp$, abort and output $\perp$. Otherwise, output $z := \mathsf{na1MA\text{-}ABFE.Enc}(\{\mathsf{abfe.sk}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}},y_{\mathsf{id}}}\}_{\mathsf{id}}, \mathsf{abfe.ct}')$.

Correctness. The correctness of the scheme follows from the correctness of na1MA-ABFE and ncMA-ABE. By correctness of ncMA-ABE, $\mathsf{abfe.ct}'$ is a valid ciphertext of na1MA-ABFE and $z = C(y_1, \ldots, y_n)$ by correctness of na1MA-ABFE.

Theorem 11.8. If $\mathsf{ncMA\text{-}ABE}$ is a non-committing MA-ABE scheme for predicate class $\mathcal{P} = \{\mathcal{P}_\lambda\}_{\lambda\in\mathbb{N}}$, $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda\in\mathbb{N}}$ (Definition 11.4), $\mathsf{na1MA\text{-}ABFE}$ is a non-adaptive $(\mathsf{poly}, 1)$-bounded MA-ABFE scheme for predicate class $\mathcal{P} = \{\mathcal{P}_\lambda\}_{\lambda\in\mathbb{N}}$, $\mathsf{P/Poly}$ circuits, $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda\in\mathbb{N}}$ (Definition 10.4), then Construction 11.7 is a $(\mathsf{poly}, 1)$-bounded MA-ABFE scheme for predicate class $\mathcal{P} = \{\mathcal{P}_\lambda\}_{\lambda\in\mathbb{N}}$, $\mathsf{P/Poly}$ circuits, $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda\in\mathbb{N}}$ (Definition 10.3).

Proof Sketch. We provide a sketch of the hybrid argument used to prove security of Construction 11.7 as follows.

Hybrid 0. This is the real experiment with Chal from Definition 10.3. We use $\mathcal{E}_0$ for $\mathsf{ncMA\text{-}ABE}$ and honest and stateless algorithms for $\mathsf{na1MA\text{-}ABFE}$.

Hybrid 1. In this hybrid, we will shift to using $\mathcal{E}_1$ for $\mathsf{ncMA\text{-}ABE}$. That is we will fake the ciphertext and only equivocate it to $\mathsf{abfe.ct}$ if adv makes a satisfying query in the post-challenge query phase. The indistinguishability of hybrids can be argued using security of $\mathsf{ncMA\text{-}ABE}$ instantiation.

Hybrid 2. In this hybrid, we will simulate the $\mathsf{na1MA\text{-}ABFE}$ instantiation either while equivocating or in the challenge query phase (if $\mathcal{A}$ makes a satisfying query in pre-challenge query phase). Hybrids 1 and 2 are indistinguishable due to security of $\mathsf{na1MA\text{-}ABFE}$. This is the description of Sim from Definition 10.3.

We provide full proof in Appendix J. □

12 Attribute-Based Distributed Client-Server Framework

In this section, we define and construct an attribute-based distributed client-server framework (adCSF). This is strengthening of dCSF defined in [GY24] which was used to construct a bounded query MAFE scheme starting from an MAFE scheme that is secure against one GID collusion. We define this object to similarly to construct a (poly, Q)-bounded MA-ABFE scheme starting from a $(\mathsf{poly}, 1)$-bounded MA-ABFE scheme that was constructed in Section 11.4.

Recall that dCSF (defined in Definition B.1) is a specialized MPC protocol between n servers (with inputs $\{x_{\mathsf{id}}\}_{\mathsf{id}}$) and one client (with circuit C) both of whom delegate their computation to N users to learn $C(x_1, \ldots, x_n)$ for each input. In adCSF, server encodings are generated for input parsed as (x, y) and client encodings for input parsed as (P, C). The main difference between dCSF and adCSF is that the adversary is now allowed to query an unbounded number of server encodings but only Q of them for satisfying queries. And the security is defined using a simulator who only uses the values of $\{C(Y_{\mathsf{GID}})\}_{\mathsf{GID}}$ for satisfying $Y_{\mathsf{GID}} = (y_{\mathsf{GID},1}, \ldots, y_{\mathsf{GID},n})$.

We construct an adCSF scheme generically from any dCSF scheme. The client encodings are set as $(P, \widehat{C}^u)$ where $\widehat{C}^u$ is client encoding from dCSF. Note that in an MA-ABE scheme, predicate P is not hidden. Hence, it is fine to reveal P as whole to adversary. The main intuition is that server encodings are deterministic. Hence, while arguing security, the reduction algorithm can generate server encodings on its own and respond to the adversary. In order to generate client encodings, when the predicate P and circuit C are given, reduction algorithm can determine which of the queries are satisfying and thus send these $\{y_{\mathsf{GID},\mathsf{id}}\}_{\mathsf{id},\mathsf{GID}}$ queries to a challenger for dCSF. We will ignore the server encodings and only use the client encodings to respond to adversary. Post-challenge query phase proceeds similarly and reduction will query the challenger for output encodings only for satisfying queries. Thus, we construct an adCSF scheme from a dCSF scheme in Section 12.2.

12.1 Definition

We provide the definition of adCSF as follows. For a complete definition, we ask the reader to compare and contrast the definition of dCSF provided in Definition B.1. We provide a sanitized definition of adCSF with the differences from dCSF highlighted below.

Definition 12.1 (adCSF)**.** An adCSF scheme $(\mathsf{ServEnc}, \mathsf{CliEnc}, \mathsf{UserComp}, \mathsf{Decode})$ is said to be an attribute-based dCSF scheme (adCSF) for predicate class $\mathcal{P} = \{\mathcal{P}_\lambda\}_{\lambda \in \mathbb{N}}$, circuit class $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$, and $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_\lambda$ is a dCSF scheme for function class $\mathcal{F} = \{(\mathcal{P}_\lambda, \mathcal{C}_\lambda)\}_{\lambda \in \mathbb{N}}$ and $\mathcal{GID}$. The major changes are highlighted below.

Syntax. The server encodings are generated for input parsed as (x, y) and client encodings are generated for input parsed as predicate and circuit pair (P, C).

Correctness. This is defined for two types of encodings, satisfying encodings where Decode outputs $C(y_1, \ldots, y_n)$ if $P(x_1, \ldots, x_n) = 1$ and unsatisfying encodings where Decode outputs $\perp$.

Security. For any admissible adversary $\mathcal{A}$, there exists a stateful simulator Sim such that $\forall\ \lambda \in \mathbb{N}$,

$$\left\{\mathsf{adCSF\text{-}Expt}^{\mathsf{Chal}}_{\mathcal{A},0}(1^\lambda)\right\} \approx_c \left\{\mathsf{adCSF\text{-}Expt}^{\mathsf{Sim}}_{\mathcal{A},1}(1^\lambda)\right\}$$

where definitions of $\mathsf{adCSF\text{-}Expt}^{\mathsf{Chal}}_{\mathcal{A},0}(1^\lambda)$, $\mathsf{adCSF\text{-}Expt}^{\mathsf{Sim}}_{\mathcal{A},1}(1^\lambda)$, and admissible adversary are similar to Definition B.1. The major changes are as follows—

- $\mathcal{A}$ makes *polynomially many* server encoding queries such that each authority is queried at most once per GID with the restriction that at most Q server encoding queries are made across n authorities for satisfying queries.
- Simulator Sim takes P as input.
- $\mathcal{V}$ used by Sim only contains $C(Y)$ for $Y = (y_{1,\mathsf{GID}}, \ldots, y_{n,\mathsf{GID}})$ if $P(x_{1,\mathsf{GID}}, \ldots, x_{n,\mathsf{GID}}) = 1$. Similarly in the post-challenge query phase Sim queries C with Y for satisfying queries.

12.2 Attribute-Based dCSF from Distributed Client-Server Framework

Construction 12.2 (adCSF)**.** We provide the construction of an adCSF scheme for predicate class $\mathcal{P} = \{\mathcal{P}_\lambda\}_{\lambda \in \mathbb{N}}$ $\mathsf{P/Poly}$ circuits, $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda \in \mathbb{N}}$ from a dCSF scheme (Definition B.1) for $\mathsf{P/Poly}$ circuits, $\mathcal{GID}$ as follows.

$\underline{\mathsf{ServEnc}(1^\lambda, 1^Q, 1^n, 1^{s_{\mathsf{abe}}}, 1^{s_{\mathsf{fe}}}, \mathsf{GID}, \mathsf{id}, x, y, \boldsymbol{\Delta})}$**.** Run $\{\widehat{y}^u_{\mathsf{GID},\mathsf{id}}\}_{u \in [N]} = \mathsf{dCSF.ServEnc}(1^\lambda, 1^Q, 1^n, 1^{s_{\mathsf{fe}}}, \mathsf{GID}, y, \Delta)$. Set $\widehat{a}^u_{\mathsf{GID},\mathsf{id}} = (\mathsf{GID}, x, \widehat{y}^u_{\mathsf{GID},\mathsf{id}})$ and output $\{\widehat{a}^u_{\mathsf{GID},\mathsf{id}}\}_u$.

$\underline{\mathsf{CliEnc}(1^\lambda, 1^Q, 1^n, 1^{s_{\mathsf{abe}}}, 1^{s_{\mathsf{fe}}}, P, C)}$**.** Sample $\{\widehat{C}^u\}_{u \in [N]} \leftarrow \mathsf{dCSF.CliEnc}(1^\lambda, 1^Q, 1^n, 1^{s_{\mathsf{fe}}}, C)$. Set $\widehat{F}^u :=$ $(P, \widehat{C}^u)$ and output $\{\widehat{F}^u\}_{u \in [N]}$.

$\underline{\mathsf{UserComp}(\{\widehat{a}^u_{\mathsf{GID},\mathsf{id}}\}_{\mathsf{id}\in[n]}, \widehat{F}^u)}$. Parse $\widehat{F}^u$ as $(P, \widehat{C}^u)$ and $\widehat{a}^u_{\mathsf{GID},\mathsf{id}}$ as $(\mathsf{GID}_{\mathsf{id}}, x_{\mathsf{id}}, \widehat{y}^u_{\mathsf{GID}_{\mathsf{id}},\mathsf{id}})$. If all $\mathsf{GID}_{\mathsf{id}}$ are not same or $P(x_1, \ldots, x_n) = 0$, abort and output $\perp$.

Otherwise, compute $\widehat{z}^u_{\mathsf{GID}} \leftarrow \mathsf{dCSF.UserComp}(\{\widehat{y}^u_{\mathsf{GID},\mathsf{id}}\}, \widehat{C}^u)$. Output $\widehat{z}^u_{\mathsf{GID}}$.

$\underline{\mathsf{Decode}(\{\widehat{z}^u_{\mathsf{GID}}\}_{u\in\mathbf{S}}, \mathbf{S})}$. If all encodings are not present, output $\perp$. Otherwise, compute and output $z = \mathsf{dCSF.Decode}(\{\widehat{z}^u_{\mathsf{GID}}\}_{u\in\mathbf{S}}, \mathbf{S})$.

Correctness. The correctness of the scheme follows from the correctness of dCSF.

Theorem 12.3. If dCSF is a distributed client-server framework for $\mathsf{P/Poly}$ circuits, $\mathcal{GID}$ (Definition B.1), then Construction 12.2 is an adCSF scheme for $\mathsf{P/Poly}$ circuits, $\mathcal{GID}$.

Proof. Assume that there exists an adversary $\mathcal{A}$ that can distinguish between honest Chal and $\mathsf{adCSF.Sim}$ with non-negligible probability $\epsilon(\lambda)$. We will construct a reduction adversary $\mathcal{B}$ that can break the security of dCSF with non-negligible probability. The description of $\mathcal{B}^{\mathcal{O}}$ is as follows.

Setup. $\mathcal{A}$ sends $(Q, n, s_{\mathsf{abe}}, s_{\mathsf{fe}}, \widetilde{\mathbf{S}})$ to $\mathcal{B}$. $\mathcal{B}$ passes $(Q, n, s_{\mathsf{fe}}, \widetilde{\mathbf{S}})$ to $\mathcal{O}$.

Pre-challenge Queries. $\mathcal{A}$ sends polynomially many queries of the form $(\mathsf{id}, \mathsf{GID}, x, y, \mathbf{\Delta}), \mathbf{S}$. Send $(x, \widehat{y}^u_{\mathsf{GID},\mathsf{id}})$ for $u \in \mathbf{S}$ and $\widehat{y}^u_{\mathsf{GID},\mathsf{id}} = (\mathsf{GID}, y, \mathbf{\Delta})$.

Challenge Query. $\mathcal{A}$ sends $P, C, \{j^*_{\mathsf{GID}}\}_{\mathsf{GID}}$. At this point $\mathcal{B}$ can figure out all the accepting attributes and GIDs. Perform pre-challenge queries for $\mathcal{O}$ using these $(\mathsf{id}, \mathsf{GID}, y, \mathbf{\Delta}), \mathbf{S}$. Ignore the responses from $\mathcal{O}$. Query $\mathcal{O}$ with $C, \{j^*_q\}_q$ to receive $\{\widehat{C}^u\}_{u\in[N]\setminus\widetilde{\mathbf{S}}}$ and $\{\widehat{y}^u_{\mathsf{GID}}\}_{\mathsf{GID},u}$. Set $\widehat{F}^u = (P, \widehat{C}^u)$ for $u \in [N]\setminus\widetilde{\mathbf{S}}$ and $\widehat{z}^u_{\mathsf{GID}} = \widehat{y}^u_{\mathsf{GID}}$ for appropriate GID, u and send $\{\widehat{F}^u\}_{u\in[N]\setminus\widetilde{\mathbf{S}}}, \{\widehat{z}^u_{\mathsf{GID}}\}_{\mathsf{GID},u}$ to $\mathcal{A}$.

Post-challenge Queries. $\mathcal{A}$ sends polynomially many queries of the form $(\mathsf{id}, \mathsf{GID}, x, y, \mathbf{\Delta}), \mathbf{S}$ and j^*_{GID} if $P(x_{\mathsf{GID},1}, \ldots, x_{\mathsf{GID},n}) = 1$ and all authorities are queried. If j^*_{GID} is present query $\mathcal{O}$ with $(\mathsf{id}, \mathsf{GID}, y_{\mathsf{id}}, \mathbf{\Delta}), \mathbf{S}$ for $\mathsf{id} \in [n]$, ignore the responses and receive $\widehat{y}^u_{\mathsf{GID}}$ for $u \in \mathbf{S}$. Otherwise, set $\widehat{a}^u_{\mathsf{GID},\mathsf{id}} = (x, \widehat{y}^u_{\mathsf{GID},\mathsf{id}})$ where $\widehat{y}^u_{\mathsf{GID},\mathsf{id}} = (\mathsf{GID}, y_{\mathsf{id}}, \mathbf{\Delta})$ and send $\{\widehat{a}^u_{\mathsf{GID},\mathsf{id}}\}_{u\in\mathbf{S}}$ and $\{\widehat{z}^u_{\mathsf{GID}}\}_{u\in\mathbf{S}}$ (if applicable) to $\mathcal{A}$.

Guess Phase. Perform required checks on $\{\mathbf{S}_{\mathsf{GID}}, \mathbf{\Delta}_{\mathsf{GID}}, j^*_{\mathsf{GID}}\}_{\mathsf{GID}}$ and output whatever $\mathcal{A}$ outputs.

As we can see, the running time of $\mathcal{B}$ is a polynomial in the running time of $\mathcal{A}$ and λ. If $\mathcal{O}$ us an honest challenger for dCSF, $\mathcal{B}$ behaves like $\mathsf{adCSF\text{-}Expt}^{\mathsf{Chal}}_{\mathcal{A},0}(1^\lambda)$ and if $\mathcal{O}$ is a simulator for dCSF, $\mathcal{B}$ behaves like $\mathsf{adCSF\text{-}Expt}^{\mathsf{Sim}}_{\mathcal{A},1}(1^\lambda)$. Hence, $\mathcal{B}$ can break the security of dCSF with advantage $\epsilon(\lambda)$. Thus, adCSF is secure. □

13 (poly, Q)-MA-ABFE from (poly, 1)-MA-ABFE and adCSF in ROM

In this section, we provide the construction of a (poly, Q)-bounded MA-ABFE scheme from $(\mathsf{poly}, 1)$-bounded MA-ABFE scheme and an adCSF scheme. The construction is similar to the construction of MAFE from sub-exponentially secure PKE in ROM from [GY24]. Here, we use a (non-programmable) random oracle $\mathcal{H}$ to sample $(\mathbf{S}, \mathbf{\Delta})$ for each GID. In the security reduction, we will guess the set of non-corrupted users and unique indices in each $\{\mathbf{\Delta}_{\mathsf{GID}}\}_{\mathsf{GID}}$ to rely on the security of adCSF. Note that we need to rely on the augmented statistical lemmas Lemma B.2 and Lemma B.3. The security proof proceeds similar to [GY24] and we omit full proof for brevity.

Construction 13.1. We construct a (poly, Q)-bounded MA-ABFE scheme for predicate class $\mathcal{P} = \{\mathcal{P}_\lambda\}_{\lambda\in\mathbb{N}}$, $\mathsf{P/Poly}$ circuits, $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda\in\mathbb{N}}$ (Definition 10.1) using a (poly, Q)-bounded MA-ABFE scheme for predicate class $\mathcal{P}$, $\mathsf{P/Poly}$ circuits, $\mathcal{GID}$ (Definition 10.3), and an adCSF scheme for $\mathcal{P}$, $\mathsf{P/Poly}$ circuits, $\mathcal{GID}$ (Definition 12.1), in the random oracle model as follows. Here, N, D, t are from Lemma B.2, T, v are from Lemma B.3.

<u>$\mathsf{GSetup}(1^\lambda, 1^Q, 1^n, 1^L, 1^{s_{\mathsf{abe}}}, 1^{s_{\mathsf{fe}}})$</u>**.** For each $u \in [N]$, sample $\mathsf{abfe.crs}_u \leftarrow \mathsf{1MA\text{-}ABFE.GSetup}(1^{\lambda'}, 1^n, 1^L, 1^{s_{\mathsf{abe}}}, 1^{s_{\mathsf{fe}}})$. Here $\lambda' = O\left((Q^2\lambda^2)^{1/\alpha}\right)$ and α is a constant to get the adversary's advantage within $2^{-O(Q^2\lambda^2)}$. Sample a hash function $\mathcal{H} : \mathcal{GID} \to 2^{[N]}\big|_D \times 2^T\big|_v$. Output $\mathsf{CRS} := (\lambda, Q, n, L, s_{\mathsf{abe}}, s_{\mathsf{fe}}, \mathcal{H}, \{\mathsf{abfe.crs}_u\}_{u\in[N]})$.

<u>$\mathsf{ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}})$</u>**.** For each $u \in [N]$, sample master public and secret keys $(\mathsf{abfe.mpk}_u, \mathsf{abfe.msk}_u) \leftarrow \mathsf{1MA\text{-}ABFE.ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}})$. Output $\mathsf{MPK} := \{\mathsf{abfe.mpk}_u\}_u$, $\mathsf{MSK} := \{\mathsf{abfe.msk}_u\}_u$.

<u>$\mathsf{KGen}^{\mathcal{H}}(\mathsf{id}, \mathsf{MSK}_{\mathsf{id}}, \{\mathsf{MPK}_{\mathsf{idx}}\}_{\mathsf{idx}\in[n]}, \mathsf{GID}, x, y)$</u>**.** Parse $\mathsf{MSK}_{\mathsf{id}}$, $\mathsf{MPK}_{\mathsf{idx}}$ as $\{\mathsf{abfe.msk}_{\mathsf{id},u}\}_u$, $\{\mathsf{abfe.mpk}_{\mathsf{idx},u}\}_u$ for each $\mathsf{idx} \in [n]$ and $u \in [N]$. Deterministically sample $(\mathbf{S}, \mathbf{\Delta}) = \mathcal{H}(\mathsf{GID})$. Compute using adCSF, $\{\widehat{a}^u_{\mathsf{GID},\mathsf{id}}\}_u \leftarrow \mathsf{adCSF.ServEnc}(1^{\lambda'}, 1^Q, 1^n, 1^{s_{\mathsf{abe}}}, 1^{s_{\mathsf{fe}}}, \mathsf{GID}, \mathsf{id}, x, y, \mathbf{\Delta})$. For each $u \in \mathbf{S}$, compute $\mathsf{abfe.sk}_u \leftarrow \mathsf{1MA\text{-}ABFE.KGen}(\mathsf{id}, \mathsf{abfe.msk}_{\mathsf{id}}, \{\mathsf{abfe.mpk}_{\mathsf{idx}}\}_{\mathsf{idx}\in[n]}, \mathsf{GID}, x, \widehat{a}^u_{\mathsf{GID},\mathsf{id}})$. Output $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x,y} := (\mathbf{S}, \{\mathsf{abfe.sk}_u\}_{u\in\mathbf{S}})$.

<u>$\mathsf{Enc}(\{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}, P, C)$</u>**.** For each $\mathsf{id} \in [n]$, parse $\mathsf{MPK}_{\mathsf{id}}$ as $\{\mathsf{abfe.mpk}_{\mathsf{id},u}\}_u$ for $u \in [N]$. Compute $\{\widehat{F}^u\}_u \leftarrow \mathsf{adCSF.CliEnc}(1^{\lambda'}, 1^Q, 1^n, 1^{s_{\mathsf{abe}}}, 1^{s_{\mathsf{fe}}}, P, C)$. $\forall\ u \in [N]$, let $G^u = \mathsf{adCSF.UserComp}(\cdot, \ldots, \cdot, \widehat{F}^u)$. Sample $\mathsf{abfe.ct}_u \leftarrow \mathsf{1MA\text{-}ABFE.Enc}(\{\mathsf{abfe.mpk}_{\mathsf{id},u}\}_{\mathsf{id}\in[n]}, P, G^u)$. Output $\mathsf{CT} := \{\mathsf{abfe.ct}_u\}_{u\in[N]}$.

<u>$\mathsf{Dec}(\{\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}},y_{\mathsf{id}}}\}_{\mathsf{id}\in[n]}, \mathsf{CT})$</u>**.** For each $\mathsf{id} \in [n]$, parse $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}},y_{\mathsf{id}}}$ as $(\mathbf{S}_{\mathsf{id}}, \{\mathsf{abfe.sk}_{\mathsf{id},u}\}_{u\in\mathbf{S}_{\mathsf{id}}})$ and CT as $\{\mathsf{abfe.ct}_u\}_{u\in[N]}$. If all $\mathbf{S}_{\mathsf{id}}$ are not the same, then abort and output $\perp$. Otherwise, for each $u \in \mathbf{S}_1$, $\widehat{z}^u_{\mathsf{GID}} = \mathsf{1MA\text{-}ABFE.Dec}(\{\mathsf{abfe.sk}_{\mathsf{id},u}\}_{\mathsf{id}\in[n]}, \mathsf{abfe.ct}_u)$. Output $z := \mathsf{adCSF.Decode}(\{\widehat{z}^u_{\mathsf{GID}}\}_{u\in\mathbf{S}_1}, \mathbf{S}_1)$.

Correctness. The correctness follows from correctness of adCSF and 1MA-ABFE.

Theorem 13.2. If 1MA-ABFE is a sub-exponentially secure $(\mathsf{poly}, 1)$-bounded MA-ABFE scheme (Definition 10.3) for predicate class $\mathcal{P}$, $\mathsf{P/Poly}$ circuits, $\mathcal{GID}$ and adCSF is a sub-exponentially secure attribute-based dCSF scheme for predicate class $\mathcal{P}$, $\mathsf{P/Poly}$ circuits, $\mathcal{GID}$ then Construction 13.1 is a sub-exponentially secure (poly, Q)-bounded MA-ABFE scheme (Definition 10.1) for predicate class $\mathcal{P}$, $\mathsf{P/Poly}$ circuits, $\mathcal{GID}$ in random oracle model.

Proof Sketch. The proof of this theorem closely follows the proof of Q-GID MAFE scheme in ROM from [GY24]. Specifically, we need to use the augmented N, D, t and T, v from Lemma B.2 and Lemma B.3 respectively as any attacker, by means of the non-programmable $\mathcal{H}$ and unsatisfying queries learn unbounded $(\mathbf{S}, \mathbf{\Delta})$ for any number of GIDs. After this, we have a subtle issue that we do not know which GID is for a satisfying query and which GID is for a unsatisfying query. Hence, we need to guess all the corrupted users and unique indices which can be used for simulation.

After guessing this string, we need to follow the artificial abort based template from [GY24] where we determine whether we can use an adversary's algorithm to attack a specific instantiation of $\mathsf{1MA\text{-}ABFE}$. Note that like in [GY24], we can consider an intermediate object of "partially adaptive $\mathsf{MA\text{-}ABFE}$ in ROM" and proceed from there to make the proof simpler. $\square$

Corollary 13.3 ((poly, Q)-$\mathsf{MA\text{-}ABFE}$ with Static Security)**.** A (poly, Q)-bounded MA-ABFE scheme is said to satisfy static security if the $\mathcal{A}$ declares all the secret key queries before receiving $\mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}}$. We remark that a statically secure (poly, Q)-bounded MA-ABFE scheme can be constructed from polynomially-hard MA-ABE schemes. This is because we can use a PRF key embedded as part of CRS to find all the satisfying GID and use adCSF readily. The construction and security proof are similar to the statically secure MAFE construction's in [GY24].

References

[AG21] Miguel Ambrona and Romain Gay. Multi-authority abe, revisited. *Cryptology ePrint Archive*, 2021.

[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 *Advances in Cryptology–CRYPTO 2013: 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, pages 500–518. Springer, 2013.

[AJ15] Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In *Annual Cryptology Conference*, pages 308–326. Springer, 2015.

[AMVY21] Shweta Agrawal, Monosij Maitra, Narasimha Sai Vempati, and Shota Yamada. Functional encryption for turing machines with dynamic bounded collusion from lwe. In *Advances in Cryptology–CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16–20, 2021, Proceedings, Part IV 41*, pages 239–269. Springer, 2021.

[ARP03] Sattam S Al-Riyami and Kenneth G Paterson. Certificateless public key cryptography. In *International conference on the theory and application of cryptology and information security*, pages 452–473. Springer, 2003.

[ARS24] Damiano Abram, Lawrence Roy, and Peter Scholl. Succinct homomorphic secret sharing. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 301–330. Springer, 2024.

[AV19] Prabhanjan Ananth and Vinod Vaikuntanathan. Optimal bounded-collusion secure functional encryption. In *Theory of Cryptography Conference*, pages 174–198. Springer, 2019.

[BCG$^+$17a] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, and Michele Orrù. Homomorphic secret sharing: optimizations and applications. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 2105–2122, 2017.

[BCG$^+$17b] Zvika Brakerski, Nishanth Chandran, Vipul Goyal, Aayush Jain, Amit Sahai, and Gil Segev. Hierarchical functional encryption. In *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[BF01] Dan Boneh and Matt Franklin. Identity-based encryption from the weil pairing. In *Annual international cryptology conference*, pages 213–229. Springer, 2001.

[BGG$^+$18] Dan Boneh, Rosario Gennaro, Steven Goldfeder, Aayush Jain, Sam Kim, Peter M. R. Rasmussen, and Amit Sahai. Threshold cryptosystems from threshold fully homomorphic encryption. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part I*, volume 10991 of *Lecture Notes in Computer Science*, pages 565–596, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Cham, Switzerland.

[BGI$^+$12] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2):6, 2012.

[BGI16] Elette Boyle, Niv Gilboa, and Yuval Ishai. Breaking the circuit size barrier for secure computation under ddh. In *Annual International Cryptology Conference*, pages 509–539. Springer, 2016.

[BHR12] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 784–796, 2012.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[BS18] Zvika Brakerski and Gil Segev. Function-private functional encryption in the private-key setting. *Journal of Cryptology*, 31:202–225, 2018.

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *Theory of Cryptography: 8th Theory of Cryptography Conference, TCC 2011, Providence, RI, USA, March 28-30, 2011. Proceedings 8*, pages 253–273. Springer, 2011.

[BV15] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. In *FOCS*, 2015.

[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.

[CCV04] Zhaohui Cheng, Richard Comley, and Luminita Vasiu. Remove key escrow from the identity-based encryption system. In *Exploring New Frontiers of Theoretical Informatics: IFIP 18th World Computer Congress TC1 3rd International Conference on Theoretical Computer Science (TCS2004) 22–27 August 2004 Toulouse, France*, pages 37–50. Springer, 2004.

[Cha07] Melissa Chase. Multi-authority attribute based encryption. In *Theory of Cryptography: 4th Theory of Cryptography Conference, TCC 2007, Amsterdam, The Netherlands, February 21-24, 2007. Proceedings 4*, pages 515–534. Springer, 2007.

[CHSS02] Liqun Chen, Keith Harrison, David Soldera, and Nigel P Smart. Applications of multiple trust authorities in pairing based cryptosystems. In *International Conference on Infrastructure Security*, pages 260–275. Springer, 2002.

[CM21] Geoffroy Couteau and Pierre Meyer. Breaking the circuit size barrier for secure computation under quasi-polynomial lpn. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 842–870. Springer, 2021.

[Coc01] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In *Cryptography and Coding: 8th IMA International Conference Cirencester, UK, December 17–19, 2001 Proceedings 8*, pages 360–363. Springer, 2001.

[DG17a] Nico Döttling and Sanjam Garg. From selective IBE to full IBE and selective HIBE. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017: 15th Theory of Cryptography Conference, Part I*, volume 10677 of *Lecture Notes in Computer Science*, pages 372–408, Baltimore, MD, USA, November 12–15, 2017. Springer, Cham, Switzerland.

[DG17b] Nico Döttling and Sanjam Garg. Identity-based encryption from the Diffie-Hellman assumption. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 537–569, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Cham, Switzerland.

[DHRW16] Yevgeniy Dodis, Shai Halevi, Ron D Rothblum, and Daniel Wichs. Spooky encryption and its applications. In *Annual International Cryptology Conference*, pages 93–122. Springer, 2016.

[DIJL23] Quang Dao, Yuval Ishai, Aayush Jain, and Huijia Lin. Multi-party homomorphic secret sharing and sublinear mpc from sparse lpn. In *Annual International Cryptology Conference*, pages 315–348. Springer, 2023.

[DKW21] Pratish Datta, Ilan Komargodski, and Brent Waters. Decentralized multi-authority abe for dnf s from lwe. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 177–209. Springer, 2021.

[DKW23] Pratish Datta, Ilan Komargodski, and Brent Waters. Decentralized multi-authority abe for nc 1 from bdh. *Journal of Cryptology*, 36(2):6, 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.

[DP23] Pratish Datta and Tapas Pal. Decentralized multi-authority attribute-based inner-product fe: Large universe and unbounded. In *IACR International Conference on Public-Key Cryptography*, pages 587–621. Springer, 2023.

[FGJS17] Nelly Fazio, Rosario Gennaro, Tahereh Jafarikhah, and William E Skeith. Homomorphic secret sharing from paillier encryption. In *Provable Security: 11th International Conference, ProvSec 2017, Xi'an, China, October 23-25, 2017, Proceedings 11*, pages 381–399. Springer, 2017.

[GGL24] Rachit Garg, Rishab Goyal, and George Lu. Dynamic collusion functional encryption and multi-authority attribute-based encryption. In *IACR International Conference on Public-Key Cryptography*, pages 69–104. Springer, 2024.

[GGLW22] Rachit Garg, Rishab Goyal, George Lu, and Brent Waters. Dynamic collusion bounded functional encryption from identity-based encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 736–763. Springer, 2022.

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th Annual ACM Symposium on Theory of Computing*, pages 467–476, Palo Alto, CA, USA, June 1–4, 2013. ACM Press.

[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 *Theory of Cryptography: 16th International Conference, TCC 2018, Panaji, India, November 11–14, 2018, Proceedings, Part I 16*, pages 689–718. Springer, 2018.

[GKW16] Rishab Goyal, Venkata Koppula, and Brent Waters. Semi-adaptive security and bundling functionalities made generic and easy. In *Theory of Cryptography Conference*, pages 361–388. Springer, 2016.

[GLW12] Shafi Goldwasser, Allison Lewko, and David A Wilson. Bounded-collusion ibe from key homomorphism. In *Theory of Cryptography: 9th Theory of Cryptography Conference, TCC 2012, Taormina, Sicily, Italy, March 19-21, 2012. Proceedings 9*, pages 564–581. Springer, 2012.

[Goy07] Vipul Goyal. Reducing trust in the pkg in identity based cryptosystems. In *Annual International Cryptology Conference*, pages 430–447. Springer, 2007.

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 89–98, 2006.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Advances in Cryptology–CRYPTO 2013: 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, pages 75–92. Springer, 2013.

[GV20] Rishab Goyal and Satyanarayana Vusirikala. Verifiable registration-based encryption. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020, Part I*, volume 12170 of *Lecture Notes in Computer Science*, pages 621–651, Santa Barbara, CA, USA, August 17–21, 2020. Springer, Cham, Switzerland.

[GVW12] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In *Advances in Cryptology–CRYPTO 2012: 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, pages 162–179. Springer, 2012.

[GY24] Rishab Goyal and Saikumar Yadugiri. Multi-authority functional encryption with bounded collusions from standard assumptions. In Elette Boyle and Mohammad Mahmoody, editors, *TCC 2024: 22nd Theory of Cryptography Conference, Part III*, volume 15366 of *Lecture Notes in Computer Science*, pages 3–30, Milan, Italy, December 2–6, 2024. Springer, Cham, Switzerland.

[GY25] Rishab Goyal and Saikumar Yadugiri. Delegatable ABE with full security from witness encryption. Cryptology ePrint Archive, Paper 2025/407, 2025.

[HKM$^+$23] Taiga Hiroka, Fuyuki Kitagawa, Tomoyuki Morimae, Ryo Nishimaki, Tapas Pal, and Takashi Yamakawa. Certified everlasting secure collusion-resistant functional encryption. Technical report, and more. Cryptology ePrint Archive, Report 2023/236, 2023.

[HMNY22] Taiga Hiroka, Tomoyuki Morimae, Ryo Nishimaki, and Takashi Yamakawa. Certified everlasting functional encryption. *arXiv preprint arXiv:2207.13878*, 2022.

[Kim19] Sam Kim. Multi-authority attribute-based encryption from lwe in the ot model. *Cryptology ePrint Archive*, 2019.

[LBD$^+$04] Byoungcheon Lee, Colin Boyd, Ed Dawson, Kwangjo Kim, Jeongmo Yang, and Seungjae Yoo. Secure key issuing in id-based cryptography. In *Proceedings of the second workshop on Australasian information security, Data Mining and Web Intelligence, and Software Internationalisation-Volume 32*, pages 69–74. Citeseer, 2004.

[LPST16] Huijia Lin, Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation with non-trivial efficiency. In *Public-Key Cryptography–PKC 2016*, pages 447–462. Springer, 2016.

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

[MJ18] Yan Michalevsky and Marc Joye. Decentralized policy-hiding abe with receiver privacy. In *Computer Security: 23rd European Symposium on Research in Computer Security, ESORICS 2018, Barcelona, Spain, September 3-7, 2018, Proceedings, Part II 23*, pages 548–567. Springer, 2018.

[MW16] Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key fhe. In *Advances in Cryptology–EUROCRYPT 2016: 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II 35*, pages 735–763. Springer, 2016.

[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.

[PS08] Kenneth G. Paterson and Sriramkrishnan Srinivasan. Security and anonymity of identity-based encryption with multiple trusted authorities. In Steven D. Galbraith and Kenneth G. Paterson, editors, *PAIRING 2008: 2nd International Conference on Pairing-based Cryptography*, volume 5209 of *Lecture Notes in Computer Science*, pages 354–375, Egham, UK, September 1–3, 2008. Springer Berlin Heidelberg, Germany.

[Rog15] Phillip Rogaway. The moral character of cryptographic work. Cryptology ePrint Archive, Paper 2015/1162, 2015.

[Sha85] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology: Proceedings of CRYPTO 84 4*, pages 47–53. Springer, 1985.

[SS10] Amit Sahai and Hakan Seyalioglu. Worry-free encryption: functional encryption with public keys. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 463–472, 2010.

[SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *Advances in Cryptology–EUROCRYPT 2005: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005. Proceedings 24*, pages 457–473. Springer, 2005.

[Tsa22] Rotem Tsabary. Candidate witness encryption from lattice techniques. In *Annual International Cryptology Conference*, pages 535–559. Springer, 2022.

[VWW22] Vinod Vaikuntanathan, Hoeteck Wee, and Daniel Wichs. Witness encryption and null-io from evasive lwe. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 195–221. Springer, 2022.

[Wat05] Brent Waters. Efficient identity-based encryption without random oracles. In *Advances in Cryptology–EUROCRYPT 2005: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005. Proceedings 24*, pages 114–127. Springer, 2005.

[Wee22] Hoeteck Wee. Optimal broadcast encryption and cp-abe from evasive lattice assumptions. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 217–241. Springer, 2022.

[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.

[WW24] Brent Waters and Daniel Wichs. Adaptively secure attribute-based encryption from witness encryption. In Elette Boyle and Mohammad Mahmoody, editors, *TCC 2024: 22nd Theory of Cryptography Conference, Part III*, volume 15366 of *Lecture Notes in Computer Science*, pages 65–90, Milan, Italy, December 2–6, 2024. Springer, Cham, Switzerland.

[WWW22] Brent Waters, Hoeteck Wee, and David J Wu. Multi-authority abe from lattices without random oracles. In *Theory of Cryptography Conference*, pages 651–679. Springer, 2022.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets. In *27th annual symposium on foundations of computer science (Sfcs 1986)*, pages 162–167. IEEE, 1986.

A Additional Preliminaries

We recall the definition of garbled circuits [Yao86] which is derived from [BHR12].

Definition A.1 (Garb)**.** A Garb scheme (Garble, Eval) is said to be a garbling scheme for P/Poly circuits if satisfies the following properties.

Correctness. For any $\lambda \in \mathbb{N}$, n-ary P/Poly circuit C,

$$\Pr\left[\begin{array}{l} y = \\ C(x_1,\ldots,x_n) \end{array} : \begin{array}{l} (\widetilde{C}, \{w_{i,b}\}_{i\in[n],b\in\{0,1\}}) \leftarrow \mathsf{Garble}(1^\lambda, C) \\ y = \mathsf{Eval}(\widetilde{C}, \{w_{i,x_i}\}_{i\in[n]}) \end{array}\right] = 1$$

Security. For any PPT adversary $\mathcal{A}$, there exists a simulator Sim, $\forall\ \lambda \in \mathbb{N}$,

$$\Pr\left[b' = b \ : \begin{array}{l} (C,x) \leftarrow \mathcal{A}(1^\lambda), b \xleftarrow{\$} \{0,1\}, \\ (\widetilde{C}^{(0)}, \{w^{(0)}_{i,\beta}\}_{i\in[n],\beta\in\{0,1\}}) \leftarrow \mathsf{Garble}(1^\lambda, C) \\ (\widetilde{C}^{(1)}, \{w^{(1)}_{i,x_i}\}_{i\in[n]}) \leftarrow \mathsf{Sim}(1^\lambda, 1^n, 1^{|C|}, C(x)) \\ b' \leftarrow \mathcal{A}(\widetilde{C}^{(b)}, \{w^{(b)}_{i,x_i}\}_{i\in[n]}) \end{array}\right] \le \frac{1}{2} + \mathsf{negl}(\lambda)$$

Remark A.2 ([Yao86])**.** Assuming the existence of one-way functions, garbled circuits for any family of P/Poly circuits exist.

Definition A.3 (PRF)**.** A secure pseudorandom function PRF is a deterministic polynomial-time function that takes inputs from $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$ and a key $K \in \{0,1\}^\lambda$ and outputs values in $\mathcal{R} = \{\mathcal{R}_\lambda\}_{\lambda\in\mathbb{N}}$. We say that the PRF is a secure pseudorandom function if it satisfies the following properties.

Pseudorandomness. For any stateful PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $\forall\ \lambda \in \mathbb{N}$,

$$\Pr\left[b' \leftarrow \mathcal{A}^{\mathcal{O}_b(\cdot)}(1^\lambda) \ : \ b \xleftarrow{\$} \{0,1\}, K \xleftarrow{\$} \{0,1\}^\lambda \right] \le \frac{1}{2} + \mathsf{negl}(\lambda)$$

where $\mathcal{O}_0(\cdot) = \mathsf{PRF}(K,\cdot)$ and $\mathcal{O}_1(\cdot)$ generates random strings in $\mathcal{R}_\lambda$.

Definition A.4 (IBE)**.** An IBE scheme $(\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ is said to be an Identity-Based Encryption scheme (IBE) for identity space $\mathcal{I} = \{\mathcal{I}_\lambda\}_{\lambda \in \mathbb{N}}$ if it satisfies the following properties.

Correctness. For any $\lambda \in \mathbb{N}$, for any identity $\mathsf{ibeID} \in \mathcal{I}_\lambda, |\mathsf{ibeID}| = z = z(\lambda)$, for any message $m \in \{0,1\}^*$[16],

$$\Pr\left[m = \mathsf{Dec}(\mathsf{SK}_{\mathsf{ibeID}}, \mathsf{CT}) \quad : \quad \begin{array}{l} (\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, 1^z), \\ \mathsf{SK}_{\mathsf{ibeID}} \leftarrow \mathsf{KGen}(\mathsf{MSK}, \mathsf{ibeID}), \\ \mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{MPK}, \mathsf{ibeID}, m) \end{array} \right] = 1$$

Security. For any admissible stateful PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for any $\lambda \in \mathbb{N}$,

$$\Pr\left[b \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(\mathsf{CT}) \quad : \quad \begin{array}{l} (\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, 1^z), \\ (\mathsf{ibeID}^*, m) \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(1^\lambda, 1^z, \mathsf{MPK}), \\ b \xleftarrow{\$} \{0,1\}, m_0 := m, m_1 := 0^{|m|}, \\ \mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{MPK}, \mathsf{ibeID}^*, m_b) \end{array} \right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where $\mathcal{O}(\mathsf{ibeID})$ outputs $\mathsf{KGen}(\mathsf{MSK}, \mathsf{ibeID})$ and an admissible adversary queries the oracle $\mathcal{O}(\cdot)$ with $\mathsf{ibeID} \neq \mathsf{ibeID}^*$.

Remark A.5 (Multi-challenge security)**.** Although Definition A.4 is defined for a single challenge identity, the same holds for to multiple challenge queries by a simple hybrid argument. We will use this version in this work.

B Relevant Material from [GY24]

In this section, we provide definitions and results from [GY24] that will be useful for the reader.

B.1 Distributed Client-Server Framework

Syntax. A distributed client-server framework (dCSF) for circuit class $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ and $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda \in \mathbb{N}}$ consists of the following polynomial time algorithms.

$\mathsf{ServEnc}(1^\lambda, 1^Q, 1^n, 1^s, \mathsf{id}, \mathsf{GID}, x, \mathbf{\Delta}) \rightarrow \{\widehat{x}^u_{\mathsf{id},\mathsf{GID}}\}_{u \in [N]}$**.** The deterministic server encoding algorithm takes as input the query bound Q, number of authorities n, maximum size of circuit s, authority identifier id, user global identifier $\mathsf{GID} \in \mathcal{GID}_\lambda$, input x, cover-free set $\mathbf{\Delta}$, and outputs input encodings $\{\widehat{x}^u_{\mathsf{id},\mathsf{GID}}\}_{u \in [N]}$.

$\mathsf{CliEnc}(1^\lambda, 1^Q, 1^n, 1^s, C) \rightarrow \{\widehat{C}^u\}_{u \in [N]}$**.** The probabilistic client encoding algorithm takes as input query bound Q, number of authorities n, maximum size of circuit s, circuit $C \in \mathcal{C}_\lambda$, and outputs client encodings $\{\widehat{C}^u\}_{u \in [N]}$.

$\mathsf{UserComp}(\{\widehat{x}^u_{\mathsf{id},\mathsf{GID}}\}_{\mathsf{id} \in [n]}, \widehat{C}^u) \rightarrow \widehat{y}^u_{\mathsf{GID}}$**.** The deterministic user computation algorithm takes the input encodings for a $\mathsf{GID} \in \mathcal{GID}_\lambda$ from all authorities, $\{\widehat{x}^u_{\mathsf{id},\mathsf{GID}}\}_{\mathsf{id} \in [n]}$, client encoding $\widehat{C}^u$, and computes the output encoding $\widehat{y}^u_{\mathsf{GID}}$.

[16]Recall that IBE can encrypt messages of arbitrary length using hybrid encryption.

$\mathsf{Decode}(\{\widehat{y}^u_{\mathsf{GID}}\}_{u\in\mathbf{S}},\mathbf{S}) \to y$**.** The deterministic decoding algorithm takes as input output encodings from users in $\mathbf{S} \subseteq [N]$, $\{\widehat{y}^u_{\mathsf{GID}}\}_{u\in\mathbf{S}}$ and outputs the value y.

Definition B.1 (dCSF)**.** A dCSF scheme ($\mathsf{ServEnc}, \mathsf{CliEnc}, \mathsf{UserComp}, \mathsf{Decode}$) is said to be a dCSF scheme for circuit class $\mathcal{C}$ and $\mathcal{GID}$ if it satisfies the following properties.

Correctness. For any $\lambda \in \mathbb{N}, Q = Q(\lambda), n = n(\lambda), s = s(\lambda), \mathsf{id} \in [n], C \in \mathcal{C}_\lambda, \mathsf{GID} \in \mathcal{GID}_\lambda, u \in \mathbf{S} \subseteq [N], |\mathbf{S}| = D$,

$$\Pr\left[\begin{array}{l} C(x_1,\ldots,x_n) = \\ \mathsf{Decode}(\{\widehat{y}^u_{\mathsf{GID}}\}_{u\in\mathbf{S}},\mathbf{S}) \end{array} : \begin{array}{l} \{\widehat{x}^u_{\mathsf{id},\mathsf{GID}}\}_{u\in[N]} = \mathsf{ServEnc}(1^\lambda, 1^Q, 1^n, \\ 1^s, \mathsf{id}, \mathsf{GID}, x, \mathbf{\Delta}), \\ \{\widehat{C}^u\}_{u\in[N]} \leftarrow \mathsf{CliEnc}(1^\lambda, 1^Q, 1^n, 1^s, C), \\ \widehat{y}^u_{\mathsf{GID}} \leftarrow \mathsf{UserComp}(\{\widehat{x}^u_{\mathsf{id},\mathsf{GID}}\}_{\mathsf{id}}, \widehat{C}^u) \end{array}\right] = 1$$

Security. For any admissible adversary $\mathcal{A}$, there exists a stateful simulator Sim such that $\forall\ \lambda \in \mathbb{N}$,

$$\left\{\mathsf{dCSF\text{-}Expt}^{\mathsf{Chal}}_{\mathcal{A},0}(1^\lambda)\right\} \approx_c \left\{\mathsf{dCSF\text{-}Expt}^{\mathsf{Sim}}_{\mathcal{A},1}(1^\lambda)\right\}$$

where definitions of $\mathsf{dCSF\text{-}Expt}^{\mathsf{Chal}}_{\mathcal{A},0}(1^\lambda)$, $\mathsf{dCSF\text{-}Expt}^{\mathsf{Sim}}_{\mathcal{A},1}(1^\lambda)$, and admissible adversary are as follows — A stateful PPT machine $\mathcal{A}$ is admissible if

- $\mathcal{A}$ makes at most Q server encoding queries across n authorities such that each authority is queried at most once per GID.
- For all server encoding queries, $\mathbf{S}_{\mathsf{GID}}, \mathbf{\Delta}_{\mathsf{GID}}$ should be unique for a GID. And all $\{\mathbf{S}_{\mathsf{GID}}\}$ and $\{\mathbf{\Delta}_{\mathsf{GID}}\}$ satisfy Lemma B.2 and Lemma B.3 respectively. $\widetilde{\mathbf{S}} = [N]\backslash(\bigcup_{\mathsf{GID}\neq\mathsf{GID}'} \mathbf{S}_{\mathsf{GID}} \cap \mathbf{S}_{\mathsf{GID}'})$ and j^*_{GID} is the unique index in $\mathbf{\Delta}_{\mathsf{GID}}$ for every GID.

<u>$\mathsf{dCSF\text{-}Expt}^{\mathsf{Chal}}_{\mathcal{A},0}(1^\lambda)$.</u>

Setup. $\mathcal{A}$ sends Q, n, s, set of non-corrupted users $\widetilde{\mathbf{S}}$.

Pre-challenge Queries. $\mathcal{A}$ queries for server encodings with $((\mathsf{id}, \mathsf{GID}, x, \mathbf{\Delta}_{\mathsf{GID}}), \mathbf{S}_{\mathsf{GID}})$. Chal runs $\{\widehat{x}^u_{\mathsf{id},\mathsf{GID}}\}_{u\in[N]} = \mathsf{ServEnc}(1^\lambda, 1^Q, 1^n, 1^s, \mathsf{id}, \mathsf{GID}, x, \mathbf{\Delta}_{\mathsf{GID}})$, and sends $\{\widehat{x}^u_{\mathsf{id},\mathsf{GID}}\}_{u\in\mathbf{S}_{\mathsf{GID}}}$.

Challenge Query. $\mathcal{A}$ sends circuit C, and $\{j^*_{\mathsf{GID}}\}_{\mathsf{GID}}$ for each completed GID in the previous phase. Chal samples $\{\widehat{C}^u\}_{u\in[N]} \leftarrow \mathsf{CliEnc}(1^\lambda, 1^Q, 1^n, 1^s, C)$ and for each GID, $\{\{\widehat{y}^u_{\mathsf{GID}}\} \leftarrow \mathsf{UserComp}(\{\widehat{x}^u_{\mathsf{id},\mathsf{GID}}\}_{\mathsf{id}\in[n]}, \widehat{C}^u)\}_{u\in\mathbf{S}_{\mathsf{GID}}}$ and sends $\{\widehat{C}^u\}_{u\notin\widetilde{\mathbf{S}}}$ and $\{\widehat{y}^u_{\mathsf{GID}}\}_{\mathsf{GID},u\in\mathbf{S}_{\mathsf{GID}}}$ to $\mathcal{A}$.

Post-challenge Queries. This is similar to pre-challenge query phase, except that $\mathcal{A}$ sends an additional input j^*_{GID} when for this GID, this is the last authority to be queried. In turn, Chal responds additionally with $\{\widehat{y}^u_{\mathsf{GID}}\}_{u\in\mathbf{S}_{\mathsf{GID}}}$ that it computes using $\mathsf{UserComp}$.

Guess. Output whatever $\mathcal{A}$ outputs.

<u>$\mathsf{dCSF\text{-}Expt}^{\mathsf{Sim}}_{\mathcal{A},1}(1^\lambda)$.</u>

Setup. $\mathcal{A}$ sends Q, n, s, set of non-corrupted users $\widetilde{\mathbf{S}}$. Initiate $\mathsf{Sim}(1^\lambda, 1^Q, 1^n, 1^s, \widetilde{\mathbf{S}})$.

Pre-challenge Queries. $\mathcal{A}$ queries for server encodings with $((\mathsf{id}, \mathsf{GID}, x, \mathbf{\Delta}_{\mathsf{GID}}), \mathbf{S}_{\mathsf{GID}})$. Respond with $\{\widehat{x}^u_{\mathsf{id},\mathsf{GID}}\}_{u \in \mathbf{S}_{\mathsf{GID}}} = \mathsf{Sim}(\mathsf{id}, \mathsf{GID}, x, \mathbf{\Delta}_{\mathsf{GID}})$.

Challenge Query. $\mathcal{A}$ sends circuit C, and $\{j^*_{\mathsf{GID}}\}_{\mathsf{GID}}$ for each completed GID in the previous phase. Respond with $\{\widehat{C}^u\}_{u \notin \widetilde{\mathbf{S}}}, \{\widehat{y}^u_{\mathsf{GID}}\}_{\mathsf{GID}, u \in \mathbf{S}_{\mathsf{GID}}} \leftarrow \mathsf{Sim}(1^{|C|}, \mathcal{V}, \{j^*\mathsf{GID}\}_{\mathsf{GID}})$. Here, $\mathcal{V} = \{(\mathsf{GID}, X = \{x_{\mathsf{id},\mathsf{GID}}\}_{\mathsf{id} \in [n]}, C(X))$ for all GID that was used to query all authorities in pre-challenge query phase$\}$.

Post-challenge Queries. This is similar to pre-challenge query phase, except that $\mathcal{A}$ sends an additional input j^*_{GID} when for this GID, this is the last authority to be queried. Respond with $(\{\widehat{x}^u_{\mathsf{id},\mathsf{GID}}\}, \{\widehat{y}^u_{\mathsf{GID}}\})_{u \in \mathbf{S}_{\mathsf{GID}}} \leftarrow \mathsf{Sim}^{C(\cdot)}(\mathsf{id}, \mathsf{GID}, x, \mathbf{\Delta}_{\mathsf{GID}}, j^*_{\mathsf{GID}})$. Here, Sim queries $C(\cdot)$ with $X = (x_{1,\mathsf{GID}}, \ldots, x_{n,\mathsf{GID}})$ and receives $C(X)$.

Guess. Output whatever $\mathcal{A}$ outputs.

B.2 Augmented Statistical Lemmas

We provide the following lemmas used in Section 12 without proofs.

Lemma B.2 (Small Pairwise Intersection)**.** Let $P = P(\lambda), Q = Q(\lambda)$ such that $P \geq Q$. Let $t = \Theta(Q\lambda), D = \Theta(t), N = \Theta(Q^2 t)$ such that $D > 3t$. Let $\mathbf{S}_1, \ldots, \mathbf{S}_P$ be independently and uniformly drawn sets from $[N]$ of size D. Then for any $\mathcal{R} \subseteq [P], |\mathcal{R}| = Q$, with all but negligible probability,

$$\left| \bigcup_{i,j \in \mathcal{R}, i \neq j} \mathbf{S}_i \cap \mathbf{S}_j \right| \leq t$$

Lemma B.3 (Cover-freeness)**.** Let $P = P(\lambda), Q = Q(\lambda)$ such that $P \geq Q$. Let $v = \Theta(Q^2\lambda), T = vQ$. Let $\mathbf{\Delta}_1, \ldots, \mathbf{\Delta}_P$ be independently and uniformly drawn sets from $[T]$ of size v. Then for any $\mathcal{R} \subseteq [P], |\mathcal{R}| = Q, i \in \mathcal{R}$, with all but negligible probability,

$$\mathbf{\Delta}_i \setminus \bigcup_{j \neq i, j \in \mathcal{R}} \mathbf{\Delta}_j \neq \emptyset$$

C Proofs from Section 5

In this section, we provide full proof of Theorem 5.3.

$\mathbf{Hyb}^{\mathcal{A}}_0(1^\lambda)$**.** This is the real experiment with Chal.

Setup. $\mathcal{A}$ sends Q, s to Chal. Chal sends $\mathsf{CRS} := (\lambda, Q, s)$ to $\mathcal{A}$.

Auth Setup Queries. $\mathcal{A}$ sends $(\mathsf{id}, \ell_{\mathsf{id}})$. Chal samples $(\mathsf{bfe.mpk}_{\mathsf{id}}, \mathsf{bfe.msk}_{\mathsf{id}}) \leftarrow \mathsf{BFE.Setup}(1^\lambda, 1^Q, 1^{\lambda + |\mathsf{GID}| + \ell_{\mathsf{id}}}, 1^{\kappa_{\mathsf{id}}})$. Set $\mathsf{MPK}_{\mathsf{id}} = (\ell_{\mathsf{id}}, \mathsf{bfe.mpk}_{\mathsf{id}})$, and $\mathsf{MSK}_{\mathsf{id}} = (\ell_{\mathsf{id}}, \mathsf{bfe.msk}_{\mathsf{id}})$. Chal sends $\mathsf{MPK}_{\mathsf{id}}$ to $\mathcal{A}$.

Secret Key Queries. $\mathcal{A}$ sends $(\mathsf{id}, \mathsf{GID}, x)$. Chal samples $\mathsf{bfe.sk}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{BFE.KGen}(\mathsf{bfe.msk}_{\mathsf{id}}, (\mathsf{id}, \mathsf{GID}, x))$ and sends $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} := \mathsf{bfe.sk}_{\mathsf{id},\mathsf{GID},x}$ to $\mathcal{A}$.

Challenge Query. $\mathcal{A}$ sends circuit $\mathcal{I}, C$ to Chal. Let $n = |\mathcal{I}|$ and $\mathsf{id}_1, \ldots, \mathsf{id}_n$ be the increasing ordering of elements in $\mathcal{I}$.

- Chal samples $\mathsf{mafe.crs}, \{(\mathsf{mafe.mpk}_i, \mathsf{mafe.msk}_i)\}_{i\in[n]} \leftarrow \mathsf{tMAFE.GSetup}(1^\lambda, 1^Q, 1^n, 1^s, 1^{\ell_{\mathsf{id}_1}}, \ldots, 1^{\ell_{\mathsf{id}_n}})$.
- $\forall\ i \in [n]$, $\mathsf{bfe.ct}_{\mathsf{id}_i} \leftarrow \mathsf{BFE.Enc}(\mathsf{bfe.mpk}_{\mathsf{id}_i}, \mathsf{tMAFE.KGen}(\cdot, \mathsf{mafe.msk}_i, \{\mathsf{mafe.mpk}_{i'}\}_{i'}, \cdot, \cdot))$.
- Sample $\mathsf{mafe.ct} \leftarrow \mathsf{tMAFE.Enc}(\{\mathsf{mafe.mpk}_i\}_{i\in[n]}, C)$.

Send $\mathsf{CT} := (\mathsf{mafe.ct}, \mathcal{I}, \{\mathsf{bfe.ct}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{I}})$ to $\mathcal{A}$.

Auth Setup Queries. $\mathcal{A}$ sends $(\mathsf{id}, \ell_{\mathsf{id}})$. Chal samples $(\mathsf{bfe.mpk}_{\mathsf{id}}, \mathsf{bfe.msk}_{\mathsf{id}}) \leftarrow \mathsf{BFE.Setup}(1^\lambda, 1^Q, 1^{\lambda+|\mathsf{GID}|+\ell_{\mathsf{id}}}, 1^{\kappa_{\mathsf{id}}})$. Set $\mathsf{MPK}_{\mathsf{id}} = (\ell_{\mathsf{id}}, \mathsf{bfe.mpk}_{\mathsf{id}})$, and $\mathsf{MSK}_{\mathsf{id}} = (\ell_{\mathsf{id}}, \mathsf{bfe.msk}_{\mathsf{id}})$. Chal sends $\mathsf{MPK}_{\mathsf{id}}$ to $\mathcal{A}$.

Secret Key Queries. $\mathcal{A}$ sends $(\mathsf{id}, \mathsf{GID}, x)$. Chal samples $\mathsf{bfe.sk}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{BFE.KGen}(\mathsf{bfe.msk}_{\mathsf{id}}, (\mathsf{id}, \mathsf{GID}, x))$ and sends $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} := \mathsf{bfe.sk}_{\mathsf{id},\mathsf{GID},x}$ to $\mathcal{A}$.

Guess Phase. Output whatever $\mathcal{A}$ outputs.

$\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$. In this hybrid, we will simulate the pre-challenge queries using $(\mathsf{S}_0, \mathsf{S}_1)$ of BFE instantiations. The changes are highlighted in red.

Setup. $\mathcal{A}$ sends Q, s to Chal. Chal sends $\mathsf{CRS} := (\lambda, Q, s)$ to $\mathcal{A}$.

Auth Setup Queries. $\mathcal{A}$ sends $(\mathsf{id}, \ell_{\mathsf{id}})$. Chal samples $(\mathsf{bfe.mpk}_{\mathsf{id}}, \mathsf{bfe.msk}_{\mathsf{id}}, \mathsf{bfe.st}_{\mathsf{id}}) \leftarrow \mathsf{BFE.S}_0(1^\lambda, 1^Q, 1^{\lambda+|\mathsf{GID}|+\ell_{\mathsf{id}}}, 1^{\kappa_{\mathsf{id}}})$. Set $\mathsf{MPK}_{\mathsf{id}} = (\ell_{\mathsf{id}}, \mathsf{bfe.mpk}_{\mathsf{id}})$, and $\mathsf{MSK}_{\mathsf{id}} = (\ell_{\mathsf{id}}, \mathsf{bfe.msk}_{\mathsf{id}})$. Chal sends $\mathsf{MPK}_{\mathsf{id}}$ to $\mathcal{A}$.

Secret Key Queries. $\mathcal{A}$ sends $(\mathsf{id}, \mathsf{GID}, x)$. Chal samples $\mathsf{bfe.sk}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{BFE.S}_1(\mathsf{bfe.mpk}_{\mathsf{id}}, \mathsf{bfe.msk}_{\mathsf{id}}, \mathsf{bfe.st}_{\mathsf{id}}, (\mathsf{id}, \mathsf{GID}, x))$ and sends $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} := \mathsf{bfe.sk}_{\mathsf{id},\mathsf{GID},x}$ to $\mathcal{A}$.

Challenge Query, Post-challenge Queries, Guess Phase. Do this exactly like $\mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)$.

$\mathbf{Hyb}_{2,j}^{\mathcal{A}}(1^\lambda)$. for $j \in [n+1]$. In this hybrid, we will simulate BFE instantiations of $\mathsf{id}_1, \ldots, \mathsf{id}_{j-1} \in \mathcal{I}$. The changes are highlighted in red.

Setup. $\mathcal{A}$ sends Q, s. Send $\mathsf{CRS} := (\lambda, Q, s)$ to $\mathcal{A}$.

Auth Setup Queries. $\mathcal{A}$ sends $(\mathsf{id}, \ell_{\mathsf{id}})$. Sample $(\mathsf{bfe.mpk}_{\mathsf{id}}, \mathsf{bfe.msk}_{\mathsf{id}}, \mathsf{bfe.st}_{\mathsf{id}}) \leftarrow \mathsf{BFE.S}_0(1^\lambda, 1^Q, 1^{\lambda+|\mathsf{GID}|+\ell_{\mathsf{id}}}, 1^{\kappa_{\mathsf{id}}})$. Set $\mathsf{MPK}_{\mathsf{id}} = (\ell_{\mathsf{id}}, \mathsf{bfe.mpk}_{\mathsf{id}})$, and $\mathsf{MSK}_{\mathsf{id}} = (\ell_{\mathsf{id}}, \mathsf{bfe.msk}_{\mathsf{id}})$. Send $\mathsf{MPK}_{\mathsf{id}}$ to $\mathcal{A}$.

Secret Key Queries. $\mathcal{A}$ sends $(\mathsf{id}, \mathsf{GID}, x)$. Sample $\mathsf{bfe.sk}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{BFE.S}_1(\mathsf{bfe.mpk}_{\mathsf{id}}, \mathsf{bfe.msk}_{\mathsf{id}}, \mathsf{bfe.st}_{\mathsf{id}}, (\mathsf{id}, \mathsf{GID}, x))$ and send $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} := \mathsf{bfe.sk}_{\mathsf{id},\mathsf{GID},x}$ to $\mathcal{A}$.

Challenge Query. $\mathcal{A}$ sends circuit $\mathcal{I}, C$. Let $n = |\mathcal{I}|$ and $\mathsf{id}_1, \ldots, \mathsf{id}_n$ be the increasing ordering of elements in $\mathcal{I}$.

- Sample $\mathsf{mafe.crs}, \{(\mathsf{mafe.mpk}_i, \mathsf{mafe.msk}_i)\}_{i\in[n]} \leftarrow \mathsf{tMAFE.GSetup}(1^\lambda, 1^Q, 1^n, 1^s, 1^{\ell_{\mathsf{id}_1}}, \ldots, 1^{\ell_{\mathsf{id}_n}})$.

- $\forall\, i \in [n]$, if $i \geq j$, $\mathsf{bfe.ct}_{\mathsf{id}_i} \leftarrow \mathsf{BFE.Enc}(\mathsf{bfe.mpk}_{\mathsf{id}_i}, F_i)$. Otherwise, $\mathsf{bfe.ct}_{\mathsf{id}_i} \leftarrow \mathsf{BFE.S}_2(\mathsf{st}_{\mathsf{id}_i}, 1^{|F_i|}, \mathcal{V}_i)$. Here $F_i = \mathsf{tMAFE.KGen}(\cdot, \mathsf{mafe.msk}_i, \{\mathsf{mafe.mpk}_{i'}\}_{i'}, \cdot, \cdot)$ and $\mathcal{V}_i = \{(X, F_i(X)) : X = (\mathsf{id}_i, \mathsf{GID}, x_{\mathsf{GID},\mathsf{id}_i})$ for all GID queries in previous phase$\}$.
- Sample $\mathsf{mafe.ct} \leftarrow \mathsf{tMAFE.Enc}(\{\mathsf{mafe.mpk}_i\}_{i\in[n]}, C)$.

Send $\mathsf{CT} := (\mathsf{mafe.ct}, \mathcal{I}, \{\mathsf{bfe.ct}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{I}})$ to $\mathcal{A}$.

Auth Setup Queries. $\mathcal{A}$ sends $(\mathsf{id}, \ell_{\mathsf{id}})$. Sample $(\mathsf{bfe.mpk}_{\mathsf{id}}, \mathsf{bfe.msk}_{\mathsf{id}}) \leftarrow \mathsf{BFE.Setup}(1^\lambda, 1^Q, 1^{\lambda+|\mathsf{GID}|+\ell_{\mathsf{id}}}, 1^{\kappa_{\mathsf{id}}})$. Set $\mathsf{MPK}_{\mathsf{id}} = (\ell_{\mathsf{id}}, \mathsf{bfe.mpk}_{\mathsf{id}})$, $\mathsf{MSK}_{\mathsf{id}} := (\ell_{\mathsf{id}}, \mathsf{bfe.msk}_{\mathsf{id}})$. Send $\mathsf{MPK}_{\mathsf{id}}$ to $\mathcal{A}$.

Secret Key Queries. $\mathcal{A}$ sends $(\mathsf{id}, \mathsf{GID}, x)$. If $\mathsf{id} \in \{\mathsf{id}_1, \ldots, \mathsf{id}_{j-1}\}$, sample $\mathsf{bfe.sk}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{BFE.S}_3^{F_{\mathsf{id}}(\cdot)}(\mathsf{st}_{\mathsf{id}}, (\mathsf{id}, \mathsf{GID}, x))$. Otherwise, $\mathsf{bfe.sk}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{BFE.KGen}(\mathsf{bfe.msk}_{\mathsf{id}}, (\mathsf{id}, \mathsf{GID}, x))$. Send $\mathsf{bfe.sk}_{\mathsf{id},\mathsf{GID},x}$ to $\mathcal{A}$.

Guess Phase. Output whatever $\mathcal{A}$ outputs.

$\mathbf{Hyb}_3^{\mathcal{A}}(1^\lambda)$. In this hybrid, we will simulate tMAFE. The changes are highlighted in red.

Setup. $\mathcal{A}$ sends Q, s. Send $\mathsf{CRS} := (\lambda, Q, s)$ to $\mathcal{A}$.

Auth Setup Queries. $\mathcal{A}$ sends $(\mathsf{id}, \ell_{\mathsf{id}})$. Sample $(\mathsf{bfe.mpk}_{\mathsf{id}}, \mathsf{bfe.msk}_{\mathsf{id}}, \mathsf{bfe.st}_{\mathsf{id}}) \leftarrow \mathsf{BFE.S}_0(1^\lambda, 1^Q, 1^{\lambda+|\mathsf{GID}|+\ell_{\mathsf{id}}}, 1^{\kappa_{\mathsf{id}}})$. Set $\mathsf{MPK}_{\mathsf{id}} = (\ell_{\mathsf{id}}, \mathsf{bfe.mpk}_{\mathsf{id}})$, and $\mathsf{MSK}_{\mathsf{id}} = (\ell_{\mathsf{id}}, \mathsf{bfe.msk}_{\mathsf{id}})$. Send $\mathsf{MPK}_{\mathsf{id}}$ to $\mathcal{A}$.

Secret Key Queries. $\mathcal{A}$ sends $(\mathsf{id}, \mathsf{GID}, x)$. Sample $\mathsf{bfe.sk}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{BFE.S}_1(\mathsf{bfe.mpk}_{\mathsf{id}}, \mathsf{bfe.msk}_{\mathsf{id}}, \mathsf{bfe.st}_{\mathsf{id}}, (\mathsf{id}, \mathsf{GID}, x))$ and send $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} := \mathsf{bfe.sk}_{\mathsf{id},\mathsf{GID},x}$ to $\mathcal{A}$.

Challenge Query. $\mathcal{A}$ sends circuit $\mathcal{I}, C$. Let $n = |\mathcal{I}|$ and $\mathsf{id}_1, \ldots, \mathsf{id}_n$ be the increasing ordering of elements in $\mathcal{I}$.

- Sample $\mathsf{mafe.crs}, \{\mathsf{mafe.mpk}_{\mathsf{id}}\}_{\mathsf{id}\in[n]} \leftarrow \mathsf{tMAFE.Sim}(1^\lambda, 1^Q, 1^n, 1^s, 1^{\ell_{\mathsf{id}_1}}, \ldots, 1^{\ell_{\mathsf{id}_n}})$.
- $\forall\, i \in [n]$, $\mathsf{bfe.ct}_{\mathsf{id}_i} \leftarrow \mathsf{BFE.S}_2(\mathsf{st}_{\mathsf{id}_i}, 1^{|F_i|}, \mathcal{V}_i)$. Here $\mathcal{V}_i = \{(X, \mathsf{tMAFE.Sim}(\mathsf{id}_i, \mathsf{GID}, x_{\mathsf{GID},\mathsf{id}_i})) :$ for all GID queries in previous phase$\}$.
- Sample $\mathsf{mafe.ct} \leftarrow \mathsf{tMAFE.Sim}(1^{|C|}, \mathcal{V}_C)$ where $\mathcal{V}_C$ is as defined in Definition 4.1.

Send $\mathsf{CT} := (\mathsf{mafe.ct}, \mathcal{I}, \{\mathsf{bfe.ct}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{I}})$ to $\mathcal{A}$.

Auth Setup Queries. $\mathcal{A}$ sends $(\mathsf{id}, \ell_{\mathsf{id}})$. Sample $(\mathsf{bfe.mpk}_{\mathsf{id}}, \mathsf{bfe.msk}_{\mathsf{id}}) \leftarrow \mathsf{BFE.Setup}(1^\lambda, 1^Q, 1^{\lambda+|\mathsf{GID}|+\ell_{\mathsf{id}}}, 1^{\kappa_{\mathsf{id}}})$. Set $\mathsf{MPK}_{\mathsf{id}} := (\ell_{\mathsf{id}}, \mathsf{bfe.mpk}_{\mathsf{id}})$, $\mathsf{MSK}_{\mathsf{id}} := (\ell_{\mathsf{id}}, \mathsf{bfe.msk}_{\mathsf{id}})$. Send $\mathsf{MPK}_{\mathsf{id}}$ to $\mathcal{A}$.

Secret Key Queries. $\mathcal{A}$ sends $(\mathsf{id}, \mathsf{GID}, x)$. If $\mathsf{id} \in \mathcal{I}$, sample $\mathsf{bfe.sk}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{BFE.S}_3^{F'_{\mathsf{id}}(\cdot)}(\mathsf{st}_{\mathsf{id}}, (\mathsf{id}, \mathsf{GID}, x))$. Otherwise, sample $\mathsf{bfe.sk}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{BFE.KGen}(\mathsf{bfe.msk}_{\mathsf{id}}, (\mathsf{id}, \mathsf{GID}, x))$ and send $\mathsf{bfe.sk}_{\mathsf{id},\mathsf{GID},x}$ to $\mathcal{A}$. Here $F'_{\mathsf{id}}(\cdot)$ responds with $\mathsf{tMAFE.Sim}(\mathsf{id}, \mathsf{GID}, x, \mathsf{V}_C)$ where V_C is as defined in Definition 4.1.

Guess Phase. Output whatever $\mathcal{A}$ outputs.

Lemma C.1. $\mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$ are identical.

Proof. This follows from the security of BFE where pre-challenge simulator $(\mathsf{BFE.S}_0, \mathsf{BFE.S}_1)$ behave exactly as $\mathsf{BFE.Setup}, \mathsf{BFE.KGen}$ respectively. ☐

Lemma C.2. $\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_{2,1}^{\mathcal{A}}(1^\lambda)$ are identical.

Proof. As we are not simulating any instantiations of BFE in $\mathbf{Hyb}_{2,1}^{\mathcal{A}}(1^\lambda)$, these hybrids are identically distributed. ☐

Lemma C.3. Assuming the security of BFE, $\mathbf{Hyb}_{2,j}^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_{2,j+1}^{\mathcal{A}}(1^\lambda)$ for $j \in [n]$ are computationally indistinguishable.

Proof. Assume that there exists a PPT adversary that can distinguish between $\mathbf{Hyb}_{2,j}^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_{2,j+1}^{\mathcal{A}}(1^\lambda)$ with non-negligible advantage $\epsilon(\lambda)$, i.e,

$$\left|\Pr\left[1 \leftarrow \mathbf{Hyb}_{2,j}^{\mathcal{A}}(1^\lambda)\right] - \Pr\left[1 \leftarrow \mathbf{Hyb}_{2,j+1}^{\mathcal{A}}(1^\lambda)\right]\right| > \epsilon(\lambda)$$

We will construct a reduction adversary $\mathcal{B}$ that can break the security of BFE with non-negligible probability. The description of $\mathcal{B}^{\mathcal{O}}$ is as follows.

Setup. $\mathcal{A}$ sends Q, s. Send $\mathsf{CRS} := (\lambda, Q, s)$ to $\mathcal{A}$. As $\mathcal{A}$ runs in polynomial time, it makes $q_{\mathsf{auth}} = \mathsf{poly}(\lambda)$ many authority setup queries. Choose $N^* \xleftarrow{\$} [q_{\mathsf{auth}}]$.

Auth Setup Queries. $\mathcal{A}$ sends $(\mathsf{id}, \ell_{\mathsf{id}})$. If this is the N^*-th authority setup query, receive $\mathsf{bfe.mpk}_{\mathsf{id}} \leftarrow \mathcal{O}(1^Q, 1^{\lambda+|\mathsf{GID}|+\ell_{\mathsf{id}}}, 1^{\kappa_{\mathsf{id}}})$ and set $\mathsf{id}^* = \mathsf{id}$. Otherwise, sample $(\mathsf{bfe.mpk}_{\mathsf{id}}, \mathsf{bfe.msk}_{\mathsf{id}}, \mathsf{bfe.st}_{\mathsf{id}}) \leftarrow \mathsf{BFE.S}_0(1^\lambda, 1^Q, 1^{\lambda+|\mathsf{GID}|+\ell_{\mathsf{id}}}, 1^{\kappa_{\mathsf{id}}})$. Set $\mathsf{MPK}_{\mathsf{id}} = (\ell_{\mathsf{id}}, \mathsf{bfe.mpk}_{\mathsf{id}})$, and $\mathsf{MSK}_{\mathsf{id}} = (\ell_{\mathsf{id}}, \mathsf{bfe.msk}_{\mathsf{id}})$. Send $\mathsf{MPK}_{\mathsf{id}}$ to $\mathcal{A}$.

Secret Key Queries. $\mathcal{A}$ sends $(\mathsf{id}, \mathsf{GID}, x)$. If $\mathsf{id}^* = \mathsf{id}$, $\mathsf{bfe.sk}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathcal{O}(\mathsf{id}, \mathsf{GID}, x)$. Sample $\mathsf{bfe.sk}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{BFE.S}_1(\mathsf{bfe.mpk}_{\mathsf{id}}, \mathsf{bfe.msk}_{\mathsf{id}}, \mathsf{bfe.st}_{\mathsf{id}}, (\mathsf{id}, \mathsf{GID}, x))$ and send $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} := \mathsf{bfe.sk}_{\mathsf{id},\mathsf{GID},x}$ to $\mathcal{A}$.

Challenge Query. $\mathcal{A}$ sends circuit $\mathcal{I}, C$. Let $n = |\mathcal{I}|$ and $\mathsf{id}_1, \ldots, \mathsf{id}_n$ be the increasing ordering of elements in $\mathcal{I}$. If $\mathsf{id}_j \neq \mathsf{id}^*$, abort and output $b' \xleftarrow{\$} \{0,1\}$. Otherwise,

- Sample $\mathsf{mafe.crs}, \{(\mathsf{mafe.mpk}_i, \mathsf{mafe.msk}_i)\}_{i\in[n]} \leftarrow \mathsf{tMAFE.GSetup}(1^\lambda, 1^Q, 1^n, 1^s, 1^{\ell_{\mathsf{id}_1}}, \ldots, 1^{\ell_{\mathsf{id}_n}})$.
- $\forall\ i \in [n]$, if $i > j$, $\mathsf{bfe.ct}_{\mathsf{id}_i} \leftarrow \mathsf{BFE.Enc}(\mathsf{bfe.mpk}_{\mathsf{id}_i}, F_i)$. Otherwise, if $i = j$, $\mathsf{bfe.ct}_{\mathsf{id}_i} \leftarrow \mathcal{O}(F_i)$. Otherwise, $\mathsf{bfe.ct}_{\mathsf{id}_i} \leftarrow \mathsf{BFE.S}_2(\mathsf{st}_{\mathsf{id}_i}, 1^{|F_i|}, \mathcal{V}_i)$. Here $F_i = \mathsf{tMAFE.KGen}(\cdot, \mathsf{mafe.msk}_i, \{\mathsf{mafe.mpk}_{i'}\}_{i'}, \cdot, \cdot)$ and $\mathcal{V}_i = \{(X, F_i(X)) : \forall\ X = (\mathsf{id}_i, \mathsf{GID}, x_{\mathsf{GID},\mathsf{id}_i})\}$.
- Sample $\mathsf{mafe.ct} \leftarrow \mathsf{tMAFE.Enc}(\{\mathsf{mafe.mpk}_i\}_{i\in[n]}, C)$.

Send $\mathsf{CT} := (\mathsf{mafe.ct}, \mathcal{I}, \{\mathsf{bfe.ct}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{I}})$ to $\mathcal{A}$.

Auth Setup Queries. $\mathcal{A}$ sends $(\mathsf{id}, \ell_{\mathsf{id}})$. Sample $(\mathsf{bfe.mpk}_{\mathsf{id}}, \mathsf{bfe.msk}_{\mathsf{id}}) \leftarrow \mathsf{BFE.Setup}(1^\lambda, 1^Q, 1^{\lambda+|\mathsf{GID}|+\ell_{\mathsf{id}}}, 1^{\kappa_{\mathsf{id}}})$. Set $\mathsf{MPK}_{\mathsf{id}} = (\ell_{\mathsf{id}}, \mathsf{bfe.mpk}_{\mathsf{id}})$, $\mathsf{MSK}_{\mathsf{id}} = (\ell_{\mathsf{id}}, \mathsf{bfe.msk}_{\mathsf{id}})$. Send $\mathsf{MPK}_{\mathsf{id}}$ to $\mathcal{A}$.

Secret Key Queries. $\mathcal{A}$ sends $(\mathsf{id}, \mathsf{GID}, x)$. If $\mathsf{id} \in \{\mathsf{id}_1, \ldots, \mathsf{id}_{j-1}\}$, $\mathsf{bfe.sk}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{BFE.S}_3^{F_{\mathsf{id}}(\cdot)}(\mathsf{st}_{\mathsf{id}}, (\mathsf{id}, \mathsf{GID}, x))$. Otherwise, if $\mathsf{id} = \mathsf{id}_j$, $\mathsf{bfe.sk}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathcal{O}(\mathsf{id}, \mathsf{GID}, x)$. Otherwise, $\mathsf{bfe.sk}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{BFE.KGen}(\mathsf{bfe.msk}_{\mathsf{id}}, (\mathsf{id}, \mathsf{GID}, x))$ and send $\mathsf{bfe.sk}_{\mathsf{id},\mathsf{GID},x}$ to $\mathcal{A}$.

Guess Phase. Output whatever $\mathcal{A}$ outputs.

The probability that $\mathcal{B}$ doesn't abort is at most $1/q_{\mathsf{auth}}$. In that case, the running time of $\mathcal{B}$ is polynomial in the running time of $\mathcal{A}$ and λ. If $\mathcal{O}$ is $(\mathsf{BFE.S}_0, \mathsf{BFE.S}_1, \mathsf{Enc}, \mathsf{KGen})$ for BFE, $\mathcal{B}$ behaves as $\mathbf{Hyb}_{2,j}^{\mathcal{A}}(1^\lambda)$ and if $\mathcal{O}$ is a $\mathsf{BFE.Sim}$, $\mathcal{B}$ behaves like $\mathbf{Hyb}_{2,j+1}^{\mathcal{A}}(1^\lambda)$. Hence, $\mathcal{B}$ is a valid adversary against the security of BFE that can breaks its security with non-negligible advantage, $\epsilon/q_{\mathsf{auth}}$. Thus, $\mathbf{Hyb}_{2,j}^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_{2,j+1}^{\mathcal{A}}(1^\lambda)$ are computationally indistinguishable. □

Lemma C.4. Assuming the security of tMAFE, $\mathbf{Hyb}_{2,n+1}^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_{3}^{\mathcal{A}}(1^\lambda)$ are computationally indistinguishable.

Proof. Assume that there exists a PPT adversary that can distinguish between $\mathbf{Hyb}_{2,n+1}^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_{3}^{\mathcal{A}}(1^\lambda)$ with non-negligible advantage $\epsilon(\lambda)$, i.e,

$$\left|\Pr\left[1 \leftarrow \mathbf{Hyb}_{2,n+1}^{\mathcal{A}}(1^\lambda)\right] - \Pr\left[1 \leftarrow \mathbf{Hyb}_{3}^{\mathcal{A}}(1^\lambda)\right]\right| > \epsilon(\lambda)$$

We will construct a reduction adversary $\mathcal{B}$ that can break the security of tMAFE with non-negligible probability. The description of $\mathcal{B}^{\mathcal{O}}$ is as follows.

Setup. $\mathcal{A}$ sends Q, s. Send $\mathsf{CRS} := (\lambda, Q, s)$ to $\mathcal{A}$.

Auth Setup Queries. $\mathcal{A}$ sends $(\mathsf{id}, \ell_{\mathsf{id}})$. Sample $(\mathsf{bfe.mpk}_{\mathsf{id}}, \mathsf{bfe.msk}_{\mathsf{id}}, \mathsf{bfe.st}_{\mathsf{id}}) \leftarrow \mathsf{BFE.S}_0(1^\lambda, 1^Q, 1^{\lambda+|\mathsf{GID}|+\ell_{\mathsf{id}}}, 1^{\kappa_{\mathsf{id}}})$. Set $\mathsf{MPK}_{\mathsf{id}} = (\ell_{\mathsf{id}}, \mathsf{bfe.mpk}_{\mathsf{id}})$, and $\mathsf{MSK}_{\mathsf{id}} = (\ell_{\mathsf{id}}, \mathsf{bfe.msk}_{\mathsf{id}})$. Send $\mathsf{MPK}_{\mathsf{id}}$ to $\mathcal{A}$.

Secret Key Queries. $\mathcal{A}$ sends $(\mathsf{id}, \mathsf{GID}, x)$. Sample $\mathsf{bfe.sk}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{BFE.S}_1(\mathsf{bfe.mpk}_{\mathsf{id}}, \mathsf{bfe.msk}_{\mathsf{id}}, \mathsf{bfe.st}_{\mathsf{id}}, (\mathsf{id}, \mathsf{GID}, x))$ and send $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} := \mathsf{bfe.sk}_{\mathsf{id},\mathsf{GID},x}$ to $\mathcal{A}$.

Challenge Query. $\mathcal{A}$ sends circuit $\mathcal{I}, C$. Let $n = |\mathcal{I}|$ and $\mathsf{id}_1, \ldots, \mathsf{id}_n$ be the increasing ordering of elements in $\mathcal{I}$.

- Receive $\mathsf{mafe.crs}, \{\mathsf{mafe.mpk}_{\mathsf{id}}\}_{\mathsf{id}\in[n]} \leftarrow \mathcal{O}(1^Q, 1^n, 1^s, 1^{\ell_{\mathsf{id}_1}}, \ldots, 1^{\ell_{\mathsf{id}_n}})$.
- $\forall\ i \in [n]$, $\mathsf{bfe.ct}_{\mathsf{id}_i} \leftarrow \mathsf{BFE.S}_2(\mathsf{st}_{\mathsf{id}_i}, 1^{|F_i|}, \mathcal{V}_i)$. Here $\mathcal{V}_i = \{(X, \mathcal{O}(\mathsf{id}_i, \mathsf{GID}, x_{\mathsf{GID},\mathsf{id}_i}))$: for all GID queries in previous phase$\}$.
- Sample $\mathsf{mafe.ct} \leftarrow \mathcal{O}(C)$.

Send $\mathsf{CT} := (\mathsf{mafe.ct}, \mathcal{I}, \{\mathsf{bfe.ct}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{I}})$ to $\mathcal{A}$.

Auth Setup Queries. $\mathcal{A}$ sends $(\mathsf{id}, \ell_{\mathsf{id}})$. Sample $(\mathsf{bfe.mpk}_{\mathsf{id}}, \mathsf{bfe.msk}_{\mathsf{id}}) \leftarrow \mathsf{BFE.Setup}(1^\lambda, 1^Q, 1^{\lambda+|\mathsf{GID}|+\ell_{\mathsf{id}}}, 1^{\kappa_{\mathsf{id}}})$. Set $\mathsf{MPK}_{\mathsf{id}} := (\ell_{\mathsf{id}}, \mathsf{bfe.mpk}_{\mathsf{id}})$, $\mathsf{MSK}_{\mathsf{id}} := (\ell_{\mathsf{id}}, \mathsf{bfe.msk}_{\mathsf{id}})$. Send $\mathsf{MPK}_{\mathsf{id}}$ to $\mathcal{A}$.

Secret Key Queries. $\mathcal{A}$ sends $(\mathsf{id}, \mathsf{GID}, x)$. If $\mathsf{id} \in \mathcal{I}$, sample $\mathsf{bfe.sk}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{BFE.S}_3^{F'_{\mathsf{id}}(\cdot)}(\mathsf{st}_{\mathsf{id}}, (\mathsf{id}, \mathsf{GID}, x))$. Otherwise, sample $\mathsf{bfe.sk}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{BFE.KGen}(\mathsf{bfe.msk}_{\mathsf{id}}, (\mathsf{id}, \mathsf{GID}, x))$ and send $\mathsf{bfe.sk}_{\mathsf{id},\mathsf{GID},x}$ to $\mathcal{A}$. Here $F'_{\mathsf{id}}(\cdot)$ responds with $\mathcal{O}(\mathsf{id}, \mathsf{GID}, x, \mathsf{V}_C)$.

Guess Phase. Output whatever $\mathcal{A}$ outputs.

And, the running time of $\mathcal{B}$ is polynomial in the running time of $\mathcal{A}$ and λ. If $\mathcal{O}$ is an honest challenger for tMAFE, $\mathcal{B}$ behaves as $\mathbf{Hyb}_{2,n+1}^{\mathcal{A}}(1^\lambda)$ and if $\mathcal{O}$ is a simulator, $\mathcal{B}$ behaves like $\mathbf{Hyb}_{3}^{\mathcal{A}}(1^\lambda)$. Hence, $\mathcal{B}$ is a valid adversary against the security of tMAFE that can breaks its security with non-negligible advantage. Thus, $\mathbf{Hyb}_{2,n+1}^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_{3}^{\mathcal{A}}(1^\lambda)$ are computationally indistinguishable. □

D Proofs from Section 6.1

In this section, we provide the full proof of Theorem 6.4.

$\mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)$. This is the real experiment from Definition 6.2.

Setup. $\mathcal{A}$ sends $(Q, n, s, \ell_1, \ldots, \ell_n)$. Set $\mathsf{CRS} := \mathsf{mafe.crs}$. Sample $K^{(0)}, K^{(1)} \xleftarrow{\$} \{0,1\}^\lambda$ and for each $\mathsf{id} \in [n]$, $(\mathsf{ibe.mpk_{id}}, \mathsf{ibe.msk_{id}}) \leftarrow \mathsf{IBE.Setup}(1^\lambda, 1^z)$. Set $\mathsf{MPK_{id}} := (\ell_{\mathsf{id}}, \mathsf{ibe.mpk_{id}})$, and $\mathsf{MSK_{id}} := (\ell_{\mathsf{id}}, \mathsf{ibe.msk_{id}}, K^{(0)}, K^{(1)})$. Send $(\mathsf{CRS}, \{\mathsf{MPK_{id}}\}_{\mathsf{id} \in [n]})$ to $\mathcal{A}$.

Pre-challenge Queries. $\mathcal{A}$ makes Q_1 queries of the form $(\mathsf{id}, \mathsf{GID}, x)$. Chal computes $\mathsf{Tag} = \mathsf{PRF}_0(K^{(0)}, \mathsf{GID})$, $R = \mathsf{PRF}_1(K^{(1)}, \mathsf{Tag})$, and $\mathsf{mafe.crs}, \{(\mathsf{mafe.mpk_{idx}}, \mathsf{mafe.msk_{idx}})\}_{\mathsf{idx}} \leftarrow \mathsf{tMAFE.GSetup}(1^\lambda, 1^\lambda, 1^n, 1^s, 1^{\ell_1}, \ldots, 1^{\ell_n}; R)$. For each $i \in [L_{\mathsf{id}}]$, sample $\mathsf{ibe.sk}_{\mathsf{Tag},i} \leftarrow \mathsf{IBE.KGen}(\mathsf{ibe.msk_{id}}, (\mathsf{Tag}, i, \mathsf{mafe.mpk_{id}}[i]))$. And, $\mathsf{mafe.sk}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{tMAFE.KGen}(\mathsf{id}, \mathsf{mafe.msk_{id}}, \{\mathsf{mafe.mpk_{idx}}\}_{\mathsf{idx}}, \mathsf{GID}, x)$. Send $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} := (\mathsf{Tag}, \mathsf{mafe.mpk_{id}}, \mathsf{mafe.sk}_{\mathsf{id},\mathsf{GID},x}, \{\mathsf{ibe.sk}_{\mathsf{Tag},i}\}_{i \in [L_{\mathsf{id}}]})$ to $\mathcal{A}$.

Challenge Query. $\mathcal{A}$ sends circuit C to Chal. For each $\mathsf{Tag} \in [Q]$,

- Sample $R^{\mathsf{Tag}} \leftarrow \{0,1\}^\lambda$, $(\widetilde{F}^{\mathsf{Tag}}, \{w_{\mathsf{id},i,b}\}_{\mathsf{id} \in [n], i \in [L_{\mathsf{id}}], b \in \{0,1\}}) \leftarrow \mathsf{Garble}(1^\lambda, F)$.
- For each $\mathsf{id} \in [n]$, $\mathsf{ibe.ct}^{\mathsf{Tag}}_{\mathsf{id},i,b} \leftarrow \mathsf{IBE.Enc}(\mathsf{ibe.mpk_{id}}, (\mathsf{Tag}, i, b), w^{\mathsf{Tag}}_{\mathsf{id},i,b})$.

Send $\mathsf{CT} := (\widetilde{F}^{\mathsf{Tag}}, \{\mathsf{ibe.ct}^{\mathsf{Tag}}_{\mathsf{id},i,b}\}_{\mathsf{id},i,b})_{\mathsf{Tag} \in [Q]}$ to $\mathcal{A}$.

Post-challenge Queries. $\mathcal{A}$ makes $Q - Q_1$ queries of the form $(\mathsf{id}, \mathsf{GID}, x)$. Chal computes $\mathsf{Tag} = \mathsf{PRF}_0(K^{(0)}, \mathsf{GID})$, $R = \mathsf{PRF}_1(K^{(1)}, \mathsf{Tag})$, and $\mathsf{mafe.crs}, \{(\mathsf{mafe.mpk_{idx}}, \mathsf{mafe.msk_{idx}})\}_{\mathsf{idx}} \leftarrow \mathsf{tMAFE.GSetup}(1^\lambda, 1^\lambda, 1^n, 1^s, 1^{\ell_1}, \ldots, 1^{\ell_n}; R)$. For each $i \in [L_{\mathsf{id}}]$, sample $\mathsf{ibe.sk}_{\mathsf{Tag},i} \leftarrow \mathsf{IBE.KGen}(\mathsf{ibe.msk_{id}}, (\mathsf{Tag}, i, \mathsf{mafe.mpk_{id}}[i]))$. And, $\mathsf{mafe.sk}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{tMAFE.KGen}(\mathsf{id}, \mathsf{mafe.msk_{id}}, \{\mathsf{mafe.mpk_{idx}}\}_{\mathsf{idx}}, \mathsf{GID}, x)$. Send $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} := (\mathsf{Tag}, \mathsf{mafe.mpk_{id}}, \mathsf{mafe.sk}_{\mathsf{id},\mathsf{GID},x}, \{\mathsf{ibe.sk}_{\mathsf{Tag},i}\}_{i \in [L_{\mathsf{id}}]})$ to $\mathcal{A}$.

Guess Phase. Output whatever $\mathcal{A}$ outputs.

$\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$. In this hybrid, we will sample all the Tags used in key generation early. The changes are highlighted in red.

Setup. $\mathcal{A}$ sends $(Q, n, s, \ell_1, \ldots, \ell_n)$. Set $\mathsf{CRS} := \mathsf{mafe.crs}$. Sample $K^{(0)}, K^{(1)} \xleftarrow{\$} \{0,1\}^\lambda$ and for each $\mathsf{id} \in [n]$, $(\mathsf{ibe.mpk_{id}}, \mathsf{ibe.msk_{id}}) \leftarrow \mathsf{IBE.Setup}(1^\lambda, 1^z)$. Set $\mathsf{MPK_{id}} := (\ell_{\mathsf{id}}, \mathsf{ibe.mpk_{id}})$, and $\mathsf{MSK_{id}} := (\ell_{\mathsf{id}}, \mathsf{ibe.msk_{id}}, K^{(0)}, K^{(1)})$. Send $(\mathsf{CRS}, \{\mathsf{MPK_{id}}\}_{\mathsf{id} \in [n]})$ to $\mathcal{A}$. Sample $\mathsf{Tag}_1, \ldots, \mathsf{Tag}_Q \xleftarrow{\$} [Q]$. Initiate counter $j = 1$ and dictionary $\mathcal{T}$.

Pre-challenge Queries. $\mathcal{A}$ makes Q_1 queries of the form $(\mathsf{id}, \mathsf{GID}, x)$. If $\mathsf{GID} \in \mathcal{T}$, set $\mathsf{Tag} = \mathcal{T}[\mathsf{GID}]$. Otherwise, set $\mathcal{T}[\mathsf{GID}] := \mathsf{Tag}_j$, $\mathsf{Tag} = \mathcal{T}[\mathsf{GID}]$ and increment j. Compute $R = \mathsf{PRF}_1(K^{(1)}, \mathsf{Tag})$, and $\mathsf{mafe.crs}, \{(\mathsf{mafe.mpk_{idx}}, \mathsf{mafe.msk_{idx}})\}_{\mathsf{idx}} \leftarrow \mathsf{tMAFE.GSetup}(1^\lambda, 1^\lambda, 1^n, 1^s, 1^{\ell_1}, \ldots, 1^{\ell_n}; R)$. For each $i \in [L_{\mathsf{id}}]$, sample $\mathsf{ibe.sk}_{\mathsf{Tag},i} \leftarrow \mathsf{IBE.KGen}(\mathsf{ibe.msk_{id}}, (\mathsf{Tag}, i, \mathsf{mafe.mpk_{id}}[i]))$. And, $\mathsf{mafe.sk}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{tMAFE.KGen}(\mathsf{id}, \mathsf{mafe.msk_{id}}, \{\mathsf{mafe.mpk_{idx}}\}_{\mathsf{idx}}, \mathsf{GID}, x)$. Send $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} := (\mathsf{Tag}, \mathsf{mafe.mpk_{id}}, \mathsf{mafe.sk}_{\mathsf{id},\mathsf{GID},x}, \{\mathsf{ibe.sk}_{\mathsf{Tag},i}\}_{i \in [L_{\mathsf{id}}]})$ to $\mathcal{A}$.

Challenge Query. $\mathcal{A}$ sends circuit C. For each $\mathsf{Tag} \in [Q]$,

- Sample $R^{\mathsf{Tag}} \leftarrow \{0,1\}^\lambda$, $(\widetilde{F}^{\mathsf{Tag}}, \{w_{\mathsf{id},i,b}\}_{\mathsf{id}\in[n],i\in[L_{\mathsf{id}}],b\in\{0,1\}}) \leftarrow \mathsf{Garble}(1^\lambda, F)$.
- For each $\mathsf{id} \in [n]$, $\mathsf{ibe.ct}^{\mathsf{Tag}}_{\mathsf{id},i,b} \leftarrow \mathsf{IBE.Enc}(\mathsf{ibe.mpk}_{\mathsf{id}}, (\mathsf{Tag}, i, b), w^{\mathsf{Tag}}_{\mathsf{id},i,b})$.

Send $\mathsf{CT} := (\widetilde{F}^{\mathsf{Tag}}, \{\mathsf{ibe.ct}^{\mathsf{Tag}}_{\mathsf{id},i,b}\}_{\mathsf{id},i,b})_{\mathsf{Tag}\in[Q]}$ to $\mathcal{A}$.

Post-challenge Queries. $\mathcal{A}$ makes $Q - Q_1$ queries of the form $(\mathsf{id}, \mathsf{GID}, x)$. If $\mathsf{GID} \in \mathcal{T}$, set $\mathsf{Tag} = \mathcal{T}[\mathsf{GID}]$. Otherwise, set $\mathcal{T}[\mathsf{GID}] := \mathsf{Tag}_j$, $\mathsf{Tag} = \mathcal{T}[\mathsf{GID}]$ and increment j. Compute $R = \mathsf{PRF}_1(K^{(1)}, \mathsf{Tag})$, and $\mathsf{mafe.crs}, \{(\mathsf{mafe.mpk}_{\mathsf{idx}}, \mathsf{mafe.msk}_{\mathsf{idx}})\}_{\mathsf{idx}} \leftarrow \mathsf{tMAFE.GSetup}(1^\lambda, 1^\lambda, 1^n, 1^s, 1^{\ell_1}, \ldots, 1^{\ell_n}; R)$. For each $i \in [L_{\mathsf{id}}]$, sample $\mathsf{ibe.sk}_{\mathsf{Tag},i} \leftarrow \mathsf{IBE.KGen}(\mathsf{ibe.msk}_{\mathsf{id}}, (\mathsf{Tag}, i, \mathsf{mafe.mpk}_{\mathsf{id}}[i]))$. And, $\mathsf{mafe.sk}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{tMAFE.KGen}(\mathsf{id}, \mathsf{mafe.msk}_{\mathsf{id}}, \{\mathsf{mafe.mpk}_{\mathsf{idx}}\}_{\mathsf{idx}}, \mathsf{GID}, x)$. Send $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} := (\mathsf{Tag}, \mathsf{mafe.mpk}_{\mathsf{id}}, \mathsf{mafe.sk}_{\mathsf{id},\mathsf{GID},x}, \{\mathsf{ibe.sk}_{\mathsf{Tag},i}\}_{i\in[L_{\mathsf{id}}]})$ to $\mathcal{A}$.

Guess Phase. Output whatever $\mathcal{A}$ outputs.

$\mathbf{Hyb}_2^{\mathcal{A}}(1^\lambda)$. In this hybrid, we check if any $\mathsf{Tag} \in [Q]$ was sampled more that λ many times in previous hybrid. The changes are highlighted in red.

Setup. $\mathcal{A}$ sends $(Q, n, s, \ell_1, \ldots, \ell_n)$. Set $\mathsf{CRS} := \mathsf{mafe.crs}$. Sample $K^{(1)} \xleftarrow{\$} \{0,1\}^\lambda$ and for each $\mathsf{id} \in [n]$, $(\mathsf{ibe.mpk}_{\mathsf{id}}, \mathsf{ibe.msk}_{\mathsf{id}}) \leftarrow \mathsf{IBE.Setup}(1^\lambda, 1^z)$. Set $\mathsf{MPK}_{\mathsf{id}} := (\ell_{\mathsf{id}}, \mathsf{ibe.mpk}_{\mathsf{id}})$, and $\mathsf{MSK}_{\mathsf{id}} := (\ell_{\mathsf{id}}, \mathsf{ibe.msk}_{\mathsf{id}}, K^{(1)})$. Send $(\mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]})$ to $\mathcal{A}$. Sample $\mathsf{Tag}_1, \ldots, \mathsf{Tag}_Q \xleftarrow{\$} [Q]$. Initiate counter $j = 1$ and dictionary $\mathcal{T}$. If there exists a $\mathsf{Tag} \in [Q]$ and indices $i_1^*, \ldots, i_q^*$ such that $q > \lambda$ and $\mathsf{Tag}_{i_1^*}, \ldots, \mathsf{Tag}_{i_q^*}$ are same as Tag, then abort and output $\perp$.

Pre-challenge, Challenge, Post-challenge Queries, Guess Phase. Same as $\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$.

$\mathbf{Hyb}_3^{\mathcal{A}}(1^\lambda)$. In this hybrid, we will sample random strings R and thus random tMAFE instantiations instead of using PRF_1. The changes are highlighted in red.

Setup. $\mathcal{A}$ sends $(Q, n, s, \ell_1, \ldots, \ell_n)$. Set $\mathsf{CRS} := \mathsf{mafe.crs}$. Sample $K^{(1)} \xleftarrow{\$} \{0,1\}^\lambda$ and for each $\mathsf{id} \in [n]$, $(\mathsf{ibe.mpk}_{\mathsf{id}}, \mathsf{ibe.msk}_{\mathsf{id}}) \leftarrow \mathsf{IBE.Setup}(1^\lambda, 1^z)$. Set $\mathsf{MPK}_{\mathsf{id}} := (\ell_{\mathsf{id}}, \mathsf{ibe.mpk}_{\mathsf{id}})$, and $\mathsf{MSK}_{\mathsf{id}} := (\ell_{\mathsf{id}}, \mathsf{ibe.msk}_{\mathsf{id}}, K^{(1)})$. Send $(\mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]})$ to $\mathcal{A}$. Sample $\mathsf{Tag}_1, \ldots, \mathsf{Tag}_Q \xleftarrow{\$} [Q]$. Initiate counter $j = 1$ and dictionary $\mathcal{T}$. If there exists a $\mathsf{Tag} \in [Q]$ and indices $i_1^*, \ldots, i_q^*$ such that $q > \lambda$ and $\mathsf{Tag}_{i_1^*}, \ldots, \mathsf{Tag}_{i_q^*}$ are same as Tag, then abort and output $\perp$. Sample $R^{(1)}, \ldots, R^{(Q)} \xleftarrow{\$} \{0,1\}^\lambda$.

Pre-challenge Queries. $\mathcal{A}$ makes Q_1 queries of the form $(\mathsf{id}, \mathsf{GID}, x)$. If $\mathsf{GID} \in \mathcal{T}$, set $\mathsf{Tag} = \mathcal{T}[\mathsf{GID}]$. Otherwise, set $\mathcal{T}[\mathsf{GID}] := \mathsf{Tag}_j$, $\mathsf{Tag} = \mathcal{T}[\mathsf{GID}]$. Increment j. Sample $\mathsf{mafe.crs}, \{(\mathsf{mafe.mpk}_{\mathsf{idx}}, \mathsf{mafe.msk}_{\mathsf{idx}})\}_{\mathsf{idx}} \leftarrow \mathsf{tMAFE.GSetup}(1^\lambda, 1^\lambda, 1^n, 1^s, 1^{\ell_1}, \ldots, 1^{\ell_n}; R^{(\mathsf{Tag})})$. For each $i \in [L_{\mathsf{id}}]$, sample $\mathsf{ibe.sk}_{\mathsf{Tag},i} \leftarrow \mathsf{IBE.KGen}(\mathsf{ibe.msk}_{\mathsf{id}}, (\mathsf{Tag}, i, \mathsf{mafe.mpk}_{\mathsf{id}}[i]))$. And, $\mathsf{mafe.sk}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{tMAFE.KGen}(\mathsf{id}, \mathsf{mafe.msk}_{\mathsf{id}}, \{\mathsf{mafe.mpk}_{\mathsf{idx}}\}_{\mathsf{idx}}, \mathsf{GID}, x)$. Send $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} := (\mathsf{Tag}, \mathsf{mafe.mpk}_{\mathsf{id}}, \mathsf{mafe.sk}_{\mathsf{id},\mathsf{GID},x}, \{\mathsf{ibe.sk}_{\mathsf{Tag},i}\}_{i\in[L_{\mathsf{id}}]})$ to $\mathcal{A}$.

Challenge Query. $\mathcal{A}$ sends circuit C. For each $\mathsf{Tag} \in [Q]$,

- Sample $R^{\mathsf{Tag}} \leftarrow \{0,1\}^\lambda$, $(\widetilde{F}^{\mathsf{Tag}}, \{w_{\mathsf{id},i,b}\}_{\mathsf{id}\in[n],i\in[L_{\mathsf{id}}],b\in\{0,1\}}) \leftarrow \mathsf{Garble}(1^\lambda, F)$.
- For each $\mathsf{id} \in [n]$, $\mathsf{ibe.ct}^{\mathsf{Tag}}_{\mathsf{id},i,b} \leftarrow \mathsf{IBE.Enc}(\mathsf{ibe.mpk}_{\mathsf{id}}, (\mathsf{Tag}, i, b), w^{\mathsf{Tag}}_{\mathsf{id},i,b})$.

Send $\mathsf{CT} := (\widetilde{F}^{\mathsf{Tag}}, \{\mathsf{ibe.ct}^{\mathsf{Tag}}_{\mathsf{id},i,b}\}_{\mathsf{id},i,b})_{\mathsf{Tag}\in[Q]}$ to $\mathcal{A}$.

Post-challenge Queries. $\mathcal{A}$ makes $Q - Q_1$ queries of the form $(\mathsf{id}, \mathsf{GID}, x)$. If $\mathsf{GID} \in \mathcal{T}$, set $\mathsf{Tag} = \mathcal{T}[\mathsf{GID}]$. Otherwise, set $\mathcal{T}[\mathsf{GID}] := \mathsf{Tag}_j, \mathsf{Tag} = \mathcal{T}[\mathsf{GID}]$ and increment j. Sample $\mathsf{mafe.crs}, \{(\mathsf{mafe.mpk}_{\mathsf{idx}}, \mathsf{mafe.msk}_{\mathsf{idx}})\}_{\mathsf{idx}} \leftarrow \mathsf{tMAFE.GSetup}(1^\lambda, 1^\lambda, 1^n, 1^s, 1^{\ell_1}, \ldots, 1^{\ell_n}; R^{(\mathsf{Tag})})$. For each $i \in [L_{\mathsf{id}}]$, sample $\mathsf{ibe.sk}_{\mathsf{Tag},i} \leftarrow \mathsf{IBE.KGen}(\mathsf{ibe.msk}_{\mathsf{id}}, (\mathsf{Tag}, i, \mathsf{mafe.mpk}_{\mathsf{id}}[i]))$. And, $\mathsf{mafe.sk}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{tMAFE.KGen}(\mathsf{id}, \mathsf{mafe.msk}_{\mathsf{id}}, \{\mathsf{mafe.mpk}_{\mathsf{idx}}\}_{\mathsf{idx}}, \mathsf{GID}, x)$. Send $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} := (\mathsf{Tag}, \mathsf{mafe.mpk}_{\mathsf{id}}, \mathsf{mafe.sk}_{\mathsf{id},\mathsf{GID},x}, \{\mathsf{ibe.sk}_{\mathsf{Tag},i}\}_{i\in[L_{\mathsf{id}}]})$ to $\mathcal{A}$.

Guess Phase. Output whatever $\mathcal{A}$ outputs.

$\mathbf{Hyb}_4^{\mathcal{A}}(1^\lambda)$. In this hybrid, we will all Q many tMAFE instantiations early and use the appropriate keys to generate secret keys. The changes are highlighted in red.

Setup. $\mathcal{A}$ sends $(Q, n, s, \ell_1, \ldots, \ell_n)$. Set $\mathsf{CRS} := \mathsf{mafe.crs}$. Sample $K^{(1)} \xleftarrow{\$} \{0,1\}^\lambda$ and for each $\mathsf{id} \in [n]$, $(\mathsf{ibe.mpk}_{\mathsf{id}}, \mathsf{ibe.msk}_{\mathsf{id}}) \leftarrow \mathsf{IBE.Setup}(1^\lambda, 1^z)$. Set $\mathsf{MPK}_{\mathsf{id}} := (\ell_{\mathsf{id}}, \mathsf{ibe.mpk}_{\mathsf{id}})$, and $\mathsf{MSK}_{\mathsf{id}} := (\ell_{\mathsf{id}}, \mathsf{ibe.msk}_{\mathsf{id}}, K^{(1)})$. Send $(\mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]})$ to $\mathcal{A}$. Sample $\mathsf{Tag}_1, \ldots, \mathsf{Tag}_Q \xleftarrow{\$} [Q]$. Initiate counter $j = 1$ and dictionary $\mathcal{T}$. If there exists a $\mathsf{Tag} \in [Q]$ and indices $i_1^*, \ldots, i_q^*$ such that $q > \lambda$ and $\mathsf{Tag}_{i_1^*}, \ldots, \mathsf{Tag}_{i_q^*}$ are same as Tag, then abort and output $\perp$. For each $\mathsf{Tag} \in [Q]$, $\mathsf{mafe.crs}^{\mathsf{Tag}}, \{(\mathsf{mafe.mpk}^{\mathsf{Tag}}_{\mathsf{idx}}, \mathsf{mafe.msk}^{\mathsf{Tag}}_{\mathsf{idx}})\}_{\mathsf{idx}} \leftarrow \mathsf{tMAFE.GSetup}(1^\lambda, 1^\lambda, 1^n, 1^s, 1^{\ell_1}, \ldots, 1^{\ell_n})$.

Pre-challenge Queries. $\mathcal{A}$ makes Q_1 queries of the form $(\mathsf{id}, \mathsf{GID}, x)$. If $\mathsf{GID} \in \mathcal{T}$, set $\mathsf{Tag} = \mathcal{T}[\mathsf{GID}]$. Otherwise, set $\mathcal{T}[\mathsf{GID}] := \mathsf{Tag}_j, \mathsf{Tag} = \mathcal{T}[\mathsf{GID}]$ and increment j. For each $i \in [L_{\mathsf{id}}]$, sample $\mathsf{ibe.sk}_{\mathsf{Tag},i} \leftarrow \mathsf{IBE.KGen}(\mathsf{ibe.msk}_{\mathsf{id}}, (\mathsf{Tag}, i, \mathsf{mafe.mpk}^{\mathsf{Tag}}_{\mathsf{id}}[i]))$. And, $\mathsf{mafe.sk}^{\mathsf{Tag}}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{tMAFE.KGen}(\mathsf{id}, \mathsf{mafe.msk}^{\mathsf{Tag}}_{\mathsf{id}},$
$\{\mathsf{mafe.mpk}^{\mathsf{Tag}}_{\mathsf{idx}}\}_{\mathsf{idx}}, \mathsf{GID}, x)$. Send $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} := (\mathsf{Tag}, \mathsf{mafe.mpk}^{\mathsf{Tag}}_{\mathsf{id}}, \mathsf{mafe.sk}^{\mathsf{Tag}}_{\mathsf{id},\mathsf{GID},x}, \{\mathsf{ibe.sk}_{\mathsf{Tag},i}\}_i)$ to $\mathcal{A}$.

Challenge Query. $\mathcal{A}$ sends circuit C. For each $\mathsf{Tag} \in [Q]$,

- Sample $R^{\mathsf{Tag}} \leftarrow \{0,1\}^\lambda$, $(\widetilde{F}^{\mathsf{Tag}}, \{w_{\mathsf{id},i,b}\}_{\mathsf{id}\in[n],i\in[L_{\mathsf{id}}],b\in\{0,1\}}) \leftarrow \mathsf{Garble}(1^\lambda, F)$.
- For each $\mathsf{id} \in [n]$, $\mathsf{ibe.ct}^{\mathsf{Tag}}_{\mathsf{id},i,b} \leftarrow \mathsf{IBE.Enc}(\mathsf{ibe.mpk}_{\mathsf{id}}, (\mathsf{Tag}, i, b), w^{\mathsf{Tag}}_{\mathsf{id},i,b})$.

Send $\mathsf{CT} := (\widetilde{F}^{\mathsf{Tag}}, \{\mathsf{ibe.ct}^{\mathsf{Tag}}_{\mathsf{id},i,b}\}_{\mathsf{id},i,b})_{\mathsf{Tag}\in[Q]}$ to $\mathcal{A}$.

Post-challenge Queries. $\mathcal{A}$ makes $Q - Q_1$ queries of the form $(\mathsf{id}, \mathsf{GID}, x)$. If $\mathsf{GID} \in \mathcal{T}$, set $\mathsf{Tag} = \mathcal{T}[\mathsf{GID}]$. Otherwise, set $\mathcal{T}[\mathsf{GID}] := \mathsf{Tag}_j, \mathsf{Tag} = \mathcal{T}[\mathsf{GID}]$ and increment j. For each $i \in [L_{\mathsf{id}}]$, sample $\mathsf{ibe.sk}_{\mathsf{Tag},i} \leftarrow \mathsf{IBE.KGen}(\mathsf{ibe.msk}_{\mathsf{id}}, (\mathsf{Tag}, i, \mathsf{mafe.mpk}^{\mathsf{Tag}}_{\mathsf{id}}[i]))$. And, $\mathsf{mafe.sk}^{\mathsf{Tag}}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{tMAFE.KGen}(\mathsf{id}, \mathsf{mafe.msk}^{\mathsf{Tag}}_{\mathsf{id}}, \{\mathsf{mafe.mpk}^{\mathsf{Tag}}_{\mathsf{idx}}\}_{\mathsf{idx}}, \mathsf{GID}, x)$. Send $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} := (\mathsf{Tag}, \mathsf{mafe.mpk}^{\mathsf{Tag}}_{\mathsf{id}}, \mathsf{mafe.sk}^{\mathsf{Tag}}_{\mathsf{id},\mathsf{GID},x}, \{\mathsf{ibe.sk}_{\mathsf{Tag},i}\}_{i\in[L_{\mathsf{id}}]})$ to $\mathcal{A}$.

Guess Phase. Output whatever $\mathcal{A}$ outputs.

$\mathbf{Hyb}^{\mathcal{A}}_{5,k}(1^\lambda)$. for $k \in [n+1]$. In this hybrid, we will set half of the IBE encryptions to all zero strings of appropriate length for the first $k-1$ authorities. The changes are highlighted in red.

Setup, Pre-challenge Queries. Same as $\mathbf{Hyb}^{\mathcal{A}}_{4}(1^\lambda)$.

Challenge Query. $\mathcal{A}$ sends circuit C. For each $\mathsf{Tag} \in [Q]$,

- Sample $R^{\mathsf{Tag}} \leftarrow \{0,1\}^\lambda$, $(\widetilde{F}^{\mathsf{Tag}}, \{w_{\mathsf{id},i,b}\}_{\mathsf{id}\in[n],i\in[L_{\mathsf{id}}],b\in\{0,1\}}) \leftarrow \mathsf{Garble}(1^\lambda, F)$.
- For each $\mathsf{id} \in [n]$,
  - If $\mathsf{id} < k, i \in [L_{\mathsf{id}}]$, let $\beta = \mathsf{mafe.mpk}^{\mathsf{Tag}}_{\mathsf{id}}[i]$, $\mathsf{ibe.ct}^{\mathsf{Tag}}_{\mathsf{id},i,\beta} \leftarrow \mathsf{IBE.Enc}(\mathsf{ibe.mpk}_{\mathsf{id}}, (\mathsf{Tag}, i, \beta), w^{\mathsf{Tag}}_{\mathsf{id},i,\beta})$ and $\mathsf{ibe.ct}^{\mathsf{Tag}}_{\mathsf{id},i,1-\beta} \leftarrow \mathsf{IBE.Enc}(\mathsf{ibe.mpk}_{\mathsf{id}}, (\mathsf{Tag}, i, 1-\beta), 0^{|w^{\mathsf{Tag}}_{\mathsf{id},i,1-\beta}|})$.
  - Otherwise, for $i \in [L_{\mathsf{id}}], b \in \{0,1\}$, $\mathsf{ibe.ct}^{\mathsf{Tag}}_{\mathsf{id},i,b} \leftarrow \mathsf{IBE.Enc}(\mathsf{ibe.mpk}_{\mathsf{id}}, (\mathsf{Tag}, i, b), w^{\mathsf{Tag}}_{\mathsf{id},i,b})$

Send $\mathsf{CT} := (\widetilde{F}^{\mathsf{Tag}}, \{\mathsf{ibe.ct}^{\mathsf{Tag}}_{\mathsf{id},i,b}\}_{\mathsf{id},i,b})_{\mathsf{Tag}\in[Q]}$ to $\mathcal{A}$.

Post-challenge Queries, Guess Phase. Same as $\mathbf{Hyb}^{\mathcal{A}}_{4}(1^\lambda)$.

$\mathbf{Hyb}^{\mathcal{A}}_{6,t}(1^\lambda)$. for $t \in [Q+1]$. In this hybrid, we will simulate the garbling of F for each $\mathsf{Tag} < t$. The changes are highlighted in red.

Setup, Pre-challenge Queries. Same as $\mathbf{Hyb}^{\mathcal{A}}_{5,n+1}(1^\lambda)$.

Challenge Query. $\mathcal{A}$ sends circuit C. For each $\mathsf{Tag} \in [Q]$,

- If $\mathsf{Tag} < t$ $(\widetilde{F}^{\mathsf{Tag}}, \{w_{\mathsf{id},i,\mathsf{mafe.mpk}^{\mathsf{Tag}}_{\mathsf{id}}[i]}\}_{\mathsf{id},i}) \leftarrow \mathsf{Garb.Sim}(1^\lambda, 1^L, 1^{|F|}, \mathsf{mafe.ct}^{\mathsf{Tag}})$ where $L = L_1 + \ldots + L_n$ and $\mathsf{mafe.ct}^{\mathsf{Tag}} \leftarrow \mathsf{tMAFE.Enc}(\{\mathsf{mafe.mpk}^{\mathsf{Tag}}_{\mathsf{idx}}\}_{\mathsf{idx}\in[n]}, C)$.
- Otherwise, $R^{\mathsf{Tag}} \leftarrow \{0,1\}^\lambda$, $(\widetilde{F}^{\mathsf{Tag}}, \{w_{\mathsf{id},i,b}\}_{\mathsf{id}\in[n],i\in[L_{\mathsf{id}}],b\in\{0,1\}}) \leftarrow \mathsf{Garble}(1^\lambda, F)$.
- $\forall$ $\mathsf{id} \in [n]$, $i \in [L_{\mathsf{id}}]$, let $\beta = \mathsf{mafe.mpk}^{\mathsf{Tag}}_{\mathsf{id}}[i]$, $\mathsf{ibe.ct}^{\mathsf{Tag}}_{\mathsf{id},i,\beta} \leftarrow \mathsf{IBE.Enc}(\mathsf{ibe.mpk}_{\mathsf{id}}, (\mathsf{Tag}, i, \beta), w^{\mathsf{Tag}}_{\mathsf{id},i,\beta})$ and $\mathsf{ibe.ct}^{\mathsf{Tag}}_{\mathsf{id},i,1-\beta} \leftarrow \mathsf{IBE.Enc}(\mathsf{ibe.mpk}_{\mathsf{id}}, (\mathsf{Tag}, i, 1-\beta), 0^{|w^{\mathsf{Tag}}_{\mathsf{id},i,1-\beta}|})$.

Send $\mathsf{CT} := (\widetilde{F}^{\mathsf{Tag}}, \{\mathsf{ibe.ct}^{\mathsf{Tag}}_{\mathsf{id},i,b}\}_{\mathsf{id},i,b})_{\mathsf{Tag}\in[Q]}$ to $\mathcal{A}$.

Post-challenge Queries, Guess Phase. Same as $\mathbf{Hyb}^{\mathcal{A}}_{5,n+1}(1^\lambda)$.

$\mathbf{Hyb}^{\mathcal{A}}_{7,t}(1^\lambda)$. for $t \in [Q+1]$. In this hybrid, we will simulate the Tag-th tMAFE instantiations for $\mathsf{Tag} < t$. The changes are highlighted in red. Here, $\mathcal{V}^{\mathsf{Tag}}$ is the set of all input and outputs relations for C as defined in Definition 4.1 except we only consider the queries where Tag is used.

Setup. $\mathcal{A}$ sends $(Q, n, s, \ell_1, \ldots, \ell_n)$. Set $\mathsf{CRS} := \mathsf{mafe.crs}$. Sample $K^{(1)} \xleftarrow{\$} \{0,1\}^\lambda$ and for each $\mathsf{id} \in [n]$, $(\mathsf{ibe.mpk}_{\mathsf{id}}, \mathsf{ibe.msk}_{\mathsf{id}}) \leftarrow \mathsf{IBE.Setup}(1^\lambda, 1^z)$. Set $\mathsf{MPK}_{\mathsf{id}} := (\ell_{\mathsf{id}}, \mathsf{ibe.mpk}_{\mathsf{id}})$, and $\mathsf{MSK}_{\mathsf{id}} := (\ell_{\mathsf{id}}, \mathsf{ibe.msk}_{\mathsf{id}}, K^{(1)})$. Send $(\mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]})$ to $\mathcal{A}$. Sample $\mathsf{Tag}_1, \ldots, \mathsf{Tag}_Q \xleftarrow{\$} [Q]$. Initiate counter $j = 1$ and dictionary $\mathcal{T}$. If there exists a $\mathsf{Tag} \in [Q]$ and indices $i^*_1, \ldots, i^*_q$ such that $q > \lambda$ and $\mathsf{Tag}_{i^*_1}, \ldots, \mathsf{Tag}_{i^*_q}$ are same as Tag, then abort and output $\perp$. For each $\mathsf{Tag} \in [Q]$, if $\mathsf{Tag} < t$, $\mathsf{mafe.crs}^{\mathsf{Tag}}, \{\mathsf{mafe.mpk}^{\mathsf{Tag}}_{\mathsf{idx}}\}_{\mathsf{idx}} \leftarrow \mathsf{tMAFE.Sim}_{\mathsf{Tag}}(1^\lambda, 1^\lambda, 1^n, 1^s, 1^{\ell_1}, \ldots, 1^{\ell_n})$. Otherwise, $\mathsf{mafe.crs}^{\mathsf{Tag}}, \{(\mathsf{mafe.mpk}^{\mathsf{Tag}}_{\mathsf{idx}}, \mathsf{mafe.msk}^{\mathsf{Tag}}_{\mathsf{idx}})\}_{\mathsf{idx}} \leftarrow \mathsf{tMAFE.GSetup}(1^\lambda, 1^\lambda, 1^n, 1^s, 1^{\ell_1}, \ldots, 1^{\ell_n})$.

Pre-challenge Queries. $\mathcal{A}$ makes Q_1 queries of the form $(\mathsf{id}, \mathsf{GID}, x)$. If $\mathsf{GID} \in \mathcal{T}$, set $\mathsf{Tag} = \mathcal{T}[\mathsf{GID}]$. Otherwise, set $\mathcal{T}[\mathsf{GID}] := \mathsf{Tag}_j, \mathsf{Tag} = \mathcal{T}[\mathsf{GID}]$ and increment j. For each $i \in [L_{\mathsf{id}}]$, sample $\mathsf{ibe.sk}_{\mathsf{Tag},i} \leftarrow \mathsf{IBE.KGen}(\mathsf{ibe.msk}_{\mathsf{id}}, (\mathsf{Tag}, i, \mathsf{mafe.mpk}_{\mathsf{id}}^{\mathsf{Tag}}[i]))$. And, if $\mathsf{Tag} < t$, $\mathsf{mafe.sk}_{\mathsf{id},\mathsf{GID},x}^{\mathsf{Tag}} \leftarrow \mathsf{tMAFE.Sim}_{\mathsf{Tag}}(\mathsf{id}, \mathsf{GID}, x)$. Otherwise, $\mathsf{mafe.sk}_{\mathsf{id},\mathsf{GID},x}^{\mathsf{Tag}} \leftarrow \mathsf{tMAFE.KGen}(\mathsf{id}, \mathsf{mafe.msk}_{\mathsf{id}}^{\mathsf{Tag}}, \{\mathsf{mafe.mpk}_{\mathsf{idx}}^{\mathsf{Tag}}\}_{\mathsf{idx}}, \mathsf{GID}, x)$. Send $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} := (\mathsf{Tag}, \mathsf{mafe.mpk}_{\mathsf{id}}^{\mathsf{Tag}}, \mathsf{mafe.sk}_{\mathsf{id},\mathsf{GID},x}^{\mathsf{Tag}}, \{\mathsf{ibe.sk}_{\mathsf{Tag},i}\}_{i\in[L_{\mathsf{id}}]})$ to $\mathcal{A}$.

Challenge Query. $\mathcal{A}$ sends circuit C. For each $\mathsf{Tag} \in [Q]$,

- $(\widetilde{F}^{\mathsf{Tag}}, \{w_{\mathsf{id},i,\mathsf{mafe.mpk}_{\mathsf{id}}^{\mathsf{Tag}}[i]}\}_{\mathsf{id},i}) \leftarrow \mathsf{Garb.Sim}(1^\lambda, 1^L, 1^{|F|}, \mathsf{mafe.ct}^{\mathsf{Tag}})$ where $L = L_1 + \ldots + L_n$ and if $\mathsf{Tag} < t$, $\mathsf{mafe.ct}^{\mathsf{Tag}} \leftarrow \mathsf{tMAFE.Sim}(1^{|C|}, \mathcal{V}^{\mathsf{Tag}})$. Otherwise, $\mathsf{mafe.ct}^{\mathsf{Tag}} \leftarrow \mathsf{tMAFE.Enc}(\{\mathsf{mafe.mpk}_{\mathsf{idx}}^{\mathsf{Tag}}\}_{\mathsf{idx}\in[n]}, C)$.
- $\forall$ $\mathsf{id} \in [n]$, $i \in [L_{\mathsf{id}}]$, let $\beta = \mathsf{mafe.mpk}_{\mathsf{id}}^{\mathsf{Tag}}[i]$, $\mathsf{ibe.ct}_{\mathsf{id},i,\beta}^{\mathsf{Tag}} \leftarrow \mathsf{IBE.Enc}(\mathsf{ibe.mpk}_{\mathsf{id}}, (\mathsf{Tag}, i, \beta), w_{\mathsf{id},i,\beta}^{\mathsf{Tag}})$ and $\mathsf{ibe.ct}_{\mathsf{id},i,1-\beta}^{\mathsf{Tag}} \leftarrow \mathsf{IBE.Enc}(\mathsf{ibe.mpk}_{\mathsf{id}}, (\mathsf{Tag}, i, 1-\beta), 0^{\left|w_{\mathsf{id},i,1-\beta}^{\mathsf{Tag}}\right|})$.

Send $\mathsf{CT} := (\widetilde{F}^{\mathsf{Tag}}, \{\mathsf{ibe.ct}_{\mathsf{id},i,b}^{\mathsf{Tag}}\}_{\mathsf{id},i,b})_{\mathsf{Tag}\in[Q]}$ to $\mathcal{A}$.

Post-challenge Queries. $\mathcal{A}$ makes $Q - Q_1$ queries of the form $(\mathsf{id}, \mathsf{GID}, x)$. If $\mathsf{GID} \in \mathcal{T}$, set $\mathsf{Tag} = \mathcal{T}[\mathsf{GID}]$. Otherwise, set $\mathcal{T}[\mathsf{GID}] := \mathsf{Tag}_j, \mathsf{Tag} = \mathcal{T}[\mathsf{GID}]$ and increment j. For each $i \in [L_{\mathsf{id}}]$, sample $\mathsf{ibe.sk}_{\mathsf{Tag},i} \leftarrow \mathsf{IBE.KGen}(\mathsf{ibe.msk}_{\mathsf{id}}, (\mathsf{Tag}, i, \mathsf{mafe.mpk}_{\mathsf{id}}^{\mathsf{Tag}}[i]))$. And, if $\mathsf{Tag} < t$, $\mathsf{mafe.sk}_{\mathsf{id},\mathsf{GID},x}^{\mathsf{Tag}} \leftarrow \mathsf{tMAFE.Sim}_{\mathsf{Tag}}^{C(\cdot)}(\mathsf{id}, \mathsf{GID}, x)$. Otherwise, $\mathsf{mafe.sk}_{\mathsf{id},\mathsf{GID},x}^{\mathsf{Tag}} \leftarrow \mathsf{tMAFE.KGen}(\mathsf{id}, \mathsf{mafe.msk}_{\mathsf{id}}^{\mathsf{Tag}}, \{\mathsf{mafe.mpk}_{\mathsf{idx}}^{\mathsf{Tag}}\}_{\mathsf{idx}}, \mathsf{GID}, x)$. Send $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} := (\mathsf{Tag}, \mathsf{mafe.mpk}_{\mathsf{id}}^{\mathsf{Tag}}, \mathsf{mafe.sk}_{\mathsf{id},\mathsf{GID},x}^{\mathsf{Tag}}, \{\mathsf{ibe.sk}_{\mathsf{Tag},i}\}_{i\in[L_{\mathsf{id}}]})$ to $\mathcal{A}$.

Guess Phase. Output whatever $\mathcal{A}$ outputs.

Lemma D.1. Assuming the pseudorandomness of PRF_0, $\mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$ are computationally indistinguishable.

Proof. Assume that there exists a PPT adversary that can distinguish between $\mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$ with non-negligible advantage $\epsilon(\lambda)$, i.e,

$$\left|\Pr\left[1 \leftarrow \mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)\right] - \Pr\left[1 \leftarrow \mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)\right]\right| > \epsilon(\lambda)$$

We will construct a reduction adversary $\mathcal{B}$ that can break the pseudorandomness of PRF_0 with non-negligible probability. The description of $\mathcal{B}^{\mathcal{O}}$ is as follows.

Setup. $\mathcal{A}$ sends $(Q, n, s, \ell_1, \ldots, \ell_n)$. Set $\mathsf{CRS} := \mathsf{mafe.crs}$. Sample $K^{(0)}, K^{(1)} \xleftarrow{\$} \{0,1\}^\lambda$ and for each $\mathsf{id} \in [n]$, $(\mathsf{ibe.mpk}_{\mathsf{id}}, \mathsf{ibe.msk}_{\mathsf{id}}) \leftarrow \mathsf{IBE.Setup}(1^\lambda, 1^z)$. Set $\mathsf{MPK}_{\mathsf{id}} := (\ell_{\mathsf{id}}, \mathsf{ibe.mpk}_{\mathsf{id}})$, and $\mathsf{MSK}_{\mathsf{id}} := (\ell_{\mathsf{id}}, \mathsf{ibe.msk}_{\mathsf{id}}, K^{(0)}, K^{(1)})$. Send $(\mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]})$ to $\mathcal{A}$.

Pre-challenge Queries. $\mathcal{A}$ makes Q_1 queries of the form $(\mathsf{id}, \mathsf{GID}, x)$. Let $\mathsf{Tag} \leftarrow \mathcal{O}(\mathsf{GID})$. Compute $R = \mathsf{PRF}_1(K^{(1)}, \mathsf{Tag})$, and $\mathsf{mafe.crs}, \{(\mathsf{mafe.mpk}_{\mathsf{idx}}, \mathsf{mafe.msk}_{\mathsf{idx}})\}_{\mathsf{idx}} \leftarrow \mathsf{tMAFE.GSetup}(1^\lambda, 1^\lambda, 1^n, 1^s, 1^{\ell_1}, \ldots, 1^{\ell_n}; R)$. For each $i \in [L_{\mathsf{id}}]$, sample $\mathsf{ibe.sk}_{\mathsf{Tag},i} \leftarrow \mathsf{IBE.KGen}(\mathsf{ibe.msk}_{\mathsf{id}}, (\mathsf{Tag}, i, \mathsf{mafe.mpk}_{\mathsf{id}}[i]))$. And, $\mathsf{mafe.sk}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{tMAFE.KGen}(\mathsf{id}, \mathsf{mafe.msk}_{\mathsf{id}}, \{\mathsf{mafe.mpk}_{\mathsf{idx}}\}_{\mathsf{idx}}, \mathsf{GID}, x)$. Send $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} := (\mathsf{Tag}, \mathsf{mafe.mpk}_{\mathsf{id}}, \mathsf{mafe.sk}_{\mathsf{id},\mathsf{GID},x}, \{\mathsf{ibe.sk}_{\mathsf{Tag},i}\}_{i\in[L_{\mathsf{id}}]})$ to $\mathcal{A}$.

Challenge Query. $\mathcal{A}$ sends circuit C. For each $\mathsf{Tag} \in [Q]$,

- Sample $R^{\mathsf{Tag}} \leftarrow \{0,1\}^\lambda$, $(\widetilde{F}^{\mathsf{Tag}}, \{w_{\mathsf{id},i,b}\}_{\mathsf{id}\in[n],i\in[L_{\mathsf{id}}],b\in\{0,1\}}) \leftarrow \mathsf{Garble}(1^\lambda, F)$.
- For each $\mathsf{id} \in [n]$, $\mathsf{ibe.ct}^{\mathsf{Tag}}_{\mathsf{id},i,b} \leftarrow \mathsf{IBE.Enc}(\mathsf{ibe.mpk}_{\mathsf{id}}, (\mathsf{Tag}, i, b), w^{\mathsf{Tag}}_{\mathsf{id},i,b})$.

Send $\mathsf{CT} := (\widetilde{F}^{\mathsf{Tag}}, \{\mathsf{ibe.ct}^{\mathsf{Tag}}_{\mathsf{id},i,b}\}_{\mathsf{id},i,b})_{\mathsf{Tag}\in[Q]}$ to $\mathcal{A}$.

Post-challenge Queries. $\mathcal{A}$ makes $Q - Q_1$ queries of the form $(\mathsf{id}, \mathsf{GID}, x)$. Let $\mathsf{Tag} \leftarrow \mathcal{O}(\mathsf{GID})$. Compute $R = \mathsf{PRF}_1(K^{(1)}, \mathsf{Tag})$, and $\mathsf{mafe.crs}, \{(\mathsf{mafe.mpk}_{\mathsf{idx}}, \mathsf{mafe.msk}_{\mathsf{idx}})\}_{\mathsf{idx}} \leftarrow \mathsf{tMAFE.GSetup}$ $(1^\lambda, 1^\lambda, 1^n, 1^s, 1^{\ell_1}, \ldots, 1^{\ell_n}; R)$. For each $i \in [L_{\mathsf{id}}]$, sample $\mathsf{ibe.sk}_{\mathsf{Tag},i} \leftarrow \mathsf{IBE.KGen}(\mathsf{ibe.msk}_{\mathsf{id}}, (\mathsf{Tag}, i, \mathsf{mafe.mpk}_{\mathsf{id}}[i]))$. And, $\mathsf{mafe.sk}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{tMAFE.KGen}(\mathsf{id}, \mathsf{mafe.msk}_{\mathsf{id}}, \{\mathsf{mafe.mpk}_{\mathsf{idx}}\}_{\mathsf{idx}}, \mathsf{GID}, x)$. Send $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} := (\mathsf{Tag}, \mathsf{mafe.mpk}_{\mathsf{id}}, \mathsf{mafe.sk}_{\mathsf{id},\mathsf{GID},x}, \{\mathsf{ibe.sk}_{\mathsf{Tag},i}\}_{i\in[L_{\mathsf{id}}]})$ to $\mathcal{A}$.

Guess Phase. Output whatever $\mathcal{A}$ outputs.

As we can see, the running time of $\mathcal{B}$ is polynomial in the running time of $\mathcal{A}$ and λ. If $\mathcal{O}$ is a honest challenger for PRF_0, $\mathcal{B}$ behaves as $\mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)$ and if $\mathcal{O}$ samples outputs randomly, $\mathcal{B}$ behaves like $\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$. Hence, $\mathcal{B}$ is a valid adversary against the pseudorandomness of PRF_0 that can breaks its security with non-negligible advantage. Thus, $\mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$ are computationally indistinguishable. $\square$

Lemma D.2. $\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_2^{\mathcal{A}}(1^\lambda)$ are statistically indistinguishable.

Proof. The proof of this lemma follows from Lemma 5.2 of [GGLW22] or Lemma 3.4 of [AMVY21] or Claim 1 of [AV19]. $\square$

Lemma D.3. Assuming the pseudorandomness of PRF_1, $\mathbf{Hyb}_2^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_3^{\mathcal{A}}(1^\lambda)$ are computationally indistinguishable.

Proof. Assume that there exists a PPT adversary that can distinguish between $\mathbf{Hyb}_2^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_3^{\mathcal{A}}(1^\lambda)$ with non-negligible advantage $\epsilon(\lambda)$, i.e,

$$\left|\Pr\left[1 \leftarrow \mathbf{Hyb}_2^{\mathcal{A}}(1^\lambda)\right] - \Pr\left[1 \leftarrow \mathbf{Hyb}_3^{\mathcal{A}}(1^\lambda)\right]\right| > \epsilon(\lambda)$$

We will construct a reduction adversary $\mathcal{B}$ that can break the pseudorandomness of PRF_1 with non-negligible probability. The description of $\mathcal{B}^{\mathcal{O}}$ is as follows.

Setup. $\mathcal{A}$ sends $(Q, n, s, \ell_1, \ldots, \ell_n)$. Set $\mathsf{CRS} := \mathsf{mafe.crs}$. Sample $K^{(1)} \xleftarrow{\$} \{0,1\}^\lambda$ and for each $\mathsf{id} \in [n]$, $(\mathsf{ibe.mpk}_{\mathsf{id}}, \mathsf{ibe.msk}_{\mathsf{id}}) \leftarrow \mathsf{IBE.Setup}(1^\lambda, 1^z)$. Set $\mathsf{MPK}_{\mathsf{id}} := (\ell_{\mathsf{id}}, \mathsf{ibe.mpk}_{\mathsf{id}})$, and $\mathsf{MSK}_{\mathsf{id}} := (\ell_{\mathsf{id}}, \mathsf{ibe.msk}_{\mathsf{id}}, K^{(1)})$. Send $(\mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]})$ to $\mathcal{A}$. Sample $\mathsf{Tag}_1, \ldots, \mathsf{Tag}_Q \xleftarrow{\$} [Q]$. Initiate counter $j = 1$ and dictionary $\mathcal{T}$. If there exists a $\mathsf{Tag} \in [Q]$ and indices $i_1^*, \ldots, i_q^*$ such that $q > \lambda$ and $\mathsf{Tag}_{i_1^*}, \ldots, \mathsf{Tag}_{i_q^*}$ are same as Tag, then abort and output $\perp$.

Pre-challenge Queries. $\mathcal{A}$ makes Q_1 queries of the form $(\mathsf{id}, \mathsf{GID}, x)$. If $\mathsf{GID} \in \mathcal{T}$, set $\mathsf{Tag} = \mathcal{T}[\mathsf{GID}]$. Otherwise, set $\mathcal{T}[\mathsf{GID}] := \mathsf{Tag}_j$, $\mathsf{Tag} = \mathcal{T}[\mathsf{GID}]$ and increment j. Sample $\mathsf{mafe.crs}, \{(\mathsf{mafe.mpk}_{\mathsf{idx}}, \mathsf{mafe.msk}_{\mathsf{idx}})\}_{\mathsf{idx}} \leftarrow \mathsf{tMAFE.GSetup}(1^\lambda, 1^\lambda, 1^n, 1^s, 1^{\ell_1}, \ldots, 1^{\ell_n}; \mathcal{O}(\mathsf{Tag}))$. For each $i \in [L_{\mathsf{id}}]$, sample $\mathsf{ibe.sk}_{\mathsf{Tag},i} \leftarrow \mathsf{IBE.KGen}(\mathsf{ibe.msk}_{\mathsf{id}}, (\mathsf{Tag}, i, \mathsf{mafe.mpk}_{\mathsf{id}}[i]))$. And, $\mathsf{mafe.sk}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{tMAFE.KGen}(\mathsf{id}, \mathsf{mafe.msk}_{\mathsf{id}}, \{\mathsf{mafe.mpk}_{\mathsf{idx}}\}_{\mathsf{idx}}, \mathsf{GID}, x)$. Send $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} := (\mathsf{Tag}, \mathsf{mafe.mpk}_{\mathsf{id}}, \mathsf{mafe.sk}_{\mathsf{id},\mathsf{GID},x}, \{\mathsf{ibe.sk}_{\mathsf{Tag},i}\}_{i\in[L_{\mathsf{id}}]})$ to $\mathcal{A}$.

Challenge Query. $\mathcal{A}$ sends circuit C. For each $\mathsf{Tag} \in [Q]$,

- Sample $R^{\mathsf{Tag}} \leftarrow \{0,1\}^\lambda$, $(\widetilde{F}^{\mathsf{Tag}}, \{w_{\mathsf{id},i,b}\}_{\mathsf{id}\in[n],i\in[L_{\mathsf{id}}],b\in\{0,1\}}) \leftarrow \mathsf{Garble}(1^\lambda, F)$.
- For each $\mathsf{id} \in [n]$, $\mathsf{ibe.ct}^{\mathsf{Tag}}_{\mathsf{id},i,b} \leftarrow \mathsf{IBE.Enc}(\mathsf{ibe.mpk}_{\mathsf{id}}, (\mathsf{Tag}, i, b), w^{\mathsf{Tag}}_{\mathsf{id},i,b})$.

Send $\mathsf{CT} := (\widetilde{F}^{\mathsf{Tag}}, \{\mathsf{ibe.ct}^{\mathsf{Tag}}_{\mathsf{id},i,b}\}_{\mathsf{id},i,b})_{\mathsf{Tag}\in[Q]}$ to $\mathcal{A}$.

Post-challenge Queries. $\mathcal{A}$ makes $Q - Q_1$ queries of the form $(\mathsf{id}, \mathsf{GID}, x)$. If $\mathsf{GID} \in \mathcal{T}$, set $\mathsf{Tag} = \mathcal{T}[\mathsf{GID}]$. Otherwise, set $\mathcal{T}[\mathsf{GID}] := \mathsf{Tag}_j, \mathsf{Tag} = \mathcal{T}[\mathsf{GID}]$ and increment j. Sample $\mathsf{mafe.crs}, \{(\mathsf{mafe.mpk}_{\mathsf{idx}}, \mathsf{mafe.msk}_{\mathsf{idx}})\}_{\mathsf{idx}} \leftarrow \mathsf{tMAFE.GSetup}(1^\lambda, 1^\lambda, 1^n, 1^s, 1^{\ell_1}, \ldots, 1^{\ell_n}; \mathcal{O}(\mathsf{Tag}))$. For each $i \in [L_{\mathsf{id}}]$, sample $\mathsf{ibe.sk}_{\mathsf{Tag},i} \leftarrow \mathsf{IBE.KGen}(\mathsf{ibe.msk}_{\mathsf{id}}, (\mathsf{Tag}, i, \mathsf{mafe.mpk}_{\mathsf{id}}[i]))$. And, $\mathsf{mafe.sk}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{tMAFE.KGen}(\mathsf{id}, \mathsf{mafe.msk}_{\mathsf{id}}, \{\mathsf{mafe.mpk}_{\mathsf{idx}}\}_{\mathsf{idx}}, \mathsf{GID}, x)$. Send $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} := (\mathsf{Tag}, \mathsf{mafe.mpk}_{\mathsf{id}}, \mathsf{mafe.sk}_{\mathsf{id},\mathsf{GID},x}, \{\mathsf{ibe.sk}_{\mathsf{Tag},i}\}_{i\in[L_{\mathsf{id}}]})$ to $\mathcal{A}$.

Guess Phase. Output whatever $\mathcal{A}$ outputs.

As we can see, the running time of $\mathcal{B}$ is polynomial in the running time of $\mathcal{A}$ and λ. If $\mathcal{O}$ is a honest challenger for PRF_1, $\mathcal{B}$ behaves as $\mathbf{Hyb}_2^{\mathcal{A}}(1^\lambda)$ and if $\mathcal{O}$ samples outputs randomly, $\mathcal{B}$ behaves like $\mathbf{Hyb}_3^{\mathcal{A}}(1^\lambda)$. Hence, $\mathcal{B}$ is a valid adversary against the pseudorandomness of PRF_1 that can breaks its security with non-negligible advantage. Thus, $\mathbf{Hyb}_2^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_3^{\mathcal{A}}(1^\lambda)$ are computationally indistinguishable. □

Lemma D.4. $\mathbf{Hyb}_3^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_4^{\mathcal{A}}(1^\lambda)$ are identical.

Proof. As we are sampling $R^{(Tag)}$ uniformly in both the hybrids (explicitly in $\mathbf{Hyb}_3^{\mathcal{A}}(1^\lambda)$ and implicitly in $\mathbf{Hyb}_4^{\mathcal{A}}(1^\lambda)$), it follows form the output distribution that both of these hybrids are identical. □

Lemma D.5. $\mathbf{Hyb}_4^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_{5,1}^{\mathcal{A}}(1^\lambda)$ are identical.

Proof. As we are not substituting any encryptions of IBE with zero strings in $\mathbf{Hyb}_{5,1}^{\mathcal{A}}(1^\lambda)$ for any authorities, both these hybrids remain identically distributed. □

Lemma D.6. Assuming the multi-challenge message security of IBE, $\mathbf{Hyb}_{5,k}^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_{5,k+1}^{\mathcal{A}}(1^\lambda)$ for any $k \in [n]$ are computationally indistinguishable.

Proof. Note that the only difference between $\mathbf{Hyb}_{5,k}^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_{5,k+1}^{\mathcal{A}}(1^\lambda)$ is that for the k-th authority we generate ciphertexts differently in $\mathbf{Hyb}_{5,k+1}^{\mathcal{A}}(1^\lambda)$. Assume that there exists a PPT adversary that can distinguish between $\mathbf{Hyb}_{5,k}^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_{5,k+1}^{\mathcal{A}}(1^\lambda)$ with non-negligible advantage $\epsilon(\lambda)$, i.e,

$$\left|\Pr\left[1 \leftarrow \mathbf{Hyb}_{5,k}^{\mathcal{A}}(1^\lambda)\right] - \Pr\left[1 \leftarrow \mathbf{Hyb}_{5,k+1}^{\mathcal{A}}(1^\lambda)\right]\right| > \epsilon(\lambda)$$

We will construct a reduction adversary $\mathcal{B}$ that can break the multi-challenge security of IBE with non-negligible probability. The description of $\mathcal{B}^{\mathcal{O}}$ is as follows.

Setup, Pre-challenge Queries. Same as $\mathbf{Hyb}_{5,k}^{\mathcal{A}}(1^\lambda)$ except that set $\mathsf{ibe.mpk}_k$ that is received from $\mathcal{O}$. And $\mathsf{ibe.msk}_k = \perp$. For pre-challenge queries, when we need secret keys for k-th authority, query $\mathcal{O}((\mathsf{Tag}, i, \mathsf{mafe.mpk}_k^{\mathsf{Tag}}[i]))$ for $i \in [L_k]$.

Challenge Query. $\mathcal{A}$ sends circuit C. For each $\mathsf{Tag} \in [Q]$,

- Sample $R^{\mathsf{Tag}} \leftarrow \{0,1\}^\lambda$, $(\widetilde{F}^{\mathsf{Tag}}, \{w_{\mathsf{id},i,b}\}_{\mathsf{id}\in[n],i\in[L_{\mathsf{id}}],b\in\{0,1\}}) \leftarrow \mathsf{Garble}(1^\lambda, F)$.
- For each $\mathsf{id} \in [n]$,
  - If $\mathsf{id} < k, i \in [L_{\mathsf{id}}]$, let $\beta = \mathsf{mafe.mpk}_{\mathsf{id}}^{\mathsf{Tag}}[i]$, $\mathsf{ibe.ct}_{\mathsf{id},i,\beta}^{\mathsf{Tag}} \leftarrow \mathsf{IBE.Enc}(\mathsf{ibe.mpk}_{\mathsf{id}}, (\mathsf{Tag}, i, \beta), w_{\mathsf{id},i,\beta}^{\mathsf{Tag}})$ and $\mathsf{ibe.ct}_{\mathsf{id},i,1-\beta}^{\mathsf{Tag}} \leftarrow \mathsf{IBE.Enc}(\mathsf{ibe.mpk}_{\mathsf{id}}, (\mathsf{Tag}, i, 1-\beta), 0^{\left|w_{\mathsf{id},i,1-\beta}^{\mathsf{Tag}}\right|})$.
  - Otherwise, if $\mathsf{id} = k$, $i \in [L_{\mathsf{id}}]$, let $\beta = \mathsf{mafe.mpk}_{\mathsf{id}}^{\mathsf{Tag}}[i]$, $\mathsf{ibe.ct}_{\mathsf{id},i,\beta}^{\mathsf{Tag}} \leftarrow \mathsf{IBE.Enc}(\mathsf{ibe.mpk}_{\mathsf{id}}, (\mathsf{Tag}, i, \beta), w_{\mathsf{id},i,\beta}^{\mathsf{Tag}})$ and $\mathsf{ibe.ct}_{\mathsf{id},i,1-\beta}^{\mathsf{Tag}} \leftarrow \mathcal{O}((\mathsf{Tag}, i, 1-\beta), w_{\mathsf{id},i,1-\beta})$.
  - Otherwise, for $i \in [L_{\mathsf{id}}], b \in \{0,1\}$, $\mathsf{ibe.ct}_{\mathsf{id},i,b}^{\mathsf{Tag}} \leftarrow \mathsf{IBE.Enc}(\mathsf{ibe.mpk}_{\mathsf{id}}, (\mathsf{Tag}, i, b), w_{\mathsf{id},i,b}^{\mathsf{Tag}})$

Send $\mathsf{CT} := (\widetilde{F}^{\mathsf{Tag}}, \{\mathsf{ibe.ct}_{\mathsf{id},i,b}^{\mathsf{Tag}}\}_{\mathsf{id},i,b})_{\mathsf{Tag}\in[Q]}$ to $\mathcal{A}$.

Post-challenge Queries, Guess Phase. Same as $\mathbf{Hyb}_{5,k}^{\mathcal{A}}(1^\lambda)$ except for post-challenge queries, when we need secret keys for k-th authority, query $\mathcal{O}((\mathsf{Tag}, i, \mathsf{mafe.mpk}_k^{\mathsf{Tag}}[i]))$ for $i \in [L_k]$.

Note that the identities queried by $\mathcal{B}$ during encryption are not the same as any identity that $\mathcal{B}$ queried a secret key for. And, the running time of $\mathcal{B}$ is polynomial in the running time of $\mathcal{A}$ and λ. If $\mathcal{O}$ encrypts for all $\mathsf{Tag} \in [Q]$, $w_{\mathsf{id},i,1-\beta[i]}^{\mathsf{Tag}}$ in $\mathsf{ibe.ct}_{\mathsf{id},i,1-\beta[i]}^{\mathsf{Tag}}$, $\mathcal{B}$ behaves as $\mathbf{Hyb}_{5,k}^{\mathcal{A}}(1^\lambda)$ and if $\mathcal{O}$ encrypts all zero string, $\mathcal{B}$ behaves like $\mathbf{Hyb}_{5,k+1}^{\mathcal{A}}(1^\lambda)$. Hence, $\mathcal{B}$ is a valid adversary against the security of IBE that can breaks its multi-challenge security with non-negligible advantage. Thus, $\mathbf{Hyb}_{5,k}^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_{5,k+1}^{\mathcal{A}}(1^\lambda)$ are computationally indistinguishable. ☐

Lemma D.7. $\mathbf{Hyb}_{5,n+1}^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_{6,1}^{\mathcal{A}}(1^\lambda)$ are identical.

Proof. As we are not simulating any garbled circuits for any Tags $\mathbf{Hyb}_{5,1}^{\mathcal{A}}(1^\lambda)$, these two hybrids remain identically distributed. ☐

Lemma D.8. Assuming the security of Garb, $\mathbf{Hyb}_{6,t}^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_{6,t+1}^{\mathcal{A}}(1^\lambda)$ for any $t \in [Q]$ are computationally indistinguishable.

Proof. Assume that there exists a PPT adversary that can distinguish between $\mathbf{Hyb}_{6,t}^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_{6,t+1}^{\mathcal{A}}(1^\lambda)$ with non-negligible advantage $\epsilon(\lambda)$, i.e,

$$\left|\Pr\left[1 \leftarrow \mathbf{Hyb}_{6,t}^{\mathcal{A}}(1^\lambda)\right] - \Pr\left[1 \leftarrow \mathbf{Hyb}_{6,t+1}^{\mathcal{A}}(1^\lambda)\right]\right| > \epsilon(\lambda)$$

We will construct a reduction adversary $\mathcal{B}$ that can break the security of Garb with non-negligible probability. The description of $\mathcal{B}^{\mathcal{O}}$ is as follows.

Setup, Pre-challenge Queries. Same as $\mathbf{Hyb}_{5,n+1}^{\mathcal{A}}(1^\lambda)$.

Challenge Query. $\mathcal{A}$ sends circuit C. For each $\mathsf{Tag} \in [Q]$,

- If $\mathsf{Tag} < t$ $(\widetilde{F}^{\mathsf{Tag}}, \{w_{\mathsf{id},i,\mathsf{mafe.mpk}_{\mathsf{id}}^{\mathsf{Tag}}[i]}\}_{\mathsf{id},i}) \leftarrow \mathsf{Garb.Sim}(1^\lambda, 1^L, 1^{|F|}, \mathsf{mafe.ct}^{\mathsf{Tag}})$ where $L = L_1 + \ldots + L_n$ and $\mathsf{mafe.ct}^{\mathsf{Tag}} \leftarrow \mathsf{tMAFE.Enc}(\{\mathsf{mafe.mpk}_{\mathsf{idx}}^{\mathsf{Tag}}\}_{\mathsf{idx}\in[n]}, C)$.

- Otherwise, $R^{\mathsf{Tag}} \leftarrow \{0,1\}^\lambda$, if $\mathsf{Tag} = t$, $(\widetilde{F}^{\mathsf{Tag}}, \{w_{\mathsf{id},i,\mathsf{mafe.mpk}^{\mathsf{Tag}}_{\mathsf{id}}[i]}\}_{\mathsf{id},i}) \leftarrow \mathcal{O}(F^{\mathsf{Tag}}, \{\mathsf{mafe.mpk}^{\mathsf{Tag}}_{\mathsf{idx}}\}_{\mathsf{idx}})$.
- Otherwise, $R^{\mathsf{Tag}} \leftarrow \{0,1\}^\lambda$, $(\widetilde{F}^{\mathsf{Tag}}, \{w_{\mathsf{id},i,b}\}_{\mathsf{id}\in[n],i\in[L_{\mathsf{id}}],b\in\{0,1\}}) \leftarrow \mathsf{Garble}(1^\lambda, F)$.
- For each $\mathsf{id} \in [n]$, $i \in [L_{\mathsf{id}}]$, let $\beta = \mathsf{mafe.mpk}^{\mathsf{Tag}}_{\mathsf{id}}[i]$, $\mathsf{ibe.ct}^{\mathsf{Tag}}_{\mathsf{id},i,\beta} \leftarrow \mathsf{IBE.Enc}(\mathsf{ibe.mpk}_{\mathsf{id}}, (\mathsf{Tag}, i, \beta), w^{\mathsf{Tag}}_{\mathsf{id},i,\beta})$ and $\mathsf{ibe.ct}^{\mathsf{Tag}}_{\mathsf{id},i,1-\beta} \leftarrow \mathsf{IBE.Enc}(\mathsf{ibe.mpk}_{\mathsf{id}}, (\mathsf{Tag}, i, 1-\beta), 0^{|w^{\mathsf{Tag}}_{\mathsf{id},i,1-\beta}|})$.

Send $\mathsf{CT} := (\widetilde{F}^{\mathsf{Tag}}, \{\mathsf{ibe.ct}^{\mathsf{Tag}}_{\mathsf{id},i,b}\}_{\mathsf{id},i,b})_{\mathsf{Tag}\in[Q]}$ to $\mathcal{A}$.

Post-challenge Queries, Guess Phase. Same as $\mathbf{Hyb}^{\mathcal{A}}_{5,n+1}(1^\lambda)$.

And, the running time of $\mathcal{B}$ is polynomial in the running time of $\mathcal{A}$ and λ. If $\mathcal{O}$ is an honest challenger for Garb, $\mathcal{B}$ behaves as $\mathbf{Hyb}^{\mathcal{A}}_{6,t}(1^\lambda)$ and if $\mathcal{O}$ simulated the garbled circuit, $\mathcal{B}$ behaves like $\mathbf{Hyb}^{\mathcal{A}}_{6,t+1}(1^\lambda)$. Hence, $\mathcal{B}$ is a valid adversary against the security of Garb that can breaks its security with non-negligible advantage. Thus, $\mathbf{Hyb}^{\mathcal{A}}_{6,t}(1^\lambda)$ and $\mathbf{Hyb}^{\mathcal{A}}_{6,t+1}(1^\lambda)$ are computationally indistinguishable. □

Lemma D.9. $\mathbf{Hyb}^{\mathcal{A}}_{6,Q+1}(1^\lambda)$ and $\mathbf{Hyb}^{\mathcal{A}}_{7,1}(1^\lambda)$ are identical.

Proof. As we are not simulating any tMAFE instantiations for any Tags in $\mathbf{Hyb}^{\mathcal{A}}_{7,1}(1^\lambda)$, these two hybrids remain identically distributed. □

Lemma D.10. Assuming the adaptive security of tMAFE, $\mathbf{Hyb}^{\mathcal{A}}_{7,t}(1^\lambda)$ and $\mathbf{Hyb}^{\mathcal{A}}_{7,t+1}(1^\lambda)$ for any $t \in [Q]$ are computationally indistinguishable.

Proof. Assume that there exists a PPT adversary that can distinguish between $\mathbf{Hyb}^{\mathcal{A}}_{7,t}(1^\lambda)$ and $\mathbf{Hyb}^{\mathcal{A}}_{7,t+1}(1^\lambda)$ with non-negligible advantage $\epsilon(\lambda)$, i.e,

$$\left|\Pr\left[1 \leftarrow \mathbf{Hyb}^{\mathcal{A}}_{7,t}(1^\lambda)\right] - \Pr\left[1 \leftarrow \mathbf{Hyb}^{\mathcal{A}}_{7,t+1}(1^\lambda)\right]\right| > \epsilon(\lambda)$$

We will construct a reduction adversary $\mathcal{B}$ that can break the security of tMAFE with non-negligible probability. The description of $\mathcal{B}^{\mathcal{O}}$ is as follows.

Setup. $\mathcal{A}$ sends $(Q, n, s, \ell_1, \ldots, \ell_n)$. Set $\mathsf{CRS} := \mathsf{mafe.crs}$. Sample $K^{(1)} \xleftarrow{\$} \{0,1\}^\lambda$ and for each $\mathsf{id} \in [n]$, $(\mathsf{ibe.mpk}_{\mathsf{id}}, \mathsf{ibe.msk}_{\mathsf{id}}) \leftarrow \mathsf{IBE.Setup}(1^\lambda, 1^z)$. Set $\mathsf{MPK}_{\mathsf{id}} := (\ell_{\mathsf{id}}, \mathsf{ibe.mpk}_{\mathsf{id}})$, and $\mathsf{MSK}_{\mathsf{id}} := (\ell_{\mathsf{id}}, \mathsf{ibe.msk}_{\mathsf{id}}, K^{(1)})$. Send $(\mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]})$ to $\mathcal{A}$. Sample $\mathsf{Tag}_1, \ldots, \mathsf{Tag}_Q \xleftarrow{\$} [Q]$. Initiate counter $j = 1$ and dictionary $\mathcal{T}$. If there exists a $\mathsf{Tag} \in [Q]$ and indices $i_1^*, \ldots, i_q^*$ such that $q > \lambda$ and $\mathsf{Tag}_{i_1^*}, \ldots, \mathsf{Tag}_{i_q^*}$ are same as Tag, then abort and output $\perp$. For each $\mathsf{Tag} \in [Q]$, if $\mathsf{Tag} < t$, $\mathsf{mafe.crs}^{\mathsf{Tag}}, \{\mathsf{mafe.mpk}^{\mathsf{Tag}}_{\mathsf{idx}}\}_{\mathsf{idx}} \leftarrow \mathsf{tMAFE.Sim}_{\mathsf{Tag}}(1^\lambda, 1^\lambda, 1^n, 1^s, 1^{\ell_1}, \ldots, 1^{\ell_n})$. Otherwise, if $\mathsf{Tag} = t$, $\mathsf{mafe.crs}^{\mathsf{Tag}}, \{\mathsf{mafe.mpk}^{\mathsf{Tag}}_{\mathsf{idx}}\}_{\mathsf{idx}} \leftarrow \mathcal{O}(1^\lambda, 1^\lambda, 1^n, 1^s, 1^{\ell_1}, \ldots, 1^{\ell_n})$. Otherwise, $\mathsf{mafe.crs}^{\mathsf{Tag}}, \{(\mathsf{mafe.mpk}^{\mathsf{Tag}}_{\mathsf{idx}}, \mathsf{mafe.msk}^{\mathsf{Tag}}_{\mathsf{idx}})\}_{\mathsf{idx}} \leftarrow \mathsf{tMAFE.GSetup}(1^\lambda, 1^\lambda, 1^n, 1^s, 1^{\ell_1}, \ldots, 1^{\ell_n})$.

Pre-challenge Queries. $\mathcal{A}$ makes Q_1 queries of the form $(\mathsf{id}, \mathsf{GID}, x)$. If $\mathsf{GID} \in \mathcal{T}$, set $\mathsf{Tag} = \mathcal{T}[\mathsf{GID}]$. Otherwise, set $\mathcal{T}[\mathsf{GID}] := \mathsf{Tag}_j, \mathsf{Tag} = \mathcal{T}[\mathsf{GID}]$ and increment j. For each $i \in [L_{\mathsf{id}}]$, sample $\mathsf{ibe.sk}_{\mathsf{Tag},i} \leftarrow \mathsf{IBE.KGen}(\mathsf{ibe.msk}_{\mathsf{id}}, (\mathsf{Tag}, i, \mathsf{mafe.mpk}^{\mathsf{Tag}}_{\mathsf{id}}[i]))$. And, if $\mathsf{Tag} < t$,

$\mathsf{mafe.sk}^{\mathsf{Tag}}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{tMAFE.Sim}_{\mathsf{Tag}}(\mathsf{id},\mathsf{GID},x)$. Otherwise, if $\mathsf{Tag} = t$, $\mathsf{mafe.sk}^{\mathsf{Tag}}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathcal{O}(\mathsf{id},\mathsf{GID}, x)$. Otherwise, $\mathsf{mafe.sk}^{\mathsf{Tag}}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{tMAFE.KGen}(\mathsf{id}, \mathsf{mafe.msk}^{\mathsf{Tag}}_{\mathsf{id}}, \{\mathsf{mafe.mpk}^{\mathsf{Tag}}_{\mathsf{idx}}\}_{\mathsf{idx}}, \mathsf{GID}, x)$. Send $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} := (\mathsf{Tag}, \mathsf{mafe.mpk}^{\mathsf{Tag}}_{\mathsf{id}}, \mathsf{mafe.sk}^{\mathsf{Tag}}_{\mathsf{id},\mathsf{GID},x}, \{\mathsf{ibe.sk}_{\mathsf{Tag},i}\}_{i\in[L_{\mathsf{id}}]})$ to $\mathcal{A}$.

Challenge Query. $\mathcal{A}$ sends circuit C. For each $\mathsf{Tag} \in [Q]$,

- $(\widetilde{F}^{\mathsf{Tag}}, \{w_{\mathsf{id},i,\mathsf{mafe.mpk}^{\mathsf{Tag}}_{\mathsf{id}}[i]}\}_{\mathsf{id},i}) \leftarrow \mathsf{Garb.Sim}(1^\lambda, 1^L, 1^{|F|}, \mathsf{mafe.ct}^{\mathsf{Tag}})$ where $L = L_1 + \ldots + L_n$ and if $\mathsf{Tag} < t$, $\mathsf{mafe.ct}^{\mathsf{Tag}} \leftarrow \mathsf{tMAFE.Sim}(1^{|C|}, \mathcal{V}^{\mathsf{Tag}})$. Otherwise, if $\mathsf{Tag} = t$, $\mathsf{mafe.ct}^{\mathsf{Tag}} \leftarrow \mathcal{O}(C)$. Otherwise, $\mathsf{mafe.ct}^{\mathsf{Tag}} \leftarrow \mathsf{tMAFE.Enc}(\{\mathsf{mafe.mpk}^{\mathsf{Tag}}_{\mathsf{idx}}\}_{\mathsf{idx}\in[n]}, C)$.
- For each $\mathsf{id} \in [n]$, $i \in [L_{\mathsf{id}}]$, let $\beta = \mathsf{mafe.mpk}^{\mathsf{Tag}}_{\mathsf{id}}[i]$, $\mathsf{ibe.ct}^{\mathsf{Tag}}_{\mathsf{id},i,\beta} \leftarrow \mathsf{IBE.Enc}(\mathsf{ibe.mpk}_{\mathsf{id}}, (\mathsf{Tag}, i, \beta), w^{\mathsf{Tag}}_{\mathsf{id},i,\beta})$ and $\mathsf{ibe.ct}^{\mathsf{Tag}}_{\mathsf{id},i,1-\beta} \leftarrow \mathsf{IBE.Enc}(\mathsf{ibe.mpk}_{\mathsf{id}}, (\mathsf{Tag}, i, 1-\beta), 0^{\left|w^{\mathsf{Tag}}_{\mathsf{id},i,1-\beta}\right|})$.

Send $\mathsf{CT} := (\widetilde{F}^{\mathsf{Tag}}, \{\mathsf{ibe.ct}^{\mathsf{Tag}}_{\mathsf{id},i,b}\}_{\mathsf{id},i,b})_{\mathsf{Tag}\in[Q]}$ to $\mathcal{A}$.

Post-challenge Queries. $\mathcal{A}$ makes $Q - Q_1$ queries of the form $(\mathsf{id},\mathsf{GID},x)$. If $\mathsf{GID} \in \mathcal{T}$, set $\mathsf{Tag} = \mathcal{T}[\mathsf{GID}]$. Otherwise, set $\mathcal{T}[\mathsf{GID}] := \mathsf{Tag}_j, \mathsf{Tag} = \mathcal{T}[\mathsf{GID}]$ and increment j. For each $i \in [L_{\mathsf{id}}]$, sample $\mathsf{ibe.sk}_{\mathsf{Tag},i} \leftarrow \mathsf{IBE.KGen}(\mathsf{ibe.msk}_{\mathsf{id}}, (\mathsf{Tag}, i, \mathsf{mafe.mpk}^{\mathsf{Tag}}_{\mathsf{id}}[i]))$. And, if $\mathsf{Tag} < t$, $\mathsf{mafe.sk}^{\mathsf{Tag}}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{tMAFE.Sim}^{C(\cdot)}_{\mathsf{Tag}}(\mathsf{id},\mathsf{GID},x)$. Otherwise, if $\mathsf{Tag} = t$, $\mathsf{mafe.sk}^{\mathsf{Tag}}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathcal{O}(\mathsf{id},\mathsf{GID},x)$. Otherwise, $\mathsf{mafe.sk}^{\mathsf{Tag}}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{tMAFE.KGen}(\mathsf{id}, \mathsf{mafe.msk}^{\mathsf{Tag}}_{\mathsf{id}}, \{\mathsf{mafe.mpk}^{\mathsf{Tag}}_{\mathsf{idx}}\}_{\mathsf{idx}}, \mathsf{GID}, x)$. Send $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} := (\mathsf{Tag}, \mathsf{mafe.mpk}^{\mathsf{Tag}}_{\mathsf{id}}, \mathsf{mafe.sk}^{\mathsf{Tag}}_{\mathsf{id},\mathsf{GID},x}, \{\mathsf{ibe.sk}_{\mathsf{Tag},i}\}_{i\in[L_{\mathsf{id}}]})$ to $\mathcal{A}$.

Guess Phase. Output whatever $\mathcal{A}$ outputs.

And, the running time of $\mathcal{B}$ is polynomial in the running time of $\mathcal{A}$ and λ. If $\mathcal{O}$ is an honest challenger for MAFE, $\mathcal{B}$ behaves as $\mathbf{Hyb}^{\mathcal{A}}_{7,t}(1^\lambda)$ and if $\mathcal{O}$ is a simulator, $\mathcal{B}$ behaves like $\mathbf{Hyb}^{\mathcal{A}}_{7,t+1}(1^\lambda)$. Hence, $\mathcal{B}$ is a valid adversary against the security of MAFE that can breaks its security with non-negligible advantage. Thus, $\mathbf{Hyb}^{\mathcal{A}}_{7,t}(1^\lambda)$ and $\mathbf{Hyb}^{\mathcal{A}}_{7,t+1}(1^\lambda)$ are computationally indistinguishable.
□

E MAFE with Dynamic Collusions from wotMAFE and BFE

Construction E.1 (MAFE with Dynamic Collusions)**.** We provide the construction of an MAFE scheme with dynamic collusions for P/Poly circuits and $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda\in\mathbb{N}}$ (Definition 6.1) using a BFE scheme with dynamic collusions for P/Poly circuits (Definition 3.4) and a wotMAFE scheme for P/Poly circuits and $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda\in\mathbb{N}}$ (Definition 6.2) as follows.

$\underline{\mathsf{GSetup}(1^\lambda, 1^s)}$**.** Output $\mathsf{CRS} := (\lambda, s)$.

$\underline{\mathsf{ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}})}$**.** Sample keys $(\mathsf{bfe.mpk}, \mathsf{bfe.msk}) \leftarrow \mathsf{dBFE.Setup}(1^\lambda, 1^{\lambda+|\mathsf{GID}|+\ell_{\mathsf{id}}}, 1^{\kappa_{\mathsf{id}}})$. Here $\kappa_{\mathsf{id}} \le \mathsf{poly}(\lambda, |\mathsf{GID}|, \ell_{\mathsf{id}})$ is the maximum size of the secret key of wotMAFE for these parameters with $\lceil \log Q \rceil = \lambda$. Output $\mathsf{MPK} := (\ell_{\mathsf{id}}, \mathsf{bfe.mpk})$ and $\mathsf{MSK} := (\ell_{\mathsf{id}}, \mathsf{bfe.msk})$.

$\underline{\mathsf{KGen}(\mathsf{id}, \mathsf{MSK}_{\mathsf{id}}, \mathsf{GID}, x)}$**.** Parse $\mathsf{MSK}_{\mathsf{id}}$ as $(\ell_{\mathsf{id}}, \mathsf{bfe.msk}_{\mathsf{id}})$. Sample $\mathsf{bfe.sk}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{dBFE.KGen}(\mathsf{bfe.msk}_{\mathsf{id}}, (\mathsf{id}, \mathsf{GID}, x))$. Output $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} := \mathsf{bfe.sk}_{\mathsf{id},\mathsf{GID},x}$.

$\underline{\mathsf{Enc}(1^Q, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{I}}, C)}$. Parse $\mathsf{MPK}_{\mathsf{id}}$ as $(\ell_{\mathsf{id}}, \mathsf{bfe.mpk}_{\mathsf{id}})$ for each $\mathsf{id} \in \mathcal{I}$. Let $n = |\mathcal{I}|$ and $\mathsf{id}_1, \ldots, \mathsf{id}_n$ be the increasing order of elements in $\mathcal{I}$.

- $\mathsf{mafe.crs}, \{(\mathsf{mafe.mpk}_i, \mathsf{mafe.msk}_i)\}_{i \in [n]} \leftarrow \mathsf{wotMAFE.GSetup}(1^\lambda, 1^n, 1^s, 1^{\ell_{\mathsf{id}_1}}, \ldots, 1^{\ell_{\mathsf{id}_n}}, Q)$.
- Compute $\mathsf{mafe.ct} \leftarrow \mathsf{wotMAFE.Enc}(1^Q, \{\mathsf{mafe.mpk}_i\}_i, C)$,
- $\forall i \in [n], \mathsf{bfe.ct}_{\mathsf{id}_i} \leftarrow \mathsf{dBFE.Enc}(\mathsf{bfe.mpk}_{\mathsf{id}_i}, \mathsf{wotMAFE.KGen}(\cdot, \mathsf{mafe.msk}_i, \{\mathsf{mafe.mpk}_{i'}\}_{i' \in [n]}, \cdot, \cdot))$.

Output $\mathsf{CT} := (\mathsf{mafe.ct}, \mathcal{I}, \{\mathsf{bfe.ct}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{I}})$.

$\underline{\mathsf{Dec}(\{\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}\}_{\mathsf{id} \in \mathcal{I}}, \mathsf{CT})}$. Parse $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}$ as $\mathsf{bfe.sk}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}$ for each $\mathsf{id} \in \mathcal{I}$ and $\mathsf{CT} := (\mathsf{mafe.ct}, \mathcal{I}', \{\mathsf{bfe.ct}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{I}'})$. If $\mathcal{I} \neq \mathcal{I}'$, abort and output $\perp$.

Otherwise, for each $\mathsf{id} \in \mathcal{I}$, compute $\mathsf{mafe.sk}_{\mathsf{id}} = \mathsf{dBFE.Dec}(\mathsf{bfe.sk}_{\mathsf{id},\mathsf{GID},x}, \mathsf{bfe.ct}_{\mathsf{id}})$. Output $y := \mathsf{wotMAFE.Dec}(\{\mathsf{mafe.sk}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{I}}, \mathsf{mafe.ct})$.

Correctness. The correctness of the scheme follows from the correctness of dBFE and wotMAFE.

F Homomorphic Secret Sharing with Strong Security

In this section, we provide the definition and construction of an HSS scheme for P/Poly circuits that obeys a stronger version of security that what is found in the present literature. Specifically, we construct an HSS scheme in which the attacker not only corrupts shares of any party but also corrupts evaluations of honest shares. We formulate our security using a simulator that only requires $C(x)$ to simulate the honest evaluated shares.

We stress that this notion of security is stronger and not obeyed readily by other HSS constructions found in literature to the best of our knowledge. We construct this object from multi-key fully homomorphic encryption defined in Section F.1. We formally define the HSS scheme in Section F.2 and construct it in Section F.3. This construction is reminiscent of the MPC protocol secure against $n-1$ corruptions from [MW16], §6.1.

F.1 Multi-Key Fully Homomorphic Encryption

Syntax. A multi-key fully homomorphic encryption (mkFHE) scheme mkFHE for input space $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda \in \mathbb{N}}$ and circuit class $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ consists of the following polynomial-time algorithms.

$\mathsf{Setup}(1^\lambda) \to \mathsf{PP}$. The probabilistic setup algorithm takes as input the security parameter λ and outputs the public parameters PP. The following algorithms take PP implicitly unless mentioned.

$\mathsf{KGen}(\mathsf{PP}) \to (\mathsf{pk}, \mathsf{sk})$. The probabilistic key generation algorithm takes as input the public parameters and outputs the public and secret key pair $(\mathsf{pk}, \mathsf{sk})$.

$\mathsf{Enc}(\mathsf{pk}, x) \to \mathsf{ct}$. The probabilistic encryption algorithm takes as input public key pk, a message $x \in \mathcal{X}_\lambda$, and outputs a ciphertext ct.

$\mathsf{Expand}(\mathsf{pk}_1,\ldots,\mathsf{pk}_n,i,\mathsf{ct}) \to \widehat{\mathsf{ct}}$**.** The probabilistic ciphertext expansion algorithm takes as input public keys for n parties, $\mathsf{pk}_1,\ldots,\mathsf{pk}_n$, ciphertext encrypted using the i-th public key ct, and outputs the expanded ciphertext $\widehat{\mathsf{ct}}$.

$\mathsf{Eval}(C,\widehat{\mathsf{ct}}_1,\ldots,\widehat{\mathsf{ct}}_n) \to \widetilde{\mathsf{ct}}$**.** The deterministic evaluation algorithm takes as input description of an n-ary circuit C, expanded ciphertexts $\widehat{\mathsf{ct}}_1,\ldots,\widehat{\mathsf{ct}}_n$, and outputs the evaluated ciphertext $\widetilde{\mathsf{ct}}$.

$\mathsf{Dec}(\mathsf{sk}_1,\ldots,\mathsf{sk}_n,\widetilde{\mathsf{ct}}) \to y$**.** The deterministic decryption algorithm takes as input the secret keys for n parties, $\mathsf{sk}_1,\ldots,\mathsf{sk}_n$, an evaluated ciphertext $\widetilde{\mathsf{ct}}$, and outputs the value y.

$\mathsf{PartDec}(\mathsf{pk}_1,\ldots,\mathsf{pk}_n,i,\mathsf{sk}_i,\widetilde{\mathsf{ct}}) \to \mathsf{sh}_{\mathsf{id}}$**.** The probabilistic partial decryption algorithm takes as input the public keys for all n parties $\mathsf{pk}_1,\ldots,\mathsf{pk}_n$, secret key of the i-th party, sk_i, evaluated ciphertext $\widetilde{\mathsf{ct}}$, and outputs the decryption share for the i-th party, sh_i.

$\mathsf{Recon}(\mathsf{sh}_1,\ldots,\mathsf{sh}_n) \to y$**.** The deterministic reconstruction algorithm takes as input decryption shares from n parties $\mathsf{sh}_1,\ldots,\mathsf{sh}_n$ and outputs the value y.

Definition F.1 (mkFHE)**.** An mkFHE scheme $(\mathsf{Setup},\mathsf{KGen},\mathsf{Enc},\mathsf{Expand},\mathsf{Eval},\mathsf{PartDec},\mathsf{Recon})$ is said to be an mkFHE scheme for circuit class $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ and message space $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$ if it satisfies the following properties.

Expansion correctness. For any $\lambda \in \mathbb{N}, n = n(\lambda)$, $x \in \mathcal{X}_\lambda$,

$$\Pr\left[x = \mathsf{Dec}(\mathsf{sk}_1,\ldots,\mathsf{sk}_n,\widehat{\mathsf{ct}}) \ : \ \begin{array}{l} \forall\, i \in [n], (\mathsf{pk}_i,\mathsf{sk}_i) \leftarrow \mathsf{KGen}(\mathsf{PP}), \\ \text{for any } i' \in [n], \mathsf{ct}_{i'} \leftarrow \mathsf{Enc}(\mathsf{pk}_{i'},x), \\ \widehat{\mathsf{ct}} \leftarrow \mathsf{Expand}(\mathsf{pk}_1,\ldots,\mathsf{pk}_n,i',\mathsf{ct}_{i'}) \end{array} \right] = 1$$

where $\mathsf{PP} \leftarrow \mathsf{Setup}(1^\lambda)$.

Reconstruction correctness. For any $\lambda \in \mathbb{N}$, $n = n(\lambda)$, any $C \in \mathcal{C}_\lambda$, for any $i \in [n], x_i \in \mathcal{X}_\lambda$,

$$\Pr\left[\begin{array}{l} C(x_1,\ldots,x_n) = \\ \mathsf{Recon}(\mathsf{sh}_1,\ldots,\mathsf{sh}_n) \end{array} \ : \ \begin{array}{l} \widehat{\mathsf{ct}}_i \leftarrow \mathsf{Expand}(\mathsf{pk}_1,\ldots,\mathsf{pk}_n,i,\mathsf{ct}_i), \\ \widetilde{\mathsf{ct}} \leftarrow \mathsf{Eval}(C,\widehat{\mathsf{ct}}_1,\ldots,\widehat{\mathsf{ct}}_n), \\ \mathsf{sh}_i = \mathsf{PartDec}(\mathsf{pk}_1,\ldots,\mathsf{pk}_n,i,\mathsf{sk}_i,\widetilde{\mathsf{ct}}) \end{array} \right] = 1$$

where $\mathsf{PP} \leftarrow \mathsf{Setup}(1^\lambda), (\mathsf{pk}_i,\mathsf{sk}_i) \leftarrow \mathsf{KGen}(\mathsf{PP}), \mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i,x_i)$.

Semantic security. For any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that $\forall\ \lambda \in \mathbb{N}$,

$$\left|\Pr\left[1 \leftarrow \mathcal{A}^{\mathsf{Enc}(\mathsf{pk},\cdot)}(1^\lambda,\mathsf{pk})\right] - \Pr\left[1 \leftarrow \mathcal{A}^{\mathsf{Enc}(\mathsf{pk},0)}(1^\lambda,\mathsf{pk})\right]\right| \leq \mathsf{negl}(\lambda)$$

where $\mathsf{PP} \leftarrow \mathsf{Setup}(1^\lambda), (\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{KGen}(\mathsf{PP})$.

Threshold decryption. There exists a stateful PPT simulator Sim such that for any $i^* \in [n]$, and any evaluated ciphertext $\widetilde{\mathsf{ct}}$, for any $\lambda \in \mathbb{N}$,

$$\{\mathsf{PartDec}(\mathsf{pk}_1,\ldots,\mathsf{pk}_n,i^*,\mathsf{sk}_{i^*},\widetilde{\mathsf{ct}})\} \approx_s \{\mathsf{Sim}(C(x_1,\ldots,x_n),i^*,\{\mathsf{sk}_i\}_{i\neq i^*},\widetilde{\mathsf{ct}})\}$$

Remark F.2 ([MW16])**.** There exists a mkFHE scheme for P/Poly circuits and $\mathcal{X} = \{0,1\}^*$ assuming the hardness of LWE with sub-exponential modulus-to-noise ratio and circular security.

Remark F.3. There are other key properties that an mkFHE scheme needs to satisfy such as correctness, compactness, etc. However, we omit them in Definition F.1 and assume that they are satisfied implicitly.

F.2 Definition

Syntax. An all-but-one homomorphic secret sharing (HSS) scheme for the circuit class $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ consists of the following polynomial-time algorithms.

$\mathsf{Share}(1^\lambda, 1^K, C) \to \{\widetilde{C}_i\}_{i \in [K]}$**.** The probabilistic sharing algorithm takes as input security parameter λ, number of shares K, description of a circuit $C \in \mathcal{C}_\lambda$, and outputs the shares $\{\widetilde{C}_i\}_{i \in [K]}$.

$\mathsf{Eval}(i, \widetilde{C}_i, x) \to y_i$**.** The deterministic evaluation algorithm takes as input the i-th share $\widetilde{C}_i$, input x, and outputs the output share y_i.

$\mathsf{Retrieve}(y_1, \ldots, y_K) \to y$**.** The deterministic retrieval algorithm takes as input the output shares $y_1, \ldots, y_K$ and outputs the output y.

Definition F.4 (HSS)**.** A HSS scheme ($\mathsf{Share}, \mathsf{Eval}, \mathsf{Retrieve}$) is said to be an all-but-one HSS scheme for the circuit class $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ if it satisfies the following properties.

Correctness. For any $\lambda \in \mathbb{N}$, and any $C \in \mathcal{C}_\lambda$,

$$\Pr\left[C(x) = \mathsf{Retrieve}(y_1, \ldots, y_K) \;:\; \begin{array}{l} \{\widetilde{C}_i\}_{i \in [K]} \leftarrow \mathsf{Share}(1^\lambda, 1^K, C), \\ \forall\, i \in [K], y_i \leftarrow \mathsf{Eval}(i, \widetilde{C}_i, x) \end{array} \right] = 1$$

Strong Security. There exists a stateful PPT simulator Sim for any stateful PPT adversary $\mathcal{A}$ such that $\forall\ \lambda \in \mathbb{N}$,

$$\left\{ \mathcal{A}^{\mathsf{Eval}(i^*, \widetilde{C}_{i^*}, \cdot)}(\{\widetilde{C}_i\}_{i \neq i^*}) : \begin{array}{l} (K, C, i^* \in [K]) \leftarrow \mathcal{A}(1^\lambda), \\ \{\widetilde{C}_i\}_{i \in [K]} \leftarrow \mathsf{Share}(1^\lambda, 1^K, C) \end{array} \right\}$$
$$\approx_c$$
$$\left\{ \mathcal{A}^{\mathsf{Sim}^{C(\cdot)}(\cdot)}(\{\widetilde{C}_i\}_{i \neq i^*}) : \begin{array}{l} (K, C, i^* \in [K]) \leftarrow \mathcal{A}(1^\lambda), \\ \{\widetilde{C}_i\}_{i \neq i^*} \leftarrow \mathsf{Sim}(1^\lambda, 1^K, 1^{|C|}, i^*) \end{array} \right\}$$

Remark F.5 (HSS with indistinguishability.)**.** We remark that a weaker version of HSS where the security is guaranteed when an adversary gets no evaluations on shares of either x_0 or x_1 can be constructed from a multitude of standard assumptions (cf. [ARS24, DIJL23, CM21, BCG+17a, FGJS17, BGI16] and the references therein). However, these are insufficient for our construction and we require the strong security property as described above.

F.3 HSS for P/Poly from mkFHE

Construction F.6 (HSS)**.** We construct a HSS scheme for $\mathsf{P/Poly}$ circuits using an mkFHE scheme for $\mathsf{P/Poly}$ circuits and unbounded length messages (Definition F.1) as follows.

$\underline{\mathsf{Share}(1^\lambda, 1^K, C)}$**.** Sample public parameters for mkFHE, $\mathsf{PP} \leftarrow \mathsf{mkFHE.Setup}(1^\lambda)$. Now, sample K key pairs, $\forall\ i \in [K], (\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{mkFHE.KGen}(\mathsf{PP})$. Encrypt the description of the circuit C under pk_1, $\mathsf{ct} \leftarrow \mathsf{mkFHE.Enc}(\mathsf{pk}_1, C)$. Now, expand the ciphertext. That is, $\widehat{\mathsf{ct}} \leftarrow \mathsf{mkFHE.Expand}(\mathsf{pk}_1, \ldots, \mathsf{pk}_K, 1, \mathsf{ct})$ Set $\widetilde{C}_i := (\mathsf{pk}_1, \ldots, \mathsf{pk}_K, \mathsf{sk}_i, \widehat{\mathsf{ct}})$ and output $\{\widetilde{C}_i\}_{i \in [K]}$.

$\underline{\mathsf{Eval}(i, \widetilde{C}_i, x)}$**.** Parse $\widetilde{C}_i$ as $(\mathsf{pk}_1, \ldots, \mathsf{pk}_K, \mathsf{sk}_i, \widehat{\mathsf{ct}})$. Evaluate $\mathcal{U}(x, \cdot)$ on $\widehat{\mathsf{ct}}$ where $\mathcal{U}$ denotes the universal circuit, $\widetilde{\mathsf{ct}} = \mathsf{mkFHE.Eval}(\mathcal{U}(x, \cdot), \widehat{\mathsf{ct}})$. Perform partial decryption on $\widetilde{\mathsf{ct}}$ to get the decryption share, $\mathsf{sh}_i = \mathsf{mkFHE.PartDec}(\mathsf{pk}_1, \ldots, \mathsf{pk}_K, i, \mathsf{sk}_i, \widetilde{\mathsf{ct}})$. Output $y_i := \mathsf{sh}_i$.

$\underline{\mathsf{Retrieve}(y_1, \ldots, y_K)}$**.** Parse y_i as sh_i for each $i \in [K]$ and use the mkFHE reconstruction algorithm to retrieve y. That is, $y = \mathsf{mkFHE.Recon}(\mathsf{sh}_1, \ldots, \mathsf{sh}_K)$, and output y.

Correctness. The correctness of the scheme follows from the reconstruction correctness of mkFHE.

Remark F.7 (mkFHE with deterministic PartDec)**.** We remark that the mkFHE scheme from [MW16] requires a smudging noise for output shares making PartDec algorithm probabilistic. However, by rounding the shares, i.e, calculating $\mathsf{sh}_i = \lceil \gamma_i \cdot 2/q \rfloor \mod 2$, we get a statistically correct partial decryption[17]. The correctness and threshold decryption argument hold because of [DHRW16], Lemma 3.4. As such, we need to modify the threshold decryption's simulation algorithm (Sim) which also uses rounding instead of smudging noise. The reconstruction algorithm now simply adds the shares.

We do incur a penalty of K in the error term e' (implicit in γ_i) which is can be handled by increasing the error bound on [GSW13]'s initial error distribution χ. In particular, the error accumulated by adding the shares in [MW16] and the negligible function probability argument from [DHRW16] (which uses an additional union bound), accumulates an additional K which can be handled by augmenting χ. This means that we will be using LWE with sub-exponential modulus-to-noise ratio and exponential modulus.

Remark F.8 (Compactness)**.** Note that our construction also satisfies a compactness requirement similar to other HSS schemes with indistinguishability security by virtue of mkFHE. We also remark that Construction 7.3 does not require this.

Theorem F.9. *Assuming* mkFHE *is a secure mkFHE scheme (Definition F.1) for* P/Poly *circuits and message space* $\{0, 1\}^*$*, Construction F.6 is an HSS scheme (Definition F.4) for* P/Poly *circuits.*

Proof. We prove the security of Construction F.6 using the following series of hybrids and lemmas. We remark that the hybrids and lemmas flow similar to the MPC protocol secure against $n - 1$ corruptions from [MW16], §6.1.

$\mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)$**.** This is the real experiment from Definition F.4.

Setup. $\mathcal{A}$ sends (K, C, i^*) to Chal. Chal samples $\mathsf{PP} \leftarrow \mathsf{mkFHE.Setup}(1^\lambda)$ and $\forall\, i \in [K], (\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{mkFHE.KGen}(\mathsf{PP})$. Next, $\mathsf{ct} \leftarrow \mathsf{mkFHE.Enc}(\mathsf{pk}_1, C)$, $\widehat{\mathsf{ct}} \leftarrow \mathsf{mkFHE.Expand}(\mathsf{pk}_1, \ldots, \mathsf{pk}_K, 1, \mathsf{ct})$. Set $\widetilde{C}_i := (\mathsf{pk}_1, \ldots, \mathsf{pk}_K, \mathsf{sk}_i, \widehat{\mathsf{ct}})$ and send $\{\widetilde{C}_i\}_{i \neq i^*}$ to $\mathcal{A}$.

Query Phase. $\mathcal{A}$ makes polynomially many queries x_q. For each query, perform $\widetilde{\mathsf{ct}}_q = \mathsf{mkFHE.Eval}$ $(\mathcal{U}(x_q, \cdot), \widehat{\mathsf{ct}})$ and $\mathsf{sh}_{i^*,q} = \mathsf{mkFHE.PartDec}(\mathsf{pk}_1, \ldots, \mathsf{pk}_K, i^*, \mathsf{sk}_{i^*}, \widetilde{\mathsf{ct}}_q)$ and send $\mathsf{sh}_{i^*,q}$ to $\mathcal{A}$.

[17]We need to this bit-wise for every bit in the output of the circuit.

$\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$. In this hybrid, we will encrypt the circuit C using the i^*-th public key. The changes are highlighted in red.

Setup. $\mathcal{A}$ sends (K, C, i^*) to Chal. Chal samples $\mathsf{PP} \leftarrow \mathsf{mkFHE.Setup}(1^\lambda)$ and $\forall\, i \in [K], (\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{mkFHE.KGen}(\mathsf{PP})$. Next, $\mathsf{ct} \leftarrow \mathsf{mkFHE.Enc}(\mathsf{pk}_{i^*}, C)$, $\widehat{\mathsf{ct}} \leftarrow \mathsf{mkFHE.Expand}(\mathsf{pk}_1, \ldots, \mathsf{pk}_K, i^*, \mathsf{ct})$. Set $\widetilde{C}_i := (\mathsf{pk}_1, \ldots, \mathsf{pk}_K, \mathsf{sk}_i, \widehat{\mathsf{ct}})$ and send $\{\widetilde{C}_i\}_{i \neq i^*}$ to $\mathcal{A}$.

Query Phase. $\mathcal{A}$ makes polynomially many queries x_q. For each query, perform $\widetilde{\mathsf{ct}}_q = \mathsf{mkFHE.Eval}(\mathcal{U}(x_q, \cdot), \widehat{\mathsf{ct}})$ and $\mathsf{sh}_{i^*,q} = \mathsf{mkFHE.PartDec}(\mathsf{pk}_1, \ldots, \mathsf{pk}_K, i^*, \mathsf{sk}_{i^*}, \widetilde{\mathsf{ct}}_q)$ and send $\mathsf{sh}_{i^*,q}$ to $\mathcal{A}$.

$\mathbf{Hyb}_2^{\mathcal{A}}(1^\lambda)$. In this hybrid, we will use the threshold decryption property of mkFHE to simulate sh_{i^*}. The changes are highlighted in red.

Setup. $\mathcal{A}$ sends (K, C, i^*) to Chal. Chal samples $\mathsf{PP} \leftarrow \mathsf{mkFHE.Setup}(1^\lambda)$ and $\forall\, i \in [K], (\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{mkFHE.KGen}(\mathsf{PP})$. Next, $\mathsf{ct} \leftarrow \mathsf{mkFHE.Enc}(\mathsf{pk}_{i^*}, C)$, $\widehat{\mathsf{ct}} \leftarrow \mathsf{mkFHE.Expand}(\mathsf{pk}_1, \ldots, \mathsf{pk}_K, i^*, \mathsf{ct})$. Set $\widetilde{C}_i := (\mathsf{pk}_1, \ldots, \mathsf{pk}_K, \mathsf{sk}_i, \widehat{\mathsf{ct}})$ and send $\{\widetilde{C}_i\}_{i \neq i^*}$ to $\mathcal{A}$.

Query Phase. $\mathcal{A}$ makes polynomially many queries x_q. For each query, perform $\widetilde{\mathsf{ct}}_q = \mathsf{mkFHE.Eval}(\mathcal{U}(x_q, \cdot), \widehat{\mathsf{ct}})$ and $\mathsf{sh}_{i^*,q} = \mathsf{mkFHE.Sim}(C(x_q), i^*, \mathsf{sk}_{i^*}, \widetilde{\mathsf{ct}}_q)$ and send $\mathsf{sh}_{i^*,q}$ to $\mathcal{A}$.

$\mathbf{Hyb}_3^{\mathcal{A}}(1^\lambda)$. In this hybrid, we will encrypt a zero string instead of C when generating shares. This is the description of Sim from Definition F.4. The changes are highlighted in red.

Setup. $\mathcal{A}$ sends (K, C, i^*) to Chal. Chal samples $\mathsf{PP} \leftarrow \mathsf{mkFHE.Setup}(1^\lambda)$ and $\forall\, i \in [K], (\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{mkFHE.KGen}(\mathsf{PP})$. Next, $\mathsf{ct} \leftarrow \mathsf{mkFHE.Enc}(\mathsf{pk}_{i^*}, 0^{|C|})$, $\widehat{\mathsf{ct}} \leftarrow \mathsf{mkFHE.Expand}(\mathsf{pk}_1, \ldots, \mathsf{pk}_K, i^*, \mathsf{ct})$. Set $\widetilde{C}_i := (\mathsf{pk}_1, \ldots, \mathsf{pk}_K, \mathsf{sk}_i, \widehat{\mathsf{ct}})$ and send $\{\widetilde{C}_i\}_{i \neq i^*}$ to $\mathcal{A}$.

Query Phase. $\mathcal{A}$ makes polynomially many queries x_q. For each query, perform $\widetilde{\mathsf{ct}}_q = \mathsf{mkFHE.Eval}(\mathcal{U}(x_q, \cdot), \widehat{\mathsf{ct}})$ and $\mathsf{sh}_{i^*,q} = \mathsf{mkFHE.Sim}(C(x_q), i^*, \mathsf{sk}_{i^*}, \widetilde{\mathsf{ct}}_q)$ and send $\mathsf{sh}_{i^*,q}$ to $\mathcal{A}$.

Lemma F.10. $\mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$ are identical.

Proof. The proof of this lemma is straight-forward from the expansion correctness of mkFHE. □

Lemma F.11. $\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_2^{\mathcal{A}}(1^\lambda)$ are statistically indistinguishable.

Proof. This holds because of the threshold decryption property of mkFHE. □

Lemma F.12. Assuming semantic security of mkFHE, $\mathbf{Hyb}_2^{\mathcal{A}}(1^\lambda)$, $\mathbf{Hyb}_3^{\mathcal{A}}(1^\lambda)$ are computationally indistinguishable.

Proof. This follows directly from the semantic security of mkFHE. □

□

G Proofs from Section 7.2

In this section, we provide full proof of Theorem 7.4.

$\mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)$. This is the real experiment with Chal.

Setup. $\mathcal{A}$ sends $Q, n, s, k, \chi, \ell_1, \ldots, \ell_n$ to Chal. For each $i \in \mathbb{I}, j \in \mathbb{J}$, Chal samples $\mathsf{crs}^{(i,j)} \leftarrow \mathsf{MAFE.GSetup}(1^\lambda, 1^Q, 1^{n-k}, 1^s)$. In addition, sample for each $\mathsf{id} \in [n]$, $(\mathsf{mpk}_{\mathsf{id}}^{(i,j)}, \mathsf{msk}_{\mathsf{id}}^{(i,j)}) \leftarrow \mathsf{MAFE.ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}})$ for i such that $\mathsf{id} \notin \mathcal{K}_i$. Set $\mathsf{CRS} := (\mathsf{crs}^{(i,j)})_{i,j}$, $\mathsf{MPK}_{\mathsf{id}} := (\mathsf{mpk}_{\mathsf{id}}^{(i,j)})_{i,j}$, and $\mathsf{MSK}_{\mathsf{id}} := (\mathsf{msk}_{\mathsf{id}}^{(i,j)})_{i,j}$ and send $(\mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id} \in [n]})$ to $\mathcal{A}$.

Secret Key Queries. $\mathcal{A}$ sends $(\mathsf{id}, \mathsf{GID}, x)$ for $q \in [Q_1]$ to Chal such that id is not corrupted. For each $i \in \mathbb{I}'_{\mathsf{id}}, j \in \mathbb{J}$, sample $\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j)} \leftarrow \mathsf{MAFE.KGen}(\mathsf{id}, \mathsf{msk}_{\mathsf{id}}^{(i,j)}, \{\mathsf{mpk}_{\mathsf{idx}}^{(i,j)}\}_{\mathsf{idx}}, \mathsf{GID}, x)$. Set $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} = (\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j)})_{i \in \mathbb{I}'_{\mathsf{id}}, j}$ and send it to $\mathcal{A}$.

Corruption Queries. $\mathcal{A}$ sends $(\mathsf{Corr}, \mathsf{id})$ to Chal. Send $(\mathsf{msk}_{\mathsf{id}}^{(i,j)})_{i,j}$ to $\mathcal{A}$.

Challenge Query. $\mathcal{A}$ sends circuit C to Chal. Chal samples $\{\widetilde{C}_i\}_{i \in \mathbb{I}'} \leftarrow \mathsf{HSS.Share}(1^\lambda, 1^{|\mathbb{I}'|}, C)$. For each $i \in \mathbb{I}', j \in \mathbb{J}$, sample $\mathsf{ct}^{(i,j)} \leftarrow \mathsf{MAFE.Enc}(\{\mathsf{mpk}_{\mathsf{id}}^{(i)}\}_{\mathsf{id} \notin \mathcal{K}_i}, F_{ij})$. Set $\mathsf{CT} = (\mathsf{ct}^{(i,j)})_{i,j}$ and send it to $\mathcal{A}$.

Secret Key Queries. $\mathcal{A}$ sends $(\mathsf{id}, \mathsf{GID}, x)$ for $q \in [Q_1+1, Q]$ to Chal such that id is not corrupted. For each $i \in \mathbb{I}'_{\mathsf{id}}, j \in \mathbb{J}$, sample $\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j)} \leftarrow \mathsf{MAFE.KGen}(\mathsf{id}, \mathsf{msk}_{\mathsf{id}}^{(i,j)}, \{\mathsf{mpk}_{\mathsf{idx}}^{(i,j)}\}_{\mathsf{idx}}, \mathsf{GID}, x)$. Set $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} = (\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j)})_{i,j}$ and send it to $\mathcal{A}$.

Corruption Queries. $\mathcal{A}$ sends $(\mathsf{Corr}, \mathsf{id})$ to Chal. Send $(\mathsf{msk}^{(i,j)})_{i,j}$ to $\mathcal{A}$.

Guess Phase. Output whatever $\mathcal{A}$ outputs.

$\mathbf{Hyb}_{1,\kappa}^{\mathcal{A}}(1^\lambda)$. For $\kappa \in [2^\chi + 1]$. In this hybrid, we randomly select $i^* \leftarrow \mathbb{I}$ and simulate the $(i^*, 1), \ldots, (i^*, \kappa - 1)$ instantiations of MAFE. The changes as highlighted in red.

Setup. $\mathcal{A}$ sends $Q, n, s, k, \chi, \ell_1, \ldots, \ell_n$. Randomly choose $i^* \leftarrow \mathbb{I}$. For $i \in \mathbb{I}, j \in \mathbb{J}$,

- If $i \neq i^*$ or $i = i^*$ and $j \geq \kappa$, $\mathsf{crs}^{(i,j)} \leftarrow \mathsf{MAFE.GSetup}(1^\lambda, 1^Q, 1^{n-k}, 1^s)$. Sample, for $\mathsf{id} \in [n], \mathsf{id} \notin \mathcal{K}_i$, $\mathsf{mpk}_{\mathsf{id}}^{(i,j)} \leftarrow \mathsf{MAFE.ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}})$.
- Otherwise, sample $\mathsf{crs}^{(i,j)} \leftarrow \mathsf{MAFE.Sim}(1^\lambda, 1^Q, 1^{n-k}, 1^s)$ and for $\mathsf{id} \notin \mathcal{K}_i$, $(\mathsf{mpk}_{\mathsf{id}}^{(i,j)}, \mathsf{msk}_{\mathsf{id}}^{(i,j)}) \leftarrow \mathsf{MAFE.Sim}(\mathsf{id}, 1^{\ell_{\mathsf{id}}})$.

Set $\mathsf{CRS} := (\mathsf{crs}^{(i,j)})_{i,j}$, $\mathsf{MPK}_{\mathsf{id}} := (\mathsf{mpk}_{\mathsf{id}}^{(i,j)})_{i,j}$, and $\mathsf{MSK}_{\mathsf{id}} := ((\mathsf{msk}_{\mathsf{id}}^{(i,j)})_{i \neq i^*, j}, (\mathsf{msk}_{\mathsf{id}}^{(i,j)})_{i = i^*, j \geq \kappa})$ and send $(\mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id} \in [n]})$ to $\mathcal{A}$.

Secret Key Queries. $\mathcal{A}$ sends $(\mathsf{id}, \mathsf{GID}, x)$ for $q \in [Q_1]$ such that id is not corrupted. For each $i \in \mathbb{I}'_{\mathsf{id}}, j \in \mathbb{J}$,

- If $i \neq i^*$ or $i = i^*$ and $j \geq \kappa$, $\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j)} \leftarrow \mathsf{MAFE.KGen}(\mathsf{id}, \mathsf{msk}_{\mathsf{id}}^{(i,j)}, \{\mathsf{mpk}_{\mathsf{idx}}^{(i,j)}\}_{\mathsf{idx}}, \mathsf{GID}, x)$.
- Otherwise, sample $\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j)} \leftarrow \mathsf{MAFE.Sim}(\mathsf{id}, \mathsf{GID}, x)$.

Set $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} = (\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j)})_{i,j}$ and send it to $\mathcal{A}$.

Corruption Queries. $\mathcal{A}$ sends $(\mathsf{Corr}, \mathsf{id})$. If $\mathsf{id} \notin \mathcal{K}_{i^*}$, abort. Otherwise, send $(\mathsf{msk}_{\mathsf{id}}^{(i,j)})_{i,j}$ to $\mathcal{A}$.

Challenge Query. $\mathcal{A}$ sends circuit C. Sample $\{\widetilde{C}_i\}_{i\in\mathbb{I}'} \leftarrow \mathsf{HSS.Share}(1^\lambda, 1^{|\mathbb{I}'|}, C)$. For each $i \in \mathbb{I}', j \in \mathbb{J}$, sample

- If $i \neq i^*$ or $i = i^*$ and $j \geq \kappa$, $\mathsf{ct}^{(i,j)} \leftarrow \mathsf{MAFE.Enc}(\{\mathsf{mpk}_{\mathsf{id}}^{(i)}\}_{\mathsf{id}\notin\mathcal{K}_i}, F_{ij})$.
- Otherwise, sample $\mathsf{ct}^{(i,j)} \leftarrow \mathsf{MAFE.Sim}(1^{|F_{ij}|}, \mathcal{V}_{ij})$ where $\mathcal{V}_{ij}$ is as defined in Definition 3.1 for the circuit F_{ij}.

Set $\mathsf{CT} = (\mathsf{ct}^{(i,j)})_{i,j}$ and send it to $\mathcal{A}$.

Secret Key Queries. $\mathcal{A}$ sends $(\mathsf{id}, \mathsf{GID}, x)$ for $q \in [Q_1 + 1, Q]$ such that id is not corrupted. For each $i \in \mathbb{I}'_{\mathsf{id}}, j \in \mathbb{J}$,

- If $i \neq i^*$ or $i = i^*$ and $j \geq \kappa$, $\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j)} \leftarrow \mathsf{MAFE.KGen}(\mathsf{id}, \mathsf{msk}_{\mathsf{id}}^{(i,j)}, \{\mathsf{mpk}_{\mathsf{idx}}^{(i,j)}\}_{\mathsf{idx}}, \mathsf{GID}, x)$.
- Otherwise, sample $\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j)} \leftarrow \mathsf{MAFE.Sim}(\mathsf{id}, \mathsf{GID}, x, \mathsf{V})$ where V is as defined in Definition 3.1 for the circuit F_{ij}.

Set $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} = (\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j)})_{i,j}$ and send it to $\mathcal{A}$.

Corruption Queries. $\mathcal{A}$ sends $(\mathsf{Corr}, \mathsf{id})$. If $\mathsf{id} \notin \mathcal{K}_{i^*}$, abort. Otherwise, send $(\mathsf{msk}_{\mathsf{id}}^{(i,j)})_{i,j}$ to $\mathcal{A}$.

Guess Phase. Output whatever $\mathcal{A}$ outputs.

$\mathsf{Hyb}_2^{\mathcal{A}}(1^\lambda)$. In this hybrid, we simulate HSS. The changes are highlighted in red.

Setup. $\mathcal{A}$ sends $Q, n, s, k, \chi, \ell_1, \ldots, \ell_n$. Randomly choose $i^* \leftarrow \mathbb{I}$. For $i \in \mathbb{I}, j \in \mathbb{J}$,

- If $i \neq i^*$, $\mathsf{crs}^{(i,j)} \leftarrow \mathsf{MAFE.GSetup}(1^\lambda, 1^Q, 1^{n-k}, 1^s)$. Sample, for $\mathsf{id} \in [n], \mathsf{id} \notin \mathcal{K}_i$, $\mathsf{mpk}_{\mathsf{id}}^{(i,j)} \leftarrow \mathsf{MAFE.Sim}(\mathsf{id}, 1^{\ell_{\mathsf{id}}})$.
- Otherwise, $\mathsf{crs}^{(i,j)} \leftarrow \mathsf{MAFE.Sim}(1^\lambda, 1^Q, 1^{n-k}, 1^s)$ and for $\mathsf{id} \notin \mathcal{K}_i, (\mathsf{mpk}_{\mathsf{id}}^{(i,j)}, \mathsf{msk}_{\mathsf{id}}^{(i,j)}) \leftarrow \mathsf{MAFE.ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}})$.

Set $\mathsf{CRS} := (\mathsf{crs}^{(i,j)})_{i,j}$, $\mathsf{MPK}_{\mathsf{id}} := (\mathsf{mpk}_{\mathsf{id}}^{(i,j)})_{i,j}$, and $\mathsf{MSK}_{\mathsf{id}} := (\mathsf{msk}_{\mathsf{id}}^{(i,j)})_{i\neq i^*,j})$ and send $(\mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]})$ to $\mathcal{A}$.

Secret Key Queries. $\mathcal{A}$ sends $(\mathsf{id}, \mathsf{GID}, x)$ for $q \in [Q_1]$ such that id is not corrupted. For each $i \in \mathbb{I}'_{\mathsf{id}}, j \in \mathbb{J}$,

- If $i \neq i^*$, sample $\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j)} \leftarrow \mathsf{MAFE.KGen}(\mathsf{id}, \mathsf{msk}_{\mathsf{id}}^{(i,j)}, \{\mathsf{mpk}_{\mathsf{idx}}^{(i,j)}\}_{\mathsf{idx}}, \mathsf{GID}, x)$.
- Otherwise, sample $\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j)} \leftarrow \mathsf{MAFE.Sim}(\mathsf{id}, \mathsf{GID}, x)$.

Set $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} = (\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j)})_{i,j}$ and send it to $\mathcal{A}$.

Corruption Queries. $\mathcal{A}$ sends $(\mathsf{Corr}, \mathsf{id})$. If $\mathsf{id} \notin \mathcal{K}_{i^*}$, abort. Otherwise, send $(\mathsf{msk}_{\mathsf{id}}^{(i,j)})_{i,j}$ to $\mathcal{A}$.

Challenge Query. $\mathcal{A}$ sends circuit C. Sample $\{\widetilde{C}_i\}_{i \neq i^*} \leftarrow \mathsf{HSS.Sim}(1^\lambda, 1^{|\mathbb{I}'|}, 1^{|C|}, i^*)$. For each $i \in \mathbb{I}', j \in \mathbb{J}$, sample

- If $i \neq i^*$, $\mathsf{ct}^{(i,j)} \leftarrow \mathsf{MAFE.Enc}(\{\mathsf{mpk}_{\mathsf{id}}^{(i)}\}_{\mathsf{id} \notin \mathcal{K}_i}, F_{ij})$.
- Otherwise, sample $\mathsf{ct}^{(i,j)} \leftarrow \mathsf{MAFE.Sim}(1^{|F_{ij}|}, \mathcal{V}_{ij})$ where $\mathcal{V}_{ij}$ is as defined in Definition 3.1 for the circuit F_{ij}. However, we will set the output as $\mathsf{HSS.Sim}(X, C(X))$ for $X \in \mathbf{X}_1 \ldots, \mathbf{X}_n$.

Set $\mathsf{CT} = (\mathsf{ct}^{(i,j)})_{i,j}$ and send it to $\mathcal{A}$.

Secret Key Queries. $\mathcal{A}$ sends $(\mathsf{id}, \mathsf{GID}, x)$ for $q \in [Q_1 + 1, Q]$ such that id is not corrupted. For each $i \in \mathbb{I}'_{\mathsf{id}}, j \in \mathbb{J}$,

- If $i \neq i^*$, sample $\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j)} \leftarrow \mathsf{MAFE.KGen}(\mathsf{id}, \mathsf{msk}_{\mathsf{id}}^{(i,j)}, \{\mathsf{mpk}_{\mathsf{idx}}^{(i,j)}\}_{\mathsf{idx}}, \mathsf{GID}, x)$.
- Otherwise, sample $\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j)} \leftarrow \mathsf{MAFE.Sim}(\mathsf{id}, \mathsf{GID}, x, \mathsf{V})$ where V is as defined in Definition 3.1 for the circuit F_{ij}. However, we will set the output as $\mathsf{HSS.Sim}(X, C(X))$ for $X \in \mathbf{X}_1 \ldots, \mathbf{X}_n$.

Set $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} = (\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j)})_{i,j}$ and send it to $\mathcal{A}$.

Corruption Query. $\mathcal{A}$ sends $(\mathsf{Corr}, \mathsf{id})$. If $\mathsf{id} \notin \mathcal{K}_{i^*}$, abort. Otherwise, send $(\mathsf{msk}_{\mathsf{id}}^{(i,j)})_{i,j}$ to $\mathcal{A}$.

Guess Phase. Output whatever $\mathcal{A}$ outputs.

Lemma G.1. $\mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)$, $\mathbf{Hyb}_{1,1}^{\mathcal{A}}(1^\lambda)$ are identically distributed.

Proof. As we are not simulating any MAFE instantiations in $\mathbf{Hyb}_{1,1}^{\mathcal{A}}(1^\lambda)$, we are only sampling i^*. So, these hybrids are identically distributed. □

Lemma G.2. Assuming the security of MAFE, $\mathbf{Hyb}_{1,\kappa}^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_{1,\kappa+1}^{\mathcal{A}}(1^\lambda)$ for $\kappa \in [|\mathbb{J}|]$ are computationally indistinguishable.

Proof. Note that the probability that $\mathbf{Hyb}_{1,\kappa}^{\mathcal{A}}(1^\lambda)$ does not abort is $1/|\mathbb{I}| = 1/\mathsf{poly}(\lambda)$. Let us denote this event by $\overline{\mathsf{Abort}}$. Assume that there exists an adversary $\mathcal{A}$ that can distinguish between $\mathbf{Hyb}_{1,\kappa}^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_{1,\kappa+1}^{\mathcal{A}}(1^\lambda)$ with non-negligible probability $\epsilon(\lambda)$, i.e,

$$\left|\Pr\left[1 \leftarrow \mathbf{Hyb}_{1,\kappa}^{\mathcal{A}}(1^\lambda)\right] - \Pr\left[1 \leftarrow \mathbf{Hyb}_{1,\kappa+1}^{\mathcal{A}}(1^\lambda)\right]\right| > \epsilon(\lambda)$$

We will construct a reduction adversary $\mathcal{B}$ that can break the security of fcMAFE with non-negligible probability. The description of $\mathcal{B}^{\mathcal{O}}$ is as follows.

Setup. $\mathcal{A}$ sends $Q, n, s, k, \chi, \ell_1, \ldots, \ell_n$. Randomly choose $i^* \leftarrow \mathbb{I}$. For $i \in \mathbb{I}, j \in \mathbb{J}$,

- If $i \neq i^*$ or $i = i^*$ and $j > \kappa$, $\mathsf{crs}^{(i,j)} \leftarrow \mathsf{MAFE.GSetup}(1^\lambda, 1^Q, 1^{n-k}, 1^s)$. Sample, for $\mathsf{id} \in [n], \mathsf{id} \notin \mathcal{K}_i$, $\mathsf{mpk}_{\mathsf{id}}^{(i,j)} \leftarrow \mathsf{MAFE.ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}})$. Otherwise, if $i = i^*, j = \kappa$, $\mathsf{crs}^{(i,j)}, \{\mathsf{mpk}_{\mathsf{id}}^{(i,j)}\}_{\mathsf{id}} \leftarrow \mathcal{O}(Q, n-k, s, \ell_1, \ldots, \ell_n)$.
- Otherwise, $\mathsf{crs}^{(i,j)} \leftarrow \mathsf{MAFE.Sim}(1^\lambda, 1^Q, 1^{n-k}, 1^s)$ and for $\mathsf{id} \notin \mathcal{K}_i$, $(\mathsf{mpk}_{\mathsf{id}}^{(i,j)}, \mathsf{msk}_{\mathsf{id}}^{(i,j)}) \leftarrow \mathsf{MAFE.Sim}(\mathsf{id}, 1^{\ell_{\mathsf{id}}})$.

Set $\mathsf{CRS} := (\mathsf{crs}^{(i,j)})_{i,j}$, $\mathsf{MPK}_{\mathsf{id}} := (\mathsf{mpk}_{\mathsf{id}}^{(i,j)})_{i,j}$, and $\mathsf{MSK}_{\mathsf{id}} := ((\mathsf{msk}_{\mathsf{id}}^{(i,j)})_{i \neq i^*,j}, (\mathsf{msk}_{\mathsf{id}}^{(i,j)})_{i=i^*,j \geq \kappa})$ and send $(\mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id} \in [n]})$ to $\mathcal{A}$.

Secret Key Queries. $\mathcal{A}$ sends $(\mathsf{id}, \mathsf{GID}, x)$ for $q \in [Q_1]$ such that id is not corrupted. For each $i \in \mathbb{I}'_{\mathsf{id}}, j \in \mathbb{J}$,

- If $i \neq i^*$ or $i = i^*$ and $j > \kappa$, $\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j)} \leftarrow \mathsf{MAFE.KGen}(\mathsf{id}, \mathsf{msk}_{\mathsf{id}}^{(i,j)}, \{\mathsf{mpk}_{\mathsf{idx}}^{(i,j)}\}_{\mathsf{idx}}, \mathsf{GID}, x)$. Otherwise, if $i = i^*, j = \kappa$, $\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j)} \leftarrow \mathcal{O}(\mathsf{id}, \mathsf{GID}, x)$.
- Otherwise, sample $\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j)} \leftarrow \mathsf{MAFE.Sim}(\mathsf{id}, \mathsf{GID}, x)$.

Set $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} = (\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j)})_{i,j}$ and send it to $\mathcal{A}$.

Corruption Queries. $\mathcal{A}$ sends $(\mathsf{Corr}, \mathsf{id})$. If $\mathsf{id} \notin \mathcal{K}_{i^*}$, abort. Otherwise, send $(\mathsf{msk}_{\mathsf{id}}^{(i,j)})_{i,j}$ to $\mathcal{A}$.

Challenge Query. $\mathcal{A}$ sends circuit C. Sample $\{\widetilde{C}_i\}_{i \in \mathbb{I}'} \leftarrow \mathsf{HSS.Share}(1^\lambda, 1^{|\mathbb{I}'|}, C)$. For each $i \in \mathbb{I}', j \in \mathbb{J}$, sample

- If $i \neq i^*$ or $i = i^*$ and $j > \kappa$, $\mathsf{ct}^{(i,j)} \leftarrow \mathsf{MAFE.Enc}(\{\mathsf{mpk}_{\mathsf{id}}^{(i)}\}_{\mathsf{id} \notin \mathcal{K}_i}, F_{ij})$. Otherwise, if $i = i^*, j = \kappa$, $\mathsf{ct}^{(i,j)} \leftarrow \mathcal{O}(F_{ij})$.
- Otherwise, sample $\mathsf{ct}^{(i,j)} \leftarrow \mathsf{MAFE.Sim}(1^{|F_{ij}|}, \mathcal{V}_{ij})$ where $\mathcal{V}_{ij}$ is as defined in Definition 3.1 for the circuit F_{ij}.

Set $\mathsf{CT} = (\mathsf{ct}^{(i,j)})_{i,j}$ and send it to $\mathcal{A}$.

Secret Key Queries. $\mathcal{A}$ sends $(\mathsf{id}, \mathsf{GID}, x)$ for $q \in [Q_1 + 1, Q]$ such that id is not corrupted. For each $i \in \mathbb{I}'_{\mathsf{id}}, j \in \mathbb{J}$,

- If $i \neq i^*$ or $i = i^*$ and $j > \kappa$, $\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j)} \leftarrow \mathsf{MAFE.KGen}(\mathsf{id}, \mathsf{msk}_{\mathsf{id}}^{(i,j)}, \{\mathsf{mpk}_{\mathsf{idx}}^{(i,j)}\}_{\mathsf{idx}}, \mathsf{GID}, x)$. Otherwise, if $i = i^*, j = \kappa$, $\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j)} \leftarrow \mathcal{O}(\mathsf{id}, \mathsf{GID}, x)$.
- Otherwise, sample $\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j)} \leftarrow \mathsf{MAFE.Sim}(\mathsf{id}, \mathsf{GID}, x, \mathsf{V})$ where V is as defined in Definition 3.1 for the circuit F_{ij}.

Set $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} = (\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j)})_{i,j}$ and send it to $\mathcal{A}$.

Corruption Queries. $\mathcal{A}$ sends $(\mathsf{Corr}, \mathsf{id})$. If $\mathsf{id} \notin \mathcal{K}_{i^*}$, abort. Otherwise, send $(\mathsf{msk}_{\mathsf{id}}^{(i,j)})_{i,j}$ to $\mathcal{A}$.

Guess Phase. Output whatever $\mathcal{A}$ outputs.

As we can see, running time of $\mathcal{B}$ is polynomial in the running time of $\mathcal{A}$ and λ. In addition, the probability that $\mathcal{B}$ does not abort is exactly $\Pr[\overline{\mathsf{Abort}}]$. Hence,

$$\begin{aligned}
\mathsf{Adv}_{\mathcal{B}} &= \left| \Pr[\mathsf{Abort}]\frac{1}{2} + \Pr[\overline{\mathsf{Abort}}] \Pr[\mathcal{A} \text{ wins}] - \frac{1}{2} \right| \\
&= \left| \left(1 - \frac{1}{|\mathbb{I}|}\right) \frac{1}{2} + \frac{1}{|\mathbb{I}|} \Pr[\mathcal{A} \text{ wins}] - \frac{1}{2} \right| \\
&= \frac{\mathsf{Adv}_{\mathcal{A}}}{|\mathbb{I}|}
\end{aligned}$$

If $\mathcal{O}$ is a simulator, $\mathcal{B}$ behaves like $\mathbf{Hyb}^{\mathcal{A}}_{1,\kappa+1}(1^\lambda)$ and if $\mathcal{O}$ is an honest challenger, $\mathcal{B}$ behaves like $\mathbf{Hyb}^{\mathcal{A}}_{1,\kappa}(1^\lambda)$. In addition, as long as $|\mathbb{I}| = \mathsf{poly}(\lambda)$, we have that $\mathcal{B}$ is a successful reduction adversary that can break the security of MAFE with non-negligible probability. $\square$

Lemma G.3. Assuming the security of HSS, $\mathbf{Hyb}^{\mathcal{A}}_{1,|\mathbb{J}|+1}(1^\lambda)$ and $\mathbf{Hyb}^{\mathcal{A}}_{2}(1^\lambda)$ are computationally indistinguishable.

Proof. Assume that there exists a PPT adversary $\mathcal{A}$ that can distinguish between $\mathbf{Hyb}^{\mathcal{A}}_{1,|\mathbb{J}|+1}(1^\lambda)$ and $\mathbf{Hyb}^{\mathcal{A}}_{2}(1^\lambda)$ with non-negligible advantage, $\epsilon(\lambda)$, i.e,

$$\left|\Pr\left[1 \leftarrow \mathbf{Hyb}^{\mathcal{A}}_{1,|\mathbb{J}|+1}(1^\lambda)\right] - \Pr\left[1 \leftarrow \mathbf{Hyb}^{\mathcal{A}}_{2}(1^\lambda)\right]\right| > \epsilon(\lambda)$$

We will construct a reduction adversary $\mathcal{B}$ that can break the security of HSS with non-negligible probability. The description of $\mathcal{B}^{\mathcal{O}}$ is as follows.

Setup. $\mathcal{A}$ sends $Q, n, s, k, \chi, \ell_1, \ldots, \ell_n$. Randomly choose $i^* \leftarrow \mathbb{I}$. For $i \in \mathbb{I}, j \in \mathbb{J}$,

- If $i \neq i^*$, $\mathsf{crs}^{(i,j)} \leftarrow \mathsf{MAFE.GSetup}(1^\lambda, 1^Q, 1^{n-k}, 1^s)$. Sample, for $\mathsf{id} \in [n], \mathsf{id} \notin \mathcal{K}_i$, $\mathsf{mpk}^{(i,j)}_{\mathsf{id}} \leftarrow \mathsf{MAFE.Sim}(\mathsf{id}, 1^{\ell_{\mathsf{id}}})$.
- Otherwise, $\mathsf{crs}^{(i,j)} \leftarrow \mathsf{MAFE.Sim}(1^\lambda, 1^Q, 1^{n-k}, 1^s)$ and for $\mathsf{id} \notin \mathcal{K}_i, (\mathsf{mpk}^{(i,j)}_{\mathsf{id}}, \mathsf{msk}^{(i,j)}_{\mathsf{id}}) \leftarrow \mathsf{MAFE.ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}})$.

Set $\mathsf{CRS} := (\mathsf{crs}^{(i,j)})_{i,j}$, $\mathsf{MPK}_{\mathsf{id}} := (\mathsf{mpk}^{(i,j)}_{\mathsf{id}})_{i,j}$, and $\mathsf{MSK}_{\mathsf{id}} := (\mathsf{msk}^{(i,j)}_{\mathsf{id}})_{i \neq i^*,j})$ and send $(\mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id} \in [n]})$ to $\mathcal{A}$.

Secret Key Queries. $\mathcal{A}$ sends $(\mathsf{id}, \mathsf{GID}, x)$ for $q \in [Q_1]$ such that id is not corrupted. For each $i \in \mathbb{I}'_{\mathsf{id}}, j \in \mathbb{J}$,

- If $i \neq i^*$, sample $\mathsf{sk}^{(i,j)}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{MAFE.KGen}(\mathsf{id}, \mathsf{msk}^{(i,j)}_{\mathsf{id}}, \{\mathsf{mpk}^{(i,j)}_{\mathsf{idx}}\}_{\mathsf{idx}}, \mathsf{GID}, x)$.
- Otherwise, sample $\mathsf{sk}^{(i,j)}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{MAFE.Sim}(\mathsf{id}, \mathsf{GID}, x)$.

Set $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} = (\mathsf{sk}^{(i,j)}_{\mathsf{id},\mathsf{GID},x})_{i,j}$ and send it to $\mathcal{A}$.

Corruption Queries. $\mathcal{A}$ sends $(\mathsf{Corr}, \mathsf{id})$. If $\mathsf{id} \notin \mathcal{K}_{i^*}$, abort. Otherwise, send $(\mathsf{msk}^{(i,j)}_{\mathsf{id}})_{i,j}$ to $\mathcal{A}$.

Challenge Query. $\mathcal{A}$ sends circuit C. Sample $\{\widetilde{C}_i\}_{i \neq i^*} \leftarrow \mathcal{O}(C, |\mathbb{I}'|, i^*)$. For each $i \in \mathbb{I}', j \in \mathbb{J}$, sample

- If $i \neq i^*$, $\mathsf{ct}^{(i,j)} \leftarrow \mathsf{MAFE.Enc}(\{\mathsf{mpk}^{(i)}_{\mathsf{id}}\}_{\mathsf{id} \notin \mathcal{K}_i}, F_{ij})$.
- Otherwise, sample $\mathsf{ct}^{(i,j)} \leftarrow \mathsf{MAFE.Sim}(1^{|F_{ij}|}, \mathcal{V}_{ij})$ where $\mathcal{V}_{ij}$ is as defined in Definition 3.1 for the circuit F_{ij}. However, we will set the output as $\mathcal{O}(X)$ for $X \in \mathbf{X}_1 \ldots, \mathbf{X}_n$.

Set $\mathsf{CT} = (\mathsf{ct}^{(i,j)})_{i,j}$ and send it to $\mathcal{A}$.

Secret Key Queries. $\mathcal{A}$ sends $(\mathsf{id}, \mathsf{GID}, x)$ for $q \in [Q_1 + 1, Q]$ such that id is not corrupted. For each $i \in \mathbb{I}'_{\mathsf{id}}, j \in \mathbb{J}$,

- If $i \neq i^*$, sample $\mathsf{sk}^{(i,j)}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{MAFE.KGen}(\mathsf{id}, \mathsf{msk}^{(i,j)}_{\mathsf{id}}, \{\mathsf{mpk}^{(i,j)}_{\mathsf{idx}}\}_{\mathsf{idx}}, \mathsf{GID}, x)$.

- Otherwise, sample $\mathsf{sk}^{(i,j)}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{MAFE.Sim}(\mathsf{id},\mathsf{GID},x,\mathsf{V})$ where V is as defined in Definition 3.1 for the circuit F_{ij}. However, we will set the output as $\mathcal{O}(X)$ for $X \in \mathbf{X}_1 \ldots, \mathbf{X}_n$.

  Set $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} = (\mathsf{sk}^{(i,j)}_{\mathsf{id},\mathsf{GID},x})_{i,j}$ and send it to $\mathcal{A}$.

Corruption Queries. $\mathcal{A}$ sends $(\mathsf{Corr},\mathsf{id})$. If $\mathsf{id} \notin \mathcal{K}_{i^*}$, abort. Otherwise, send $(\mathsf{msk}^{(i,j)}_{\mathsf{id}})_{i,j}$ to $\mathcal{A}$.

Guess Phase. Output whatever $\mathcal{A}$ outputs.

As we can see, the running time of $\mathcal{B}$ is polynomial in the running time of $\mathcal{A}$, λ. If $\mathcal{O}$ is a honest challenger for HSS, $\mathcal{B}$ behaves like $\mathbf{Hyb}^{\mathcal{A}}_{1,|\mathbb{J}|+1}(1^\lambda)$ and if $\mathcal{O}$ is a simulator for HSS, $\mathcal{B}$ behaves like $\mathbf{Hyb}^{\mathcal{A}}_{2}(1^\lambda)$. Hence, $\mathcal{B}$ is a valid adversary that break the security of HSS with non-negligible probability. Thus, $\mathbf{Hyb}^{\mathcal{A}}_{1,|\mathbb{J}|+1}(1^\lambda)$ and $\mathbf{Hyb}^{\mathcal{A}}_{2}(1^\lambda)$ are computationally indistinguishable.

□

H Proofs from Section 11.2

In this section, we provide full proof of Theorem 11.3.

$\mathbf{Hyb}^{\mathcal{A}}_{0}(1^\lambda)$. This is the real experiment with Chal from Definition 10.2.

Setup. $\mathcal{A}$ sends $(n, L, s_{\mathsf{abe}}, s_{\mathsf{fe}}, \ell_1, \ldots, \ell_n)$ to Chal. Chal samples $\forall\ i \in [n], j \in [L], b \in \{0,1\}$, $\mathsf{crs}^{(i,j,b)} \leftarrow \mathsf{MA\text{-}ABE.GSetup}(1^\lambda, 1^n, 1^{s_{\mathsf{abe}}})$ and $(\mathsf{mpk}^{(i,j,b)}_{\mathsf{id}}, \mathsf{msk}^{(i,j,b)}_{\mathsf{id}}) \leftarrow \mathsf{MA\text{-}ABE.ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}})$. Set $\mathsf{CRS} := (\mathsf{crs}^{(i,j,b)})_{i,j,b}$, $\mathsf{MPK}_{\mathsf{id}} := (\mathsf{mpk}^{(i,j,b)}_{\mathsf{id}})_{i,j,b}$, and $\mathsf{MSK} := (\mathsf{msk}^{(i,j,b)}_{\mathsf{id}})_{i,j,b}$ and send $(\mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]})$ to $\mathcal{A}$.

Pre-challenge Queries. $\mathcal{A}$ sends polynomially many queries of the form $(\mathsf{id},\mathsf{GID},x,y)$. Pad y such that $|y| = L$. For each $i \in [n]$,

- If $i \neq \mathsf{id}$, $\forall\ j, b$, sample $\mathsf{sk}^{(i,j,b)}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{MA\text{-}ABE.KGen}(\mathsf{id}, \mathsf{msk}^{(i,j,b)}_{\mathsf{id}}, \{\mathsf{msk}^{(i,j,b)}_{\mathsf{idx}}\}_{\mathsf{idx}}, \mathsf{GID}, x)$.
- Otherwise, for $j \in [L]$, $\mathsf{sk}^{(i,j,y[j])}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{MA\text{-}ABE.KGen}(\mathsf{id}, \mathsf{msk}^{(i,j,y[j])}_{\mathsf{id}}, \{\mathsf{mpk}^{(i,j,y[j])}_{\mathsf{idx}}\}_{\mathsf{idx}}, \mathsf{GID}, x)$.

  Send $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x,y} := (\{\mathsf{sk}^{(i,j,b)}_{\mathsf{id},\mathsf{GID},x}\}_{i\neq\mathsf{id},j,b}, \{\mathsf{sk}^{(i,j,y[j])}_{\mathsf{id},\mathsf{GID},x}\}_{i=\mathsf{id},j}, x, y)$ to $\mathcal{A}$.

Challenge Query. $\mathcal{A}$ sends predicate P, circuit C. Sample $(\widetilde{C}, \{w_{i,j,b}\}_{i,j,b}) \leftarrow \mathsf{Garble}(1^\lambda, 1^{nL}, C)$. For i, j, $\mathsf{ct}^{(i,j,b)} \leftarrow \mathsf{MA\text{-}ABE.Enc}(\{\mathsf{mpk}^{(i,j,b)}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}, P, w_{i,j,b})$. Send $\mathsf{CT} := (\widetilde{C}, \{\mathsf{ct}^{(i,j,b)}\}_{i,j,b})$ to $\mathcal{A}$.

Post-challenge Queries. $\mathcal{A}$ sends polynomially many queries of the form $(\mathsf{id},\mathsf{GID},x,y)$. Pad y such that $|y| = L$. For each $i \in [n]$,

- If $i \neq \mathsf{id}$, $\forall\ j, b$, sample $\mathsf{sk}^{(i,j,b)}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{MA\text{-}ABE.KGen}(\mathsf{id}, \mathsf{msk}^{(i,j,b)}_{\mathsf{id}}, \{\mathsf{msk}^{(i,j,b)}_{\mathsf{idx}}\}_{\mathsf{idx}}, \mathsf{GID}, x)$.
- Otherwise, for $j \in [L]$, $\mathsf{sk}^{(i,j,y[j])}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{MA\text{-}ABE.KGen}(\mathsf{id}, \mathsf{msk}^{(i,j,y[j])}_{\mathsf{id}}, \{\mathsf{mpk}^{(i,j,y[j])}_{\mathsf{idx}}\}_{\mathsf{idx}}, \mathsf{GID}, x)$.

  Send $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x,y} := (\{\mathsf{sk}^{(i,j,b)}_{\mathsf{id},\mathsf{GID},x}\}_{i\neq\mathsf{id},j,b}, \{\mathsf{sk}^{(i,j,y[j])}_{\mathsf{id},\mathsf{GID},x}\}_{i=\mathsf{id},j}, x, y)$ to $\mathcal{A}$.

Guess Phase. Output whatever $\mathcal{A}$ outputs.

$\mathbf{Hyb}^{\mathcal{A}}_{1,\iota,\gamma}(1^\lambda)$. for $\iota \in [n], \gamma \in [L]$. In this hybrid, we will substitute $w_{i,j,1-y_i[j]}$ with all zero string for $i < \iota, j < \gamma$.

Setup, Pre-challenge Queries. Same as $\mathbf{Hyb}^{\mathcal{A}}_0(1^\lambda)$.

Challenge Query. $\mathcal{A}$ sends predicate P, circuit C. Initiate $\mathsf{flag} = 0$. If there is a satisfying query from the previous phase, set $\mathsf{flag} = 1$ and extract corresponding $Y = (y_1, \ldots, y_n)$. Sample $(\widetilde{C}, \{w_{i,j,b}\}_{i,j,b}) \leftarrow \mathsf{Garble}(1^\lambda, 1^{nL}, C)$. For i, j,

- If $i < \iota$ and $j < \gamma$ and $\mathsf{flag} = 1$, $\mathsf{ct}^{(i,j,1-y_i[j])} \leftarrow \mathsf{MA\text{-}ABE.Enc}(\{\mathsf{mpk}^{(i,j,1-y_i[j])}_{\mathsf{id}}\}_{\mathsf{id}}, P, \overline{0})$[18] and $\mathsf{ct}^{(i,j,y_i[j])} \leftarrow \mathsf{MA\text{-}ABE.Enc}(\{\mathsf{mpk}^{(i,j,y_i[j])}_{\mathsf{id}}\}_{\mathsf{id}}, P, w_{i,j,y_i[j]})$.
- Otherwise, for $b \in \{0,1\}$, $\mathsf{ct}^{(i,j,b)} \leftarrow \mathsf{MA\text{-}ABE.Enc}(\{\mathsf{mpk}^{(i,j,b)}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}, P, w_{i,j,b})$.

Send $\mathsf{CT} := (\widetilde{C}, \{\mathsf{ct}^{(i,j,b)}\}_{i,j,b})$ to $\mathcal{A}$.

Post-challenge, Guess Phase. Same as $\mathbf{Hyb}^{\mathcal{A}}_0(1^\lambda)$.

$\mathbf{Hyb}^{\mathcal{A}}_2(1^\lambda)$. In this hybrid, we will substitute half the wire labels with all zero strings.

Setup, Pre-challenge Queries. Same as $\mathbf{Hyb}^{\mathcal{A}}_{1,n,L}(1^\lambda)$.

Challenge Query. $\mathcal{A}$ sends predicate P, circuit C. If a predicate accepting query is made in previous phase, extract $Y = (y_1, \ldots, y_n)$. Sample $(\widetilde{C}, \{w_{i,j,b}\}_{i,j,b}) \leftarrow \mathsf{Garble}(1^\lambda, 1^{nL}, C)$. For i, j, $\mathsf{ct}^{(i,j,1-y_i[j])} \leftarrow \mathsf{MA\text{-}ABE.Enc}(\{\mathsf{mpk}^{(i,j,1-y_i[j])}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}, P, \overline{0})$ and $\mathsf{ct}^{(i,j,y_i[j])} \leftarrow \mathsf{MA\text{-}ABE.Enc}$ $(\{\mathsf{mpk}^{(i,j,y_i[j])}_{\mathsf{id}}\}_{\mathsf{id}}, P, w_{i,j,y_i[j]})$. If no satisfying query is made, garble a random circuit C' and set $\widetilde{C}$ and ignore the wire labels. For $i, j, b, \mathsf{ct}^{(i,j,b)} \leftarrow \mathsf{MA\text{-}ABE.Enc}(\{\mathsf{mpk}^{(i,j,b)}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}, P, \overline{0})$. Send $\mathsf{CT} := (\widetilde{C}, \{\mathsf{ct}^{(i,j,b)}\}_{i,j,b})$ to $\mathcal{A}$.

Post-challenge, Guess Phase. Same as $\mathbf{Hyb}^{\mathcal{A}}_{1,n,L}(1^\lambda)$.

$\mathbf{Hyb}^{\mathcal{A}}_3(1^\lambda)$. In this hybrid, we will simulate the $(\mathsf{Garble}, \mathsf{Eval})$ instantiation.

Setup, Pre-challenge Queries. Same as $\mathbf{Hyb}^{\mathcal{A}}_2(1^\lambda)$.

Challenge Query. $\mathcal{A}$ sends predicate P, circuit C. If a predicate accepting query is made in previous phase, extract $Y = (y_1, \ldots, y_n)$. Sample $(\widetilde{C}, \{w_{i,j,y_i[j]}\}_{i,j}) \leftarrow \mathsf{Sim}(1^\lambda, 1^{nL}, 1^{|C|}, C(Y))$. For i, j, $\mathsf{ct}^{(i,j,1-y_i[j])} \leftarrow \mathsf{MA\text{-}ABE.Enc}(\{\mathsf{mpk}^{(i,j,1-y_i[j])}_{\mathsf{id}}\}_{\mathsf{id}}, P, \overline{0})$ and $\mathsf{ct}^{(i,j,y_i[j])} \leftarrow \mathsf{MA\text{-}ABE.Enc}$ $(\{\mathsf{mpk}^{(i,j,y_i[j])}_{\mathsf{id}}\}_{\mathsf{id}}, P, w_{i,j,y_i[j]})$. If no satisfying query is made, garble a random circuit C' and set $\widetilde{C}$ and ignore the wire labels. For $i, j, b, \mathsf{ct}^{(i,j,b)} \leftarrow \mathsf{MA\text{-}ABE.Enc}(\{\mathsf{mpk}^{(i,j,b)}_{\mathsf{id}}\}_{\mathsf{id}}, P, \overline{0})$. Send $\mathsf{CT} := (\widetilde{C}, \{\mathsf{ct}^{(i,j,b)}\}_{i,j,b})$ to $\mathcal{A}$.

Post-challenge, Guess Phase. Same as $\mathbf{Hyb}^{\mathcal{A}}_2(1^\lambda)$.

Lemma H.1. $\mathbf{Hyb}^{\mathcal{A}}_0(1^\lambda)$ and $\mathbf{Hyb}^{\mathcal{A}}_{1,1,1}(1^\lambda)$ are identical.

[18]By $\overline{0}$ we denote an all zero string of sufficient length.

Proof. As we are not substituting any wire labels with all zero strings in $\mathbf{Hyb}^{\mathcal{A}}_{1,1,1}(1^\lambda)$, these two hybrids are identically distributed. $\square$

Lemma H.2. Assuming the security of $\mathsf{MA\text{-}ABE}$, $\mathbf{Hyb}^{\mathcal{A}}_{1,\iota,\gamma}(1^\lambda)$ and $\mathbf{Hyb}^{\mathcal{A}}_{1,\iota,\gamma+1}(1^\lambda)$ for $\iota \in [n], \gamma \in [L-1]$ are computationally indistinguishable.

Proof. Assume that there exists a PPT adversary that can distinguish between $\mathbf{Hyb}^{\mathcal{A}}_{1,\iota,\gamma}(1^\lambda)$ and $\mathbf{Hyb}^{\mathcal{A}}_{1,\iota,\gamma+1}(1^\lambda)$ with non-negligible advantage $\epsilon(\lambda)$, i.e,

$$\left|\Pr\left[1 \leftarrow \mathbf{Hyb}^{\mathcal{A}}_{1,\iota,\gamma}(1^\lambda)\right] - \Pr\left[1 \leftarrow \mathbf{Hyb}^{\mathcal{A}}_{1,\iota,\gamma+1}(1^\lambda)\right]\right| > \epsilon(\lambda)$$

We will construct a reduction adversary $\mathcal{B}$ that can break the security of $\mathsf{MA\text{-}ABE}$ with non-negligible probability. The description of $\mathcal{B}^{\mathcal{O}}$ is as follows.

Setup. $\mathcal{A}$ sends $(n, L, s_{\mathsf{abe}}, s_{\mathsf{fe}}, \ell_1, \dots, \ell_n)$. Receive $\mathsf{crs}, \{\mathsf{mpk}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}$ from $\mathcal{O}(1^\lambda, 1^n, 1^{s_{\mathsf{abe}}}, 1^{\ell_1}, \dots, 1^{\ell_n})$. Choose $b^* \xleftarrow{\$} \{0,1\}$ and set $\mathsf{crs}^{(\iota,\gamma,1-b^*)} := \mathsf{crs}$ and $\mathsf{mpk}_{\mathsf{id}}^{(\iota,\gamma,1-b^*)} := \mathsf{mpk}_{\mathsf{id}}$ for $\mathsf{id} \in [n]$. Sample for other i, j, b, $\mathsf{crs}^{(i,j,b)} \leftarrow \mathsf{MA\text{-}ABE.GSetup}(1^\lambda, 1^n, 1^{s_{\mathsf{abe}}})$, $(\mathsf{mpk}_{\mathsf{id}}^{(i,j,b)}, \mathsf{msk}_{\mathsf{id}}^{(i,j,b)}) \leftarrow \mathsf{MA\text{-}ABE.ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}})$. Set $\mathsf{CRS} := (\mathsf{crs}^{(i,j,b)})_{i,j,b}$, $\mathsf{MPK}_{\mathsf{id}} := (\mathsf{mpk}_{\mathsf{id}}^{(i,j,b)})_{i,j,b}$, and $\mathsf{MSK} := (\mathsf{msk}_{\mathsf{id}}^{(i,j,b)})_{i,j,b}$ and send $(\mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}})$ to $\mathcal{A}$.

Pre-challenge Queries. $\mathcal{A}$ sends polynomially many queries of the form $(\mathsf{id}, \mathsf{GID}, x, y)$. Pad y such that $|y| = L$. For each $i \in [n]$, proceed similarly to $\mathbf{Hyb}^{\mathcal{A}}_{1,\iota,\gamma}(1^\lambda)$ except that if we need to query $(\iota, \gamma, 1-b^*)$-th instantiation of $\mathsf{MA\text{-}ABE}$, set $\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{\iota,\gamma,1-b^*)} \leftarrow \mathcal{O}(\mathsf{id}, \mathsf{GID}, x)$. Send $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x,y} := (\{\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j,b)}\}_{i\neq\mathsf{id},j,b}, \{\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j,y[j])}\}_{i=\mathsf{id},j}, x, y)$ to $\mathcal{A}$.

Challenge Query. $\mathcal{A}$ sends predicate P, circuit C. Initiate $\mathsf{flag} = 0$. If there is a satisfying query from the previous phase, set $\mathsf{flag} = 1$ and extract corresponding $Y = (y_1, \dots, y_n)$. If $y_\iota[\gamma] \neq b^*$, abort and output $b' \xleftarrow{\$} \{0,1\}$. Otherwise, Sample $(\widetilde{C}, \{w_{i,j,b}\}_{i,j,b}) \leftarrow \mathsf{Garble}(1^\lambda, 1^{nL}, C)$. For i, j,

- If $i < \iota$ and $j < \gamma$ and $\mathsf{flag} = 1$, $\mathsf{ct}^{(i,j,1-y_i[j])} \leftarrow \mathsf{MA\text{-}ABE.Enc}(\{\mathsf{mpk}_{\mathsf{id}}^{(i,j,1-y_i[j])}\}_{\mathsf{id}}, P, \overline{0})$[19] and $\mathsf{ct}^{(i,j,y_i[j])} \leftarrow \mathsf{MA\text{-}ABE.Enc}(\{\mathsf{mpk}_{\mathsf{id}}^{(i,j,y_i[j])}\}_{\mathsf{id}\in[n]}, P, w_{i,j,y_i[j]})$.
- Otherwise, if $i = \iota, j = \gamma$, $\mathsf{ct}^{i,j,1-b^*} \leftarrow \mathcal{O}(w_{\iota,\gamma,1-b^*}, \overline{0})$, $\mathsf{ct}^{(i,j,b^*)} \leftarrow \mathsf{MA\text{-}ABE.Enc}(\{\mathsf{mpk}_{\mathsf{id}}^{(i,j,b^*)}\}_{\mathsf{id}}, P, w_{i,j,b^*})$.
- Otherwise, for $b \in \{0,1\}$, $\mathsf{ct}^{(i,j,b)} \leftarrow \mathsf{MA\text{-}ABE.Enc}(\{\mathsf{mpk}_{\mathsf{id}}^{(i,j,b)}\}_{\mathsf{id}\in[n]}, P, w_{i,j,b})$.

Send $\mathsf{CT} := (\widetilde{C}, \{\mathsf{ct}^{(i,j,b)}\}_{i,j,b})$ to $\mathcal{A}$.

Post-challenge Queries. Proceed similarly to $\mathbf{Hyb}^{\mathcal{A}}_{1,\iota,\gamma}(1^\lambda)$ except that if we need to query $(\iota, \gamma, 1-b^*)$-th instantiation of $\mathsf{MA\text{-}ABE}$, set $\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{\iota,\gamma,1-b^*)} \leftarrow \mathcal{O}(\mathsf{id}, \mathsf{GID}, x)$.

Guess Phase. Output whatever $\mathcal{A}$ outputs.

[19]By $\overline{0}$ we denote an all zero string of sufficient length.

The probability that $\mathcal{B}$ doesn't abort is 1/2. And, the running time of $\mathcal{B}$ is polynomial in the running time of $\mathcal{A}$ and λ. If $\mathcal{O}$ is encrypts $w_{\iota,\gamma,1-y_\iota[\gamma]}$, $\mathcal{B}$ behaves as $\mathbf{Hyb}^{\mathcal{A}}_{1,\iota,\gamma}(1^\lambda)$ and if $\mathcal{O}$ encrypts $\overline{0}$, $\mathcal{B}$ behaves like $\mathbf{Hyb}^{\mathcal{A}}_{1,\iota,\gamma+1}(1^\lambda)$. Hence, $\mathcal{B}$ is a valid adversary against the security of $\mathsf{MA\text{-}ABE}$ that can break its security with non-negligible advantage, $\epsilon/2$. Thus, $\mathbf{Hyb}^{\mathcal{A}}_{1,\iota,\gamma}(1^\lambda)$ and $\mathbf{Hyb}^{\mathcal{A}}_{1,\iota,\gamma+1}(1^\lambda)$ are computationally indistinguishable. □

Lemma H.3. Assuming the security of $\mathsf{MA\text{-}ABE}$, $\mathbf{Hyb}^{\mathcal{A}}_{1,\iota,L}(1^\lambda)$ and $\mathbf{Hyb}^{\mathcal{A}}_{1,\iota+1,1}(1^\lambda)$ for $\iota \in [n-1]$ are computationally indistinguishable.

Proof. As the only difference between $\mathbf{Hyb}^{\mathcal{A}}_{1,\iota,L}(1^\lambda)$ and $\mathbf{Hyb}^{\mathcal{A}}_{1,\iota+1,1}(1^\lambda)$ is that we substitute $(\iota, L, 1-y_\iota[L])$-th wire label with all zero string in $\mathbf{Hyb}^{\mathcal{A}}_{1,\iota+1,1}(1^\lambda)$, the proof of this lemma is similar to proof of Lemma H.2. □

Lemma H.4. Assuming the security of $\mathsf{MA\text{-}ABE}$, $\mathbf{Hyb}^{\mathcal{A}}_{1,n,L}(1^\lambda)$ and $\mathbf{Hyb}^{\mathcal{A}}_{2}(1^\lambda)$ are computationally indistinguishable.

Proof. As the only difference between $\mathbf{Hyb}^{\mathcal{A}}_{1,n,L}(1^\lambda)$ and $\mathbf{Hyb}^{\mathcal{A}}_{2}(1^\lambda)$ is that we substitute $(n, L, 1-y_n[L])$-th wire label with all zero string in $\mathbf{Hyb}^{\mathcal{A}}_{2}(1^\lambda)$, the proof of this lemma is similar to proof of Lemma H.2. □

Lemma H.5. Assuming the security of $(\mathsf{Garble}, \mathsf{Eval})$, $\mathbf{Hyb}^{\mathcal{A}}_{2}(1^\lambda)$ and $\mathbf{Hyb}^{\mathcal{A}}_{3}(1^\lambda)$ are computationally indistinguishable.

Proof. Assume that there exists a PPT adversary that can distinguish between $\mathbf{Hyb}^{\mathcal{A}}_{2}(1^\lambda)$ and $\mathbf{Hyb}^{\mathcal{A}}_{3}(1^\lambda)$ with non-negligible advantage $\epsilon(\lambda)$, i.e,

$$\left|\Pr\left[1 \leftarrow \mathbf{Hyb}^{\mathcal{A}}_{2}(1^\lambda)\right] - \Pr\left[1 \leftarrow \mathbf{Hyb}^{\mathcal{A}}_{3}(1^\lambda)\right]\right| > \epsilon(\lambda)$$

We will construct a reduction adversary $\mathcal{B}$ that can break the security of $(\mathsf{Garble}, \mathsf{Eval})$ with non-negligible probability. The description of $\mathcal{B}^{\mathcal{O}}$ is as follows.

Setup, Pre-challenge Queries. Similar to $\mathbf{Hyb}^{\mathcal{A}}_{3}(1^\lambda)$.

Challenge Query. $\mathcal{A}$ sends predicate P, circuit C. If a predicate accepting query is made in previous phase, extract $Y = (y_1, \ldots, y_n)$. Sample $(\widetilde{C}, \{w_{i,j,y_i[j]}\}_{i,j}) \leftarrow \mathcal{O}(C, Y)$. For i, j, $\mathsf{ct}^{(i,j,1-y_i[j])} \leftarrow \mathsf{MA\text{-}ABE.Enc}(\{\mathsf{mpk}^{(i,j,1-y_i[j])}_{\mathsf{id}}\}_{\mathsf{id}}, P, \overline{0})$, $\mathsf{ct}^{(i,j,y_i[j])} \leftarrow \mathsf{MA\text{-}ABE.Enc}(\{\mathsf{mpk}^{(i,j,y_i[j])}_{\mathsf{id}}\}_{\mathsf{id}}, P, w_{i,j,y_i[j]})$. If no satisfying query is made, garble a random circuit C' and set $\widetilde{C}$ and ignore the wire labels. For $i, j, b, \mathsf{ct}^{(i,j,b)} \leftarrow \mathsf{MA\text{-}ABE.Enc}(\{\mathsf{mpk}^{(i,j,b)}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}, P, \overline{0})$. Send $\mathsf{CT} := (\widetilde{C}, \{\mathsf{ct}^{(i,j,b)}\}_{i,j,b})$ to $\mathcal{A}$.

Post-challenge, Guess Phase. Similar to $\mathbf{Hyb}^{\mathcal{A}}_{3}(1^\lambda)$.

And, the running time of $\mathcal{B}$ is polynomial in the running time of $\mathcal{A}$ and λ. If $\mathcal{O}$ garbles C honestly, $\mathcal{B}$ behaves as $\mathbf{Hyb}^{\mathcal{A}}_{2}(1^\lambda)$ and if $\mathcal{O}$ simulates using $C(Y)$, $\mathcal{B}$ behaves like $\mathbf{Hyb}^{\mathcal{A}}_{3}(1^\lambda)$. Hence, $\mathcal{B}$ is a valid adversary against the security of $(\mathsf{Garble}, \mathsf{Eval})$ that can break its security with non-negligible advantage. Thus, $\mathbf{Hyb}^{\mathcal{A}}_{2}(1^\lambda)$ and $\mathbf{Hyb}^{\mathcal{A}}_{3}(1^\lambda)$ are computationally indistinguishable. □

I Proofs from Section 11.3

In this section, we provide full proof of Theorem 11.6.

$\mathsf{Hyb}_0^{\mathcal{A}}(1^\lambda)$. This is the real experiment using $\mathcal{E}_0$ from Definition 11.4.

Setup. $\mathcal{A}$ sends $(n, s_{\mathsf{abe}}, L, \ell_1, \ldots, \ell_n)$. For each i, j, b, id, $\mathsf{crs}^{(i,j,b)} \leftarrow \mathsf{MA\text{-}ABE.GSetup}(1^\lambda, 1^n, 1^{s_{\mathsf{abe}}})$. Set $\mathsf{CRS} := (\mathsf{crs}^{(i,j,b)})_{i,j,b}$. Sample $(\mathsf{mpk}_{\mathsf{id}}^{(i,j,b)}, \mathsf{msk}_{\mathsf{id}}^{(i,j,b)}) \leftarrow \mathsf{MA\text{-}ABE.ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}})$. Set $\mathsf{MPK}_{\mathsf{id}} := (\mathsf{mpk}_{\mathsf{id}}^{(i,j,b)})_{i,j,b}$, and $\mathsf{MSK}_{\mathsf{id}} := (\mathsf{msk}_{\mathsf{id}}^{(i,j,b)})_{i,j,b}$. Send $(\mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}})$ to $\mathcal{A}$.

Pre-challenge Queries. $\mathcal{A}$ sends $(\mathsf{id}, \mathsf{GID}, x)$. Sample $\rho \xleftarrow{\$} \{0,1\}^L$. For $i \in [n]$,

- If $i \neq \mathsf{id}$, for each j, b, $\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j,b)} \leftarrow \mathsf{MA\text{-}ABE.KGen}(\mathsf{id}, \mathsf{msk}_{\mathsf{id}}^{(i,j,b)}, \{\mathsf{mpk}_{\mathsf{idx}}^{(i,j,b)}\}_{\mathsf{idx}}, \mathsf{GID}, x)$.
- If $i = \mathsf{id}$, for each j, $\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j,\rho[j])} \leftarrow \mathsf{MA\text{-}ABE.KGen}(\mathsf{id}, \mathsf{msk}_{\mathsf{id}}^{(i,j,\rho[j])}, \{\mathsf{mpk}_{\mathsf{idx}}^{(i,j,\rho[j])}\}_{\mathsf{idx}}, \mathsf{GID}, x)$.

Send $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} := \left(\{\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j,b)}\}_{i\neq\mathsf{id},j,b}, \{\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j,\rho[j])}\}_{i=\mathsf{id},j}, x, \rho\right)$ to $\mathcal{A}$.

Challenge Query. Sample $R_1, \ldots, R_n \xleftarrow{\$} \{0,1\}^L$ and set $\widetilde{R} := R_1 \oplus \ldots \oplus R_n \oplus m$. Compute for each i, j, b, $\mathsf{ct}^{(i,j,b)} \leftarrow \mathsf{MA\text{-}ABE.Enc}(\{\mathsf{mpk}_{\mathsf{id}}^{(i,j,b)}\}, P, R_i[j])$. Send $\mathsf{CT} := (\widetilde{R}, (\mathsf{ct}^{(i,j,b)})_{i,j,b})$ to $\mathcal{A}$.

Post-challenge Queries. $\mathcal{A}$ sends $(\mathsf{id}, \mathsf{GID}, x)$. Sample $\rho \xleftarrow{\$} \{0,1\}^L$. For $i \in [n]$,

- If $i \neq \mathsf{id}$, for each j, b, $\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j,b)} \leftarrow \mathsf{MA\text{-}ABE.KGen}(\mathsf{id}, \mathsf{msk}_{\mathsf{id}}^{(i,j,b)}, \{\mathsf{mpk}_{\mathsf{idx}}^{(i,j,b)}\}_{\mathsf{idx}}, \mathsf{GID}, x)$.
- If $i = \mathsf{id}$, for each j, $\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j,\rho[j])} \leftarrow \mathsf{MA\text{-}ABE.KGen}(\mathsf{id}, \mathsf{msk}_{\mathsf{id}}^{(i,j,\rho[j])}, \{\mathsf{mpk}_{\mathsf{idx}}^{(i,j,\rho[j])}\}_{\mathsf{idx}}, \mathsf{GID}, x)$.

Send $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} := \left(\{\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j,b)}\}_{i\neq\mathsf{id},j,b}, \{\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j,\rho[j])}\}_{i=\mathsf{id},j}, x, \rho\right)$ to $\mathcal{A}$.

Guess Phase. Output whatever $\mathcal{A}$ outputs.

$\mathsf{Hyb}_1^{\mathcal{A}}(1^\lambda)$. In this hybrid, we will sample $\rho_1^*, \ldots, \rho_n^*$ uniformly during setup and use these for the satisfying query whose number in the sequence of secret key queries is also sampled. We will abort if this guess turns out to be wrong.

Setup. $\mathcal{A}$ sends $(n, s_{\mathsf{abe}}, L, \ell_1, \ldots, \ell_n)$. For each i, j, b, id, $\mathsf{crs}^{(i,j,b)} \leftarrow \mathsf{MA\text{-}ABE.GSetup}(1^\lambda, 1^n, 1^{s_{\mathsf{abe}}})$. Set $\mathsf{CRS} := (\mathsf{crs}^{(i,j,b)})_{i,j,b}$. Sample $(\mathsf{mpk}_{\mathsf{id}}^{(i,j,b)}, \mathsf{msk}_{\mathsf{id}}^{(i,j,b)}) \leftarrow \mathsf{MA\text{-}ABE.ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}})$. Set $\mathsf{MPK}_{\mathsf{id}} := (\mathsf{mpk}_{\mathsf{id}}^{(i,j,b)})_{i,j,b}$, $\mathsf{MSK}_{\mathsf{id}} := (\mathsf{msk}_{\mathsf{id}}^{(i,j,b)})_{i,j,b}$. Send $(\mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}})$ to $\mathcal{A}$.

Sample $N^* \leftarrow [q]$ where q is the number unique GIDs $\mathcal{A}$ queries. Sample $\rho_1^*, \ldots, \rho_n^* \xleftarrow{\$} \{0,1\}^L$.

Pre-challenge Queries. $\mathcal{A}$ sends $(\mathsf{id}, \mathsf{GID}, x)$. If this is the N^*-th GID, use ρ_{id}^*. Otherwise, sample $\rho \xleftarrow{\$} \{0,1\}^L$. If this is the last authority for N^*-th GID query and $P(x_{1,\mathsf{GID}}, \ldots, x_{n,\mathsf{GID}}) \neq 1$, abort and output $b' \xleftarrow{\$} \{0,1\}$. For $i \in [n]$,

- If $i \neq \mathsf{id}$, for each j, b, $\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j,b)} \leftarrow \mathsf{MA\text{-}ABE.KGen}(\mathsf{id}, \mathsf{msk}_{\mathsf{id}}^{(i,j,b)}, \{\mathsf{mpk}_{\mathsf{idx}}^{(i,j,b)}\}_{\mathsf{idx}}, \mathsf{GID}, x)$.

- If $i = \mathsf{id}$, for each j, $\mathsf{sk}^{(i,j,\rho[j])}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{MA\text{-}ABE.KGen}(\mathsf{id}, \mathsf{msk}^{(i,j,\rho[j])}_{\mathsf{id}}, \{\mathsf{mpk}^{(i,j,\rho[j])}_{\mathsf{idx}}\}_{\mathsf{idx}}, \mathsf{GID}, x)$.

Send $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} := \left(\{\mathsf{sk}^{(i,j,b)}_{\mathsf{id},\mathsf{GID},x}\}_{i\neq\mathsf{id},j,b}, \{\mathsf{sk}^{(i,j,\rho[j])}_{\mathsf{id},\mathsf{GID},x}\}_{i=\mathsf{id},j}, x, \rho\right)$ to $\mathcal{A}$.

Challenge Query. Sample $R_1, \ldots, R_n \xleftarrow{\$} \{0,1\}^L$ and set $\widetilde{R} := R_1 \oplus \ldots \oplus R_n \oplus m$. Compute for each i, j, b, $\mathsf{ct}^{(i,j,b)} \leftarrow \mathsf{MA\text{-}ABE.Enc}(\{\mathsf{mpk}^{(i,j,b)}_{\mathsf{id}}\}, P, R_i[j])$. Send $\mathsf{CT} := (\widetilde{R}, (\mathsf{ct}^{(i,j,b)})_{i,j,b})$ to $\mathcal{A}$.

Post-challenge Queries. $\mathcal{A}$ sends $(\mathsf{id}, \mathsf{GID}, x)$. If this is the N^*-th GID, use ρ^*_{id}. Otherwise, sample $\rho \xleftarrow{\$} \{0,1\}^L$. If this is the last authority for N^*-th GID query and $P(x_{1,\mathsf{GID}}, \ldots, x_{n,\mathsf{GID}}) \neq 1$, abort and output $b' \xleftarrow{\$} \{0,1\}$. For $i \in [n]$,

- If $i \neq \mathsf{id}$, for each j, b, $\mathsf{sk}^{(i,j,b)}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{MA\text{-}ABE.KGen}(\mathsf{id}, \mathsf{msk}^{(i,j,b)}_{\mathsf{id}}, \{\mathsf{mpk}^{(i,j,b)}_{\mathsf{idx}}\}_{\mathsf{idx}}, \mathsf{GID}, x)$.
- If $i = \mathsf{id}$, for each j, $\mathsf{sk}^{(i,j,\rho[j])}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{MA\text{-}ABE.KGen}(\mathsf{id}, \mathsf{msk}^{(i,j,\rho[j])}_{\mathsf{id}}, \{\mathsf{mpk}^{(i,j,\rho[j])}_{\mathsf{idx}}\}_{\mathsf{idx}}, \mathsf{GID}, x)$.

Send $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} := \left(\{\mathsf{sk}^{(i,j,b)}_{\mathsf{id},\mathsf{GID},x}\}_{i\neq\mathsf{id},j,b}, \{\mathsf{sk}^{(i,j,\rho[j])}_{\mathsf{id},\mathsf{GID},x}\}_{i=\mathsf{id},j}, x, \rho\right)$ to $\mathcal{A}$.

Guess Phase. Output whatever $\mathcal{A}$ outputs.

$\mathbf{Hyb}^{\mathcal{A}}_{2,\iota,\gamma}(1^\lambda)$. for $\iota \in [n], \gamma \in [L]$. In this hybrid, we will replace the $(i, j, 1-\rho^*_i[j])$-th instantiation's encryption message with $1 - R_i[j]$ for $i < \iota$ and $j < \gamma$.

Setup, Pre-challenge Queries. Same as $\mathbf{Hyb}^{\mathcal{A}}_1(1^\lambda)$.

Challenge Query. Sample $R_1, \ldots, R_n \xleftarrow{\$} \{0,1\}^L$ and set $\widetilde{R} := R_1 \oplus \ldots \oplus R_n \oplus m$. Compute for each i, j,

- If $i < \iota, j < \gamma$ $\mathsf{ct}^{(i,j,\rho^*_i[j])} \leftarrow \mathsf{MA\text{-}ABE.Enc}(\{\mathsf{mpk}^{(i,j,\rho^*_i[j])}_{\mathsf{id}}\}, P, R_i[j])$ and $\mathsf{ct}^{(i,j,1-\rho^*_i[j])} \leftarrow \mathsf{MA\text{-}ABE.Enc}(\{\mathsf{mpk}^{(i,j,1-\rho^*_i[j])}_{\mathsf{id}}\}, P, 1 - R_i[j])$.
- Otherwise, for $b \in \{0,1\}$, $\mathsf{ct}^{(i,j,b)} \leftarrow \mathsf{MA\text{-}ABE.Enc}(\{\mathsf{mpk}^{(i,j,b)}_{\mathsf{id}}\}, P, R_i[j])$.

Send $\mathsf{CT} := (\widetilde{R}, (\mathsf{ct}^{(i,j,b)})_{i,j,b})$ to $\mathcal{A}$.

Post-challenge Queries, Guess Phase. Same as $\mathbf{Hyb}^{\mathcal{A}}_1(1^\lambda)$.

$\mathbf{Hyb}^{\mathcal{A}}_3(1^\lambda)$. In this hybrid, each $(i, j, 1 - \rho^*_i[j])$-th instantiation's encryption message is $1 - R_i[j]$.

Setup, Pre-challenge Queries. Same as $\mathbf{Hyb}^{\mathcal{A}}_{2,n,L}(1^\lambda)$.

Challenge Query. Sample $R_1, \ldots, R_n \xleftarrow{\$} \{0,1\}^L$ and set $\widetilde{R} := R_1 \oplus \ldots \oplus R_n \oplus m$. Compute for each i, j, $\mathsf{ct}^{(i,j,\rho^*_i[j])} \leftarrow \mathsf{MA\text{-}ABE.Enc}(\{\mathsf{mpk}^{(i,j,\rho^*_i[j])}_{\mathsf{id}}\}, P, R_i[j])$ and $\mathsf{ct}^{(i,j,1-\rho^*_i[j])} \leftarrow \mathsf{MA\text{-}ABE.Enc}(\{\mathsf{mpk}^{(i,j,1-\rho^*_i[j])}_{\mathsf{id}}\}, P, 1 - R_i[j])$. Send $\mathsf{CT} := (\widetilde{R}, (\mathsf{ct}^{(i,j,b)})_{i,j,b})$ to $\mathcal{A}$.

Post-challenge Queries, Guess Phase. Same as $\mathbf{Hyb}^{\mathcal{A}}_{2,n,L}(1^\lambda)$.

$\mathbf{Hyb}_4^{\mathcal{A}}(1^\lambda)$. In this hybrid, we will use $\mathcal{E}_1$.

Setup. $\mathcal{A}$ sends $(n, s_{\mathsf{abe}}, L, \ell_1, \ldots, \ell_n)$. For each i, j, b, id, $\mathsf{crs}^{(i,j,b)} \leftarrow \mathsf{MA\text{-}ABE.GSetup}(1^\lambda, 1^n, 1^{s_{\mathsf{abe}}})$. Set $\mathsf{CRS} := (\mathsf{crs}^{(i,j,b)})_{i,j,b}$. Sample $(\mathsf{mpk}_{\mathsf{id}}^{(i,j,b)}, \mathsf{msk}_{\mathsf{id}}^{(i,j,b)}) \leftarrow \mathsf{MA\text{-}ABE.ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}})$. Set $\mathsf{MPK}_{\mathsf{id}} := (\mathsf{mpk}_{\mathsf{id}}^{(i,j,b)})_{i,j,b}$, $\mathsf{MSK}_{\mathsf{id}} := (\mathsf{msk}_{\mathsf{id}}^{(i,j,b)})_{i,j,b}$. Send $(\mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}})$ to $\mathcal{A}$.

Pre-challenge Queries. $\mathcal{A}$ sends $(\mathsf{id}, \mathsf{GID}, x)$. Sample $\rho \xleftarrow{\$} \{0,1\}^L$. For $i \in [n]$,

- If $i \neq \mathsf{id}$, for each j, b, $\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j,b)} \leftarrow \mathsf{MA\text{-}ABE.KGen}(\mathsf{id}, \mathsf{msk}_{\mathsf{id}}^{(i,j,b)}, \{\mathsf{mpk}_{\mathsf{idx}}^{(i,j,b)}\}_{\mathsf{idx}}, \mathsf{GID}, x)$.
- If $i = \mathsf{id}$, for each j, $\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j,\rho[j])} \leftarrow \mathsf{MA\text{-}ABE.KGen}(\mathsf{id}, \mathsf{msk}_{\mathsf{id}}^{(i,j,\rho[j])}, \{\mathsf{mpk}_{\mathsf{idx}}^{(i,j,\rho[j])}\}_{\mathsf{idx}}, \mathsf{GID}, x)$.

Send $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} := \left(\{\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j,b)}\}_{i\neq\mathsf{id},j,b}, \{\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j,\rho[j])}\}_{i=\mathsf{id},j}, x, \rho\right)$ to $\mathcal{A}$.

Challenge Query. Sample $R_1, \ldots, R_n \xleftarrow{\$} \{0,1\}^L$.

- If satisfying query is made, set $\widetilde{R} := R_1 \oplus \ldots \oplus R_n \oplus m$. Compute for each i, j, b, $\mathsf{ct}^{(i,j,b)} \leftarrow \mathsf{MA\text{-}ABE.Enc}(\{\mathsf{mpk}_{\mathsf{id}}^{(i,j,b)}\}, P, R_i[j])$.
- Otherwise, sample $\widetilde{R} \leftarrow \{0,1\}^L$. Compute for each i, j, $\mathsf{ct}^{(i,j,R_i[j])} \leftarrow \mathsf{MA\text{-}ABE.Enc}(\{\mathsf{mpk}_{\mathsf{id}}^{(i,j,R_i[j])}\}, P, 0)$ and $\mathsf{ct}^{(i,j,1-R_i[j])} \leftarrow \mathsf{MA\text{-}ABE.Enc}(\{\mathsf{mpk}_{\mathsf{id}}^{(i,j,1-R_i[j])}\}, P, 1)$.

Send $\mathsf{CT} := (\widetilde{R}, (\mathsf{ct}^{(i,j,b)})_{i,j,b})$ to $\mathcal{A}$.

Post-challenge Queries. $\mathcal{A}$ sends $(\mathsf{id}, \mathsf{GID}, x)$. If this is the GID with predicate accepting query and this is the last id, set $\rho := (\bigoplus_{\mathsf{idx}\neq\mathsf{id}} \rho_{\mathsf{idx},\mathsf{GID}} \oplus R_{\mathsf{idx}}) \oplus R_{\mathsf{id}} \oplus \widetilde{R} \oplus m$. Otherwise, sample $\rho \xleftarrow{\$} \{0,1\}^L$. For $i \in [n]$,

- If $i \neq \mathsf{id}$, for each j, b, $\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j,b)} \leftarrow \mathsf{MA\text{-}ABE.KGen}(\mathsf{id}, \mathsf{msk}_{\mathsf{id}}^{(i,j,b)}, \{\mathsf{mpk}_{\mathsf{idx}}^{(i,j,b)}\}_{\mathsf{idx}}, \mathsf{GID}, x)$.
- If $i = \mathsf{id}$, for each j, $\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j,\rho[j])} \leftarrow \mathsf{MA\text{-}ABE.KGen}(\mathsf{id}, \mathsf{msk}_{\mathsf{id}}^{(i,j,\rho[j])}, \{\mathsf{mpk}_{\mathsf{idx}}^{(i,j,\rho[j])}\}_{\mathsf{idx}}, \mathsf{GID}, x)$.

Send $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} := \left(\{\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j,b)}\}_{i\neq\mathsf{id},j,b}, \{\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j,\rho[j])}\}_{i=\mathsf{id},j}, x, \rho\right)$ to $\mathcal{A}$.

Guess Phase. Output whatever $\mathcal{A}$ outputs.

Lemma I.1. $\left|\Pr\left[1 \leftarrow \mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)\right] - \Pr\left[1 \leftarrow \mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)\right]\right| \leq 1/q$.

Proof. As $\mathcal{A}$ makes at most one satisfying query which is one of the q GID queries made, Chal's guess will be correct with at most $1/q$ probability. If this happens, as we sample ρ_{id} uniformly during secret key generation, the output distribution remains identical in both the hybrids. Thus the statistical distance between these hybrids is at most $1/q$. □

Lemma I.2. $\left|\Pr\left[1 \leftarrow \mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)\right] - \Pr\left[1 \leftarrow \mathbf{Hyb}_{2,1,1}^{\mathcal{A}}(1^\lambda)\right]\right| \leq 1/q$.

Proof. As we are changing any encryptions in $\mathbf{Hyb}_{2,1,1}^{\mathcal{A}}(1^\lambda)$, the proof of this claim in similar to proof of Lemma I.1. □

Lemma I.3. Assuming the security of $\mathsf{MA\text{-}ABE}$, $\mathbf{Hyb}^{\mathcal{A}}_{2,\iota,\gamma}(1^\lambda)$ and $\mathbf{Hyb}^{\mathcal{A}}_{2,\iota,\gamma+1}(1^\lambda)$ for $\iota \in [n], \gamma \in [L-1]$ are computationally indistinguishable.

Proof. Assume that there exists a PPT adversary that can distinguish between $\mathbf{Hyb}^{\mathcal{A}}_{2,\iota,\gamma}(1^\lambda)$ and $\mathbf{Hyb}^{\mathcal{A}}_{2,\iota,\gamma+1}(1^\lambda)$ with non-negligible advantage $\epsilon(\lambda)$, i.e,

$$\left|\Pr\left[1 \leftarrow \mathbf{Hyb}^{\mathcal{A}}_{2,\iota,\gamma}(1^\lambda)\right] - \Pr\left[1 \leftarrow \mathbf{Hyb}^{\mathcal{A}}_{2,\iota,\gamma+1}(1^\lambda)\right]\right| > \epsilon(\lambda)$$

We will construct a reduction adversary $\mathcal{B}$ that can break the security of $\mathsf{MA\text{-}ABE}$ with non-negligible probability. The description of $\mathcal{B}^{\mathcal{O}}$ is as follows.

Setup. $\mathcal{A}$ sends $(n, s_{\mathsf{abe}}, L, \ell_1, \ldots, \ell_n)$. Sample $\rho_1^*, \ldots, \rho_n^* \xleftarrow{\$} \{0,1\}^L$. Receive $\mathsf{crs}^{(i,j,1-\rho_\iota^*[\gamma])}$ and $\mathsf{mpk}_{\mathsf{id}}^{(i,j,1-\rho_\iota^*[\gamma])}\}_{\mathsf{id}\in[n]}$. For other i, j, b, id, $\mathsf{crs}^{(i,j,b)} \leftarrow \mathsf{MA\text{-}ABE.GSetup}(1^\lambda, 1^n, 1^{s_{\mathsf{abe}}})$. Set $\mathsf{CRS} := (\mathsf{crs}^{(i,j,b)})_{i,j,b}$. Run $(\mathsf{mpk}_{\mathsf{id}}^{(i,j,b)}, \mathsf{msk}_{\mathsf{id}}^{(i,j,b)}) \leftarrow \mathsf{MA\text{-}ABE.ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}})$. Set $\mathsf{MPK}_{\mathsf{id}} := (\mathsf{mpk}_{\mathsf{id}}^{(i,j,b)})_{i,j,b}$, $\mathsf{MSK}_{\mathsf{id}} := (\mathsf{msk}_{\mathsf{id}}^{(i,j,b)})_{i,j,b}$. Send $(\mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}})$ to $\mathcal{A}$. Sample $N^* \leftarrow [q]$ where q is the number unique GIDs $\mathcal{A}$ queries.

Pre-challenge Queries. Proceed similar to $\mathbf{Hyb}^{\mathcal{A}}_{1,\iota,\gamma}(1^\lambda)$. If we need to generate secret keys for $(\iota, \gamma, 1-\rho_i^*[j])$-th instantiation, set $\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(\iota,\gamma,1-\rho_i^*[j])} \leftarrow \mathcal{O}(\mathsf{id}, \mathsf{GID}, x)$.

Challenge Query. Sample $R_1, \ldots, R_n \xleftarrow{\$} \{0,1\}^L$ and set $\widetilde{R} := R_1 \oplus \ldots \oplus R_n \oplus m$. Compute for each i, j,

- If $i < \iota$ or $i = \iota, j < \gamma$ $\mathsf{ct}^{(i,j,\rho_i^*[j])} \leftarrow \mathsf{MA\text{-}ABE.Enc}(\{\mathsf{mpk}_{\mathsf{id}}^{(i,j,\rho_i^*[j])}\}, P, R_i[j])$ and $\mathsf{ct}^{(i,j,1-\rho_i^*[j])} \leftarrow \mathsf{MA\text{-}ABE.Enc}(\{\mathsf{mpk}_{\mathsf{id}}^{(i,j,1-\rho_i^*[j])}\}, P, 1-R_i[j])$.
- Otherwise, if $i = \iota, j = \gamma$, $\mathsf{ct}^{(i,j,1-\rho_i^*[j])} \leftarrow \mathcal{O}(P, R_i[j], 1-R_i[j])$ and $\mathsf{ct}^{(i,j,\rho_i^*[j])} \leftarrow \mathsf{MA\text{-}ABE.Enc}(\{\mathsf{mpk}_{\mathsf{id}}^{(i,j,\rho_i^*[j])}\}, P, R_i[j])$.
- Otherwise, for $b \in \{0,1\}$, $\mathsf{ct}^{(i,j,b)} \leftarrow \mathsf{MA\text{-}ABE.Enc}(\{\mathsf{mpk}_{\mathsf{id}}^{(i,j,b)}\}, P, R_i[j])$.

Send $\mathsf{CT} := (\widetilde{R}, (\mathsf{ct}^{(i,j,b)})_{i,j,b})$ to $\mathcal{A}$.

Post-challenge Queries. Proceed similar to $\mathbf{Hyb}^{\mathcal{A}}_{1,\iota,\gamma}(1^\lambda)$. If we need to generate secret keys for $(\iota, \gamma, 1-\rho_i^*[j])$-th instantiation, set $\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(\iota,\gamma,1-\rho_i^*[j])} \leftarrow \mathcal{O}(\mathsf{id}, \mathsf{GID}, x)$.

Guess Phase. Output whatever $\mathcal{A}$ outputs.

The probability that $\mathcal{B}$ doesn't abort is $1/q$. And, the running time of $\mathcal{B}$ is polynomial in the running time of $\mathcal{A}$ and λ. If $\mathcal{O}$ is encrypts $R_\iota[\gamma]$, $\mathcal{B}$ behaves as $\mathbf{Hyb}^{\mathcal{A}}_{2,\iota,\gamma}(1^\lambda)$ and if $\mathcal{O}$ encrypts $1 - R_\iota[\gamma]$, $\mathcal{B}$ behaves like $\mathbf{Hyb}^{\mathcal{A}}_{2,\iota,\gamma+1}(1^\lambda)$. Hence, $\mathcal{B}$ is a valid adversary against the security of $\mathsf{MA\text{-}ABE}$ that can break its security with non-negligible advantage, ϵ/q. Thus, $\mathbf{Hyb}^{\mathcal{A}}_{2,\iota,\gamma}(1^\lambda)$ and $\mathbf{Hyb}^{\mathcal{A}}_{2,\iota,\gamma+1}(1^\lambda)$ are computationally indistinguishable. $\square$

Lemma I.4. Assuming the security of $\mathsf{MA\text{-}ABE}$, $\mathbf{Hyb}^{\mathcal{A}}_{2,\iota,L}(1^\lambda)$ and $\mathbf{Hyb}^{\mathcal{A}}_{2,\iota+1,1}(1^\lambda)$ for $\iota \in [n-1]$ are computationally indistinguishable.

Proof. The proof of this lemma is similar to proof of Lemma I.3. $\square$

Lemma I.5. Assuming the security of $\mathsf{MA\text{-}ABE}$, $\mathbf{Hyb}_{2,n,L}^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_3^{\mathcal{A}}(1^\lambda)$ are computationally indistinguishable.

Proof. The proof of this lemma is similar to proof of Lemma I.3. ☐

Lemma I.6. $\mathbf{Hyb}_3^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_4^{\mathcal{A}}(1^\lambda)$ are identically distributed.

Proof. This is apparent from the output distributions. Specifically, for the satisfying query, let $\rho_1, \ldots, \rho_n$ be the random strings given to attacker in secret keys. Using these, the attacker can decrypt 0 if $\rho_{\mathsf{id}}[j] = R_{\mathsf{id}}[j]$ for $j \in [L]$ and 1 otherwise. Thus, the attacker is decrypting $\{(\rho_{\mathsf{id}} \oplus R_{\mathsf{id}})\}_{\mathsf{id}}$ using the secret keys. Using Reveal, we are setting the last ρ_{id} to be $(\bigoplus_{\mathsf{idx} \neq \mathsf{id}} \rho_{\mathsf{idx}} \oplus R_{\mathsf{idx}}) \oplus R_{\mathsf{id}} \oplus \widetilde{R} \oplus m$. By this, we get $\bigoplus_{\mathsf{id} \in [n]} \rho_{\mathsf{id}} \oplus R_{\mathsf{id}} = \widetilde{R} \oplus m$. From this, we can guarantee that the attacker will decrypt m. ☐

J Proofs from Section 11.4

In this section, we provide full proof of Theorem 11.8.

$\mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)$. This is the real experiment with Chal from Definition 10.3.

Setup. $\mathcal{A}$ sends $(n, L, s_{\mathsf{abe}}, s_{\mathsf{fe}}, \ell_1, \ldots, \ell_n)$. Sample $\mathsf{abfe.crs} \leftarrow \mathsf{na1MA\text{-}ABFE.GSetup}(1^\lambda, 1^n, 1^L, 1^{s_{\mathsf{abe}}}, 1^{s_{\mathsf{fe}}})$. Let $G = \mathsf{poly}(\lambda, n, s_{\mathsf{abe}}, s_{\mathsf{fe}}, L)$ be $|\mathsf{na1MA\text{-}ABFE.ct}|$ for these parameters. $\mathsf{abe.crs} \leftarrow \mathsf{ncMA\text{-}ABE.GSetup}(1^\lambda, 1^n, 1^{s_{\mathsf{abe}}}, 1^G)$. Set $\mathsf{CRS} := (\mathsf{abfe.crs}, \mathsf{abe.crs})$. $\forall$ $\mathsf{id} \in [n]$, $(\mathsf{abfe.mpk}_{\mathsf{id}}, \mathsf{abfe.msk}_{\mathsf{id}}) \leftarrow \mathsf{na1MA\text{-}ABFE.ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}})$, $(\mathsf{abe.mpk}_{\mathsf{id}}, \mathsf{abe.msk}_{\mathsf{id}}) \leftarrow \mathsf{ncMA\text{-}ABE.ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}})$. Set $\mathsf{MPK}_{\mathsf{id}} := (\mathsf{abfe.mpk}_{\mathsf{id}}, \mathsf{abe.mpk}_{\mathsf{id}}), \mathsf{MSK}_{\mathsf{id}} := (\mathsf{abfe.msk}_{\mathsf{id}}, \mathsf{abe.msk}_{\mathsf{id}})$ and send $(\mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}})$ to $\mathcal{A}$.

Pre-challenge Queries. $\mathcal{A}$ sends $(\mathsf{id}, \mathsf{GID}, x)$. Sample $\mathsf{abfe.sk}_{\mathsf{id},\mathsf{GID},x,y} \leftarrow \mathsf{na1MA\text{-}ABFE.KGen}(\mathsf{id}, \mathsf{abfe.msk}_{\mathsf{id}}, \{\mathsf{abfe.mpk}_{\mathsf{idx}}\}_{\mathsf{idx}}, \mathsf{GID}, x, y)$ and $\mathsf{abe.sk}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{ncMA\text{-}ABE.KGen}(\mathsf{id}, \mathsf{abe.msk}_{\mathsf{id}}, \{\mathsf{abe.mpk}_{\mathsf{idx}}\}_{\mathsf{idx}}, \mathsf{GID}, x)$. Send $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x,y} := (\mathsf{abfe.sk}_{\mathsf{id},\mathsf{GID},x,y}, \mathsf{abe.sk}_{\mathsf{id},\mathsf{GID},x})$ to $\mathcal{A}$.

Challenge Query. $\mathcal{A}$ sends P, C. $\mathsf{abfe.ct} \leftarrow \mathsf{na1MA\text{-}ABFE.Enc}(\{\mathsf{abfe.mpk}_{\mathsf{id}}\}_{\mathsf{id}}, P, C)$. Now, run $\mathsf{abe.ct} \leftarrow \mathsf{ncMA\text{-}ABE.Enc}(\{\mathsf{abe.mpk}_{\mathsf{id}}\}_{\mathsf{id}}, P, \mathsf{abfe.ct})$. Send $\mathsf{CT} := \mathsf{abe.ct}$ to $\mathcal{A}$.

Post-challenge Queries. $\mathcal{A}$ sends $(\mathsf{id}, \mathsf{GID}, x)$. Sample $\mathsf{abfe.sk}_{\mathsf{id},\mathsf{GID},x,y} \leftarrow \mathsf{na1MA\text{-}ABFE.KGen}(\mathsf{id}, \mathsf{abfe.msk}_{\mathsf{id}}, \{\mathsf{abfe.mpk}_{\mathsf{idx}}\}_{\mathsf{idx}}, \mathsf{GID}, x, y)$ and $\mathsf{abe.sk}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{ncMA\text{-}ABE.KGen}(\mathsf{id}, \mathsf{abe.msk}_{\mathsf{id}}, \{\mathsf{abe.mpk}_{\mathsf{idx}}\}_{\mathsf{idx}}, \mathsf{GID}, x)$. Send $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x,y} := (\mathsf{abfe.sk}_{\mathsf{id},\mathsf{GID},x,y}, \mathsf{abe.sk}_{\mathsf{id},\mathsf{GID},x})$ to $\mathcal{A}$.

Guess Phase. Output whatever $\mathcal{A}$ outputs.

$\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$. In this hybrid, we will fake the $\mathsf{ncMA\text{-}ABE}$ ciphertext and reveal it when the adversary makes a satisfying query.

Setup, Pre-challenge Queries. Same as $\mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)$.

Challenge Query. $\mathcal{A}$ sends P, C. Now,

- If $\mathcal{A}$ made a satisfying query, $\mathsf{abfe.ct} \leftarrow \mathsf{na1MA\text{-}ABFE.Enc}(\{\mathsf{abfe.mpk_{id}}\}_{\mathsf{id}\in[n]}, P, C)$ and $\mathsf{abe.ct} \leftarrow \mathsf{ncMA\text{-}ABE.Enc}(\{\mathsf{abe.mpk_{id}}\}_{\mathsf{id}}, P, \mathsf{abfe.ct})$.
- Otherwise, $(\mathsf{abe.ct}, \mathsf{aux}) \leftarrow \mathsf{ncMA\text{-}ABE.Fake}(\{\mathsf{abe.mpk_{id}}\}_{\mathsf{id}}, P)$.

Send $\mathsf{CT} := \mathsf{abe.ct}$ to $\mathcal{A}$.

Post-challenge Queries. $\mathcal{A}$ sends $(\mathsf{id}, \mathsf{GID}, x)$. Sample $\mathsf{abfe.sk_{id,GID,}}_{x,y} \leftarrow \mathsf{na1MA\text{-}ABFE.KGen}(\mathsf{id}, \mathsf{abfe.msk_{id}}, \{\mathsf{abfe.mpk_{idx}}\}_{\mathsf{idx}}, \mathsf{GID}, x, y)$.

- If this is the last authority for satisfying query, sample $\mathsf{abfe.ct} \leftarrow \mathsf{na1MA\text{-}ABFE.Enc}(\{\mathsf{abfe.mpk_{id}}\}_{\mathsf{id}\in[n]}, P,$ $\mathsf{abe.sk_{id,GID,}}_{x} \leftarrow \mathsf{ncMA\text{-}ABE.Reveal}(\mathsf{id}, \mathsf{abe.mpk_{id}}, \mathsf{abe.msk_{id}}, \{(\mathsf{abe.mpk_{idx}}, \mathsf{abe.}\rho_{\mathsf{idx}})\}_{\mathsf{idx}\neq\mathsf{id}}, \mathsf{GID}, x, \mathsf{aux}, \mathsf{abfe.ct})$.
- Otherwise, $\mathsf{abe.sk_{id,GID,}}_{x} \leftarrow \mathsf{ncMA\text{-}ABE.KGen}(\mathsf{id}, \mathsf{abe.msk_{id}}, \{\mathsf{abe.mpk_{idx}}\}_{\mathsf{idx}}, \mathsf{GID}, x)$.

Send $\mathsf{SK_{id,GID,}}_{x,y} := (\mathsf{abfe.sk_{id,GID,}}_{x,y}, \mathsf{abe.sk_{id,GID,}}_{x})$ to $\mathcal{A}$.

Guess Phase. Output whatever $\mathcal{A}$ outputs.

$\mathbf{Hyb}_2^{\mathcal{A}}(1^\lambda)$. In this hybrid, we will simulate the $\mathsf{na1MA\text{-}ABFE}$ instantiation either while equivocating or in challenge query phase.

Setup. $\mathcal{A}$ sends $(n, L, s_{\mathsf{abe}}, s_{\mathsf{fe}}, \ell_1, \ldots, \ell_n)$. Sample $(\mathsf{abfe.crs}, \{\mathsf{abfe.mpk_{id}}\}_{\mathsf{id}\in[n]}) \leftarrow \mathsf{na1MA\text{-}ABFE.Sim}(1^\lambda, 1^n, 1^L, 1^{s_{\mathsf{abe}}}, 1^{s_{\mathsf{fe}}}, 1^{\ell_1}, \ldots, 1^{\ell_n})$. Let $G = \mathsf{poly}(\lambda, n, s_{\mathsf{abe}}, s_{\mathsf{fe}}, L) = |\mathsf{abfe.ct}|$ for these parameters. $\mathsf{abe.crs} \leftarrow \mathsf{ncMA\text{-}ABE.GSetup}(1^\lambda, 1^n, 1^{s_{\mathsf{abe}}}, 1^G)$. Set $\mathsf{CRS} := (\mathsf{abfe.crs}, \mathsf{abe.crs})$. For each $\mathsf{id} \in [n]$, $(\mathsf{abe.mpk_{id}}, \mathsf{abe.msk_{id}}) \leftarrow \mathsf{ncMA\text{-}ABE.ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}})$. Set $\mathsf{MPK_{id}} := (\mathsf{abfe.mpk_{id}}, \mathsf{abe.mpk_{id}})$, $\mathsf{MSK_{id}} := \mathsf{abe.msk_{id}}$ and send $(\mathsf{CRS}, \{\mathsf{MPK_{id}}\}_{\mathsf{id}\in[n]})$ to $\mathcal{A}$.

Pre-challenge Queries. $\mathcal{A}$ sends $(\mathsf{id}, \mathsf{GID}, x)$. $\mathsf{abfe.sk_{id,GID,}}_{x,y} \leftarrow \mathsf{na1MA\text{-}ABFE.Sim}(\mathsf{id}, \mathsf{GID}, x, y)$, $\mathsf{abe.sk_{id,GID,}}_{x} \leftarrow \mathsf{ncMA\text{-}ABE.KGen}(\mathsf{id}, \mathsf{abe.msk_{id}}, \{\mathsf{abe.mpk_{idx}}\}_{\mathsf{idx}}, \mathsf{GID}, x)$. Send $\mathsf{SK_{id,GID,}}_{x,y} := (\mathsf{abfe.sk_{id,GID,}}_{x,y}, \mathsf{abe.sk_{id,GID,}}_{x})$ to $\mathcal{A}$.

Challenge Query. $\mathcal{A}$ sends P, C. Now,

- If $\mathcal{A}$ made a satisfying query, $\mathsf{abfe.ct} \leftarrow \mathsf{na1MA\text{-}ABFE.Sim}(P, 1^{|C|}, C(y_1, \ldots, y_n))$ and $\mathsf{abe.ct} \leftarrow \mathsf{ncMA\text{-}ABE.Enc}(\{\mathsf{abe.mpk_{id}}\}_{\mathsf{id}}, P, \mathsf{abfe.ct})$.
- Otherwise, $(\mathsf{abe.ct}, \mathsf{aux}) \leftarrow \mathsf{ncMA\text{-}ABE.Fake}(\{\mathsf{abe.mpk_{id}}\}_{\mathsf{id}}, P)$.

Send $\mathsf{CT} := \mathsf{abe.ct}$ to $\mathcal{A}$.

Post-challenge Queries. $\mathcal{A}$ sends $(\mathsf{id}, \mathsf{GID}, x)$. $\mathsf{abfe.sk_{id,GID,}}_{x,y} \leftarrow \mathsf{na1MA\text{-}ABFE.Sim}(\mathsf{id}, \mathsf{GID}, x, y)$.

- If this is the last authority for satisfying query, sample $\mathsf{abfe.ct} \leftarrow \mathsf{na1MA\text{-}ABFE.Sim}(P, 1^{|C|}, C(y_1, \ldots, y_n))$ $\mathsf{abe.sk_{id,GID,}}_{x} \leftarrow \mathsf{ncMA\text{-}ABE.Reveal}(\mathsf{id}, \mathsf{abe.mpk_{id}}, \mathsf{abe.msk_{id}}, \{(\mathsf{abe.mpk_{idx}}, \mathsf{abe.}\rho_{\mathsf{idx}})\}_{\mathsf{idx}\neq\mathsf{id}}, \mathsf{GID}, x, \mathsf{aux}, \mathsf{abfe}$
- Otherwise, $\mathsf{abe.sk_{id,GID,}}_{x} \leftarrow \mathsf{ncMA\text{-}ABE.KGen}(\mathsf{id}, \mathsf{abe.msk_{id}}, \{\mathsf{abe.mpk_{idx}}\}_{\mathsf{idx}}, \mathsf{GID}, x)$.

Send $\mathsf{SK_{id,GID,}}_{x,y} := (\mathsf{abfe.sk_{id,GID,}}_{x,y}, \mathsf{abe.sk_{id,GID,}}_{x})$ to $\mathcal{A}$.

Guess Phase. Output whatever $\mathcal{A}$ outputs.

Lemma J.1. Assuming the security of $\mathsf{ncMA\text{-}ABE}$, $\mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$ are computationally indistinguishable.

Proof. Assume that there exists a PPT adversary that can distinguish between $\mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$ with non-negligible advantage $\epsilon(\lambda)$, i.e,

$$\left|\Pr\left[1 \leftarrow \mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)\right] - \Pr\left[1 \leftarrow \mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)\right]\right| > \epsilon(\lambda)$$

We will construct a reduction adversary $\mathcal{B}$ that can break the security of $\mathsf{ncMA\text{-}ABE}$ with non-negligible probability. The description of $\mathcal{B}^{\mathcal{O}}$ is as follows.

Setup. $\mathcal{A}$ sends $(n, L, s_{\mathsf{abe}}, s_{\mathsf{fe}}, \ell_1, \ldots, \ell_n)$. $\mathsf{abfe.crs} \leftarrow \mathsf{na1MA\text{-}ABFE.GSetup}(1^\lambda, 1^n, 1^L, 1^{s_{\mathsf{abe}}}, 1^{s_{\mathsf{fe}}})$. Let $|\mathsf{abfe.ct}| = G$ for these parameters. Receive $\mathsf{abe.crs}, \{\mathsf{abe.mpk_{id}}\}_{\mathsf{id} \in [n]} \leftarrow \mathcal{O}(1^\lambda, 1^n, 1^{s_{\mathsf{abe}}}, 1^G, 1^{\ell_1}, \ldots, 1^{\ell_n})$. Set $\mathsf{CRS} := (\mathsf{abfe.crs}, \mathsf{abe.crs})$. For each $\mathsf{id} \in [n]$, sample $(\mathsf{abfe.mpk_{id}}, \mathsf{abfe.msk_{id}}) \leftarrow \mathsf{na1MA\text{-}ABFE.ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}})$. Set $\mathsf{MPK_{id}} := (\mathsf{abfe.mpk_{id}}, \mathsf{abe.mpk_{id}})$, $\mathsf{MSK_{id}} := \mathsf{abfe.msk_{id}}$ and send $(\mathsf{CRS}, \{\mathsf{MPK_{id}}\}_{\mathsf{id}})$ to $\mathcal{A}$.

Pre-challenge Queries. $\mathcal{A}$ sends $(\mathsf{id}, \mathsf{GID}, x)$. $\mathsf{abfe.sk}_{\mathsf{id},\mathsf{GID},x,y} \leftarrow \mathsf{na1MA\text{-}ABFE.KGen}(\mathsf{id}, \mathsf{abfe.msk_{id}}, \{\mathsf{abfe.mpk_{idx}}\}_{\mathsf{idx}}, \mathsf{GID}, x, y)$, $\mathsf{abe.sk}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathcal{O}(\mathsf{id}, \mathsf{GID}, x)$. Send $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x,y} := (\mathsf{abfe.sk}_{\mathsf{id},\mathsf{GID},x,y}, \mathsf{abe.sk}_{\mathsf{id},\mathsf{GID},x})$ to $\mathcal{A}$.

Challenge Query. $\mathcal{A}$ sends P, C. $\mathsf{abfe.ct} \leftarrow \mathsf{na1MA\text{-}ABFE.Enc}(\{\mathsf{abfe.mpk_{id}}\}_{\mathsf{id}}, P, C)$ and $\mathsf{abe.ct} \leftarrow \mathcal{O}(P, \mathsf{abfe.ct})$. Send $\mathsf{CT} := \mathsf{abe.ct}$ to $\mathcal{A}$.

Post-challenge Queries. $\mathcal{A}$ sends $(\mathsf{id}, \mathsf{GID}, x)$. $\mathsf{abfe.sk}_{\mathsf{id},\mathsf{GID},x,y} \leftarrow \mathsf{na1MA\text{-}ABFE.KGen}(\mathsf{id}, \mathsf{abfe.msk_{id}}, \{\mathsf{abfe.mpk_{idx}}\}_{\mathsf{idx}}, \mathsf{GID}, x, y)$, $\mathsf{abe.sk}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathcal{O}(\mathsf{id}, \mathsf{GID}, x)$. Send $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x,y} := (\mathsf{abfe.sk}_{\mathsf{id},\mathsf{GID},x,y}, \mathsf{abe.sk}_{\mathsf{id},\mathsf{GID},x})$ to $\mathcal{A}$.

Guess Phase. Output whatever $\mathcal{A}$ outputs.

And, the running time of $\mathcal{B}$ is polynomial in the running time of $\mathcal{A}$ and λ. If $\mathcal{O}$ is $\mathcal{E}_0$ for $\mathsf{ncMA\text{-}ABE}$, $\mathcal{B}$ behaves as $\mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)$ and if $\mathcal{O}$ uses $\mathcal{E}_1$ for $\mathsf{ncMA\text{-}ABE}$, $\mathcal{B}$ behaves like $\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$. Hence, $\mathcal{B}$ is a valid adversary against the security of $\mathsf{ncMA\text{-}ABE}$ that can break its security with non-negligible advantage. Thus, $\mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$ are computationally indistinguishable. $\square$

Lemma J.2. Assuming the security of $\mathsf{na1MA\text{-}ABFE}$, $\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_2^{\mathcal{A}}(1^\lambda)$ are computationally indistinguishable.

Proof. Assume that there exists a PPT adversary that can distinguish between $\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_2^{\mathcal{A}}(1^\lambda)$ with non-negligible advantage $\epsilon(\lambda)$, i.e,

$$\left|\Pr\left[1 \leftarrow \mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)\right] - \Pr\left[1 \leftarrow \mathbf{Hyb}_2^{\mathcal{A}}(1^\lambda)\right]\right| > \epsilon(\lambda)$$

We will construct a reduction adversary $\mathcal{B}$ that can break the security of $\mathsf{na1MA\text{-}ABFE}$ with non-negligible probability. The description of $\mathcal{B}^{\mathcal{O}}$ is as follows.

Setup. $\mathcal{A}$ sends $(n, L, s_{\mathsf{abe}}, s_{\mathsf{fe}}, \ell_1, \ldots, \ell_n)$. Sample $(\mathsf{abfe.crs}, \{\mathsf{abfe.mpk_{id}}\}_{\mathsf{id} \in [n]}) \leftarrow \mathcal{O}(1^\lambda, 1^n, 1^L, 1^{s_{\mathsf{abe}}}, 1^{s_{\mathsf{fe}}}, 1^{\ell_1}, \ldots, 1^{\ell_n})$. Let $G = \mathsf{poly}(\lambda, n, s_{\mathsf{abe}}, s_{\mathsf{fe}}, L) = \mathsf{abfe.ct}$ for these parameters. $\mathsf{abe.crs} \leftarrow \mathsf{ncMA\text{-}ABE.GSetup}(1^\lambda, 1^n, 1^{s_{\mathsf{abe}}}, 1^G)$. Set $\mathsf{CRS} := (\mathsf{abfe.crs}, \mathsf{abe.crs})$. For each $\mathsf{id} \in [n]$, $(\mathsf{abe.mpk_{id}}, \mathsf{abe.msk_{id}}) \leftarrow \mathsf{ncMA\text{-}ABE.ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}})$. Set $\mathsf{MPK_{id}} := (\mathsf{abfe.mpk_{id}}, \mathsf{abe.mpk_{id}})$, $\mathsf{MSK_{id}} := \mathsf{abe.msk_{id}}$ and send $(\mathsf{CRS}, \{\mathsf{MPK_{id}}\}_{\mathsf{id} \in [n]})$ to $\mathcal{A}$.

Pre-challenge Queries. $\mathcal{A}$ sends $(\mathsf{id}, \mathsf{GID}, x)$. Get $\mathsf{abfe.sk}_{\mathsf{id},\mathsf{GID},x,y} \leftarrow \mathcal{O}(\mathsf{id}, \mathsf{GID}, x, y)$, $\mathsf{abe.sk}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{ncMA\text{-}ABE.KGen}(\mathsf{id}, \mathsf{abe.msk}_{\mathsf{id}}, \{\mathsf{abe.mpk}_{\mathsf{idx}}\}_{\mathsf{idx}}, \mathsf{GID}, x)$. Send $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x,y} := (\mathsf{abfe.sk}_{\mathsf{id},\mathsf{GID},x,y}, \mathsf{abe.sk}_{\mathsf{id},\mathsf{GID},x})$ to $\mathcal{A}$.

Challenge Query. $\mathcal{A}$ sends P, C. Now,

- If $\mathcal{A}$ made a satisfying query, $\mathsf{abfe.ct} \leftarrow \mathcal{O}(P, C)$ and $\mathsf{abe.ct} \leftarrow \mathsf{ncMA\text{-}ABE.Enc}(\{\mathsf{abe.mpk}_{\mathsf{id}}\}_{\mathsf{id}}, P, \mathsf{abfe.ct})$.
- Otherwise, $(\mathsf{abe.ct}, \mathsf{aux}) \leftarrow \mathsf{ncMA\text{-}ABE.Fake}(\{\mathsf{abe.mpk}_{\mathsf{id}}\}_{\mathsf{id}}, P)$.

Send $\mathsf{CT} := \mathsf{abe.ct}$ to $\mathcal{A}$.

Post-challenge Queries. $\mathcal{A}$ sends $(\mathsf{id}, \mathsf{GID}, x)$. Sample $\mathsf{abfe.sk}_{\mathsf{id},\mathsf{GID},x,y} \leftarrow \mathcal{O}(\mathsf{id}, \mathsf{GID}, x, y)$.

- If this is the last authority for satisfying query, sample $\mathsf{abfe.ct} \leftarrow \mathcal{O}(P, C)$. $\mathsf{abe.sk}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{ncMA\text{-}ABE.Reveal}(\mathsf{id}, \mathsf{abe.mpk}_{\mathsf{id}}, \mathsf{abe.msk}_{\mathsf{id}},$
$\{(\mathsf{abe.mpk}_{\mathsf{idx}}, \mathsf{abe.}\rho_{\mathsf{idx}})\}_{\mathsf{idx} \neq \mathsf{id}}, \mathsf{GID}, x, \mathsf{aux}, \mathsf{abfe.ct})$.
- Otherwise, $\mathsf{abe.sk}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{ncMA\text{-}ABE.KGen}(\mathsf{id}, \mathsf{abe.msk}_{\mathsf{id}}, \{\mathsf{abe.mpk}_{\mathsf{idx}}\}_{\mathsf{idx}}, \mathsf{GID}, x)$.

Send $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x,y} := (\mathsf{abfe.sk}_{\mathsf{id},\mathsf{GID},x,y}, \mathsf{abe.sk}_{\mathsf{id},\mathsf{GID},x})$ to $\mathcal{A}$.

Guess Phase. Output whatever $\mathcal{A}$ outputs.

Note that we will only query $\mathcal{O}$ once for ciphertext. And, the running time of $\mathcal{B}$ is polynomial in the running time of $\mathcal{A}$ and λ. If $\mathcal{O}$ is an honest challenger for $\mathsf{na1MA\text{-}ABFE}$, $\mathcal{B}$ behaves as $\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$ and if $\mathcal{O}$ is a simulator for $\mathsf{na1MA\text{-}ABFE}$, $\mathcal{B}$ behaves like $\mathbf{Hyb}_2^{\mathcal{A}}(1^\lambda)$. Hence, $\mathcal{B}$ is a valid adversary against the security of $\mathsf{na1MA\text{-}ABFE}$ that can break its security with non-negligible advantage. Thus, $\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_2^{\mathcal{A}}(1^\lambda)$ are computationally indistinguishable. □

K Statically Secure MA-ABE from Witness Encryption

In this section, we will show how to generically transform the selectively secure ABE scheme for general predicates from [GGSW13] which was constructed using witness encryption into a statically secure MA-ABE scheme. Recall that in their construction, secret keys for an attribute x are a "constrain signature" schemes and encryption for a predicate P is a witness encryption that checks $P(x) \overset{?}{=} 1$ and the signature verifies. To argue the security, given a challenge predicate P^* selectively, we can augment the signature to be constrained on $P^*(x) = 0$ for any attribute queried to the challenger.

This constrain signature scheme is constructed using non-interactive witness indistinguishability schemes (NIWI) and a perfectly binding commitment language (COM) that binds to 0. Our observation is that this signature scheme is divisible among multiple authorities. The idea is that we check if the signature from each of the secret keys $\{\mathsf{SK}_{\mathsf{id},x_{\mathsf{id}}}\}_{\mathsf{id}}$ verifies as part of the witness encryption (WE) language. As NIWIs provide statistical soundness, any attacker cannot construct a valid signature commitment that doesn't bind to 0.

However, this construction doesn't provide the same selective security that a single-authority MA-ABE (i.e, a ciphertext-policy ABE scheme) provides. The reason for this is because of the decentralized nature of authorities, in the security argument, we cannot exactly constrain the secret keys to a whole attribute such that $P^*(X) = 0, X = (x_1, \ldots, x_n)$. To see this let's work in the selective setting for a monotone-policy predicate P^*. In this case, we can set $X = (0, \ldots, x_{\mathsf{id}}, \ldots, 0)$ while generating secret key for id-th authority and constrain the signature on this X. By functionality of monotone circuits and admissibility criterion for adversaries for an MA-ABE scheme, this will be a valid way to constrain the signature. However, now we cannot rely on security of WE as there are multiple valid witnesses for the language used in WE. Recall that only if the instance is unsatisfiable, we can rely on security of WE. That is, although for any $\{X_{\mathsf{id}} = (0, \ldots, x_{\mathsf{id}}, \ldots, 0)\}, \{\mathsf{SK}_{\mathsf{id},X_{\mathsf{id}}}\}_{\mathsf{id}}$ are valid secret keys, it could be the case that $P(x_1, \ldots, x_n) = 1$ (regardless of attacker's admissibility criterion). Hence, we cannot exactly argue the security of WE.

However, static security is achievable in this construction. By static we mean that adversary should declare secret key queries for all authorities before receiving the public parameters of the system. Then, as we know the entire attribute for a given GID, we can constrain the secret keys just like [GGSW13]. We can also inculcate authority corruptions by setting the attribute for a corrupted authority to be a random string. We provide this construction to show that our MA-ABFE scheme can be meaningfully instantiated from various assumptions to achieve non-trivial security. We provide the definitions of NIWI, COM, and WE used in our construction in Section K.1 and construct the scheme in Section K.2.

K.1 Preliminaries

Definition K.1 (COM)**.** A COM scheme $(\mathsf{Com}, \mathsf{Verify})$ is said to be a perfectly binding commitment scheme (COM) for message space $\mathcal{M} = \{0,1\}^*$ if it satisfies the following properties.

Completeness. For any $\lambda \in \mathbb{N}$, m, $r = r(\lambda)$, if $\mathsf{com} = \mathsf{Com}(1^\lambda, m; r)$, then $\mathsf{Verify}(\mathsf{com}, m, r) = 1$

Computational Hiding. For any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that,

$$\Pr\left[b \leftarrow \mathcal{A}(\mathsf{com}) \quad : \quad \begin{array}{l} (m_0, m_1) \leftarrow \mathcal{A}(1^\lambda), b \xleftarrow{\$} \{0,1\}, \\ \mathsf{com} \leftarrow \mathsf{Com}(1^\lambda, m_b; r) \end{array} \right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

Perfect Binding. For any $\lambda \in \mathbb{N}$, m, $r = r(\lambda)$, if $\mathsf{com} = \mathsf{Com}(1^\lambda, m; r)$, then for any $m' \neq m$ and any r', $\mathsf{Verify}(\mathsf{com}, m', r') \neq 1$.

Definition K.2 (WE)**.** A WE scheme $(\mathsf{Enc}, \mathsf{Dec})$ is said to be a witness encryption scheme (WE) for language $\mathcal{L} = \{x : \exists\ w, \mathsf{R}(x, w) = 1\}$ and message space $\mathcal{M} \in \{0,1\}^*$ if it satisfies the following properties.

Correctness. For any $\lambda \in \mathbb{N}$, $m \in \mathcal{M}_\lambda$, and $\mathcal{L}$, if $\mathsf{R}(x, w) = 1, \Pr[m = \mathsf{Dec}(\mathsf{CT}, w) : \mathsf{CT} \leftarrow \mathsf{Enc}(1^\lambda, x, m)] = 1$

Semantic Security. For any stateful PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $\forall\ \lambda \in \mathbb{N}, x \notin \mathcal{L}$,

$$\Pr\left[b \leftarrow \mathcal{A}(\mathsf{CT}) \quad : \quad \begin{array}{l} (x, m_0, m_1) \leftarrow \mathcal{A}(1^\lambda), b \xleftarrow{\$} \{0,1\}, \\ \mathsf{CT} \leftarrow \mathsf{Enc}(1^\lambda, x, m_b) \end{array} \right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

Definition K.3 (NIWI)**.** A NIWI scheme $(\mathsf{Prove}, \mathsf{Verify})$ is said to be a non-interactive witness indistinguishability scheme (NIWI) for $\mathcal{L} = \{x : \exists\ w, \mathsf{R}(x, w) = 1\}$ if it satisfies the following properties.

Completeness. For all x, y such that $\mathsf{R}(x, w) = 1$, $\Pr[\mathsf{Verify}(x, \pi) = 1 : \pi \leftarrow \mathsf{Prove}(\mathsf{CRS}, x, w)] = 1$

Statistical Soundness. For any $x \notin \mathcal{L}, \pi \in \{0,1\}^*$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $\Pr[\mathsf{Verify}(x, \pi) = 1] \leq \mathsf{negl}(\lambda)$.

Computational Witness Indistinguishability. For any x, w_1, w_2 such that $\mathsf{R}(x, w_1) = 1$ and $\mathsf{R}(x, w_2) = 1$, $\{\pi_1 : \pi_1 \leftarrow \mathsf{Prove}(x, w_1)\} \approx_c \{\pi_2 : \pi_2 \leftarrow \mathsf{Prove}(x, w_2)\}$.

K.2 Construction

Before diving into the construction, we formally provide the definition of an MA-ABE scheme that is statically secure.

Definition K.4 (Statically secure MA-ABE)**.** We say that an $\mathsf{MA\text{-}ABE}$ scheme $(\mathsf{GSetup}, \mathsf{ASetup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ is said to be statically secure if in security game of Definition 11.1, the admissible adversary $\mathcal{A}$ declares the challenge predicate P^*, corrupted authorities $\mathcal{K} \subset [n]$, the queries $\{\mathsf{GID}_q, \{x_{\mathsf{id},q}\}_{\mathsf{id} \notin \mathcal{K}}\}_q$ for some $q = \mathsf{poly}(\lambda)$ at the beginning of the security game. The adversary also receives $\{\mathsf{MSK}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{K}}$ along with public parameters and secret keys. We modify that admissibility criterion that for any $X \in \mathbf{X}_1 \times \ldots \times \mathbf{X}_{\mathsf{id}}$, $P^*(X) = 0$. Here, $\mathbf{X}_{\mathsf{id}} = \{0,1\}^{\ell}_{\mathsf{id}}$ if id-th authority is corrupted and otherwise the set of all queried attributes to this id.

Construction K.5. We provide the construction of a statically secure MA-ABE scheme for any polynomial-size predicate family and $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda \in \mathbb{N}}$ (Definition K.4) using a NIWI scheme for $\mathcal{L}_{\mathsf{NIWI}}$ (Definition K.3), a WE scheme for $\mathcal{L}_{\mathsf{WE}}$ (Definition K.2), and a perfectly binding commitment scheme (Definition K.1) as follows. The descriptions of $\mathcal{L}_{\mathsf{NIWI}}$ and $\mathcal{L}_{\mathsf{WE}}$ are provided in Figure 1 and Figure 2 respectively. In Figure 1, F is a circuit that checks if $\mathsf{GID}_{\mathsf{idx}}$ is the same as GID for all idx and then executes $P(x_1, \ldots, x_n)$.

Language $\mathcal{L}_{\mathsf{NIWI}}$

Instance: $\mathsf{id}, \mathsf{GID}, x, \mathsf{com}^{(0)}, \mathsf{com}^{(1)}$
Witness: $(r^{(0)}, r^{(1)}, P, \{(\mathsf{idx}, \mathsf{GID}_{\mathsf{idx}}, x_{\mathsf{idx}})\}_{\mathsf{idx} \neq \mathsf{id}})$
Relation: Output 1 if and only if $\phi_1 \vee (\phi_2 \wedge \phi_3) = 1$ where,

- $\phi_1 = \mathsf{COM.Verify}(\mathsf{com}^{(0)}, 0, r^{(0)}) \stackrel{?}{=} 1$
- $\phi_2 = \mathsf{COM.Verify}(\mathsf{com}^{(1)}, P, r^{(1)}) \stackrel{?}{=} 1$
- $\phi_3 = F(\mathsf{id}, \mathsf{GID}, x, \{(\mathsf{idx}, \mathsf{GID}_{\mathsf{idx}}, x_{\mathsf{idx}})\}_{\mathsf{idx} \neq \mathsf{id}}, P) \stackrel{?}{=} 0.$

Figure 1: Description of $\mathcal{L}_{\mathsf{NIWI}}$

<u>$\mathsf{GSetup}(1^\lambda, 1^n, 1^s)$.</u> Output $\mathsf{CRS} = (\lambda, n, s)$.

Language $\mathcal{L}_{\mathsf{WE}}$

Instance: $P, (\mathsf{com}_1^{(0)}, \mathsf{com}_1^{(1)}), \ldots, (\mathsf{com}_n^{(0)}, \mathsf{com}_n^{(1)})$
Witness: $\{(\mathsf{id}, \mathsf{GID}_{\mathsf{id}}, x_{\mathsf{id}}, \pi_{\mathsf{id}})\}_{\mathsf{id}\in[n]}$
Relation: Output 1 if and only if

– $\forall$ $\mathsf{id} \in [n]$, $\mathsf{NIWI.Verify}((\mathsf{id}, \mathsf{GID}_{\mathsf{id}}, x_{\mathsf{id}}, \mathsf{com}_{\mathsf{id}}^{(0)}, \mathsf{com}_{\mathsf{id}}^{(1)}), \pi_{\mathsf{id}}) = 1$

– $\mathsf{GID}_{\mathsf{id}}$ is same for all $\mathsf{id} \in [n]$.

– $P(x_1, \ldots, x_n) = 1$.

Figure 2: Description of $\mathcal{L}_{\mathsf{WE}}$

$\underline{\mathsf{ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}})}$. Sample a commitment for 0 with randomness $r^{(0)}$, $\mathsf{com}^{(0)} \leftarrow \mathsf{COM.Com}(1^\lambda, 0; r^{(0)})$ and commitment for 0^S with randomness $r^{(1)}$. Here, $S = \mathsf{poly}(s)$. $\mathsf{com}^{(1)} \leftarrow \mathsf{Com}(1^\lambda, 0^S; r^{(1)})$. Output $\mathsf{MPK} := (\mathsf{com}^{(0)}, \mathsf{com}^{(1)})$ and $\mathsf{MSK} := (\mathsf{MPK}, r^{(0)})$.

$\underline{\mathsf{KGen}(\mathsf{id}, \mathsf{MSK}_{\mathsf{id}}, \mathsf{GID}, x)}$. Parse $\mathsf{MSK}_{\mathsf{id}}$ as $(\mathsf{com}_{\mathsf{id}}^{(0)}, \mathsf{com}_{\mathsf{id}}^{(1)}, r_{\mathsf{id}}^{(0)})$. Generate a NIWI proof, $\pi_{\mathsf{id}} \leftarrow \mathsf{NIWI.Prove}((\mathsf{id}, \mathsf{GID}, x, \mathsf{com}^{(0)}, \mathsf{com}^{(1)}), (r_{\mathsf{id}}^{(0)}, \bar{0}))$[20]. Output $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} := (\mathsf{id}, \mathsf{GID}, x, \pi_{\mathsf{id}})$.

$\underline{\mathsf{Enc}(\{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}, P, m)}$. Generate a WE ciphertext for instance $(P, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]})$ and message m, $\mathsf{we.ct} \leftarrow \mathsf{WE.Enc}(1^\lambda, (P, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}))$. Output $\mathsf{CT} := (\mathsf{we.ct}, P)$.

$\underline{\mathsf{Dec}(\{\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}\}_{\mathsf{id}\in[n]}, \mathsf{CT})}$. Parse CT as $(\mathsf{we.ct}, P)$ and $\forall$ $\mathsf{id} \in [n]$, $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}$ as $(\mathsf{id}, \mathsf{GID}_{\mathsf{id}}, x_{\mathsf{id}}, \pi_{\mathsf{id}})$. If $\mathsf{GID}_{\mathsf{id}}$ are not same for $\mathsf{id} \in [n]$ or $P(x_1, \ldots, x_n) = 0$, abort and output $\perp$. Otherwise, decrypt using WE, $m' = \mathsf{WE.Dec}(\mathsf{we.ct}, \{(\mathsf{id}, \mathsf{GID}_{\mathsf{id}}, x_{\mathsf{id}}, \pi_{\mathsf{id}})\}_{\mathsf{id}})$. Output m'.

Correctness. The correctness of Construction K.5 follows from the correctness of WE, completeness of NIWI and COM. In particular, from the completeness of COM and NIWI, we have that $\forall$ $\mathsf{id} \in [n]$, $\mathsf{NIWI.Verify}((\mathsf{id}, \mathsf{GID}_{\mathsf{id}}, x_{\mathsf{id}}, \mathsf{com}_{\mathsf{id}}^{(0)}, \mathsf{com}_{\mathsf{id}}^{(1)}), (r_{\mathsf{id}}^{(0)}, \bar{0})) = 1$ as $\mathsf{COM.Verify}(\mathsf{com}_{\mathsf{id}}^{(0)}, 0, r_{\mathsf{id}}^{(0)}) = 1$. Additionally, when all secret keys correspond to the same GID and $P(x_1, \ldots, x_n) = 1$, we have that $(P, (\mathsf{com}_1^{(0)}, \mathsf{com}_1^{(1)}), \ldots, (\mathsf{com}_n^{(0)}, \mathsf{com}_n^{(1)}))$ is a valid instance of $\mathcal{L}_{\mathsf{WE}}$. Hence, by the correctness of WE, $m' = m$.

Theorem K.6. If NIWI is a NIWI scheme for language $\mathcal{L}_{\mathsf{NIWI}}$ (Definition K.3), COM is a perfectly binding commitment scheme (Definition K.1), and WE is a witness encryption scheme for language $\mathcal{L}_{\mathsf{WE}}$ (Definition K.2), then Construction K.5 is a statically secure MA-ABE scheme (Definition K.4) for any polynomial size predicate class and $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda\in\mathbb{N}}$.

Proof. We show that Construction K.5 is a statically secure MA-ABE scheme using the following series of hybrids and lemmas.

$\mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)$. This is the real experiment with honest challenger from Definition K.4 with $b = 0/1$.

Static Queries. $\mathcal{A}$ sends $(n, s, \ell_1, \ldots, \ell_n, \mathcal{K} \subset [n], P, \{\mathsf{GID}_q, \{x_{\mathsf{id},q}\}_{\mathsf{id}\notin\mathcal{K}}\}_q)$ to Chal for some $q = \mathsf{poly}(\lambda)$. Chal sets $\mathsf{CRS} := (\lambda, n, s)$. Chal samples $\forall$ $\mathsf{id} \in [n], \mathsf{com}_{\mathsf{id}}^{(0)} \leftarrow \mathsf{COM.Com}(1^\lambda, 0; r_{\mathsf{id}}^{(0)})$

[20]By $\bar{0}$, we denote an all zero string of sufficient length.

and $\mathsf{com}_{\mathsf{id}}^{(1)} \leftarrow \mathsf{COM.Com}(1^\lambda, 0^S; r_{\mathsf{id}}^{(1)})$. Chal sets $\mathsf{MPK}_{\mathsf{id}} := (\mathsf{com}_{\mathsf{id}}^{(0)}, \mathsf{com}_{\mathsf{id}}^{(1)})$ and $\mathsf{MSK}_{\mathsf{id}} := (\mathsf{MPK}_{\mathsf{id}}, r_{\mathsf{id}}^{(0)})$.

Response. For each q, $\mathsf{id} \notin \mathcal{K}$, Chal samples $\pi_{\mathsf{id},q} \leftarrow \mathsf{NIWI.Prove}((\mathsf{id}, \mathsf{GID}_q, x_{\mathsf{id},q}, \mathsf{MPK}_{\mathsf{id}}), (\mathsf{MSK}_{\mathsf{id}}, \overline{0}))$ and sets $\mathsf{SK}_{\mathsf{id},q} := (\mathsf{id}, \mathsf{GID}_q, x_{\mathsf{id},q}, \pi_{\mathsf{id},q})$. Send $(\mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}}, \{\mathsf{MSK}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{K}}, \{\mathsf{SK}_{\mathsf{id},q}\}_{\mathsf{id},q})$ to $\mathcal{A}$.

Challenge Query. $\mathcal{A}$ sends (m_0, m_1) to Chal. Chal chooses $b \xleftarrow{\$} \{0,1\}$ and using WE, encrypts $\mathsf{we.ct} \leftarrow \mathsf{WE.Enc}(1^\lambda, (P, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}), m_b)$. Send $(\mathsf{we.ct}, P)$ to $\mathcal{A}$.

Guess Phase. Output whatever $\mathcal{A}$ outputs.

$\mathbf{Hyb}_{1,j}^{\mathcal{A}}(1^\lambda)$. For $j \in [n+1]$. In this hybrid, we change the $\mathsf{com}_{\mathsf{id}}^{(1)}$ to be a commitment of P for all honest $\mathsf{id} < j$. The changes are highlighted in red.

Static Queries $\mathcal{A}$ sends $(n, s, \ell_1, \ldots, \ell_n, \mathcal{K} \subset [n], P, \{\mathsf{GID}_q, \{x_{\mathsf{id},q}\}_{\mathsf{id}\notin\mathcal{K}}\}_q)$ for some $q = \mathsf{poly}(\lambda)$. Set $\mathsf{CRS} := (\lambda, n, s)$. Sample for each $\mathsf{id} \in [n], \mathsf{com}_{\mathsf{id}}^{(0)} \leftarrow \mathsf{COM.Com}(1^\lambda, 0; r_{\mathsf{id}}^{(0)})$. If $\mathsf{id} \notin \mathcal{K}, \mathsf{id} < j$, $\mathsf{com}_{\mathsf{id}}^{(1)} \leftarrow \mathsf{COM.Com}(1^\lambda, P; r_{\mathsf{id}}^{(1)})$. Otherwise, $\mathsf{com}_{\mathsf{id}}^{(1)} \leftarrow \mathsf{COM.Com}(1^\lambda, 0^S; r_{\mathsf{id}}^{(1)})$. Set $\mathsf{MPK}_{\mathsf{id}} := (\mathsf{com}_{\mathsf{id}}^{(0)}, \mathsf{com}_{\mathsf{id}}^{(1)})$ and $\mathsf{MSK}_{\mathsf{id}} := (\mathsf{MPK}_{\mathsf{id}}, r_{\mathsf{id}}^{(0)})$.

Response For each q, $\mathsf{id} \notin \mathcal{K}$, sample $\pi_{\mathsf{id},q} \leftarrow \mathsf{NIWI.Prove}(\mathsf{id}, \mathsf{GID}_q, x_{\mathsf{id},q}, \mathsf{MPK}_{\mathsf{id}}), (\mathsf{MSK}_{\mathsf{id}}, \overline{0}))$ and set $\mathsf{SK}_{\mathsf{id},q} := (\mathsf{id}, \mathsf{GID}_q, x_{\mathsf{id},q}, \pi_{\mathsf{id},q})$. Send $(\mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}}, \{\mathsf{MSK}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{K}}, \{\mathsf{SK}_{\mathsf{id},q}\}_{\mathsf{id},q})$ to $\mathcal{A}$.

Challenge Query $\mathcal{A}$ sends (m_0, m_1). Choose $b \xleftarrow{\$} \{0,1\}$ and using WE, encrypt $\mathsf{we.ct} \leftarrow \mathsf{WE.Enc}$ $(1^\lambda, (P, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}), m_b)$. Send $(\mathsf{we.ct}, P)$ to $\mathcal{A}$.

Guess Phase. Output whatever $\mathcal{A}$ outputs.

$\mathbf{Hyb}_{2,\varphi}^{\mathcal{A}}(1^\lambda)$. for $\varphi \in [q+1]$. In this hybrid, we simulate all the NIWI proofs in the key generation for all honest authorities. The changes are highlighted in red.

Static Queries. $\mathcal{A}$ sends $(n, s, \ell_1, \ldots, \ell_n, \mathcal{K} \subset [n], P, \{\mathsf{GID}_q, \{x_{\mathsf{id},q}\}_{\mathsf{id}\notin\mathcal{K}}\}_q)$ for some $q = \mathsf{poly}(\lambda)$. Set $\mathsf{CRS} := (\lambda, n, s)$. Sample for each $\mathsf{id} \in [n], \mathsf{com}_{\mathsf{id}}^{(0)} \leftarrow \mathsf{COM.Com}(1^\lambda, 0; r_{\mathsf{id}}^{(0)})$ and $\mathsf{com}_{\mathsf{id}}^{(1)} \leftarrow \mathsf{COM.Com}(1^\lambda, 0^S; r_{\mathsf{id}}^{(1)})$. Set $\mathsf{MPK}_{\mathsf{id}} := (\mathsf{com}_{\mathsf{id}}^{(0)}, \mathsf{com}_{\mathsf{id}}^{(1)})$ and $\mathsf{MSK}_{\mathsf{id}} := (\mathsf{MPK}_{\mathsf{id}}, r_{\mathsf{id}}^{(0)})$.

Response. For each q, $\mathsf{id} \notin \mathcal{K}$, if $q < \varphi$, $\pi_{\mathsf{id},q} \leftarrow \mathsf{NIWI.Prove}((\mathsf{id}, \mathsf{GID}_q, x_{\mathsf{id},q}, \mathsf{MPK}_{\mathsf{id}}, (\overline{0}, r_{\mathsf{id}}^{(1)}, P, \mathsf{V}))$ where $\mathsf{V} = \{(\mathsf{id}, \mathsf{GID}_q, x_{\mathsf{id},q}) : \mathsf{id} \notin \mathcal{K}\} \cup \{(\mathsf{id}, \mathsf{GID}_q, r_{\mathsf{id}}) : \mathsf{id} \in \mathcal{K}\}$ where $r_{\mathsf{id}} \xleftarrow{\$} \{0,1\}^{\ell_{\mathsf{id}}}$. Otherwise, $\pi_{\mathsf{id},q} \leftarrow \mathsf{NIWI.Prove}((\mathsf{id}, \mathsf{GID}_q, x_{\mathsf{id},q}, \mathsf{MPK}_{\mathsf{id}}), (\mathsf{MSK}_{\mathsf{id}}, \overline{0}))$ and set $\mathsf{SK}_{\mathsf{id},q} := (\mathsf{id}, \mathsf{GID}_q, x_{\mathsf{id},q}, \pi_{\mathsf{id},q})$. Send $(\mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}}, \{\mathsf{MSK}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{K}}, \{\mathsf{SK}_{\mathsf{id},q}\}_{\mathsf{id},q})$ to $\mathcal{A}$.

Challenge Query. $\mathcal{A}$ sends (m_0, m_1). Choose $b \xleftarrow{\$} \{0,1\}$ and using WE, encrypt $\mathsf{we.ct} \leftarrow \mathsf{WE.Enc}(1^\lambda, (P, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}), m_b)$. Send $(\mathsf{we.ct}, P)$ to $\mathcal{A}$.

Guess Phase. Output whatever $\mathcal{A}$ outputs.

$\mathbf{Hyb}_{4,j}^{\mathcal{A}}(1^\lambda)$. For $j \in [n+1]$. In this hybrid, we change the $\mathsf{com}_{\mathsf{id}}^{(0)}$ to be a commitment of 1 for all honest $\mathsf{id} < j$. The changes are highlighted in red.

Static Queries. $\mathcal{A}$ sends $(n, s, \ell_1, \ldots, \ell_n, \mathcal{K} \subset [n], P, \{\mathsf{GID}_q, \{x_{\mathsf{id},q}\}_{\mathsf{id} \notin \mathcal{K}}\}_q)$ for some $q = \mathsf{poly}(\lambda)$. Set $\mathsf{CRS} := (\lambda, n, s)$. Sample for each $\mathsf{id} \in [n]$, if $\mathsf{id} \notin \mathcal{K}, \mathsf{id} < j, \mathsf{com}_{\mathsf{id}}^{(0)} \leftarrow \mathsf{COM.Com}(1^\lambda, 1; r_{\mathsf{id}}^{(0)})$. Otherwise, $\mathsf{com}_{\mathsf{id}}^{(0)} \leftarrow \mathsf{COM.Com}(1^\lambda, 0; r_{\mathsf{id}}^{(0)})$. $\mathsf{com}_{\mathsf{id}}^{(1)} \leftarrow \mathsf{COM.Com}(1^\lambda, 0^S; r_{\mathsf{id}}^{(1)})$. Set $\mathsf{MPK}_{\mathsf{id}} := (\mathsf{com}_{\mathsf{id}}^{(0)}, \mathsf{com}_{\mathsf{id}}^{(1)})$.

Response, Challenge Query, Guess Phase. Same as $\mathbf{Hyb}_{2,q+1}^{\mathcal{A}}(1^\lambda)$.

$\mathbf{Hyb}_5^{\mathcal{A}}(1^\lambda)$. In this hybrid, we change the message in witness encryption to be $0^{|m_0|}$. The changes are highlighted in red.

Static Queries, Response. Same as $\mathbf{Hyb}_{3,n+1}^{\mathcal{A}}(1^\lambda)$.

Challenge Query. $\mathcal{A}$ sends (m_0, m_1). Using WE, encrypts $\mathsf{we.ct} \leftarrow \mathsf{WE.Enc}(1^\lambda, (\widetilde{P}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}}), 0^{|m_0|})$. Send $(\mathsf{we.ct}, P)$ to $\mathcal{A}$.

Guess Phase. Output whatever $\mathcal{A}$ outputs.

Lemma K.7. $\mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_{1,1}^{\mathcal{A}}(1^\lambda)$ are identical.

Proof. As we are not changing any commitments $\mathsf{com}_{\mathsf{id}}^{(1)}$ in $\mathbf{Hyb}_{1,1}^{\mathcal{A}}(1^\lambda)$, these two hybrids are identically distributed. ☐

Lemma K.8. Assuming computational hiding property of COM, $\mathbf{Hyb}_{1,j}^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_{1,j+1}^{\mathcal{A}}(1^\lambda)$ for $j \in [n]$ are computationally indistinguishable.

Proof. W.l.o.g assume that $j \notin \mathcal{K}$. Assume that there exists an adversary $\mathcal{A}$ that can distinguish between $\mathbf{Hyb}_{1,j}^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_{1,j+1}^{\mathcal{A}}(1^\lambda)$ with non-negligible probability $\epsilon(\lambda)$, i.e,

$$\left|\Pr\left[1 \leftarrow \mathbf{Hyb}_{1,j}^{\mathcal{A}}(1^\lambda)\right] - \Pr\left[1 \leftarrow \mathbf{Hyb}_{1,j+1}^{\mathcal{A}}(1^\lambda)\right]\right| > \epsilon(\lambda)$$

We will construct a reduction adversary $\mathcal{B}$ that can break the computational hiding property of COM with non-negligible probability. The description of $\mathcal{B}^{\mathcal{O}}$ is as follows.

Static Queries. $\mathcal{A}$ sends $(n, s, \ell_1, \ldots, \ell_n, \mathcal{K} \subset [n], P, \{\mathsf{GID}_q, \{x_{\mathsf{id},q}\}_{\mathsf{id} \notin \mathcal{K}}\}_q)$ for some $q = \mathsf{poly}(\lambda)$. Set $\mathsf{CRS} := (\lambda, n, s)$. Sample for each $\mathsf{id} \in [n], \mathsf{com}_{\mathsf{id}}^{(0)} \leftarrow \mathsf{COM.Com}(1^\lambda, 0; r_{\mathsf{id}}^{(0)})$. If $\mathsf{id} \notin \mathcal{K}, \mathsf{id} < j$, $\mathsf{com}_{\mathsf{id}}^{(1)} \leftarrow \mathsf{COM.Com}(1^\lambda, P; r_{\mathsf{id}}^{(1)})$. Otherwise if $\mathsf{id} > j$, $\mathsf{com}_{\mathsf{id}}^{(1)} \leftarrow \mathsf{COM.Com}(1^\lambda, 0^S; r_{\mathsf{id}}^{(1)})$. Otherwise, if $\mathsf{id} = j$, set $\mathsf{com}_{\mathsf{id}}^{(1)} \leftarrow \mathcal{O}(0^S, P)$. Set $\mathsf{MPK}_{\mathsf{id}} := (\mathsf{com}_{\mathsf{id}}^{(0)}, \mathsf{com}_{\mathsf{id}}^{(1)})$ and $\mathsf{MSK}_{\mathsf{id}} := (\mathsf{MPK}_{\mathsf{id}}, r_{\mathsf{id}}^{(0)})$.

Response, Challenge Query, Guess Phase. Similar to $\mathbf{Hyb}_{1,j+1}^{\mathcal{A}}(1^\lambda)$.

As we can see, the running time of $\mathcal{B}$ is polynomial in the running time of $\mathcal{A}$ and λ. If $\mathcal{O}$ returns a commitment of P, $\mathcal{B}$ behaves like $\mathbf{Hyb}^{\mathcal{A}}_{1,j+1}(1^\lambda)$ and if $\mathcal{O}$ returns a commitment of 0^S, $\mathcal{B}$ behaves like $\mathbf{Hyb}^{\mathcal{A}}_{1,j}(1^\lambda)$. Hence, $\mathcal{B}$ is a valid adversary against the computational hiding property of COM that can break its security with probability $\epsilon(\lambda)$. Thus $\mathbf{Hyb}^{\mathcal{A}}_{1,j}(1^\lambda)$ and $\mathbf{Hyb}^{\mathcal{A}}_{1,j+1}(1^\lambda)$ are computationally indistinguishable. □

Lemma K.9. $\mathbf{Hyb}^{\mathcal{A}}_{1,n+1}(1^\lambda)$ and $\mathbf{Hyb}^{\mathcal{A}}_{2,1}(1^\lambda)$ are identical.

Proof. As we are not changing any NIWI proofs in $\mathbf{Hyb}^{\mathcal{A}}_{2,1}(1^\lambda)$, these two hybrids are identically distributed. □

Lemma K.10. Assuming the witness indistinguishability property of NIWI, $\mathbf{Hyb}^{\mathcal{A}}_{2,\varphi}(1^\lambda)$, $\mathbf{Hyb}^{\mathcal{A}}_{2,\varphi+1}(1^\lambda)$ for $\varphi \in [q]$ are computationally indistinguishable.

Proof. Assume that there exists an adversary $\mathcal{A}$ that can distinguish between $\mathbf{Hyb}^{\mathcal{A}}_{2,\varphi}(1^\lambda)$ and $\mathbf{Hyb}^{\mathcal{A}}_{2,\varphi+1}(1^\lambda)$ with non-negligible probability $\epsilon(\lambda)$, i.e,

$$\left|\Pr\left[1 \leftarrow \mathbf{Hyb}^{\mathcal{A}}_{2,\varphi}(1^\lambda)\right] - \Pr\left[1 \leftarrow \mathbf{Hyb}^{\mathcal{A}}_{2,\varphi+1}(1^\lambda)\right]\right| > \epsilon(\lambda)$$

We will construct a reduction adversary $\mathcal{B}$ that can break the witness indistinguishability property of NIWI with non-negligible probability. The description of $\mathcal{B}^{\mathcal{O}}$ is as follows.

Static Queries. $\mathcal{A}$ sends $(n, s, \ell_1, \ldots, \ell_n, \mathcal{K} \subset [n], P, \{\mathsf{GID}_q, \{x_{\mathsf{id},q}\}_{\mathsf{id}\notin\mathcal{K}}\}_q)$ for some $q = \mathsf{poly}(\lambda)$. Set $\mathsf{CRS} := (\lambda, n, s)$. Sample for each $\mathsf{id} \in [n], \mathsf{com}^{(0)}_{\mathsf{id}} \leftarrow \mathsf{COM.Com}(1^\lambda, 0; r^{(0)}_{\mathsf{id}})$ and $\mathsf{com}^{(1)}_{\mathsf{id}} \leftarrow \mathsf{COM.Com}(1^\lambda, P; r^{(1)}_{\mathsf{id}})$. Set $\mathsf{MPK}_{\mathsf{id}} := (\mathsf{com}^{(0)}_{\mathsf{id}}, \mathsf{com}^{(1)}_{\mathsf{id}})$ and $\mathsf{MSK}_{\mathsf{id}} := (\mathsf{MPK}_{\mathsf{id}}, r^{(0)}_{\mathsf{id}})$.

Response. For each q, $\mathsf{id} \notin \mathcal{K}, q < \varphi$, sample $\pi_{\mathsf{id},q} \leftarrow \mathsf{NIWI.Prove}((\mathsf{id}, \mathsf{GID}_q, x_{\mathsf{id},q}, \mathsf{MPK}_{\mathsf{id}}), (r^{(1)}_{\mathsf{id}}, P, \mathsf{V}))$ where V is as defined in $\mathbf{Hyb}^{\mathcal{A}}_{2,\varphi}(1^\lambda)$ and $q > \varphi$, $\pi_{\mathsf{id},q} \leftarrow \mathsf{NIWI.Prove}((\mathsf{id}, \mathsf{GID}_q, x_{\mathsf{id},q}, \mathsf{MPK}_{\mathsf{id}}), (r^{(0)}_{\mathsf{id}}, \overline{0}))$. If $q = \varphi$, $\pi_{\mathsf{id},q} \leftarrow \mathcal{O}((\mathsf{id}, \mathsf{GID}_q, x_{\mathsf{id},q}, \mathsf{MPK}_{\mathsf{id}}), (r^{(1)}_{\mathsf{id}}, P, \mathsf{V}), (r^{(0)}_{\mathsf{id}}, \overline{0}))$. Set $\mathsf{SK}_{\mathsf{id},q} := (\mathsf{id}, \mathsf{GID}_q, x_{\mathsf{id},q}, \pi_{\mathsf{id},q})$. Send $(\mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}, \{\mathsf{MSK}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{K}}, \{\mathsf{SK}_{\mathsf{id},q}\}_{\mathsf{id},q})$ to $\mathcal{A}$.

Challenge Query. $\mathcal{A}$ sends (m_0, m_1). Choose $b \xleftarrow{\$} \{0,1\}$ and using WE, encrypt $\mathsf{we.ct} \leftarrow \mathsf{WE.Enc}(1^\lambda, (P, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}), m_b)$. Send $(\mathsf{we.ct}, P)$ to $\mathcal{A}$.

Guess Phase. Output whatever $\mathcal{A}$ outputs.

As we can see, the running time of $\mathcal{B}$ is polynomial in the running time of $\mathcal{A}$ and λ. If $\mathcal{O}$ uses $(r^{(1)}_{\mathsf{id}}, P, \mathsf{V})$, $\mathcal{B}$ behaves like $\mathbf{Hyb}^{\mathcal{A}}_{2,\varphi+1}(1^\lambda)$ and if $\mathcal{O}$ uses $(r^{(0)}_{\mathsf{id}}, \overline{0})$, $\mathcal{B}$ behaves like $\mathbf{Hyb}^{\mathcal{A}}_{2,\varphi}(1^\lambda)$. Hence, $\mathcal{B}$ is a valid adversary against witness indistinguishability property of NIWI that can break its security with probability $\epsilon(\lambda)$. Thus $\mathbf{Hyb}^{\mathcal{A}}_{2,\varphi}(1^\lambda)$ and $\mathbf{Hyb}^{\mathcal{A}}_{2,\varphi+1}(1^\lambda)$ are computationally indistinguishable. □

Lemma K.11. $\mathbf{Hyb}^{\mathcal{A}}_{2,q+1}(1^\lambda)$ and $\mathbf{Hyb}^{\mathcal{A}}_{3,1}(1^\lambda)$ are identical.

Proof. As we are not changing any commitments $\mathsf{com}^{(0)}_{\mathsf{id}}$ in $\mathbf{Hyb}^{\mathcal{A}}_{4,1}(1^\lambda)$, these two hybrids are identically distributed. □

Lemma K.12. Assuming the computational hiding property of COM, $\mathbf{Hyb}^{\mathcal{A}}_{3,j}(1^\lambda)$ and $\mathbf{Hyb}^{\mathcal{A}}_{3,j+1}(1^\lambda)$ for $j \in [n]$ are computationally indistinguishable.

Proof. W.l.o.g assume j is honest. As we are not revealing $r_j^{(0)}$ for honest j in these hybrids, computational hiding holds and the proof of this lemma is similar to proof of Lemma K.8. □

Lemma K.13. Assuming the security of WE, $\mathbf{Hyb}^{\mathcal{A}}_{3,n+1}(1^\lambda)$ and $\mathbf{Hyb}^{\mathcal{A}}_{4}(1^\lambda)$ are computationally indistinguishable.

Proof. Consider an $\mathsf{id}^* \notin \mathcal{K}$ and that this is the only honest adversary. For any secret key generated for id^*, we have that $\mathsf{COM.Verify}(\mathsf{com}^{(0)}_{\mathsf{id}^*}, 0, r^{(0)}_{\mathsf{id}^*}) = 0$ as we are generating $\mathsf{com}^{(0)}_{\mathsf{id}^*}$ as commitment of 1 in both hybrids and by the perfect binding property of COM. If $\mathsf{NIWI.Verify}((\mathsf{id}^*, \mathsf{GID}_q, x_{\mathsf{id}^*,q}, \mathsf{com}^{(0)}_{\mathsf{id}^*}, \mathsf{com}^{(1)}_{\mathsf{id}^*}), \pi_{\mathsf{id}^*,q}) = 1$, this implies that with high probability ($1 - \nu(\lambda)$, ν is soundness error of NIWI), $P(*, \dots, x_{\mathsf{id}^*,q}, \dots, *) = 0$. For any admissible adversary, it is the case that for any value where the id^*-th attribute is $x_{\mathsf{id}^*,q}$, P will not be satisfied. In addition, we also have that $\mathsf{com}^{(1)}_{\mathsf{id}^*}$ is perfectly binding to P. Thus, with $(1 - \nu(\lambda))$ probability, for any $X_q = (x_{1,q}, \dots, x_{\mathsf{id}^*,q}, \dots, x_{n,q})$ such that $P(X_q) = 1$, the NIWI proof will *not* accept. Thus, the instance for WE language is not in the language and we can rely on its security.

Assume that there exists an adversary $\mathcal{A}$ that can distinguish between $\mathbf{Hyb}^{\mathcal{A}}_{3,n+1}(1^\lambda)$ and $\mathbf{Hyb}^{\mathcal{A}}_{4}(1^\lambda)$ with non-negligible probability $\epsilon(\lambda)$, i.e,

$$\left|\Pr\left[1 \leftarrow \mathbf{Hyb}^{\mathcal{A}}_{3,n+1}(1^\lambda)\right] - \Pr\left[1 \leftarrow \mathbf{Hyb}^{\mathcal{A}}_{4}(1^\lambda)\right]\right| > \epsilon(\lambda)$$

We will construct a reduction adversary $\mathcal{B}$ that can break the security of WE with non-negligible probability. The description of $\mathcal{B}^{\mathcal{O}}$ is as follows.

Static Queries. $\mathcal{A}$ sends $(n, s, \ell_1, \dots, \ell_n, \mathcal{K} \subset [n], P, \{\mathsf{GID}_q, \{x_{\mathsf{id},q}\}_{\mathsf{id} \notin \mathcal{K}}\}_q)$ for some $q = \mathsf{poly}(\lambda)$. Set $\mathsf{CRS} := (\lambda, n, s)$. Sample for each $\mathsf{id} \in [n]$, $\mathsf{com}^{(0)}_{\mathsf{id}} \leftarrow \mathsf{COM.Com}(1^\lambda, 0; r^{(0)}_{\mathsf{id}})$ and $\mathsf{com}^{(1)}_{\mathsf{id}} \leftarrow \mathsf{COM.Com}(1^\lambda, P; r^{(1)}_{\mathsf{id}})$. Set $\mathsf{MPK}_{\mathsf{id}} := (\mathsf{com}^{(0)}_{\mathsf{id}}, \mathsf{com}^{(1)}_{\mathsf{id}})$.

Response. For each q, $\mathsf{id} \notin \mathcal{K}$, sample $\pi_{\mathsf{id},q} \leftarrow \mathsf{NIWI.Prove}((\mathsf{id}, \mathsf{GID}_q, x_{\mathsf{id},q}, \mathsf{MPK}_{\mathsf{id}}), (r^{(1)}_{\mathsf{id}}, P, \mathsf{V}))$ where V is as defined in $\mathbf{Hyb}^{\mathcal{A}}_{3,n+1}(1^\lambda)$ and set $\mathsf{SK}_{\mathsf{id},q} := (\mathsf{id}, \mathsf{GID}_q, x_{\mathsf{id},q}, \pi_{\mathsf{id},q})$. Send $(\mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}}, \{\mathsf{MSK}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{K}}, \{\mathsf{SK}_{\mathsf{id},q}\}_{\mathsf{id},q})$ to $\mathcal{A}$.

Challenge Query. $\mathcal{A}$ sends (m_0, m_1). Set $\mathsf{we.ct} \leftarrow \mathcal{O}((P, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id} \in [n]}), m_b, 0^{|m_b|})$. Send $(\mathsf{we.ct}, P)$ to $\mathcal{A}$.

Guess Phase. $\mathcal{A}$ outputs b'. If $b' = b$, output 0. Otherwise, output 1.

As we can see, the running time of $\mathcal{B}$ is polynomial in the running time of $\mathcal{A}$ and λ. If $\mathcal{O}$ returns an encryption of m_b, $\mathcal{B}$ behaves like $\mathbf{Hyb}^{\mathcal{A}}_{3,n+1}(1^\lambda)$ and if $\mathcal{O}$ is returns an encryption of $0^{|m_b|}$, $\mathcal{B}$ behaves like $\mathbf{Hyb}^{\mathcal{A}}_{4}(1^\lambda)$. Hence, $\mathcal{B}$ is a valid adversary against the security of WE that can break its security with non-negligible probability. Thus $\mathbf{Hyb}^{\mathcal{A}}_{3,n+1}(1^\lambda)$ and $\mathbf{Hyb}^{\mathcal{A}}_{4}(1^\lambda)$ are computationally indistinguishable. □

□