# Decentralized Multi-Authority Functional Encryption with Malicious Authorities from Standard Assumptions

Rishab Goyal
UW-Madison*

Saikumar Yadugiri
UW-Madison†

**Abstract**

We design decentralized multi-authority functional encryption (MAFE) for $\mathsf{P}/\mathsf{Poly}$ circuits from minimal assumptions in both the bounded-collusion and fully collusion-resistant models, i.e., from public-key encryption and from functional encryption, respectively. Prior constructions required sub-exponentially hard public-key encryption with random oracles (bounded collusion) or sub-exponentially-hard obfuscation (collusion-resistant).

Under the additional assumption of SXDH or LWE, our constructions remain secure against an adversary that corrupts any $k$ of the $n$ local authorities, provided $\binom{n}{k} = \mathsf{poly}(\lambda)$, covering, for example, $k = O(1)$ and $n = \mathsf{poly}(\lambda)$ or $k = O(\log \lambda)$ and $n = O(\log \lambda)$. Complementing this, we show that any MAFE scheme that tolerates corruption of more than $\chi = O(\log \lambda)$ authorities and remains non-trivially compact (i.e., parameter sizes scale sub-linearly in $2^\chi$) implies indistinguishability obfuscation. Thus, our scheme is essentially optimal across a wide parameter regime.

Along the way, we introduce two abstractions, MAFE with *synchronous setup* and *synchronous key generation* that provide a path to constructing multi-authority encryption schemes from their centralized counterparts, and may be useful for designing other multi-authority encryption systems.

*Email: rishab@cs.wisc.edu.
†Email: saikumar@cs.wisc.edu.

# Contents


**1 Introduction** **1**
- 1.1 Our Contributions . . . 2
- 1.2 Applications . . . 3

**2 Technical Overview** **5**
- 2.1 MAFE with Malicious Authorities . . . 6
- 2.2 Bounded Collusion MAFE with Semi-Honest Authorities from PKE . . . 8
- 2.3 Collusion Resistant MAFE with Semi-Honest Authorities from FE . . . 10

**3 Preliminaries** **14**
- 3.1 Pseudorandom Functions . . . 14
- 3.2 Secret-Key Encryption with Pseudorandom Ciphertexts . . . 14
- 3.3 Garbled Ciruits . . . 15
- 3.4 (Bounded-Collusion) Functional Encryption . . . 16
- 3.5 Function Secret Sharing with Strong Security . . . 17

**4 MAFE from Public-Key Encryption** **18**
- 4.1 Definition . . . 18
- 4.2 MAFE with Synchronous Setup . . . 20
- 4.3 MAFE from ssMAFE . . . 22

**5 Bounded Collusion MAFE with Malicious Authorities** **24**
- 5.1 Definition . . . 24
- 5.2 MAFE with Adaptive Corruptions from FSS . . . 24

**6 Collusion Resistant MAFE from Functional Encryption** **27**
- 6.1 Definition . . . 27
- 6.2 Selectively Secure MAFE with Synchronous Key Generation . . . 27
- 6.3 Private-Key One-Ciphertext MAFE . . . 30
- 6.4 Adaptively Secure MAFE with Synchronous Key Generation . . . 31
- 6.5 Vector Non-Committing Encryption . . . 33
- 6.6 MAFE from FE . . . 34

**7 Collusion Resistant MAFE with Malicious Authorities** **36**
- 7.1 Definition . . . 36
- 7.2 Construction . . . 36

**8 Lower Bounds on MAFE with Authority Corruptions** **39**
- 8.1 Preliminaries . . . 39
- 8.2 Definitions . . . 40
- 8.3 XiO from fst1MAFE . . . 41
- 8.4 Weakly Size-Succinct FE from stMAFE . . . 42

**A Relevant Material from [GY24]** **51**
A.1 Distributed Client-Server Framework . . . . . 51
A.2 Augmented Statistical Lemmas . . . . . 53

**B Proofs from Section 4** **53**

**C Proofs of Theorem 5.4** **58**

**D Generic Compilers for Authority Corruptions** **63**

**E Proofs from Section 6.2** **65**

**F One-Ciphertext Private-Key MAFE** **71**
F.1 One-Key One-Challenge SIM-secure MCFE for P/Poly . . . . . 72
F.2 One-Key Many-Challenge MCFE for P/Poly . . . . . 80
F.3 One-CT Private-Key MAFE . . . . . 81

**G Proofs from Section 6.4** **83**

**H Vector Non-Committing Encryption from IBE** **89**
H.1 Preliminaries . . . . . 89
H.2 Non-committing Encryption . . . . . 89
H.3 Identity-Based Encryption . . . . . 90
H.4 Construction . . . . . 91

**I Proofs from Section 6.6** **96**

**J Proofs from Section 7.2** **99**

# 1 Introduction

Multi-authority encryption model [Cha07, CC09, LW11, BCG+17] is a well-established model to shield advanced encryption systems from the key escrow problem [Rog15]. It decentralizes the job of the *singular trusted* master key authority, in monolithic systems such as attribute-based (ABE) and functional encryption (FE) [SW05, GPSW06, BW07, KSW08, BSW11], to a group of asynchronized, non-interacting, and mistrusting local key authorities. For example, entities such as the Department of Motor Vehicles (DMV), Department of Homeland Security (DHS), Internal Revenue Service (IRS) can come together to provide fine-grained access to encrypted data, while shielding from adversarial corruption of one or more such entities. Decentralization to multiple key authorities diversifies the trusted computing base and reduces the risk due to corruption of, or malicious behavior by, one or more key authorities.

Over the last two decades, a lot of research has focused on designing expressive multi-authority functional encryption systems given their vast applications across a huge swathe of cryptography. Despite great interest and significant effort, we have just a handful of multi-authority functional encryption systems. These can be broadly categorized as follows:

**Sub-exponentially-hard obfuscation.** Using the heavy hammer of sub-exponentially-hard indistinguishability obfuscation [BGI+12, GGH+16] and one-way functions, we have multi-authority functional encryption (MAFE) for all $\mathsf{P/Poly}$ circuits [BCG+17].

**Semi-Honest authorities and bounded collusion.** Second line of work designs MAFE/ABE [GGL24, GY24] for all $\mathsf{P/Poly}$ circuits, but with significantly weaker guarantees. The state-of-the-art is MAFE for $\mathsf{P/Poly}$ circuits from sub-exponentially-hard public-key encryption (PKE) in the random oracle model (ROM). However, its security only holds as long as *all* local key authorities are 'semi-honest' and only an a-priori bounded number of keys get corrupted. *Requiring every local authority to be semi-honest goes against the original motivation.*

This has left significant gaps in our current understanding of multi-authority encryption systems. In particular, outside of sub-exponentially-hard obfuscation, we do not have any multi-authority functional encryption system, beyond all-or-nothing encryption for $\mathsf{NC}^1$ predicates (MA-ABE), that is secure in presence of malicious authorities. This prompted the following question central to this work.

(Q1) *Can we design MAFE for* $\mathsf{P/Poly}$ *circuits secure against malicious authorities without obfuscation?*

The above question is completely open in, both, the collusion-resistant and bounded-collusion model. Unfortunately, the state of affairs is not much better even when we additionally restrict all local authorities to be always semi-honest. Consider the following natural question–

(Q2) *If all local authorities are semi-honest, can we design MAFE for* $\mathsf{P/Poly}$ *circuits from minimal assumptions?*

Unfortunately, irrespective of the collusion model (collusion-resistant or bounded-collusion), the above question is still unresolved. This is very much unlike the traditional single-authority model where analogous questions have long been resolved [SS10, GVW12, AJ15, BV15, AV19]. The fact that above questions are still open highlights the difficulty in designing MAFE, irrespective of whether the local authorities are semi-honest or malicious.

### 1.1 Our Contributions

We answer these questions, thereby addressing the fundamental gaps in our current understanding of multi-authority functional encryption.

**Answering Q1.** We answer Q1 and design MAFE for $\mathsf{P/Poly}$ circuits, where an adversary can corrupt any $k$ out of $n$ local authorities as long as $\binom{n}{k} = \mathsf{poly}(\lambda)$. We give two constructions. Our first construction is a bounded-collusion MAFE scheme under the hardness of either SXDH or LWE, and our second construction is a fully-collusion-resistant MAFE under the hardness of poly-hard (single-authority) functional encryption and either SXDH or LWE. Additionally, we show that any MAFE scheme, where an adversary can corrupt more than $\chi = O(\log \lambda)$ authority bits [1], yet parameter sizes remain non-trivially compact,[2] implies obfuscation. Together with our positive results, this shows that our constructions are essentially optimal for a wide range of parameters.

On the technical front, we show a generic compiler that lifts semi-honest MAFE to malicious-authority MAFE in the bounded-collusion model, and a similar transformation in the collusion-resistant model with bounded pre-challenge queries. This generic compiler requires the additional assumption of hardness of $\mathsf{SXDH}$ or $\mathsf{LWE}$. Our results for Q1 are summarized below.

**Informal Theorem 1.** There exists an MAFE scheme for $\mathsf{P/Poly}$ circuits that supports $n$ authorities that could behave maliciously in the *standard model* with

- *Bounded collusions* assuming the polynomial hardness of $\mathsf{SXDH}$ or $\mathsf{LWE}$ assumptions.
- *Collusion resistance* assuming the polynomial hardness of (single-authority) FE and $\mathsf{SXDH}$ or $\mathsf{LWE}$ assumptions where the number of pre-challenge queries are declared a-priori.

In both these results, we achieve full security where both partial secret key queries and $k$-many authority corruption queries can be made in an adaptive manner as long as $\binom{n}{k} = \mathsf{poly}(\lambda)$.

**Answering Q2.** We also completely resolve Q2 and design MAFE for $\mathsf{P/Poly}$ circuits from minimal assumptions in both, the collusion-resistant and bounded-collusion models as long as the authorities act semi-honestly. On the technical front, we provide new security definitions for (single-authority) FE systems that generically yields a multi-authority FE system. In the bounded collusion model, we consider '*synchronous setup*' where a central authority samples master secret keys for each authority. Although this notion appears contradictory to the task of decentralizing in context of multi-authority systems, we show later in Section 2.2, it is a very useful interim object. We show that any MAFE with synchronous setup can be generically compiled into a standard MAFE without it, via a delegation layer based on single-authority bounded-collusion FE. In the collusion resistant model, we consider '*synchronous key generation*' where each authority must synchronously, yet independently sample partial secret keys for a user. We similarly show that MAFE with synchronous key generation can be lifted to standard MAFE using identity-based encryption. We believe that these abstractions can help in designing other kinds of multi-authority encryption systems. Our results for Q2 are summarized below.

**Informal Theorem 2.** There exists an MAFE scheme for $\mathsf{P/Poly}$ circuits that supports an arbitrary number of authorities that behave semi-honestly in the *standard model* with

[1] As we explain later, the phrase "authority bits" refers to the total number of attribute bits controlled by a specific group of authorities.

[2] Specifically, if parameter sizes grow with $2^{\chi(1-\xi)}$, where constant $\xi > 0$.

- *Bounded collusions* assuming the polynomial hardness of PKE (minimal assumption).
- *Collusion resistance* assuming the polynomial hardness of FE (minimal assumption).

In both these results, we achieve adaptive security where partial secret key queries can be made adaptively.

We compare our results with state-of-the-art for Q1 and Q2 in Tables 1 and 2 respectively.

| Reference | Assumption | # of Authorities | Malicious | No RO |
|---|---|---|---|---|
| [GVW12] | DDH/ LWE | 1 | ✗ | ✓ |
| [AV19] | PKE | 1 | ✗ | ✓ |
| [GY24] | Sub-exp PKE | poly($\lambda$) | ✗ | ✗ |
| Thm 4.10 | PKE | poly($\lambda$) | ✗ | ✓ |
| Thm 5.4 | LWE/SXDH | poly($\lambda$) | ✓ | ✓ |

Table 1: Comparison of our results with prior works on bounded collusion FE for P/Poly circuits and adaptive security. By **Malicious**, we denote the presence of malicious authorities and **No RO** indicates the absence of random oracles in the system.

| Reference | Assumption | # of Authorities | Malicious | No RO |
|---|---|---|---|---|
| [Wat15] | i$\mathcal{O}$ | 1 | ✗ | ✓ |
| [JLS21, JLS22] | Sub-exp(LPN, SXDH, PRG in $\mathsf{NC}^0$) | 1 | ✗ | ✓ |
| [RVV24] | Sub-exp(LPN, SXDH, Sparse LPN) | 1 | ✗ | ✓ |
| [BCG+17] | Sub-exp i$\mathcal{O}$+ OWF | poly($\lambda$) | ✓ | ✓ |
| Thm 6.11 | FE | poly($\lambda$) | ✗ | ✓ |
| Thm 7.3 | FE+ LWE/SXDH | poly($\lambda$) | ✓ | ✓ |

Table 2: Comparison of our results with prior works on collusion-resistant FE for P/Poly circuits and adaptive security. By **Malicious**, we denote the presence of malicious authorities and **No RO** indicates the absence of random oracles in the system.

### 1.2 Applications

Beyond the traditional applications as discussed in many prior works [Cha07, CC09, LW11, BCG+17], MAFE schemes could be very useful in a variety of new contexts such as decentralized tracing and threshold cryptography. We briefly sketch some notions below and leave definitional and efficiency constraints for further exploration.

**I: Multi-authority traitor tracing.** Traditional traitor tracing (TT) systems [CFN94] rely on a central authority to manage master keys and identify leaked decoder keys, creating a single point of failure. A compromised authority can decrypt all ciphertexts and lose the ability to trace corrupt users trustfully. Recent works [Luo24, BLM+24, CMMS25] attempt to distribute this trust but introduce substantial overhead and synchronization requirements, often necessitating a full system update whenever a new user joins. We propose *Multi-Authority Traitor Tracing* (MA-TT), where the single trusted authority is replaced by a set of asynchronous, non-interacting, mutually mistrusting local authorities, so that semantic security and traceability hold even if a subset of

authorities is malicious. Technically, MA-TT can be obtained by generalizing the Private Linear Broadcast Encryption paradigm [BSW06] to the multi-authority setting. Following [GKW18], a multi-authority mixed FE can be built by combining our bounded-collusion MAFE with lockable obfuscation [GKW17, WZ17], and integrating it with a collusion-resistant MA-ABE construction yields a robust MA-TT system. In two subsequent works Goyal-Snyder-Yadugiri and Datta-Schädlich-Tairi [GSY26, DST26] formulated and constructed MA-TT from standard assumptions.

**II: Beyond threshold policies in distributed encryption with silent setup.** Threshold encryption [Des88, Fra90, DF90] typically requires a central trusted or interactive setup phase to distribute secret key shares among $n$ parties. Recent works [GKPW24, DJWW25] have challenged this constraint with "silent setup" encryption for thresholds and monotone policies, where setup is non-interactive. Our MAFE (or its weakening to MA-ABE) generically realizes distributed encryption for *arbitrary* monotone policies with silent setup. Each party acts as an independent authority, asynchronously publishing public parameters to a bulletin board, and an encryptor can dynamically select a set of parties and an access structure $\phi$ such that only authorized subsets can jointly decrypt. A unique feature of this approach is natural support for *hierarchical key delegation*. Standard distributed encryption systems lack hierarchy. A participating authority must either share its full secret key or share nothing. Our framework lets each authority issue delegated keys for specific identities ($\mathsf{GID}$s) without exposing its master key. For example, a consortium of universities can each generate partial keys for their PIs and collaborators that seamlessly combine with partial keys from other institutions (e.g., to support a joint research center spanning multiple universities), offering a strictly more expressive alternative to existing silent setup schemes.

**III: Beyond threshold policies in distributed signatures.** Decentralization is also fundamental to signatures, where signing capabilities are distributed such that a valid signature can only be produced by a specific subset of users [Des88, Fra90, DF90]. Our MAFE (or its weakening to MA-ABE) yields distributed signatures for far more general policies than standard threshold signatures, with the added benefits of silent setup and asynchronous signing. Generalizing the folklore Naor transformation [BF01, FGH+17] to the multi-authority setting, we can treat each signer as an authority. To sign a message $\mu$, a signer issues a secret key for $\mathsf{GID} := \mu$ with attribute $x = 1$, and a verifier encrypts a random challenge under a policy and checks whether the aggregated signatures decrypt it. Crucially, this enables a property we call *predicated verification*. In this notion, each signer associates an attribute $x_i$ with their signature, and verification succeeds only if the signers' attributes satisfy a verifier-specified policy $P(x_1, \ldots, x_n)$. This is particularly useful in settings like permissioned blockchains, where a transaction might require approval not from any 5 nodes, but from a specific subset of auditors and validators as defined by the network's governance policy.

**Related work.** Over the last three decades, the problem of reducing the trust in the central master key authority and solving key escrow in FE has received a lot of attention. Numerous approaches have been studied to weaken the trust model in FE and its various specializations [Sha85, Coc01, BF01, GPSW06, BSW11]. The most well-studied models include the multi-authority (studied in this work) [Cha07, CC09, LW11, BCG+17], accountable-authority [Goy07, GLSW08], certificate-less encryption [ARP03, CCV04], anonymous encryption [BDOP04, Cho09, WQT18], and registration-based model [GHMR18, GHM+19, GV20].

**Registration vs. Multi-Authority model.** The registration-based model, introduced by [GHMR18, GHM+19], has seen significant success recently, with a flurry of works providing constructions in various contexts (cf. non-exhaustive list [GHMR18, GHM+19, GV20, HLWW23,

WW25, BLM+24, WW26] and references therein). In this model, each user creates their own public-secret key pair, while a *central* deterministic curator provides the encryption key and auxiliary secret keys to aid in decryption.

A significant limitation of the registration-based model is that the central curator's running time depends on the total number of users. Consider a system with a large user base (e.g., $2^{20} \approx 1$ million users). To support a policy $f$, the curator must aggregate public and secret keys from the entire population to generate valid encryption keys. In contrast, the multi-authority model's algorithmic complexity is independent of the user population. Users procure keys through local interactions with authorities, and the running times of key generation and encryption do not grow with the number of users. This gap matters concretely in many deployment settings. Consider settings where no user is willing to perform the expensive pre-processing step for public key accumulation, or a low-power sender that can broadcast observed data but does not receive firmware updates to refresh master keys.

Beyond efficiency, the multi-authority model also better captures the decentralized nature of real-world. As highlighted by Lu, Waters, and Wu [LWW25], a centralized curator is unrealistic when attributes are controlled by independent organizations that do not coordinate, e.g., a user obtaining a certificate for "student" from a university, "citizen" from a government body, and "engineer" from a private employer, then combining them locally. The registration-based model effectively forces coordination among these organizations.

## 2 Technical Overview

**Reviewing MAFE.** An MAFE scheme contains four main algorithms $(\mathsf{ASetup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$. The authority setup algorithm $\mathsf{ASetup}$ 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 $\mathsf{KGen}$ algorithm is used by each authority to compute a partial secret key for a portion of the input $x_i$ using their $\mathsf{MSK}_i$. $\mathsf{Enc}$ and $\mathsf{Dec}$ 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 $\mathsf{GID}$. We refer the reader to prior works [LW11] to get a better understanding of $\mathsf{GID}$. For this high level overview, we mostly ignore $\mathsf{GID}$s while explaining our main ideas. All of our constructions use a deterministic $\mathsf{GSetup}$ algorithm that simply outputs the parameters it was fed. We also ignore this algorithm throughout this section.

In MAFE, there are two types of corruptions – (a) secret key corruptions, (b) authority corruptions. A type-(a) attacker can only query secret keys from each authority, but not obtain their master keys. Type-(a+b) is a stronger attacker, who can additionally corrupt any $\mathsf{MSK}_i$ of its choice. In this work, we study, both, the bounded collusion and general collusion-resistant models. In the bounded collusion setting, an attacker makes an a-priori fixed (say $Q$) number of key queries and security holds against a simulator $\mathsf{Sim}$. In the collusion resistant setting, attacker makes an unbounded number of key queries and security is defined as an indistinguishability game.

## 2.1 MAFE with Malicious Authorities

As a starting point, let us consider an easier goal of designing MAFE with malicious authorities, where all malicious authorities are declared a-priori. That is, the MAFE system is initialized given a set of malicious authorities $\mathcal{K} \subset [n]$. While such "declared" malicious authorities do not capture any real-world attacks, we will show later how this can be used as stepping stone for handling arbitrary malicious authorities.

Consider $\mathcal{K} = \{1\}$, i.e. only first authority is malicious. Such a system will allow any attacker to learn $C(0, x_2, \ldots, x_n)$ and $C(1, x_2, \ldots, x_n)$ from a ciphertext $\mathsf{CT}$ and secret keys $\mathsf{sk}_{2,x_2}, \ldots, \mathsf{sk}_{n,x_n}$ by honestly running $\mathsf{KGen}$ and $\mathsf{Dec}$. Our intuition is to start with an MAFE that is secure against semi-honest authorities, and once $\mathcal{K} = \{1\}$ is available during setup, then simply initialize two parallel (semi-honest authority) MAFE systems, $\mathsf{MAFE}^{(0)}$, $\mathsf{MAFE}^{(1)}$ that encrypt $C(0, \cdot)$ and $C(1, \cdot)$. And, each semi-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 $\mathcal{K}$ is at most $O(\log \lambda)$.

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 highlights why assuming which authorities are malicious at setup time is meaningless. Regardless, it does serve as a good springboard as we show next.

**Hiding $\mathcal{K}$ via FSS.** Our idea is that to consider all feasible $\mathcal{K}$ and secret share the computation under them, rather than guessing $\mathcal{K}$. Let us elaborate.

Suppose $k$ denotes the maximum number of authorities that can be malicious. Let $\mathcal{C}$ denote the set of all size-$k$ subsets of $[n]$. Thus, $\mathcal{C}$ 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 malicious authorities*, $\mathsf{MAFE}^{(0)}, \ldots, \mathsf{MAFE}^{(|\mathcal{C}|)}$. In $\mathsf{MAFE}^{(i)}$, the malicious authority 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 $\mathsf{SK}_{u,x_u} = (\mathsf{sk}^{(i)}_{u,x_u})_{\{i:u\notin\mathcal{K}_i\}}$.

Now our idea is to rely on function secret sharing (FSS) [BGI15, BGI16] to perform encryption. In particular, we use an FSS 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 combines them by using reconstruction feature of FSS.

To prove security, we need a strong simulation guarantee for FSS. In a few words, we require that an attacker can learn evaluated shares corresponding to various honest parties in addition to corrupting some original share holders. And, even then it still does not learn anything beyond the final reconstructed value. An FSS with strong simulation readily follows from multi-party reusable non-interactive MPC (mrNISC) [BL20]. Previously, this was studied under the name 'Secret Sharing VBB' and was built using any mrNISC scheme secure against semi-honest adversaries. We leave it as a very interesting open problem to construct FSS schemes with strong simulation from other standard assumptions (other than ones that imply mrNISC like $\mathsf{SXDH}$ or $\mathsf{LWE}$) as it would readily give an MAFE scheme with malicious authorities using our approach.

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 $n = \mathsf{poly}(\lambda), k = O(1)$, $n = \mathsf{poly}(\log \lambda), k = O(\log^{1-\xi} \lambda)$ for $\xi > 0$, or $n = O(\log \lambda), k = O(\log \lambda)$, then above construction could be efficiently instantiated. These parameter regimes are summarized in Figure 1.

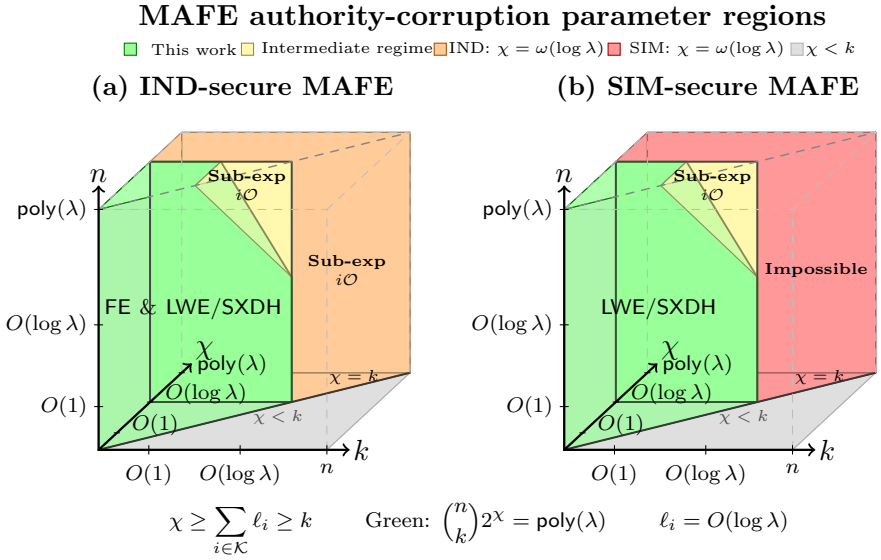


Figure 1: Qualitative parameter regimes for malicious-authority MAFE. Here $n$ is the total number of authorities, $k$ is the maximum number of corrupted authorities, and $\chi$ is the number of corrupted bits. Since every corrupted authority controls at least one bit, feasible parameters satisfy $\chi \geq k$. Prior work [BCG+17] required sub-exponential $i\mathcal{O}$ for all regions in IND-secure MAFE (left) and did not provide any results for SIM-secure MAFE. Although we do not have any results for parameter regime in yellow region, [BCG+17] shows that it can be achieved by sub-exponential $i\mathcal{O}$. However, we think it doesn't require obfuscation and hence portrayed it differently in the above figures. We leave it as an interesting open problem to design MAFE for this parameter range *without* obfuscation. Moreover, it would be interesting to design MAFE for orange region from minimal assumptions in collusion resistant setting.

**Security of bounded collusion MAFE.** Our design for MAFE with malicious authorities works almost generically in the bounded collusion model. We can readily take advantage of the simulation security provided by MAFE and only output evaluated shares of FSS to argue security. In a bit more detail, we guess an $i^* \in \mathcal{C}$ and assume that the attacker will corrupt authorities in $\mathcal{K}_{i^*}$. Note that we will only incur $1/|\mathcal{C}| = 1/\mathsf{poly}(\lambda)$ loss in advantage. Then we will simulate $\mathsf{MAFE}^{(i^*)}$-th instantiation to argue security using strong simulation property of FSS. For more details, we refer the reader to Section 5.

**Security of collusion resistant MAFE.** In the bounded collusion setting, we took advantage of simulation security and only rely on FSS evaluation outputs to argue security. However, in the collusion resistant setting, we don't have that facility due to known impossibilities [BSW11, AGVW13]. However, if we know the number of pre-challenge queries, say $Q_1$, we can create $Q_1$ different slots in the ciphertext for evaluated FSS shares. This technique was used previously to construct simulation secure FE from indistinguishably-secure FE [DIJ+13] in plain model.

In a bit more detail, for each $\mathsf{GID}$ query attacker makes before challenge ciphertext query, $i$-th authority for input $x_i$, additionally samples a random $\rho_i$. During encryption, we will add a trapdoor to the circuit being encrypted which if it detects the input as $(x_1, \rho_1, \ldots, x_n, \rho_n)$, outputs the evaluated FSS share of $C(x_1, \ldots, x_n)$. By encryption phase for challenge circuit, we know both

$C$ and all pre-challenge inputs $(x_1, \ldots, x_n)$. So, this can be done by relying on indistinguishability of MAFE. Post-challenge queries can be simulated using SKE with pseudorandom ciphertexts. Thus, we can construct an MAFE scheme with unbounded post-challenge queries and a bounded $(Q_1)$ pre-challenge queries using MAFE and FSS. We leave it as an interesting open problem to remove the pre-challenge bound in MAFE with malicious authorities in the collusion resistant setting.

Lastly, we complement our positive results regarding authority corruptions with almost tight lower bounds. We discuss those briefly below.

**Beyond $O(\log \lambda)$-bit corruptions.** In our design for MAFE with malicious authorities using FSS, we instantiated 2 MAFE schemes per one corrupted bit. Extending this generically, we would need $2^\chi$ parallel compositions if an attacker can corrupt a maximum of $\chi$ 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 $2^\chi$, 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-authority FE scheme (1-FE) 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 8 we show how to construct 1-FE with sub-linearity from an MAFE scheme with 'selective' corruptions[3]. 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 $\mathsf{MSK}$ 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 an *exponentially-efficient* iO scheme (XiO) with sub-linear size, an object that in conjunction with LWE yields iO [LPST16]. Moreover, simulation-secure versions of above are impossible in the standard model [BSW11, AGVW13]. Thus, our MAFE schemes with malicious authorities are nearly optimal.

## 2.2 Bounded Collusion MAFE with Semi-Honest Authorities from PKE

Next, to instantiate our malicious authority MAFE schemes as discussed above, we need to design MAFE with semi-honest authorities. As discussed in Section 1, even if all authorities behave honestly (but asynchronously), it is not known how to design MAFE from minimal assumptions. In this section, we address this problem in the bounded collusion setting.

Our plan is to break down designing an $n$-authority MAFE scheme to composing $n$ *single-authority* FE systems with certain special properties. At a high level, we introduce a new notion of MAFE system with *very weak security* below and show that it is a very important tool for designing MAFE.

**MAFE with synchronous setup.** The new notion of MAFE we use is *MAFE with synchronous setup* (ssMAFE). In ssMAFE, there is a *central 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 $\mathsf{ASetup}$ algorithm, ssMAFE has a global setup $\mathsf{GSetup}$ algorithm to generate $\{(\mathsf{MPK}_i, \mathsf{MSK}_i)\}_i$ for all authorities. Other than this, ssMAFE 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 $\mathsf{GSetup}$ instead

[3] Attacker declares malicious authorities selectively before viewing public parameters. This is clearly stronger than declaring malicious authorities for a system.

of $\mathsf{ASetup}$ $n$ times. Next, we show how to generically bootstrap ssMAFE to a standard (non-synchronous-setup) MAFE scheme.

**ssMAFE $\implies$ MAFE.** Note that ssMAFE 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 ssMAFE's global setup to encryption, by relying on another FE layer to handle the synchronous 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 use these independent systems to unlock the ssMAFE secret keys. Let us explain further.

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

The security of the above template can be quite readily reduced to the $n$ 1-FE instantiations and then ssMAFE security. Our compiler is described in detail later in Section 4. We highlight that we do not need any synchronization among the authorities, they are even oblivious to the existence of each other. We thus achieve dynamic authorities as well.

**Constructing ssMAFE.** Since 1-FE is already known from minimal assumptions [GVW12, AV19], thus we only need ssMAFE under same set of assumptions to instantiate our compiler.

A natural question the reader might have is: *isn't ssMAFE already pretty close to a single-authority FE (1-FE) system?* Basically, since there is a synchronous central setup, then what prevents us from sampling just a 1-FE system during the global setup, $\mathsf{GSetup}$, 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 ssMAFE might not be so hard after all. Although this is a good first idea, it does not work and highlights a key challenge towards MAFE.

In (ss)MAFE, each authority only generates a partial key for *its portion of the attribute* without knowing any other authority's inputs. 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) fails. Our strategy starts with the observation that the recent (semi-honest authority) MAFE construction by Goyal-Yadugiri [GY24] implicitly constructed an ssMAFE scheme, though they did not identify or formalize it as a standalone abstraction. Their core approach decentralized a 1-FE scheme [GVW12, AV19] by requiring authorities to make certain (randomized) choices to answer a partial key query for a $\mathsf{GID}$ consistently across asynchronous and non-interacting authorities.

Because [GY24] aimed for a standard MAFE without any synchronous setup, they took convoluted approaches to achieve this consistent randomness, relying on non-interactive key exchange or the ROM heuristic combined with sub-exponentially-hard assumptions.

Our insight is to extract this implicit core. By introducing the synchronous setup abstraction, the primary technical hurdle faced by [GY24], non-interactive random number sampling, can be rather easily solved. We simply have the central party sample a random PRF key $K$ and hand the same key to each authority. Using this shared key $K$, each authority can sample a consistent random number per $\mathsf{GID}$ with zero interaction. While there are some additional technical nuances in executing the above idea, isolating this component shows that ssMAFE can indeed be designed from the minimal assumption of (poly-hard, vanilla) PKE. We provide more details in Section 4.2.

Furthermore, using standard techniques established in the literature, we can also achieve dynamic collusion MAFE from the minimal assumption of IBE by appropriately strengthening our ssMAFE scheme. Moreover, the structure of our compiler from ssMAFE to MAFE provides some interesting relationships between various notions of authority corruptions, as we show in Section D.

### 2.3 Collusion Resistant MAFE with Semi-Honest Authorities from FE

In this section, we show how to construct a collusion resistant MAFE scheme where authorities behave semi-honestly from the minimal assumption of functional encryption (FE). At a very high level, our core idea for designing collusion-resistant MAFE is the same as for our bounded-collusion construction. That is, to design an $n$-authority MAFE, we compose $n$ distinct 1-authority functional encryption (FE) schemes[4]. However, we take a slightly different approach to combine them.

**From FE to MAFE.** In our scheme, each authority samples an FE public and secret key $(\mathsf{fe.mpk}_i, \mathsf{fe.msk}_i)$ and uses them as its own master public and secret keys. In order to encrypt a circuit $C$, the encryptor samples a PRF key $K$ and encrypts the tuple $(C, K)$ under each $\mathsf{fe.mpk}_i$ to generate $\mathsf{fe.ct}_i$. Final ciphertext is a combination of all $(\mathsf{fe.ct}_i)_i$. To generate a partial secret key for $(\mathsf{GID}, x_i)$, the $i$-th authority samples a uniform random string $\psi_i$ and generates FE secret key $\mathsf{fe.sk}_i$ for the following trapdoor circuit[5]:

- $\underline{F:}$ If input is $(C, K)$, garble the circuit $C$ using $\mathsf{PRF}(K, \mathsf{GID})$ as randomness and output the garbled circuit $\widetilde{C}$ and $i$-th wire label for bit $x_i$[6], $w_{i,x_i}$. Otherwise, if input is $\mathsf{ske.sk}^*$, decrypt $\psi_i$ with $\mathsf{ske.sk}^*$ and output the message.

If all $\mathsf{fe.sk}_i$ are generated using same $\mathsf{GID}$ correctness will be immediate. Specifically, decryption proceeds in two steps (i) using $\mathsf{fe.sk}_i$ and $\mathsf{fe.ct}_i$, we can obtain a $\widetilde{C}, w_{i,x_i}$, (ii) using $(w_{i,x_i})_i$, we can evaluate $\widetilde{C}$ to obtain $C(x_1, \ldots, x_n)$ as randomness used for garbling is the same across all authorities. Now, we will argue security of our scheme by relying on security of FE.

**Arguing selective security.** We now show how our design achieves selective security. Recall that in the selective setting, the challenge circuits $C_0, C_1$ are chosen by the adversary before seeing

[4]We remark that FE is a weaker assumption than indistinguishability obfuscation (iO) as it does not require sub-exponential-hard assumptions. This is evident by prior compilers from public-key FE to iO when FE offers sub-exponential hardness [AJ15, AJS15, BV15]. However, such results are not known when FE offers regular polynomial hardness.

[5]Counterintuitively, we use key-policy collusion resistant FE throughout this overview.

[6]Throughout this overview, for ease of exposition, we assume that input to each authority is 1 bit. This can be generalized easily as we demonstrate in Section 6.2.

$\{\mathsf{MPK}_i\}_i$. Security follows in 4 steps. In step 1, we use a secret-key encryption scheme with pseudorandom ciphertexts (SKE) to replace each $\psi_i$ sampled by each authority with an SKE ciphertext of $(\widetilde{C}_0, w_{i,x_i})$ under some key $\mathsf{ske.sk}^*$. This is done for every partial secret key query across all authorities. In step 2, we invoke the trapdoor circuit method [DIJ+13, GHRW14, BS18, ABSV15] to encrypt $\mathsf{ske.sk}^*$ under each $\mathsf{fe.mpk}_i$, where indistinguishability holds since the outputs match for both inputs. Step 3 relies on PRF security to garble the circuit with uniform randomness, and in step 4 we simulate the garbling.

At first sight, step 4 seems to yield selective security since it depends only on $C_0(x_1, \ldots, x_n)$, ensuring indistinguishability between $C_0$ and $C_1$. However, this is not the case. Simulating the garbling requires entire input $(x_1, \ldots, x_n)$, whereas each authority only observes its own input $x_i$. To resolve this mismatch, we argue security in a weaker model that we call *synchronous key generation*, where authorities generate keys *synchronously* yet independently when $x$ is provided as $(x_1, \ldots, x_n)$ together. This notion captures the security our construction achieves, and we expand on it below.

**Interlude: Synchronous key generation.** In a multi-authority system with synchronous key generation, the adversary submits inputs to all authorities together as a batch, i.e. $(\mathsf{GID}, x_1, \ldots, x_n)$ and receives secret keys together, i.e. $(\mathsf{SK}_{1,x_1}, \ldots, \mathsf{SK}_{n,x_n})$. We stress that authorities can generate their own keys independently of others and can generate partial secret keys without interacting one another. However, they have to *synchronously*, in any order, generate partial secret keys for inputs corresponding to any $\mathsf{GID}$. Although this hinders the asynchronous property of a multi-authority system, as we show later, any MAFE scheme secure in the synchronous key generation setting can be compiled into a scheme with asynchronous partial key generation easily. We defer the details to the end of the section and for now proceed with strengthening selective to adaptive security in synchronous key generation setting.

**Achieving adaptive security.** In order to achieve adaptive security (with synchronous key generation), we rely on the hybrid functional encryption idea from [ABSV15]. More specifically, we utilize a one-ciphertext *private-key* MAFE scheme (1-CT MAFE). A 1-CT MAFE scheme is defined similarly to a private-key MAFE scheme where encryption uses $\{\mathsf{MSK}_i\}_i$ to generate ciphertext for circuit $C$. Security holds only against attackers that corrupt partial secret keys for any unbounded $\mathsf{GID}$ and at most *one* challenge ciphertext. Using this object in conjunction with widely-used techniques from FE domain such as trapdoor circuits, SKE with pseudorandom ciphertexts, etc., we can readily achieve an MAFE scheme with adaptive security in the synchronous key generation setting. For more details, we refer the reader to Section 6.4. We will now illustrate how to construct a 1-CT MAFE scheme.

**Constructing 1-CT MAFE.** Deriving 1-CT MAFE from our bounded-collusion MAFE or the scheme of [GY24] is not straightforward. Bounded-collusion MAFE restricts the number of corrupted partial secret keys, whereas 1-CT MAFE restricts challenge encryptions while allowing *unbounded* key corruptions.

To resolve this, we pivot to *multi-client* functional encryption (MCFE) [GGG+14, CDSG+18], which distributes encryption across clients rather than key generation. Given an MCFE scheme secure against a single key corruption, we can apply function-hiding techniques [BS18] to swap the roles of encryption and key generation, directly yielding a 1-CT MAFE scheme. While general MCFE requires obfuscation [GGG+14], our setting only requires 1-key MCFE in the weaker synchronous ciphertext generation model (analogous to synchronous key generation). By composing $n$ private-key 1-key FE schemes [SS10, GVW12] and embedding a shared PRF key in each encryption

key $\mathsf{ek}_i$, we achieve this from the minimal assumption of one-way functions (see Appendix F for details). Leaving the construction of general MCFE from minimal assumptions as an open problem, we now proceed to address the synchronous key generation restriction of our MAFE designs.

**Removing synchronous key generation.** As mentioned, synchronous key generation hinders the asynchronous nature of multi-authority systems. One idea to remove this is to use an MPC scheme between all authorities for identity function, i.e., $f(x_1, \ldots, x_n) = (x_1, \ldots, x_n)$. However, as multi-authority systems require no interaction, this method might not exactly work. In our work, we provide a more general solution using non-committing encryption. Recall that in a non-committing encryption scheme ($\mathsf{NCE}$), is a public key encryption scheme with the additional property that we can "fake" a ciphertext and later reveal it to be encryption of message $m$ by constructing a tailor-made secret key for $m$. Formal syntax and properties is provided in Definition H.1.

For now, let's start with a simpler goal with $n = 2$ and delaying generation of $\mathsf{SK}_{1,x_1}$ until second authority is queried. In order to do so, each authority samples two $\mathsf{NCE}$ instantiations. That is, the first authority samples $(\mathsf{NCE}_1^{(1)}, \mathsf{NCE}_2^{(1)})$ and the second authority samples $(\mathsf{NCE}_1^{(2)}, \mathsf{NCE}_2^{(2)})$. After authority 1 generates $\mathsf{SK}_{1,x_1}$ using the MAFE with synchronous key generation, it samples $R_1, R_2$ and sets its output as

$$\left(\left(\mathsf{NCE}_1^{(1)}.\mathsf{Enc}(R_1), \mathsf{NCE}_1^{(2)}.\mathsf{Enc}(R_2)\right), \mathsf{NCE}_1^{(1)}.\mathsf{SK}, \mathsf{NCE}_2^{(1)}.\mathsf{SK}, R_1 \oplus R_2 \oplus \mathsf{SK}_{1,x_1}\right)$$

Second authority does something similar and only after $\mathsf{NCE}_1^{(2)}.\mathsf{SK}$ is revealed from second authority, $R_2$ is revealed and thus $\mathsf{SK}_{1,x_1}$ is revealed. In fact as second authority queried later, first authority can set its output as $\left(\left(\mathsf{NCE}_1^{(1)}.\mathsf{Enc}(R_1), \mathsf{NCE}_1^{(2)}.\mathsf{Fake}\right), \mathsf{NCE}_1^{(1)}.\mathsf{SK}, \mathsf{NCE}_2^{(1)}.\mathsf{SK}, \widetilde{R}\right)$ for uniformly random $\widetilde{R}$. When the second authority's input is revealed, it will set $\mathsf{NCE}_1^{(2)}.\mathsf{SK}$ as a secret key for message $\widetilde{R} \oplus R_1 \oplus \mathsf{SK}_{1,x_1}$. This way, we can delay generation of $\mathsf{SK}_{1,x_1}$ until we receive second input. In order to do this for $n$ authorities, each authority will need $n$ instantiations of $\mathsf{NCE}$.

One final hurdle to fix is that traditional $\mathsf{NCE}$ offers one-time security in that once a $\mathsf{NCE.SK}$ is lost, we cannot use its security anymore. To handle unbounded partial secret key queries, we can replace PKE-based NCE of [GVW12, HKM+24] with identity-based encryption (IBE). This way, we can output one $\mathsf{NCE.SK}_{\mathsf{GID}}$ for any each of $\mathsf{GID}$ query. We call this scheme a vector non-committing encryption scheme (VNCE) and provide its full construction in Appendix H. We finally remark that our presentation and main construction is in the 'bounded' authority setting where the $\mathsf{GSetup}$ algorithm depends on the number of authorities $n$.

**Synchronous key generation: Alternate perspective.** Another way of looking at our result is that in 1-authority FE systems (and possibly 1-authority ABE systems), if we can generate a slight split in secret key generation such that secret key of input $x = (x_1, \ldots, x_n)$ can be parsed as $\mathsf{SK}_{1,x_1}, \ldots, \mathsf{SK}_{n,x_n}$, we can compile such a key generation to the multi-authority setting. Techniques such as these are implicit in [SS10, GVW12] which [GY24] explored to construct MAFE in the bounded collusion setting. We show that this technique is more general and can be applied in the collusion resistant MAFE systems too.

**Overview of Constructions.** Our techniques from Sections 2.1, 2.2, and 2.3 utilize various techniques. For the ease of reader we summarize our techniques in Figure 2.

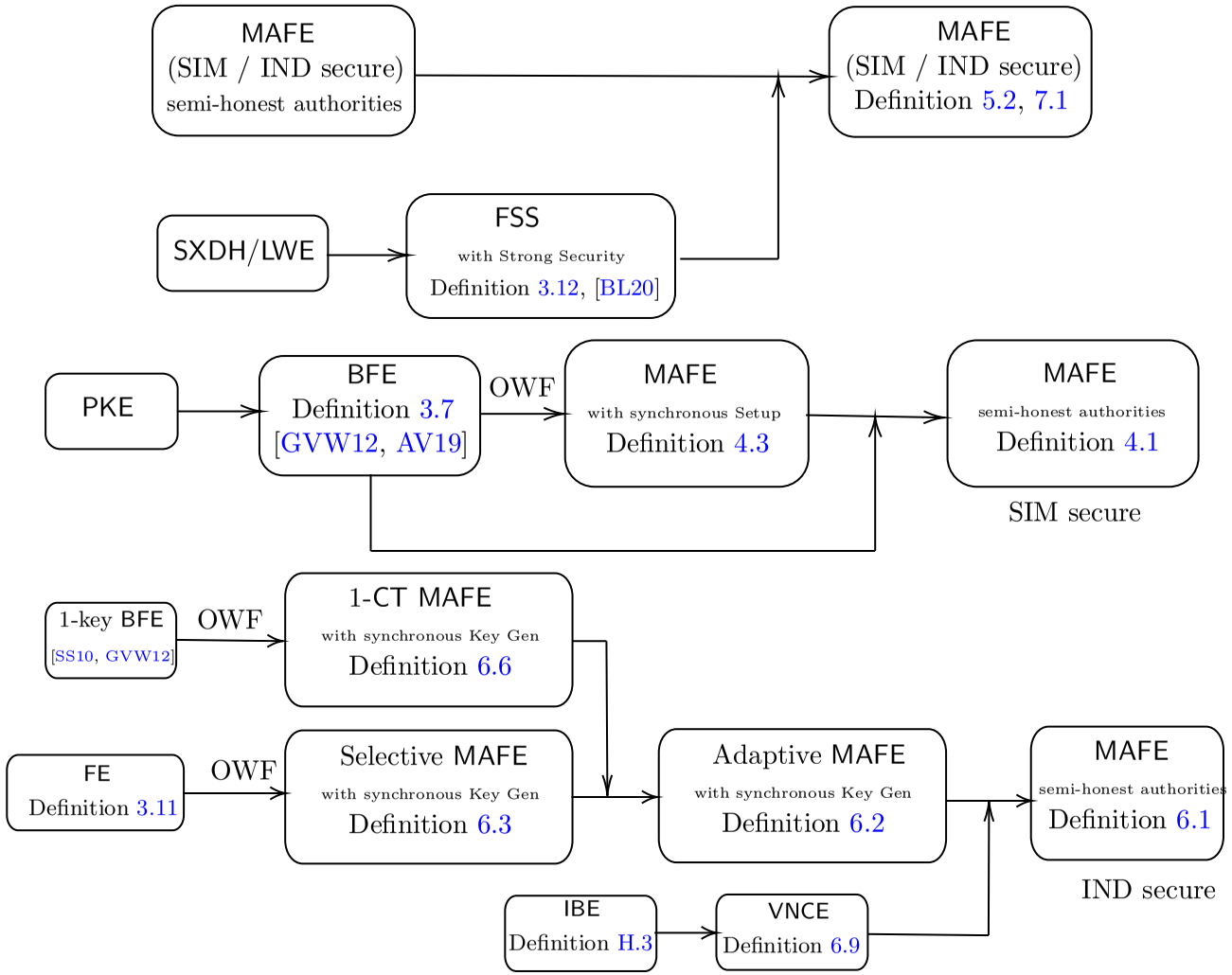


Figure 2: Summary of results from Sections 2.1, 2.2, 2.3.

# 3 Preliminaries

**Notation.** By $\mathsf{PPT}$ we denote probabilistic polynomial-time. We denote the security parameter by $\lambda$ 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 $\mathsf{PPT}$ 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 $\mathsf{poly}$, 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 $\mathsf{PPT}$ distinguisher $\mathcal{D} = \{\mathcal{D}_\lambda\}_{\lambda\in\mathbb{N}}$, and for large enough $\lambda \in \mathbb{N}$,

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

We denote statistical indistinguishability by $\approx_s$ and identical distributions by $\equiv$.

## 3.1 Pseudorandom Functions

**Definition 3.1** ($\mathsf{PRF}$)**.** A secure pseudorandom function $\mathsf{PRF}$ 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 $\mathsf{PRF}$ is a secure pseudorandom function if it satisfies the following properties.

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

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

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

**Remark 3.2** ([GGM84])**.** Assuming the existence of one-way functions, there exists an $\mathsf{PRF}$ scheme.

## 3.2 Secret-Key Encryption with Pseudorandom Ciphertexts

**Syntax.** A secret-key encryption scheme (SKE) with pseudorandom ciphertexts for message space $\{0,1\}^*$[7] consists of the following polynomial-time algorithms.

$\mathsf{Setup}(1^\lambda) \to \mathsf{SK}$**.** The randomized setup algorithm takes as input security parameter $\lambda$, and outputs secret key $\mathsf{SK}$.

$\mathsf{Enc}(\mathsf{SK}, m) \to \mathsf{CT}$**.** The randomized encryption algorithm takes as input secret key $\mathsf{SK}$, message $m \in \{0,1\}^*$, and outputs a ciphertext $\mathsf{CT}$.

[7]Recall that SKE can encrypt messages of arbitrary length using hybrid encryption.

$\mathsf{Dec}(\mathsf{SK}, \mathsf{CT}) \to m'$**.** The deterministic decryption algorithm takes as input secret key $\mathsf{SK}$, ciphertext $\mathsf{CT}$, and outputs message $m'$.

**Definition 3.3** ($\mathsf{SKE}$ with Pseudorandom Ciphertexts)**.** An $\mathsf{SKE}$ scheme $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ is said to be an SKE scheme with pseudorandom ciphertexts if it satisfies the following properties.

**Correctness.** For any $\lambda \in \mathbb{N}$, for any message $m \in \{0,1\}^*$,

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

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

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

where $\mathcal{O}_0$ takes as input $m \in \{0,1\}^*$ and samples $\mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{SK}, m)$ and outputs $\mathsf{CT}$. On the other hand, $\mathcal{O}_1$ on input $m \in \{0,1\}^*$ samples uniform random string of equivalent length. Note that this definition implies standard IND-CPA security for SKE.

**Remark 3.4** ([Gol04])**.** Assuming the existence of one-way functions, there exists an $\mathsf{SKE}$ scheme.

### 3.3 Garbled Ciruits

**Syntax.** We recall the definition of garbled circuits [Yao86] which is derived from [BHR12]. A garbling scheme for $\mathsf{P/Poly}$ circuits consists of the following polynomial time algorithms.

$\mathsf{Garble}(1^\lambda, C) \to \widetilde{C}, \{w_{i,b}\}_{i\in[n],b\in\{0,1\}}$**.** The randomized garbling algorithm takes as input security parameter $\lambda$, $\mathsf{P/Poly}$ circuit $C$ with $n$ input bits, outputs garbled circuit $\widetilde{C}$ and wire lables $\{w_{i,b}\}_{i\in[n],b\in\{0,1\}}$.

$\mathsf{Eval}(\widetilde{C}, \{w_{i,x_i}\}_{i\in[n]}) \to y$**.** The deterministic evaluation algorithm takes as input garbled circuit $\widetilde{C}$, wire labels for $n$-bit input $x$, $\{w_{i,x_i}\}_{i\in[n]}$, and outputs value $y$.

**Definition 3.5** ($\mathsf{Garb}$)**.** A $\mathsf{Garb}$ scheme $(\mathsf{Garble}, \mathsf{Eval})$ is said to be a garbling scheme for $\mathsf{P/Poly}$ circuits if satisfies the following properties.

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

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

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

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

**Remark 3.6** ([Yao86])**.** Assuming the existence of one-way functions, garbled circuits for any family of $\mathsf{P/Poly}$ circuits exist.

### 3.4 (Bounded-Collusion) Functional Encryption

In this section, we provide the definition of a (ciphertext-policy) Bounded-Collusion Functional Encryption (BFE) for circuit class $\mathcal{C} = \{\mathcal{C}_s\}_{s\in\mathbb{N}}$ and a (key-policy, collusion-resistant) Functional Encryption (FE) scheme for circuit class $\mathcal{C} = \{\mathcal{C}_s\}_{s\in\mathbb{N}}$. They the centralized / single-authority analogues of bounded-collision and collusion-resistant MAFE schemes respectively.

**Bounded-Collusion Functional Encryption.** A BFE scheme for circuit class $\mathcal{C} = \{\mathcal{C}_s\}_{s\in\mathbb{N}}$ consists of the following polynomial-time algorithms.

$\mathsf{Setup}(1^\lambda, 1^Q, 1^s) \to (\mathsf{MPK}, \mathsf{MSK})$**.** The randomized setup algorithm takes as input security parameter $\lambda$, collusion bound $Q$, maximum circuit size $s$, and outputs master public-secret key pair $(\mathsf{MPK}, \mathsf{MSK})$.

$\mathsf{KGen}(\mathsf{MSK}, x) \to \mathsf{SK}_x$**.** The possibly randomized key generation algorithm takes as input master secret key $\mathsf{MSK}$, input $x$, and output secret key for $x$, $\mathsf{SK}_x$.

$\mathsf{Enc}(\mathsf{MPK}, C) \to \mathsf{CT}$**.** The randomized encryption algorithm takes as input $\mathsf{MPK}$, circuit $C \in \mathcal{C}_s$, and outputs ciphertext $\mathsf{CT}$.

$\mathsf{Dec}(\mathsf{SK}_x, \mathsf{CT}) \to y$**.** The deterministic decryption algorithm takes as input secret key for $x$, $\mathsf{SK}_x$, ciphertext $\mathsf{CT}$, and outputs value $y$.

**Definition 3.7** (BFE)**.** A BFE scheme $(\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ for circuit class $\mathcal{C}$ satisfies the following properties.

**Correctness.** For any $\lambda \in \mathbb{N}, s = s(\lambda), Q = Q(\lambda), C \in \mathcal{C}_s$,

$$\Pr\left[\; C(x) = \mathsf{Dec}(\mathsf{SK}_x, \mathsf{CT}) \;\; : \;\; \begin{array}{l} (\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, 1^Q, 1^s), \\ \mathsf{SK}_x \leftarrow \mathsf{KGen}(\mathsf{MSK}, x), \\ \mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{MPK}, C) \end{array} \right] = 1$$

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

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

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

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

where $\mathcal{V} = \{(x, C(x)) : \text{ for each } x \text{ queried in pre-challenge phase}\}$. 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.

**Remark 3.8** ([GVW12, AV19])**.** The BFE schemes for P/Poly circuits designed by [GVW12, AV19] implicitly satisfy the strong simulation security property.

**Definition 3.9** (Dynamic BFE)**.** A BFE scheme $(\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ 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 3.7 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 $\lambda, s$ and running time of Enc is polynomial in $\lambda, Q, s$. Security of the scheme is defined accordingly.

**Remark 3.10** ([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.

**Definition 3.11** (Functional Encryption (FE))**.** A (key-policy, collusion-resistant) FE scheme is syntactically similar to a BFE scheme from Definition 3.7 except that there is no a-priori bound on the number of secret-keys an adversary can corrupt. Hence, we drop $1^Q$ from Setup. Moreover, KGen uses $C \in \mathcal{C}_s$ and Enc uses $x$. The security definition is slightly altered to indistinguishability-based security which we provide below.

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

$$\Pr\left[ b \leftarrow \mathcal{A}^{\mathsf{KGen}(\mathsf{MSK},\cdot)}(\mathsf{CT}) \quad : \quad \begin{array}{l} s \leftarrow \mathcal{A}(1^\lambda), \\ (\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, 1^s), \\ (x_0, x_1) \leftarrow \mathcal{A}^{\mathsf{KGen}(\mathsf{MSK},\cdot)}(\mathsf{MPK}), \\ b \xleftarrow{\$} \{0,1\}, \mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{MPK}, x_b) \end{array} \right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

A PPT machine $\mathcal{A}$ is said to be admissible if each $C \in \mathcal{C}_\lambda$ queried by $\mathcal{A}$ is at most size $s$, $C(x_0) = C(x_1)$ and $|x_0| = |x_1|$.

## 3.5 Function Secret Sharing with Strong Security

In this section, we provide the definition of an FSS scheme for P/Poly circuits that obeys a stronger version of security that what is found in the present literature. Specifically, we construct an FSS 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.

**Syntax.** A Function Secret Sharing (FSS) 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 $\lambda$, 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 3.12** (FSS)**.** A FSS scheme $(\mathsf{Share}, \mathsf{Eval}, \mathsf{Retrieve})$ is said to be an all-but-one FSS 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 $C \in \mathcal{C}_\lambda$,

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

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

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

$$\approx_c$$

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

**Theorem 3.13** ([BL20, BJKL21, AJJM21, Shi22])**.** Assuming SXDH or LWE, there exists a FSS scheme with strong security for any class of P/Poly circuits.

*Proof.* As shown in [BL20], any semi-honestly secure multi-party reusable non-interactive secure computation (MrNISC) scheme for P/Poly circuits implies an FSS scheme with strong security for any class of P/Poly circuits. We remark that in [BL20], this notion was referred to this notion as Secret Sharing VBB. □

# 4 MAFE from Public-Key Encryption

In this section, we provide the construction of an MAFE scheme for P/Poly circuits using ssMAFE scheme and a 1-authority MAFE (also called BFE) scheme for P/Poly circuits. In this section and Section 5, by MAFE we mean our bounded collusion simulation-secure version whose definition is provided in Definition 4.1.

## 4.1 Definition

A multi-authority functional encryption (MAFE) scheme MAFE for circuit class $\mathcal{C} = \{\mathcal{C}_s\}_{s\in\mathbb{N}}$ 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 $\lambda$, the query bound $Q$, the maximum circuit size $s$, and outputs $\mathsf{CRS}$. The following algorithms take $\mathsf{CRS}$ 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 $\mathsf{id}$-th authority $\ell_{\mathsf{id}}$, 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 $\mathsf{id}$-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 $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x}$.

$\mathsf{Enc}(\{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{I}}, C) \to \mathsf{CT}$**.** 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 $C \in \mathcal{C}_s$, and outputs ciphertext $\mathsf{CT}$.

$\mathsf{Dec}(\{\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}\}_{\mathsf{id}\in\mathcal{I}}, \mathsf{CT}) \to y$**.** 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 $\mathsf{CT}$, and outputs the value $y$.

**Definition 4.1** (MAFE)**.** An $\mathsf{MAFE}$ 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 $\lambda \in \mathbb{N}$, $Q = Q(\lambda), n = n(\lambda), s = s(\lambda)$, $C \in \mathcal{C}_\lambda, \mathsf{GID} \in \mathcal{GID}_\lambda$, $\forall\ \mathcal{I}, \ell_{\mathsf{id}} = \ell_{\mathsf{id}}(\lambda), x_{\mathsf{id}} \in \{0,1\}^{\ell_{\mathsf{id}}}$,

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

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

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

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 $\mathsf{id}$-th authority with input size $\ell_{\mathsf{id}}$. Gets $\mathsf{MPK}_{\mathsf{id}}$ in response.
- $(\mathsf{KGen}, (\mathsf{id}, \mathsf{GID}, x))$ $\mathcal{A}$ queries to corrupt secret key for $\mathsf{id}$-th authority for $\mathsf{GID}$ 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 $\mathsf{Sim}$ to respond to these queries. $\mathcal{O}_1$ also provides $\mathsf{Sim}$ 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 $\mathsf{GID}$ 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 $\mathsf{Sim}$.

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 4.2** (MAFE with "bounded" authorities)**.** An MAFE scheme (GSetup, ASetup, KGen, Enc, 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, $\{\mathsf{MPK}_{\mathsf{idx}}\}_{\mathsf{idx}\in[n]}$. 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 with at most $Q$ GIDs.

## 4.2 MAFE with Synchronous Setup

In this section, we define and show the construction of an important variant of MAFE, *MAFE with synchronous setup* (ssMAFE). In an ssMAFE 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 4.3** (ssMAFE)**.** An ssMAFE scheme (GSetup, KGen, Enc, Dec) is said to be an MAFE scheme with synchronous setup (ssMAFE) for circuit class $\mathcal{C}$ and $\mathcal{GID}$ if there is a central authority that generates all system parameters (i.e, $\mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}, \mathsf{MSK}_{\mathsf{id}}\}_{\mathsf{id}}$). 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 to Definition 4.2 with the above change.

We now provide the construction of an ssMAFE scheme for 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 is that if we restrict ourselves to the synchronous 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. Parts of the following construction are taken verbatim from [GY24].

**Construction 4.4** (ssMAFE)**.** We provide the construction of an ssMAFE 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* $\mathsf{GID}$ ([GY24]), a $\mathsf{dCSF}$ scheme for $\mathsf{P/Poly}$ circuits and $\mathcal{GID}$ (Definition A.1), and $\mathsf{PRF} : \{0,1\}^\lambda \times \mathcal{GID}_\lambda \to 2^{[N]}\big|_D \times 2^{[T]}\big|_v$ (Definition 3.1) as follows. Here $N, D, T, v$ are from Lemma A.3 and Lemma A.4.

$\underline{\mathsf{GSetup}(1^\lambda, 1^Q, 1^n, 1^s, 1^{\ell_1}, \ldots, 1^{\ell_n})}$**.** $\forall\ u \in [N]$, 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 $(\mathbf{S}, \boldsymbol{\Delta}) = \mathsf{PRF}(K, \mathsf{GID})$.

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

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 $\mathsf{CT}$ 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 4.5.** If $\mathsf{1MAFE}$ is a 1-$\mathsf{GID}$ MAFE scheme for $\mathsf{P/Poly}$ circuits, $\mathsf{dCSF}$ is a secure dCSF scheme for $\mathsf{P/Poly}$ circuits and $\mathcal{GID}$ (Definition A.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 3.1), then Construction 4.4 is an ssMAFE scheme for $\mathsf{P/Poly}$ 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 $(\mathbf{S}, \boldsymbol{\Delta})$ for each of the $Q$ $\mathsf{GID}$ queries during setup.

*As the PRF key is not leaked to the adversary, this change remains unnoticeable by the security of* $\mathsf{PRF}$*.*

**Hybrid 2.** In this hybrid, we will check if $\{\mathbf{S}_{\mathsf{GID}}\}_{\mathsf{GID}}$ obey Lemma A.3 and $\{\boldsymbol{\Delta}_{\mathsf{GID}}\}_{\mathsf{GID}}$ obey Lemma A.4.

*Hybrids 1 and 2 are statistically indistinguishable due to these lemmas. We will also compute the set of corrupted users and set of unique indices in each* $\boldsymbol{\Delta}_{\mathsf{GID}}$*.*

**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 ssMAFE.Sim.

The full proof is similar to the non-interactive key exchange based construction of MAFE from [GY24]. We omit this for brevity. □

**Corollary 4.6** ([GY24])**.** Assuming the existence of public-key encryption, there exists an ssMAFE 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. □

**Remark 4.7.** Assuming the existence of one-way functions, there exists a secret-key ssMAFE scheme for P/Poly circuits and $\mathcal{GID}$.

### 4.3 MAFE from ssMAFE

We now provide the construction of an MAFE scheme as follows.

**Remark 4.8** (Deterministic KGen)**.** In the construction below, we assume w.l.o.g that ssMAFE.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.

**Construction 4.9** (MAFE)**.** We provide the construction of an MAFE scheme for P/Poly circuits and $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda\in\mathbb{N}}$ using a BFE scheme for P/Poly circuits (Definition 3.7) and an ssMAFE scheme for P/Poly circuits and $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda\in\mathbb{N}}$ (Definition 4.3) as follows.

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

$\mathsf{ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}})$**.** Sample keys $(\mathsf{bfe.mpk}, \mathsf{bfe.msk}) \leftarrow \mathsf{BFE.Setup}(1^\lambda, 1^Q, 1^{s'})$. Here $s' = \mathsf{poly}(\lambda, Q, s, \ell_{\mathsf{id}})$ is the maximum size of key generation circuit of ssMAFE for these parameters. Output $\mathsf{MPK} := (\ell_{\mathsf{id}}, \mathsf{bfe.mpk})$ and $\mathsf{MSK} := (\ell_{\mathsf{id}}, \mathsf{bfe.msk})$.

$\mathsf{KGen}(\mathsf{id}, \mathsf{MSK}_{\mathsf{id}}, \mathsf{GID}, x)$**.** 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 $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} := \mathsf{bfe.sk}_{\mathsf{id},\mathsf{GID},x}$.

$\mathsf{Enc}(\{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{I}}, C)$**.** 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 $\mathsf{mafe.crs}, \{(\mathsf{mafe.mpk}_i, \mathsf{mafe.msk}_i)\}_i \leftarrow \mathsf{ssMAFE.GSetup}(1^\lambda, 1^Q, 1^n, 1^s, 1^{\ell_{\mathsf{id}_1}}, \ldots, 1^{\ell_{\mathsf{id}_n}})$.

- Compute $\mathsf{mafe.ct} \leftarrow \mathsf{ssMAFE.Enc}(\{\mathsf{mafe.mpk}_i\}_i, C)$,
- $\forall i \in [n], \mathsf{bfe.ct}_{\mathsf{id}_i} \leftarrow \mathsf{BFE.Enc}(\mathsf{bfe.mpk}_{\mathsf{id}_i}, \mathsf{ssMAFE.KGen}(\cdot, \mathsf{mafe.msk}_i, \{\mathsf{mafe.mpk}_{i'}\}_{i' \in [n]}, \cdot, \cdot))$.

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 $\bot$.

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 $y := \mathsf{ssMAFE.Dec}(\{\mathsf{mafe.sk}_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{I}}, \mathsf{mafe.ct})$.

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

**Theorem 4.10.** If ssMAFE is an ssMAFE scheme (Definition 4.3) 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 3.7), then Construction 4.9 is an MAFE scheme for P/Poly circuits and $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda \in \mathbb{N}}$ (Definition 4.1).

*Proof Sketch.* We provide a sketch of the hybrid argument used to prove security of Construction 4.9 as follows.

**Hybrid 0.** This is the same as the real experiment from Definition 4.1. We use honest and stateless BFE and ssMAFE 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 ssMAFE. Now that all $\mathsf{id} \in \mathcal{I}$ are simulated, we will only be using secret keys and not he master secret keys of ssMAFE.

*Hence, indistinguishability can be argued from security of* ssMAFE.

This is the description of Sim from Definition 4.1.

We provide the full proof in Appendix B. □

**Corollary 4.11.** Assuming the existence of public-key encryption, there exists an MAFE scheme (with unbounded authorities) for P/Poly circuits and $\mathcal{GID}$.

*Proof.* The proof of this corollary follows from Corollary 4.6, and construction of BFE for P/Poly circuits assuming PKE from [AV19]. □

**Remark 4.12.** 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[8].

# 5 Bounded Collusion MAFE with Malicious Authorities

In this section, we provide the definition and construction for an MAFE scheme (recall MAFE in bounded collusions and simulation secure, Definition 4.1) with malicious authorities. For an MAFE scheme with malicious authorities (cMAFE), the main difference is that it is secure even if some of the master secret keys for authorities are lost.

## 5.1 Definition

**Definition 5.1** (cMAFE)**.** A $\mathsf{cMAFE}$ scheme, $(\mathsf{GSetup}, \mathsf{ASetup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ is said to be an MAFE scheme that supports $\chi$ 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 $\chi$. We take the number of authorities $n$ and $\chi$ as inputs to $\mathsf{GSetup}$ and work in the "bounded" authority setting of MAFE (Definition 4.2). Security of the scheme is defined analogously with the following important changes.

- $\mathcal{O}$ responds to $\mathsf{KGen}$ 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 $\mathsf{Sim}$ 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 $\mathsf{GID}$ that was used to query all authorities. If $\mathsf{id}$-th authority corrupted, then $\mathbf{X}_{\mathsf{GID},\mathsf{id}} = \{0,1\}^{\ell_{\mathsf{id}}}\}$. Similarly, the simulator $\mathsf{Sim}$ queries $C$ for every $X \in\in \mathbf{X}_{\mathsf{GID},1} \times \ldots \mathbf{X}_{\mathsf{GID},n}$ where for corrupted $\mathsf{id}$, $\mathbf{X}_{\mathsf{GID},\mathsf{id}} = \{0,1\}^{\ell_{\mathsf{id}}}$ and a singleton $x_{\mathsf{GID},\mathsf{id}}$ otherwise.

A stateful $\mathsf{PPT}$ 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 $\mathsf{id}$ is corrupted, $\mathcal{A}$ should not issue for secret key queries to $\mathsf{id}$.

## 5.2 MAFE with Adaptive Corruptions from FSS

In this section, we construct an MAFE scheme with adaptive authority corruptions using a function 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 our construction, we use a functional secret sharing scheme with strong simulation security (Section 3.5). Below, we formally define and construct $(n,k)$-$\mathsf{cMAFE}$.

**Definition 5.2** ($(n,k)$-$\mathsf{cMAFE}$)**.** An $(n,k)$-$\mathsf{cMAFE}$ scheme for circuit class $\mathcal{C}$ and $\mathcal{GID}$ is defined similarly to Definition 5.1 and the maximum number of corruptible authorities, $k$ are declared a-priori as an input to the $\mathsf{GSetup}$ algorithm. In addition, we require that $\binom{n}{k} = \mathsf{poly}(\lambda)$. We satisfy a the following notion of security in Construction 5.3. For any admissible adversary $\mathcal{A}$ that meets the admissibility criterion in Definition 5.1, there exists a $\mathsf{PPT}$ simulator $\mathsf{Sim}$ such that $\forall\ \lambda \in \mathbb{N}$,

[8] As we work in the ciphertext-policy setting, message-privacy means that an adversary cannot distinguish the input used in $\mathsf{KGen}$.

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

$$\approx_c$$

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

where $\mathcal{O}$ is a stateful oracle that responds to key generation using $\mathsf{KGen}$ 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 5.3** ($(n,k)$-cMAFE)**.** We provide the construction of $(n,k)$-cMAFE for P/Poly circuits and $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda\in\mathbb{N}}$ (Definition 5.2) using an MAFE scheme for P/Poly circuits and $\mathcal{GID}$ (Definition 4.2), a FSS scheme for P/Poly circuits (Definition 3.12) 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 $2^{[n]}\big|_k$[9] 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 $[n]\setminus\mathcal{K}_i$[10], i.e, for the $(i,j)$-th instantiation of MAFE, $\mathsf{id} \notin \mathcal{K}_i$[11]. 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 $\ell_{\mathsf{idx}}$. Using $(\ell_1, \ldots, \ell_n)$, define a maximal corruptible set $\mathcal{K}' \subseteq [n]$ such that $\ell_{\mathsf{id}} = O(\log\lambda)$ for $\mathsf{id} \in \mathcal{K}'$. This is done to figure out the exact set of authorities which can be corrupted which may be less than $k$.

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

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

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

[9] Recall from Appendix 3 that $2^{\mathbf{A}}\big|_a$ is the set of all subsets of set $\mathbf{A}$ of size $a$.
[10] We use $\mathsf{id} \notin \mathcal{K}_i$ and $\mathsf{id} \in [n]\setminus\mathcal{K}_i$ interchangeably.
[11] This is done to avoid complicated notation required in the subsequent algorithms. We use this fact implicitly.

$\underline{\mathsf{Enc}(\{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}, C)}$. Parse $\mathsf{MPK}_{\mathsf{id}}$ as $(\mathsf{mpk}_{\mathsf{id}}^{(i,j)})_{i\in\mathbb{I},j\in\mathbb{J}}$ for each $\mathsf{id}\in[n]$. Parse $\mathcal{K}'$ similar to $\mathsf{KGen}$. Share the circuit $C$ using $\mathsf{FSS}$ for all $i\in\mathbb{I}'$, $\{\widetilde{C}_i\}_{i\in\mathbb{I}'}\leftarrow\mathsf{FSS.Share}(1^\lambda, 1^Q, 1^{|\mathbb{I}'|}, C)$. 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 $\mathsf{FSS.Eval}(i,\widetilde{C}_i,x)$.

That is, for every $i\in\mathbb{I}', j\in\mathbb{J}, \mathsf{ct}^{(i,j)}\leftarrow\mathsf{MAFE.Enc}(\{\mathsf{mpk}_{\mathsf{id}}^{(i,j)}\}_{\mathsf{id}\notin\mathcal{K}_i}, F_{ij})$. 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 $\mathsf{CT}$ 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 $(\mathsf{mafe.sk}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}^{(i,j)})_{i,j}$ 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 $\mathsf{MAFE}$, $y_{i,j}=\mathsf{MAFE.Dec}(\{\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j)}\}_{\mathsf{id}\notin\mathcal{K}_i}, \mathsf{ct}^{(i,j)})$. Output $y:=\mathsf{FSS.Retrieve}(\{y_{i,j}\}_{i\in\mathbb{I}'})$.

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

**Theorem 5.4.** If $\mathsf{FSS}$ is a secure FSS scheme for $\mathsf{P/Poly}$ circuits (Definition 3.12), $\mathsf{MAFE}$ is a secure MAFE scheme for $\mathsf{P/Poly}$ circuits and $\mathcal{GID}$ (Definition 4.2), then Construction 5.3 is a secure $(n,k)$-$\mathsf{cMAFE}$ 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 5.3 as follows.

**Hybrid 0.** This is the real experiment from Definition 5.2. We use the stateless $\mathsf{MAFE}$ and $\mathsf{FSS}$ 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 $\mathsf{MAFE}$.

*The indistinguishability of hybrids can be argued using the security of* $\mathsf{MAFE}$. *In addition, if the adversary corrupts any* $\mathsf{id}$ *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 malicious authority set correctly, 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* $\mathsf{MAFE}$ *security.*

**Hybrid 2.** In this hybrid, we will simulate the $\mathsf{FSS}$ instantiation. Now that all $\mathsf{MAFE}$ instantiations for $i^*$ (after correctly guessing at the beginning) are simulated, we would only be using the outputs of share evaluation.

*We can readily rely on the strong security of* $\mathsf{FSS}$ *to show indistinguishability.*

This is the description of $\mathsf{Sim}$ from Definition 5.2.

The full proof is in Appendix C. □

# 6 Collusion Resistant MAFE from Functional Encryption

In this section, we construct a collusion-resistant MAFE scheme for any class of $\mathsf{P/Poly}$ circuits from the minimal assumption of an FE scheme for any class of $\mathsf{P/Poly}$ circuits. In this section and Section 7, for simplicity, by MAFE we mean collusion-resistant version with indistinguishability security. We define this Definition 6.1. Our construction proceeds in three steps.

1. We construct a weaker version of MAFE scheme that is *selectively* secure and with *synchronous key generation*. We formally define these notions in Section 6.2.
2. We then generically bootstrap any selectively secure MAFE scheme to an adaptively secure version with synchronous key generation using a private-key one-ciphertext MAFE scheme.
3. We then use *vector* non-committing encryption to realize adaptively secure MAFE as per Definition 6.1.

## 6.1 Definition

We now formally define our MAFE scheme.

**Definition 6.1** (MAFE)**.** An $\mathsf{MAFE}$ scheme $(\mathsf{GSetup}, \mathsf{ASetup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ is said to be an MAFE scheme if there is no a-priori bound on the number of $\mathsf{GID}$'s an adversary can corrupt. Syntactically, we drop the bound $1^Q$ from $\mathsf{GSetup}$. Note that this is a strengthening of Definition 4.1. MAFE in the bounded authorities model is defined similarly to Definition 4.2 with the aforementioned strengthening. From henceforth, whenever we refer to MAFE, we refer to the bounded authority version. Security definition is altered to a weaker indistinguishability notion defined below.

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

$$\Pr\left[ b \leftarrow \mathcal{A}^{\mathcal{O}(\cdot,\cdot,\cdot)}(\mathsf{CT}) \quad : \quad \begin{array}{l} (n, s, \ell_1, \ldots, \ell_n) \leftarrow \mathcal{A}(1^\lambda), L = \max_{\mathsf{id}}\{\ell_{\mathsf{id}}\} \\ \mathsf{CRS} \leftarrow \mathsf{GSetup}(1^\lambda, 1^n, 1^s, 1^L), \\ \forall\ \mathsf{id} \in [n], (\mathsf{MPK}_{\mathsf{id}}, \mathsf{MSK}_{\mathsf{id}}) \leftarrow \mathsf{ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}}), \\ (C_0, C_1) \leftarrow \mathcal{A}^{\mathcal{O}(\cdot,\cdot,\cdot)}(\mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}), \\ b \xleftarrow{\$} \{0,1\}, \mathsf{CT} \leftarrow \mathsf{Enc}(\{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}, C_b) \end{array} \right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where $\mathcal{O}$ is a key-generation oracle that takes as input $(\mathsf{id}, \mathsf{GID}, x_{\mathsf{id}}) \in [n] \times \mathcal{GID} \times \{0,1\}^{\ell_{\mathsf{id}}}$ and outputs $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{KGen}(\mathsf{id}, \mathsf{MSK}_{\mathsf{id}}, \{\mathsf{MPK}_{\mathsf{idx}}\}_{\mathsf{idx}\in[n]}, \mathsf{GID}, x_{\mathsf{id}})$. A stateful $\mathsf{PPT}$ machine $\mathcal{A}$ is said to be an admissible adversary if for each $\mathsf{GID} \in \mathcal{GID}$ queried by $\mathcal{A}$ with the corresponding queries $(1, \mathsf{GID}, x_1), \ldots, (n, \mathsf{GID}, x_n)$, $C_0(x_1, \ldots, x_n) = C_0(x_1, \ldots, x_n)$. Moreover, $C_0, C_1 \in \mathcal{C}_\lambda, |C_0| = |C_1|$ and are of size at most $s$.

## 6.2 Selectively Secure MAFE with Synchronous Key Generation

In this section, we provide the definition and construction of a selectively-secure MAFE with synchronous key generation from a (key-policy, collusion-resistant) FE scheme (Definition 3.11).

Firstly, we define an MAFE scheme with synchronous key generation in Definition 6.2 and then define selectively secure version as a natural weakening of this definition. At a high level, synchronous key generation security requires that the adversary declare all the input $x_1, \ldots, x_n$ for $n$ authorities for the same $\mathsf{GID}$ at once together. Note that $\mathsf{KGen}$ of each $\mathsf{id} \in [n]$ is still independent of other.

**Definition 6.2.** An asMAFE scheme $(\mathsf{GSetup}, \mathsf{ASetup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ is said to be an MAFE with synchronous key generation if it satisfies the correctness property from Definition 6.1 and following (weaker) security definition.

**IND Security with Synchronous Key Generation.** For any PPT admissible adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

$$\left| \Pr \left[ b \leftarrow \mathcal{A}^{\mathcal{O}(\cdot,\cdot)}(\mathsf{CT}) \quad : \quad \begin{array}{l} (n, s, \ell_1, \ldots, \ell_n) \leftarrow \mathcal{A}(1^\lambda), L = \max_{\mathsf{id}}\{\ell_{\mathsf{id}}\}, \\ \mathsf{CRS} \leftarrow \mathsf{GSetup}(1^\lambda, 1^n, 1^s, 1^L), \\ \forall\ \mathsf{id} \in [n], (\mathsf{MPK}_{\mathsf{id}}, \mathsf{MSK}_{\mathsf{id}}) \leftarrow \mathsf{ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}}), \\ (C_0, C_1) \leftarrow \mathcal{A}^{\mathcal{O}(\cdot,\cdot)}(\mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}), \\ b \xleftarrow{\$} \{0,1\}, \mathsf{CT} \leftarrow \mathsf{Enc}(\{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}, C_b) \end{array} \right] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda)$$

where $\mathcal{O}$ is a key-generation oracle that takes as input $(\mathsf{GID}, (x_1, \ldots, x_n)) \in \mathcal{GID} \times \{0,1\}^{\ell_1} \times \ldots \times \{0,1\}^{\ell_n}$ and outputs $(\mathsf{SK}_{1,\mathsf{GID},x_1}, \ldots, \mathsf{SK}_{n,\mathsf{GID},x_n})$ where $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{KGen}(\mathsf{id}, \mathsf{MSK}_{\mathsf{id}}, \{\mathsf{MPK}_{\mathsf{idx}}\}_{\mathsf{idx}}, \mathsf{GID}, x_{\mathsf{id}})$ for each $\mathsf{id} \in [n]$. A stateful PPT machine $\mathcal{A}$ is said to be an admissible adversary if for each $\mathsf{GID} \in \mathcal{GID}$ queried by $\mathcal{A}$ with the corresponding query $(\mathsf{GID}, x_1, \ldots, x_n)$, $C_0(x_1, \ldots, x_n) = C_0(x_1, \ldots, x_n)$. Moreover, $C_0, C_1 \in \mathcal{C}_\lambda, |C_0| = |C_1|$ and are of size at most $s$.

**Definition 6.3.** An selsMAFE scheme $(\mathsf{GSetup}, \mathsf{ASetup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ is said to be a *selectively* secure MAFE with synchronous key generation if in the above security game, the circuits $C_0, C_1$ are declared selectively by $\mathcal{A}$, even before receiving $(\mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}})$. $\mathcal{A}$ then receives $\mathsf{CT}$ along with $(\mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}})$.

Now, we construct an selsMAFE scheme satisfying Definition 6.3.

**Construction 6.4** (selsMAFE)**.** We provide the construction of a selectively-secure MAFE scheme with synchronous key generation for P/Poly circuits and $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda\in\mathbb{N}}$ using an FE scheme for P/Poly circuits (Definition 3.11), pseudorandom function $\mathsf{PRF} : \{0,1\}^\lambda \times \mathcal{GID} \to \{0,1\}^\lambda$ (Definition 3.1), an SKE with pseudorandom ciphertexts (Definition 3.3), and garbled circuit scheme $(\mathsf{Garb}, \mathsf{Eval})$ for P/Poly circuits (Definition 3.5) as follows.

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

$\underline{\mathsf{ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}})}$**.** Given $\mathsf{id} \in [n], \ell_{\mathsf{id}} \leq L$, sample FE keys, $(\mathsf{fe.mpk}, \mathsf{fe.msk}) \leftarrow \mathsf{FE.Setup}(1^\lambda, 1^{s'})$ where $s' = \mathsf{poly}(\lambda, n, L, s)$ is the maximum size of the circuit from $\mathsf{KGen}$. Output $\mathsf{MPK} := (\mathsf{id}, \ell_{\mathsf{id}}, \mathsf{fe.mpk})$ and $\mathsf{MSK} := (\mathsf{id}, \mathsf{fe.msk})$.

$\underline{\mathsf{KGen}(\mathsf{id}, \mathsf{MSK}_{\mathsf{id}}, \{\mathsf{MPK}_{\mathsf{idx}}\}_{\mathsf{idx}\in[n]}, \mathsf{GID}, x)}$**.** Parse $\mathsf{MSK}_{\mathsf{id}}$ as $(\mathsf{id}, \mathsf{fe.msk}_{\mathsf{id}})$ and for each $\mathsf{idx} \in [n]$, $\mathsf{MPK}_{\mathsf{idx}}$ as $(\mathsf{idx}, \ell_{\mathsf{idx}}, \mathsf{fe.mpk}_{\mathsf{idx}})$. Sample $\mathsf{ske.ct}^* \xleftarrow{\$} \{0,1\}^{\ell^*_{\mathsf{id}}}$ where $\ell^*_{\mathsf{id}} = \mathsf{poly}(\lambda, n, L, s)$ is the size of the garbled circuit for $s$-size circuits and $\ell_{\mathsf{id}}$-many wire labels padded appropriately. Create the circuit $F$ as described in Figure 3.

Sample an FE secret key $\mathsf{fe.sk}_F \leftarrow \mathsf{FE.KGen}(\mathsf{fe.msk}_{\mathsf{id}}, F)$. Output $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} := \mathsf{fe.sk}_F$.

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

Sample key for $\mathsf{PRF}$, $K \xleftarrow{\$} \{0,1\}^\lambda$. Define $n$ sets of indices $\mathbb{I}_1, \ldots, \mathbb{I}_n$ where $\mathbb{I}_1 := \{1, \ldots, \ell_1\}$ and $\mathbb{I}_{\mathsf{id}} = \left\{\left(\sum_{i=1}^{\mathsf{id}-1} \ell_i\right) + 1, \ldots, \left(\sum_{i=1}^{\mathsf{id}} \ell_i\right)\right\}$ for $\mathsf{id} \in [2, n]$. Similarly, define indices $\alpha_1, \ldots, \alpha_n$ to be the numerically first indices in sets $\mathbb{I}_1, \ldots, \mathbb{I}_n$ respectively.

For each $\mathsf{id} \in [n]$, sample an $\mathsf{FE}$ ciphertext, $\mathsf{fe.ct}_{\mathsf{id}} \leftarrow \mathsf{FE.Enc}(\mathsf{fe.mpk}_{\mathsf{id}}, (C, K, \alpha_{\mathsf{id}}, \mathbb{I}_{\mathsf{id}}, 0, \bot))$.

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

$\mathsf{Dec}(\{\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}\}_{\mathsf{id}\in[n]}, \mathsf{CT})$. Parse $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}$ as $\mathsf{fe.sk}_{F_{\mathsf{id}}}$ for each $\mathsf{id} \in [n]$ and $\mathsf{CT}$ as $(\mathsf{fe.ct}_{\mathsf{id}})_{\mathsf{id}\in[n]}$.

Compute $(\widetilde{C}_{\mathsf{id}}, \overline{w}_{\mathsf{id}}) = \mathsf{FE.Dec}(\mathsf{fe.sk}_{F_{\mathsf{id}}}, \mathsf{fe.ct}_{\mathsf{id}})$. If all $\widetilde{C}_{\mathsf{id}}$ are not the same, abort and output $\bot$. Otherwise, let $\widetilde{C} = \widetilde{C}_1$ and output $y = \mathsf{Eval}(\widetilde{C}, (\overline{w}_{\mathsf{id}})_{\mathsf{id}\in[n]})$.

**Circuit $F$**

**Input:** $C, K, \alpha, \mathbb{I}$, mode, $\mathsf{td}$
**Hardwired:** $\mathsf{ske.ct}^*, \mathsf{id}, \mathsf{GID}, x$
**Output:**

- If mode $= 0$, compute $\widetilde{C}, \{w_{i,b}\}_{b\in\{0,1\},i} = \mathsf{Garb}(1^\lambda, C; \mathsf{PRF}(K, \mathsf{GID}))$. Output $\widetilde{C}, \{w_{i,x[i-\alpha+1]}\}_{i\in\mathbb{I}}$, where $x[j]$ is the $j$-th bit of $x \in \{0,1\}^{\ell_{\mathsf{id}}}$.
- Otherwise, if mode $= 1$, parse $\mathsf{td}$ as $\mathsf{ske.sk}$ and output $\mathsf{SKE.Dec}(\mathsf{ske.sk}, \mathsf{ske.ct}^*)$.

Figure 3: Description of circuit $F$

**Correctness.** The correctness of Construction 6.4 follows from correctness of $\mathsf{FE}$, garbling scheme, and deterministic nature of $\mathsf{PRF}$. By correctness of $\mathsf{FE}$, we have that for each $\mathsf{id} \in [n]$, $(\widetilde{C}_{\mathsf{id}}, \overline{w}_{\mathsf{id}}) = F_{\mathsf{id}}(C, K, \alpha_{\mathsf{id}}, \mathbb{I}_{\mathsf{id}}, 0, \bot)$. By Figure 3, we have that as mode $= 0$, $\widetilde{C}_{\mathsf{id}}$ is the garbling of $C$ and $\overline{w}_{\mathsf{id}}$ is the set of wire labels for the $\mathsf{id}$-th input $x_{\mathsf{id}} \in \{0,1\}^{\ell_{\mathsf{id}}}$. For any $\mathsf{id} \in [n]$ these are generated using the randomness $\mathsf{PRF}(K, \mathsf{GID})$. Hence, if all $\mathsf{GID}$ are the same, all $\widetilde{C}_{\mathsf{id}}$ are the same and $\overline{w}_{\mathsf{id}}$ are the wire labels for the input $(x_1, \ldots, x_n)$ to the circuit $C$. Hence, $y = C(x_1, \ldots, x_n)$.

**Lemma 6.5.** If $\mathsf{FE}$ is an FE scheme for $\mathsf{P/Poly}$ circuits (Definition 3.11), $\mathsf{PRF} : \{0,1\}^\lambda \times \mathcal{GID}_\lambda \to \{0,1\}^\lambda$ is a PRF scheme (Definition 3.1), $\mathsf{SKE}$ is an SKE scheme with pseudorandom ciphertexts (Definition 3.3), and $(\mathsf{Garb}, \mathsf{Eval})$ is a garbling scheme for $\mathsf{P/Poly}$ circuits (Definition 3.5), then Construction 6.4 is a selectively secure MAFE scheme with synchronous key generation (Definition 6.3) for $\mathsf{P/Poly}$ circuits and $\mathcal{GID}$.

*Proof Sketch.* We provide a sketch of the hybrid argument used to prove security of Construction 6.4 as follows.

**Hybrid 0.** This is the real experiment from Definition 6.3 with $b = 0/1$.

**Hybrid 1.** In this experiment, we will sample $\mathsf{ske.sk} \leftarrow \mathsf{SKE.Setup}(1^\lambda)$ and set $\mathsf{ske.ct}^*_{\mathsf{id},\mathsf{GID}}$ to be encryptions of the string $(\widetilde{C}, \{w_{i,x_{\mathsf{id}}[i-\alpha_{\mathsf{id}}]}\}_{i\in\mathbb{I}_{\mathsf{id}}})$ for each $\mathsf{id} \in [n]$ and each $\mathsf{GID}$ queried by $\mathcal{A}$

where $(\widetilde{C}, \{w_{i,b}\}_{i,b}) = \mathsf{Garb}(1^\lambda, C_b; \mathsf{PRF}(K, \mathsf{GID}))$. Note that this is the same as the output of $F_{\mathsf{id}}$ when mode $= 0$.

*Hybrids 0 and 1 are indistinguishable by security of* SKE *which has pseudorandom ciphertexts.*

**Hybrid** $2, j$**.** for $j \in [n+1]$. In this hybrid, we will set the input to the first $j-1$ FE encryptions as $(\overline{0}, \text{mode} = 1, \mathsf{ske.sk})$[12].

*When mode* $= 1$*, the output is the plaintext of* $\mathsf{ske.ct}^*_{\mathsf{id}}$*. By previous hybrid, we set it to be the same as output when mode* $= 0$*. Indistinguishability follows by* FE *security.*

**Hybrid 3.** In this hybrid, we sample a uniformly random string $R_{\mathsf{GID}} \xleftarrow{\$} \{0,1\}^\lambda$ for each GID instead of using $\mathsf{PRF}(K, \cdot)$.

*Hybrid* $2, n+1$ *and 3 are indistinguishable by* PRF *security.*

**Hybrid** $4, j$**.** for $j \in [Q+1]$. Here, $Q = \mathsf{poly}(\lambda)$ is the number of queries made by $\mathcal{A}$ to the challenger. In this hybrid, we simulate the first $j-1$ garblings of circuit $C_b$.

*As we know* $C_0, C_1$ *selectively, we can use* $C_b(x_1, \ldots, x_n)$ *for each* GID *to perform this operation. Thus, indistinguishability follows by* Garb *security. At the end of Hybrid* $4, Q+1$*, challenger only relies on* $C_b(x_1, \ldots, x_n)$*. As* $C_0(x_1, \ldots, x_n) = C_1(x_1, \ldots, x_n)$*, Hybrid* $4, Q+1$ *is independent of bit* $b$ *and advantage of any adversary in this experiment is exactly* $0$*.*

We provide full proof in Appendix E. □

### 6.3 Private-Key One-Ciphertext MAFE

We define our private-key one-ciphertext MAFE scheme for P/Poly circuits below which is used to boost the MAFE with synchronous key generation and selective security from previous section to an adaptively secure version in the next section.

**Syntax.** A private-key one-ciphertext MAFE scheme 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.

$\mathsf{GSetup}(1^\lambda, 1^n, 1^s, 1^{\ell_1}, \ldots, 1^{\ell_n}) \to (\mathsf{CRS}, \{\mathsf{MSK}_{\mathsf{id}}\}_{\mathsf{id} \in [n]})$**.** The probabilistic global setup algorithm takes as input the security parameter $\lambda$, number of authorities $n$, the maximum circuit size $s$, outputs CRS, and master secret keys for all authorities, $\{\mathsf{MSK}_{\mathsf{id}}\}_{\mathsf{id} \in [n]}$. The following algorithms take CRS implicitly.

$\mathsf{KGen}(\mathsf{id}, \mathsf{MSK}_{\mathsf{id}}, \mathsf{GID}, x) \to \mathsf{SK}_{\mathsf{id},\mathsf{GID},x}$**.** 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 $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x}$.

$\mathsf{Enc}(\{\mathsf{MSK}_{\mathsf{id}}\}_{\mathsf{id} \in [n]}, C) \to \mathsf{CT}$**.** The probabilistic encryption algorithm takes as input the master secret keys for all authorities, $\{\mathsf{MSK}_{\mathsf{id}}\}_{\mathsf{id} \in [n]}$, a circuit $C \in \mathcal{C}_\lambda$, and outputs ciphertext CT.

$\mathsf{Dec}(\{\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}\}_{\mathsf{id} \in [n]}, \mathsf{CT}) \to y$**.** The deterministic decryption algorithm takes as input the secret keys for all authorities, $\{\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}\}_{\mathsf{id} \in [n]}$, ciphertext CT, and outputs the value $y$.

[12] By $\overline{0}$ we mean an all zero string of sufficient length.

**Definition 6.6** (opMAFE)**.** An opMAFE scheme $(\mathsf{GSetup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ is said to be a one-ciphertext private-key MAFE scheme for circuit class $\mathcal{C}$ and $\mathcal{GID}$ if it satisfies the following properties.

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

$$\Pr\left[\begin{array}{l} C(x_1, \dots x_n) = \\ \mathsf{Dec}(\{\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}\}_{\mathsf{id}}, \mathsf{CT}) \end{array} : \begin{array}{l} (\mathsf{CRS}, \{\mathsf{MSK}_{\mathsf{id}}\}_{\mathsf{id}}) \leftarrow \mathsf{GSetup}(1^\lambda, 1^n, 1^s, 1^{\ell_1}, \dots, 1^{\ell_n}), \\ \forall\ \mathsf{id} \in [n], \mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}} \leftarrow \mathsf{KGen}(\mathsf{id}, \mathsf{MSK}_{\mathsf{id}}, \mathsf{GID}, x_{\mathsf{id}}), \\ \mathsf{CT} \leftarrow \mathsf{Enc}(\{\mathsf{MSK}_{\mathsf{id}}\}_{\mathsf{id}}, C) \end{array}\right] = 1$$

**IND Security with Synchronous Key Generation.** For any PPT admissible adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

$$\Pr\left[b \leftarrow \mathcal{A}^{\mathcal{O}(\cdot,\cdot)}(\mathsf{CT}) \ : \begin{array}{l} (n, s, \ell_1, \dots, \ell_n) \leftarrow \mathcal{A}(1^\lambda), \\ (\mathsf{CRS}, \{\mathsf{MSK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}) \leftarrow \mathsf{GSetup}(1^\lambda, 1^n, 1^s, 1^{\ell_1}, \dots, 1^{\ell_n}), \\ (C_0, C_1) \leftarrow \mathcal{A}^{\mathcal{O}(\cdot,\cdot)}(\mathsf{CRS}), \\ b \xleftarrow{\$} \{0,1\}, \mathsf{CT} \leftarrow \mathsf{Enc}(\{\mathsf{MSK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}, C_b) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where $\mathcal{O}$ is a key-generation oracle that takes as input $(\mathsf{GID}, (x_1, \dots, x_n)) \in \mathcal{GID} \times \{0,1\}^{\ell_1} \times \dots \times \{0,1\}^{\ell_n}$ and outputs $(\mathsf{SK}_{1,\mathsf{GID},x_1}, \dots, \mathsf{SK}_{n,\mathsf{GID},x_n})$ where $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{KGen}(\mathsf{id}, \mathsf{MSK}_{\mathsf{id}}, \mathsf{GID}, x_{\mathsf{id}})$ for each $\mathsf{id} \in [n]$. A stateful PPT machine $\mathcal{A}$ is said to be an admissible adversary if for each $\mathsf{GID} \in \mathcal{GID}$ queried by $\mathcal{A}$ with the corresponding query $(\mathsf{GID}, x_1, \dots, x_n)$, $C_0(x_1, \dots, x_n) = C_1(x_1, \dots, x_n)$. Moreover, $C_0, C_1 \in \mathcal{C}_\lambda, |C_0| = |C_1|$ and are of size at most $s$.

We construct this primitive from one-way functions in Appendix F.

## 6.4 Adaptively Secure MAFE with Synchronous Key Generation

In this section, we construct an adaptively secure MAFE scheme that satisfies Definition 6.2. For our construction, we require a private-key one-ciphertext MAFE scheme with IND security as defined in Definition 6.6.

**Construction 6.7** (asMAFE)**.** We provide the construction of an MAFE scheme with synchronous key generation for P/Poly circuits and $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda\in\mathbb{N}}$ using a selectively-secure MAFE scheme with synchronous key generation for P/Poly circuits (Definition 6.3) and $\mathcal{GID}$, an SKE with pseudorandom ciphertexts (Definition 3.3), pseudorandom function $\mathsf{PRF} : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^\lambda$ (Definition 3.1), and a one-ciphertext private-key MAFE scheme for P/Poly circuits and $\mathcal{GID}$ (Definition 6.6) as follows.

$\underline{\mathsf{GSetup}(1^\lambda, 1^n, 1^L, 1^s)}$**.** Sample $\mathsf{selmafe.crs} \leftarrow \mathsf{selsMAFE.GSetup}(1^\lambda, 1^n, 1^L, 1^s)$. Output $\mathsf{CRS} := \mathsf{selmafe.crs}$.

$\underline{\mathsf{ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}})}$**.** Given $\mathsf{id} \in [n], \ell_{\mathsf{id}} \leq L$, sample selsMAFE keys, $(\mathsf{selmafe.mpk}, \mathsf{selmafe.msk}) \leftarrow \mathsf{selsMAFE.ASetup}(\mathsf{id}, 1^{\ell'_{\mathsf{id}}})$ where $\ell'_{\mathsf{id}} = \mathsf{poly}(\lambda, n, \ell_1, \dots, \ell_n, s)$ is the input size for the id-th authority from the circuit in Figure 4. Output $\mathsf{MPK} := \mathsf{selmafe.mpk}$ and $\mathsf{MSK} := \mathsf{selmafe.msk}$.

$\mathsf{KGen}(\mathsf{id}, \mathsf{MSK}_{\mathsf{id}}, \{\mathsf{MPK}_{\mathsf{idx}}\}_{\mathsf{idx}\in[n]}, \mathsf{GID}, x)$. Parse $\mathsf{MSK}_{\mathsf{id}}$ as $\mathsf{selmafe.msk}_{\mathsf{id}}$ and for each $\mathsf{idx} \in [n]$, $\mathsf{MPK}_{\mathsf{idx}}$ as $\mathsf{selmafe.mpk}_{\mathsf{idx}}$. Sample $\rho \xleftarrow{\$} \{0,1\}^\lambda, \mathsf{ske.ct}^* \xleftarrow{\$} \{0,1\}^{\ell^*_{\mathsf{id}}}$ where $\ell^*_{\mathsf{id}} = \mathsf{poly}(\lambda, n, \ell_1, \ldots, \ell_n, s)$ is the size of the opMAFE secret keys for the parameters specified.

Sample an selsMAFE secret key $\mathsf{selmafe.sk}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{selsMAFE.KGen}(\mathsf{id}, \mathsf{selmafe.msk}_{\mathsf{id}}, \{\mathsf{selmafe.mpk}_{\mathsf{idx}}\}_{\mathsf{idx}}, \mathsf{GID}, (\mathsf{id}, \mathsf{GID}, x, \rho, \mathsf{ske.ct}^*))$. Output $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} := \mathsf{selmafe.sk}_{\mathsf{id},\mathsf{GID},x}$.

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

Sample an opMAFE instantiation $(\mathsf{pmafe.crs}, \{\mathsf{pmafe.msk}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}) \leftarrow \mathsf{opMAFE.GSetup}(1^\lambda, 1^n, 1^s, 1^{\ell_1}, \ldots, 1^{\ell_n})$ and PRF key, $K \xleftarrow{\$} \{0,1\}^\lambda$. Construct the circuit $G$ specified in Figure 4.

Sample opMAFE ciphertext for $C$, $\mathsf{pmafe.ct} \leftarrow \mathsf{opMAFE.Enc}(\{\mathsf{pmafe.msk}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}, C)$ and selsMAFE ciphertext for circuit $G$, $\mathsf{selmafe.ct} \leftarrow \mathsf{selsMAFE.Enc}(\{\mathsf{selmafe.mpk}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}, G)$.

Output $\mathsf{CT} := (\mathsf{pmafe.ct}, \mathsf{selmafe.ct})$.

$\mathsf{Dec}(\{\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}\}_{\mathsf{id}\in[n]}, \mathsf{CT})$. Parse $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}$ as $\mathsf{selmafe.sk}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}$ for each $\mathsf{id} \in [n]$ and CT as $(\mathsf{pmafe.ct}, \mathsf{selmafe.ct})$.

Compute $(\mathsf{pmafe.sk}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}})_{\mathsf{id}\in[n]} = \mathsf{selsMAFE.Dec}(\mathsf{selmafe.sk}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}, \mathsf{selmafe.ct})$. If this aborts, then abort and output $\perp$.

Otherwise, compute and output $y = \mathsf{opMAFE.Dec}(\{\mathsf{pmafe.sk}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}\}_{\mathsf{id}\in[n]}, \mathsf{pmafe.ct})$.

**Circuit $G$**

**Input:** $1, \mathsf{GID}_1, x_1, \rho_1, \mathsf{ske.ct}^*_1, \ldots, n, \mathsf{GID}_n, x_n, \rho_n, \mathsf{ske.ct}^*_n$
**Hardwired:** $\{\mathsf{pmafe.msk}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}, K$
**Output:** If all $\mathsf{GID}_{\mathsf{id}}$ are not same, abort and output $\perp$. Otherwise, let $\mathsf{GID} := \mathsf{GID}_1$,

- For each $\mathsf{id} \in [n]$, sample $\mathsf{pmafe.sk}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}} = \mathsf{opMAFE.KGen}(\mathsf{id}, \mathsf{pmafe.msk}_{\mathsf{id}}, \mathsf{GID}, x_{\mathsf{id}}; \mathsf{PRF}(K, \rho_{\mathsf{id}}))$.
- Output $\{\mathsf{pmafe.sk}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}\}_{\mathsf{id}\in[n]}$.

Figure 4: Description of circuit $G$

**Correctness.** The correctness of Construction 6.7 follows from correctness of selsMAFE and opMAFE. By correctness of selsMAFE, we have that for each $\mathsf{id} \in [n]$, $\mathsf{pmafe.sk}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}} = \mathsf{opMAFE.KGen}(\mathsf{id}, \mathsf{pmafe.msk}_{\mathsf{id}}, \mathsf{GID}, x_{\mathsf{id}}; \mathsf{PRF}(K, \rho_{\mathsf{id}}))$. Then, by correctness of opMAFE, if all GID are the same, $y = C(x_1, \ldots, x_n)$.

**Lemma 6.8.** If selsMAFE is a selectively-secure MAFE scheme for P/Poly circuits and $\mathcal{GID}$ (Definition 6.3), $\mathsf{PRF} : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^\lambda$ is a PRF scheme (Definition 3.1), SKE is an SKE scheme with pseudorandom ciphertexts (Definition 3.3), and opMAFE is a one-ciphertext private-key MAFE scheme for P/Poly circuits and $\mathcal{GID}$ (Definition 6.6), then Construction 6.7 is an MAFE scheme with synchronous key generation (Definition 6.2) for P/Poly circuits and $\mathcal{GID}$.

*Proof Sketch.* We provide a sketch of the hybrid argument used to prove security of Construction 6.7 as follows.

**Hybrid 0.** This is the real experiment from Definition 6.3 with $b = 0/1$.

**Hybrid 1.** In this experiment, we will sample $\mathsf{ske.sk} \leftarrow \mathsf{SKE.Setup}(1^\lambda)$ and set $\mathsf{ske.ct}^*_{\mathsf{id},\mathsf{GID}}$ to be encryptions of the string $\mathsf{pmafe.sk}_{\mathsf{id},\mathsf{GID},x}$ for each $\mathsf{id} \in [n]$ and each $\mathsf{GID}$ queried by $\mathcal{A}$ where $\mathsf{pmafe.sk}_{\mathsf{id},\mathsf{GID},x} = \mathsf{opMAFE.KGen}(\mathsf{id}, \mathsf{pmafe.msk}_{\mathsf{id}}, \mathsf{GID}, x; \mathsf{PRF}(K, \rho_{\mathsf{id},\mathsf{GID}}))$. Note that this is the same as one of the output of $G$. To do so, we will sample $K, \{\mathsf{pmafe.msk}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}$ early which doesn't change the distribution of ciphertext in Hybrid 0.

*Hybrid 0 and 1 are indistinguishable by security of* SKE *which has pseudorandom ciphertexts.*

**Hybrid 2.** In this hybrid, we will change the circuit $G$ to $G'$ which has the same size as $G$, but instead of hardwiring $\mathsf{pmafe.msk}_{\mathsf{id}}$'s and $K$, we hardwire $\mathsf{ske.sk}$ in $G'$. We also change the functionality of $G'$ to simply decrypt and output $\mathsf{ske.ct}^*_{\mathsf{id},\mathsf{GID}}$.

*By previous hybrid, we set* $\mathsf{ske.ct}^*_{\mathsf{id},\mathsf{GID}}$ *to be the same as output of* $G$*. Thus, indistinguishability follows by* selsMAFE *security as we can determine* $G$ *and* $G'$ *at the beginning of the experiment.*

**Hybrid 3.** In this hybrid, we sample a uniformly random string $R_{\mathsf{id},\mathsf{GID}} \xleftarrow{\$} \{0,1\}^\lambda$ for each $\mathsf{GID}$ query instead of using $\mathsf{PRF}(K, \rho_{\mathsf{id},\mathsf{GID}})$.

*Hybrid* 2 *and 3 are indistinguishable by* PRF *security.*

**Hybrid** 4. In this hybrid, we use opMAFE security to switch $\mathsf{pmafe.ct}$ from an encryption of $C_b$ to $C_{1-b}$.

*Indistinguishability follows by* opMAFE *security. From here, we can follow the same sequence of hybrids to reach Hybrid 1 with* $1 - b$*.*

We provide full proof in Appendix G. □

## 6.5 Vector Non-Committing Encryption

In this section, we introduce the notion of vector non-committing encryption (VNCE) and in the next section, use it to remove the synchronous key generation requirement from our MAFE scheme. VNCE can be used to commit a vector of messages $(\mu_1, \ldots, \mu_n)$ where each $\mu_{\mathsf{id}}$ for $\mathsf{id} \in [n]$ is from the $\mathsf{id}$-th party. The idea is that each party encrypts its own input with respect to a specific $\mathsf{GID}$ and all other party's public information. These messages will only be revealed after collecting encryptions from all parties.

We need this scheme to be reusable for various messages but for a specific $\mathsf{GID}$. That is, $(\mu_1, \ldots, \mu_n)$ can only be revealed if $\mathsf{id}$-th party encrypts $(\mathsf{GID}, \mu_{\mathsf{id}})$ for $\mathsf{id} \in [n]$. We describe the scheme in the notation of MAFE for ease of exposition. In a way, this primitive can be thought of reusable MPC for identity functions with delayed output generation.

**Syntax.** A vector non-committing encryption scheme (VNCE) for global identifier domain $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda\in\mathbb{N}}$ consists of the following polynomial-time algorithms.

$\mathsf{GSetup}(1^\lambda, 1^n, 1^L) \to \mathsf{CRS}$**.** The possibly randomized global setup algorithm takes as input the security parameter $\lambda$, number of authorities $n$, each party's input size $L$, and outputs $\mathsf{CRS}$. The following algorithms take $\mathsf{CRS}$ implicitly.

$\mathsf{ASetup}(\mathsf{id}) \to (\mathsf{MPK}, \mathsf{MSK})$**.** The probabilistic authority setup algorithm outputs master public and secret key pair $(\mathsf{MPK}, \mathsf{MSK})$.

$\mathsf{Enc}(\mathsf{id}, \mathsf{MSK}_{\mathsf{id}}, \{\mathsf{MPK}_{\mathsf{idx}}\}_{\mathsf{idx}\in[n]}, \mathsf{GID}, \mu) \to \mathsf{ct}_{\mathsf{id},\mathsf{GID}}$**.** The probabilistic encryption algorithm takes as input the $\mathsf{id}$-th master secret key $\mathsf{MSK}_{\mathsf{id}}$, the global identifier for the user $\mathsf{GID} \in \mathcal{GID}_\lambda$, an input $\mu$ such that $|\mu| = L$, and outputs encryption $\mathsf{ct}_{\mathsf{id},\mathsf{GID}}$.

$\mathsf{Dec}(\{\mathsf{ct}_{\mathsf{id},\mathsf{GID}}\}_{\mathsf{id}\in[n]}) \to y$**.** The deterministic decryption algorithm takes as input encryptions from all authorities, $\{\mathsf{ct}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}\}_{\mathsf{id}\in[n]}$, ciphertext $\mathsf{CT}$, and outputs the value $y \in \{0,1\}^{nL} \cup \{\perp\}$.

**Definition 6.9** (VNCE)**.** An $\mathsf{VNCE}$ scheme $(\mathsf{GSetup}, \mathsf{ASetup}, \mathsf{Enc}, \mathsf{Dec})$ is said to be a VNCE scheme for $\mathcal{GID}$ if it satisfies the following properties.

**Correctness.** For any $\lambda \in \mathbb{N}$, $n = n(\lambda), L = L(\lambda)$, $\mathsf{GID} \in \mathcal{GID}_\lambda$, $\forall\ \mathsf{id} \in [n], \ell_{\mathsf{id}} = \ell_{\mathsf{id}}(\lambda), \mu_{\mathsf{id}} \in \{0,1\}^L$,

$$\Pr\left[\begin{array}{l|l} (\mu_1, \dots \mu_n) = & \mathsf{CRS} \leftarrow \mathsf{GSetup}(1^\lambda, 1^n, 1^L), \forall\ \mathsf{id} \in [n], \\ \mathsf{Dec}(\{\mathsf{ct}_{\mathsf{id},\mathsf{GID}}\}_{\mathsf{id}}) & (\mathsf{MPK}_{\mathsf{id}}, \mathsf{MSK}_{\mathsf{id}}) \leftarrow \mathsf{ASetup}(\mathsf{id}), \\ & \mathsf{ct}_{\mathsf{id},\mathsf{GID}} \leftarrow \mathsf{Enc}(\mathsf{id}, \mathsf{MSK}_{\mathsf{id}}, \{\mathsf{MPK}_{\mathsf{idx}}\}_{\mathsf{idx}\in[n]}, \mathsf{GID}, \mu_{\mathsf{id}}) \end{array}\right] = 1$$

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

$$\left\{\mathcal{A}^{\mathcal{O}(\cdot,\cdot,\cdot)}(\mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}) : \begin{array}{l} n, L \leftarrow \mathcal{A}(1^\lambda), \\ \mathsf{CRS} \leftarrow \mathsf{GSetup}(1^\lambda, 1^n, 1^L), \\ \forall\ \mathsf{id} \in [n], (\mathsf{MPK}_{\mathsf{id}}, \mathsf{MSK}_{\mathsf{id}}) \leftarrow \mathsf{ASetup}(\mathsf{id}) \end{array}\right\}$$

$$\approx_c$$

$$\left\{\mathcal{A}^{\mathsf{Sim}(\cdot,\cdot,\mathsf{V})}(\mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}) : \begin{array}{l} n, L \leftarrow \mathcal{A}(1^\lambda), \\ \mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]} \leftarrow \mathsf{Sim}(1^\lambda, 1^n, 1^L) \end{array}\right\}$$

where both $\mathcal{O}, \mathsf{Sim}$ respond to queries from $\mathcal{A}$ of the form $(\mathsf{id}, \mathsf{GID}, \mu_{\mathsf{id},\mathsf{GID}})$. $\mathcal{O}$ outputs $\mathsf{ct}_{\mathsf{id},\mathsf{GID}} \leftarrow \mathsf{Enc}(\mathsf{id}, \mathsf{MSK}_{\mathsf{id}}, \{\mathsf{MPK}_{\mathsf{idx}}\}_{\mathsf{idx}\in[n]}, \mathsf{GID}, \mu)$. $\mathsf{Sim}$ only uses $(\mathsf{id}, \mathsf{GID})$ and additionally uses the value $\mathsf{V}$. Here, $\mathsf{V} = (\mu_{1,\mathsf{GID}}, \dots, \mu_{n,\mathsf{GID}})$ if this is the last authority that needs to be queried for this $\mathsf{GID}$ and $\perp$ otherwise.

In Appendix H we show how to construct a VNCE scheme from IBE.

## 6.6 MAFE from FE

In this section, we construct a collusion resistant MAFE scheme satisfying Definition 6.1. We will use an MAFE scheme with synchronous key generation (Definition 6.2) and VNCE (Definition 6.9). The idea essentially is that using VNCE, we can "flatten" the key generation queries and rely on synchronous key generation security.

**Construction 6.10.** We provide the construction of an MAFE scheme for $\mathsf{P/Poly}$ circuits and $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda\in\mathbb{N}}$ using an MAFE scheme with synchronous key generation for $\mathsf{P/Poly}$ circuits (Definition 6.2) and $\mathcal{GID}$, and a VNCE scheme for $\mathcal{GID}$ (Definition 6.9) as follows.

$\underline{\mathsf{GSetup}(1^\lambda, 1^n, 1^L, 1^s)}$. Sample $\mathsf{mafe.crs} \leftarrow \mathsf{asMAFE.GSetup}(1^\lambda, 1^n, 1^L, 1^s)$ and $\mathsf{nce.crs} \leftarrow \mathsf{VNCE.GSetup}(1^\lambda, 1^n, 1^{L'})$ where $L' = \mathsf{poly}(\lambda, n, s, L)$ is the maximum length of asMAFE secret keys for any authority. Output $\mathsf{CRS} := (\mathsf{mafe.crs}, \mathsf{nce.crs})$.

$\underline{\mathsf{ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}})}$. Given $\mathsf{id} \in [n], \ell_{\mathsf{id}} \leq L$, sample asMAFE keys, $(\mathsf{mafe.mpk}, \mathsf{mafe.msk}) \leftarrow \mathsf{asMAFE.ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}})$ and VNCE keys $(\mathsf{nce.mpk}, \mathsf{nce.msk}) \leftarrow \mathsf{VNCE.ASetup}(\mathsf{id})$. Output $\mathsf{MPK} := (\mathsf{mafe.mpk}, \mathsf{nce.mpk})$ and $\mathsf{MSK} := (\mathsf{mafe.msk}, \mathsf{nce.msk})$.

$\underline{\mathsf{KGen}(\mathsf{id}, \mathsf{MSK}_{\mathsf{id}}, \{\mathsf{MPK}_{\mathsf{idx}}\}_{\mathsf{idx}\in[n]}, \mathsf{GID}, x)}$. Parse $\mathsf{MSK}_{\mathsf{id}}$ as $(\mathsf{mafe.msk}_{\mathsf{id}}, \mathsf{nce.msk}_{\mathsf{id}})$ and for each $\mathsf{idx} \in [n]$, $\mathsf{MPK}_{\mathsf{idx}}$ as $(\mathsf{mafe.mpk}_{\mathsf{idx}}, \mathsf{nce.mpk}_{\mathsf{idx}})$.

Sample an asMAFE secret key $\mathsf{mafe.sk}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{asMAFE.KGen}(\mathsf{id}, \mathsf{mafe.msk}_{\mathsf{id}}, \{\mathsf{mafe.mpk}_{\mathsf{idx}}\}_{\mathsf{idx}}, \mathsf{GID}, x)$. Now, encrypt this string under VNCE, $\mathsf{nce.sk}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{VNCE.Enc}(\mathsf{id}, \mathsf{nce.msk}_{\mathsf{id}}, \{\mathsf{nce.mpk}_{\mathsf{idx}}\}_{\mathsf{idx}\in[n]}, \mathsf{GID}, \mathsf{mafe.sk}_{\mathsf{id},\mathsf{GID},x})$.

Output $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} := \mathsf{nce.sk}_{\mathsf{id},\mathsf{GID},x}$.

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

Sample an asMAFE ciphertext for $C$, $\mathsf{mafe.ct} \leftarrow \mathsf{asMAFE.Enc}(\{\mathsf{mafe.mpk}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}, C)$.

Output $\mathsf{CT} := \mathsf{mafe.ct}$.

$\underline{\mathsf{Dec}(\{\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}\}_{\mathsf{id}\in[n]}, \mathsf{CT})}$. Parse $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}$ as $\mathsf{nce.sk}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}$ for each $\mathsf{id} \in [n]$ and CT as mafe.ct.

Compute $\{\mathsf{mafe.sk}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}\}_{\mathsf{id}\in[n]} = \mathsf{VNCE.Dec}(\{\mathsf{nce.sk}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}\}_{\mathsf{id}\in[n]})$. If this aborts, then abort and output $\perp$.

Otherwise, compute and output $y = \mathsf{asMAFE.Dec}(\{\mathsf{mafe.sk}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}\}_{\mathsf{id}\in[n]}, \mathsf{mafe.ct})$.

**Correctness.** Correctness of Construction 6.10 follows from correctness of asMAFE and VNCE.

**Theorem 6.11.** If asMAFE is an MAFE scheme with synchronous key generation for P/Poly circuits and $\mathcal{GID}$ (Definition 6.2), and VNCE is a vector non-committing encryption scheme for $\mathcal{GID}$ (Definition 6.9), then Construction 6.10 is an MAFE scheme (Definition 6.1) for P/Poly circuits and $\mathcal{GID}$.

*Proof Sketch.* We provide a sketch of the hybrid argument used to prove security of Construction 6.10 as follows.

**Hybrid 0.** This is the honest experiment with challenger with $b = 0/1$.

**Hybrid 1.** In this hybrid, we simulate the VNCE instantiation. This way, all secret keys will only be revealed after all the authorities are queried.

*Hybrids 0 and 1 are indistinguishable by VNCE security.*

**Hybrid 2.** In this hybrid, we encrypt $C_{1-b}$ using asMAFE.

*Indistinguishability of Hybrids 1 and 2 can be argued by adaptive synchronous key generation security of* asMAFE.

**Hybrid 3.** This is honest experiment with $1 - b$. Hybrid 2 and 3 are indistinguishable by VNCE security.

We provide full proof in Appendix I. □

# 7 Collusion Resistant MAFE with Malicious Authorities

In this section we construct an MAFE scheme (recall by MAFE, we mean MAFE in the collusion-resistant setting) with malicious authorities. We consider the setting where $k$ out of $n$ authorities are malicious such that $\binom{n}{k} = \mathsf{poly}(\lambda)$. Moreover, in our construction, we require the pre-challenge secret key queries to be bounded. We provide full definition in Section 7.1. Our construction uses FSS with strong simulation security and is provided in Section 7.2

## 7.1 Definition

We formally define an MAFE with malicious authorities where at most $k$ out of $n$ authorities can be malicious ($(n,k)$-cMAFE) below.

**Definition 7.1.** An $(n,k)$-cMAFE scheme is syntactically almost same as MAFE scheme. Except, we now take a *pre-challenge collusion bound* $Q$, maximum number of malicious authorities $k$, maximum number of corrupt bits $\chi$ as input to the $\mathsf{GSetup}$ algorithm. Security is altered to accommodate malicious corruptions as follows.

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

$$\left|\Pr\left[\begin{array}{ll} & (Q,n,s,k,\chi,\ell_1,\ldots,\ell_n) \leftarrow \mathcal{A}(1^\lambda), L = \max_{\mathsf{id}}\{\ell_{\mathsf{id}}\} \\ & \mathsf{CRS} \leftarrow \mathsf{GSetup}(1^\lambda, 1^Q, 1^n, 1^s, 1^k, 1^\chi, 1^L), \\ b \leftarrow \mathcal{A}^{\mathcal{O}(\cdot,\cdot)}(\mathsf{CT}) & : \forall\ \mathsf{id} \in [n], (\mathsf{MPK}_{\mathsf{id}}, \mathsf{MSK}_{\mathsf{id}}) \leftarrow \mathsf{ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}}), \\ & (C_0, C_1) \leftarrow \mathcal{A}^{\mathcal{O}(\cdot,\cdot)}(\mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}), \\ & b \xleftarrow{\$} \{0,1\}, \mathsf{CT} \leftarrow \mathsf{Enc}(\{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}, C_b) \end{array}\right] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda)$$

where $\mathcal{O}$ is a stateful oracle that responds to two types of queries from $\mathcal{A}$

$(\mathsf{KGen}, (\mathsf{id}, \mathsf{GID}, x))$**.** $\mathcal{A}$ sends $(\mathsf{id}, \mathsf{GID}, x) \in [n] \times \mathcal{GID} \times \{0,1\}^{\ell_{\mathsf{id}}}$ 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 $\mathcal{O}$.

$(\mathsf{Corr}, \mathsf{id})$**.** $\mathcal{A}$ sends $\mathsf{id} \in [n]$. $\mathcal{O}$ responds with $\mathsf{MSK}_{\mathsf{id}}$.

A stateful PPT machine $\mathcal{A}$ is said to be an admissible adversary if $|C_0| = |C_1|$ and are both of size at most $s$ and for each $(\mathsf{GID}, X) \in \mathcal{V}$, $C_0(X) = C_1(X)$. Here $\mathcal{V} = \{(\mathsf{GID}, X) : X \in \mathbf{X}_{\mathsf{GID},1} \times \ldots \mathbf{X}_{\mathsf{GID},n}$ for every $\mathsf{GID}$ that was used to query all authorities. If $\mathsf{id}$-th authority corrupted, then $\mathbf{X}_{\mathsf{GID},\mathsf{id}} = \{0,1\}^{\ell_{\mathsf{id}}}$. Otherwise, $\mathbf{X}_{\mathsf{GID},\mathsf{id}}$ is a singleton $x$ corresponding to the query $(\mathsf{KGen}, (\mathsf{id}, \mathsf{GID}, x))\}$. Moreover, $\mathcal{V}_{\mathsf{pre}} \subseteq \mathcal{V}$ is the set of pre-challenge queries for a particular $\mathsf{GID}$ for which all non-malicious authorities are queried. Then, $|\mathcal{V}_{\mathsf{pre}}| \leq Q$[13].

## 7.2 Construction

**Construction 7.2.** We provide the construction of a $(n,k)$-cMAFE scheme for P/Poly circuits and $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda\in\mathbb{N}}$ (Definition 7.1) using MAFE scheme for P/Poly circuits and $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda\in\mathbb{N}}$ (Definition 6.1) and a FSS scheme with strong security for P/Poly circuits (Definition 3.12), and a SKE scheme (Definition 3.3) as follows.

[13]Note that $\mathcal{A}$ can still make unbounded pre-challenge queries. It just makes $Q$ queries with a unique $\mathsf{GID}$ each such that for this $\mathsf{GID}$ all non-malicious authorities are queried.

$\underline{\mathsf{GSetup}(1^\lambda, 1^Q, 1^n, 1^s, 1^k, 1^\chi, 1^L)}$. Let $\mathbb{I} = \left[\binom{n}{k}\right]$ and $\mathbb{J} = [2^\chi]$. We will assume some fixed ordering of $2^{[n]}\big|_k$ and say $\mathcal{K}_i \in 2^{[n]}\big|_k$ for $i \in \mathbb{I}$. Let $\ell_{\mathsf{out}}$ represent the SKE encryption size of a message of output length of $\mathcal{C}_s$. For each $i \in \mathbb{I}, j \in \mathbb{J}$, instantiate an $\mathsf{MAFE}$,

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

We assume w.l.o.g that for any $i \in \mathbb{I}$, the authority identifiers are elements of the set $[n] \setminus \mathcal{K}_i$, i.e, for the $(i,j)$-th instantiation of $\mathsf{MAFE}$, $\mathsf{id} \notin \mathcal{K}_i$. 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 $\mathsf{MAFE}$ if $\mathsf{id} \notin \mathcal{K}_i$, $(\mathsf{mpk}^{(i,j)}, \mathsf{msk}^{(i,j)}) \leftarrow \mathsf{MAFE.ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}+\lambda+\ell_{\mathsf{out}}})$. 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 $\ell_{\mathsf{idx}}$. Using $(\ell_1, \ldots, \ell_n)$, define a maximal corruptible set $\mathcal{K}' \subseteq [n]$ such that $\ell_{\mathsf{id}} = O(\log \lambda)$ for $\mathsf{id} \in \mathcal{K}'$. This is done to figure out the exact set of authorities which can be corrupted which may be less than $k$.

Let $\mathbb{I}' = \{i : i \in \mathbb{I}, \mathcal{K}_i \subseteq \mathcal{K}', \sum_{\mathsf{idx}\in\mathcal{K}_i} \ell_{\mathsf{idx}} \leq \chi\}$. Let $\mathbb{I}'_{\mathsf{id}} = \{i : i \in \mathbb{I}', \mathsf{id} \notin \mathcal{K}_i\}$. For every $i \in \mathbb{I}'_{\mathsf{id}}, j \in \mathbb{J}$ sample $r_{i,j} \xleftarrow{\$} \{0,1\}^\lambda, \mathsf{ske.ct}^*_{ij} \xleftarrow{\$} \{0,1\}^{\ell_{\mathsf{out}}}$ where $\ell_{\mathsf{out}}$ is the output size of circuits in $\mathcal{C}_s$ and

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

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 $\mathsf{KGen}$. Share the circuit $C$ using $\mathsf{FSS}$ for all $i \in \mathbb{I}'$, $\{\widetilde{C}_i\}_{i\in\mathbb{I}'} \leftarrow \mathsf{FSS.Share}(1^\lambda, 1^{|\mathbb{I}'|}, C)$. For each $i \in \mathbb{I}'$, encrypt the circuit $F_{ij}$ from Figure 5 after padding it to the size of circuit from Figure 9.

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

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

**Circuit $F_{ij}$**

**Input:** $x_1, r_1, \mathsf{ske.ct}^*_1, \ldots, x_{n-k}, r_{n-k}, \mathsf{ske.ct}^*_{n-k}$
**Hardwired:** $i, j, \widetilde{C}_i$
**Output:** Parse $x_1, \ldots, x_{n-k}$ as $(\ell_1 + \ldots + \ell_n) - \chi$ bits of input for $\{x_{\mathsf{id}}\}_{\mathsf{id} \notin \mathcal{K}_i}$ and set $\{x_{\mathsf{id}}\}_{\mathsf{id} \in \mathcal{K}_i} = j$ (truncations done appropriately). Compute and output $\mathsf{FSS.Eval}(i, \widetilde{C}_i, x_1, \ldots, x_n)$.

Figure 5: Description of circuit $F_{ij}$

**Theorem 7.3.** If FSS is a secure FSS scheme for P/Poly circuits (Definition 3.12), MAFE is a secure MAFE scheme for P/Poly circuits and $\mathcal{GID}$ (Definition 6.1), SKE with pseudorandom ciphertexts (Definition 3.3), then Construction 7.2 is a secure $(n, k)$-cMAFE scheme for P/Poly circuits and $\mathcal{GID}$ (Definition 7.1).

*Proof Sketch.* We provide a sketch of the hybrid argument used to prove security of Construction 7.2 as follows.

**Hybrid 0.** This is the honest experiment from Definition 7.1 with $b = 0/1$.

**Hybrid 1.** In this hybrid, we sample $i^* \xleftarrow{\$} \mathbb{I}'$ and abort if any $\mathsf{id} \notin \mathcal{K}_{i^*}$ is corrupted.

**Hybrid 2.** In this hybrid, for each GID query, we will sample $\rho_{i^*j}$ and set $r^{(\mathsf{id})}_{i^*j}$ used in secret key generation for id-th authority to be such that $\bigoplus_{\mathsf{id}} r^{(\mathsf{id})}_{i^*j} = \rho_{i^*j}$ and also ensure $\rho_{i^*j}$ for each GID and $j$ are unique.

**Hybrid 3.** In this hybrid, we will set $\mathsf{ske.ct}^*_{i^*j}$ for each id to be encryptions of additive secret shares of $\mathsf{FSS.Eval}(i^*, \widetilde{C}_{i^*}, \{\{x_{\mathsf{id}}\}_{\mathsf{id} \notin \mathcal{K}_{i^*}}, j\})$. That is, for a specific GID, $\mathsf{ske.ct}^*_1, \ldots, \mathsf{ske.ct}^*_n$ are encryptions of $\mathsf{sh}_1, \ldots, \mathsf{sh}_n$ such that $\bigoplus_{\mathsf{id}} \mathsf{sh}_{\mathsf{id}} = \mathsf{FSS.Eval}(i^*, \widetilde{C}_{i^*}, \{\{x_{\mathsf{id}}\}_{\mathsf{id} \notin \mathcal{K}_{i^*}}, j\})$.

*Indistinguishability follows by SKE security.*

**Hybrid** $4, \kappa$**.** For $\kappa \in [|\mathbb{J}| + 1]$. In this hybrid, we will encrypt a circuit $F'_{i^*j}$ for $j \in [\kappa - 1]$ instantiations of MAFE. Here $F'_{i^*j}$ decrypts $\mathsf{ske.ct}^*$ and uses $r_{\mathsf{id}}$ to output values instead of using $\widetilde{C}_{i^*}$.

*The indistinguishability of hybrids can be argued using the security of* MAFE.

**Hybrid 5.** In this hybrid, we will simulate the FSS instantiation. Now that all MAFE instantiations for $i^*$ (after correctly guessing at the beginning) do not use $\widetilde{C}_{i^*}$, we would only be using the outputs of share evaluation.

*We can readily rely on the strong security of* FSS *to show indistinguishability. In this hybrid,* $\mathcal{A}$ *has* 0 *advantage.*

The full proof is in Appendix J. □

# 8 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 $\chi$-bit corruptions, the size of its ciphertexts and master keys *must* grow with $2^{\chi}$. 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[14]. This result implies that Construction 5.3 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 8.5) 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 $\mathsf{GID}$ query and a weakly succinct FE scheme for $\mathsf{P}/\mathsf{Poly}$ [BV15] if it supports at most $Q$ $\mathsf{GID}$ queries. Our implications are formally proved in Section 8.3 and Section 8.4. Note that an XiO scheme for $\mathsf{P}^{\log}/\mathsf{Poly}$ in conjunction with $\mathsf{LWE}$ implies an iO scheme whereas a weakly succinct FE scheme for $\mathsf{P}/\mathsf{Poly}$ implies it generically. We provide the definitions of these preliminaries in Section 8.1 and formally define statically secure MAFE schemes in Section 8.2.

## 8.1 Preliminaries

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

**Definition 8.1** (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 $\mathsf{PPT}$ adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $\forall\lambda \in \mathbb{N}, C_0, C_1 \in \mathcal{C}_\lambda, |C_0| = |C_1|, \forall x, C_0(x) = C_1(x)$,

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

**Definition 8.2** (XiO)**.** An exponentially-efficient indistinguishability obfuscation (XiO) scheme $\mathsf{Xi}\mathcal{O}$ 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 8.1 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 $\mathsf{Obf}$ runs in time $\mathsf{poly}(\lambda, |C|, 2^n)$ and output size $|\widetilde{C}| = \mathsf{poly}(\lambda, |C|) \cdot 2^{n(1-\xi)}$.

[14] We consider indistinguishability-based security for our schemes in this section and simulation-based security is impossible for such schemes (cf. Remark 8.13).

**Definition 8.3** (Circuit class $\mathsf{P}^{\log}/\mathsf{Poly}$, [LPST16])**.** $\mathsf{P}^{\log}/\mathsf{Poly}$ is a subclass of $\mathsf{P}/\mathsf{Poly}$ 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 8.4** (sFE, [BV15])**.** An $\mathsf{sFE}$ 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 $x$,

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

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

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

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

## 8.2 Definitions

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

**Syntax.** A static $Q$-$\mathsf{GID}$ 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 $\chi$ bits with non-trivial efficiency consists of algorithms $(\mathsf{GSetup}, \mathsf{ASetup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ defined similar to Definition 5.1.

**Definition 8.5** (stMAFE)**.** An $\mathsf{stMAFE}$ 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 $\lambda \in \mathbb{N}, Q = Q(\lambda), n = n(\lambda), s = s(\lambda), k < n, \chi = \chi(\lambda), C \in \mathcal{C}_\lambda, \mathsf{GID} \in \mathcal{GID}_\lambda, \forall\ \mathsf{id} \in [n], \ell_{\mathsf{id}} = \ell_{\mathsf{id}}(\lambda)$,

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

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 $\forall\ \lambda \in \mathbb{N}$,

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

where $\mathsf{CRS} \leftarrow \mathsf{GSetup}(1^\lambda, 1^Q, 1^n, 1^s, 1^k, 1^\chi)$ and $\mathsf{KGen}$ 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 8.6** (fst1MAFE)**.** A $\mathsf{fst1MAFE}$ scheme $(\mathsf{GSetup}, \mathsf{ASetup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ 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 $\mathsf{GID}$ 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 $\mathsf{CRS}$, $\mathsf{MSK}_{\mathsf{id}}$, $\mathsf{MPK}_{\mathsf{id}}$ for any $\mathsf{id} \in [n]$, $\mathsf{SK}_{\mathsf{id},x}, x \in \{0,1\}^{\ell_{\mathsf{id}}}$, and $\mathsf{CT}$ 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 $\mathsf{GSetup}, \mathsf{ASetup}, \mathsf{KGen}, \mathsf{Enc}$ are at most $\mathsf{poly}(\lambda, \ell_1, \ldots, \ell_n, |C|, 2^\chi)$.

## 8.3 XiO from fst1MAFE

**Construction 8.7** ($\mathsf{Xi}\mathcal{O}$)**.** We provide the construction of an XiO scheme (Definition 8.2) for $\mathsf{P}^{\log}/\mathsf{Poly}$ circuits (Definition 8.3) $\mathcal{C} = \{\mathcal{C}_\lambda : \{0,1\}^{n(\lambda)} \to \mathcal{O}_\lambda\}_{\lambda\in\mathbb{N}}$ from an $\mathsf{fst1MAFE}$ scheme for $\mathsf{P}/\mathsf{Poly}$ circuits (Definition 8.6) 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 $\mathsf{fst1MAFE}$ instantiation with $N = 2$ authorities, $k = 1$ corrupts where $\ell_1 = n, \ell_2 = \mathsf{poly}(\lambda), \chi = \ell_1$,

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

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

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

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

**Non-trivial Efficiency.** As all algorithms of $\mathsf{fst1MAFE}$ used in $\mathsf{Obf}$ run in $\mathsf{poly}(\lambda, n, \ell_2, |C|, 2^n)$, running time of $\mathsf{Obf}$ 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 8.8.** If $\mathsf{fst1MAFE}$ is a fully static 1-$\mathsf{GID}$ $\mathsf{MAFE}$ scheme for $\mathsf{P/Poly}$ circuits that can handle $O(\log \lambda)$ corruptions (Definition 8.6), then Construction 8.7 is an $\mathsf{X}i\mathcal{O}$ scheme for $\mathsf{P}^{\log}/\mathsf{Poly}$ circuits (Definition 8.2).

*Proof.* We prove the security of Construction 8.7 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 $\epsilon$. 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 $\mathsf{Chal}$ 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 $\mathsf{Chal}$. Parse the information sent by $\mathsf{Chal}$ 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 $\lambda$. In addition, $\mathcal{B}$ acts as a valid adversary for $\mathsf{Chal}$ and a valid challenger for $\mathcal{A}$. Advantage of $\mathcal{B}$ is exactly $\epsilon$. □

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 8.9** ([LPST16])**.** Assuming the existence of sub-exponentially secure $\mathsf{X}i\mathcal{O}$ for $\mathsf{P}^{\log}/\mathsf{Poly}$ and sub-exponentially secure $\mathsf{LWE}$, there exists a sub-exponentially secure $i\mathcal{O}$ scheme for $\mathsf{P/Poly}$ circuits (Definition 8.1).

## 8.4 Weakly Size-Succinct FE from stMAFE

**Construction 8.10** ($\mathsf{sFE}$)**.** We provide the construction of a weakly size-succinct $\mathsf{FE}$ scheme for $\mathsf{P/Poly}$ circuits (Definition 8.4) from an $\mathsf{stMAFE}$ scheme (Definition 8.5) 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 $\mathsf{Enc}$.

Generate an $\mathsf{stMAFE}$ 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 $Q$ $\mathsf{GID}$s 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 $\mathsf{GID}_q, C_q$, $\mathsf{sk}_{2,q,C_q} \leftarrow \mathsf{KGen}(2, \mathsf{msk}_2, \mathsf{GID}_q, C_q)$.

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 $\mathsf{MPK}$ 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 $\mathsf{stMAFE}$, $\mathsf{ct} \leftarrow \mathsf{stMAFE.Enc}(\mathsf{mpk}_1, \mathsf{mpk}_2, F)$ and output $\mathsf{CT} := \mathsf{ct}$.

$\underline{\mathsf{Dec}(\mathsf{SK}, \mathsf{CT})}$**.** Parse $\mathsf{CT}$ as $\mathsf{ct}$ and $\mathsf{SK}$ as $(\mathsf{crs}, \mathsf{mpk}_1, \mathsf{mpk}_2, \mathsf{msk}_1, \mathsf{GID}_q, C_q, \mathsf{sk}_{2,q,C})$. Run $\mathsf{stMAFE}$ 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 $\mathsf{stMAFE}$.

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

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

**Theorem 8.11.** If $\mathsf{stMAFE}$ is a secure static fixed-corruptible $Q$-$\mathsf{GID}$ $\mathsf{MAFE}$ scheme for $\mathsf{P/Poly}$ circuits and $\mathcal{GID}$ (Definition 8.5), then Construction 8.10 is a secure weakly size-succinct $\mathsf{FE}$ scheme for $\mathsf{P/Poly}$ circuits.

*Proof.* We prove the security of Construction 8.10 by showing a reduction between any adversary that breaks the security of $\mathsf{sFE}$ and a challenger for $\mathsf{stMAFE}$. Assume that there exists a $t$-time adversary $\mathcal{A}$ that runs in time $t$ against $\mathsf{sFE}$ as per Definition 8.4 with advantage $\epsilon$. We prove the security of $\mathsf{sFE}$ by constructing a reduction $\mathcal{B}$ between $\mathcal{A}$ and a challenger $\mathsf{Chal}$ for $\mathsf{stMAFE}$ 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 $\mathsf{Chal}$. Receive $(\mathsf{crs}, \mathsf{mpk}_1, \mathsf{mpk}_2, \mathsf{msk}_1, \{\mathsf{sk}_{2,q,C_q}\}_q)$ from $\mathsf{Chal}$. 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 $\mathsf{Chal}$. Receive $\mathsf{CT}$ from $\mathsf{Chal}$ and run $\mathcal{A}$ on $\mathsf{CT}$. Output whatever $\mathcal{A}$ outputs. Note that the running time of $\mathcal{B}$ is polynomial in the running time of $\mathcal{A}$ and $\lambda$. In addition, $\mathcal{B}$ acts as a valid adversary for $\mathsf{Chal}$ and a valid challenger for $\mathcal{A}$. Advantage of $\mathcal{B}$ is exactly $\epsilon$. □

Thus, if we have a sub-exponentially secure $\mathsf{fst1MAFE}$ for $\mathsf{P/Poly}$ circuits, we can build a sub-exponentially secure $\mathsf{sFE}$ 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 $\mathsf{sFE}$ for $\mathsf{P/Poly}$ circuits.

**Theorem 8.12** ([BV15])**.** Assuming the existence of sub-exponentially secure $\mathsf{sFE}$ for $\mathsf{P/Poly}$ circuits, there exists a sub-exponentially secure $i\mathcal{O}$ scheme for $\mathsf{P/Poly}$ circuits.

**Remark 8.13** (Impossibilities for $\mathsf{SIM}$-security)**.** By an incompressibility argument similar to [BSW11, AGVW13], $\mathsf{SIM}$-secure $\mathsf{fst1MAFE}$ and $\mathsf{stMAFE}$ are impossible in the standard model.

## References


[ABSV15] Prabhanjan Ananth, Zvika Brakerski, Gil Segev, and Vinod Vaikuntanathan. From selective to adaptive security in functional encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 657–677, Santa Barbara, CA, USA, August 16–20, 2015. Springer Berlin Heidelberg, Germany.

[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.

[AJJM21] Prabhanjan Ananth, Abhishek Jain, Zhengzhong Jin, and Giulio Malavolta. Unbounded multi-party computation from learning with errors. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 754–781. Springer, 2021.

[AJS15] Prabhanjan Ananth, Abhishek Jain, and Amit Sahai. Indistinguishability obfuscation from functional encryption for simple functions. *Cryptology ePrint Archive*, 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.

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

[BCG+17] 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.

[BDOP04] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 506–522, Interlaken, Switzerland, May 2–6, 2004. Springer Berlin Heidelberg, Germany.

[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.

[BGI15] Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 337–367, Sofia, Bulgaria, April 26–30, 2015. Springer Berlin Heidelberg, Germany.

[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.

[BJKL21] Fabrice Benhamouda, Aayush Jain, Ilan Komargodski, and Huijia Lin. Multiparty reusable non-interactive secure computation from LWE. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 724–753. Springer, 2021.

[BL20] Fabrice Benhamouda and Huijia Lin. Mr NISC: Multiparty reusable non-interactive secure computation. In Rafael Pass and Krzysztof Pietrzak, editors, *Theory of Cryptography*, pages 349–378, Cham, 2020. Springer International Publishing.

[BLM+24] Pedro Branco, Russell W. F. Lai, Monosij Maitra, Giulio Malavolta, Ahmadreza Rahimi, and Ivy K. Y. Woo. Traitor tracing without trusted authority from registered functional encryption. In Kai-Min Chung and Yu Sasaki, editors, *Advances in Cryptology – ASIACRYPT 2024, Part III*, volume 15486 of *Lecture Notes in Computer Science*, pages 33–66, Kolkata, India, December 9–13, 2024. Springer, Singapore, Singapore.

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

[BSW06] Dan Boneh, Amit Sahai, and Brent Waters. Fully collusion resistant traitor tracing with short ciphertexts and private keys. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages

573–592, St. Petersburg, Russia, May 28 – June 1, 2006. Springer Berlin Heidelberg, Germany.

[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.

[BW07] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In *Theory of Cryptography: 4th Theory of Cryptography Conference, TCC 2007, Amsterdam, The Netherlands, February 21-24, 2007. Proceedings 4*, pages 535–554. Springer, 2007.

[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.

[CDSG+18] Jérémy Chotard, Edouard Dufour Sans, Romain Gay, Duong Hieu Phan, and David Pointcheval. Decentralized multi-client functional encryption for inner product. In *Advances in Cryptology–ASIACRYPT 2018: 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2–6, 2018, Proceedings, Part II 24*, pages 703–732. Springer, 2018.

[CFGN96] Ran Canetti, Uriel Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In *28th Annual ACM Symposium on Theory of Computing*, pages 639–648, Philadephia, PA, USA, May 22–24, 1996. ACM Press.

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

[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.

[Cho09] Sherman S. M. Chow. Removing escrow from identity-based encryption. In Stanislaw Jarecki and Gene Tsudik, editors, *PKC 2009: 12th International Conference on Theory and Practice of Public Key Cryptography*, volume 5443 of *Lecture Notes in Computer Science*, pages 256–276, Irvine, CA, USA, March 18–20, 2009. Springer Berlin Heidelberg, Germany.

[CMMS25] Anirban Chakrabarti, Monosij Maitra, Arup Mondal, and Kushaz Sehgal. Silent threshold traitor tracing & enhancing mempool privacy. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*, CCS '25, page 1769–1783, New York, NY, USA, 2025. Association for Computing Machinery.

[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.

[Des88] Yvo Desmedt. Society and group oriented cryptography: A new concept. In Carl Pomerance, editor, *Advances in Cryptology – CRYPTO'87*, volume 293 of *Lecture Notes in Computer Science*, pages 120–127, Santa Barbara, CA, USA, August 16–20, 1988. Springer Berlin Heidelberg, Germany.

[DF90] Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 307–315, Santa Barbara, CA, USA, August 20–24, 1990. Springer, New York, USA.

[DIJ+13] Angelo De Caro, Vincenzo Iovino, Abhishek Jain, Adam O'Neill, Omer Paneth, and Giuseppe Persiano. On the achievability of simulation-based security for functional encryption. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 519–535, Santa Barbara, CA, USA, August 18–22, 2013. Springer Berlin Heidelberg, Germany.

[DJWW25] Lalita Devadas, Abhishek Jain, Brent Waters, and David J. Wu. Succinct witness encryption for batch languages and applications. In Goichiro Hanaoka and Bo-Yin Yang, editors, *Advances in Cryptology – ASIACRYPT 2025, Part VIII*, volume 16252 of *Lecture Notes in Computer Science*, pages 130–162, Melbourne, VIC, Australia, December 8–12, 2025. Springer, Singapore, Singapore.

[DST26] Pratish Datta, Robert Schädlich, and Erkan Tairi. Decentralized multi-authority (attribute-based) traitor tracing. Cryptology ePrint Archive, Paper 2026/1268, 2026.

[FGH+17] Cody Freitag, Rishab Goyal, Susan Hohenberger, Venkata Koppula, Eysa Lee, Tatsuaki Okamoto, Jordan Tran, and Brent Waters. Signature schemes with randomized verification. In *International Conference on Applied Cryptography and Network Security*, pages 373–389. Springer, 2017.

[Fra90] Yair Frankel. A practical protocol for large group oriented networks. In Jean-Jacques Quisquater and Joos Vandewalle, editors, *Advances in Cryptology – EUROCRYPT'89*, volume 434 of *Lecture Notes in Computer Science*, pages 56–61, Houthalen, Belgium, April 10–13, 1990. Springer Berlin Heidelberg, Germany.

[GGG+14] Shafi Goldwasser, S. Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. Multi-input functional encryption. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology*

– *EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 578–602, Copenhagen, Denmark, May 11–15, 2014. Springer Berlin Heidelberg, Germany.

[GGH+16] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. *SIAM Journal on Computing*, 45(3):882–929, 2016.

[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.

[GGM84] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions (extended abstract). In *25th Annual Symposium on Foundations of Computer Science*, pages 464–479, Singer Island, Florida, October 24–26, 1984. IEEE Computer Society 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.

[GHRW14] Craig Gentry, Shai Halevi, Mariana Raykova, and Daniel Wichs. Outsourcing private RAM computation. In *55th Annual Symposium on Foundations of Computer Science*, pages 404–413, Philadelphia, PA, USA, October 18–21, 2014. IEEE Computer Society Press.

[GKPW24] Sanjam Garg, Dimitris Kolonelos, Guru-Vamsi Policharla, and Mingyuan Wang. Threshold encryption with silent setup. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology – CRYPTO 2024, Part VII*, volume 14926 of *Lecture Notes in Computer Science*, pages 352–386, Santa Barbara, CA, USA, August 18–22, 2024. Springer, Cham, Switzerland.

[GKW17] Rishab Goyal, Venkata Koppula, and Brent Waters. Lockable obfuscation. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017*, pages 612–621, 2017.

[GKW18] Rishab Goyal, Venkata Koppula, and Brent Waters. Collusion resistant traitor tracing from learning with errors. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 660–670, 2018.

[GLSW08] Vipul Goyal, Steve Lu, Amit Sahai, and Brent Waters. Black-box accountable authority identity-based encryption. In Peng Ning, Paul F. Syverson, and Somesh Jha, editors, *ACM CCS 2008: 15th Conference on Computer and Communications Security*, pages 427–436, Alexandria, Virginia, USA, October 27–31, 2008. ACM Press.

[Gol04] Oded Goldreich. *Foundations of Cryptography: Basic Applications*, volume 2. Cambridge University Press, Cambridge, UK, 2004.

[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.

[GSY26] Rishab Goyal, Alex Snyder, and Saikumar Yadugiri. Decentralizing traitor tracing: A multi-authority approach. Cryptology ePrint Archive, Paper 2026/1076, 2026.

[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.

[HKM+24] Taiga Hiroka, Fuyuki Kitagawa, Tomoyuki Morimae, Ryo Nishimaki, Tapas Pal, and Takashi Yamakawa. Certified everlasting secure collusion-resistant functional encryption, and more. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 434–456. Springer, 2024.

[HLWW23] Susan Hohenberger, George Lu, Brent Waters, and David J. Wu. Registered attribute-based encryption. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023, Part III*, volume 14006 of *Lecture Notes in Computer Science*, pages 511–542, Lyon, France, April 23–27, 2023. Springer, Cham, Switzerland.

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

[JLS22] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from lpn over f p, dlin, and prgs in nc 0. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 670–699. Springer, 2022.

[KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *Advances in Cryptology–EUROCRYPT 2008: 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings 27*, pages 146–162. Springer, 2008.

[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.

[Luo24] Ji Luo. Ad hoc broadcast, trace, and revoke: Plus time-space trade-offs for attribute-based encryption. *IACR Communications in Cryptology (CiC)*, 1(2):15, 2024.

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

[LWW25] George Lu, Brent Waters, and David J. Wu. Multi-authority registered attribute-based encryption. In Serge Fehr and Pierre-Alain Fouque, editors, *Advances in Cryptology – EUROCRYPT 2025, Part III*, volume 15603 of *Lecture Notes in Computer Science*, pages 3–33, Madrid, Spain, May 4–8, 2025. Springer, Cham, Switzerland.

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

[RVV24] Seyoon Ragavan, Neekon Vafa, and Vinod Vaikuntanathan. Indistinguishability obfuscation from bilinear maps and LPN variants. In Elette Boyle and Mohammad Mahmoody, editors, *TCC 2024: 22nd Theory of Cryptography Conference, Part IV*, volume 15367 of *Lecture Notes in Computer Science*, pages 3–36, Milan, Italy, December 2–6, 2024. Springer, Cham, Switzerland.

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

[Shi22] Sina Shiehian. mrnisc from lwe with polynomial modulus. In *International Conference on Security and Cryptography for Networks*, pages 481–493. Springer, 2022.

[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.

[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.

[Wat15] Brent Waters. A punctured programming approach to adaptively secure functional encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 678–697, Santa Barbara, CA, USA, August 16–20, 2015. Springer Berlin Heidelberg, Germany.

[WQT18] Quanyun Wei, Fang Qi, and Zhe Tang. Remove key escrow from the bf and gentry identity-based encryption with non-interactive key generation. *Telecommunication Systems*, 69(2):253–262, 2018.

[WW25] Hoeteck Wee and David J. Wu. Unbounded distributed broadcast encryption and registered ABE from succinct LWE. In Yael Tauman Kalai and Seny F. Kamara, editors, *Advances in Cryptology – CRYPTO 2025, Part III*, volume 16002 of *Lecture Notes in Computer Science*, pages 204–235, Santa Barbara, CA, USA, August 17–21, 2025. Springer, Cham, Switzerland.

[WW26] Brent Waters and David J. Wu. Silent threshold cryptography from pairings: Expressive policies in the plain model. Cryptology ePrint Archive, Paper 2025/1547, To appear in Eurocrypt, 2026.

[WZ17] Daniel Wichs and Giorgos Zirdelis. Obfuscating compute-and-compare programs under LWE. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017*, pages 600–611, 2017.

[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 Relevant Material from [GY24]

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

## A.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.

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

$\mathsf{CliEnc}(1^\lambda, 1^Q, 1^n, 1^s, C) \to \{\widehat{C}^u\}_{u\in[N]}$. 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 $\{\widehat{C}^u\}_{u\in[N]}$.

$\mathsf{UserComp}(\{\widehat{x}^u_{\mathsf{id},\mathsf{GID}}\}_{\mathsf{id}\in[n]}, \widehat{C}^u) \to \widehat{y}^u_{\mathsf{GID}}$. 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 $\widehat{y}^u_{\mathsf{GID}}$.

$\mathsf{Decode}(\{\widehat{y}^u_{\mathsf{GID}}\}_{u\in\mathbf{S}}, \mathbf{S}) \to y$. 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 A.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 $\lambda \in \mathbb{N}, Q = Q(\lambda), n = n(\lambda), s = s(\lambda), \mathsf{id} \in [n], C \in \mathcal{C}_\lambda, \mathsf{GID} \in \mathcal{GID}_\lambda, u \in \mathbf{S} \subseteq [N], |\mathbf{S}| = D$,

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

**Security.** For any admissible adversary $\mathcal{A}$, there exists a stateful simulator $\mathsf{Sim}$ 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, $\mathbf{S}_{\mathsf{GID}}, \boldsymbol{\Delta}_{\mathsf{GID}}$ should be unique for a GID. And all $\{\mathbf{S}_{\mathsf{GID}}\}$ and $\{\boldsymbol{\Delta}_{\mathsf{GID}}\}$ satisfy Lemma A.3 and Lemma A.4 respectively. $\widetilde{\mathbf{S}} = [N] \setminus (\bigcup_{\mathsf{GID}\neq\mathsf{GID}'} \mathbf{S}_{\mathsf{GID}} \cap \mathbf{S}_{\mathsf{GID}'})$ and $j^*_{\mathsf{GID}}$ is the unique index in $\boldsymbol{\Delta}_{\mathsf{GID}}$ for every GID.

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

**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 $((\mathsf{id}, \mathsf{GID}, x, \boldsymbol{\Delta}_{\mathsf{GID}}), \mathbf{S}_{\mathsf{GID}})$. Chal runs $\{\widehat{x}^u_{\mathsf{id},\mathsf{GID}}\}_{u\in[N]} = \mathsf{ServEnc}(1^\lambda, 1^Q, 1^n, 1^s, \mathsf{id}, \mathsf{GID}, x, \boldsymbol{\Delta}_{\mathsf{GID}})$, 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 $j^*_{\mathsf{GID}}$ 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 UserComp.

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

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

**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 $\mathsf{GID}$ 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 $\mathsf{GID}$ 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 $j^*_{\mathsf{GID}}$ when for this $\mathsf{GID}$, 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, $\mathsf{Sim}$ queries $C(\cdot)$ with $X = (x_{1,\mathsf{GID}}, \ldots, x_{n,\mathsf{GID}})$ and receives $C(X)$.

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

**Theorem A.2** ([GY24])**.** Assuming the existence of one-way functions, there exists a $\mathsf{dCSF}$ scheme for $\mathsf{P/Poly}$ circuits.

### A.2 Augmented Statistical Lemmas

We provide the following lemmas from [GY24] used in Section 4.2.

**Lemma A.3** (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 A.4** (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,

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

## B Proofs from Section 4

In this section, we provide full proof of Theorem 4.10.

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

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

**Auth Setup Queries.** $\mathcal{A}$ sends $(\mathsf{id}, \ell_{\mathsf{id}})$. $\mathsf{Chal}$ samples $(\mathsf{bfe.mpk}_{\mathsf{id}}, \mathsf{bfe.msk}_{\mathsf{id}}) \leftarrow \mathsf{BFE.Setup}(1^\lambda, 1^Q, 1^{s'})$. Set $\mathsf{MPK}_{\mathsf{id}} = (\ell_{\mathsf{id}}, \mathsf{bfe.mpk}_{\mathsf{id}})$, and $\mathsf{MSK}_{\mathsf{id}} = (\ell_{\mathsf{id}}, \mathsf{bfe.msk}_{\mathsf{id}})$. $\mathsf{Chal}$ 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 $\mathsf{mafe.crs}, \{(\mathsf{mafe.mpk}_i, \mathsf{mafe.msk}_i)\}_{i\in[n]} \leftarrow \mathsf{ssMAFE.GSetup}(1^\lambda, 1^Q, 1^n, 1^s, 1^{\ell_{\mathsf{id}_1}}, \ldots, 1^{\ell_{\mathsf{id}_n}})$.
- $\forall\, i \in [n]$, $\mathsf{bfe.ct}_{\mathsf{id}_i} \leftarrow \mathsf{BFE.Enc}(\mathsf{bfe.mpk}_{\mathsf{id}_i}, \mathsf{ssMAFE.KGen}(\cdot, \mathsf{mafe.msk}_i, \{\mathsf{mafe.mpk}_{i'}\}_{i'}, \cdot, \cdot))$.
- Sample $\mathsf{mafe.ct} \leftarrow \mathsf{ssMAFE.Enc}(\{\mathsf{mafe.mpk}_i\}_{i\in[n]}, C)$.

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

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 ssMAFE. 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 $\mathsf{mafe.crs}, \{\mathsf{mafe.mpk}_{\mathsf{id}}\}_{\mathsf{id}\in[n]} \leftarrow \mathsf{ssMAFE.Sim}(1^\lambda, 1^Q, 1^n, 1^s, 1^{\ell_{\mathsf{id}_1}}, \ldots, 1^{\ell_{\mathsf{id}_n}})$.
- $\forall\, i \in [n]$, $\mathsf{bfe.ct}_{\mathsf{id}_i} \leftarrow \mathsf{BFE.S}_2(\mathsf{st}_{\mathsf{id}_i}, 1^{|F_i|}, \mathcal{V}_i)$. Here $\mathcal{V}_i = \{(X, \mathsf{ssMAFE.Sim}(\mathsf{id}_i, \mathsf{GID}, x_{\mathsf{GID},\mathsf{id}_i})) :$ for all GID queries in previous phase$\}$.
- Sample $\mathsf{mafe.ct} \leftarrow \mathsf{ssMAFE.Sim}(1^{|C|}, \mathcal{V}_C)$ where $\mathcal{V}_C$ is as defined in Definition 4.3.

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 $\mathsf{ssMAFE.Sim}(\mathsf{id}, \mathsf{GID}, x, \mathsf{V}_C)$ where $\mathsf{V}_C$ is as defined in Definition 4.3.

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

**Lemma B.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 $\mathsf{BFE}$ where pre-challenge simulator $(\mathsf{BFE.S}_0, \mathsf{BFE.S}_1)$ behave exactly as $\mathsf{BFE.Setup}, \mathsf{BFE.KGen}$ respectively. □

**Lemma B.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 $\mathsf{BFE}$ in $\mathbf{Hyb}_{2,1}^{\mathcal{A}}(1^\lambda)$, these hybrids are identically distributed. □

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

*Proof.* Assume that there exists a $\mathsf{PPT}$ 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 $\mathsf{BFE}$ 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 $\mathsf{mafe.crs}, \{(\mathsf{mafe.mpk}_i, \mathsf{mafe.msk}_i)\}_{i\in[n]} \leftarrow \mathsf{ssMAFE.GSetup}(1^\lambda, 1^Q, 1^n, 1^s, 1^{\ell_{\mathsf{id}_1}}, \ldots, 1^{\ell_{\mathsf{id}_n}})$.
- $\forall\ i \in [n]$, 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 $F_i = \mathsf{ssMAFE.KGen}(\cdot, \mathsf{mafe.msk}_i, \{\mathsf{mafe.mpk}_{i'}\}_{i'}, \cdot, \cdot)$ and $\mathcal{V}_i = \{(X, F_i(X)) : \forall\ X = (\mathsf{id}_i, \mathsf{GID}, x_{\mathsf{GID},\mathsf{id}_i})\}$.
- Sample $\mathsf{mafe.ct} \leftarrow \mathsf{ssMAFE.Enc}(\{\mathsf{mafe.mpk}_i\}_{i\in[n]}, C)$.

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 $(\mathsf{bfe.mpk}_{\mathsf{id}}, \mathsf{bfe.msk}_{\mathsf{id}}) \leftarrow \mathsf{BFE.Setup}(1^\lambda, 1^Q, 1^{s'})$. 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 $\lambda$. If $\mathcal{O}$ is $(\mathsf{BFE.S}_0, \mathsf{BFE.S}_1, \mathsf{Enc}, \mathsf{KGen})$ for BFE, $\mathcal{B}$ behaves as $\mathbf{Hyb}^{\mathcal{A}}_{2,j}(1^\lambda)$ and if $\mathcal{O}$ is a $\mathsf{BFE.Sim}$, $\mathcal{B}$ behaves like $\mathbf{Hyb}^{\mathcal{A}}_{2,j+1}(1^\lambda)$. 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, $\mathbf{Hyb}^{\mathcal{A}}_{2,j}(1^\lambda)$ and $\mathbf{Hyb}^{\mathcal{A}}_{2,j+1}(1^\lambda)$ are computationally indistinguishable. □

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

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

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

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

# C Proofs of Theorem 5.4

In this section, we provide full proof of Theorem 5.4.

**$\mathbf{Hyb}^{\mathcal{A}}_{0}(1^\lambda)$.** 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 $\mathsf{id} \in [n]$, $(\mathsf{mpk}^{(i,j)}_{\mathsf{id}}, \mathsf{msk}^{(i,j)}_{\mathsf{id}}) \leftarrow \mathsf{MAFE.ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}})$ for $i$ such that $\mathsf{id} \notin \mathcal{K}_i$. Set $\mathsf{CRS} := (\mathsf{crs}^{(i,j)})_{i,j}$, $\mathsf{MPK}_{\mathsf{id}} := (\mathsf{mpk}^{(i,j)}_{\mathsf{id}})_{i,j}$, and $\mathsf{MSK}_{\mathsf{id}} := (\mathsf{msk}^{(i,j)}_{\mathsf{id}})_{i,j}$ 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 $\mathsf{sk}^{(i,j)}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{MAFE.KGen}(\mathsf{id}, \mathsf{msk}^{(i,j)}_{\mathsf{id}}, \{\mathsf{mpk}^{(i,j)}_{\mathsf{idx}}\}_{\mathsf{idx}}, \mathsf{GID}, x)$. Set $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} = (\mathsf{sk}^{(i,j)}_{\mathsf{id},\mathsf{GID},x})_{i\in\mathbb{I}'_{\mathsf{id}},j}$ and send it to $\mathcal{A}$.

**Corruption Queries.** $\mathcal{A}$ sends $(\mathsf{Corr}, \mathsf{id})$ to Chal. Send $(\mathsf{msk}^{(i,j)}_{\mathsf{id}})_{i,j}$ to $\mathcal{A}$.

**Challenge Query.** $\mathcal{A}$ sends circuit $C$ to Chal. Chal samples $\{\widetilde{C}_i\}_{i\in\mathbb{I}'} \leftarrow \mathsf{FSS.Share}(1^\lambda, 1^{|\mathbb{I}'|}, C)$. For each $i \in \mathbb{I}', j \in \mathbb{J}$, sample $\mathsf{ct}^{(i,j)} \leftarrow \mathsf{MAFE.Enc}(\{\mathsf{mpk}^{(i,j)}_{\mathsf{id}}\}_{\mathsf{id}\notin\mathcal{K}_i}, F_{ij})$. 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 $\mathsf{sk}^{(i,j)}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{MAFE.KGen}(\mathsf{id}, \mathsf{msk}^{(i,j)}_{\mathsf{id}}, \{\mathsf{mpk}^{(i,j)}_{\mathsf{idx}}\}_{\mathsf{idx}}, \mathsf{GID}, x)$. 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 $(\mathsf{Corr}, \mathsf{id})$ to Chal. Send $(\mathsf{msk}^{(i,j)})_{i,j}$ to $\mathcal{A}$.

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

**$\mathbf{Hyb}^{\mathcal{A}}_{1,\kappa}(1^\lambda)$.** 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 $\mathsf{id} \in [n], \mathsf{id} \notin \mathcal{K}_i$, $\mathsf{mpk}^{(i,j)}_{\mathsf{id}} \leftarrow \mathsf{MAFE.ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}})$.

- 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 $\mathsf{id}$ 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 $\{\widetilde{C}_i\}_{i\in\mathbb{I}'} \leftarrow \mathsf{FSS.Share}(1^\lambda, 1^{|\mathbb{I}'|}, C)$. For each $i \in \mathbb{I}', j \in \mathbb{J}$, sample

- If $i \neq i^*$ or $i = i^*$ and $j \geq \kappa$, $\mathsf{ct}^{(i,j)} \leftarrow \mathsf{MAFE.Enc}(\{\mathsf{mpk}_{\mathsf{id}}^{(i,j)}\}_{\mathsf{id}\notin\mathcal{K}_i}, F_{ij})$.
- 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 4.1 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 $\mathsf{id}$ 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 $\mathsf{V}$ is as defined in Definition 4.1 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.

**$\mathbf{Hyb}_2^{\mathcal{A}}(1^\lambda)$.** In this hybrid, we simulate $\mathsf{FSS}$. 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 $\mathsf{id}$ 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 $\{\widetilde{C}_i\}_{i\neq i^*} \leftarrow \mathsf{FSS.Sim}(1^\lambda, 1^{|\mathbb{I}'|}, 1^{|C|}, i^*)$. For each $i \in \mathbb{I}', j \in \mathbb{J}$, sample

- If $i \neq i^*$, $\mathsf{ct}^{(i,j)} \leftarrow \mathsf{MAFE.Enc}(\{\mathsf{mpk}_{\mathsf{id}}^{(i,j)}\}_{\mathsf{id}\notin\mathcal{K}_i}, F_{ij})$.
- 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 4.1 for the circuit $F_{ij}$. However, we will set the output as $\mathsf{FSS.Sim}(X, C(X))$ 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 $\mathsf{id}$ 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 $\mathsf{V}$ is as defined in Definition 4.1 for the circuit $F_{ij}$. However, we will set the output as $\mathsf{FSS.Sim}(X, C(X))$ 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 C.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 $\mathsf{MAFE}$ instantiations in $\mathbf{Hyb}_{1,1}^{\mathcal{A}}(1^\lambda)$, we are only sampling $i^*$. So, these hybrids are identically distributed. □

**Lemma C.2.** Assuming the security of $\mathsf{MAFE}$, $\mathbf{Hyb}_{1,\kappa}^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_{1,\kappa+1}^{\mathcal{A}}(1^\lambda)$ for $\kappa \in [|\mathbb{J}|]$ are computationally indistinguishable.

*Proof.* Note that the probability that $\mathbf{Hyb}^{\mathcal{A}}_{1,\kappa}(1^\lambda)$ 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 $\mathbf{Hyb}^{\mathcal{A}}_{1,\kappa}(1^\lambda)$ and $\mathbf{Hyb}^{\mathcal{A}}_{1,\kappa+1}(1^\lambda)$ with non-negligible probability $\epsilon(\lambda)$, i.e,

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

We will construct a reduction adversary $\mathcal{B}$ that can break the security of $\mathsf{fcMAFE}$ 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 $\mathsf{id} \in [n], \mathsf{id} \notin \mathcal{K}_i$, $\mathsf{mpk}^{(i,j)}_{\mathsf{id}} \leftarrow \mathsf{MAFE.ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}})$. Otherwise, if $i = i^*, j = \kappa$, $\mathsf{crs}^{(i,j)}, \{\mathsf{mpk}^{(i,j)}_{\mathsf{id}}\}_{\mathsf{id}} \leftarrow \mathcal{O}(Q, n-k, s, \ell_1, \ldots, \ell_n)$.
- Otherwise, $\mathsf{crs}^{(i,j)} \leftarrow \mathsf{MAFE.Sim}(1^\lambda, 1^Q, 1^{n-k}, 1^s)$ and for $\mathsf{id} \notin \mathcal{K}_i, (\mathsf{mpk}^{(i,j)}_{\mathsf{id}}, \mathsf{msk}^{(i,j)}_{\mathsf{id}}) \leftarrow \mathsf{MAFE.Sim}(\mathsf{id}, 1^{\ell_{\mathsf{id}}})$.

Set $\mathsf{CRS} := (\mathsf{crs}^{(i,j)})_{i,j}$, $\mathsf{MPK}_{\mathsf{id}} := (\mathsf{mpk}^{(i,j)}_{\mathsf{id}})_{i,j}$, and $\mathsf{MSK}_{\mathsf{id}} := ((\mathsf{msk}^{(i,j)}_{\mathsf{id}})_{i\neq i^*,j}, (\mathsf{msk}^{(i,j)}_{\mathsf{id}})_{i=i^*,j\geq\kappa})$ 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 $\mathsf{id}$ is not corrupted. For each $i \in \mathbb{I}'_{\mathsf{id}}, j \in \mathbb{J}$,

- If $i \neq i^*$ or $i = i^*$ and $j > \kappa$, $\mathsf{sk}^{(i,j)}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{MAFE.KGen}(\mathsf{id}, \mathsf{msk}^{(i,j)}_{\mathsf{id}}, \{\mathsf{mpk}^{(i,j)}_{\mathsf{idx}}\}_{\mathsf{idx}}, \mathsf{GID}, x)$. Otherwise, if $i = i^*, j = \kappa$, $\mathsf{sk}^{(i,j)}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathcal{O}(\mathsf{id}, \mathsf{GID}, x)$.
- 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 $\{\widetilde{C}_i\}_{i\in\mathbb{I}'} \leftarrow \mathsf{FSS.Share}(1^\lambda, 1^{|\mathbb{I}'|}, C)$. For each $i \in \mathbb{I}', j \in \mathbb{J}$, sample

- If $i \neq i^*$ or $i = i^*$ and $j > \kappa$, $\mathsf{ct}^{(i,j)} \leftarrow \mathsf{MAFE.Enc}(\{\mathsf{mpk}^{(i,j)}_{\mathsf{id}}\}_{\mathsf{id}\notin\mathcal{K}_i}, F_{ij})$. 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 4.1 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 $\mathsf{id}$ is not corrupted. For each $i \in \mathbb{I}'_{\mathsf{id}}, j \in \mathbb{J}$,

- If $i \neq i^*$ or $i = i^*$ and $j > \kappa$, $\mathsf{sk}^{(i,j)}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{MAFE.KGen}(\mathsf{id}, \mathsf{msk}^{(i,j)}_{\mathsf{id}}, \{\mathsf{mpk}^{(i,j)}_{\mathsf{idx}}\}_{\mathsf{idx}}, \mathsf{GID}, x)$. Otherwise, if $i = i^*, j = \kappa$, $\mathsf{sk}^{(i,j)}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathcal{O}(\mathsf{id}, \mathsf{GID}, x)$.

- Otherwise, sample $\mathsf{sk}^{(i,j)}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{MAFE.Sim}(\mathsf{id}, \mathsf{GID}, x, \mathsf{V})$ where $\mathsf{V}$ is as defined in Definition 4.1 for the circuit $F_{ij}$.

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, running time of $\mathcal{B}$ is polynomial in the running time of $\mathcal{A}$ and $\lambda$. 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 $\mathsf{MAFE}$ with non-negligible probability. □

**Lemma C.3.** Assuming the security of $\mathsf{FSS}$, $\mathbf{Hyb}^{\mathcal{A}}_{1,|\mathbb{J}|+1}(1^\lambda)$ and $\mathbf{Hyb}^{\mathcal{A}}_{2}(1^\lambda)$ are computationally indistinguishable.

*Proof.* Assume that there exists a $\mathsf{PPT}$ 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 $\mathsf{FSS}$ 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 $\mathsf{id}$ 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 $i \neq i^*$, $\mathsf{ct}^{(i,j)} \leftarrow \mathsf{MAFE.Enc}(\{\mathsf{mpk}^{(i,j)}_{\mathsf{id}}\}_{\mathsf{id} \notin \mathcal{K}_i}, F_{ij})$.
- 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 4.1 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 $\mathsf{id}$ 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 $\mathsf{V}$ is as defined in Definition 4.1 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 $\mathcal{A}$, $\lambda$. If $\mathcal{O}$ is a honest challenger for $\mathsf{FSS}$, $\mathcal{B}$ behaves like $\mathbf{Hyb}^{\mathcal{A}}_{1,|\mathbb{J}|+1}(1^\lambda)$ and if $\mathcal{O}$ is a simulator for $\mathsf{FSS}$, $\mathcal{B}$ behaves like $\mathbf{Hyb}^{\mathcal{A}}_{2}(1^\lambda)$. Hence, $\mathcal{B}$ is a valid adversary that break the security of $\mathsf{FSS}$ 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.

□

# D Generic Compilers for Authority Corruptions

In this section, we show relations between various notions of authority corruptions in MAFE as defined in Definition 5.1. 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 D.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 5.1 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 D.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 5.1 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 4. The idea is that instead of an ssMAFE scheme, we use a sel-cMAFE scheme to achieve non-adaptive corruptions[15]. 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 (na-cMAFE) using a selectively corruptible MAFE scheme (sel-cMAFE).

**Construction D.3** (na-cMAFE)**.** We provide the construction of na-cMAFE for P/Poly circuits and $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda\in\mathbb{N}}$ using a sel-cMAFE scheme for P/Poly circuits and $\mathcal{GID}$ (Definition D.1) and a BFE scheme for P/Poly circuits (Definition 3.7) as follows.

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

$\mathsf{ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}})$**.** 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 $\kappa_{\mathsf{id}} \le \mathsf{poly}(\lambda, Q, |\mathsf{GID}|, \ell_{\mathsf{id}})$. Output $\mathsf{MPK} := (\ell_{\mathsf{id}}, \mathsf{bfe.mpk})$ and $\mathsf{MSK} := (\ell_{\mathsf{id}}, \mathsf{bfe.msk})$.

$\mathsf{KGen}(\mathsf{id}, \mathsf{MSK}_{\mathsf{id}}, \{\mathsf{MPK}_{\mathsf{idx}}\}_{\mathsf{idx}\in[n]}, \mathsf{GID}, x)$**.** 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 $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} := \mathsf{bfe.sk}_{\mathsf{id},\mathsf{GID},x}$.

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

$\mathsf{Dec}(\{\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}\}_{\mathsf{id}\in[n]}, \mathsf{CT})$**.** 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 $y := \mathsf{sel\text{-}cMAFE.Dec}(\{\mathsf{mafe.sk}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}, \mathsf{mafe.ct})$.

[15] Note that even in Construction 4.9, we could rely on selectively secure ssMAFE to achieve the same result.

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

**Theorem D.4.** If sel-cMAFE is a sel-cMAFE scheme (Definition D.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 3.7), then Construction D.3 is an na-cMAFE scheme for P/Poly circuits and $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda\in\mathbb{N}}$ (Definition D.2).

*Proof Sketch.* The proof of this theorem is similar to the proof of Theorem 4.10. We provide a sketch of the hybrid argument used to prove security of Construction D.3 as follows.

**Hybrid 0.** This is the real experiment from Definition D.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 D.2.

We omit full proof for brevity. □

**Remark D.5** (Adaptive Corruptions)**.** Assuming the quasi-polynomial security of sel-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 sel-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 $2^{-O(\log^2\lambda)}$.

# E Proofs from Section 6.2

In this section, we provide full proof of Lemma 6.5.

**$\mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)$.** This is the real experiment with $\mathsf{Chal}$ with $b = 0/1$.

**Setup.** $\mathcal{A}$ sends $(n, s, \ell_1, \ldots, \ell_n, C_0, C_1)$ to $\mathsf{Chal}$. $\mathsf{Chal}$ sets $\mathsf{CRS} := (\lambda, n, s)$. Then, $\mathsf{Chal}$ samples $(\mathsf{fe.mpk}_{\mathsf{id}}, \mathsf{fe.msk}_{\mathsf{id}}) \leftarrow \mathsf{FE.Setup}(1^\lambda, 1^{s'_{\mathsf{id}}})$ for each $\mathsf{id} \in [n]$. $\mathsf{Chal}$ sets $\mathsf{MPK}_{\mathsf{id}} := (\mathsf{id}, \ell_{\mathsf{id}}, \mathsf{fe.mpk}_{\mathsf{id}})$ and $\mathsf{MSK}_{\mathsf{id}} := (\mathsf{id}, \mathsf{fe.msk}_{\mathsf{id}})$.

Now, $\mathsf{Chal}$ computes $\mathbb{I}_1, \ldots, \mathbb{I}_n$, $\alpha_1, \ldots, \alpha_n$, samples $K \xleftarrow{\$} \{0,1\}^\lambda$ and $\mathsf{fe.ct}_{\mathsf{id}} \leftarrow \mathsf{FE.Enc}(\mathsf{fe.mpk}_{\mathsf{id}}, (C_b, K, \alpha_{\mathsf{id}}, \mathbb{I}_{\mathsf{id}}, 0, \bot))$, and sets $\mathsf{CT} := (\mathsf{fe.ct}_{\mathsf{id}})_{\mathsf{id}\in[n]}$.

$\mathsf{Chal}$ sends $(\mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}, \mathsf{CT})$ to $\mathcal{A}$.

**Secret Key Queries.** $\mathcal{A}$ now sends $Q = \mathsf{poly}(\lambda)$ many secret key queries of the form $(\mathsf{GID}, x_1, \ldots, x_n)$.

$\mathsf{Chal}$ now samples $\mathsf{ske.ct}^*_{\mathsf{id}} \xleftarrow{\$} \{0,1\}^{\ell^*_{\mathsf{id}}}$ for each $\mathsf{id} \in [n]$ and creates circuits $F_1, \ldots, F_n$ as described in Figure 3. Then sample $\mathsf{fe.sk}_{F_{\mathsf{id}}} \leftarrow \mathsf{FE.KGen}(\mathsf{fe.msk}_{\mathsf{id}}, F_{\mathsf{id}})$. Set $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}} := \mathsf{fe.sk}_{F_{\mathsf{id}}}$ and send $\{\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}\}_{\mathsf{id}\in[n]}$ to $\mathcal{A}$.

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

**$\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$.** In this hybrid, we set $\mathsf{ske.ct}^*_{\mathsf{id}}$ to be an $\mathsf{SKE}$ ciphertext of $F_{\mathsf{id}}(C_b, K, \alpha_{\mathsf{id}}, \mathbb{I}_{\mathsf{id}}, 0, \bot)$. The changes are highlighted in red.

**Setup.** $\mathcal{A}$ sends $(n, s, \ell_1, \ldots, \ell_n, C_0, C_1)$. Set $\mathsf{CRS} := (\lambda, n, s)$. Then, sample $(\mathsf{fe.mpk}_{\mathsf{id}}, \mathsf{fe.msk}_{\mathsf{id}}) \leftarrow \mathsf{FE.Setup}(1^\lambda, 1^{s'_{\mathsf{id}}})$ for each $\mathsf{id} \in [n]$. Set $\mathsf{MPK}_{\mathsf{id}} := (\mathsf{id}, \ell_{\mathsf{id}}, \mathsf{fe.mpk}_{\mathsf{id}})$ and $\mathsf{MSK}_{\mathsf{id}} := (\mathsf{id}, \mathsf{fe.msk}_{\mathsf{id}})$. In addition, sample $\mathsf{ske.sk} \leftarrow \mathsf{SKE.Setup}(1^\lambda)$.

Now, compute $\mathbb{I}_1, \ldots, \mathbb{I}_n$, $\alpha_1, \ldots, \alpha_n$, samples $K \xleftarrow{\$} \{0,1\}^\lambda$ and $\mathsf{fe.ct}_{\mathsf{id}} \leftarrow \mathsf{FE.Enc}(\mathsf{fe.mpk}_{\mathsf{id}}, (C_b, K, \alpha_{\mathsf{id}}, \mathbb{I}_{\mathsf{id}}, 0, \bot))$, and set $\mathsf{CT} := (\mathsf{fe.ct}_{\mathsf{id}})_{\mathsf{id}\in[n]}$.

Send $(\mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}, \mathsf{CT})$ to $\mathcal{A}$.

**Secret Key Queries.** $\mathcal{A}$ now sends $Q = \mathsf{poly}(\lambda)$ many secret key queries of the form $(\mathsf{GID}, x_1, \ldots, x_n)$.

Compute $\widetilde{C}, \{w_{i,b}\}_{i\in[\ell_1+\ldots+\ell_n], b\in\{0,1\}} \leftarrow \mathsf{Garb}(1^\lambda, C_b; \mathsf{PRF}(K, \mathsf{GID}))$. Sample $\mathsf{ske.ct}^*_{\mathsf{id}} \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}, (\widetilde{C}, \{w_{i,x_{\mathsf{id}}[i-\alpha_{\mathsf{id}}+1]}\}_{i\in\mathbb{I}_{\mathsf{id}}}))$ for each $\mathsf{id} \in [n]$.

Create circuits $F_1, \ldots, F_n$ as described in Figure 3. Then sample $\mathsf{fe.sk}_{F_{\mathsf{id}}} \leftarrow \mathsf{FE.KGen}(\mathsf{fe.msk}_{\mathsf{id}}, F_{\mathsf{id}})$. Set $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}} := \mathsf{fe.sk}_{F_{\mathsf{id}}}$ and send $\{\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}\}_{\mathsf{id}\in[n]}$ to $\mathcal{A}$.

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

**$\mathbf{Hyb}_{2,j}^{\mathcal{A}}(1^\lambda)$ for $j \in [n+1]$.** In this hybrid, we set input to the first $j-1$ instantiations of $\mathsf{FE}$ to be $(\overline{0}, 1, \mathsf{ske.sk})$. The changes are highlighted in red.

**Setup.** $\mathcal{A}$ sends $(n, s, \ell_1, \ldots, \ell_n, C_0, C_1)$. Set $\mathsf{CRS} := (\lambda, n, s)$. Then, sample $(\mathsf{fe.mpk}_{\mathsf{id}}, \mathsf{fe.msk}_{\mathsf{id}}) \leftarrow \mathsf{FE.Setup}(1^\lambda, 1^{s'_{\mathsf{id}}})$ for each $\mathsf{id} \in [n]$. Set $\mathsf{MPK}_{\mathsf{id}} := (\mathsf{id}, \ell_{\mathsf{id}}, \mathsf{fe.mpk}_{\mathsf{id}})$ and $\mathsf{MSK}_{\mathsf{id}} := (\mathsf{id}, \mathsf{fe.msk}_{\mathsf{id}})$. Sample $\mathsf{ske.sk} \leftarrow \mathsf{SKE.Setup}(1^\lambda)$.

Now, compute $\mathbb{I}_1, \ldots, \mathbb{I}_n$, $\alpha_1, \ldots, \alpha_n$, sample $K \xleftarrow{\$} \{0,1\}^\lambda$. If $\mathsf{id} < j$, $\mathsf{fe.ct}_{\mathsf{id}} \leftarrow \mathsf{FE.Enc}(\mathsf{fe.mpk}_{\mathsf{id}}, (\overline{0}, 1, \mathsf{ske.sk}))$. Otherwise, $\mathsf{fe.ct}_{\mathsf{id}} \leftarrow \mathsf{FE.Enc}(\mathsf{fe.mpk}_{\mathsf{id}}, (C_b, K, \alpha_{\mathsf{id}}, \mathbb{I}_{\mathsf{id}}, 0, \bot))$. Set $\mathsf{CT} := (\mathsf{fe.ct}_{\mathsf{id}})_{\mathsf{id}\in[n]}$.

Sends $(\mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}, \mathsf{CT})$ to $\mathcal{A}$.

**Secret Key Queries.** $\mathcal{A}$ now sends $Q = \mathsf{poly}(\lambda)$ many secret key queries of the form $(\mathsf{GID}, x_1, \ldots, x_n)$.

Compute $\widetilde{C}, \{w_{i,b}\}_{i\in[\ell_1+\ldots+\ell_n],b\in\{0,1\}} \leftarrow \mathsf{Garb}(1^\lambda, C_b; \mathsf{PRF}(K, \mathsf{GID}))$. Sample $\mathsf{ske.ct}^*_{\mathsf{id}} \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}, (\widetilde{C}, \{w_{i,x_{\mathsf{id}}[i-\alpha_{\mathsf{id}}+1]}\}_{i\in\mathbb{I}_{\mathsf{id}}}))$ for each $\mathsf{id} \in [n]$.

Create circuits $F_1, \ldots, F_n$ as described in Figure 3. Then sample $\mathsf{fe.sk}_{F_{\mathsf{id}}} \leftarrow \mathsf{FE.KGen}(\mathsf{fe.msk}_{\mathsf{id}}, F_{\mathsf{id}})$. Set $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}} := \mathsf{fe.sk}_{F_{\mathsf{id}}}$ and send $\{\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}\}_{\mathsf{id}\in[n]}$ to $\mathcal{A}$.

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

**$\mathbf{Hyb}_3^{\mathcal{A}}(1^\lambda)$.** In this hybrid, we sample uniformly random strings for garbling randomness instead of using $\mathsf{PRF}(K, \mathsf{GID})$. The changes are highlighted in red.

**Setup.** $\mathcal{A}$ sends $(n, s, \ell_1, \ldots, \ell_n, C_0, C_1)$. Set $\mathsf{CRS} := (\lambda, n, s)$. Then, sample $(\mathsf{fe.mpk}_{\mathsf{id}}, \mathsf{fe.msk}_{\mathsf{id}}) \leftarrow \mathsf{FE.Setup}(1^\lambda, 1^{s'_{\mathsf{id}}})$ for each $\mathsf{id} \in [n]$. Set $\mathsf{MPK}_{\mathsf{id}} := (\mathsf{id}, \ell_{\mathsf{id}}, \mathsf{fe.mpk}_{\mathsf{id}})$ and $\mathsf{MSK}_{\mathsf{id}} := (\mathsf{id}, \mathsf{fe.msk}_{\mathsf{id}})$. Sample $\mathsf{ske.sk} \leftarrow \mathsf{SKE.Setup}(1^\lambda)$.

Now, compute $\mathbb{I}_1, \ldots, \mathbb{I}_n$, $\alpha_1, \ldots, \alpha_n$. Sample $\mathsf{fe.ct}_{\mathsf{id}} \leftarrow \mathsf{FE.Enc}(\mathsf{fe.mpk}_{\mathsf{id}}, (\overline{0}, 1, \mathsf{ske.sk}))$. Set $\mathsf{CT} := (\mathsf{fe.ct}_{\mathsf{id}})_{\mathsf{id}\in[n]}$.

Sends $(\mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}, \mathsf{CT})$ to $\mathcal{A}$.

**Secret Key Queries.** $\mathcal{A}$ now sends $Q = \mathsf{poly}(\lambda)$ many secret key queries of the form $(\mathsf{GID}, x_1, \ldots, x_n)$.

Compute $\widetilde{C}, \{w_{i,b}\}_{i\in[\ell_1+\ldots+\ell_n],b\in\{0,1\}} \leftarrow \mathsf{Garb}(1^\lambda, C_b; R)$ where $R \xleftarrow{\$} \{0,1\}^\lambda$. Sample $\mathsf{ske.ct}^*_{\mathsf{id}} \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}, (\widetilde{C}, \{w_{i,x_{\mathsf{id}}[i-\alpha_{\mathsf{id}}+1]}\}_{i\in\mathbb{I}_{\mathsf{id}}}))$ for each $\mathsf{id} \in [n]$.

Create circuits $F_1, \ldots, F_n$ as described in Figure 3. Then sample $\mathsf{fe.sk}_{F_{\mathsf{id}}} \leftarrow \mathsf{FE.KGen}(\mathsf{fe.msk}_{\mathsf{id}}, F_{\mathsf{id}})$. Set $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}} := \mathsf{fe.sk}_{F_{\mathsf{id}}}$ and send $\{\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}\}_{\mathsf{id}\in[n]}$ to $\mathcal{A}$.

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

**$\mathbf{Hyb}_{4,j}^{\mathcal{A}}(1^\lambda)$ for $j \in [Q+1]$.** In this hybrid, we simulate the first $j-1$ garbling instantiations using $\mathsf{Garb}(1^\lambda, C_b)$. The changes are highlighted in red.

**Setup.** $\mathcal{A}$ sends $(n, s, \ell_1, \ldots, \ell_n, C_0, C_1)$. Set $\mathsf{CRS} := (\lambda, n, s)$. Then, sample $(\mathsf{fe.mpk}_{\mathsf{id}}, \mathsf{fe.msk}_{\mathsf{id}}) \leftarrow \mathsf{FE.Setup}(1^\lambda, 1^{s'_{\mathsf{id}}})$ for each $\mathsf{id} \in [n]$. Set $\mathsf{MPK}_{\mathsf{id}} := (\mathsf{id}, \ell_{\mathsf{id}}, \mathsf{fe.mpk}_{\mathsf{id}})$ and $\mathsf{MSK}_{\mathsf{id}} := (\mathsf{id}, \mathsf{fe.msk}_{\mathsf{id}})$. Sample $\mathsf{ske.sk} \leftarrow \mathsf{SKE.Setup}(1^\lambda)$.

Now, compute $\mathbb{I}_1, \ldots, \mathbb{I}_n$, $\alpha_1, \ldots, \alpha_n$. Sample $\mathsf{fe.ct}_{\mathsf{id}} \leftarrow \mathsf{FE.Enc}(\mathsf{fe.mpk}_{\mathsf{id}}, (\overline{0}, 1, \mathsf{ske.sk}))$. Set $\mathsf{CT} := (\mathsf{fe.ct}_{\mathsf{id}})_{\mathsf{id}\in[n]}$.

Sends $(\mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}, \mathsf{CT})$ to $\mathcal{A}$.

**Secret Key Queries.** $\mathcal{A}$ now sends $Q = \mathsf{poly}(\lambda)$ many secret key queries of the form $(\mathsf{GID}, x_1, \ldots, x_n)$.

Let $X = (x_1, \ldots, x_n) \in \{0,1\}^{\ell_1+\ldots+\ell_n}$ and $y = C_b(X)$.

If this is the $q$-th query where $q < j$, compute $\widetilde{C}, \{w_{i,X[i]}\}_{i\in[\ell_1+\ldots+\ell_n]} \leftarrow \mathsf{Garb.Sim}(1^\lambda, 1^{|C_b|}, y)$. Otherwise, compute $\widetilde{C}, \{w_{i,X[i]}\}_{i\in[\ell_1+\ldots+\ell_n]} \leftarrow \mathsf{Garb}(1^\lambda, C_b)$. Sample $\mathsf{ske.ct}^*_{\mathsf{id}} \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}, (\widetilde{C}, \{w_{i,X[i]}\}_{i\in\mathbb{I}_{\mathsf{id}}}))$ for each $\mathsf{id} \in [n]$.

Create circuits $F_1, \ldots, F_n$ as described in Figure 3. Then sample $\mathsf{fe.sk}_{F_{\mathsf{id}}} \leftarrow \mathsf{FE.KGen}(\mathsf{fe.msk}_{\mathsf{id}}, F_{\mathsf{id}})$. Set $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}} := \mathsf{fe.sk}_{F_{\mathsf{id}}}$ and send $\{\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}\}_{\mathsf{id}\in[n]}$ to $\mathcal{A}$.

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

**Claim E.1.** Assuming the pseudorandom ciphertext security of $\mathsf{SKE}$, $\mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$ are computationally indistinguishable.

*Proof.* Assume that there exists an adversary $\mathcal{A}$ that can distinguish between $\mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$ with non-negligible probability $\epsilon(\lambda)$, 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 $\mathsf{SKE}$ with non-negligible probability. The description of $\mathcal{B}^{\mathcal{O}}$ is as follows.

**Setup.** $\mathcal{A}$ sends $(n, s, \ell_1, \ldots, \ell_n, C_0, C_1)$. Set $\mathsf{CRS} := (\lambda, n, s)$. Then, sample $(\mathsf{fe.mpk}_{\mathsf{id}}, \mathsf{fe.msk}_{\mathsf{id}}) \leftarrow \mathsf{FE.Setup}(1^\lambda, 1^{s'_{\mathsf{id}}})$ for each $\mathsf{id} \in [n]$. Set $\mathsf{MPK}_{\mathsf{id}} := (\mathsf{id}, \ell_{\mathsf{id}}, \mathsf{fe.mpk}_{\mathsf{id}})$ and $\mathsf{MSK}_{\mathsf{id}} := (\mathsf{id}, \mathsf{fe.msk}_{\mathsf{id}})$.

Now, compute $\mathbb{I}_1, \ldots, \mathbb{I}_n$, $\alpha_1, \ldots, \alpha_n$, samples $K \xleftarrow{\$} \{0,1\}^\lambda$ and $\mathsf{fe.ct}_{\mathsf{id}} \leftarrow \mathsf{FE.Enc}(\mathsf{fe.mpk}_{\mathsf{id}}, (C_b, K, \alpha_{\mathsf{id}}, \mathbb{I}_{\mathsf{id}}, 0, \bot))$, and set $\mathsf{CT} := (\mathsf{fe.ct}_{\mathsf{id}})_{\mathsf{id}\in[n]}$.

Send $(\mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}, \mathsf{CT})$ to $\mathcal{A}$.

**Secret Key Queries.** $\mathcal{A}$ now sends $Q = \mathsf{poly}(\lambda)$ many secret key queries of the form $(\mathsf{GID}, x_1, \ldots, x_n)$.

Compute $\widetilde{C}, \{w_{i,b}\}_{i\in[\ell_1+\ldots+\ell_n], b\in\{0,1\}} \leftarrow \mathsf{Garb}(1^\lambda, C_b; \mathsf{PRF}(K, \mathsf{GID}))$. Query $\mathcal{O}$ to get $\mathsf{ske.ct}^*_{\mathsf{id}} \leftarrow \mathcal{O}((\widetilde{C}, \{w_{i,x_{\mathsf{id}}[i-\alpha_{\mathsf{id}}+1]}\}_{i\in\mathbb{I}_{\mathsf{id}}}))$ for each $\mathsf{id} \in [n]$.

Create circuits $F_1, \ldots, F_n$ as described in Figure 3. Then sample $\mathsf{fe.sk}_{F_{\mathsf{id}}} \leftarrow \mathsf{FE.KGen}(\mathsf{fe.msk}_{\mathsf{id}}, F_{\mathsf{id}})$. Set $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}} := \mathsf{fe.sk}_{F_{\mathsf{id}}}$ and send $\{\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}\}_{\mathsf{id}\in[n]}$ to $\mathcal{A}$.

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

As we can see, the running time of $\mathcal{B}$ is polynomial in the running time of $\mathcal{A}$, $\lambda$. If $\mathcal{O}$ generates uniformly random strings, $\mathcal{B}$ behaves like $\mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)$ and if $\mathcal{O}$ generates encryptions of message, $\mathcal{B}$ behaves like $\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$. Hence, $\mathcal{B}$ is a valid adversary that break the security of $\mathsf{SKE}$ with non-negligible probability. Thus, $\mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$ are computationally indistinguishable. □

**Claim E.2.** $\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_{2,1}^{\mathcal{A}}(1^\lambda)$ are identically distributed.

*Proof.* As we are not switching inputs of any $\mathsf{FE}$ instantiations in $\mathbf{Hyb}_{2,1}^{\mathcal{A}}(1^\lambda)$, these hybrids are identically distributed. □

**Claim E.3.** Assuming the IND security of $\mathsf{FE}$, $\mathbf{Hyb}_{2,j}^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_{2,j+1}^{\mathcal{A}}(1^\lambda)$ for any $j \in [n]$ are computationally indistinguishable.

*Proof.* Assume that there exists an adversary $\mathcal{A}$ that can distinguish between $\mathbf{Hyb}_{2,j}^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_{2,j+1}^{\mathcal{A}}(1^\lambda)$ for some $j \in [n]$ with non-negligible probability $\epsilon(\lambda)$, 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 IND security of $\mathsf{FE}$ with non-negligible probability. The description of $\mathcal{B}^{\mathcal{O}}$ is as follows.

**Setup.** Receive $\mathsf{mpk}$ from $\mathcal{O}$ and set $\mathsf{fe.mpk}_j := \mathsf{mpk}$. $\mathcal{A}$ sends $(n, s, \ell_1, \ldots, \ell_n, C_0, C_1)$. Set $\mathsf{CRS} := (\lambda, n, s)$. Then, sample $(\mathsf{fe.mpk}_{\mathsf{id}}, \mathsf{fe.msk}_{\mathsf{id}}) \leftarrow \mathsf{FE.Setup}(1^\lambda, 1^{s'_{\mathsf{id}}})$ for each $\mathsf{id} \in [n] \setminus \{j\}$. Set $\mathsf{MPK}_{\mathsf{id}} := (\mathsf{id}, \ell_{\mathsf{id}}, \mathsf{fe.mpk}_{\mathsf{id}})$ and $\mathsf{MSK}_{\mathsf{idx}} := (\mathsf{idx}, \mathsf{fe.msk}_{\mathsf{idx}})$ for $\mathsf{idx} \in [n] \setminus \{j\}$. Sample $\mathsf{ske.sk} \leftarrow \mathsf{SKE.Setup}(1^\lambda)$.

Now, compute $\mathbb{I}_1, \ldots, \mathbb{I}_n, \alpha_1, \ldots, \alpha_n$, sample $K \xleftarrow{\$} \{0,1\}^\lambda$. If $\mathsf{id} < j$, $\mathsf{fe.ct}_{\mathsf{id}} \leftarrow \mathsf{FE.Enc}(\mathsf{fe.mpk}_{\mathsf{id}}, (\overline{0}, 1, \mathsf{ske.sk}))$. Otherwise, if $\mathsf{id} = j$, query $\mathcal{O}$ to receive $\mathsf{fe.ct}_{\mathsf{id}} \leftarrow \mathcal{O}(m_0 := (C_b, K, \alpha_{\mathsf{id}}, \mathbb{I}_{\mathsf{id}}, 0, \bot), m_1 := (\overline{0}, 1, \mathsf{ske.sk}))$. Otherwise, $\mathsf{fe.ct}_{\mathsf{id}} \leftarrow \mathsf{FE.Enc}(\mathsf{fe.mpk}_{\mathsf{id}}, (C_b, K, \alpha_{\mathsf{id}}, \mathbb{I}_{\mathsf{id}}, 0, \bot))$. Set $\mathsf{CT} := (\mathsf{fe.ct}_{\mathsf{id}})_{\mathsf{id} \in [n]}$.

Sends $(\mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id} \in [n]}, \mathsf{CT})$ to $\mathcal{A}$.

**Secret Key Queries.** $\mathcal{A}$ now sends $Q = \mathsf{poly}(\lambda)$ many secret key queries of the form $(\mathsf{GID}, x_1, \ldots, x_n)$.

Compute $\widetilde{C}, \{w_{i,b}\}_{i \in [\ell_1 + \ldots + \ell_n], b \in \{0,1\}} \leftarrow \mathsf{Garb}(1^\lambda, C_b; \mathsf{PRF}(K, \mathsf{GID}))$. Sample $\mathsf{ske.ct}^*_{\mathsf{id}} \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}, (\widetilde{C}, \{w_{i, x_{\mathsf{id}}[i - \alpha_{\mathsf{id}} + 1]}\}_{i \in \mathbb{I}_{\mathsf{id}}}))$ for each $\mathsf{id} \in [n]$.

Create circuits $F_1, \ldots, F_n$ as described in Figure 3. Then sample $\mathsf{fe.sk}_{F_{\mathsf{id}}} \leftarrow \mathsf{FE.KGen}(\mathsf{fe.msk}_{\mathsf{id}}, F_{\mathsf{id}})$ for $\mathsf{id} \neq j$. Send $F_j$ to $\mathcal{O}$ to receive $\mathsf{fe.sk}_{F_j}$. Set $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}} := \mathsf{fe.sk}_{F_{\mathsf{id}}}$ and send $\{\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}\}_{\mathsf{id} \in [n]}$ to $\mathcal{A}$.

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

As we can see, the running time of $\mathcal{B}$ is polynomial in the running time of $\mathcal{A}$, $\lambda$. If $\mathcal{O}$ uses $m_0$, $\mathcal{B}$ behaves like $\mathbf{Hyb}^{\mathcal{A}}_{2,j}(1^\lambda)$ and if $\mathcal{O}$ generates encryptions of message $m_1$, $\mathcal{B}$ behaves like $\mathbf{Hyb}^{\mathcal{A}}_{2,j+1}(1^\lambda)$. Hence, $\mathcal{B}$ is a valid adversary that break the security of $\mathsf{SKE}$ with non-negligible probability. Thus, $\mathbf{Hyb}^{\mathcal{A}}_{2,j}(1^\lambda)$ and $\mathbf{Hyb}^{\mathcal{A}}_{2,j+1}(1^\lambda)$ are computationally indistinguishable. □

**Claim E.4.** Assuming the pseudorandomness property of $\mathsf{PRF}$, $\mathbf{Hyb}^{\mathcal{A}}_{2,n+1}(1^\lambda)$ and $\mathbf{Hyb}^{\mathcal{A}}_{3}(1^\lambda)$ are computationally indistinguishable.

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

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

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

**Setup.** $\mathcal{A}$ sends $(n, s, \ell_1, \ldots, \ell_n, C_0, C_1)$. Set $\mathsf{CRS} := (\lambda, n, s)$. Then, sample $(\mathsf{fe.mpk}_{\mathsf{id}}, \mathsf{fe.msk}_{\mathsf{id}}) \leftarrow \mathsf{FE.Setup}(1^\lambda, 1^{s'_{\mathsf{id}}})$ for each $\mathsf{id} \in [n]$. Set $\mathsf{MPK}_{\mathsf{id}} := (\mathsf{id}, \ell_{\mathsf{id}}, \mathsf{fe.mpk}_{\mathsf{id}})$ and $\mathsf{MSK}_{\mathsf{id}} := (\mathsf{id}, \mathsf{fe.msk}_{\mathsf{id}})$. Sample $\mathsf{ske.sk} \leftarrow \mathsf{SKE.Setup}(1^\lambda)$.

Now, compute $\mathbb{I}_1, \ldots, \mathbb{I}_n$, $\alpha_1, \ldots, \alpha_n$. Sample $\mathsf{fe.ct}_{\mathsf{id}} \leftarrow \mathsf{FE.Enc}(\mathsf{fe.mpk}_{\mathsf{id}}, (\overline{0}, 1, \mathsf{ske.sk}))$. Set $\mathsf{CT} := (\mathsf{fe.ct}_{\mathsf{id}})_{\mathsf{id} \in [n]}$.

Sends $(\mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id} \in [n]}, \mathsf{CT})$ to $\mathcal{A}$.

**Secret Key Queries.** $\mathcal{A}$ now sends $Q = \mathsf{poly}(\lambda)$ many secret key queries of the form $(\mathsf{GID}, x_1, \ldots, x_n)$.

Compute $\widetilde{C}, \{w_{i,b}\}_{i \in [\ell_1 + \ldots + \ell_n], b \in \{0,1\}} \leftarrow \mathsf{Garb}(1^\lambda, C_b; \mathcal{O}(\mathsf{GID}))$. Sample $\mathsf{ske.ct}^*_{\mathsf{id}} \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}, (\widetilde{C}, \{w_{i, x_{\mathsf{id}}[i - \alpha_{\mathsf{id}} + 1]}\}_{i \in \mathbb{I}_{\mathsf{id}}}))$ for each $\mathsf{id} \in [n]$.

Create circuits $F_1, \ldots, F_n$ as described in Figure 3. Then sample $\mathsf{fe.sk}_{F_{\mathsf{id}}} \leftarrow \mathsf{FE.KGen}(\mathsf{fe.msk}_{\mathsf{id}}, F_{\mathsf{id}})$. Set $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}} := \mathsf{fe.sk}_{F_{\mathsf{id}}}$ and send $\{\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}\}_{\mathsf{id}\in[n]}$ to $\mathcal{A}$.

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

As we can see, the running time of $\mathcal{B}$ is polynomial in the running time of $\mathcal{A}$, $\lambda$. If $\mathcal{O}$ is $\mathsf{PRF}(K,\cdot)$ for some $K \xleftarrow{\$} \{0,1\}^\lambda$, $\mathcal{B}$ behaves like $\mathbf{Hyb}^{\mathcal{A}}_{2,n+1}(1^\lambda)$ and if $\mathcal{O}$ generates uniformly random strings $\mathcal{B}$ behaves like $\mathbf{Hyb}^{\mathcal{A}}_{3}(1^\lambda)$. Hence, $\mathcal{B}$ is a valid adversary that break the security of $\mathsf{PRF}$ with non-negligible probability. Thus, $\mathbf{Hyb}^{\mathcal{A}}_{2,n+1}(1^\lambda)$ and $\mathbf{Hyb}^{\mathcal{A}}_{3}(1^\lambda)$ are computationally indistinguishable. □

**Claim E.5.** $\mathbf{Hyb}^{\mathcal{A}}_{3}(1^\lambda)$ and $\mathbf{Hyb}^{\mathcal{A}}_{4,1}(1^\lambda)$ are identically distributed.

*Proof.* As we are not simulating any garblings of $C_b$ in $\mathbf{Hyb}^{\mathcal{A}}_{4,1}(1^\lambda)$, these hybrids are identically distributed. □

**Claim E.6.** Assuming the security of $(\mathsf{Garb},\mathsf{Eval})$, $\mathbf{Hyb}^{\mathcal{A}}_{4,j}(1^\lambda)$ and $\mathbf{Hyb}^{\mathcal{A}}_{4,j+1}(1^\lambda)$ for any $j \in [Q]$ are computationally indistinguishable.

*Proof.* Assume that there exists an adversary $\mathcal{A}$ that can distinguish between $\mathbf{Hyb}^{\mathcal{A}}_{4,j}(1^\lambda)$ and $\mathbf{Hyb}^{\mathcal{A}}_{4,j+1}(1^\lambda)$ for some $j \in [Q]$ with non-negligible probability $\epsilon(\lambda)$, i.e.,

$$\left|\Pr\left[1 \leftarrow \mathbf{Hyb}^{\mathcal{A}}_{4,j}(1^\lambda)\right] - \Pr\left[1 \leftarrow \mathbf{Hyb}^{\mathcal{A}}_{4,j+1}(1^\lambda)\right]\right| > \epsilon(\lambda)$$

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

**Setup.** $\mathcal{A}$ sends $(n, s, \ell_1, \ldots, \ell_n, C_0, C_1)$. Set $\mathsf{CRS} := (\lambda, n, s)$. Then, sample $(\mathsf{fe.mpk}_{\mathsf{id}}, \mathsf{fe.msk}_{\mathsf{id}}) \leftarrow \mathsf{FE.Setup}(1^\lambda, 1^{s'_{\mathsf{id}}})$ for each $\mathsf{id} \in [n]$. Set $\mathsf{MPK}_{\mathsf{id}} := (\mathsf{id}, \ell_{\mathsf{id}}, \mathsf{fe.mpk}_{\mathsf{id}})$ and $\mathsf{MSK}_{\mathsf{id}} := (\mathsf{id}, \mathsf{fe.msk}_{\mathsf{id}})$. Sample $\mathsf{ske.sk} \leftarrow \mathsf{SKE.Setup}(1^\lambda)$.

Now, compute $\mathbb{I}_1, \ldots, \mathbb{I}_n$, $\alpha_1, \ldots, \alpha_n$. Sample $\mathsf{fe.ct}_{\mathsf{id}} \leftarrow \mathsf{FE.Enc}(\mathsf{fe.mpk}_{\mathsf{id}}, (\overline{0}, 1, \mathsf{ske.sk}))$. Set $\mathsf{CT} := (\mathsf{fe.ct}_{\mathsf{id}})_{\mathsf{id}\in[n]}$.

Sends $(\mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}, \mathsf{CT})$ to $\mathcal{A}$.

**Secret Key Queries.** $\mathcal{A}$ now sends $Q = \mathsf{poly}(\lambda)$ many secret key queries of the form $(\mathsf{GID}, x_1, \ldots, x_n)$. Let $X = (x_1, \ldots, x_n) \in \{0,1\}^{\ell_1+\ldots+\ell_n}$ and $y = C_b(X)$.

If this is the $q$-th query where $q < j$, compute $\widetilde{C}, \{w_{i,X[i]}\}_{i\in[\ell_1+\ldots+\ell_n]} \leftarrow \mathsf{Garb.Sim}(1^\lambda, 1^{|C_b|}, y)$. Otherwise, if this is the $j$-th query, send $(C_b, X)$ to $\mathcal{O}$ to receive $\widetilde{C}, \{w_{i,X[i]}\}_{i\in[\ell_1+\ldots+\ell_n]}$. Otherwise, compute $\widetilde{C}, \{w_{i,X[i]}\}_{i\in[\ell_1+\ldots+\ell_n]} \leftarrow \mathsf{Garb}(1^\lambda, C_b)$. Sample $\mathsf{ske.ct}^*_{\mathsf{id}} \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}, (\widetilde{C}, \{w_{i,X[i]}\}_{i\in\mathbb{I}_{\mathsf{id}}}))$ for each $\mathsf{id} \in [n]$.

Create circuits $F_1, \ldots, F_n$ as described in Figure 3. Then sample $\mathsf{fe.sk}_{F_{\mathsf{id}}} \leftarrow \mathsf{FE.KGen}(\mathsf{fe.msk}_{\mathsf{id}}, F_{\mathsf{id}})$. Set $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}} := \mathsf{fe.sk}_{F_{\mathsf{id}}}$ and send $\{\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}\}_{\mathsf{id}\in[n]}$ to $\mathcal{A}$.

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

As we can see, the running time of $\mathcal{B}$ is polynomial in the running time of $\mathcal{A}$, $\lambda$. If $\mathcal{O}$ honestly garbles $C_b$ $\mathcal{B}$ behaves like $\mathbf{Hyb}^{\mathcal{A}}_{4,j}(1^\lambda)$ and if $\mathcal{O}$ simulates garbling using $C_b(X)$, $\mathcal{B}$ behaves like $\mathbf{Hyb}^{\mathcal{A}}_{4,j+1}(1^\lambda)$. Hence, $\mathcal{B}$ is a valid adversary that break the security of $(\mathsf{Garb}, \mathsf{Eval})$ with non-negligible probability. Thus, $\mathbf{Hyb}^{\mathcal{A}}_{4,j}(1^\lambda)$ and $\mathbf{Hyb}^{\mathcal{A}}_{4,j+1}(1^\lambda)$ are computationally indistinguishable. □

# F One-Ciphertext Private-Key MAFE

In this section, we show how to construct a one-ciphertext private-key MAFE scheme for $\mathsf{P/Poly}$ circuits (Definition 6.6) from public-key encryption. At a first glance, it looks like one could follow a similar blueprint as [ABSV15] where we start from private-key 1MAFE ([GY24]), make it function-hiding similarly to [BS18] and then switch the roles of $\mathsf{KGen}$ and $\mathsf{Enc}$. However, the major issue is that in 1MAFE, the key generation is split rather than encryption and we can only give out one key for each authority to the attacker for security to hold. The same is true even in the indistinguishability domain.

The proposed definition can be realized via the pipeline specified in [ABSV15] if one starts from a *multi-client* functional encryption [GGG+14, CDSG+18] (MCFE) rather than a multi-authority functional encryption. Essentially, in the former, the encryption algorithm is performed with respect to a $\mathsf{GID}$[16] and split across multiple "clients". The resultant ciphertexts can be combined only if $\mathsf{GID}$ is same across all encryptions rather than key generation as in MAFE. We provide construction of IND-secure private-key MCFE for $\mathsf{P/Poly}$ circuits with synchronous key generation assuming the existence of one-way functions. Then, we make it function hiding using [BS18] at which point, we can semantically switch the roles of $\mathsf{Enc}$ and $\mathsf{KGen}$ to realize Definition 6.6. We remark that as we use 1-CT private-key MAFE only for constructing MAFE with synchronous key generation and as such do not require authority corruptions, we will not assume that $\mathsf{EK}_{\mathsf{id}}$ for our MCFE will be corrupted. Our construction uses a private-key 1-key FE scheme which we define below.

**Definition F.1.** A private-key one-key FE scheme for $\mathsf{P/Poly}$ circuits is a BFE scheme for $\mathsf{P/Poly}$ circuits (Definition 3.7) where the collusion bound $Q = 1$. Hence, we ignore $Q$ from the syntax. The security of the scheme can also be strengthened to allow unbounded encryptions for different circuits but only one challenge circuit against which simulation security holds. Specifically, the security definition is defined below.

**Security.** For any $\mathsf{PPT}$ stateful admissible adversary $\mathcal{A}$, there exists a stateful $\mathsf{PPT}$ simulator such

[16] We remark that in the literature for MCFE, 'labels'/'tags' are used as opposed to $\mathsf{GID}$. However, to maintain notational consistency, we use the language of $\mathsf{GID}$s.

that, for all $\lambda \in \mathbb{N}$,

$$\left\{ \mathcal{A}^{\mathcal{O}(\mathsf{MSK},\cdot,\cdot)}(\mathsf{CT}^*) : \begin{array}{l} s \leftarrow \mathcal{A}(1^\lambda), \\ \mathsf{MSK} \leftarrow \mathsf{Setup}(1^\lambda, 1^s), \\ C^* \leftarrow \mathcal{A}^{\mathcal{O}(\mathsf{MSK},\cdot,\cdot)}(1^\lambda), \\ \mathsf{CT}^* \leftarrow \mathsf{Enc}(\mathsf{MSK}, C^*) \end{array} \right\}$$

$$\approx_c$$

$$\left\{ \mathcal{A}^{\mathsf{Sim}^{C^*(\cdot)}(\mathsf{st},\cdot,\cdot)}(\mathsf{CT}^*) : \begin{array}{l} s \leftarrow \mathcal{A}(1^\lambda), \\ \mathsf{st} \leftarrow \mathsf{Sim}(1^\lambda, 1^s), \\ C^* \leftarrow \mathcal{A}^{\mathsf{Sim}(\mathsf{st},\cdot,\cdot)}(1^\lambda), \\ (\mathsf{CT}^*, \mathsf{st}) \leftarrow \mathsf{Sim}(1^{|C^*|}, \mathsf{V}) \end{array} \right\}$$

where $\mathsf{V} = \{(x, C^*(x)) : \text{if key generation for } x \text{ was queried before challenge query }\}$. $\mathcal{O}$ and $\mathsf{Sim}$ respond to the following queries from $\mathcal{A}$.

- $(\mathsf{Enc}, C)$. $\mathcal{A}$ queries for an encryption of circuit $C$ with size at most $s$. $\mathcal{O}$ responds with $\mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{MSK}, C)$. On the other hand, $\mathsf{Sim}$ responds with $\mathsf{CT} \leftarrow \mathsf{Sim}(\mathsf{st}, C)$.
- $(\mathsf{KGen}, x)$. $\mathcal{A}$ queries for secret key for input $x$. $\mathcal{O}$ responds with $\mathsf{SK}_x \leftarrow \mathsf{KGen}(\mathsf{MSK}, x)$ and $\mathsf{Sim}$ responds with $\mathsf{SK}_x \leftarrow \mathsf{Sim}(\mathsf{st}, x)$. In addition, if secret key query was done after challenge query for $C^*$, $\mathsf{Sim}$ uses $\mathsf{V} = \{(x, C^*(x)\}$.

A PPT machine $\mathcal{A}$ is said to be admissible if it makes at most on secret key query for $x$ and at most one challenge query for circuit $C^*$ of size at most $s$.

**Theorem F.2** ([GVW12, BS18])**.** A private-key one-key FE scheme for P/Poly circuits exists assuming the hardness of one-way functions.

## F.1 One-Key One-Challenge SIM-secure MCFE for P/Poly

In this section, we will construct a private-key MCFE scheme for P/Poly circuits where the scheme remains secure if at most one key for circuit $C \in \mathcal{C}_\lambda$ and at most one challenge ciphertext per client for inputs $x_1, \ldots, x_n$ are corrupted by the adversary. Moreover, there is a simulator that only utilizes $(C, C(x_1, \ldots, x_n))$ to simulate the challenge ciphertext and secret key. Specifically, we will construct an MCFE scheme that satisfies the following definition.

**Syntax.** A one-key private-key MCFE scheme 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.

$\mathsf{GSetup}(1^\lambda, 1^n, 1^{\ell_1}, \ldots, 1^{\ell_n}, 1^s) \to (\mathsf{CRS}, \mathsf{MSK}, \{\mathsf{EK}_{\mathsf{id}}\}_{\mathsf{id} \in [n]})$**.** The probabilistic global setup algorithm takes as input the security parameter $\lambda$, number of authorities $n$, input sizes for all authorities $\ell_1, \ldots, \ell_n$, the maximum circuit size $s$, outputs $\mathsf{CRS}$, master secret key $\mathsf{MSK}$, and encryption keys for all authorities, $\{\mathsf{EK}_{\mathsf{id}}\}_{\mathsf{id} \in [n]}$. The following algorithms take $\mathsf{CRS}$ implicitly.

$\mathsf{Enc}(\mathsf{id}, \mathsf{EK}_{\mathsf{id}}, \mathsf{GID}, x) \to \mathsf{CT}_{\mathsf{id},\mathsf{GID}}$**.** The probabilistic encryption algorithm takes as input the $\mathsf{id}$-th encryption key $\mathsf{EK}_{\mathsf{id}}$, the global identifier for the user $\mathsf{GID} \in \mathcal{GID}_\lambda$, an input $x \in \{0,1\}^{\ell_{\mathsf{id}}}$, and outputs ciphertext $\mathsf{CT}_{\mathsf{id},\mathsf{GID}}$.

$\mathsf{KGen}(\mathsf{MSK}, C) \to \mathsf{SK}_C$**.** The possibly randomized key generation algorithm takes as input the master secret key, $\mathsf{MSK}$, a circuit $C \in \mathcal{C}_\lambda$, and outputs secret key $\mathsf{SK}_C$.

$\mathsf{Dec}(\{\mathsf{CT}_{\mathsf{id},\mathsf{GID}}\}_{\mathsf{id}\in[n]}, \mathsf{SK}_C) \to y$**.** The deterministic decryption algorithm takes as input the ciphertexts for all authorities, $\{\mathsf{CT}_{\mathsf{id},\mathsf{GID}}\}_{\mathsf{id}\in[n]}$, secret key $\mathsf{SK}_C$, and outputs the value $y$.

**Definition F.3** (oMCFE)**.** An oMCFE scheme $(\mathsf{GSetup}, \mathsf{Enc}, \mathsf{KGen}, \mathsf{Dec})$ is said to be a one-key, one-challenge MCFE scheme for circuit class $\mathcal{C}$ and $\mathcal{GID}$ if it satisfies the following properties.

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

$$\Pr\left[\begin{array}{l} C(x_1,\dots x_n) = \\ \mathsf{Dec}(\{\mathsf{CT}_{\mathsf{id},\mathsf{GID}}\}_{\mathsf{id}}, \mathsf{SK}_C) \end{array} : \begin{array}{l} (\mathsf{CRS}, \mathsf{MSK}, \{\mathsf{EK}_{\mathsf{id}}\}_{\mathsf{id}}) \leftarrow \mathsf{GSetup}(1^\lambda, 1^n, 1^{\ell_1}, \dots, 1^{\ell_n}, 1^s), \\ \forall\ \mathsf{id} \in [n], \mathsf{CT}_{\mathsf{id},\mathsf{GID}} \leftarrow \mathsf{Enc}(\mathsf{id}, \mathsf{EK}_{\mathsf{id}}, \mathsf{GID}, x_{\mathsf{id}}), \\ \mathsf{SK}_C \leftarrow \mathsf{KGen}(\mathsf{MSK}, C) \end{array}\right] = 1$$

**SIM Security with Synchronous Encryption.** For any PPT admissible adversary $\mathcal{A}$, there exists a stateful PPT simulator $\mathsf{Sim}$ such that,

$$\left\{ \mathcal{A}^{\mathcal{O}(\cdot,\cdot)}(\{\mathsf{CT}^*_{\mathsf{id},\mathsf{GID}^*}\}_{\mathsf{id}\in[n]}) : \begin{array}{l} n, \ell_1, \dots, \ell_n, s \leftarrow \mathcal{A}(1^\lambda), \\ (\mathsf{CRS}, \mathsf{MSK}, \{\mathsf{EK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}) \leftarrow \mathsf{GSetup}(1^\lambda, 1^n, 1^{\ell_1}, \dots, 1^{\ell_n}, 1^s), \\ (\mathsf{GID}^*, x_1^*, \dots, x_n^*) \leftarrow \mathcal{A}^{\mathcal{O}(\cdot,\cdot)}(\mathsf{CRS}), \\ \forall\ \mathsf{id} \in [n], \mathsf{CT}^*_{\mathsf{id},\mathsf{GID}^*} \leftarrow \mathsf{Enc}(\mathsf{id}, \mathsf{EK}_{\mathsf{id}}, \mathsf{GID}^*, x^*_{\mathsf{id}}) \end{array} \right\}$$

$$\approx_c$$

$$\left\{ \mathcal{A}^{\mathsf{Sim}^{\mathcal{U}(\cdot, x_1^*, \dots, x_n^*)}(\mathsf{st},\cdot,\cdot)}(\{\mathsf{CT}^*_{\mathsf{id},\mathsf{GID}^*}\}_{\mathsf{id}\in[n]}) : \begin{array}{l} n, \ell_1, \dots, \ell_n, s \leftarrow \mathcal{A}(1^\lambda), \\ (\mathsf{CRS}, \mathsf{st}) \leftarrow \mathsf{Sim}(1^\lambda, 1^n, 1^{\ell_1}, \dots, 1^{\ell_n}, 1^s), \\ (\mathsf{GID}^*, x_1^*, \dots, x_n^*) \leftarrow \mathcal{A}^{\mathsf{Sim}(\mathsf{st},\cdot,\cdot)}(\mathsf{CRS}), \\ \{\mathsf{CT}^*_{\mathsf{id},\mathsf{GID}^*}\}_{\mathsf{id}\in[n]} \leftarrow \mathsf{Sim}(\mathsf{V}) \end{array} \right\}$$

where $\mathsf{V} = \{(C, C(x_1^*, \dots, x_n^*)) :$ if key generation for $C$ was queried before challenge query $\}$. $\mathcal{O}$ and $\mathsf{Sim}$ respond to the following queries from $\mathcal{A}$.

- $\underline{(\mathsf{Enc}, (\mathsf{GID}, x_1, \dots, x_n))}$. $\mathcal{A}$ queries for an encryption of inputs $x_1, \dots, x_n$ and $\mathsf{GID}$.. $\mathcal{O}$ responds with $\{\mathsf{CT}_{\mathsf{id},\mathsf{GID}}\}_{\mathsf{id}\in[n]}$ where $\mathsf{CT}_{\mathsf{id},\mathsf{GID}} \leftarrow \mathsf{Enc}(\mathsf{id}, \mathsf{EK}_{\mathsf{id}}, \mathsf{GID}, x_{\mathsf{id}})$. On the other hand, $\mathsf{Sim}$ responds with $\{\mathsf{CT}_{\mathsf{id},\mathsf{GID}}\}_{\mathsf{id}\in[n]} \leftarrow \mathsf{Sim}(\mathsf{st}, \mathsf{GID}, x_1, \dots, x_n)$.
- $\underline{(\mathsf{KGen}, x)}$. $\mathcal{A}$ queries for secret key for circuit $C$ of size at most $s$. $\mathcal{O}$ responds with $\mathsf{SK}_C \leftarrow \overline{\mathsf{KGen}(\mathsf{MSK}, C)}$ and $\mathsf{Sim}$ responds with $\mathsf{SK}_C \leftarrow \mathsf{Sim}(\mathsf{st}, C)$. In addition, if secret key query was done after challenge query for $(\mathsf{GID}^*, x_1^*, \dots, x_n^*)$, $\mathsf{Sim}$ uses $\mathsf{V} = \{(C, C(x_1^*, \dots, x_n^*)\}$.

A PPT machine $\mathcal{A}$ is said to be admissible if it makes at most one secret key query for $C$ of size at most $s$, at most one challenge query for $(\mathsf{GID}^*, x_1^*, \dots, x_n^*)$, and makes unbounded $\mathsf{Enc}$ queries with a unique $\mathsf{GID}$ each.

**Construction F.4.** We provide the construction of a MCFE scheme with synchronous encryption for P/Poly circuits and $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda\in\mathbb{N}}$ using a one-key private-key FE scheme for P/Poly circuits (Definition F.1), a pseudorandom function $\mathsf{PRF} : \{0,1\}^\lambda \times \mathcal{GID}_\lambda \to \{0,1\}^\lambda$ (Definition 3.1), and garbled circuit scheme $(\mathsf{Garb}, \mathsf{Eval})$ for P/Poly circuits (Definition 3.5) as follows.

$\underline{\mathsf{GSetup}(1^\lambda, 1^n, 1^{\ell_1}, \ldots, 1^{\ell_n}, 1^s)}$**.** Sample PRF key $K \xleftarrow{\$} \{0,1\}^\lambda$. For each $\mathsf{id} \in [n]$, sample $\text{1-fe.msk}_{\mathsf{id}} \leftarrow \text{1-FE.Setup}(1^\lambda, 1^{s'})$ where $s' = \mathsf{poly}(\lambda, n, \ell_1, \ldots, \ell_n, s)$ is the maximum size of the circuit in Figure 6. Set $\mathsf{CRS} := (\lambda, n, \ell_1, \ldots, \ell_n, s)$, $\mathsf{EK}_{\mathsf{id}} := (K, \text{1-fe.msk}_{\mathsf{id}})$ for each $\mathsf{id} \in [n]$, $\mathsf{MSK} = \{\text{1-fe.msk}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}$ and output $(\mathsf{CRS}, \mathsf{MSK}, \{\mathsf{EK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]})$.

$\underline{\mathsf{Enc}(\mathsf{id}, \mathsf{EK}_{\mathsf{id}}, \mathsf{GID}, x)}$**.** Parse $\mathsf{EK}_{\mathsf{id}}$ as $(K, \text{1-fe.msk}_{\mathsf{id}})$. Given $x$ such that $|x| \le \ell_{\mathsf{id}}$, compute $\alpha_{\mathsf{id}} = \left(\sum_{i=1}^{\mathsf{id}-1} \ell_i\right) + 1$ and $\mathbb{I}_{\mathsf{id}} = \left\{\left(\sum_{i=1}^{\mathsf{id}-1} \ell_i\right) + 1, \ldots, \left(\sum_{i=1}^{\mathsf{id}} \ell_i\right)\right\}$. Compute $R = \mathsf{PRF}(K, \mathsf{GID})$, and construct the circuit $F$ as specified in Figure 6.

Encrypt $F$ using 1-FE as $\text{1-fe.ct}_{\mathsf{id}} \leftarrow \text{1-FE.Enc}(\text{1-fe.msk}_{\mathsf{id}}, F)$. Output $\mathsf{CT}_{\mathsf{id},\mathsf{GID}} := (\mathsf{GID}, \text{1-fe.ct}_{\mathsf{id}})$.

$\underline{\mathsf{KGen}(\mathsf{MSK}, C)}$**.** Parse $\mathsf{MSK}$ as $\{\text{1-fe.msk}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}$.

For each $\mathsf{id} \in [n]$, sample $\text{1-fe.sk}_{C,\mathsf{id}} \leftarrow \text{1-FE.KGen}(\text{1-fe.msk}_{\mathsf{id}}, C)$. Output $\mathsf{SK}_C := \{\text{1-fe.sk}_{C,\mathsf{id}}\}_{\mathsf{id}\in[n]}$.

$\underline{\mathsf{Dec}(\mathsf{SK}_C, \{\mathsf{CT}_{\mathsf{id},\mathsf{GID}_{\mathsf{id}}}\}_{\mathsf{id}\in[n]})}$**.** Parse $\mathsf{SK}_C$ as $\{\text{1-fe.sk}_{C,\mathsf{id}}\}_{\mathsf{id}\in[n]}$ and $\mathsf{CT}_{\mathsf{id},\mathsf{GID}_{\mathsf{id}}}$ as $(\mathsf{GID}_{\mathsf{id}}, \text{1-fe.ct}_{\mathsf{id}})$ for each $\mathsf{id} \in [n]$. If all $\mathsf{GID}_{\mathsf{id}}$ are not the same, abort and output $\perp$. Otherwise, let $\mathsf{GID} := \mathsf{GID}_1$.

For each $\mathsf{id} \in [n]$, compute $y_{\mathsf{id}} := \text{1-FE.Dec}(\text{1-fe.sk}_{C,\mathsf{id}}, \text{1-fe.ct}_{\mathsf{id}})$. Parse each $y_{\mathsf{id}}$ as $(\widetilde{C}_{\mathsf{id}}, \overline{w}_{\mathsf{id}})$. If all $\widetilde{C}_{\mathsf{id}}$ are not the same, abort and output $\perp$. Otherwise, let $\widetilde{C} := \widetilde{C}_{\mathsf{id}}$.

Then, compute and output $y := \mathsf{Eval}(\widetilde{C}, \{\overline{w}_{\mathsf{id}}\}_{\mathsf{id}\in[n]})$.

**Circuit** $F$

**Input:** $C$
**Hardwired:** $R, \alpha, \mathbb{I}, x$
**Output:** Compute $\widetilde{C}, \{w_{i,b}\}_{b\in\{0,1\},i} = \mathsf{Garb}(1^\lambda, C; R)$. Output $\widetilde{C}, \{w_{i,x[i-\alpha+1]}\}_{i\in\mathbb{I}}$, where $x[j]$ is the $j$-th bit of $x \in \{0,1\}^{\ell_{\mathsf{id}}}$.

Figure 6: Description of circuit $F$

**Correctness.** The correctness of the scheme follows from the correctness of 1-FE, garbling scheme $(\mathsf{Garb}, \mathsf{Eval})$, and the deterministic nature of PRF. By correctness of 1-FE, $y_{\mathsf{id}} = (\widetilde{C}, \overline{w}_{\mathsf{id}})$ where $\widetilde{C}, \{\overline{w}_{\mathsf{id}}\}_{\mathsf{id}\in[n]} \leftarrow \mathsf{Garb}(1^\lambda, C; R)$ and $R = \mathsf{PRF}(K, \mathsf{GID})$. If all GID are the same, then so are $\widetilde{C}$ due to deterministic nature of PRF. Thus, by correctness of garbling scheme, $y = C(x_1, \ldots, x_n)$.

**Lemma F.5.** If 1-FE is an adaptively secure one-key private-key FE scheme for P/Poly circuits (Definition F.1), $\mathsf{PRF} : \{0,1\}^\lambda \times \mathsf{GID}_\lambda \to \{0,1\}^\lambda$ is a secure pseudorandom function (Definition 3.1), and $(\mathsf{Garb}, \mathsf{Eval})$ is a secure garbling scheme for P/Poly circuits (Definition 3.5), then Construction F.4 is an adaptively secure one-key MCFE scheme for P/Poly circuits and $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda\in\mathbb{N}}$.

*Proof.* We prove this lemma using the following hybrids and claims.

$\mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)$**.** This is almost the real experiment with honest challenger. We sample $\alpha_{\mathsf{id}}, \mathbb{I}_{\mathsf{id}}$ early during setup. Nevertheless, the distributions remain identical.

**Setup Phase.** $\mathcal{A}$ sends $(n, s, \ell_1, \ldots, \ell_n)$ to $\mathsf{Chal}$. $\mathsf{Chal}$ samples $\text{1-fe.msk}_{\mathsf{id}} \leftarrow \text{1-FE.Setup}(1^\lambda, 1^{s'})$. $\mathsf{Chal}$ also samples $K \leftarrow \{0,1\}^\lambda$.

$\mathsf{Chal}$ also computes $\alpha_{\mathsf{id}}, \mathbb{I}_{\mathsf{id}}$ as described in $\mathsf{Enc}$ of Construction F.4.

$\mathsf{Chal}$ sets $\mathsf{CRS} := (\lambda, n, \ell_1, \ldots, \ell_n, s)$ and $\mathsf{MSK} = \{\text{1-fe.msk}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}, \mathsf{EK}_{\mathsf{id}} := (K, \text{1-fe.msk}_{\mathsf{id}})$ for each $\mathsf{id} \in [n]$ and sends $\mathsf{CRS}$ to $\mathcal{A}$.

**Adaptive Queries.** $\mathcal{A}$ can now makes one of each of the following queries in an adaptive manner.

**Encryption Query.** $\mathcal{A}$ sends $(\mathsf{GID}, x_1, \ldots, x_n)$. $\mathsf{Chal}$ computes $R = \mathsf{PRF}(K, \mathsf{GID})$, constructs circuits $F_{\mathsf{id}}$, and samples $\text{1-fe.ct}_{\mathsf{id}} \leftarrow \text{1-FE.Enc}(\text{1-fe.msk}_{\mathsf{id}}, F_{\mathsf{id}})$ for each $\mathsf{id} \in [n]$. $\mathsf{Chal}$ sets $\mathsf{CT}_{\mathsf{id},\mathsf{GID}} := (\mathsf{GID}, \text{1-fe.ct}_{\mathsf{id}})$ and sends $\{\mathsf{CT}_{\mathsf{id},\mathsf{GID}}\}_{\mathsf{id}\in[n]}$ to $\mathcal{A}$.

**Challenge Query.** $\mathcal{A}$ sends $(\mathsf{GID}^*, x_1^*, \ldots, x_n^*)$. $\mathsf{Chal}$ computes $R^* = \mathsf{PRF}(K, \mathsf{GID}^*)$, constructs circuits $F_{\mathsf{id}}^*$, and samples $\text{1-fe.ct}_{\mathsf{id}}^* \leftarrow \text{1-FE.Enc}(\text{1-fe.msk}_{\mathsf{id}}, F_{\mathsf{id}}^*)$ for each $\mathsf{id} \in [n]$. $\mathsf{Chal}$ sets $\mathsf{CT}_{\mathsf{id},\mathsf{GID}^*}^* := (\mathsf{GID}^*, \text{1-fe.ct}_{\mathsf{id}}^*)$ and sends $\{\mathsf{CT}_{\mathsf{id},\mathsf{GID}^*}^*\}_{\mathsf{id}\in[n]}$ to $\mathcal{A}$.

**Secret Key Query.** $\mathcal{A}$ sends circuit $C$. $\mathsf{Chal}$ computes $\text{1-fe.sk}_{C,\mathsf{id}} \leftarrow \text{1-FE.KGen}(\text{1-fe.msk}_{\mathsf{id}}, C)$ for each $\mathsf{id} \in [n]$ and sends $\mathsf{SK}_C := \{\text{1-fe.sk}_{C,\mathsf{id}}\}_{\mathsf{id}\in[n]}$ to $\mathcal{A}$.

**Guess Phase.** $\mathcal{A}$ sends guess $b'$. Output $b'$.

**$\mathbf{Hyb}_{1,j}^{\mathcal{A}}(1^\lambda)$ for $j \in [n+1]$.** In this hybrid, we simulate the first $j-1$ instantiations of 1-FE. The changes are highlighted in red.

**Setup Phase.** $\mathcal{A}$ sends $(n, s, \ell_1, \ldots, \ell_n)$. For each $\mathsf{id} \in [n]$, if $\mathsf{id} < j$, sample $\text{st}_{\mathsf{id}} \leftarrow \text{1-FE.Sim}(1^\lambda, 1^{s'})$. Otherwise, if $\mathsf{id} \geq j$, sample $\text{1-fe.msk}_{\mathsf{id}} \leftarrow \text{1-FE.Setup}(1^\lambda, 1^{s'})$. Also sample $K \leftarrow \{0,1\}^\lambda$.

Compute $\alpha_{\mathsf{id}}, \mathbb{I}_{\mathsf{id}}$ as described in $\mathsf{Enc}$ of Construction F.4.

Set $\mathsf{CRS} := (\lambda, n, \ell_1, \ldots, \ell_n, s)$ and send $\mathsf{CRS}$ to $\mathcal{A}$.

**Adaptive Queries.** $\mathcal{A}$ can now makes one of each of the following queries in an adaptive manner.

**Encryption Query.** $\mathcal{A}$ sends $(\mathsf{GID}, x_1, \ldots, x_n)$. Compute $R = \mathsf{PRF}(K, \mathsf{GID})$, construct circuits $F_{\mathsf{id}}$. For each $\mathsf{id} \in [n]$,

- If $\mathsf{id} < j$, sample $\text{1-fe.ct}_{\mathsf{id}} \leftarrow \text{1-FE.Sim}(\text{st}_{\mathsf{id}}, \mathsf{Enc}, F_{\mathsf{id}})$.
- Otherwise, sample $\text{1-fe.ct}_{\mathsf{id}} \leftarrow \text{1-FE.Enc}(\text{1-fe.mpk}_{\mathsf{id}}, F_{\mathsf{id}})$.

Set $\mathsf{CT}_{\mathsf{id},\mathsf{GID}} := (\mathsf{GID}, \text{1-fe.ct}_{\mathsf{id}})$ and sends $\{\mathsf{CT}_{\mathsf{id},\mathsf{GID}}\}_{\mathsf{id}\in[n]}$ to $\mathcal{A}$.

**Challenge Query.** $\mathcal{A}$ sends $(\mathsf{GID}^*, x_1^*, \ldots, x_n^*)$. Compute $R^* = \mathsf{PRF}(K, \mathsf{GID}^*)$, construct circuits $F_{\mathsf{id}}^*$. For each $\mathsf{id} \in [n]$,

- If $\mathsf{id} < j$, sample $\text{1-fe.ct}_{\mathsf{id}}^* \leftarrow \text{1-FE.Sim}(\text{st}_{\mathsf{id}}, 1^{|F_{\mathsf{id}}^*|}, \mathsf{V})$ where $\mathsf{V} = F_{\mathsf{id}}^*(C)$ if $\mathcal{A}$ already queried for secret key. Otherwise, $\mathsf{V} = \bot$.
- Otherwise, sample $\text{1-fe.ct}_{\mathsf{id}}^* \leftarrow \text{1-FE.Enc}(\text{1-fe.msk}_{\mathsf{id}}, F_{\mathsf{id}}^*)$.

Set $\mathsf{CT}_{\mathsf{id},\mathsf{GID}^*}^* := (\mathsf{GID}^*, \text{1-fe.ct}_{\mathsf{id}}^*)$ and sends $\{\mathsf{CT}_{\mathsf{id},\mathsf{GID}^*}^*\}_{\mathsf{id}\in[n]}$ to $\mathcal{A}$.

**Secret Key Query.** $\mathcal{A}$ sends circuit $C$. For each $\mathsf{id} \in [n]$,

- If $\mathsf{id} < j$, sample $\text{1-fe.sk}_{C,\mathsf{id}} \leftarrow \text{1-FE.Sim}(\text{st}_{\mathsf{id}}, \mathsf{KGen}, C, \mathsf{V})$ where $\mathsf{V} = F_{\mathsf{id}}(C)$ if $\mathcal{A}$ already queried for encryption. Otherwise, $\mathsf{V} = \bot$.

- Otherwise, sample $\text{1-fe.sk}_{C,\mathsf{id}} \leftarrow \text{1-FE.KGen}(\text{1-fe.msk}_{\mathsf{id}}, C)$.

Send $\mathsf{SK}_C := \{\text{1-fe.sk}_{C,\mathsf{id}}\}_{\mathsf{id}\in[n]}$ to $\mathcal{A}$.

**Guess Phase.** $\mathcal{A}$ sends guess $b'$. Output $b'$.

**$\mathbf{Hyb}_2^{\mathcal{A}}(1^\lambda)$.** In this hybrid, we set $R$ to be a uniformly random string instead of PRF output on GID. The changes are highlighted in red.

**Setup Phase.** $\mathcal{A}$ sends $(n, s, \ell_1, \ldots, \ell_n)$. For each $\mathsf{id} \in [n]$, sample $\mathsf{st}_{\mathsf{id}} \leftarrow \text{1-FE.Sim}(1^\lambda, 1^{s'})$.

Compute $\alpha_{\mathsf{id}}, \mathbb{I}_{\mathsf{id}}$ as described in Enc of Construction F.4.

Set $\mathsf{CRS} := (\lambda, n, \ell_1, \ldots, \ell_n, s)$ and send CRS to $\mathcal{A}$.

**Adaptive Queries.** $\mathcal{A}$ can now makes one of each of the following queries in an adaptive manner.

**Encryption Query.** $\mathcal{A}$ sends $(\mathsf{GID}, x_1, \ldots, x_n)$. Construct the circuit $F_{\mathsf{id}}$ with $R \leftarrow \{0,1\}^\lambda$. Sample $\text{1-fe.ct}_{\mathsf{id}} \leftarrow \text{1-FE.Sim}(\mathsf{st}_{\mathsf{id}}, \mathsf{Enc}, F_{\mathsf{id}})$. Set $\mathsf{CT}_{\mathsf{id},\mathsf{GID}} := (\mathsf{GID}, \text{1-fe.ct}_{\mathsf{id}})$ and sends $\{\mathsf{CT}_{\mathsf{id},\mathsf{GID}}\}_{\mathsf{id}\in[n]}$ to $\mathcal{A}$.

**Challenge Query.** $\mathcal{A}$ sends $(\mathsf{GID}^*, x_1^*, \ldots, x_n^*)$. If $\mathcal{A}$ already queried for secret key with circuit $C$, sample $(\widetilde{C}, \{w_{i,b}\}_{i,b}) \leftarrow \mathsf{Garb}(1^\lambda, C; R)$ where $R \leftarrow \{0,1\}^\lambda$. Then, set $\mathsf{V}_{\mathsf{id}} = (\widetilde{C}, \overline{w}_{\mathsf{id}})$ for each $\mathsf{id} \in [n]]$. Otherwise, set $\mathsf{V}_{\mathsf{id}} = \bot$. Sample $\text{1-fe.ct}^*_{\mathsf{id}} \leftarrow \text{1-FE.Sim}(\mathsf{st}_{\mathsf{id}}, 1^{|F^*_{\mathsf{id}}|}, \mathsf{V}_{\mathsf{id}})$. Set $\mathsf{CT}^*_{\mathsf{id},\mathsf{GID}^*} := (\mathsf{GID}^*, \text{1-fe.ct}^*_{\mathsf{id}})$ and sends $\{\mathsf{CT}^*_{\mathsf{id},\mathsf{GID}^*}\}_{\mathsf{id}\in[n]}$ to $\mathcal{A}$.

**Secret Key Query.** $\mathcal{A}$ sends circuit $C$. If $\mathcal{A}$ already queried for encryption with input $\mathsf{GID}, x_1, \ldots, x_n$, sample $(\widetilde{C}, \{w_{i,b}\}_{i,b}) \leftarrow \mathsf{Garb}(1^\lambda, C; R)$ where $R \leftarrow \{0,1\}^\lambda$. Then, set $\mathsf{V}_{\mathsf{id}} = (\widetilde{C}, \overline{w}_{\mathsf{id}})$ for each $\mathsf{id} \in [n]]$. Otherwise, set $\mathsf{V}_{\mathsf{id}} = \bot$. For each $\mathsf{id} \in [n]$, sample $\text{1-fe.sk}_{C,\mathsf{id}} \leftarrow \text{1-FE.Sim}(\mathsf{st}_{\mathsf{id}}, \mathsf{KGen}, C, \mathsf{V}_{\mathsf{id}})$. Send $\mathsf{SK}_C := \{\text{1-fe.sk}_{C,\mathsf{id}}\}_{\mathsf{id}\in[n]}$ to $\mathcal{A}$.

**Guess Phase.** $\mathcal{A}$ sends guess $b'$. Output $b'$.

**$\mathbf{Hyb}_3^{\mathcal{A}}(1^\lambda)$.** In this hybrid, we will simulate the garbling scheme to use only $C(x_1, \ldots, x_n)$. This is the description of simulator Sim. The changes are highlighted in red.

**Setup Phase.** $\mathcal{A}$ sends $(n, s, \ell_1, \ldots, \ell_n)$. For each $\mathsf{id} \in [n]$, sample $\mathsf{st}_{\mathsf{id}} \leftarrow \text{1-FE.Sim}(1^\lambda, 1^{s'})$.

Compute $\alpha_{\mathsf{id}}, \mathbb{I}_{\mathsf{id}}$ as described in Enc of Construction F.4.

Set $\mathsf{CRS} := (\lambda, n, \ell_1, \ldots, \ell_n, s)$ and send CRS to $\mathcal{A}$.

**Adaptive Queries.** $\mathcal{A}$ can now makes one of each of the following queries in an adaptive manner.

**Encryption Query.** $\mathcal{A}$ sends $(\mathsf{GID}, x_1, \ldots, x_n)$. Construct the circuit $F_{\mathsf{id}}$ with $R \leftarrow \{0,1\}^\lambda$. Sample $\text{1-fe.ct}_{\mathsf{id}} \leftarrow \text{1-FE.Sim}(\mathsf{st}_{\mathsf{id}}, \mathsf{Enc}, F_{\mathsf{id}})$. Set $\mathsf{CT}_{\mathsf{id},\mathsf{GID}} := (\mathsf{GID}, \text{1-fe.ct}_{\mathsf{id}})$ and sends $\{\mathsf{CT}_{\mathsf{id},\mathsf{GID}}\}_{\mathsf{id}\in[n]}$ to $\mathcal{A}$.

**Challenge Query.** $\mathcal{A}$ sends $(\mathsf{GID}^*, x_1^*, \ldots, x_n^*)$. If $\mathcal{A}$ already queried for secret key with circuit $C$, sample $(\widetilde{C}, \{\overline{w}_{\mathsf{id}}\}_{\mathsf{id}}) \leftarrow \mathsf{Garb.Sim}(1^\lambda, 1^{|C|}, C(x_1^*, \ldots, x_n^*))$. Then, set $\mathsf{V}_{\mathsf{id}} = (\widetilde{C}, \overline{w}_{\mathsf{id}})$ for each $\mathsf{id} \in [n]]$. Otherwise, set $\mathsf{V}_{\mathsf{id}} = \bot$. Sample $\text{1-fe.ct}^*_{\mathsf{id}} \leftarrow \text{1-FE.Sim}(\mathsf{st}_{\mathsf{id}}, 1^{|F^*_{\mathsf{id}}|}, \mathsf{V}_{\mathsf{id}})$. Set $\mathsf{CT}^*_{\mathsf{id},\mathsf{GID}^*} := (\mathsf{GID}^*, \text{1-fe.ct}^*_{\mathsf{id}})$ and sends $\{\mathsf{CT}^*_{\mathsf{id},\mathsf{GID}^*}\}_{\mathsf{id}\in[n]}$ to $\mathcal{A}$.

**Secret Key Query.** $\mathcal{A}$ sends circuit $C$. If $\mathcal{A}$ already queried for encryption with input $\mathsf{GID}, x_1, \ldots, x_n$, sample $(\widetilde{C}, \{\overline{w}_{\mathsf{id}}\}_{\mathsf{id}}) \leftarrow \mathsf{Garb.Sim}(1^\lambda, 1^{|C|}, C(x_1^*, \ldots, x_n^*))$. Then, set $\mathsf{V}_{\mathsf{id}} = (\widetilde{C}, \overline{w}_{\mathsf{id}})$ for each $\mathsf{id} \in [n]]$. Otherwise, set $\mathsf{V}_{\mathsf{id}} = \bot$. For each $\mathsf{id} \in [n]$, sample $\text{1-fe.sk}_{C,\mathsf{id}} \leftarrow \text{1-FE.Sim}(\mathsf{st}_{\mathsf{id}}, \mathsf{KGen}, C, \mathsf{V}_{\mathsf{id}})$. Send $\mathsf{SK}_C := \{\text{1-fe.sk}_{C,\mathsf{id}}\}_{\mathsf{id}\in[n]}$ to $\mathcal{A}$.

**Guess Phase.** $\mathcal{A}$ sends guess $b'$. Output $b'$.

**Claim F.6.** $\mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_{1,1}^{\mathcal{A}}(1^\lambda)$ are identically distributed.

*Proof.* As we are not simulating any 1-FE instantiations in $\mathbf{Hyb}_{1,1}^{\mathcal{A}}(1^\lambda)$, these hybrids are identically distributed. □

**Claim F.7.** Assuming the security of 1-FE, $\mathbf{Hyb}_{1,j}^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_{1,j+1}^{\mathcal{A}}(1^\lambda)$ for any $j \in [n]$ are computationally indistinguishable.

*Proof.* Assume that there exists an adversary $\mathcal{A}$ that can distinguish between $\mathbf{Hyb}_{1,j}^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_{1,j+1}^{\mathcal{A}}(1^\lambda)$ for some $j \in [n]$ 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 security of 1-FE with non-negligible probability. The description of $\mathcal{B}^{\mathcal{O}}$ is as follows.

**Setup Phase.** $\mathcal{A}$ sends $(n, s, \ell_1, \ldots, \ell_n)$. For each $\mathsf{id} \in [n]$, if $\mathsf{id} < j$, sample $\mathsf{st}_{\mathsf{id}} \leftarrow \text{1-FE.Sim}(1^\lambda, 1^{s'})$. Otherwise, if $\mathsf{id} > j$, sample $\text{1-fe.msk}_{\mathsf{id}} \leftarrow \text{1-FE.Setup}(1^\lambda, 1^{s'})$. Also sample $K \leftarrow \{0,1\}^\lambda$.

Compute $\alpha_{\mathsf{id}}, \mathbb{I}_{\mathsf{id}}$ as described in $\mathsf{Enc}$ of Construction F.4.

Set $\mathsf{CRS} := (\lambda, n, \ell_1, \ldots, \ell_n, s)$ and send $\mathsf{CRS}$ to $\mathcal{A}$.

**Adaptive Queries.** $\mathcal{A}$ can now makes one of each of the following queries in an adaptive manner.

**Encryption Query.** $\mathcal{A}$ sends $(\mathsf{GID}, x_1, \ldots, x_n)$. Compute $R = \mathsf{PRF}(K, \mathsf{GID})$, construct circuits $F_{\mathsf{id}}$. For each $\mathsf{id} \in [n]$,

- If $\mathsf{id} < j$, sample $\text{1-fe.ct}_{\mathsf{id}} \leftarrow \text{1-FE.Sim}(\mathsf{st}_{\mathsf{id}}, \mathsf{Enc}, |F_{\mathsf{id}}|)$.
- Otherwise, if $\mathsf{id} = j$, send $F_{\mathsf{id}}$ to $\mathcal{O}$ to receive $\text{1-fe.ct}_{\mathsf{id}}$.
- Otherwise, sample $\text{1-fe.ct}_{\mathsf{id}} \leftarrow \text{1-FE.Enc}(\text{1-fe.msk}_{\mathsf{id}}, F_{\mathsf{id}})$.

Set $\mathsf{CT}_{\mathsf{id},\mathsf{GID}} := (\mathsf{GID}, \text{1-fe.ct}_{\mathsf{id}})$ and sends $\{\mathsf{CT}_{\mathsf{id},\mathsf{GID}}\}_{\mathsf{id}\in[n]}$ to $\mathcal{A}$.

**Challenge Query.** $\mathcal{A}$ sends $(\mathsf{GID}^*, x_1^*, \ldots, x_n^*)$. Compute $R = \mathsf{PRF}(K, \mathsf{GID}^*)$, construct circuits $F_{\mathsf{id}}^*$. For each $\mathsf{id} \in [n]$,

- If $\mathsf{id} < j$, sample $\text{1-fe.ct}_{\mathsf{id}}^* \leftarrow \text{1-FE.Sim}(\mathsf{st}_{\mathsf{id}}, 1^{|F_{\mathsf{id}}^*|}, \mathsf{V}_{\mathsf{id}})$ where $\mathsf{V} = F_{\mathsf{id}}(C)$ if $\mathcal{A}$ already queried for secret key for $C$. Otherwise, $\mathsf{V} = \bot$.
- Otherwise, if $\mathsf{id} = j$, send $F_{\mathsf{id}}$ to $\mathcal{O}$ to receive $\text{1-fe.ct}_{\mathsf{id}}^*$.
- Otherwise, sample $\text{1-fe.ct}_{\mathsf{id}}^* \leftarrow \text{1-FE.Enc}(\text{1-fe.msk}_{\mathsf{id}}, F_{\mathsf{id}}^*)$.

Set $\mathsf{CT}_{\mathsf{id},\mathsf{GID}^*}^* := (\mathsf{GID}^*, \text{1-fe.ct}_{\mathsf{id}}^*)$ and sends $\{\mathsf{CT}_{\mathsf{id},\mathsf{GID}^*}^*\}_{\mathsf{id}\in[n]}$ to $\mathcal{A}$.

**Secret Key Query.** $\mathcal{A}$ sends circuit $C$. For each $\mathsf{id} \in [n]$,

- If $\mathsf{id} < j$, sample $\text{1-fe.sk}_{C,\mathsf{id}} \leftarrow \text{1-FE.Sim}(\mathsf{st}_{\mathsf{id}}, \mathsf{KGen}, C, \mathsf{V})$ where $\mathsf{V} = F_{\mathsf{id}}(C)$ if $\mathcal{A}$ already queried for encryption. Otherwise, $\mathsf{V} = \bot$.
- Otherwise, if $\mathsf{id} = j$, send $C$ to $\mathcal{O}$ to receive $\text{1-fe.sk}_{C,\mathsf{id}}$.
- Otherwise, sample $\text{1-fe.sk}_{C,\mathsf{id}} \leftarrow \text{1-FE.KGen}(\text{1-fe.msk}_{\mathsf{id}}, C)$.

Send $\mathsf{SK}_C := \{\text{1-fe.sk}_{C,\mathsf{id}}\}_{\mathsf{id}\in[n]}$ to $\mathcal{A}$.

**Guess Phase.** $\mathcal{A}$ sends guess $b'$. Output $b'$.

As we can see, the running time of $\mathcal{B}$ is polynomial in the running time of $\mathcal{A}, \lambda$. If $\mathcal{O}$ is an honest challenger, $\mathcal{B}$ behaves like $\mathbf{Hyb}^{\mathcal{A}}_{1,j}(1^\lambda)$ and if $\mathcal{O}$ simulate 1-FE, then $\mathcal{B}$ behaves like $\mathbf{Hyb}^{\mathcal{A}}_{1,j+1}(1^\lambda)$. Hence, $\mathcal{B}$ is a valid adversary that breaks the security of 1-FE with non-negligible probability. Thus, $\mathbf{Hyb}^{\mathcal{A}}_{1,j}(1^\lambda)$ and $\mathbf{Hyb}^{\mathcal{A}}_{1,j+1}(1^\lambda)$ are computationally indistinguishable. □

**Claim F.8.** Assuming the security of PRF, $\mathbf{Hyb}^{\mathcal{A}}_{1,n+1}(1^\lambda)$ and $\mathbf{Hyb}^{\mathcal{A}}_{2}(1^\lambda)$ are computationally indistinguishable.

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

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

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

**Setup Phase.** $\mathcal{A}$ sends $(n, s, \ell_1, \ldots, \ell_n)$. For each $\mathsf{id} \in [n]$, sample $\mathsf{st}_{\mathsf{id}} \leftarrow \text{1-FE.Sim}(1^\lambda, 1^{s'})$.

Compute $\alpha_{\mathsf{id}}, \mathbb{I}_{\mathsf{id}}$ as described in Enc of Construction F.4.

Set $\mathsf{CRS} := (\lambda, n, \ell_1, \ldots, \ell_n, s)$ and send CRS to $\mathcal{A}$.

**Adaptive Queries.** $\mathcal{A}$ can now makes one of each of the following queries in an adaptive manner.

**Encryption Query.** $\mathcal{A}$ sends $(\mathsf{GID}, x_1, \ldots, x_n)$. Construct the circuit $F_{\mathsf{id}}$ with $R \leftarrow \mathcal{O}(\mathsf{GID})$. Sample $\text{1-fe.ct}_{\mathsf{id}} \leftarrow \text{1-FE.Sim}(\mathsf{st}_{\mathsf{id}}, \mathsf{Enc}, F_{\mathsf{id}})$. Set $\mathsf{CT}_{\mathsf{id},\mathsf{GID}} := (\mathsf{GID}, \text{1-fe.ct}_{\mathsf{id}})$ and sends $\{\mathsf{CT}_{\mathsf{id},\mathsf{GID}}\}_{\mathsf{id}\in[n]}$ to $\mathcal{A}$.

**Challenge Query.** $\mathcal{A}$ sends $(\mathsf{GID}^*, x_1^*, \ldots, x_n^*)$. If $\mathcal{A}$ already queried for secret key with circuit $C$, sample $(\widetilde{C}, \{w_{i,b}\}_{i,b}) \leftarrow \mathsf{Garb}(1^\lambda, C; R)$ where $R \leftarrow \mathcal{O}(\mathsf{GID})$. Then, set $\mathsf{V}_{\mathsf{id}} = (\widetilde{C}, \overline{w}_{\mathsf{id}})$ for each $\mathsf{id} \in [n]]$. Otherwise, set $\mathsf{V}_{\mathsf{id}} = \bot$. Sample $\text{1-fe.ct}^*_{\mathsf{id}} \leftarrow \text{1-FE.Sim}(\mathsf{st}_{\mathsf{id}}, 1^{|F^*_{\mathsf{id}}|}, \mathsf{V}_{\mathsf{id}})$. Set $\mathsf{CT}^*_{\mathsf{id},\mathsf{GID}^*} := (\mathsf{GID}^*, \text{1-fe.ct}^*_{\mathsf{id}})$ and sends $\{\mathsf{CT}^*_{\mathsf{id},\mathsf{GID}^*}\}_{\mathsf{id}\in[n]}$ to $\mathcal{A}$.

**Secret Key Query.** $\mathcal{A}$ sends circuit $C$. If $\mathcal{A}$ already queried for encryption with input $\mathsf{GID}, x_1, \ldots, x_n$, sample $(\widetilde{C}, \{w_{i,b}\}_{i,b}) \leftarrow \mathsf{Garb}(1^\lambda, C; R)$ where $R \leftarrow \mathcal{O}(\mathsf{GID}^*)$. Then, set $\mathsf{V}_{\mathsf{id}} = (\widetilde{C}, \overline{w}_{\mathsf{id}})$ for each $\mathsf{id} \in [n]]$. Otherwise, set $\mathsf{V}_{\mathsf{id}} = \bot$. For each $\mathsf{id} \in [n]$, sample $\text{1-fe.sk}_{C,\mathsf{id}} \leftarrow \text{1-FE.Sim}(\mathsf{st}_{\mathsf{id}}, \mathsf{KGen}, C, \mathsf{V}_{\mathsf{id}})$. Send $\mathsf{SK}_C := \{\text{1-fe.sk}_{C,\mathsf{id}}\}_{\mathsf{id}\in[n]}$ to $\mathcal{A}$.

**Guess Phase.** $\mathcal{A}$ sends guess $b'$. Output $b'$.

As we can see, the running time of $\mathcal{B}$ is polynomial in the running time of $\mathcal{A}, \lambda$. If $\mathcal{O}$ is sends pseudorandom strings with some $K \leftarrow \{0,1\}^\lambda$, $\mathcal{B}$ behaves like $\mathbf{Hyb}^{\mathcal{A}}_{1,n+1}(1^\lambda)$ and if $\mathcal{O}$ sends uniformly random strings, then $\mathcal{B}$ behaves like $\mathbf{Hyb}^{\mathcal{A}}_{2}(1^\lambda)$. Hence, $\mathcal{B}$ is a valid adversary that breaks the security of $\mathsf{PRF}$ with non-negligible probability. Thus, $\mathbf{Hyb}^{\mathcal{A}}_{1,n+1}(1^\lambda)$ and $\mathbf{Hyb}^{\mathcal{A}}_{2}(1^\lambda)$ are computationally indistinguishable. □

**Claim F.9.** Assuming the security of garbling scheme $(\mathsf{Garb}, \mathsf{Eval})$, $\mathbf{Hyb}^{\mathcal{A}}_{2}(1^\lambda)$ and $\mathbf{Hyb}^{\mathcal{A}}_{3}(1^\lambda)$ are computationally indistinguishable.

*Proof.* Assume that there exists an adversary $\mathcal{A}$ that can distinguish between $\mathbf{Hyb}^{\mathcal{A}}_{2}(1^\lambda)$ and $\mathbf{Hyb}^{\mathcal{A}}_{3}(1^\lambda)$ with non-negligible probability $\epsilon(\lambda)$, 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 $(\mathsf{Garb}, \mathsf{Eval})$ with non-negligible probability. The description of $\mathcal{B}^{\mathcal{O}}$ is as follows.

**Setup Phase.** $\mathcal{A}$ sends $(n, s, \ell_1, \ldots, \ell_n)$. For each $\mathsf{id} \in [n]$, sample $\mathsf{st}_{\mathsf{id}} \leftarrow \text{1-}\mathsf{FE.Sim}(1^\lambda, 1^{s'})$.

Compute $\alpha_{\mathsf{id}}, \mathbb{I}_{\mathsf{id}}$ as described in $\mathsf{Enc}$ of Construction F.4.

Set $\mathsf{CRS} := (\lambda, n, \ell_1, \ldots, \ell_n, s)$ and send $\mathsf{CRS}$ to $\mathcal{A}$.

**Adaptive Queries.** $\mathcal{A}$ can now makes one of each of the following queries in an adaptive manner.

- **Encryption Query.** $\mathcal{A}$ sends $(\mathsf{GID}, x_1, \ldots, x_n)$. Construct the circuit $F_{\mathsf{id}}$ with $R \leftarrow \{0,1\}^\lambda$. Sample $\text{1-}\mathsf{fe.ct}_{\mathsf{id}} \leftarrow \text{1-}\mathsf{FE.Sim}(\mathsf{st}_{\mathsf{id}}, \mathsf{Enc}, F_{\mathsf{id}})$. Set $\mathsf{CT}_{\mathsf{id},\mathsf{GID}} := (\mathsf{GID}, \text{1-}\mathsf{fe.ct}_{\mathsf{id}})$ and sends $\{\mathsf{CT}_{\mathsf{id},\mathsf{GID}}\}_{\mathsf{id}\in[n]}$ to $\mathcal{A}$.
- **Challenge Query.** $\mathcal{A}$ sends $(\mathsf{GID}^*, x^*_1, \ldots, x^*_n)$. If $\mathcal{A}$ already queried for secret key with circuit $C$, sample $(\widetilde{C}, \{\overline{w}_{\mathsf{id}}\}_{\mathsf{id}}) \leftarrow \mathcal{O}(C, (x^*_1, \ldots, x^*_n))$. Then, set $\mathsf{V}_{\mathsf{id}} = (\widetilde{C}, \overline{w}_{\mathsf{id}})$ for each $\mathsf{id} \in [n]]$. Otherwise, set $\mathsf{V}_{\mathsf{id}} = \bot$. Sample $\text{1-}\mathsf{fe.ct}^*_{\mathsf{id}} \leftarrow \text{1-}\mathsf{FE.Sim}(\mathsf{st}_{\mathsf{id}}, 1^{|F^*_{\mathsf{id}}|}, \mathsf{V}_{\mathsf{id}})$. Set $\mathsf{CT}^*_{\mathsf{id},\mathsf{GID}^*} := (\mathsf{GID}^*, \text{1-}\mathsf{fe.ct}^*_{\mathsf{id}})$ and sends $\{\mathsf{CT}^*_{\mathsf{id},\mathsf{GID}^*}\}_{\mathsf{id}\in[n]}$ to $\mathcal{A}$.
- **Secret Key Query.** $\mathcal{A}$ sends circuit $C$. If $\mathcal{A}$ already queried for encryption with input $\mathsf{GID}, x_1, \ldots, x_n$, sample $(\widetilde{C}, \{\overline{w}_{\mathsf{id}}\}_{\mathsf{id}}) \leftarrow \mathcal{O}(C, (x^*_1, \ldots, x^*_n))$. Then, set $\mathsf{V}_{\mathsf{id}} = (\widetilde{C}, \overline{w}_{\mathsf{id}})$ for each $\mathsf{id} \in [n]]$. Otherwise, set $\mathsf{V}_{\mathsf{id}} = \bot$. For each $\mathsf{id} \in [n]$, sample $\text{1-}\mathsf{fe.sk}_{C,\mathsf{id}} \leftarrow \text{1-}\mathsf{FE.Sim}(\mathsf{st}_{\mathsf{id}}, \mathsf{KGen}, C, \mathsf{V}_{\mathsf{id}})$. Send $\mathsf{SK}_C := \{\text{1-}\mathsf{fe.sk}_{C,\mathsf{id}}\}_{\mathsf{id}\in[n]}$ to $\mathcal{A}$.

**Guess Phase.** $\mathcal{A}$ sends guess $b'$. Output $b'$.

As we can see, the running time of $\mathcal{B}$ is polynomial in the running time of $\mathcal{A}, \lambda$. If $\mathcal{O}$ is an honest challenger, $\mathcal{B}$ behaves like $\mathbf{Hyb}^{\mathcal{A}}_{2}(1^\lambda)$ and if $\mathcal{O}$ simulates garbling, then $\mathcal{B}$ behaves like $\mathbf{Hyb}^{\mathcal{A}}_{3}(1^\lambda)$. Hence, $\mathcal{B}$ is a valid adversary that breaks the security of $(\mathsf{Garb}, \mathsf{Eval})$ with non-negligible probability. Thus, $\mathbf{Hyb}^{\mathcal{A}}_{2}(1^\lambda)$ and $\mathbf{Hyb}^{\mathcal{A}}_{3}(1^\lambda)$ are computationally indistinguishable. □

□

## F.2 One-Key Many-Challenge MCFE for P/Poly

In this section, we will move towards IND security rather than SIM security from previous section. This allows us to query multiple challenge inputs rather than a single challenge input via a couple of standard hybrid arguments. We will show that the MCFE construction from previous section already satisfies these IND secure versions defined below.

**Definition F.10.** An MCFE scheme $(\mathsf{GSetup}, \mathsf{Enc}, \mathsf{KGen}, \mathsf{Dec})$ is said to be a one-key, one-challenge MCFE scheme for circuit class $\mathcal{C}$ and $\mathcal{GID}$ with IND security if it satisfies the correctness property from Definition F.3 and also the following IND-security property.

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

$$\left| \Pr \left[ b \leftarrow \mathcal{A}^{\mathcal{O}(\cdot,\cdot)}(\mathsf{CT}^*) \quad : \quad \begin{array}{l} b \leftarrow \{0,1\}, n, \ell_1, \ldots, \ell_n, s \leftarrow \mathcal{A}(1^\lambda), \\ (\mathsf{GID}^*, (x^*_{0,1}, \ldots, x^*_{0,n}), \\ (x^*_{1,1}, \ldots, x^*_{1,n})) \leftarrow \mathcal{A}^{\mathcal{O}(\cdot,\cdot)}(\mathsf{CRS}), \\ \forall\ \mathsf{id} \in [n], \mathsf{CT}^*_{\mathsf{id},\mathsf{GID}^*} \leftarrow \mathsf{Enc}(\mathsf{id}, \mathsf{EK}_{\mathsf{id}}, \mathsf{GID}^*, x^*_{b,\mathsf{id}}), \\ \text{Let } \mathsf{CT}^* := \{\mathsf{CT}^*_{\mathsf{id},\mathsf{GID}^*}\}_{\mathsf{id}\in[n]} \end{array} \right] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda)$$

where $(\mathsf{CRS}, \mathsf{MSK}, \{\mathsf{EK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}) \leftarrow \mathsf{GSetup}(1^\lambda, 1^n, 1^{\ell_1}, \ldots, 1^{\ell_n}, 1^s)$. $\mathcal{O}$ responds to the following queries from $\mathcal{A}$.

- $\underline{(\mathsf{Enc}, (\mathsf{GID}, x_1, \ldots, x_n))}$. $\mathcal{A}$ queries for an encryption of inputs $x_1, \ldots, x_n$ and $\mathsf{GID}$. $\mathcal{O}$ responds with $\overline{\{\mathsf{CT}_{\mathsf{id},\mathsf{GID}}\}_{\mathsf{id}\in[n]}}$ where $\mathsf{CT}_{\mathsf{id},\mathsf{GID}} \leftarrow \mathsf{Enc}(\mathsf{id}, \mathsf{EK}_{\mathsf{id}}, \mathsf{GID}, x_{\mathsf{id}})$.
- $\underline{(\mathsf{KGen}, C)}$. $\mathcal{A}$ queries for secret key for circuit $C$ of size at most $s$. $\mathcal{O}$ responds with $\overline{\mathsf{SK}_C \leftarrow \mathsf{KGen}(\mathsf{MSK}, C)}$.

A $\mathsf{PPT}$ machine $\mathcal{A}$ is said to be admissible if it makes at most one $\mathsf{KGen}$ query for $C$ of size at most $s$, at most one challenge query for $(\mathsf{GID}^*, (x^*_{0,1}, \ldots, x^*_{0,n}), (x^*_{1,1}, \ldots, x^*_{1,n}))$ such that $C(x^*_{0,1}, \ldots, x^*_{0,n}) = C(x^*_{1,1}, \ldots, x^*_{1,n})$, and makes unbounded $\mathsf{Enc}$ queries with a unique $\mathsf{GID}$ each.

**Lemma F.11.** Any MCFE scheme for $\mathsf{P/Poly}$ circuits that satisfies Definition F.3 satisfies Definition F.10.

*Proof.* This follows from SIM-secure MCFE with synchronous encryption. Concretely, in order to prove IND-Security between $b = 0$ and $b = 1$, we can use an intermediate hybrid in which we move using $\mathsf{oMCFE.Sim}$. This change is computationally indistinguishable by any $\mathsf{PPT}$ $\mathcal{A}$. Moreover, as $C(x^*_{0,1}, \ldots, x^*_{0,n}) = C(x^*_{1,1}, \ldots, x^*_{1,n})$, any adversary has zero advantage and this lemma follows. □

Now, we will lift an IND-secure one-key one-challenge MCFE scheme to a one-key many-challenge MCFE by a standard hybrid argument.

**Definition F.12.** An MCFE scheme $(\mathsf{GSetup}, \mathsf{Enc}, \mathsf{KGen}, \mathsf{Dec})$ is said to be a one-key, *many*-challenge MCFE scheme for circuit class $\mathcal{C}$ and $\mathcal{GID}$ with IND security if it satisfies the correctness property from Definition F.3 and also the following IND-security property.

**Many-Challenge IND Security with Synchronous Encryption.** For any PPT admissible adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

$$\left|\Pr\left[b \leftarrow \mathcal{A}^{\mathcal{O}_b(\cdot,\cdot)}(\mathsf{CRS}) \quad : \quad \begin{array}{l} b \leftarrow \{0,1\}, n, \ell_1, \ldots, \ell_n, s \leftarrow \mathcal{A}(1^\lambda), \\ (\mathsf{CRS}, \mathsf{MSK}, \{\mathsf{EK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}) \leftarrow \\ \mathsf{GSetup}(1^\lambda, 1^n, 1^{\ell_1}, \ldots, 1^{\ell_n}, 1^s) \end{array}\right] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda)$$

where $\mathcal{O}_b$ responds to the following queries from $\mathcal{A}$.

- $\underline{(\mathsf{Enc}, (\mathsf{GID}, (x_{0,1}, \ldots, x_{0,n}), (x_{1,1}, \ldots, x_{1,n}))}$. $\mathcal{A}$ queries for an encryption of inputs for $\mathsf{GID}$. $\mathcal{O}$ responds with $\{\mathsf{CT}_{\mathsf{id},\mathsf{GID}}\}_{\mathsf{id}\in[n]}$ where $\underline{\mathsf{CT}_{\mathsf{id},\mathsf{GID}} \leftarrow \mathsf{Enc}(\mathsf{id}, \mathsf{EK}_{\mathsf{id}}, \mathsf{GID}, x_{b,\mathsf{id}})}$.
- $\underline{(\mathsf{KGen}, C)}$. $\mathcal{A}$ queries for secret key for circuit $C$ of size at most $s$. $\mathcal{O}$ responds with $\underline{\mathsf{SK}_C \leftarrow \mathsf{KGen}(\mathsf{MSK}, C)}$.

A PPT machine $\mathcal{A}$ is said to be admissible if it makes at most one $\mathsf{KGen}$ query for $C$ of size at most $s$, and unbounded encryption queries with a unique $\mathsf{GID}$ each $(\mathsf{GID}, (x_{0,1}, \ldots, x_{0,n}), (x_{1,1}, \ldots, x_{1,n}))$ such that $C(x_{0,1}, \ldots, x_{0,n}) = C(x_{1,1}, \ldots, x_{1,n})$ for each of them.

**Lemma F.13.** Any MCFE scheme for P/Poly circuits that satisfies Definition F.10 satisfies Definition F.12.

*Proof.* This follows by a standard hybrid argument. □

## F.3 One-CT Private-Key MAFE

Now, we are finally ready to construct a one-ciphertext private-key MAFE for P/Poly circuits that satisfies Definition 6.6. The idea is to use function-hiding techniques similar to [BS18]. We provide full construction below.

**Construction F.14** (opMAFE)**.** We provide the construction of a one-ciphertext private-key MAFE scheme with synchronous key generation for P/Poly circuits and $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda\in\mathbb{N}}$ (Definition 6.6) using a one-key private-key MCFE scheme for P/Poly circuits (Definition F.12) and $\mathcal{GID}$ and a secret-key encryption scheme $\mathsf{SKE}$ (Definition 3.3) as follows.

$\underline{\mathsf{GSetup}(1^\lambda, 1^n, 1^s, 1^{\ell_1}, \ldots, 1^{\ell_n})}$**.** Let $s' = \mathsf{poly}(\lambda, n, s, \ell_1, \ldots, \ell_n)$ be the maximum size of the circuit described in Figure 7 and $\ell'_{\mathsf{id}} = \mathsf{poly}(\lambda, \ell_{\mathsf{id}})$ is the input size of $\mathsf{id}$-th authority. Sample MCFE keys $(\mathsf{mcfe.crs}, \{\mathsf{mcfe.ek}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}, \mathsf{mcfe.msk}) \leftarrow \mathsf{oMCFE.GSetup}(1^\lambda, 1^n, 1^{s'}, 1^{\ell'_1}, \ldots, 1^{\ell'_n})$. Sample two SKE key $\mathsf{ske.sk}, \mathsf{ske.sk}' \leftarrow \mathsf{SKE.Setup}(1^\lambda)$. Set $\mathsf{CRS} := \mathsf{mcfe.crs}$, $\mathsf{MSK}_{\mathsf{id}} := (\mathsf{mcfe.ek}_{\mathsf{id}}, \mathsf{mcfe.msk}, \mathsf{ske.sk}, \mathsf{ske.sk}')$ for each $\mathsf{id} \in [n]$.

Output $(\mathsf{CRS}, \{\mathsf{MSK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]})$.

$\underline{\mathsf{KGen}(\mathsf{id}, \mathsf{MSK}_{\mathsf{id}}, \mathsf{GID}, x)}$**.** Parse $\mathsf{MSK}_{\mathsf{id}}$ as $(\mathsf{mcfe.ek}_{\mathsf{id}}, \mathsf{mcfe.msk}, \mathsf{ske.sk}, \mathsf{ske.sk}')$. Sample encryption for MCFE, $\mathsf{mcfe.ct}_{\mathsf{id},\mathsf{GID}} \leftarrow \mathsf{oMCFE.Enc}(\mathsf{id}, \mathsf{mcfe.ek}_{\mathsf{id}}, \mathsf{GID}, (x, \mathsf{ske.sk}, \bot, \bot))$. Output $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} := (\mathsf{id}, \mathsf{GID}, \mathsf{mcfe.ct}_{\mathsf{id},\mathsf{GID}})$.

$\underline{\mathsf{Enc}(\{\mathsf{MSK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}, C)}$**.** Parse $\mathsf{MSK}_{\mathsf{id}}$ as $(\mathsf{mcfe.ek}_{\mathsf{id}}, \mathsf{mcfe.msk}, \mathsf{ske.sk}, \mathsf{ske.sk}')$ for each $\mathsf{id} \in [n]$. Sample ciphertexts $\mathsf{ske.ct} \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}, C)$ and $\mathsf{ske.ct}' \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.}sk', C)$. Construct the circuit $F$ as shown in Figure 7. Sample MCFE secret key $\mathsf{mcfe.sk}_F \leftarrow \mathsf{oMCFE.KGen}(\mathsf{mcfe.msk}, F)$. Output $\mathsf{CT} := \mathsf{mcfe.sk}_F$.

$\underline{\mathsf{Dec}(\{\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}\}_{\mathsf{id}\in[n]},\mathsf{CT})}$. Parse $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}$ as $(\mathsf{id},\mathsf{GID},\mathsf{mcfe.ct}_{\mathsf{id},\mathsf{GID}})$ for each $\mathsf{id}\in[n]$ and $\mathsf{CT}$ as $\mathsf{mcfe.sk}_F$. Output $y=\mathsf{oMCFE.Dec}(\{\mathsf{mcfe.ct}_{\mathsf{id},\mathsf{GID}}\}_{\mathsf{id}\in[n]},\mathsf{mcfe.sk}_F)$.

**Circuit $F$**

**Input:** $(x_1,\mathsf{ske.sk}_1,x'_1,\mathsf{ske.sk}'_1),\ldots,(x_n,\mathsf{ske.sk}_n,x'_n,\mathsf{ske.sk}'_n)$
**Hardwired:** $\mathsf{ske.ct},\mathsf{ske.ct}'$
**Output:** If all of $\mathsf{ske.sk}_{\mathsf{id}}$ are not the same or all of $\mathsf{ske.sk}'_{\mathsf{id}}$ are not the same for $\mathsf{id}\in[n]$, output $\bot$. Otherwise, let $\mathsf{ske.sk}=\mathsf{ske.sk}_1,\mathsf{ske.sk}'=\mathsf{ske.sk}'_1$.

- If $\mathsf{ske.sk}\neq\bot$, compute $C=\mathsf{SKE.Dec}(\mathsf{ske.sk},\mathsf{ske.ct})$. Output $C(x_1,\ldots,x_n)$.
- Otherwise, if $\mathsf{ske.sk}'\neq\bot$, compute $C=\mathsf{SKE.Dec}(\mathsf{ske.sk}',\mathsf{ske.ct}')$. Output $C(x'_1,\ldots,x'_n)$.
- Otherwise, output $\bot$.

Figure 7: Description of circuit $F$

**Correctness.** Correctness of the scheme follows from correctness of SKE and oMCFE. By correctness of oMCFE, $y=F((x_1,\mathsf{ske.sk},\bot,\bot),\ldots,(x_n,\mathsf{ske.sk},\bot,\bot))$. By the functionality of $F$ and correctness of SKE, we have that $y=C(x_1,\ldots,x_n)$.

**Lemma F.15.** If oMCFE is an MCFE scheme for P/Poly circuits and $\mathcal{GID}$ (Definition F.12), SKE is an IND-CPA secure SKE (Definition 3.3), then Construction F.14 is a one-ciphertext private-key MAFE scheme for P/Poly circuits and $\mathcal{GID}=\{\mathcal{GID}_\lambda\}_{\lambda\in\mathbb{N}}$.

*Proof.* Proof proceeds similar to [BS18]. Hence, we only provide an overview of the hybrids.

**Hybrid 0.** This is the experiment from Definition 6.6 with $b=0$.

**Hybrid 1.** In this hybrid, as we only encrypt $(x_{\mathsf{id}},\mathsf{ske.sk},\bot,\bot)$ from previous experiment, $\mathsf{ske.sk}'$ is completely hidden from $\mathcal{A}$'s view. Hence, we can set $\mathsf{ske.ct}'$ to be encryption of $C_1$. The indistinguishability readily holds by IND-CPA security of SKE.

**Hybrid 2.** In this hybrid, we leverage the fact that $C_0(x_{1,\mathsf{GID}},\ldots,x_{n,\mathsf{GID}})=C_1(x_{1,\mathsf{GID}},\ldots,x_{n,\mathsf{GID}})$ for each GID query and encrypt $(\bot,\bot,x_{\mathsf{id},\mathsf{GID}},\mathsf{ske.sk}')$. By functionality of $F$ (Figure 7), outputs are equal. Hence, indistinguishability can be argued using many-challenge IND security with synchronous encryption property of oMCFE.

**Hybrid 3.** In the hybrid, as we only encrypt $(\bot,\bot,x_{\mathsf{id}},\mathsf{ske.sk}')$ from previous experiment, $\mathsf{ske.sk}$ is completely hidden from $\mathcal{A}$'s view. Hence, we can set $\mathsf{ske.ct}$ to be encryption of $C_1$. The indistinguishability readily holds by IND-CPA security of SKE.

**Hybrid 4.** This is the experiment with $b=1$. Indistinguishability can be argued using many-challenge IND security with synchronous encryption property of oMCFE.

□

**Circuit $G'$**

**Input:** $1, \mathsf{GID}_1, x_1, \rho_1, \mathsf{ske.ct}^*_1, \ldots, n, \mathsf{GID}_n, x_n, \rho_n, \mathsf{ske.ct}^*_n$
**Hardwired:** $\mathsf{ske.sk}^*$
**Output:**

- For each $\mathsf{id} \in [n]$, compute $\mathsf{pmafe.sk}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}} = \mathsf{SKE.Dec}(\mathsf{ske.sk}^*, \mathsf{ske.ct}^*_{\mathsf{id}})$.
- Output $\{\mathsf{pmafe.sk}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}\}_{\mathsf{id}\in[n]}$.

Figure 8: Description of circuit $G'$

# G Proofs from Section 6.4

In this section, we provide full proof of Lemma 6.8.

**$\mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)$.** This is almost the real experiment with Chal with $b = 0/1$. We sample $K$ and opMAFE keys early. Nonetheless, distribution remains identical.

**Setup Phase.** $\mathcal{A}$ sends $(n, s, \ell_1, \ldots, \ell_n)$ to Chal. Define $L = \max_{\mathsf{id}}\{\ell_{\mathsf{id}}\}$. Chal samples $\mathsf{selmafe.crs} \leftarrow \mathsf{selsMAFE.GSetup}(1^\lambda, 1^n, 1^{s'}, 1^L)$ where $s' = \mathsf{poly}(\lambda, n, L, s)$ is the maximum size of circuit from Figure 4 and Figure 8. Set $\mathsf{CRS} := \mathsf{selmafe.crs}$. Then, Chal samples $(\mathsf{selmafe.mpk}_{\mathsf{id}}, \mathsf{selmafe.msk}_{\mathsf{id}}) \leftarrow \mathsf{selsMAFE.ASetup}(\mathsf{id}, 1^{\ell'_{\mathsf{id}}})$ for each $\mathsf{id} \in [n]$. Chal sets $\mathsf{MPK}_{\mathsf{id}} := \mathsf{selmafe.mpk}_{\mathsf{id}}$ and $\mathsf{MSK}_{\mathsf{id}} := \mathsf{selmafe.msk}_{\mathsf{id}}$.

Now, Chal samples $K \xleftarrow{\$} \{0,1\}^\lambda$ and $\mathsf{pmafe.crs}, \{\mathsf{pmafe.msk}_{\mathsf{id}}\}_{\mathsf{id}\in[n]} \leftarrow \mathsf{opMAFE.GSetup}(1^\lambda, 1^n, 1^s, 1^L)$. Construct circuit $G$ from Figure 4.

Chal sends $(\mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]})$ to $\mathcal{A}$.

**Pre-challenge Queries.** $\mathcal{A}$ sends queries of the form $(\mathsf{GID}, x_1, \ldots, x_n)$.

Chal now samples $\rho_{\mathsf{id}} \xleftarrow{\$} \{0,1\}^\lambda, \mathsf{ske.ct}^*_{\mathsf{id}} \xleftarrow{\$} \{0,1\}^{\ell^*_{\mathsf{id}}}$ and samples $\mathsf{selmafe.sk}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}} \leftarrow \mathsf{selsMAFE.KGen}(\mathsf{id}, \mathsf{selmafe.msk}_{\mathsf{id}}, \{\mathsf{selmafe.mpk}_{\mathsf{idx}}\}_{\mathsf{idx}\in[n]}, \mathsf{GID}, (\mathsf{id}, \mathsf{GID}, x_{\mathsf{id}}, \rho_{\mathsf{id}}, \mathsf{ske.ct}^*_{\mathsf{id}}))$ for each $\mathsf{id} \in [n]$.

Set $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}} := \mathsf{selmafe.sk}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}$ and send $\{\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}\}_{\mathsf{id}\in[n]}$ to $\mathcal{A}$.

**Challenge Query.** $\mathcal{A}$ sends $C_0, C_1$. Sample $\mathsf{pmafe.ct} \leftarrow \mathsf{opMAFE.Enc}(\{\mathsf{pmafe.msk}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}, C_b)$ and $\mathsf{selmafe.ct} \leftarrow \mathsf{selsMAFE.Enc}(\{\mathsf{selmafe.mpk}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}, G)$. Send $\mathsf{CT} := (\mathsf{selmafe.ct}, \mathsf{pmafe.ct})$ to $\mathcal{A}$.

**Post-challenge Queries.** This is same as pre-challenge queries.

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

**$\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$.** In this hybrid, we set $\mathsf{ske.ct}^*_{\mathsf{id}}$ to be an SKE ciphertext of the string $\mathsf{pmafe.sk}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}} = \mathsf{opMAFE.KGen}(\mathsf{id}, \mathsf{pmafe.msk}_{\mathsf{id}}, \mathsf{GID}, x_{\mathsf{id}}; \mathsf{PRF}(K, \rho_{\mathsf{id},\mathsf{GID}}))$. The changes are highlighted in red.

**Setup Phase.** $\mathcal{A}$ sends $(n, s, \ell_1, \ldots, \ell_n)$. Define $L = \max_{\mathsf{id}}\{\ell_{\mathsf{id}}\}$. Sample $\mathsf{selmafe.crs} \leftarrow \mathsf{selsMAFE.GSetup}(1^\lambda, 1^n, 1^{s'}, 1^L)$ where $s' = \mathsf{poly}(\lambda, n, L, s)$ is the maximum size of circuit from Figure 4 and Figure 8. Set $\mathsf{CRS} := \mathsf{selmafe.crs}$. Then, sample $(\mathsf{selmafe.mpk}_{\mathsf{id}}, \mathsf{selmafe.msk}_{\mathsf{id}}) \leftarrow \mathsf{selsMAFE.ASetup}(\mathsf{id}, 1^{\ell'_{\mathsf{id}}})$ for each $\mathsf{id} \in [n]$. Set $\mathsf{MPK}_{\mathsf{id}} := \mathsf{selmafe.mpk}_{\mathsf{id}}$ and $\mathsf{MSK}_{\mathsf{id}} := \mathsf{selmafe.msk}_{\mathsf{id}}$.

Now, sample $K \xleftarrow{\$} \{0,1\}^\lambda$ and $\mathsf{pmafe.crs}, \{\mathsf{pmafe.msk_{id}}\}_{\mathsf{id}\in[n]} \leftarrow \mathsf{opMAFE.GSetup}(1^\lambda, 1^n, 1^s, 1^L)$. Construct circuit $G$ from Figure 4. In addition, sample $\mathsf{ske.sk}^* \leftarrow \mathsf{SKE.Setup}(1^\lambda)$ and construct $G'$ from Figure 8.

Send $(\mathsf{CRS}, \{\mathsf{MPK_{id}}\}_{\mathsf{id}\in[n]})$ to $\mathcal{A}$.

**Pre-challenge Queries.** $\mathcal{A}$ sends queries of the form $(\mathsf{GID}, x_1, \ldots, x_n)$.

Now sample $\rho_{\mathsf{id}} \xleftarrow{\$} \{0,1\}^\lambda$, $\mathsf{ske.ct}^*_{\mathsf{id}} \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}^*, \mathsf{opMAFE.KGen}(\mathsf{id}, \mathsf{pmafe.msk_{id}}, \mathsf{GID}, x_{\mathsf{id}}; \mathsf{PRF}(K, \rho_{\mathsf{id}})))$. Sample $\mathsf{selmafe.sk_{id,GID},}_{x_{\mathsf{id}}} \leftarrow \mathsf{selsMAFE.KGen}(\mathsf{id}, \mathsf{selmafe.msk_{id}}, \{\mathsf{selmafe.mpk_{idx}}\}_{\mathsf{idx}\in[n]}, \mathsf{GID}, (\mathsf{id}, \mathsf{GID}, x_{\mathsf{id}}, \rho_{\mathsf{id}}, \mathsf{ske.ct}^*_{\mathsf{id}}))$ for each $\mathsf{id} \in [n]$.

Set $\mathsf{SK_{id,GID},}_{x_{\mathsf{id}}} := \mathsf{selmafe.sk_{id,GID},}_{x_{\mathsf{id}}}$ and send $\{\mathsf{SK_{id,GID},}_{x_{\mathsf{id}}}\}_{\mathsf{id}\in[n]}$ to $\mathcal{A}$.

**Challenge Query.** $\mathcal{A}$ sends $C_0, C_1$. Sample $\mathsf{pmafe.ct} \leftarrow \mathsf{opMAFE.Enc}(\{\mathsf{pmafe.msk_{id}}\}_{\mathsf{id}\in[n]}, C_b)$ and $\mathsf{selmafe.ct} \leftarrow \mathsf{selsMAFE.Enc}(\{\mathsf{selmafe.mpk_{id}}\}_{\mathsf{id}\in[n]}, G)$. Send $\mathsf{CT} := (\mathsf{selmafe.ct}, \mathsf{pmafe.ct})$ to $\mathcal{A}$.

**Post-challenge Queries.** This is same as pre-challenge queries.

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

**$\mathbf{Hyb}_2^{\mathcal{A}}(1^\lambda)$.** In this hybrid, we encrypt $G'$ using selsMAFE. The changes are highlighted in red.

**Setup Phase.** $\mathcal{A}$ sends $(n, s, \ell_1, \ldots, \ell_n)$. Define $L = \max_{\mathsf{id}}\{\ell_{\mathsf{id}}\}$. Sample $\mathsf{selmafe.crs} \leftarrow \mathsf{selsMAFE.GSetup}(1^\lambda, 1^n, 1^{s'}, 1^L)$ where $s' = \mathsf{poly}(\lambda, n, L, s)$ is the maximum size of circuit from Figure 4 and Figure 8. Set $\mathsf{CRS} := \mathsf{selmafe.crs}$. Then, sample $(\mathsf{selmafe.mpk_{id}}, \mathsf{selmafe.msk_{id}}) \leftarrow \mathsf{selsMAFE.ASetup}(\mathsf{id}, 1^{\ell'_{\mathsf{id}}})$ for each $\mathsf{id} \in [n]$. Set $\mathsf{MPK_{id}} := \mathsf{selmafe.mpk_{id}}$ and $\mathsf{MSK_{id}} := \mathsf{selmafe.msk_{id}}$.

Now, sample $K \xleftarrow{\$} \{0,1\}^\lambda$ and $\mathsf{pmafe.crs}, \{\mathsf{pmafe.msk_{id}}\}_{\mathsf{id}\in[n]} \leftarrow \mathsf{opMAFE.GSetup}(1^\lambda, 1^n, 1^s, 1^L)$. Construct circuit $G$ from Figure 4. In addition, sample $\mathsf{ske.sk}^* \leftarrow \mathsf{SKE.Setup}(1^\lambda)$ and construct $G'$ from Figure 8.

Send $(\mathsf{CRS}, \{\mathsf{MPK_{id}}\}_{\mathsf{id}\in[n]})$ to $\mathcal{A}$.

**Pre-challenge Queries.** $\mathcal{A}$ sends queries of the form $(\mathsf{GID}, x_1, \ldots, x_n)$.

Now sample $\rho_{\mathsf{id}} \xleftarrow{\$} \{0,1\}^\lambda$, $\mathsf{ske.ct}^*_{\mathsf{id}} \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}^*, \mathsf{opMAFE.KGen}(\mathsf{id}, \mathsf{pmafe.msk_{id}}, \mathsf{GID}, x_{\mathsf{id}}; \mathsf{PRF}(K, \rho_{\mathsf{id}})))$. Sample $\mathsf{selmafe.sk_{id,GID},}_{x_{\mathsf{id}}} \leftarrow \mathsf{selsMAFE.KGen}(\mathsf{id}, \mathsf{selmafe.msk_{id}}, \{\mathsf{selmafe.mpk_{idx}}\}_{\mathsf{idx}\in[n]}, \mathsf{GID}, (\mathsf{id}, \mathsf{GID}, x_{\mathsf{id}}, \rho_{\mathsf{id}}, \mathsf{ske.ct}^*_{\mathsf{id}}))$ for each $\mathsf{id} \in [n]$.

Set $\mathsf{SK_{id,GID},}_{x_{\mathsf{id}}} := \mathsf{selmafe.sk_{id,GID},}_{x_{\mathsf{id}}}$ and send $\{\mathsf{SK_{id,GID},}_{x_{\mathsf{id}}}\}_{\mathsf{id}\in[n]}$ to $\mathcal{A}$.

**Challenge Query.** $\mathcal{A}$ sends $C_0, C_1$. Sample $\mathsf{pmafe.ct} \leftarrow \mathsf{opMAFE.Enc}(\{\mathsf{pmafe.msk_{id}}\}_{\mathsf{id}\in[n]}, C_b)$ and $\mathsf{selmafe.ct} \leftarrow \mathsf{selsMAFE.Enc}(\{\mathsf{selmafe.mpk_{id}}\}_{\mathsf{id}\in[n]}, G')$. Send $\mathsf{CT} := (\mathsf{selmafe.ct}, \mathsf{pmafe.ct})$ to $\mathcal{A}$.

**Post-challenge Queries.** This is same as pre-challenge queries.

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

**$\mathbf{Hyb}_3^{\mathcal{A}}(1^\lambda)$.** In this hybrid, we sample uniformly random strings for garbling randomness instead of using $\mathsf{PRF}(K, \rho_{\mathsf{id},\mathsf{GID}})$. The changes are highlighted in red.

**Setup Phase.** $\mathcal{A}$ sends $(n, s, \ell_1, \ldots, \ell_n)$. Define $L = \max_{\mathsf{id}}\{\ell_{\mathsf{id}}\}$. Sample $\mathsf{selmafe.crs} \leftarrow \mathsf{selsMAFE.GSetup}(1^\lambda, 1^n, 1^{s'}, 1^L)$ where $s' = \mathsf{poly}(\lambda, n, L, s)$ is the maximum size of circuit from Figure 4 and Figure 8. Set $\mathsf{CRS} := \mathsf{selmafe.crs}$. Then, sample $(\mathsf{selmafe.mpk}_{\mathsf{id}}, \mathsf{selmafe.msk}_{\mathsf{id}}) \leftarrow \mathsf{selsMAFE.ASetup}(\mathsf{id}, 1^{\ell'_{\mathsf{id}}})$ for each $\mathsf{id} \in [n]$. Set $\mathsf{MPK}_{\mathsf{id}} := \mathsf{selmafe.mpk}_{\mathsf{id}}$ and $\mathsf{MSK}_{\mathsf{id}} := \mathsf{selmafe.msk}_{\mathsf{id}}$.

Now, sample $K \xleftarrow{\$} \{0,1\}^\lambda$ and $\mathsf{pmafe.crs}, \{\mathsf{pmafe.msk}_{\mathsf{id}}\}_{\mathsf{id}\in[n]} \leftarrow \mathsf{opMAFE.GSetup}(1^\lambda, 1^n, 1^s, 1^L)$. Construct circuit $G$ from Figure 4. In addition, sample $\mathsf{ske.sk}^* \leftarrow \mathsf{SKE.Setup}(1^\lambda)$ and construct $G'$ from Figure 8.

Send $(\mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]})$ to $\mathcal{A}$.

**Pre-challenge Queries.** $\mathcal{A}$ sends queries of the form $(\mathsf{GID}, x_1, \ldots, x_n)$.

Now sample $R_{\mathsf{id},\mathsf{GID}} \xleftarrow{\$} \{0,1\}^\lambda$, $\mathsf{ske.ct}^*_{\mathsf{id}} \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}^*, \mathsf{opMAFE.KGen}(\mathsf{id}, \mathsf{pmafe.msk}_{\mathsf{id}}, \mathsf{GID}, x_{\mathsf{id}}; R_{\mathsf{id},\mathsf{GID}}))$. Sample $\mathsf{selmafe.sk}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}} \leftarrow \mathsf{selsMAFE.KGen}(\mathsf{id}, \mathsf{selmafe.msk}_{\mathsf{id}}, \{\mathsf{selmafe.mpk}_{\mathsf{idx}}\}_{\mathsf{idx}\in[n]}, \mathsf{GID}, (\mathsf{id}, \mathsf{GID}, x_{\mathsf{id}}, \rho_{\mathsf{id}}, \mathsf{ske.ct}^*_{\mathsf{id}}))$ for each $\mathsf{id} \in [n]$.

Set $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}} := \mathsf{selmafe.sk}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}$ and send $\{\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}\}_{\mathsf{id}\in[n]}$ to $\mathcal{A}$.

**Challenge Query.** $\mathcal{A}$ sends $C_0, C_1$. Sample $\mathsf{pmafe.ct} \leftarrow \mathsf{opMAFE.Enc}(\{\mathsf{pmafe.msk}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}, C_b)$ and $\mathsf{selmafe.ct} \leftarrow \mathsf{selsMAFE.Enc}(\{\mathsf{selmafe.mpk}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}, G')$. Send $\mathsf{CT} := (\mathsf{selmafe.ct}, \mathsf{pmafe.ct})$ to $\mathcal{A}$.

**Post-challenge Queries.** This is same as pre-challenge queries.

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

**$\mathbf{Hyb}_4^{\mathcal{A}}(1^\lambda)$.** In this hybrid, we encrypt $C_{1-b}$ using opMAFE. The changes are highlighted in red.

**Setup Phase.** $\mathcal{A}$ sends $(n, s, \ell_1, \ldots, \ell_n)$. Define $L = \max_{\mathsf{id}}\{\ell_{\mathsf{id}}\}$. Sample $\mathsf{selmafe.crs} \leftarrow \mathsf{selsMAFE.GSetup}(1^\lambda, 1^n, 1^{s'}, 1^L)$ where $s' = \mathsf{poly}(\lambda, n, L, s)$ is the maximum size of circuit from Figure 4 and Figure 8. Set $\mathsf{CRS} := \mathsf{selmafe.crs}$. Then, sample $(\mathsf{selmafe.mpk}_{\mathsf{id}}, \mathsf{selmafe.msk}_{\mathsf{id}}) \leftarrow \mathsf{selsMAFE.ASetup}(\mathsf{id}, 1^{\ell'_{\mathsf{id}}})$ for each $\mathsf{id} \in [n]$. Set $\mathsf{MPK}_{\mathsf{id}} := \mathsf{selmafe.mpk}_{\mathsf{id}}$ and $\mathsf{MSK}_{\mathsf{id}} := \mathsf{selmafe.msk}_{\mathsf{id}}$.

Now, sample $K \xleftarrow{\$} \{0,1\}^\lambda$ and $\mathsf{pmafe.crs}, \{\mathsf{pmafe.msk}_{\mathsf{id}}\}_{\mathsf{id}\in[n]} \leftarrow \mathsf{opMAFE.GSetup}(1^\lambda, 1^n, 1^s, 1^L)$. Construct circuit $G$ from Figure 4. In addition, sample $\mathsf{ske.sk}^* \leftarrow \mathsf{SKE.Setup}(1^\lambda)$ and construct $G'$ from Figure 8.

Send $(\mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]})$ to $\mathcal{A}$.

**Pre-challenge Queries.** $\mathcal{A}$ sends queries of the form $(\mathsf{GID}, x_1, \ldots, x_n)$.

Now sample $R_{\mathsf{id},\mathsf{GID}} \xleftarrow{\$} \{0,1\}^\lambda$, $\mathsf{ske.ct}^*_{\mathsf{id}} \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}^*, \mathsf{opMAFE.KGen}(\mathsf{id}, \mathsf{pmafe.msk}_{\mathsf{id}}, \mathsf{GID}, x_{\mathsf{id}}; R_{\mathsf{id},\mathsf{GID}}))$. Sample $\mathsf{selmafe.sk}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}} \leftarrow \mathsf{selsMAFE.KGen}(\mathsf{id}, \mathsf{selmafe.msk}_{\mathsf{id}}, \{\mathsf{selmafe.mpk}_{\mathsf{idx}}\}_{\mathsf{idx}\in[n]}, \mathsf{GID}, (\mathsf{id}, \mathsf{GID}, x_{\mathsf{id}}, \rho_{\mathsf{id}}, \mathsf{ske.ct}^*_{\mathsf{id}}))$ for each $\mathsf{id} \in [n]$.

Set $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}} := \mathsf{selmafe.sk}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}$ and send $\{\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}\}_{\mathsf{id}\in[n]}$ to $\mathcal{A}$.

**Challenge Query.** $\mathcal{A}$ sends $C_0, C_1$. Sample $\mathsf{pmafe.ct} \leftarrow \mathsf{opMAFE.Enc}(\{\mathsf{pmafe.msk_{id}}\}_{\mathsf{id}\in[n]}, C_{1-b})$ and $\mathsf{selmafe.ct} \leftarrow \mathsf{selsMAFE.Enc}(\{\mathsf{selmafe.mpk_{id}}\}_{\mathsf{id}\in[n]}, G')$. Send $\mathsf{CT} := (\mathsf{selmafe.ct}, \mathsf{pmafe.ct})$ to $\mathcal{A}$.

**Post-challenge Queries.** This is same as pre-challenge queries.

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

**$\mathbf{Hyb}_5^{\mathcal{A}}(1^\lambda)$.** This is real experiment with $\mathsf{Chal}$ with $1-b$.

**Claim G.1.** Assuming the pseudorandom ciphertext security of $\mathsf{SKE}$, $\mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$ are computationally indistinguishable.

*Proof.* Assume that there exists an adversary $\mathcal{A}$ that can distinguish between $\mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$ with non-negligible probability $\epsilon(\lambda)$, 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 $\mathsf{SKE}$ with non-negligible probability. The description of $\mathcal{B}^{\mathcal{O}}$ is as follows.

**Setup Phase.** $\mathcal{A}$ sends $(n, s, \ell_1, \ldots, \ell_n)$ to $\mathsf{Chal}$. Define $L = \max_{\mathsf{id}}\{\ell_{\mathsf{id}}\}$. Sample $\mathsf{selmafe.crs} \leftarrow \mathsf{selsMAFE.GSetup}(1^\lambda, 1^n, 1^{s'}, 1^L)$ where $s' = \mathsf{poly}(\lambda, n, L, s)$ is the maximum size of circuit from Figure 4 and Figure 8. Set $\mathsf{CRS} := \mathsf{selmafe.crs}$. Then, sample $(\mathsf{selmafe.mpk_{id}}, \mathsf{selmafe.msk_{id}}) \leftarrow \mathsf{selsMAFE.ASetup}(\mathsf{id}, 1^{\ell'_{\mathsf{id}}})$ for each $\mathsf{id} \in [n]$. Set $\mathsf{MPK_{id}} := \mathsf{selmafe.mpk_{id}}$ and $\mathsf{MSK_{id}} := \mathsf{selmafe.msk_{id}}$.

Now, sample $K \xleftarrow{\$} \{0,1\}^\lambda$ and $\mathsf{pmafe.crs}, \{\mathsf{pmafe.msk_{id}}\}_{\mathsf{id}\in[n]} \leftarrow \mathsf{opMAFE.GSetup}(1^\lambda, 1^n, 1^s, 1^L)$. Construct circuit $G$ from Figure 4.

Send $(\mathsf{CRS}, \{\mathsf{MPK_{id}}\}_{\mathsf{id}\in[n]})$ to $\mathcal{A}$.

**Pre-challenge Queries.** $\mathcal{A}$ sends queries of the form $(\mathsf{GID}, x_1, \ldots, x_n)$.

Now sample $\rho_{\mathsf{id}} \xleftarrow{\$} \{0,1\}^\lambda$, send $\mathsf{opMAFE.KGen}(\mathsf{id}, \mathsf{pmafe.msk_{id}}, \mathsf{GID}, x_{\mathsf{id}}; \mathsf{PRF}(K, \rho_{\mathsf{id}}))$ to $\mathcal{O}$ to receive $\mathsf{ske.ct}^*_{\mathsf{id}}$. Sample $\mathsf{selmafe.sk_{id,GID},x_{id}} \leftarrow \mathsf{selsMAFE.KGen}(\mathsf{id}, \mathsf{selmafe.msk_{id}}, \{\mathsf{selmafe.mpk_{idx}}\}_{\mathsf{idx}\in[n]}, \mathsf{GID}, (\mathsf{id}, \mathsf{GID}, x_{\mathsf{id}}, \rho_{\mathsf{id}}, \mathsf{ske.ct}^*_{\mathsf{id}}))$ for each $\mathsf{id} \in [n]$.

Set $\mathsf{SK_{id,GID},x_{id}} := \mathsf{selmafe.sk_{id,GID},x_{id}}$ and send $\{\mathsf{SK_{id,GID},x_{id}}\}_{\mathsf{id}\in[n]}$ to $\mathcal{A}$.

**Challenge Query.** $\mathcal{A}$ sends $C_0, C_1$. Sample $\mathsf{pmafe.ct} \leftarrow \mathsf{opMAFE.Enc}(\{\mathsf{pmafe.msk_{id}}\}_{\mathsf{id}\in[n]}, C_b)$ and $\mathsf{selmafe.ct} \leftarrow \mathsf{selsMAFE.Enc}(\{\mathsf{selmafe.mpk_{id}}\}_{\mathsf{id}\in[n]}, G)$. Send $\mathsf{CT} := (\mathsf{selmafe.ct}, \mathsf{pmafe.ct})$ to $\mathcal{A}$.

**Post-challenge Queries.** This is same as pre-challenge queries.

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

As we can see, the running time of $\mathcal{B}$ is polynomial in the running time of $\mathcal{A}$, $\lambda$. If $\mathcal{O}$ generates uniformly random strings, $\mathcal{B}$ behaves like $\mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)$ and if $\mathcal{O}$ generates encryptions of message, $\mathcal{B}$ behaves like $\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$. Hence, $\mathcal{B}$ is a valid adversary that break the security of $\mathsf{SKE}$ with non-negligible probability. Thus, $\mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$ are computationally indistinguishable. □

**Claim G.2.** Assuming the security of $\mathsf{selsMAFE}$, $\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_2^{\mathcal{A}}(1^\lambda)$ are computationally indistinguishable.

*Proof.* Assume that there exists an adversary $\mathcal{A}$ that can distinguish between $\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_2^{\mathcal{A}}(1^\lambda)$ with non-negligible probability $\epsilon(\lambda)$, 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 $\mathsf{selsMAFE}$ with non-negligible probability. The description of $\mathcal{B}^{\mathcal{O}}$ is as follows.

**Setup Phase.** $\mathcal{A}$ sends $(n, s, \ell_1, \ldots, \ell_n)$. Define $L = \max_{\mathsf{id}}\{\ell_{\mathsf{id}}\}$. Now, sample $K \xleftarrow{\$} \{0,1\}^\lambda$ and $\mathsf{pmafe.crs}, \{\mathsf{pmafe.msk_{id}}\}_{\mathsf{id}\in[n]} \leftarrow \mathsf{opMAFE.GSetup}(1^\lambda, 1^n, 1^s, 1^L)$. Construct circuit $G$ from Figure 4. In addition, sample $\mathsf{ske.sk}^* \leftarrow \mathsf{SKE.Setup}(1^\lambda)$ and construct $G'$ from Figure 8.

Send $(n, s', \ell'_1, \ldots, \ell'_n, G, G')$ to $\mathcal{O}$ to receive $\mathsf{selmafe.crs}, \{\mathsf{selmafe.mpk_{id}}\}_{\mathsf{id}\in[n]}, \mathsf{selmafe.ct}$. Set $\mathsf{CRS} := \mathsf{selmafe.crs}$, $\mathsf{MPK_{id}} := \mathsf{selmafe.mpk_{id}}$ for each $\mathsf{id} \in [n]$.

Send $(\mathsf{CRS}, \{\mathsf{MPK_{id}}\}_{\mathsf{id}\in[n]})$ to $\mathcal{A}$.

**Pre-challenge Queries.** $\mathcal{A}$ sends queries of the form $(\mathsf{GID}, x_1, \ldots, x_n)$.

Now sample $\rho_{\mathsf{id}} \xleftarrow{\$} \{0,1\}^\lambda$, $\mathsf{ske.ct}^*_{\mathsf{id}} \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}^*, \mathsf{opMAFE.KGen}(\mathsf{id}, \mathsf{pmafe.msk_{id}}, \mathsf{GID}, x_{\mathsf{id}}; \mathsf{PRF}(K, \rho_{\mathsf{id}})))$ for each $\mathsf{id} \in [n]$. Send $((\mathsf{id}, \mathsf{GID}, x_{\mathsf{id}}, \rho_{\mathsf{id}}, \mathsf{ske.ct}^*_{\mathsf{id}}))_{\mathsf{id}\in[n]}$ to $\mathcal{O}$ to receive $\{\mathsf{selmafe.sk_{id,GID},}x_{\mathsf{id}}\}_{\mathsf{id}\in[n]}$.

Set $\mathsf{SK_{id,GID},}x_{\mathsf{id}} := \mathsf{selmafe.sk_{id,GID},}x_{\mathsf{id}}$ and send $\{\mathsf{SK_{id,GID},}x_{\mathsf{id}}\}_{\mathsf{id}\in[n]}$ to $\mathcal{A}$.

**Challenge Query.** $\mathcal{A}$ sends $C_0, C_1$. Sample $\mathsf{pmafe.ct} \leftarrow \mathsf{opMAFE.Enc}(\{\mathsf{pmafe.msk_{id}}\}_{\mathsf{id}\in[n]}, C_b)$. Send $\mathsf{CT} := (\mathsf{selmafe.ct}, \mathsf{pmafe.ct})$ to $\mathcal{A}$.

**Post-challenge Queries.** This is same as pre-challenge queries.

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

As we can see, the running time of $\mathcal{B}$ is polynomial in the running time of $\mathcal{A}$, $\lambda$. If $\mathcal{O}$ encrypts $G$, $\mathcal{B}$ behaves like $\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$ and if $\mathcal{O}$ encrypts $G'$, $\mathcal{B}$ behaves like $\mathbf{Hyb}_2^{\mathcal{A}}(1^\lambda)$. Hence, $\mathcal{B}$ is a valid adversary that break the security of $\mathsf{selsMAFE}$ with non-negligible probability. Thus, $\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_2^{\mathcal{A}}(1^\lambda)$ are computationally indistinguishable. □

**Claim G.3.** Assuming the pseudorandomness property of $\mathsf{PRF}$, $\mathbf{Hyb}_2^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_3^{\mathcal{A}}(1^\lambda)$ are computationally indistinguishable.

*Proof.* Assume that there exists an adversary $\mathcal{A}$ that can distinguish between $\mathbf{Hyb}_2^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_3^{\mathcal{A}}(1^\lambda)$ with non-negligible probability $\epsilon(\lambda)$, 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 security of $\mathsf{PRF}$ with non-negligible probability. The description of $\mathcal{B}^{\mathcal{O}}$ is as follows.

**Setup Phase.** $\mathcal{A}$ sends $(n, s, \ell_1, \ldots, \ell_n)$. Define $L = \max_{\mathsf{id}}\{\ell_{\mathsf{id}}\}$. Sample $\mathsf{selmafe.crs} \leftarrow \mathsf{selsMAFE.GSetup}(1^\lambda, 1^n, 1^{s'}, 1^L)$ where $s' = \mathsf{poly}(\lambda, n, L, s)$ is the maximum size of circuit from Figure 4 and Figure 8. Set $\mathsf{CRS} := \mathsf{selmafe.crs}$. Then, sample $(\mathsf{selmafe.mpk_{id}}, \mathsf{selmafe.msk_{id}}) \leftarrow \mathsf{selsMAFE.ASetup}(\mathsf{id}, 1^{\ell'_{\mathsf{id}}})$ for each $\mathsf{id} \in [n]$. Set $\mathsf{MPK_{id}} := \mathsf{selmafe.mpk_{id}}$ and $\mathsf{MSK_{id}} := \mathsf{selmafe.msk_{id}}$.

Now, sample $\mathsf{pmafe.crs}, \{\mathsf{pmafe.msk_{id}}\}_{\mathsf{id}\in[n]} \leftarrow \mathsf{opMAFE.GSetup}(1^\lambda, 1^n, 1^s, 1^L)$. In addition, sample $\mathsf{ske.sk}^* \leftarrow \mathsf{SKE.Setup}(1^\lambda)$ and construct $G'$ from Figure 8.

Send $(\mathsf{CRS}, \{\mathsf{MPK_{id}}\}_{\mathsf{id}\in[n]})$ to $\mathcal{A}$.

**Pre-challenge Queries.** $\mathcal{A}$ sends queries of the form $(\mathsf{GID}, x_1, \ldots, x_n)$.

Now sample $\rho_{\mathsf{id}} \xleftarrow{\$} \{0,1\}^\lambda$ and send it to $\mathcal{O}$ to receive $R_{\mathsf{id,GID}}$, $\mathsf{ske.ct}^*_{\mathsf{id}} \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}^*, \mathsf{opMAFE.KGen}(\mathsf{id}, \mathsf{pmafe.msk_{id}}, \mathsf{GID}, x_{\mathsf{id}}; R_{\mathsf{id,GID}}))$. Sample $\mathsf{selmafe.sk_{id,GID},x_{id}} \leftarrow \mathsf{selsMAFE.KGen}(\mathsf{id}, \mathsf{selmafe.msk_{id}}, \{\mathsf{selmafe.mpk_{idx}}\}_{\mathsf{idx}\in[n]}, \mathsf{GID}, (\mathsf{id}, \mathsf{GID}, x_{\mathsf{id}}, \rho_{\mathsf{id}}, \mathsf{ske.ct}^*_{\mathsf{id}}))$ for each $\mathsf{id} \in [n]$.

Set $\mathsf{SK_{id,GID},x_{id}} := \mathsf{selmafe.sk_{id,GID},x_{id}}$ and send $\{\mathsf{SK_{id,GID},x_{id}}\}_{\mathsf{id}\in[n]}$ to $\mathcal{A}$.

**Challenge Query.** $\mathcal{A}$ sends $C_0, C_1$. Sample $\mathsf{pmafe.ct} \leftarrow \mathsf{opMAFE.Enc}(\{\mathsf{pmafe.msk_{id}}\}_{\mathsf{id}\in[n]}, C_b)$ and $\mathsf{selmafe.ct} \leftarrow \mathsf{selsMAFE.Enc}(\{\mathsf{selmafe.mpk_{id}}\}_{\mathsf{id}\in[n]}, G')$. Send $\mathsf{CT} := (\mathsf{selmafe.ct}, \mathsf{pmafe.ct})$ to $\mathcal{A}$.

**Post-challenge Queries.** This is same as pre-challenge queries.

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

As we can see, the running time of $\mathcal{B}$ is polynomial in the running time of $\mathcal{A}$, $\lambda$. If $\mathcal{O}$ is $\mathsf{PRF}(K, \cdot)$ for some $K \xleftarrow{\$} \{0,1\}^\lambda$, $\mathcal{B}$ behaves like $\mathbf{Hyb}_2^{\mathcal{A}}(1^\lambda)$ and if $\mathcal{O}$ generates uniformly random strings $\mathcal{B}$ behaves like $\mathbf{Hyb}_3^{\mathcal{A}}(1^\lambda)$. Hence, $\mathcal{B}$ is a valid adversary that break the security of PRF with non-negligible probability. Thus, $\mathbf{Hyb}_2^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_3^{\mathcal{A}}(1^\lambda)$ are computationally indistinguishable. □

**Claim G.4.** Assuming the security of opMAFE, $\mathbf{Hyb}_3^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_4^{\mathcal{A}}(1^\lambda)$ are computationally indistinguishable.

*Proof.* Assume that there exists an adversary $\mathcal{A}$ that can distinguish between $\mathbf{Hyb}_3^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_4^{\mathcal{A}}(1^\lambda)$ with non-negligible probability $\epsilon(\lambda)$, i.e.,

$$\left|\Pr\left[1 \leftarrow \mathbf{Hyb}_3^{\mathcal{A}}(1^\lambda)\right] - \Pr\left[1 \leftarrow \mathbf{Hyb}_4^{\mathcal{A}}(1^\lambda)\right]\right| > \epsilon(\lambda)$$

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

**Setup Phase.** $\mathcal{A}$ sends $(n, s, \ell_1, \ldots, \ell_n)$. Define $L = \max_{\mathsf{id}}\{\ell_{\mathsf{id}}\}$. Sample $\mathsf{selmafe.crs} \leftarrow \mathsf{selsMAFE.GSetup}(1^\lambda, 1^n, 1^{s'}, 1^L)$ where $s' = \mathsf{poly}(\lambda, n, L, s)$ is the maximum size of circuit from Figure 4 and Figure 8. Set $\mathsf{CRS} := \mathsf{selmafe.crs}$. Then, sample $(\mathsf{selmafe.mpk_{id}}, \mathsf{selmafe.msk_{id}}) \leftarrow \mathsf{selsMAFE.ASetup}(\mathsf{id}, 1^{\ell'_{\mathsf{id}}})$ for each $\mathsf{id} \in [n]$. Set $\mathsf{MPK_{id}} := \mathsf{selmafe.mpk_{id}}$ and $\mathsf{MSK_{id}} := \mathsf{selmafe.msk_{id}}$.

Now, send $n, s, \ell_1, \ldots, \ell_n$ to $\mathcal{O}$ to receive $\mathsf{pmafe.crs}$. In addition, sample $\mathsf{ske.sk}^* \leftarrow \mathsf{SKE.Setup}(1^\lambda)$ and construct $G'$ from Figure 8.

Send $(\mathsf{CRS}, \{\mathsf{MPK_{id}}\}_{\mathsf{id}\in[n]})$ to $\mathcal{A}$.

**Pre-challenge Queries.** $\mathcal{A}$ sends queries of the form $(\mathsf{GID}, x_1, \ldots, x_n)$.

Now send $(\mathsf{GID}, x_1, \ldots, x_n)$ to $\mathcal{O}$ to receive $\{\mathsf{pmafe.sk}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}\}_{\mathsf{id}\in[n]}$ sample $\mathsf{ske.ct}^*_{\mathsf{id}} \leftarrow \mathsf{SKE.Enc}($ $\mathsf{ske.sk}^*, \mathsf{pmafe.sk}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}})$. Sample $\mathsf{selmafe.sk}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}} \leftarrow \mathsf{selsMAFE.KGen}(\mathsf{id}, \mathsf{selmafe.msk}_{\mathsf{id}}, \{\mathsf{selmafe.mpk}_{\mathsf{idx}}$ $\}_{\mathsf{idx}\in[n]}, \mathsf{GID}, (\mathsf{id}, \mathsf{GID}, x_{\mathsf{id}}, \rho_{\mathsf{id}}, \mathsf{ske.ct}^*_{\mathsf{id}}))$ for each $\mathsf{id} \in [n]$.

Set $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}} := \mathsf{selmafe.sk}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}$ and send $\{\mathsf{SK}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}}\}_{\mathsf{id}\in[n]}$ to $\mathcal{A}$.

**Challenge Query.** $\mathcal{A}$ sends $C_0, C_1$. Send $(C_b, C_{1-b})$ to $\mathcal{O}$ to receive $\mathsf{pmafe.ct}$ and sample $\mathsf{selmafe.ct} \leftarrow$ $\mathsf{selsMAFE.Enc}(\{\mathsf{selmafe.mpk}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}, G')$. Send $\mathsf{CT} := (\mathsf{selmafe.ct}, \mathsf{pmafe.ct})$ to $\mathcal{A}$.

**Post-challenge Queries.** This is same as pre-challenge queries.

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

As we can see, the running time of $\mathcal{B}$ is polynomial in the running time of $\mathcal{A}$, $\lambda$. If $\mathcal{O}$ encrypts $C_b$ $\mathcal{B}$ behaves like $\mathbf{Hyb}_3^{\mathcal{A}}(1^\lambda)$ and if $\mathcal{O}$ encrypts $C_{1-b}$, $\mathcal{B}$ behaves like $\mathbf{Hyb}_4^{\mathcal{A}}(1^\lambda)$. Hence, $\mathcal{B}$ is a valid adversary that break the security of $\mathsf{opMAFE}$ with non-negligible probability. Thus, $\mathbf{Hyb}_3^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_4^{\mathcal{A}}(1^\lambda)$ are computationally indistinguishable. □

**Claim G.5.** $\mathbf{Hyb}_4^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_5^{\mathcal{A}}(1^\lambda)$ are computationally indistinguishable.

*Proof.* The proof of this claim is similar to proofs of Claim G.1, Claim G.2, Claim G.3, and Claim G.4. □

# H Vector Non-Committing Encryption from IBE

In this section, we construct a VNCE scheme (Definition 6.9) from any IBE scheme (Definition H.3). The main idea is that each party encrypts its input under each other party's NCE instantiation and also reveals keys for $\mathsf{GID}$. If there are $n$ parties interacting and each party consists of $n$ independent $\mathsf{NCE}$ instantiations, we can form a $n \times n$ matrix of $\mathsf{NCE}$ ciphertexts where each party reveals its own column of secret keys. We can use the final column to be revealed to equivocate. We formally implement this idea below.

## H.1 Preliminaries

## H.2 Non-committing Encryption

**Syntax.** A non-committing encryption scheme ($\mathsf{NCE}$) [CFGN96] is a public-key encryption scheme with an additional property that we can later reveal a ciphertext to be the encryption of any message by tailoring the secret key for the scheme accordingly. A reusable non-committing encryption scheme $\mathsf{NCE}$ for $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda\in\mathbb{N}}$ consists of the following polynomial time algorithms.

$\mathsf{Setup}(1^\lambda, 1^\ell) \to (\mathsf{MPK}, \mathsf{MSK})$**.** The probabilistic setup algorithm takes as input security parameter $\lambda$ and length of the messages $\ell$, outputs master public-secret keys $(\mathsf{MPK}, \mathsf{MSK})$.

$\mathsf{KGen}(\mathsf{MSK}, \mathsf{GID}) \to \mathsf{SK}$**.** The probabilistic key generation algorithm takes as input the master secret key $\mathsf{MSK}$, a $\mathsf{GID} \in \mathcal{GID}_\lambda$, and outputs a secret key $\mathsf{SK}$.

$\mathsf{Enc}(\mathsf{MPK}, \mathsf{GID}, m) \to \mathsf{CT}$. The probabilistic encryption algorithm takes as input master public key $\mathsf{MPK}$, a $\mathsf{GID} \in \mathcal{GID}_\lambda$, a message $m \in \{0,1\}^\ell$, and outputs the corresponding ciphertext $\mathsf{CT}$.

$\mathsf{Dec}(\mathsf{SK}, \mathsf{CT}) \to m'$. The decryption algorithm takes as input the secret key $\mathsf{SK}$ and a ciphertext $\mathsf{CT}$, outputs $m'$.

$\mathsf{Fake}(\mathsf{MPK}, \mathsf{GID}) \to (\widetilde{\mathsf{CT}}, \mathsf{aux})$. The probabilistic faking algorithm takes as input the master public key $\mathsf{MPK}$, a $\mathsf{GID} \in \mathcal{GID}_\lambda$, and outputs a fake ciphertext $\widetilde{\mathsf{CT}}$ and auxiliary information $\mathsf{aux}$.

$\mathsf{Reveal}(\mathsf{MSK}, \mathsf{GID}, \mathsf{aux}, m) \to \widetilde{\mathsf{SK}}$. The revealing algorithm takes as input the master secret key $\mathsf{MSK}$, a $\mathsf{GID} \in \mathcal{GID}_\lambda$, auxiliary information $\mathsf{aux}$, message $m \in \{0,1\}^\ell$, and outputs secret key $\widetilde{\mathsf{SK}}$.

**Definition H.1.** An NCE scheme $\mathsf{NCE} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Fake}, \mathsf{Reveal})$ is said to be a reusable non-committing encryption scheme for $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda\in\mathbb{N}}$ if it satisfies the following properties.

**Correctness.** For any $\lambda \in \mathbb{N}, \ell = \ell(\lambda), \mathsf{GID} \in \mathcal{GID}_\lambda$, any $m \in \{0,1\}^\ell$,

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

**Security.** For any PPT admissible adversary $\mathcal{A}$, there exists a stateful PPT simulator $\mathsf{Sim}$ such that,

$$\left\{\mathcal{A}^{\mathcal{O}(\cdot,\cdot)}(\mathsf{MPK}) : \begin{array}{l}\ell \leftarrow \mathcal{A}(1^\lambda),\\ (\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)\end{array}\right\}$$
$$\approx_c$$
$$\left\{\mathcal{A}^{\mathsf{Sim}(\cdot,\mathcal{V})}(\mathsf{MPK}) : \begin{array}{l}\ell \leftarrow \mathcal{A}(1^\lambda), \mathcal{V} := \emptyset,\\ \mathsf{MPK} \leftarrow \mathsf{Sim}(1^\lambda, 1^\ell),\end{array}\right\}$$

A stateful PPT machine $\mathcal{A}$ is said to be admissible if it makes two kinds of queries — encryption query $(\mathsf{Enc}, (\mathsf{GID}, m))$ to receive $\mathsf{CT}$, secret key query $(\mathsf{KGen}, \mathsf{GID})$ to receive $\mathsf{SK}$, and makes at most one encryption and secret key query per $\mathsf{GID}$. $\mathcal{O}$ uses $\mathsf{Enc}, \mathsf{KGen}$ to respond to $\mathcal{A}$'s queries. $\mathcal{V} = \{(\mathsf{GID}, m) :$ for each $\mathsf{GID}$ queried by $\mathcal{A}\}$ is updated at each query.

$\mathsf{Sim}$ on the other hand uses $\mathsf{Enc}, \mathsf{KGen}$ if $\mathsf{KGen}$ query for $\mathsf{GID}$ comes before $\mathsf{Enc}$ query and $\mathsf{Fake}, \mathsf{Reveal}$ otherwise. While using $\mathsf{Enc}$ in the former scenario or $\mathsf{Reveal}$ in the latter, $\mathsf{Sim}$ queries $\mathcal{V}$ to receive $m$, message for this $\mathsf{GID}$.

**Remark H.2.** An NCE scheme (Definition H.1) can be generically obtained by replacing PKE with IBE (Definition H.3) in the non-committing encryption constructions of [GVW12, HKM+24].

### H.3 Identity-Based Encryption

**Syntax.** An identity-based encryption scheme for identity space $\mathcal{I} = \{\mathcal{I}_\lambda\}_{\lambda\in\mathbb{N}}$ and message space $\{0,1\}^*$[17] consists of the following polynomial time algorithms.

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

$\mathsf{Setup}(1^\lambda, 1^z) \to (\mathsf{MPK}, \mathsf{MSK})$**.** The randomized setup algorithm takes as input security parameter $\lambda$, identity length $z = z(\lambda)$, and outputs master public-secret key pair $\mathsf{MPK}, \mathsf{MSK}$.

$\mathsf{KGen}(\mathsf{MSK}, \mathsf{ibeID}) \to \mathsf{SK}_{\mathsf{ibeID}}$**.** The possibly randomized key generation algorithm takes as input master secret key $\mathsf{MSK}$, identity $\mathsf{ibeID} \in \mathcal{I}_\lambda$, $|\mathsf{ibeID}| = z$, and outputs secret key $\mathsf{SK}_{\mathsf{ibeID}}$.

$\mathsf{Enc}(\mathsf{MPK}, \mathsf{ibeID}, m) \to \mathsf{CT}$**.** The randomized encryption algorithm takes as input master public key $\mathsf{MPK}$, identity $\mathsf{ibeID}$, message $m \in \{0,1\}^\lambda$, and outputs ciphertext $\mathsf{CT}$.

$\mathsf{Dec}(\mathsf{SK}_{\mathsf{ibeID}}, \mathsf{CT}) \to m'$**.** The deterministic decryption algorithm takes as input secret key for $\mathsf{ibeID}$, $\mathsf{SK}_{\mathsf{ibeID}}$, ciphertext $\mathsf{CT}$, and outputs message $m'$.

**Definition H.3** (IBE)**.** An IBE scheme $(\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ is said to be an IBE scheme 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 $m \in \{0,1\}^*$,

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

**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 H.4** (Multi-challenge security)**.** Although Definition H.3 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.

## H.4 Construction

**Notation.** For ease of exposition, we introduce some notation for this construction. We denote matrices by bold uppercase letters $\mathbf{M}$, etc. We denote the $i$-th row, $j$-th column value of matrix $\mathbf{M}$ by $M_{i,j}$. It would easier to understand the construction if we consider the components of NCE scheme with sub- and superscripts in this construction as follows: superscript values are column numbers and subscript values are row numbers.

**Construction H.5.** We provide the construction of a vector NCE scheme (Definition 6.9) for $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda\in\mathbb{N}}$ and $n$ parties using reusable NCE scheme for $\mathcal{GID} = \{\mathcal{GID}_\lambda\}_{\lambda\in\mathbb{N}}$ as follows.

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

$\underline{\mathsf{ASetup}(\mathsf{id})}$**.** Sample $n$ instantiations of $\mathsf{NCE}$, $(\mathsf{nce.mpk}_i, \mathsf{nce.msk}_i) \leftarrow \mathsf{NCE.Setup}(1^\lambda, 1^L)$ for $i \in [n]$. Set $\mathsf{MPK} := (\mathsf{nce.mpk}_i)_{i\in[n]}$, $\mathsf{MSK} := (\mathsf{nce.msk}_i)_{i\in[n]}$ and output $(\mathsf{MPK}, \mathsf{MSK})$.

$\underline{\mathsf{Enc}(\mathsf{id}, \mathsf{MSK}_{\mathsf{id}}, \{\mathsf{MPK}_{\mathsf{idx}}\}_{\mathsf{idx}\in[n]}, \mathsf{GID}, \mu)}$**.** Parse $\mathsf{MSK}_{\mathsf{id}}$ as $(\mathsf{nce.msk}_i^{(\mathsf{id})})_{i\in[n]}$ and $\mathsf{MPK}_{\mathsf{idx}}$ as $(\mathsf{nce.mpk}_i^{(\mathsf{idx})})_{i\in[n]}$ for each $\mathsf{idx} \in [n]$.

- Sample $R_1, \ldots, R_n \leftarrow \{0,1\}^L$. Set $\widetilde{R} = (\bigoplus_{j\in[n]} R_j) \oplus \mu$.
- Sample for each $j \in [n]$, $\mathsf{nce.ct}_{\mathsf{id},\mathsf{GID}}^{(j)} \leftarrow \mathsf{NCE.Enc}(\mathsf{nce.mpk}_{\mathsf{id}}^{(j)}, \mathsf{GID}, R_j)$.
- Sample for each $i \in [n]$, $\mathsf{nce.sk}_i^{(\mathsf{id})} \leftarrow \mathsf{NCE.KGen}(\mathsf{nce.msk}_i^{(\mathsf{id})}, \mathsf{GID})$.

Output $\mathsf{ct}_{\mathsf{id},\mathsf{GID}} := \left((\mathsf{nce.sk}_i^{(\mathsf{id})})_{i\in[n]}, (\mathsf{nce.ct}_{\mathsf{id}}^{(j)})_{j\in[n]}, \widetilde{R}\right)$.

$\underline{\mathsf{Dec}(\{\mathsf{ct}_{\mathsf{id},\mathsf{GID}}\}_{\mathsf{id}\in[n]})}$**.** Parse each $\mathsf{ct}_{\mathsf{id},\mathsf{GID}}$ as $\left((\mathsf{nce.sk}_i^{(\mathsf{id})})_{i\in[n]}, (\mathsf{nce.ct}_{\mathsf{id}}^{(j)})_{j\in[n]}, \widetilde{R}_{\mathsf{id}}\right)$.

- Place each $\mathsf{nce.ct}_{\mathsf{id}}^{(j)}$ as the $(\mathsf{id}, j)$-th entry in the $n \times n$ matrix $\mathbf{C}$ for each $\mathsf{id} \in [n], j \in [n]$.
- Place $\mathsf{nce.sk}_i^{(\mathsf{id})}$ in the $(i, \mathsf{id})$-th entry in the $n \times n$ matrix $\mathbf{K}$ for each $i \in [n], \mathsf{id} \in [n]$.
- Perform component-wise decryption of $\mathbf{C}$ and $\mathbf{K}$, i.e., use the $(i,j)$-th entry of $\mathbf{K}$ to decrypt the $(i,j)$-th entry of $\mathbf{C}$ for each $i, j \in [n]$ and place the resultant value in the $(i,j)$-th position, of $n \times n$ matrix $\mathbf{P}$.
- For each $\mathsf{id} \in [n]$, let $\rho_{\mathsf{id}} = \bigoplus_{j\in[n]} P_{\mathsf{id},j}$. Compute $\mu'_{\mathsf{id}} = \widetilde{R}_{\mathsf{id}} \oplus \rho_{\mathsf{id}}$.

Output $(\mu'_1, \ldots, \mu'_n)$.

**Correctness.** Note that for each $\mathsf{id} \in [n], i \in [n]$, the $\mathsf{id}$-th authority samples keys for the $(i, \mathsf{id})$-th entries for the $n \times n$ matrices $\mathbf{C}, \mathbf{K}$ we form in $\mathsf{Dec}$. That is, $\mathsf{id}$-th column corresponds to $\mathsf{id}$-th authority. Thus, all $(i,j)$-th entries of matrices $\mathbf{C}, \mathbf{K}$ are generated corresponding to the keys $(\mathsf{nce.mpk}_i^{(j)}, \mathsf{nce.msk}_i^{(j)})$ and w.r.to $\mathsf{GID}$. Hence, $P_{\mathsf{id},j}$ is the value $R_j$ generated by the $\mathsf{id}$-th authority by correctness of $\mathsf{NCE}$. Thus, $\rho_{\mathsf{id}} = \bigoplus_{j\in[n]} R_j$. This in turn implies that $(\mu'_1, \ldots, \mu'_n) = (\mu_1, \ldots, \mu_n)$.

**Lemma H.6.** If $\mathsf{NCE}$ is a reusable non-committing encryption scheme for $\mathcal{GID}$ (Definition H.1), then Construction H.5 is a vector non-committing encryption scheme for $\mathcal{GID}$.

*Proof.* We prove this lemma using the following hybrids and claims.

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

**Setup Phase.** $\mathcal{A}$ sends $n, L$. For each $\mathsf{id} \in [n], i \in [n]$, sample $(\mathsf{nce.mpk}_i^{(\mathsf{id})}, \mathsf{nce.msk}_i^{(\mathsf{id})}) \leftarrow \mathsf{NCE.Setup}(1^\lambda, 1^L)$. Set $\mathsf{CRS} := (\lambda, n, L), \mathsf{MPK}_{\mathsf{id}} = (\mathsf{nce.mpk}_i^{(\mathsf{id})})_{i\in[n]}, \mathsf{MSK}_{\mathsf{id}} = (\mathsf{nce.msk}_i^{(\mathsf{id})})_{i\in[n]}$. Send $\{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}$ to $\mathcal{A}$.

**Query Phase.** $\mathcal{A}$ sends $(\mathsf{id}, \mathsf{GID}, x) \in [n] \times \mathcal{GID}_\lambda \times \{0,1\}^L$.

- Sample $R_1, \ldots, R_n \leftarrow \{0,1\}^L$. Set $\widetilde{R} = (\bigoplus_{j\in[n]} R_j) \oplus x$.

- Sample for each $j \in [n]$, $\mathsf{nce.ct}_{\mathsf{id}}^{(j)} \leftarrow \mathsf{NCE.Enc}(\mathsf{nce.mpk}_{\mathsf{id}}^{(j)}, \mathsf{GID}, R_j)$.
- Sample for each $i \in [n]$, $\mathsf{nce.sk}_i^{(\mathsf{id})} \leftarrow \mathsf{NCE.KGen}(\mathsf{nce.msk}_i^{(\mathsf{id})}, \mathsf{GID})$.

Send $\mathsf{ct}_{\mathsf{id},\mathsf{GID}} := \left((\mathsf{nce.sk}_i^{(\mathsf{id})})_{i\in[n]}, (\mathsf{nce.ct}_{\mathsf{id}}^{(j)})_{j\in[n]}, \widetilde{R}\right)$ to $\mathcal{A}$.

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

**$\mathbf{Hyb}^{\mathcal{A}}_{1,\iota,\kappa}(1^\lambda)$ for $\iota \in [n], \kappa \in [n+1]$.** In this hybrid, we will simulate the first $\kappa - 1$ instantiations of NCE by the first $\iota$ authorities. Changes are highlighted in red.

**Setup Phase.** $\mathcal{A}$ sends $n, L$. For each $\mathsf{id} \in [n], i \in [n]$, if $(\mathsf{id} > \iota)$ or $(\mathsf{id} \leq \iota, i \geq \kappa)$, sample $(\mathsf{nce.mpk}_i^{(\mathsf{id})}, \mathsf{nce.msk}_i^{(\mathsf{id})}) \leftarrow \mathsf{NCE.Setup}(1^\lambda, 1^L)$. Otherwise, sample $\mathsf{nce.mpk}_i^{(\mathsf{id})} \leftarrow \mathsf{NCE.Sim}_i^{(\mathsf{id})}(1^\lambda, 1^L)$. Set $\mathsf{CRS} := (\lambda, n, L), \mathsf{MPK}_{\mathsf{id}} = (\mathsf{nce.mpk}_i^{(\mathsf{id})})_{i\in[n]}$. Send $\{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}$ to $\mathcal{A}$.

Start dictionaries $\mathcal{V}_i^{(j)}$ for each $j \in [n], i \in [n]$.

**Query Phase.** $\mathcal{A}$ sends $(\mathsf{id}, \mathsf{GID}, x) \in [n] \times \mathcal{GID}_\lambda \times \{0,1\}^L$.

- Sample $R_1, \ldots, R_n \leftarrow \{0,1\}^L$. Set $\widetilde{R} = (\bigoplus_{j\in[n]} R_j) \oplus x$. Add $(\mathsf{GID}, R_j)$ to $\mathcal{V}_{\mathsf{id}}^{(j)}$.
- Sample for each $j \in [n]$ if $j > \iota$ or $(j \leq \iota, \mathsf{id} \geq \kappa)$, $\mathsf{nce.ct}_{\mathsf{id}}^{(j)} \leftarrow \mathsf{NCE.Enc}(\mathsf{nce.mpk}_{\mathsf{id}}^{(j)}, \mathsf{GID}, R_j)$. Otherwise, $\mathsf{nce.ct}_{\mathsf{id}}^{(j)} \leftarrow \mathsf{NCE.Sim}_{\mathsf{id}}^{(j)}(\mathsf{Enc}, \mathsf{GID})$. Any query made by Sim is answered using dictionaries.
- Sample for each $i \in [n]$, if $\mathsf{id} > \iota$ or $(\mathsf{id} \leq \iota, i \geq \kappa)$, $\mathsf{nce.sk}_i^{(\mathsf{id})} \leftarrow \mathsf{NCE.KGen}(\mathsf{nce.msk}_i^{(\mathsf{id})}, \mathsf{GID})$. Otherwise, $\mathsf{nce.sk}_i^{(\mathsf{id})} \leftarrow \mathsf{NCE.Sim}_i^{(\mathsf{id})}(\mathsf{KGen}, \mathsf{GID})$. Any query made by Sim is answered using dictionaries.

Send $\mathsf{ct}_{\mathsf{id},\mathsf{GID}} := \left((\mathsf{nce.sk}_i^{(\mathsf{id})})_{i\in[n]}, (\mathsf{nce.ct}_{\mathsf{id}}^{(j)})_{j\in[n]}, \widetilde{R}\right)$ to $\mathcal{A}$.

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

**$\mathbf{Hyb}_2^{\mathcal{A}}(1^\lambda)$.** In this hybrid, all NCE instantiations are simulated. This is same as $\mathbf{Hyb}^{\mathcal{A}}_{1,n,n+1}(1^\lambda)$. We write it explicitly for clarity.

**Setup Phase.** $\mathcal{A}$ sends $n, L$. For each $\mathsf{id} \in [n], i \in [n]$, sample $\mathsf{nce.mpk}_i^{(\mathsf{id})} \leftarrow \mathsf{NCE.Sim}_i^{(\mathsf{id})}(1^\lambda, 1^L)$. Set $\mathsf{CRS} := (\lambda, n, L), \mathsf{MPK}_{\mathsf{id}} = (\mathsf{nce.mpk}_i^{(\mathsf{id})})_{i\in[n]}$. Send $\{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}$ to $\mathcal{A}$.

Start dictionaries $\mathcal{V}_i^{(j)}$ for each $j \in [n], i \in [n]$.

**Query Phase.** $\mathcal{A}$ sends $(\mathsf{id}, \mathsf{GID}, x) \in [n] \times \mathcal{GID}_\lambda \times \{0,1\}^L$.

- Sample $R_1, \ldots, R_n \leftarrow \{0,1\}^L$. Set $\widetilde{R} = (\bigoplus_{j\in[n]} R_j) \oplus x$. Add $(\mathsf{GID}, R_j)$ to $\mathcal{V}_{\mathsf{id}}^{(j)}$.
- Sample for each $j \in [n]$, $\mathsf{nce.ct}_{\mathsf{id}}^{(j)} \leftarrow \mathsf{NCE.Sim}_{\mathsf{id}}^{(j)}(\mathsf{Enc}, \mathsf{GID})$. Any query made by Sim is answered using dictionaries.
- Sample for each $i \in [n]$, $\mathsf{nce.sk}_i^{(\mathsf{id})} \leftarrow \mathsf{NCE.Sim}_i^{(\mathsf{id})}(\mathsf{KGen}, \mathsf{GID})$. Any query made by Sim is answered using dictionaries.

Send $\mathsf{ct}_{\mathsf{id},\mathsf{GID}} := \left((\mathsf{nce.sk}_i^{(\mathsf{id})})_{i\in[n]}, (\mathsf{nce.ct}_{\mathsf{id}}^{(j)})_{j\in[n]}, \widetilde{R}\right)$ to $\mathcal{A}$.

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

**$\mathbf{Hyb}_3^{\mathcal{A}}(1^\lambda)$.** In this hybrid, for each $\mathsf{GID}$ query, we sample a matrix $\mathbf{R}^{\mathsf{GID}} \xleftarrow{\$} (\{0,1\}^L)^{n\times n}$ and use the values in this matrix for responding to queries corresponding to $\mathsf{GID}$. The changes are highlighted in red.

**Setup Phase.** $\mathcal{A}$ sends $n, L$. For each $\mathsf{id} \in [n], i \in [n]$, sample $\mathsf{nce.mpk}_i^{(\mathsf{id})} \leftarrow \mathsf{NCE.Sim}_i^{(\mathsf{id})}(1^\lambda, 1^L)$. Set $\mathsf{CRS} := (\lambda, n, L), \mathsf{MPK}_{\mathsf{id}} = (\mathsf{nce.mpk}_i^{(\mathsf{id})})_{i\in[n]}$. Send $\{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}$ to $\mathcal{A}$.

Start dictionary $\mathcal{R}$ which is initially empty.

**Query Phase.** $\mathcal{A}$ sends $(\mathsf{id}, \mathsf{GID}, x) \in [n] \times \mathcal{GID}_\lambda \times \{0,1\}^L$. If this is a new $\mathsf{GID}$ query, sample $\mathbf{R}^{\mathsf{GID}} \xleftarrow{\$} (\{0,1\}^L)^{n\times n}$. Otherwise, retrieve $\mathbf{R}^{\mathsf{GID}}$ from $\mathcal{R}$. Parse $\mathbf{R}^{\mathsf{GID}}$'s $\mathsf{id}$-th row as $R_1, \ldots, R_n$.

- Set $\widetilde{R} = (\bigoplus_{j\in[n]} R_j) \oplus x$.
- Sample for each $j \in [n]$, $\mathsf{nce.ct}_{\mathsf{id}}^{(j)} \leftarrow \mathsf{NCE.Sim}_{\mathsf{id}}^{(j)}(\mathsf{Enc}, \mathsf{GID})$. Any query made by $\mathsf{Sim}$ is answered using $\mathbf{R}^{\mathsf{GID}}$.
- Sample for each $i \in [n]$, $\mathsf{nce.sk}_i^{(\mathsf{id})} \leftarrow \mathsf{NCE.Sim}_i^{(\mathsf{id})}(\mathsf{KGen}, \mathsf{GID})$. Any query made by $\mathsf{Sim}$ is answered using $\mathbf{R}^{\mathsf{GID}}$.

Send $\mathsf{ct}_{\mathsf{id},\mathsf{GID}} := \left((\mathsf{nce.sk}_i^{(\mathsf{id})})_{i\in[n]}, (\mathsf{nce.ct}_{\mathsf{id}}^{(j)})_{j\in[n]}, \widetilde{R}\right)$ to $\mathcal{A}$.

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

**$\mathbf{Hyb}_4^{\mathcal{A}}(1^\lambda)$.** In this hybrid, we will set $\widetilde{R}$ to be uniformly random if this is not the last authority query and program the values of $x_1, \ldots, x_n$ if this is the last query. This is the description of $\mathsf{Sim}$ from Definition 6.9. The changes are highlighted in red.

**Setup Phase.** $\mathcal{A}$ sends $n, L$. For each $\mathsf{id} \in [n], i \in [n]$, sample $\mathsf{nce.mpk}_i^{(\mathsf{id})} \leftarrow \mathsf{NCE.Sim}_i^{(\mathsf{id})}(1^\lambda, 1^L)$. Set $\mathsf{CRS} := (\lambda, n, L), \mathsf{MPK}_{\mathsf{id}} = (\mathsf{nce.mpk}_i^{(\mathsf{id})})_{i\in[n]}$. Send $\{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}$ to $\mathcal{A}$.

Start dictionary $\mathcal{R}$ which is initially empty. Also start dictionary $\mathcal{V}, \mathcal{R}'$ initially empty.

**Query Phase.** $\mathcal{A}$ sends $(\mathsf{id}, \mathsf{GID}, x) \in [n] \times \mathcal{GID}_\lambda \times \{0,1\}^L$. Store $(\mathsf{id}, \mathsf{GID}, x)$ in $\mathcal{V}$. If this is a new $\mathsf{GID}$ query, sample $\mathbf{R}^{\mathsf{GID}} \xleftarrow{\$} (\{0,1\}^L)^{n\times n}$. Otherwise, retrieve $\mathbf{R}^{\mathsf{GID}}$ from $\mathcal{R}$.

- If this is not the last authority to be queried for $\mathsf{GID}$, sample $\widetilde{R} \leftarrow \{0,1\}^L$. Store $(\mathsf{id}, \mathsf{GID}, \widetilde{R})$ in $\mathcal{R}'$. Otherwise, retrieve and $(x_1, \ldots, x_n)$ from $\mathcal{V}$ and $(\widetilde{R}_{\mathsf{idx}})_{\mathsf{idx}\neq\mathsf{id}}$ from $\mathcal{R}'$ for this $\mathsf{GID}$. Sample $\widetilde{R}_{\mathsf{id}} \xleftarrow{\$} \{0,1\}^L$. Replace $\mathsf{id}$-th column of $\mathbf{R}^{\mathsf{GID}}$ with values $\left(\widetilde{R} \oplus \left(\bigoplus_{j\neq\mathsf{id}} R_{\mathsf{id},j}^{\mathsf{GID}}\right) \oplus x_{\mathsf{id}}\right)$.
- Sample for each $j \in [n]$, $\mathsf{nce.ct}_{\mathsf{id}}^{(j)} \leftarrow \mathsf{NCE.Sim}_{\mathsf{id}}^{(j)}(\mathsf{Enc}, \mathsf{GID})$. Any query made by $\mathsf{Sim}$ is answered using $\mathbf{R}^{\mathsf{GID}}$.

- Sample for each $i \in [n]$, $\mathsf{nce.sk}_i^{(\mathsf{id})} \leftarrow \mathsf{NCE.Sim}_i^{(\mathsf{id})}(\mathsf{KGen}, \mathsf{GID})$. Any query made by $\mathsf{Sim}$ is answered using $\mathbf{R}^{\mathsf{GID}}$.

Send $\mathsf{ct}_{\mathsf{id},\mathsf{GID}} := \left((\mathsf{nce.sk}_i^{(\mathsf{id})})_{i\in[n]}, (\mathsf{nce.ct}_{\mathsf{id}}^{(j)})_{j\in[n]}, \widetilde{R}\right)$ to $\mathcal{A}$.

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

**Claim H.7.** $\mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_{1,1,1}^{\mathcal{A}}(1^\lambda)$ are identically distributed.

*Proof.* As we are not simulating any $\mathsf{VNCE}$ instantiations in $\mathbf{Hyb}_{1,1,1}^{\mathcal{A}}(1^\lambda)$, these two hybrids are identically distributed. □

**Claim H.8.** Assuming the security of $\mathsf{NCE}$, $\mathbf{Hyb}_{1,\iota,\kappa}^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_{1,\iota,\kappa+1}^{\mathcal{A}}(1^\lambda)$ for any $\iota \in [n], \kappa \in [n]$ are computationally indistinguishable.

*Proof.* Assume that there exists an adversary $\mathcal{A}$ that can distinguish between $\mathbf{Hyb}_{1,\iota,\kappa}^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_{1,\iota,\kappa+1}^{\mathcal{A}}(1^\lambda)$ for some $\iota \in [n], \kappa \in [n]$ with non-negligible probability $\epsilon(\lambda)$, i.e.,

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

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

**Setup Phase.** $\mathcal{A}$ sends $n, L$. Send $1^L$ to $\mathcal{O}$ and receive $\mathsf{nce.mpk}$. For each $\mathsf{id} \in [n], i \in [n]$, if $(\mathsf{id} > \iota)$ or $(\mathsf{id} \leq \iota, i > \kappa)$, sample $(\mathsf{nce.mpk}_i^{(\mathsf{id})}, \mathsf{nce.msk}_i^{(\mathsf{id})}) \leftarrow \mathsf{NCE.Setup}(1^\lambda, 1^L)$. Otherwise, if $\mathsf{id} = \iota, i = \kappa$, set $\mathsf{nce.mpk}_i^{(\mathsf{id})} := \mathsf{nce.mpk}$. Otherwise, sample $\mathsf{nce.mpk}_i^{(\mathsf{id})} \leftarrow \mathsf{NCE.Sim}_i^{(\mathsf{id})}(1^\lambda, 1^L)$. Set $\mathsf{CRS} := (\lambda, n, L), \mathsf{MPK}_{\mathsf{id}} = (\mathsf{nce.mpk}_i^{(\mathsf{id})})_{i\in[n]}$. Send $\{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}$ to $\mathcal{A}$.

Start dictionaries $\mathcal{V}_i^{(j)}$ for each $j \in [n], i \in [n]$.

**Query Phase.** $\mathcal{A}$ sends $(\mathsf{id}, \mathsf{GID}, x) \in [n] \times \mathcal{GID}_\lambda \times \{0,1\}^L$.

- Sample $R_1, \ldots, R_n \leftarrow \{0,1\}^L$. Set $\widetilde{R} = (\bigoplus_{j\in[n]} R_j) \oplus x$. Add $(\mathsf{GID}, R_j)$ to $\mathcal{V}_{\mathsf{id}}^{(j)}$.
- Sample for each $j \in [n]$ if $j > \iota$ or $(j \leq \iota, \mathsf{id} > \kappa)$, $\mathsf{nce.ct}_{\mathsf{id}}^{(j)} \leftarrow \mathsf{NCE.Enc}(\mathsf{nce.mpk}_{\mathsf{id}}^{(j)}, \mathsf{GID}, R_j)$. Otherwise, if $j = \iota, \mathsf{id} = \kappa$, send $(\mathsf{Enc}, R_j)$ to $\mathcal{O}$ to receive $\mathsf{nce.ct}_{\mathsf{id}}^{(j)}$. Otherwise, $\mathsf{nce.ct}_{\mathsf{id}}^{(j)} \leftarrow \mathsf{NCE.Sim}_{\mathsf{id}}^{(j)}(\mathsf{Enc}, \mathsf{GID})$. Any query made by $\mathsf{Sim}$ is answered using dictionaries.
- Sample for each $i \in [n]$, if $\mathsf{id} > \iota$ or $(\mathsf{id} \leq \iota, i > \kappa)$, $\mathsf{nce.sk}_i^{(\mathsf{id})} \leftarrow \mathsf{NCE.KGen}(\mathsf{nce.msk}_i^{(\mathsf{id})}, \mathsf{GID})$. Otherwise, if $\mathsf{id} = \iota, i = \kappa$, send $(\mathsf{KGen}, \mathsf{GID})$ to $\mathcal{O}$ to receive $\mathsf{nce.sk}_i^{(\mathsf{id})}$. Otherwise, $\mathsf{nce.sk}_i^{(\mathsf{id})} \leftarrow \mathsf{NCE.Sim}_i^{(\mathsf{id})}(\mathsf{KGen}, \mathsf{GID})$. Any query made by $\mathsf{Sim}$ is answered using dictionaries.

Send $\mathsf{ct}_{\mathsf{id},\mathsf{GID}} := \left((\mathsf{nce.sk}_i^{(\mathsf{id})})_{i\in[n]}, (\mathsf{nce.ct}_{\mathsf{id}}^{(j)})_{j\in[n]}, \widetilde{R}\right)$ 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 $\mathcal{A}$, $\lambda$. If $\mathcal{O}$ uses $\mathsf{Enc}, \mathsf{KGen}$ $\mathcal{B}$ behaves like $\mathbf{Hyb}^{\mathcal{A}}_{1,\iota,\kappa}(1^\lambda)$ and if $\mathcal{O}$ use $\mathsf{Fake}, \mathsf{Reveal}$, $\mathcal{B}$ behaves like $\mathbf{Hyb}^{\mathcal{A}}_{1,\iota,\kappa+1}(1^\lambda)$. Hence, $\mathcal{B}$ is a valid adversary that break the security of NCE with non-negligible probability. Thus, $\mathbf{Hyb}^{\mathcal{A}}_{1,\iota,\kappa}(1^\lambda)$ and $\mathbf{Hyb}^{\mathcal{A}}_{1,\iota,\kappa+1}(1^\lambda)$ are computationally indistinguishable. □

**Claim H.9.** Assuming the security of NCE, $\mathbf{Hyb}^{\mathcal{A}}_{1,\iota,n+1}(1^\lambda)$ and $\mathbf{Hyb}^{\mathcal{A}}_{1,\iota+1,1}(1^\lambda)$ for any $\iota \in [n-1]$ are computationally indistinguishable.

*Proof.* As the only difference between these hybrids is that we simulate the $(\iota, n)$-th instantiation of the $\iota$-th authority, the proof of this claim is similar to proof of Claim H.8. □

**Claim H.10.** $\mathbf{Hyb}^{\mathcal{A}}_2(1^\lambda)$, $\mathbf{Hyb}^{\mathcal{A}}_3(1^\lambda)$, and $\mathbf{Hyb}^{\mathcal{A}}_4(1^\lambda)$ are identically distributed.

*Proof.* As the only difference between $\mathbf{Hyb}^{\mathcal{A}}_2(1^\lambda)$ and $\mathbf{Hyb}^{\mathcal{A}}_3(1^\lambda)$ is that in $\mathbf{Hyb}^{\mathcal{A}}_3(1^\lambda)$, we are sampling all the random strings $\{R_i\}_i$ a-priori for each authority and in $\mathbf{Hyb}^{\mathcal{A}}_2(1^\lambda)$, on-the-fly. As both as identically distributed, these two hybrids are identically distributed.

For the case of $\mathbf{Hyb}^{\mathcal{A}}_2(1^\lambda)$ and $\mathbf{Hyb}^{\mathcal{A}}_3(1^\lambda)$, consider the matrix $\mathbf{R}^{\mathsf{GID}}$ and add *one* more column to it corresponding to $R_{\mathsf{id}}$ sampled by each $\mathsf{id}$ for $\mathsf{GID}$. The $i$-th row of this resultant matrix are additive secret shares of $x_i$. It is the same for both $\mathbf{Hyb}^{\mathcal{A}}_2(1^\lambda)$ and $\mathbf{Hyb}^{\mathcal{A}}_3(1^\lambda)$. We are just changing where the input $x_i$ is added. Thus, both of these hybrids are identically distributed. □

□

# I Proofs from Section 6.6

In this section, we give the full proof of Theorem 6.11.

**$\mathbf{Hyb}^{\mathcal{A}}_0(1^\lambda)$.** This is the real experiment with $\mathsf{Chal}$ with $b = 0/1$.

**Setup Phase.** $\mathcal{A}$ sends $n, s, \ell_1, \ldots, \ell_n$. Define $L = \max_{\mathsf{id}}\{\ell_{\mathsf{id}}\}$. $\mathsf{Chal}$ samples $\mathsf{mafe.crs} \leftarrow \mathsf{asMAFE.GSetup}(1^\lambda, 1^n, 1^s, 1^L)$ and $\mathsf{nce.crs} \leftarrow \mathsf{VNCE.GSetup}(1^\lambda, 1^n, 1^{L'})$. $\mathsf{Chal}$ sets $\mathsf{CRS} := (\mathsf{mafe.crs}, \mathsf{nce.crs})$.

$\mathsf{Chal}$ then samples for each $\mathsf{id} \in [n]$, $(\mathsf{mafe.mpk}_{\mathsf{id}}, \mathsf{mafe.msk}_{\mathsf{id}}) \leftarrow \mathsf{asMAFE.ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}})$ and $(\mathsf{nce.mpk}, \mathsf{nce.msk}_{\mathsf{id}}) \leftarrow \mathsf{VNCE.ASetup}(\mathsf{id})$. Set $\mathsf{MPK}_{\mathsf{id}} := (\mathsf{mafe.mpk}_{\mathsf{id}}, \mathsf{nce.mpk}_{\mathsf{id}})$ and $\mathsf{MSK}_{\mathsf{id}} := (\mathsf{mafe.msk}_{\mathsf{id}}, \mathsf{nce.msk}_{\mathsf{id}})$.

$\mathsf{Chal}$ sends $\mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}$ to $\mathcal{A}$.

**Pre-challenge Queries.** $\mathcal{A}$ sends queries of the form $(\mathsf{id}, \mathsf{GID}, x)$. Sample $\mathsf{mafe.sk}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{asMAFE.KGen}(\mathsf{id}, \mathsf{mafe.msk}_{\mathsf{id}}, \{\mathsf{mafe.mpk}_{\mathsf{idx}}\}_{\mathsf{idx}\in[n]}, \mathsf{GID}, x)$. Then encrypt using $\mathsf{VNCE}$, $\mathsf{nce.sk}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{VNCE.Enc}(\mathsf{id}, \mathsf{nce.msk}_{\mathsf{id}}, \{\mathsf{nce.mpk}_{\mathsf{idx}}\}_{\mathsf{idx}\in[n]}, \mathsf{GID}, \mathsf{mafe.sk}_{\mathsf{id},\mathsf{GID},x})$.

Send $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} := \mathsf{nce.sk}_{\mathsf{id},\mathsf{GID},x}$ to $\mathcal{A}$.

**Challenge Query.** $\mathcal{A}$ sends circuits $C_0, C_1 \in \mathcal{C}_\lambda$ such that $|C_0| = |C_1|$ and they are of size at most $s$. $\mathsf{Chal}$ samples $\mathsf{mafe.ct} \leftarrow \mathsf{asMAFE.Enc}(\{\mathsf{mafe.mpk}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}, C_b)$ and sends $\mathsf{CT} := \mathsf{mafe.ct}$ to $\mathcal{A}$.

**Post-challenge Queries.** This is same as pre-challenge queries.

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

**$\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$.** In this hybrid, we simulate the VNCE instantiation. Changes are highlighted in red.

**Setup Phase.** $\mathcal{A}$ sends $n, s, \ell_1, \ldots, \ell_n$. Define $L = \max_{\mathsf{id}}\{\ell_{\mathsf{id}}\}$. Sample $\mathsf{mafe.crs} \leftarrow \mathsf{asMAFE.GSetup}(1^\lambda, 1^n, 1^s, 1^L)$. Then, for each $\mathsf{id} \in [n]$, $(\mathsf{mafe.mpk_{id}}, \mathsf{mafe.msk_{id}}) \leftarrow \mathsf{asMAFE.ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}})$. Also, $(\mathsf{nce.crs}, \{\mathsf{nce.mpk_{id}}\}_{\mathsf{id}\in[n]}) \leftarrow \mathsf{VNCE.Sim}(1^\lambda, 1^n, 1^{L'})$. Also initiate an empty dictionary of queries $\mathcal{Q}$.

Set $\mathsf{CRS} := (\mathsf{mafe.crs}, \mathsf{nce.crs})$. Set $\mathsf{MPK_{id}} := (\mathsf{mafe.mpk_{id}}, \mathsf{nce.mpk_{id}})$ and $\mathsf{MSK_{id}} := \mathsf{mafe.msk_{id}}$.

Send $\mathsf{CRS}, \{\mathsf{MPK_{id}}\}_{\mathsf{id}\in[n]}$ to $\mathcal{A}$.

**Pre-challenge Queries.** $\mathcal{A}$ sends queries of the form $(\mathsf{id}, \mathsf{GID}, x)$.

- If this is the last query for this GID, sample for each $\mathsf{id} \in [n]$, $\mathsf{mafe.sk_{id,GID},x} \leftarrow \mathsf{asMAFE.KGen}(\mathsf{id}, \mathsf{mafe.msk_{id}}, \{\mathsf{mafe.mpk_{idx}}\}_{\mathsf{idx}\in[n]}, \mathsf{GID}, x_{\mathsf{id,GID}})$ by retrieving $((1, \mathsf{GID}, x_{1,\mathsf{GID}}), \ldots, (n, \mathsf{GID}, x_{n,\mathsf{GID}}))$ from $\mathcal{Q}$. Let $\mathsf{V} := (\mathsf{mafe.sk}_{1,\mathsf{GID},x_{1,\mathsf{GID}}}, \ldots, \mathsf{mafe.sk}_{n,\mathsf{GID},x_{n,\mathsf{GID}}})$. Sample $\mathsf{nce.sk_{id,GID},x} \leftarrow \mathsf{VNCE.Sim}(\mathsf{id}, \mathsf{GID}, x, \mathsf{V})$.
- Otherwise, add $(\mathsf{id}, \mathsf{GID}, x)$ to $\mathcal{Q}$ and simulate $\mathsf{nce.sk}_{\mathsf{id,GID},x_{\mathsf{id}}} \leftarrow \mathsf{VNCE.Sim}(\mathsf{id}, \mathsf{GID}, x, \perp)$.

Send $\mathsf{SK_{id,GID},x} := \mathsf{nce.sk_{id,GID},x}$ to $\mathcal{A}$.

**Challenge Query.** $\mathcal{A}$ sends circuits $C_0, C_1 \in \mathcal{C}_\lambda$ such that $|C_0| = |C_1|$ and they are of size at most $s$. Sample $\mathsf{mafe.ct} \leftarrow \mathsf{asMAFE.Enc}(\{\mathsf{mafe.mpk_{id}}\}_{\mathsf{id}\in[n]}, C_b)$ and sends $\mathsf{CT} := \mathsf{mafe.ct}$ to $\mathcal{A}$.

**Post-challenge Queries.** This is same as pre-challenge queries.

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

**$\mathbf{Hyb}_2^{\mathcal{A}}(1^\lambda)$.** In this hybrid, we change the circuit encrypted in asMAFE to $C_{1-b}$. Changes are highlighted in red.

**Setup Phase.** $\mathcal{A}$ sends $n, s, \ell_1, \ldots, \ell_n$. Define $L = \max_{\mathsf{id}}\{\ell_{\mathsf{id}}\}$. Sample $\mathsf{mafe.crs} \leftarrow \mathsf{asMAFE.GSetup}(1^\lambda, 1^n, 1^s, 1^L)$. Then, for each $\mathsf{id} \in [n]$, $(\mathsf{mafe.mpk_{id}}, \mathsf{mafe.msk_{id}}) \leftarrow \mathsf{asMAFE.ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}})$. Also, $(\mathsf{nce.crs}, \{\mathsf{nce.mpk_{id}}\}_{\mathsf{id}\in[n]}) \leftarrow \mathsf{VNCE.Sim}(1^\lambda, 1^n, 1^{L'})$. Also initiate an empty dictionary of queries $\mathcal{Q}$.

Set $\mathsf{CRS} := (\mathsf{mafe.crs}, \mathsf{nce.crs})$. Set $\mathsf{MPK_{id}} := (\mathsf{mafe.mpk_{id}}, \mathsf{nce.mpk_{id}})$ and $\mathsf{MSK_{id}} := (\mathsf{mafe.msk_{id}}, \mathsf{nce.msk_{id}})$.

Send $\mathsf{CRS}, \{\mathsf{MPK_{id}}\}_{\mathsf{id}\in[n]}$ to $\mathcal{A}$.

**Pre-challenge Queries.** $\mathcal{A}$ sends queries of the form $(\mathsf{id}, \mathsf{GID}, x)$.

- If this is the last query for this GID, sample for each $\mathsf{id} \in [n]$, $\mathsf{mafe.sk_{id,GID},x} \leftarrow \mathsf{asMAFE.KGen}(\mathsf{id}, \mathsf{mafe.msk_{id}}, \{\mathsf{mafe.mpk_{idx}}\}_{\mathsf{idx}\in[n]}, \mathsf{GID}, x_{\mathsf{id,GID}})$ by retrieving $((1, \mathsf{GID}, x_{1,\mathsf{GID}}), \ldots, (n, \mathsf{GID}, x_{n,\mathsf{GID}}))$ from $\mathcal{Q}$. Let $\mathsf{V} := (\mathsf{mafe.sk}_{1,\mathsf{GID},x_{1,\mathsf{GID}}}, \ldots, \mathsf{mafe.sk}_{n,\mathsf{GID},x_{n,\mathsf{GID}}})$. Sample $\mathsf{nce.sk_{id,GID},x} \leftarrow \mathsf{VNCE.Sim}(\mathsf{id}, \mathsf{GID}, x, \mathsf{V})$.
- Otherwise, add $(\mathsf{id}, \mathsf{GID}, x)$ to $\mathcal{Q}$ and simulate $\mathsf{nce.sk}_{\mathsf{id,GID},x_{\mathsf{id}}} \leftarrow \mathsf{VNCE.Sim}(\mathsf{id}, \mathsf{GID}, x, \perp)$.

Send $\mathsf{SK_{id,GID},x} := \mathsf{nce.sk_{id,GID},x}$ to $\mathcal{A}$.

**Challenge Query.** $\mathcal{A}$ sends circuits $C_0, C_1 \in \mathcal{C}_\lambda$ such that $|C_0| = |C_1|$ and they are of size at most $s$. Sample $\mathsf{mafe.ct} \leftarrow \mathsf{asMAFE.Enc}(\{\mathsf{mafe.mpk_{id}}\}_{\mathsf{id}\in[n]}, C_{1-b})$ and sends $\mathsf{CT} := \mathsf{mafe.ct}$ to $\mathcal{A}$.

**Post-challenge Queries.** This is same as pre-challenge queries.

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

**$\mathbf{Hyb}_3^{\mathcal{A}}(1^\lambda)$.** This is experiment Hybrid 0 with $1-b$.

**Lemma I.1.** Assuming the security of VNCE, $\mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$ are computationally indistinguishable.

*Proof.* Assume that there exists an adversary $\mathcal{A}$ that can distinguish between $\mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$ with non-negligible probability $\epsilon(\lambda)$, 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 VNCE with non-negligible probability. The description of $\mathcal{B}^{\mathcal{O}}$ is as follows.

**Setup Phase.** $\mathcal{A}$ sends $n, s, \ell_1, \ldots, \ell_n$. Define $L = \max_{\mathsf{id}}\{\ell_{\mathsf{id}}\}$. Sample $\mathsf{mafe.crs} \leftarrow \mathsf{asMAFE.GSetup}(1^\lambda, 1^n, 1^s, 1^L)$. Then, for each $\mathsf{id} \in [n]$, $(\mathsf{mafe.mpk_{id}}, \mathsf{mafe.msk_{id}}) \leftarrow \mathsf{asMAFE.ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}})$. Receive, $(\mathsf{nce.crs}, \{\mathsf{nce.mpk_{id}}\}_{\mathsf{id}\in[n]})$ from $\mathcal{O}$. Also initiate an empty dictionary of queries $\mathcal{Q}$.

Set $\mathsf{CRS} := (\mathsf{mafe.crs}, \mathsf{nce.crs})$. Set $\mathsf{MPK_{id}} := (\mathsf{mafe.mpk_{id}}, \mathsf{nce.mpk_{id}})$ and $\mathsf{MSK_{id}} := \mathsf{mafe.msk_{id}}$.

Send $\mathsf{CRS}, \{\mathsf{MPK_{id}}\}_{\mathsf{id}\in[n]}$ to $\mathcal{A}$.

**Pre-challenge Queries.** $\mathcal{A}$ sends queries of the form $(\mathsf{id}, \mathsf{GID}, x)$. Sample $\mathsf{mafe.sk_{id,GID},x} \leftarrow \mathsf{asMAFE.KGen}(\mathsf{id}, \mathsf{mafe.msk_{id}}, \{\mathsf{mafe.mpk_{idx}}\}_{\mathsf{idx}\in[n]}, \mathsf{GID}, x)$. Then send $(\mathsf{id}, \mathsf{GID}, \mathsf{mafe.sk_{id,GID},x})$ to $\mathcal{O}$ to receive $\mathsf{nce.sk_{id,GID},x}$.

Send $\mathsf{SK_{id,GID},x} := \mathsf{nce.sk_{id,GID},x}$ to $\mathcal{A}$.

**Challenge Query.** $\mathcal{A}$ sends circuits $C_0, C_1 \in \mathcal{C}_\lambda$ such that $|C_0| = |C_1|$ and they are of size at most $s$. Sample $\mathsf{mafe.ct} \leftarrow \mathsf{asMAFE.Enc}(\{\mathsf{mafe.mpk_{id}}\}_{\mathsf{id}\in[n]}, C_b)$ and sends $\mathsf{CT} := \mathsf{mafe.ct}$ to $\mathcal{A}$.

**Post-challenge Queries.** This is same as pre-challenge queries.

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

As we can see, the running time of $\mathcal{B}$ is polynomial in the running time of $\mathcal{A}$, $\lambda$. If $\mathcal{O}$ is an honest challenger for VNCE $\mathcal{B}$ behaves like $\mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)$ and if $\mathcal{O}$ simulates VNCE $\mathcal{B}$ behaves like $\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$. Hence, $\mathcal{B}$ is a valid adversary that break the security of VNCE with non-negligible probability. Thus, $\mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$ are computationally indistinguishable. □

**Lemma I.2.** Assuming the security of asMAFE, $\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_2^{\mathcal{A}}(1^\lambda)$ are computationally indistinguishable.

*Proof.* Assume that there exists an adversary $\mathcal{A}$ that can distinguish between $\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_2^{\mathcal{A}}(1^\lambda)$ with non-negligible probability $\epsilon(\lambda)$, 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 $\mathsf{asMAFE}$ with non-negligible probability. The description of $\mathcal{B}^{\mathcal{O}}$ is as follows.

**Setup Phase.** $\mathcal{A}$ sends $n, s, \ell_1, \ldots, \ell_n$. Define $L = \max_{\mathsf{id}}\{\ell_{\mathsf{id}}\}$. Receive $\mathsf{mafe.crs}, \{\mathsf{mafe.mpk}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}$ from $\mathcal{O}$. Also, $(\mathsf{nce.crs}, \{\mathsf{nce.mpk}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}) \leftarrow \mathsf{VNCE.Sim}(1^\lambda, 1^n, 1^{L'})$. Also, initiate an empty dictionary of queries $\mathcal{Q}$.

Set $\mathsf{CRS} := (\mathsf{mafe.crs}, \mathsf{nce.crs})$. Set $\mathsf{MPK}_{\mathsf{id}} := (\mathsf{mafe.mpk}_{\mathsf{id}}, \mathsf{nce.mpk}_{\mathsf{id}})$.

Send $\mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id}\in[n]}$ to $\mathcal{A}$.

**Pre-challenge Queries.** $\mathcal{A}$ sends queries of the form $(\mathsf{id}, \mathsf{GID}, x)$.

- If this is the last query for this $\mathsf{GID}$, retrieve $((1, \mathsf{GID}, x_{1,\mathsf{GID}}), \ldots, (n, \mathsf{GID}, x_{n,\mathsf{GID}}))$ from $\mathcal{Q}$. Send $\mathsf{GID}, x_{1,\mathsf{GID}}, \ldots, x_{n,\mathsf{GID}}$ to $\mathcal{O}$ to receive $\mathsf{mafe.sk}_{1,\mathsf{GID},x_{1,\mathsf{GID}}}, \ldots, \mathsf{mafe.sk}_{n,\mathsf{GID},x_{n,\mathsf{GID}}}$. Let $\mathsf{V} := (\mathsf{mafe.sk}_{1,\mathsf{GID},x_{1,\mathsf{GID}}}, \ldots, \mathsf{mafe.sk}_{n,\mathsf{GID},x_{n,\mathsf{GID}}})$. Sample $\mathsf{nce.sk}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{VNCE.Sim}(\mathsf{id}, \mathsf{GID}, x, \mathsf{V})$.
- Otherwise, add $(\mathsf{id}, \mathsf{GID}, x)$ to $\mathcal{Q}$ and simulate $\mathsf{nce.sk}_{\mathsf{id},\mathsf{GID},x_{\mathsf{id}}} \leftarrow \mathsf{VNCE.Sim}(\mathsf{id}, \mathsf{GID}, x, \perp)$.

Send $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} := \mathsf{nce.sk}_{\mathsf{id},\mathsf{GID},x}$ to $\mathcal{A}$.

**Challenge Query.** $\mathcal{A}$ sends circuits $C_0, C_1 \in \mathcal{C}_\lambda$ such that $|C_0| = |C_1|$ and they are of size at most $s$. Send $C_b, C_{1-b}$ to $\mathcal{O}$ to receive $\mathsf{mafe.ct}$ and sends $\mathsf{CT} := \mathsf{mafe.ct}$ to $\mathcal{A}$.

**Post-challenge Queries.** This is same as pre-challenge queries.

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

As we can see, the running time of $\mathcal{B}$ is polynomial in the running time of $\mathcal{A}$, $\lambda$. If $\mathcal{O}$ encryption $C_b$ $\mathcal{B}$ behaves like $\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$ and if $\mathcal{O}$ encrypts $C_{1-b}$ $\mathcal{B}$ behaves like $\mathbf{Hyb}_2^{\mathcal{A}}(1^\lambda)$. Hence, $\mathcal{B}$ is a valid adversary that break the security of $\mathsf{asMAFE}$ with non-negligible probability. Thus, $\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_2^{\mathcal{A}}(1^\lambda)$ are computationally indistinguishable. □

**Lemma I.3.** Assuming the security of $\mathsf{VNCE}$, $\mathbf{Hyb}_2^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_3^{\mathcal{A}}(1^\lambda)$ are computationally indistinguishable.

*Proof.* The proof of this lemma is similar to proof of Lemma I.1. □

# J Proofs from Section 7.2

In this section, we provide full proof of Theorem 7.3.

**$\mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)$.** This is the honest experiment with $\mathsf{Chal}$ with $b = 0/1$.

**Setup.** $\mathcal{A}$ sends $Q, n, s, k, \chi, \ell_1, \ldots, \ell_n$ to $\mathsf{Chal}$. For each $i \in \mathbb{I}, j \in \mathbb{J}$, $\mathsf{Chal}$ samples $\mathsf{crs}^{(i,j)} \leftarrow \mathsf{MAFE.GSetup}(1^\lambda, 1^Q, 1^{n-k}, 1^s, 1^{L+\lambda+\ell_{\mathsf{out}}})$. In addition, sample for each $\mathsf{id} \in [n]$, $(\mathsf{mpk}_{\mathsf{id}}^{(i,j)}, \mathsf{msk}_{\mathsf{id}}^{(i,j)}) \leftarrow \mathsf{MAFE.ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}+\lambda+\ell_{\mathsf{out}}})$ 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 $q \in [Q]$ to $\mathsf{Chal}$ such that $\mathsf{id}$ is not corrupted. For each $i \in \mathbb{I}'_{\mathsf{id}}, j \in \mathbb{J}$, sample $r_{ij} \xleftarrow{\$} \{0,1\}^\lambda, \mathsf{ske.ct}^*_{ij} \leftarrow \{0,1\}^{\ell_{\mathsf{out}}}$, and $\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j)} \leftarrow \mathsf{MAFE.KGen}(\mathsf{id}, \mathsf{msk}_{\mathsf{id}}^{(i,j)}, \{\mathsf{mpk}_{\mathsf{idx}}^{(i,j)}\}_{\mathsf{idx}}, \mathsf{GID}, (x, r_{ij}, \mathsf{ske.ct}^*_{ij}))$. 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 $\mathsf{Chal}$. Send $(\mathsf{msk}_{\mathsf{id}}^{(i,j)})_{i,j}$ to $\mathcal{A}$.

**Challenge Query.** $\mathcal{A}$ sends circuits $C_0, C_1$ to $\mathsf{Chal}$. $\mathsf{Chal}$ samples $\{\widetilde{C}_i\}_{i\in\mathbb{I}'} \leftarrow \mathsf{FSS.Share}(1^\lambda, 1^{|\mathbb{I}'|}, C)$. For each $i \in \mathbb{I}', j \in \mathbb{J}$, sample $\mathsf{ct}^{(i,j)} \leftarrow \mathsf{MAFE.Enc}(\{\mathsf{mpk}_{\mathsf{id}}^{(i,j)}\}_{\mathsf{id}\notin\mathcal{K}_i}, F_{ij})$. Set $\mathsf{CT} = (\mathsf{ct}^{(i,j)})_{i,j}$ and send it to $\mathcal{A}$.

**Secret Key Queries.** $\mathcal{A}$ sends $(\mathsf{id}, \mathsf{GID}, x)$ to $\mathsf{Chal}$ such that $\mathsf{id}$ is not corrupted. For each $i \in \mathbb{I}'_{\mathsf{id}}, j \in \mathbb{J}$, sample $r_{ij} \xleftarrow{\$} \{0,1\}^\lambda, \mathsf{ske.ct}^*_{ij} \leftarrow \{0,1\}^{\ell_{\mathsf{out}}}$, and $\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j)} \leftarrow \mathsf{MAFE.KGen}(\mathsf{id}, \mathsf{msk}_{\mathsf{id}}^{(i,j)}, \{\mathsf{mpk}_{\mathsf{idx}}^{(i,j)}\}_{\mathsf{idx}}, \mathsf{GID}, (x, r_{ij}, \mathsf{ske.ct}^*_{ij}))$. 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 $\mathsf{Chal}$. Send $(\mathsf{msk}^{(i,j)})_{i,j}$ to $\mathcal{A}$.

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

**$\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$.** In this hybrid, we simply guess $i^* \xleftarrow{\$} \mathbb{I}'$. If any $\mathsf{id} \notin \mathcal{K}_{i^*}$ is corrupted, we will abort. The changes are highlighted in red.

**Setup.** $\mathcal{A}$ sends $Q, n, s, k, \chi, \ell_1, \ldots, \ell_n$. For each $i \in \mathbb{I}, j \in \mathbb{J}$, sample $\mathsf{crs}^{(i,j)} \leftarrow \mathsf{MAFE.GSetup}(1^\lambda, 1^Q, 1^{n-k}, 1^s, 1^{L+\lambda+\ell_{\mathsf{out}}})$. In addition, sample for each $\mathsf{id} \in [n]$, $(\mathsf{mpk}_{\mathsf{id}}^{(i,j)}, \mathsf{msk}_{\mathsf{id}}^{(i,j)}) \leftarrow \mathsf{MAFE.ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}+\lambda+\ell_{\mathsf{out}}})$ 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 $\mathcal{A}$.

Sample $i^* \xleftarrow{\$} \mathbb{I}'$.

**Secret Key Queries.** $\mathcal{A}$ sends $(\mathsf{id}, \mathsf{GID}, x)$ for $q \in [Q]$ such that $\mathsf{id}$ is not corrupted. For each $i \in \mathbb{I}'_{\mathsf{id}}, j \in \mathbb{J}$, sample $r_{ij} \xleftarrow{\$} \{0,1\}^\lambda, \mathsf{ske.ct}^*_{ij} \leftarrow \{0,1\}^{\ell_{\mathsf{out}}}$, and $\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j)} \leftarrow \mathsf{MAFE.KGen}(\mathsf{id}, \mathsf{msk}_{\mathsf{id}}^{(i,j)}, \{\mathsf{mpk}_{\mathsf{idx}}^{(i,j)}\}_{\mathsf{idx}}, \mathsf{GID}, (x, r_{ij}, \mathsf{ske.ct}^*_{ij}))$. 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 $(\mathsf{Corr}, \mathsf{id})$. If $\mathsf{id} \notin \mathcal{K}_{i^*}$, abort. Send $(\mathsf{msk}_{\mathsf{id}}^{(i,j)})_{i,j}$ to $\mathcal{A}$.

**Challenge Query.** $\mathcal{A}$ sends circuit $C_0, C_1$. Sample $\{\widetilde{C}_i\}_{i\in\mathbb{I}'} \leftarrow \mathsf{FSS.Share}(1^\lambda, 1^{|\mathbb{I}'|}, C_b)$. For each $i \in \mathbb{I}', j \in \mathbb{J}$, sample $\mathsf{ct}^{(i,j)} \leftarrow \mathsf{MAFE.Enc}(\{\mathsf{mpk}_{\mathsf{id}}^{(i,j)}\}_{\mathsf{id}\notin\mathcal{K}_i}, F_{ij})$. Set $\mathsf{CT} = (\mathsf{ct}^{(i,j)})_{i,j}$ and send it to $\mathcal{A}$.

**Secret Key Queries.** $\mathcal{A}$ sends $(\mathsf{id}, \mathsf{GID}, x)$ such that $\mathsf{id}$ is not corrupted. For each $i \in \mathbb{I}'_{\mathsf{id}}, j \in \mathbb{J}$, sample $r_{ij} \xleftarrow{\$} \{0,1\}^\lambda, \mathsf{ske.ct}^*_{ij} \leftarrow \{0,1\}^{\ell_{\mathsf{out}}}$, and $\mathsf{sk}^{(i,j)}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{MAFE.KGen}(\mathsf{id}, \mathsf{msk}^{(i,j)}_{\mathsf{id}}, \{\mathsf{mpk}^{(i,j)}_{\mathsf{idx}}\}_{\mathsf{idx}}, \mathsf{GID}, (x, r_{ij}, \mathsf{ske.ct}^*_{ij}))$. Set $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} = (\mathsf{sk}^{(i,j)}_{\mathsf{id},\mathsf{GID},x})_{i \in \mathbb{I}'_{\mathsf{id}}, j}$ and send it to $\mathcal{A}$.

**Corruption Queries.** $\mathcal{A}$ sends $(\mathsf{Corr}, \mathsf{id})$. If $\mathsf{id} \notin \mathcal{K}_{i^*}$, abort. Send $(\mathsf{msk}^{(i,j)})_{i,j}$ to $\mathcal{A}$.

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

**$\mathbf{Hyb}^{\mathcal{A}}_2(1^\lambda)$.** In this hybrid, we will sample a unique $\rho_{i^*j}$ per $\mathsf{GID}$ for each $j \in \mathbb{J}$. We will set each $r_{i^*j}$ for each authority as additive secret shares of $\rho_{i^*j}$. The changes are highlighted in red.

**Setup.** $\mathcal{A}$ sends $Q, n, s, k, \chi, \ell_1, \ldots, \ell_n$. For each $i \in \mathbb{I}, j \in \mathbb{J}$, sample $\mathsf{crs}^{(i,j)} \leftarrow \mathsf{MAFE.GSetup}(1^\lambda, 1^Q, 1^{n-k}, 1^s, 1^{L+\lambda+\ell_{\mathsf{out}}})$. In addition, sample for each $\mathsf{id} \in [n]$, $(\mathsf{mpk}^{(i,j)}_{\mathsf{id}}, \mathsf{msk}^{(i,j)}_{\mathsf{id}}) \leftarrow \mathsf{MAFE.ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}+\lambda+\ell_{\mathsf{out}}})$ for $i$ such that $\mathsf{id} \notin \mathcal{K}_i$. Set $\mathsf{CRS} := (\mathsf{crs}^{(i,j)})_{i,j}$, $\mathsf{MPK}_{\mathsf{id}} := (\mathsf{mpk}^{(i,j)}_{\mathsf{id}})_{i,j}$, and $\mathsf{MSK}_{\mathsf{id}} := (\mathsf{msk}^{(i,j)}_{\mathsf{id}})_{i,j}$ and send $(\mathsf{CRS}, \{\mathsf{MPK}_{\mathsf{id}}\}_{\mathsf{id} \in [n]})$ to $\mathcal{A}$.

Sample $i^* \xleftarrow{\$} \mathbb{I}'$. Initiate an empty dictionary $\mathcal{R}$. Sample $Q$ many $\rho_j$ for each $j \in \mathbb{J}$. If any of these are repeated, abort and output $\perp$. Otherwise, compute $r^{(1)}_j, \ldots, r^{(n)}_j$ such that $\bigoplus_{\mathsf{id}} r^{(\mathsf{id})}_j = \rho_j$ and add $(\mathsf{GID}, (r^{(\mathsf{id})}_j)_{j,\mathsf{id}})$ to $\mathcal{R}$.

**Secret Key Queries.** $\mathcal{A}$ sends $(\mathsf{id}, \mathsf{GID}, x)$ for $q \in [Q]$ such that $\mathsf{id}$ is not corrupted. For each $i \in \mathbb{I}'_{\mathsf{id}}, j \in \mathbb{J}$,

- If $i = i^*$, retrieve $(r^{(\mathsf{id})}_j)_{\mathsf{id},j}$ from $\mathcal{R}$. Set $r_{ij} := r^{(\mathsf{id})}_j$. Sample $\mathsf{ske.ct}^*_{ij} \leftarrow \{0,1\}^{\ell_{\mathsf{out}}}$, and $\mathsf{sk}^{(i,j)}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{MAFE.KGen}(\mathsf{id}, \mathsf{msk}^{(i,j)}_{\mathsf{id}}, \{\mathsf{mpk}^{(i,j)}_{\mathsf{idx}}\}_{\mathsf{idx}}, \mathsf{GID}, (x, r_{ij}, \mathsf{ske.ct}^*_{ij}))$.
- Otherwise, proceed similarly to $\mathbf{Hyb}^{\mathcal{A}}_1(1^\lambda)$.

Set $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} = (\mathsf{sk}^{(i,j)}_{\mathsf{id},\mathsf{GID},x})_{i \in \mathbb{I}'_{\mathsf{id}}, j}$ and send it to $\mathcal{A}$.

**Corruption Queries.** $\mathcal{A}$ sends $(\mathsf{Corr}, \mathsf{id})$. If $\mathsf{id} \notin \mathcal{K}_{i^*}$, abort. Send $(\mathsf{msk}^{(i,j)}_{\mathsf{id}})_{i,j}$ to $\mathcal{A}$.

**Challenge Query.** $\mathcal{A}$ sends circuit $C_0, C_1$. Sample $\{\widetilde{C}_i\}_{i \in \mathbb{I}'} \leftarrow \mathsf{FSS.Share}(1^\lambda, 1^{|\mathbb{I}'|}, C_b)$. For each $i \in \mathbb{I}', j \in \mathbb{J}$, sample $\mathsf{ct}^{(i,j)} \leftarrow \mathsf{MAFE.Enc}(\{\mathsf{mpk}^{(i,j)}_{\mathsf{id}}\}_{\mathsf{id} \notin \mathcal{K}_i}, F_{ij})$. Set $\mathsf{CT} = (\mathsf{ct}^{(i,j)})_{i,j}$ and send it to $\mathcal{A}$.

**Secret Key Queries.** $\mathcal{A}$ sends $(\mathsf{id}, \mathsf{GID}, x)$ such that $\mathsf{id}$ is not corrupted. For each $i \in \mathbb{I}'_{\mathsf{id}}, j \in \mathbb{J}$, sample $r_{ij} \xleftarrow{\$} \{0,1\}^\lambda, \mathsf{ske.ct}^*_{ij} \leftarrow \{0,1\}^{\ell_{\mathsf{out}}}$, and $\mathsf{sk}^{(i,j)}_{\mathsf{id},\mathsf{GID},x} \leftarrow \mathsf{MAFE.KGen}(\mathsf{id}, \mathsf{msk}^{(i,j)}_{\mathsf{id}}, \{\mathsf{mpk}^{(i,j)}_{\mathsf{idx}}\}_{\mathsf{idx}}, \mathsf{GID}, (x, r_{ij}, \mathsf{ske.ct}^*_{ij}))$. Set $\mathsf{SK}_{\mathsf{id},\mathsf{GID},x} = (\mathsf{sk}^{(i,j)}_{\mathsf{id},\mathsf{GID},x})_{i \in \mathbb{I}'_{\mathsf{id}}, j}$ and send it to $\mathcal{A}$.

**Corruption Queries.** $\mathcal{A}$ sends $(\mathsf{Corr}, \mathsf{id})$. If $\mathsf{id} \notin \mathcal{K}_{i^*}$, abort. Send $(\mathsf{msk}^{(i,j)})_{i,j}$ to $\mathcal{A}$.

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

**$\mathbf{Hyb}_3^{\mathcal{A}}(1^\lambda)$.** In this hybrid, we will set $\mathsf{ske.ct}^*_{i^*j}$ to be encryptions of output of $F_{i^*j}$ in the post-challenge phase for $i^*$. The changes are highlighted in red.

**Setup.** $\mathcal{A}$ sends $Q, n, s, k, \chi, \ell_1, \ldots, \ell_n$. For each $i \in \mathbb{I}, j \in \mathbb{J}$, sample $\mathsf{crs}^{(i,j)} \leftarrow \mathsf{MAFE.GSetup}(1^\lambda, 1^Q, 1^{n-k}, 1^s, 1^{L+\lambda+\ell_{\mathsf{out}}})$. In addition, sample for each $\mathsf{id} \in [n]$, $(\mathsf{mpk}_{\mathsf{id}}^{(i,j)}, \mathsf{msk}_{\mathsf{id}}^{(i,j)}) \leftarrow \mathsf{MAFE.ASetup}(\mathsf{id}, 1^{\ell_{\mathsf{id}}+\lambda+\ell_{\mathsf{out}}})$ 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 $\mathcal{A}$.

Sample $i^* \xleftarrow{\$} \mathbb{I}'$. Initiate an empty dictionary $\mathcal{R}$. Sample $Q$ many $\rho_j$ for each $j \in \mathbb{J}$. If any of these are repeated, abort and output $\bot$. Otherwise, compute $r_j^{(1)}, \ldots, r_j^{(n)}$ such that $\bigoplus_{\mathsf{id}} r_j^{(\mathsf{id})} = \rho_j$ and add $(\mathsf{GID}, (r_j^{(\mathsf{id})})_{j,\mathsf{id}})$ to $\mathcal{R}$. Sample $\mathsf{ske.sk}^* \leftarrow \mathsf{SKE.Setup}(1^\lambda)$.

**Secret Key Queries.** $\mathcal{A}$ sends $(\mathsf{id}, \mathsf{GID}, x)$ for $q \in [Q]$ such that $\mathsf{id}$ is not corrupted. For each $i \in \mathbb{I}'_{\mathsf{id}}, j \in \mathbb{J}$,

- If $i = i^*$, retrieve $(r_j^{(\mathsf{id})})_{\mathsf{id},j}$ from $\mathcal{R}$. Set $r_{ij} := r_j^{(\mathsf{id})}$. Sample $\mathsf{ske.ct}^*_{ij} \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}^*, r^*_{ij})$ for random $r^*_{ij}$ whose size is same as output size of $F_{i^*j}$, and $\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j)} \leftarrow \mathsf{MAFE.KGen}(\mathsf{id}, \mathsf{msk}_{\mathsf{id}}^{(i,j)}, \{\mathsf{mpk}_{\mathsf{idx}}^{(i,j)}\}_{\mathsf{idx}}, \mathsf{GID}, (x, r_{ij}, \mathsf{ske.ct}^*_{ij}))$.
- Otherwise, proceed similarly to $\mathbf{Hyb}_2^{\mathcal{A}}(1^\lambda)$.

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 $(\mathsf{Corr}, \mathsf{id})$. If $\mathsf{id} \notin \mathcal{K}_{i^*}$, abort. Send $(\mathsf{msk}_{\mathsf{id}}^{(i,j)})_{i,j}$ to $\mathcal{A}$.

**Challenge Query.** $\mathcal{A}$ sends circuit $C_0, C_1$. Sample $\{\widetilde{C}_i\}_{i\in\mathbb{I}'} \leftarrow \mathsf{FSS.Share}(1^\lambda, 1^{|\mathbb{I}'|}, C_b)$. For each $i \in \mathbb{I}', j \in \mathbb{J}$, sample $\mathsf{ct}^{(i,j)} \leftarrow \mathsf{MAFE.Enc}(\{\mathsf{mpk}_{\mathsf{id}}^{(i,j)}\}_{\mathsf{id}\notin\mathcal{K}_i}, F_{ij})$. 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 $q \in [Q]$ such that $\mathsf{id}$ is not corrupted. For each $i \in \mathbb{I}'_{\mathsf{id}}, j \in \mathbb{J}$,

- If $i = i^*$, retrieve $(r_j^{(\mathsf{id})})_{\mathsf{id},j}$ from $\mathcal{R}$. Set $r_{ij} := r_j^{(\mathsf{id})}$. If this is the last authority that will be queried for this GID, retrieve $X = \{x_{\mathsf{id}}\}_{\mathsf{id}\notin\mathcal{K}_{i^*}}$ and $\{r^{*(\mathsf{idx})}_{i^*j}\}_{\mathsf{idx}\neq\mathsf{id}}$ which are encrypted in each of $\mathsf{ske.ct}^*_{i^*j}$ for other authorities. Sample $\mathsf{ske.ct}^*_{ij} = \mathsf{SKE.Enc}(\mathsf{ske.sk}^*, \bigoplus_{\mathsf{idx}\neq\mathsf{id}} r^{*(\mathsf{idx})}_{ij} \oplus \mathsf{FSS.Eval}(i^*, \widetilde{C}_{i^*}, \{X, j\}))$. Otherwise, sample $\mathsf{ske.ct}^*_{ij} \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}^*, r^*_{ij})$ for random $r^*_{ij}$ whose size is same as output size of $F_{i^*j}$, and $\mathsf{sk}_{\mathsf{id},\mathsf{GID},x}^{(i,j)} \leftarrow \mathsf{MAFE.KGen}(\mathsf{id}, \mathsf{msk}_{\mathsf{id}}^{(i,j)}, \{\mathsf{mpk}_{\mathsf{idx}}^{(i,j)}\}_{\mathsf{idx}}, \mathsf{GID}, (x, r_{ij}, \mathsf{ske.ct}^*_{ij}))$.
- Otherwise, proceed similarly to $\mathbf{Hyb}_2^{\mathcal{A}}(1^\lambda)$.

**Corruption Queries.** $\mathcal{A}$ sends $(\mathsf{Corr}, \mathsf{id})$. If $\mathsf{id} \notin \mathcal{K}_{i^*}$, abort. Send $(\mathsf{msk}^{(i,j)})_{i,j}$ to $\mathcal{A}$.

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

**Circuit $F'_{i^*j}$**

**Input:** $x_1, r_1, \mathsf{ske.ct}^*_1, \ldots, x_{n-k}, r_{n-k}, \mathsf{ske.ct}^*_{n-k}$
**Hardwired:** $i, j, \mathsf{ske.sk}^*, \{(\rho_j^{\mathsf{GID}}, y_j)\}_{\mathsf{GID}}$ for each $\mathsf{GID}$ queried in pre-challenge secret key queries.
**Output:** If $(\bigoplus_i r_i)$ is any $\rho_j^{\mathsf{GID}}$, output $y_j$. Otherwise, let $r_i^* = \mathsf{SKE.Dec}(\mathsf{ske.sk}^*, \mathsf{ske.ct}_i^*)$ and output $\bigoplus_i r_i^*$.

Figure 9: Description of circuit $F'_{i^*j}$

**$\mathbf{Hyb}^{\mathcal{A}}_{4,\kappa}(1^\lambda)$.** For $\kappa \in [2^\chi + 1]$. In this hybrid, we will encrypt $F'_{i^*j}$ for $j \in [\kappa - 1]$ instantiations of $\mathsf{MAFE}$. The changes are highlighted in red.

**Setup, Secret Key Queries, Corruption Queries.** Same as $\mathbf{Hyb}^{\mathcal{A}}_3(1^\lambda)$.

**Challenge Query.** $\mathcal{A}$ sends circuit $C_0, C_1$. Sample $\{\widetilde{C}_i\}_{i\in\mathbb{I}'} \leftarrow \mathsf{FSS.Share}(1^\lambda, 1^{|\mathbb{I}'|}, C_b)$. For each $i \in \mathbb{I}', j \in \mathbb{J}$, if $i = i^*$, compute $y_j = \mathsf{FSS.Eval}(i^*, \widetilde{C}_{i^*}, \{X, j\})$ for each $(\mathsf{GID}, X)$ of the pre-challenge queries where for this $\mathsf{GID}$, all $n-k$ authorities are queried and sample $\mathsf{ct}^{(i,j)} \leftarrow \mathsf{MAFE.Enc}(\{\mathsf{mpk}_{\mathsf{id}}^{(i,j)}\}_{\mathsf{id}\notin\mathcal{K}_i}, F'_{i^*j})$. Otherwise, sample $\mathsf{ct}^{(i,j)} \leftarrow \mathsf{MAFE.Enc}(\{\mathsf{mpk}_{\mathsf{id}}^{(i,j)}\}_{\mathsf{id}\notin\mathcal{K}_i}, F_{ij})$. Set $\mathsf{CT} = (\mathsf{ct}^{(i,j)})_{i,j}$ and send it to $\mathcal{A}$.

**Secret Key Queries, Corruption Queries, Guess Phase.** Same as $\mathbf{Hyb}^{\mathcal{A}}_3(1^\lambda)$.

**$\mathbf{Hyb}^{\mathcal{A}}_5(1^\lambda)$.** In this hybrid, we simulate $\mathsf{FSS}$. The changes are highlighted in red.

**Setup, Secret Key Queries, Corruption Queries.** Same as $\mathbf{Hyb}^{\mathcal{A}}_{4,2^\chi+1}(1^\lambda)$.

**Challenge Query.** $\mathcal{A}$ sends circuit $C_0, C_1$. Sample $\{\widetilde{C}_i\}_{i\neq i^*, i\in\mathbb{I}'} \leftarrow \mathsf{FSS.Sim}(1^\lambda, 1^{|\mathbb{I}'|}, 1^{|C_b|}, 1^*)$. For each $i \in \mathbb{I}', j \in \mathbb{J}$, if $i = i^*$, compute $y_j = \mathsf{FSS.Sim}(X, C_b(X))$ for each $(\mathsf{GID}, X)$ of the pre-challenge queries where for this $\mathsf{GID}$ and sample $\mathsf{ct}^{(i,j)} \leftarrow \mathsf{MAFE.Enc}(\{\mathsf{mpk}_{\mathsf{id}}^{(i,j)}\}_{\mathsf{id}\notin\mathcal{K}_i}, F'_{i^*j})$. Otherwise, sample $\mathsf{ct}^{(i,j)} \leftarrow \mathsf{MAFE.Enc}(\{\mathsf{mpk}_{\mathsf{id}}^{(i,j)}\}_{\mathsf{id}\notin\mathcal{K}_i}, F_{ij})$. Set $\mathsf{CT} = (\mathsf{ct}^{(i,j)})_{i,j}$ and send it to $\mathcal{A}$.

**Secret Key Queries, Corruption Queries, Guess Phase.** Same as $\mathbf{Hyb}^{\mathcal{A}}_{4,2^\chi+1}(1^\lambda)$ except use $\mathsf{FSS.Sim}$ for the output of FSS evaluation on $\widetilde{C}_{i^*}$.

**Lemma J.1.** $\Pr[1 \leftarrow \mathbf{Hyb}^{\mathcal{A}}_1(1^\lambda)] \geq \Pr[1 \leftarrow \mathbf{Hyb}^{\mathcal{A}}_0(1^\lambda)]/\mathsf{poly}(\lambda)$.

*Proof.* In $\mathbf{Hyb}^{\mathcal{A}}_1(1^\lambda)$, we guess an $i^*$ such that each authority in $\mathcal{K}_{i^*}$ is corrupted. As $\mathcal{A}$ corrupts at most $k$ authorities, we can see that our guess is correct with at least $1/\binom{n}{k}$ probability. Hence, in $\mathbf{Hyb}^{\mathcal{A}}_1(1^\lambda)$ we do not abort with at least $1/\binom{n}{k} = 1/\mathsf{poly}(\lambda)$ probability. Thus, the lemma holds. □

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

*Proof.* In $\mathbf{Hyb}^{\mathcal{A}}_1(1^\lambda)$, we sample $r_{i^*j}$ for each $\mathsf{id}$ independently whereas in $\mathbf{Hyb}^{\mathcal{A}}_2(1^\lambda)$, we sample a random $\rho_{i^*j}$ and then set $r_{i^*j}$ for each authority as additive secret shares of $\rho_{i^*j}$. Both of these are identically distributed. In addition, we abort when each $\rho_{i^*j}$ is not unique. As this only happens with negligible probability, the lemma holds. □

**Lemma J.3.** Assuming the security of $\mathsf{SKE}$, $\mathbf{Hyb}^{\mathcal{A}}_2(1^\lambda)$ and $\mathbf{Hyb}^{\mathcal{A}}_3(1^\lambda)$ are computationally close.

*Proof.* In both hybrids $\mathsf{ske.ct}^*$ is not present in the view of $\mathcal{A}$. Thus, the lemma follows by security of SKE with pseudorandom ciphertexts. Otherwise, we can reduce a distinguishing adversary to the break the security of $\mathsf{SKE}$ similarly to Claim G.1. □

**Lemma J.4.** $\mathbf{Hyb}_3^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_{4,1}^{\mathcal{A}}(1^\lambda)$ are identically distributed.

*Proof.* As we are not encrypting $F'_{i^*j}$ for any $j$ in $\mathbf{Hyb}_{4,1}^{\mathcal{A}}(1^\lambda)$, these hybrids are identically distributed. □

**Lemma J.5.** Assuming the security of $\mathsf{MAFE}$, $\mathbf{Hyb}_{4,\kappa}^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_{4,\kappa+1}^{\mathcal{A}}(1^\lambda)$ for $\kappa \in [2^\chi]$ are computationally indistinguishable.

*Proof.* Assume that there exists an adversary $\mathcal{A}$ that can distinguish between $\mathbf{Hyb}_{4,\kappa}^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_{4,\kappa+1}^{\mathcal{A}}(1^\lambda)$ with non-negligible probability $\epsilon(\lambda)$, i.e,

$$\left|\Pr\left[1 \leftarrow \mathbf{Hyb}_{4,\kappa}^{\mathcal{A}}(1^\lambda)\right] - \Pr\left[1 \leftarrow \mathbf{Hyb}_{4,\kappa+1}^{\mathcal{A}}(1^\lambda)\right]\right| > \epsilon(\lambda)$$

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

**Setup, Secret Key Queries, Corruption Queries.** Same as $\mathbf{Hyb}_{4,\kappa}^{\mathcal{A}}(1^\lambda)$ except when generating secret key for $(i^*, \kappa)$-th instantiation, ask $\mathcal{O}$ for a secret key for the corresponding input.

**Challenge Query.** $\mathcal{A}$ sends circuit $C_0, C_1$. Sample $\{\widetilde{C}_i\}_{i\in\mathbb{I}'} \leftarrow \mathsf{FSS.Share}(1^\lambda, 1^{|\mathbb{I}'|}, C_b)$. For each $i \in \mathbb{I}', j \in \mathbb{J}$, if $i = i^*$, compute $y_j = \mathsf{FSS.Eval}(i^*, \widetilde{C}_{i^*}, \{X, j\})$ for each $(\mathsf{GID}, X)$ of the pre-challenge queries where for this $\mathsf{GID}$, all $n - k$ authorities are queried and sample

- If $j < \kappa$, $\mathsf{ct}^{(i,j)} \leftarrow \mathsf{MAFE.Enc}(\{\mathsf{mpk}_{\mathsf{id}}^{(i,j)}\}_{\mathsf{id}\notin\mathcal{K}_i}, F'_{i^*j})$.
- Otherwise, if $j > \kappa$, sample $\mathsf{ct}^{(i,j)} \leftarrow \mathsf{MAFE.Enc}(\{\mathsf{mpk}_{\mathsf{id}}^{(i,j)}\}_{\mathsf{id}\notin\mathcal{K}_i}, F_{ij})$.
- Otherwise, $\mathsf{ct}^{(i,j)} \leftarrow \mathcal{O}(F_{ij}, F'_{i^*j})$.

Set $\mathsf{CT} = (\mathsf{ct}^{(i,j)})_{i,j}$ and send it to $\mathcal{A}$.

**Secret Key Queries, Corruption Queries, Guess Phase.** Same as $\mathbf{Hyb}_{4,\kappa}^{\mathcal{A}}(1^\lambda)$ except when generating secret key for $(i^*, \kappa)$-th instantiation, ask $\mathcal{O}$ for a secret key for the corresponding input.

As we can see, running time of $\mathcal{B}$ is polynomial in the running time of $\mathcal{A}$ and $\lambda$. If $\mathcal{O}$ is encrypts $F'_{i^*j}$, $\mathcal{B}$ behaves like $\mathbf{Hyb}_{4,\kappa+1}^{\mathcal{A}}(1^\lambda)$ and if $\mathcal{O}$ encrypts $F_{i^*j}$, $\mathcal{B}$ behaves like $\mathbf{Hyb}_{4,\kappa}^{\mathcal{A}}(1^\lambda)$. In addition, as long as $\mathcal{B}$ does not abort, it is a successful reduction adversary that can break the security of $\mathsf{MAFE}$ with non-negligible probability. □

**Lemma J.6.** Assuming the security of $\mathsf{FSS}$, $\mathbf{Hyb}_{4,2^\chi+1}^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_5^{\mathcal{A}}(1^\lambda)$ are computationally indistinguishable.

*Proof.* As the only difference between these hybrids is that we are simulating FSS using output values, these two hybrids can be argued to be computationally indistinguishable similarly to Lemma C.3. □