Dynamic Collusion Bounded Functional Encryption from Identity-Based Encryption

Rachit Garg
UT Austin*

Rishab Goyal
MIT†

George Lu
UT Austin‡

Brent Waters
UT Austin and NTT Research§

Abstract

Functional Encryption is a powerful notion of encryption in which each decryption key is associated with a function f such that decryption recovers the function evaluation $f(m)$. Informally, security states that a user with access to function keys $\mathsf{sk}_{f_1}, \mathsf{sk}_{f_2}, \ldots$ (and so on) can only learn $f_1(m), f_2(m), \ldots$ (and so on) but nothing more about the message. The system is said to be q-bounded collusion resistant if the security holds as long as an adversary gets access to at most $q = q(\lambda)$ function keys. A major drawback of such *statically* bounded collusion systems is that the collusion bound q must be declared at setup time and is fixed for the entire lifetime of the system.

We initiate the study of *dynamically* bounded collusion resistant functional encryption systems which provide more flexibility in terms of selecting the collusion bound, while reaping the benefits of statically bounded collusion FE systems (such as quantum resistance, simulation security, and general assumptions). Briefly, the virtues of a dynamically bounded scheme can be summarized as:

Fine-grained individualized selection. It lets each encryptor select the collusion bound by weighing the trade-off between performance overhead and amount of collusion resilience.

Evolving encryption strategies. Since the system is no longer tied to a single collusion bound, thus it allows to dynamically adjust the desired collusion resilience based on any number of evolving factors such as the age of the system, or number of active users etc.

Ease and simplicity of updatability. None of the system parameters have to be updated when adjusting the collusion bound. That is, the same key sk_f can be used to decrypt ciphertexts for collusion bound $q = 2$ as well as $q = 2^\lambda$.

We construct such a dynamically bounded functional encryption scheme for the class of all polynomial-size circuits under the general assumption of Identity-Based Encryption.

*Email: rachg96@cs.utexas.edu.

†Email: goyal@utexas.edu. Research supported in part by NSF CNS Award #1718161, an IBM-MIT grant, and by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR00112020023. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Government or DARPA.

‡Email: gclu@cs.utexas.edu.

§Email: bwaters@cs.utexas.edu. Supported by NSF CNS-1908611, CNS-1414082, Packard Foundation Fellowship, and Simons Investigator Award.

1 Introduction

Public-key encryption [DH76] is one of the most fundamental concepts in cryptography. Traditionally, public-key encryption was defined to provide an "all-or-nothing" type functionality and security, where given a decryption key sk, a user can either recover the entire plaintext m from a ciphertext ct or nothing at all. In the recent years, an extremely powerful notion of encryption called Functional Encryption (FE) [SW05, BSW11] has emerged.

FE provides a fine-grained access control mechanism over encrypted data where a decryption key is now associated with a function f and the decryptor recovers the function evaluation $f(m)$ from the ciphertext. Moreover, a user with access to function keys $\mathsf{sk}_{f_1}, \ldots, \mathsf{sk}_{f_n}$ can only learn $f_1(m), \ldots, f_n(m)$ but nothing more about the message. This security requirement is commonly captured in a game based indistinguishability definition, where the adversary submits two messages, m_0 and m_1, as a challenge and must be unable to distinguish between encryptions of m_0 and m_1 with non-negligible probability given that $f_i(m_0) = f_i(m_1)$ hold for all keys in adversary's possession.

Over the last several years, FE has been studied extensively. Significant progress has been made towards building various expressive forms of FE under such indistinguishability-based definitions. Starting with initial works [BW07, KSW08] that built specific forms of predicate encryption over bilinear maps, the search for FE for general circuits under standard cryptographic assumptions culminated in the recent breakthrough work of Jain, Lin, and Sahai [JLS21]. They proposed an FE scheme for general circuits from a combination of PRGs in $\mathbf{NC}^0$, Symmetric eXternal Diffie-Hellman (SXDH), Learning with Errors (LWE), and Learning Parity with Noise (LPN) over large fields assumptions. While this is tremendous progress, an unfortunate limitation of this FE scheme is that it is susceptible to quantum attacks due to the post-quantum insecurity of the SXDH assumption. But even more broadly, pursuing the direction of indistinguishability-based security for FE suffers from the drawback that it is unclear how it captures the intuition that an attacker learns *at most the function evaluation but nothing more*.

For these reasons, FE has also been investigated in the bounded collusion model under simulation-based definitions. In the bounded collusion model, the FE system declares a bound q at the setup time, such that all the system parameters are allowed to grow polynomially with q (in addition to the security parameter λ). Additionally, the security requirement is captured via a simulation-based game, which says that as long as the attacker does not make more than q key queries, the adversary's view - which includes the ciphertext ct_m and function keys $\mathsf{sk}_{f_1}, \ldots, \mathsf{sk}_{f_q}$ - can be "simulated" given only the function evaluations $f_1(m), \ldots, f_q(m)$ and nothing more about m. Although this more closely captures the intuition behind FE, if the attacker corrupts more than q keys, then no security is provided. Despite its limitations, the bounded collusion model for FE has been very useful in various contexts such as proving negative results in differential privacy [KMUW18], applications to tracing [GKW18, CVW+18], etc. In some cases, it is the only currently known pathway to certain applications in the post-quantum regime. A notable feature of the bounded collusion model is that under them, FE can be built from the minimal assumption of public-key encryption (and OWFs in case of private-key FE) as studied in a long line of works [SS10, GVW12, AR17, Agr17, GKW18, CVW+18, AV19].

The question. A major drawback of such bounded collusion FE systems is that the setup authority needs to declare the collusion bound q at the very beginning, and the bound q is fixed, once and for all, for the entire lifetime of the system. This puts the authority in a difficult situation, as it

requires an incredible amount of foresight at the setup time. In particular, if the authority sets the bound q lower than the eventual number of compromised keys, then the system will be insecure; whereas overestimating the bound q would result in significant performance overhead. Now when the collusion bound is breached, the only option to would to do a fresh setup and redistribute the keys which is at best inefficient, and possibly infeasible in certain scenarios. Switching to the state-of-the-art fully collusion resistant FE schemes would suffer from drawbacks discussed above.

With the aforementioned limitations of existing FE systems, we ask the following –

> *Can we build an FE system for general circuits that reaps the benefits of bounded collusion FE systems – post-quantum security, simulation security, and general assumptions – while at the same time provide more flexibility to the authority in terms of selecting the collusion bound? And, would such an FE system lead to results in the domain of full collusion resistance?*

In this work, we study the above question. We answer the first part in affirmative by introducing a new flexible corruption model that we call the "dynamic collusion" model, and building a simulation secure FE system in the dynamic collusion model from the general assumption of Identity-Based Encryption (IBE) [Sha85, Coc01, BF01] (for which we have quantum-safe instantiations [GPV08, CHKP10, ABB10]). Since it is widely believed that the FE for general circuits is significantly more expressive than plain IBE, this seems to answer the latter part negatively. Next, we define our dynamic collusion model and provide a high level overview of our techniques.

Defining Dynamically Bounded Collusion Resistance

In this work, we refer to the traditional notion of bounded collusion resistance for FE as *statically bounded collusion resistance*. Recall that, syntactically, a statically bounded FE is defined exactly as fully collusion resistant FE, that is using four polynomial time algorithms – Setup, KeyGen, Enc, and Dec – except the Setup algorithm now additionally takes the target collusion bound q as an input. As mentioned previously, declaring the collusion bound q upfront lets the setup authority set up the system parameters with enough redundancy, and this typically leads to the running time and sizes of all system parameters (i.e., the keys and ciphertexts) to grow polynomially with q.

In the dynamic collusion model, the Setup algorithm no longer takes the collusion bound as input, but instead the Enc algorithm selects the collusion bound per ciphertext. That is, the setup and key generation algorithms no longer depend on the collusion bound q, but only the encryptor needs to specify the collusion bound.[1] *Basically, this lets the encryptor dynamically decide the size of set of colluding users against which it wants to hide its message.* As a consequence, in the dynamic collusion model, only the size of the ciphertexts potentially grows with the collusion bound q, but the running times of the Setup and KeyGen algorithms (therefore the public and secret keys) are independent of q. The security requirement is again captured via a simulation-based game but where the admissibility constraints on the attacker are lifted such that the number of key queries the attacker is permitted can be adaptively specified at the time of committing the challenge m instead of beginning of the game as in the static model.

Our dynamic collusion model and its comparison with the static model is discussed in detail in Section 3. Below we briefly highlight the virtues of the dynamic collusion model.

[1] However, note that it is essential that the master public-secret keys and every function key is resuable for all values of the collusion bound.

Fine-grained individualized selection. A dynamically bounded collusion FE scheme allows each user to select the collusion bound by weighing the trade-off between the performance overhead and amount of collusion resilience at encryption time. For example, depending upon the factors such as available computing resources, or the bandwidth on the communication channel, or the sensitivity of data etc, an encryptor might want to increase/decrease the amount of collusion resilience to better fit the computing/communication/privacy constraints.

Evolving encryption strategies. Since the system is no longer statically tied to a single collusion bound at setup time, thus it allows to dynamically adjust the desired collusion resilience based on any number of evolving factors such as the age of the system, or number of active users etc. Thus, the authority does not need to have any foresight about the attackers at setup time in contrast to statically bounded collusion FE systems.

(Of course the ciphertexts in which the collusion bound was exceeded will not be secure, but future attacks can be prevented by adapting to a larger collusion bound.)

Ease and simplicity of updatability. While the above features are already highly desirable, a noteworthy property of these systems is that none of the parameters have to be updated when adjusting the collusion bound. That is, the same function key sk_f can be used to decrypt ciphertexts for collusion bound $q = 2$ as well as $q = 2^\lambda$ without requiring any updates. Also, the storage space for the parameters is bounded by a fixed polynomial in λ.

Next, we provide an overview of our approach and describe the technical ideas. Later on, we discuss some related works and open questions.

1.1 Technical Overview

In this section, we provide a high level overview of our framework and FE construction. The overview is split into four parts. First, we informally define the notion of dynamically bounded collusion resistant FE for general circuits. Second, we define an efficiency property that we refer to as *weak optimality* for statically bounded collusion FE systems, and show how this could be generically lifted to a dynamically bounded collusion FE scheme. Therefore, to get our final result we simply need to construct a statically bounded collusion FE scheme from IBE that satisfies the weak optimality property. In the third part, we discuss how to use ideas from the bounded collusion FE construction for $\mathbf{NC}^1$ circuits by Gorbunov-Vaikuntanathan-Wee [GVW12], and construct a weakly optimal statically bounded collusion resistant FE scheme. Combining this with our abstraction, this results in a dynamically bounded collusion resistant FE for $\mathbf{NC}^1$ circuits under the assumption of IBE. However, to generically bootstrap this to a (dynamically bounded) FE scheme for the class of all polynomial sized circuits, we need to rely on PRFs computable in $\mathbf{NC}^1$ as in [GVW12]. In the final part, we show how to avoid making the additional assumption regarding low-depth computable PRFs by using ideas from the statically bounded FE construction from [AV19]. We describe that by abstracting out the main ideas from the third part, we could use similar ideas to extend [AV19] to get the desired dynamically bounded collusion resistant FE for all polynomial sized circuits.

Dynamic vs. Static Bounded Collusion Model

Let us start by recalling the syntax of functional encryption in the static collusion model. An FE scheme in the static collusion model consists of four algorithms with the following semantics:

– Setup takes as input the collusion bound q and samples the master public-secret key pair $(\mathsf{mpk}, \mathsf{msk})$.

– KeyGen generates a function key sk_f given function f and master key msk.

– Enc encrypts a message m to a ciphertext ct.

– Dec recovers $f(m)$ from the ciphertext and decryption key.

In the dynamic collusion model, the collusion bound q is not fixed at the system setup, but instead the encryptor chooses the amount of collusion resilience it wants every time a fresh ciphertext is created. This is reflected with the following changes:

– Setup no longer takes the collusion bound q as an input.

– Enc takes the desired collusion bound q as an additional input for sampling the ciphertext.

Note that since the collusion bound q is not specified during setup or key generation at all, thus the efficiency condition for a dynamically bounded collusion FE scheme requires the running time of Setup and KeyGen to be fixed polynomials in λ, whereas in static setting they are allowed to grow polynomially with q.

Dynamic to Static via Weak Optimality

As we mentioned before, our first observation is that a dynamically bounded collusion FE scheme can be constructed from any statically bounded scheme if it satisfies the 'weak optimality' property. Intuitively, the weak optimality property says that the running time of the setup and key generation algorithms grows only poly-logarithmically in the collusion bound q.

Now looking closely at the notion of weakly-optimal statically bounded collusion FE, we observe that the major difference between this and a dynamic system is that the Setup algorithm requires q as an explicit input in the static setting, but not in the dynamic setting. Our idea to get around this is to exploit the efficiency property of the static scheme, where the dynamic collusion FE scheme essentially runs λ independent instances of the static collusion FE scheme in parallel with geometrically increasing collusion bounds. That is, i-th subsystem (running a single instance of the static scheme) is set up with collusion bound $q_i = 2^i$. And, now the master public-secret key pair as well as each function key in the dynamic system contains λ independently sampled keys where the i-th sub-key is sampled using the i-th static FE system. Since the encryption algorithm receives the target collusion bound q as input, thus the encryptor uniquely selects a static FE sub-system under which it encrypts the message. The target collusion bound to subsystem index mapping can simply be defined $i := \lceil \log q \rceil$ (i.e., nearest power of two). Note that setting up the system this way ensures the dynamic system achieves the desired efficiency. This is because the setup and key generation will be efficient (by weak optimality of the static FE scheme), and since $2^i = 2^{\lceil \log q \rceil} < 2q$, thus the running time of encryption and decryption is a polynomial in q.

Since the above transformation is very natural, one would expect the simulation security of the resulting dynamic FE system to also follow directly from the simulation security of the underlying static FE schemes. However, this is not the case. To better understand the technical barrier, let us first consider the most natural simulation strategy described next. The simulator for the dynamic system simply runs the simulator for each of the underlying static systems in parallel, where the ciphertext simulator is only run for the static system corresponding to the adversarially selected

challenge target collusion bound q^*. While this seems to compile, there are two subtle issues that need to be carefully handled.

First, the running time of each static FE simulator grows with the underlying collusion bound which grows as large as exponential in λ. For avoiding the problem of inefficient simulation, we additionally require the underlying static FE scheme to have weakly-optimal simulators as well which means that all but the ciphertext simulation phase of the static FE could be performed optimally (i.e., the simulator running time grows only poly-logarithmically in q). However, this is still not enough for proving simulation security. The reason is that typically the simulation security states that the distribution of secret keys and ciphertext are simulatable as long as the adversary does not make more key queries than what is specified by the static collusion bound. That is, if the adversary makes more key queries then no guarantee is provided. Now our dynamic FE simulator must invoke the underlying static FE simulator even for collusion bounds smaller than q^*, thus the standard simulation guarantee is insufficient. To get around this issue, we define a notion called *strong* simulation security for static-bounded-collusion FE schemes under which we require that the real and ideal worlds are also indistinguishable even when the adversary makes more key queries than that specified by the collusion bound as long as the adversary does not make any challenge message queries. More details are provided in Section 3.2.

Weakly Optimal Static FE for $\mathbf{NC}^1$ from IBE via [GVW12]

The main ingredients for our construction are an identity-based encryption scheme IBE and garbled circuits GC. Our construction is inspired from the q-bounded collusion resistant FE scheme for $\mathbf{NC}^1$ by Gorbunov, Vaikuntanathan, and Wee (GVW) [GVW12]. At a high level, their approach was to provide a general bootstrapping theorem that upgrades any 1-bounded collusion resistant FE scheme $\mathsf{1KeyFE}$ (such as [SS10]) to a q-bounded collusion resistant FE scheme for $\mathbf{NC}^1$ unconditionally. Let D be the maximum degree of the circuits in the family. Their transformation executes a non-interactive BGW-style multiparty computation protocol [BGW88] in the head, i.e. pieces of the ciphertext serve as analogues to the parties. More concretely,

- Setup samples N independent master public-secret key pair $(\mathsf{mpk}_i, \mathsf{msk}_i)$ for the $\mathsf{1KeyFE}$ scheme. These N key pairs are set as the master public-secret key pairs for this scheme respectively.

- KeyGen associates the decryption key with random subset/pieces $\Gamma \subseteq [\mathsf{N}]$ of size $\mathsf{p} = \mathsf{Dt} + 1$ (t is a some fixed polynomial in the security parameter). The secret keys are a collection of msk_i for $i \in \Gamma$.

- Enc chooses ℓ random degree t polynomials $\mu_1, \ldots, \mu_\ell$ with constant terms $x_1, \ldots, x_\ell$.[2] It computes ct_i as the $\mathsf{1KeyFE}$ encryption of $(\mu_1(i), \ldots, \mu_\ell(i))$ for $i \in [\mathsf{N}]$. It outputs $(\mathsf{ct}_1, \ldots, \mathsf{ct}_\mathsf{N})$.

- Dec computes p evaluations of the polynomial $P(\cdot) = C(\mu_1(\cdot), \ldots, \mu_\ell(\cdot))$. Note that P is a degree $\mathsf{p} - 1 = \mathsf{Dt}$ polynomial (as degree of C and μ is D and degree t, respectively) with constant term $C(x)$. Since we have msk_i for $i \in \Gamma$, we can decrypt ct_i to learn $P(i)$. As we have p evaluations for a polynomial with degree $\mathsf{p}-1$, we can interpolate and compute $P(0) = C(x)$.

The correctness of the scheme follows from the correctness of the $\mathsf{1KeyFE}$ scheme. The main observation is that the security holds for a q collusion of keys if a combinatorial property on the

[2] Here and throughout, $x = (x_1, \ldots, x_\ell)$ corresponds to the message being encrypted.

set Γ is satisfied. Let $\Gamma_j \subseteq [\mathsf{N}]$ be the sampled subset when j^{th} query is made. Whenever two sets $\Gamma_j, \Gamma_{j'}$ intersect, we learn two keys for the underlying $\mathsf{1KeyFE}$ scheme thereby breaching its security and adversary can learn the corresponding plaintexts. Observe that if the security is broken for enough $\mathsf{1KeyFE}$ copies, i.e. $> \mathsf{t}$, then we can decrypt similar to an honest party. Thus we need the combinatorial property that the two sets have small pairwise intersection. Additionally note that revealing the entire polynomial P can reveal additional information about x as the hiding of circuit C is not guaranteed in our scheme. Thus we need to randomize the polynomial P for every keygen query. This is done by sampling a cover free set $\Delta \subseteq [\mathsf{N}']$ that has at least one cover free element for every query. The scheme now computes $\mathsf{1KeyFE}$ keygen's corresponding to the circuit,

$$G_{C,\Delta}(x, Z_1, \ldots, Z_{\mathsf{N}'}) = C(x) + \sum_{i \in \Delta} Z_i$$

where $Z_1, \ldots, Z_{\mathsf{N}'}$ are used to randomize P and $|\Delta| = \mathsf{p}'$.

For upgrading the GVW construction to achieve weak optimality, we face the following challenges.

- To satisfy small pairwise intersection property, N needs to be a polynomial function of q. This makes Setup run in $\mathsf{poly}(\mathsf{q})$ time, rather than $\mathsf{poly}(\log \mathsf{q})$.
- In GVW, each decryption key contains p many secret keys for the single key system where p is polynomially dependent on q. Thus KeyGen also runs in time $\mathsf{poly}(\mathsf{q})$.
- Additionally, the circuit used by the 1-bounded FE system is $G_{C,\Delta}$ takes inputs size $\ell + \mathsf{N}'$. To satisfy the cover free property, N' needs to be a polynomial function of q, therefore the size of circuit $G_{C,\Delta}$ also grows polynomially with q. Thus, each invocation of the KeyGen for the $\mathsf{1KeyFE}$ scheme grows polynomially with q.

Broadly, we solve the above problems by opening up the underlying 1-bounded FE scheme $\mathsf{1KeyFE}$, relying on better combinatorial techniques, and leveraging IBE for obtaining necessary compression in the key space.

- To tackle the first issue, we rely on an IBE scheme to compress public keys. Since $\mathsf{1KeyFE}$ scheme outputs $\mathsf{poly}(\mathsf{q})$ many public keys and secret keys, an IBE system can be used to compress the $\mathsf{1KeyFE}$ public keys (which contain $\mathsf{poly}(\mathsf{q})$ many PKE public keys) to a single IBE public key which can be generated in time $\mathsf{poly}(\log(\mathsf{q}))$ and make it weakly optimal.
- We observe that by increasing N by only a polynomial factor, we can reduce the number p of secret keys required to recover to be dependent only on κ (not q) while still maintaining the same combinatorial properties. This relies on better concentration bounds.
- Lastly, observe that to randomize the polynomials, we only need to add a size p' subset of values in the GVW construction. By opening up the $\mathsf{1KeyFE}$ construction, we can decompose and individually encrypt garbled labels to reduce the circuit size to be proportional to p' rather than N', additionally using IBE to perform the selection of labels corresponding to $\{Z_i\}_{i \in \Delta}$. Concretely, we split $G_{C,\Delta}$ into two sub-circuits G_C and Sel_Δ with the functionality that

$$\mathsf{Sel}_\Delta(Z_1, \ldots, Z_{\mathsf{N}'}) = (Z_{\delta_1}, \ldots, Z_{\delta_{\mathsf{p}'}}), \qquad G_C(x, Y_1, \ldots, Y_{\mathsf{p}'}) = C(x) + \sum_{i \in [\mathsf{p}']} Y_i,$$

where $\Delta = \{\delta_1, \ldots, \delta_{\mathsf{p}'}\}$. Note that $G_{C,\Delta}(x, Z_1, \ldots, Z_{\mathsf{N}'}) = G_C(x, \mathsf{Sel}_\Delta(Z_1, \ldots, Z_{\mathsf{N}'}))$, that is it is a composition of the two circuits. Basically, our idea is to use garbled circuits to encode G_C whereas rely on IBE to encode the Sel_Δ selector circuit, and this solves the circuit size problem.

Formally, our construction does the following:

- Setup initializes two IBE schemes, one with space $[\mathsf{N}] \times [\mathsf{N}']$ and one with space $[\mathsf{N}] \times ([|C|] \times \{0,1\})$.
- KeyGen selects a subset $\mathsf{taglist}^1 \subseteq [\mathsf{N}]$, as well as a subset $\mathsf{taglist}^2 \subseteq [\mathsf{N}']$. For every element $\mathsf{tag}^1 \in \mathsf{taglist}^1$, $\mathsf{tag}^2 \in \mathsf{taglist}^2$, and bit C_i in the function circuit, give an IBE secret key on identity $(\mathsf{tag}^1, \mathsf{tag}^2)$ in the first scheme and an IBE secret key on identity $(\mathsf{tag}^1, (i, C_i))$ in the second scheme.
- Enc splits input x into ℓ random degree t polynomials $\mu_1, \ldots, \mu_\ell$ with constant terms $x_1, \ldots, x_\ell$ as before. Additionally, it samples N' random polynomials $\zeta_1, \ldots, \zeta_{\mathsf{N}'}$ of degree $\mathsf{p} - 1$ with constant term 0.

  For each $\mathsf{tag}^1 \in [\mathsf{N}]$, sample a unique field element e and compute a garbling of circuit $\widetilde{\mathcal{U}}^{\mathsf{tag}^1}$ of the circuit $\mathcal{U}[\mu_1(e), \ldots, \mu_\ell(e)](C, Z_1, \ldots, Z_{\mathsf{p}'}) = C(\mu_1(e), \ldots, \mu_\ell(e)) + \sum_{i=1}^{\mathsf{p}'} Z_i$ where $\mu_1(e), \ldots, \mu_\ell(e)$ are hardcoded inside the circuit. We then use the first IBE scheme to encrypt the labels for a share of the randomizing polynomials ζ and the second IBE scheme to encrypt the labels corresponding to the circuit wires.
- Dec computes p evaluations of the polynomial $\eta(\cdot) = C(\mu_1(\cdot), \ldots, \mu_\ell(\cdot)) + \sum_{i=1}^{\mathsf{p}'} \zeta_{\mathsf{tag}^2{}_i}(\cdot)$. Note that P is a degree $\mathsf{p} - 1 = \mathsf{Dt}$ polynomial as degree of C is D and degree of μ is t and constant term $C(x)$.

  Now using identities given by $\mathsf{taglist}^1, \mathsf{taglist}^2, \mathsf{taglist}^3$ and given IBE secret keys to reveal the input labels corresponding to positions for C and positions for the randomizing polynomial Z_i. Since we can decrypt to learn $\eta(i)$. As we have p evaluations for a polynomial with degree $\mathsf{p} - 1$, we can interpolate and compute the entire polynomial η and hence $\eta(0) = C(x)$.

Our weakly optimal FE construction is described in detail later in Section 4. Below we sketch main ideas behind the proof which relies on a careful case analysis.

Case 1: For tag^1 where the pairwise intersection fails, observe that there exist at least two decryption key queries where secret keys for the same IBE identity might be issued. Thus we cannot rely on IBE security for these queries and assume that the IBE plaintext is revealed.

Case 2: For tag^1 where the pairwise intersection holds and there exists a query j^* that chose tag^1, observe that here we only rely on security for identities $(\mathsf{tag}^1, \mathsf{tag}^2)$ where $\mathsf{tag}^2 \notin \mathsf{taglist}^{2,(j^*)}$ or identities $(\mathsf{tag}^1, \mathsf{tag}^3)$ where $\mathsf{tag}^3 \notin \mathsf{taglist}^{3,(j^*)}$. Since the query j^* was made on C_{j^*} and these sets of identities revealed, we want to avoid relying on security of these identities. The labels corresponding to these identities do not reveal any information apart from $C_{j^*}(x)$ due to the simulatability of garled circuits.

Case 3: For tag^1 where the pairwise intersection holds and there exists no prior query, we can rely on IBE security as know no identity is revealed in any of the prior queries. Thus all the

labels of the garbled circuit are computationally hidden to an adversary and no information is revealed.

If the small set intersection property succeeds then $\leq \mathsf{p} - 1$ evaluations are revealed for polynomial η and the constant term is hidden. More details are provided in Section 4.1.

Improving $\mathbf{NC}^1$ to $\mathsf{P/poly}$ via [AV19]

Although the above construction is restricted to the function class $\mathbf{NC}^1$, it can be transformed into a static bounded collusion FE scheme for $\mathsf{P/poly}$ using the generic 'bootstrapping' transformation presented in [GVW12], which maintains the weak optimality we require. The idea here is for the $\mathbf{NC}^1$ FE scheme to output keys which compute a specific type of garbled circuit of C and the corresponding input labels for its input x rather than directly outputting x itself. The garbled circuit and its labels can then be evaluated in the clear. Thanks to the work of [AIK06], we know there exists garbled circuits (for $\mathsf{P/poly}$) which can be computed by constant depth circuits with respect to the input x and randomness used. Unfortunately, however, this result makes use of an additional assumption of length-doubling PRGs in $\mathbf{NC}^1$, which is not known to implied by IBE.

To achieve weakly optimal static bounded collusion FE for $\mathsf{P/poly}$ without such an additional assumption, we apply our techniques to the static bounded collusion FE for $\mathsf{P/poly}$ scheme of [AV19], which can be built from only public key encryption (and implied primitives, including plain PRGs and garbled circuits). The construction of [AV19] follows a similar structure to the 'bootstrapped' GVW construction described above. However, the key insight made here is that rather than relying on the function secret keys to evaluate the PRG needed to compute these specialized garbled circuits, the encryptor can simply evaluate the PRG seed randomness required and include said PRG outputs in the FE ciphertext. Since the PRG evaluations are now being done by the party generating encryptions rather than the functional secret keys themselves, this can be implemented using any (not just $\mathbf{NC}^1$) pseudorandom generators, which are implied by IBE.

However, this modification on [GVW12] introduces additional technical difficulties when attempting to achieve weak optimatity. Whereas previously, the scheme generated 1-bounded functional keys for the circuit $G_{C,\Delta}(x, Z_1, \ldots, Z_S) = C(x) + \sum_{i \in \Delta} Z_i$, now the scheme generates functional keys for a much more complex circuit which computes a garbling of C on x. More specifically, the 1-bounded functional keys now need to compute a $\mathsf{CorrGarb}$ functionality which computes a garbled circuit of the input circuit and labels. However, much like in the $G_{C,\Delta}$ circuit, the only reason this circuit grows polynomially with the collusion bound is the necessity for seperate parties to receive garbled circuits generated under what appears to be independent randomness derived from the same ciphertext. This is done by independently selecting a random subsets of the input randomness for each key given out. IBE can be utilized here in a similar way as we did for $G_{C,\Delta}$ — that is, by using IBE secret keys to moderate which subset of garbled labels are accessible, we can ensure that the garbled circuit scales *only* with the size of the subset of randomness, rather than the entire input randomness. Since increasing the input randomness is now effectively 'free', we can increase the input randomness to reduce the size of the subset required to be independent of the collusion bound. Our scheme is decribed in detail later in Section 5.

Lastly, we remark that currently we prove the security of our constructions against non-adaptive adversaries. Recall that a non-adaptive adversary is prohibited from making any key queries in the post-challenge phase (that is, after receiving the challenge ciphertext). We do not consider this to be a limitation of our techniques, and believe that our constructions can also be made adaptively

secure by relying on simple ideas as in [GVW12, AV19]. We leave further analysis for the full version.

1.2 Related Work and Future Directions

Prior work on bounded collusion resistance. The intial works on bounded collusion resistance for FE were for the specific class of IBE systems. Dodis et al. [DKXY02] and Goldwasser, Lewko, and Wilson [GLW12] constructed bounded collusion secure IBE with varying parametere size from regular public-key encryption and special types of linearly key homomorphic public-key encryption, respectively. For more expressive classes of FE, Sahai and Seyalioglu [SS10] proposed general functional encryption schemes resilient against a single function-key query using garbled circuits [Yao86]. Following [SS10], GVW [GVW12] build a statically bounded collusion resistant FE scheme for $\mathbf{NC}^1$ circuits from any public-key encryption scheme, and also provided a generic compiler to improve to the class of all polynomial time computable functions by additionally relying on PRFs computable in $\mathbf{NC}^1$. Afterwards, a number of follow-up works [AR17, Agr17, GKW18, CVW+18] improved the concrete efficiency of the statically bounded collusion resistant FE scheme wherein they improved the dependence of the FE scheme parameters on the collusion bound q by relying on more structured algebraic assumptions. Most recently, Ananth and Vaikuntanathan [AV19] achieved optimally efficient statically secure FE scheme from the minimal assumption of public-key encryption. The optimal efficiency states that the system parameters grow only linearly with the collusion bound q, since any further improvement would lead to a fully collusion resistant FE scheme via the bootstrapping theorems from [GGH+13, SW14, AJ15, BV15, AJS15].

Comparison with bundling functionalities and encrypt ahead FE. Goyal, Koppula, and Waters (GKW) [GKW16] proposed the concept of bundling functionalities in FE systems, where bundling functionalities in an FE scheme meant having the property that a single set of public parameters can support the union of all message/function spaces supported by the underlying FE system. They provided a generic transformation that started with IBE (and other implied primitives) and was able to upgrade any FE scheme to its bundled counterpart. One might ask that whether applying the [GKW16] transformation to the family of bounded collusion FE, where the the collusion bound q is treated as part of the functionality index that is bundled, already leads to a dynamically bounded collusion FE system. It turns out this is not the case because such a generic transformation suffers from the limitation that a function key for a given collusion bound is not reusable for other collusion bounds. In particular, this necessitates each user to make additional queries to the authority for obtaining function keys for desired collusion bound, and this only solves the problem of removing the problem of removing the collusion bound dependence for the setup algorithm. Additionally, GKW proposed a novel variant of FE called encrypt ahead FE. One could ask the same question about relationship between encrypt ahead FE and dynamically bounded collusion resistant FE, and the answer is the same as for the case of bundling functionalities which is they are insufficient.

Open questions. Our work introduces a new interesting avenue for exploring dynamic collusion resilience in FE systems. And, we provide positive results under the general assumption of IBE. Although we know that public-key encryption is both the necessary and sufficient assumption for building optimally efficient *statically* bounded collusion resistant FE systems (due to the work of

Ananth and Vaikuntanathan [AV19]), it is unclear whether public-key encryption is also sufficient for building *dynamically* bounded collusion resistant FE systems, or whether IBE is a necessary assumption for building them. Another interesting research direction is studying similar concepts of dynamic "query" resilience in other cryptographic contexts. For example, one could ask the same question for the concept of CCA-secure encryption where we know that CPA-secure public-key encryption implies (statically-)bounded-query-CCA security for public-key encryption [CHH+07]. We believe answering the question of dynamically bounded-query-CCA security might provide more insight in resolving the longstanding open problem of constructing a (general) CCA-secure encryption scheme from a CPA-secure one.

1.3 Concurrent Work

In a concurrent and independent work, Agrawal et al. [AMVY21] also achieved similar results as this work with some differences in the presentation and abstractions. They introduced the concept of dynamic collusion bounded functional encryption as we do to provide more flexibility for selecting the collusion bound. And, similar to us, they construct such functional encryption schemes for all polynomial-sized circuits from identity-based encryption. In addition, they also extend their results to uniform computation models while relying on specific algebraic assumptions.

2 Preliminaries

Notations. Let PPT denote probabilistic polynomial-time. For any integer $q \geq 2$, we let $\mathbb{Z}_q$ denote the ring of integers modulo q. We denote the set of all positive integers upto n as $[n] := \{1, \ldots, n\}$. For any finite set S, $x \leftarrow S$ denotes a uniformly random element x from the set S. Similarly, for any distribution $\mathcal{D}$, $x \leftarrow \mathcal{D}$ denotes an element x drawn from distribution $\mathcal{D}$. The distribution $\mathcal{D}^n$ is used to represent a distribution over vectors of n components, where each component is drawn independently from the distribution $\mathcal{D}$. Two distributions $\mathcal{D}_1$ and $\mathcal{D}_2$, parameterized by security parameter λ, are said to be computationally indistinguishable, represented by $\mathcal{D}_1 \approx_c \mathcal{D}_2$, if for all PPT adversaries $\mathcal{A}$, $|\Pr[\mathcal{A}(x) = 1 : x \leftarrow \mathcal{D}_1] - \Pr[\mathcal{A}(x) = 1 : x \leftarrow \mathcal{D}_2]| \leq \mathsf{negl}(\lambda)$.

2.1 Garbled Circuits

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

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

We define additional notation that will help us in our constructions.

Definition 2.1. Let $(\widetilde{C}, \mathcal{W} = \{w_{i,b}\}_{i\leq n, b\in\{0,1\}}) \leftarrow \mathsf{Garble}(1^\lambda, C \in \mathcal{C}_n)$. We will use $\mathcal{W}_{\{x_a,\ldots x_b\},\{\hat{x}_a,,\ldots\hat{x}_b\}}$ to denote the the subset of wires $\{w_{i,\hat{x}_i}\}_{i\in\{x_a,\ldots,x_b\}}$. Similarly, for $\mathcal{W}' = \{w_i\}_{i\leq n}$, i.e. a set of input garbled wires. We will use $\mathcal{W}'_{\{x_a,\ldots x_b\}} = \{w_i\}_{i\in\{x_a,\ldots,x_b\}}$ to denote the subset of input wires corresponding to these positions.

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

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

Definition 2.2. A garbling scheme $\mathsf{GC} = (\mathsf{Garble}, \mathsf{Eval})$ for a class of circuits $\mathcal{C} = \{\mathcal{C}_n\}_n$ is said to be a secure garbling scheme if there exists a polynomial-time simulator Sim such that for all n, $C \in \mathcal{C}_n$ and $x \in \{0,1\}^n$, the following distributions are computationally indistinguishable:

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

The following corollary follows from the definition.

Corollary 2.1. If GC is a secure garbling scheme for a class of circuits $\mathcal{C} = \{\mathcal{C}_n\}_n$, then for all n, $C \in \mathcal{C}_n$ and $x \in \{0,1\}^n$, the following distributions are computationally indistinguishable:

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

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

2.2 Identity-Based Encryption

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

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

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

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

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

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

Definition 2.3. We say an IBE scheme $\mathsf{IBE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is secure if for any stateful PPT adversary $\mathcal{A}$ there exists a negligible function $\mathsf{negl}(\cdot)$, such that for all $\lambda, n \in \mathbb{N}$, the following holds

$$\Pr\left[\mathcal{A}_1^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{st},\mathsf{ct}) = b : \begin{array}{c}(\mathsf{mpk},\mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n); \quad b \leftarrow \{0,1\} \\ (m_0, m_1, \mathsf{id}^*) \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(1^\lambda, 1^n, \mathsf{mpk}) \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{id}^*, m_b)\end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

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

3 Functional Encryption: Dynamic Bounded Collusion

In this section, we define the notion of functional encryption (FE) where we start by recalling the regime of (statically) bounded collusion secure FE systems as studied in prior works [SS10, GVW12]. We follow that by extending the notion to *dynamic* collusion bounded secure FE systems. And, along the way we also introduce a special compactness property for statically bounded collusion secure FE schemes. This will serve as an appropriate intermediate abstraction to build a fully dynamic collusion bounded FE schemes.

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

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

$\mathsf{Setup}(1^\lambda, 1^n) \to (\mathsf{mpk}, \mathsf{msk})$. The setup algorithm takes as input the security parameter λ and the functionality index n[3] (in unary), and outputs the master public-secret key pair $(\mathsf{mpk}, \mathsf{msk})$.

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

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

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

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

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

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

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

3.1 Bounded Collusion FE: Static and Dynamic

Informally, a functional encryption scheme is said to be secure if an adversary having secret keys for functions $\{f_i\}_{i\leq q}$ and a ciphertext ct for message m learns only $\{f_i(m)\}_{i\leq q}$, and nothing else about the underlying message m.

The Static Setting. Now in the "static" bounded collusion setting, the scheme is said to guarantee security so long as q is a polynomial in the security parameter λ and *fixed a-priori at the setup time*. Thus, the syntax of the setup algorithm changes as follows:

$\mathsf{Setup}(1^\lambda, 1^n, q) \to (\mathsf{mpk}, \mathsf{msk})$. The setup algorithm takes as input the security parameter λ and the functionality index n (in unary), and also takes as input the 'collusion bound' q (in binary).[4] It outputs the master public-secret key pair $(\mathsf{mpk}, \mathsf{msk})$.

Efficiency. Although the collusion bound q is given in binary to the setup algorithm, the efficiency condition for a statically bounded collusion FE scheme only requires that the running time of the all the algorithms is a fixed polynomial in λ, n and q. That is, the running time of Setup, KeyGen, Enc, and Dec is allowed to polynomially grow with the collusion bound q.

Static bounded collusion security. This is formally captured via the following 'simulation based' security definition as follows. We first provide the adaptive definition, and later provide the non-adaptive definition.

Definition 3.1 (static-bounded-collusion simulation-security)**.** A functional encryption scheme $\mathsf{FE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ is said to be statically-bounded-collusion simulation-secure if there exists a stateful PPT simulator $\mathsf{Sim} = (\mathsf{S}_0, \mathsf{S}_1, \mathsf{S}_2, \mathsf{S}_3)$ such that for every stateful PPT adversary $\mathcal{A}$, the following distributions are computationally indistinguishable:

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

$$\approx_c$$

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

whenever the following admissibility constraints and properties are satisfied:

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

[4] Although most prior works on bounded collusion security consider the collusion bound q to either be a global parameter, or given in unary to the setup algorithm. Here we instead pass it in binary for technical reasons as will become clear in the sequel. See Remark 3.1 for more details.

– $\mathcal{A}$ makes at most q queries combined to the key generation oracles in the corresponding games.

– S_3 for each queried function f_i, in the post-challenge phase, makes a single query to its message oracle U_m on the same f_i itself.

Remark 3.1 (unary vs binary)**.** Note that in the above security games, we require the adversary to specify the collusion bound q in unary at the beginning. This is in contrast to the setup algorithm which gets q in binary as an input. The reason for this distinction is that in the security game for bounded collusion security we do not want to allow the attacker to specify super-polynomial collusion bounds, whereas (as we point out later) allowing the setup algorithm to be run on super-polynomial values of the collusion bound is important for our dynamic collusion bounded FE schemes.

Weak optimality. Additionally, we also introduce the notion of a "weakly optimal" statically-bounded-collusion secure FE scheme where this system provides better efficiency properties. That is, in a weakly optimal static bounded collusion system, the running time of the setup and key generation algorithms grows only poly-logarithmically in the collusion bound q. Concretely, we define it below.

Definition 3.2 (weakly optimal statically-bounded-collusion)**.** A functional encryption scheme $\mathsf{FE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ is said to be 'weakly optimal' statically-bounded-collusion FE scheme if the running time of the Setup and KeyGen algorithm is additionally upper bounded by a fixed polynomial in λ, n and $\log q$.

Strengthening the simulation guarantee. In this work, we consider a strengthening of the above simulation-secure properties (for the class of weakly optimal static-bounded-collusion FE schemes) which will be be crucial towards building a dynamic-bounded-collusion functional encryption scheme. Note that typically the simulation security states that the distribution of secret keys and ciphertext are simulatable as long as the adversary does not make more key queries than what is specified by the static collusion bound. That is, if the adversary makes more key queries then no guarantee is provided. However, we consider a stronger simulation guarantee below wherein the real world is still simulatable even when the adversary makes more key queries than that specified by the collusion bound as long as the adversary does not make any challenge message queries. That is, either the collusion bound is not crossed, or no challenge ciphertext is queried. In addition to this, we require the running time of the simulator algorithms $\mathsf{S}_0, \mathsf{S}_1$ and S_3 (that is, all except the ciphertext simulator S_2) grow only poly-logarithmically in the static collusion bound q. Formally, we define it below.

Definition 3.3 (strong simulation-security)**.** A functional encryption scheme $\mathsf{FE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ is said to be statically-bounded-collusion *strong* simulation-secure if, in the security game defined in Definition 3.1, the following additional conditions hold:

1. the number of key queries made by adversary is allowed to exceed the static collusion bound q as long as the adversary does not submit any challenge message, and

2. the running time of the simulator algorithms $\mathsf{S}_0, \mathsf{S}_1$ and S_3 is upper bounded by a fixed polynomial in λ, n and $\log q$.

Lastly, we also define the non-adaptive variant of the simulation security.

Definition 3.4 (non-adaptive simulation-security)**.** A functional encryption scheme $\mathsf{FE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ is said to be statically-bounded-collusion *non-adaptive* (regular/strong) simulation-secure if the adversary is prohibited from making any key queries in the post-challenge phase (that is, after receiving the challenge ciphertext) in its respective security game.

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

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

(Note that thus syntactically the setup of a dynamic bounded collusion scheme is same as that of a fully collusion resistant scheme.)

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

Efficiency. Since the collusion bound q is not specified during setup or key generation at all, thus the efficiency condition for a dynamically bounded collusion FE scheme requires the running time of Setup and KeyGen to be fixed polynomials in λ and n. While since the encryptor takes q as input in unary, thus the running time of the Enc algorithm could grow polynomially with collusion bound q. Similarly, the running time of Dec is also allowed to grow polynomially with collusion bound q.

Dynamic bounded collusion security. This is formally captured via a 'simulation based' security definition as in the static setting. The game is similar to that provided in Definition 3.1, except now the attacker specifies the collusion bound q while making the challenge ciphertext query and the simulator also only receives the collusion bound as input at that point. For completeness, we describe it formally below (both the adaptive and non-adaptive variants).

Definition 3.5 (dynamic-bounded-collusion simulation-security)**.** A functional encryption scheme $\mathsf{FE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ is said to be dynamically-bounded-collusion simulation-secure if there exists a stateful PPT simulator Sim such that for every stateful PPT adversary $\mathcal{A}$, the following distributions are computationally indistinguishable:

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

$$\approx_c$$

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

whenever the admissibility constraints and properties, as defined in Definition 3.1, are satisfied.

Definition 3.6 (non-adaptive simulation-security)**.** A functional encryption scheme $\mathsf{FE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ is said to be dynamically-bounded-collusion *non-adaptive* simulation-secure if, in the security game defined in Definition 3.5, the adversary is prohibited from making any key queries in the post-challenge phase (that is, after receiving the challenge ciphertext).

3.2 Upgrading Static to Dynamic Bounded Collusion FE via Weak Optimal Efficiency

In this section, we provide a generic construction of a dynamic-bounded-collusion FE scheme from any static-bounded-collusion FE scheme that satisfies the strong simulation property (Definition 3.3) and the weak optimality property (Definition 3.2). Below we provide our construction followed by correctness and security proofs.

3.2.1 Construction

Let $\mathsf{Static\text{-}FE} = (\mathsf{S\text{-}FE.Setup}, \mathsf{S\text{-}FE.Enc}, \mathsf{S\text{-}FE.KeyGen}, \mathsf{S\text{-}FE.Dec})$ be a weakly-optimal static-bounded-collusion FE scheme for a family of function classes $\{\mathcal{F}_n\}_{n\in\mathbb{N}}$ and message spaces $\{\mathcal{M}_n\}_{n\in\mathbb{N}}$. We use $\mathsf{Static\text{-}FE}$ to build a dynamic-bounded-collusion FE scheme $\mathsf{FE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ as follows.

$\mathsf{Setup}(1^\lambda, 1^n) \to (\mathsf{mpk}, \mathsf{msk})$. The setup algorithm runs the $\mathsf{Static\text{-}FE}$ setup algorithm λ times with increasing values of the static collusion bound q as follows:

$$\forall i \in [\lambda], \quad (\mathsf{mpk}_i, \mathsf{msk}_i) \leftarrow \mathsf{S\text{-}FE.Setup}(1^\lambda, 1^n, q = 2^i).$$

It then sets the master secret and public keys as an λ-tuple of all these keys, i.e. $\mathsf{msk} = (\mathsf{msk}_i)_{i\in[\lambda]}$ and $\mathsf{mpk} = (\mathsf{mpk}_i)_{i\in[\lambda]}$.

$\mathsf{KeyGen}(\mathsf{msk}, f) \to \mathsf{sk}_f$. Let $\mathsf{msk} = (\mathsf{msk}_i)_{i\in[\lambda]}$. The key generation algorithm runs the $\mathsf{Static\text{-}FE}$ key generation algorithm with all λ keys independently as $\mathsf{sk}_{i,f} \leftarrow \mathsf{S\text{-}FE.KeyGen}(\mathsf{msk}_i, f)$ for $i \in [\lambda]$. It outputs the secret key sk as $\mathsf{sk} = (\mathsf{sk}_{i,f})_{i\in[\lambda]}$.

$\mathsf{Enc}(\mathsf{mpk}, m, 1^Q) \to \mathsf{ct}$. Let $\mathsf{mpk} = (\mathsf{mpk}_i)_{i\in[\lambda]}$. The encryption algorithm simply encrypts the message m under $\lceil \log Q \rceil$-th master public key as $\mathsf{ct} \leftarrow \mathsf{S\text{-}FE.Enc}(\mathsf{mpk}_{\lceil \log Q \rceil}, m)$. (It also includes Q as part of the ciphertext.)

$\mathsf{Dec}(\mathsf{sk}_f, \mathsf{ct}) \to z$. Let $\mathsf{sk}_f = (\mathsf{sk}_{i,f})_{i\in[\lambda]}$. The decryption algorithm runs the $\mathsf{Static\text{-}FE}$ decryption using the $\lceil \log Q \rceil$-th function key as $z \leftarrow \mathsf{S\text{-}FE.Dec}(\mathsf{sk}_{\lceil \log Q \rceil, f}, \mathsf{ct})$.

3.2.2 Correctness, Efficiency, and Security

The correctness of the above scheme follows directly from the correctness of the underlying static-bounded-collusion FE system, while for the desired efficiency consider the following arguments. First, note that by weak optimality of $\mathsf{Static\text{-}FE}$ we have that the running time of $\mathsf{S\text{-}FE.Setup}$ and $\mathsf{S\text{-}FE.KeyGen}$ grows as $\mathsf{poly}(\lambda, n, \log q)$. Since the Setup and $\mathsf{S\text{-}FE.KeyGen}$ algorithms run $\mathsf{S\text{-}FE.Setup}$ and $\mathsf{S\text{-}FE.KeyGen}$ (respectively) λ many times for $\log q \in \{1, \ldots, \lambda\}$, thus we get that running time of Setup and KeyGen is $\mathsf{poly}(\lambda, n)$ as desired. Lastly, the encryption and decryption algorithm run in time at most $\mathsf{poly}(\lambda, n, 2^{\lceil \log Q \rceil}) = \mathsf{poly}(\lambda, n, Q)$ since the $\lceil \log Q \rceil$-th static-bounded-collusion FE

system uses $2^{\lceil \log Q \rceil} \leq 2 \cdot Q$ as the static collusion bound. Thus, the resulting FE scheme satisfies the required efficiency properties.
To conclude, we prove the following.

Theorem 3.1. If $\mathsf{Static\text{-}FE} = (\mathsf{S\text{-}FE.Setup}, \mathsf{S\text{-}FE.Enc}, \mathsf{S\text{-}FE.KeyGen}, \mathsf{S\text{-}FE.Dec})$ is a weakly-optimal static-bounded-collusion simulation-secure FE scheme (as per Definitions 3.2 and 3.3), then the above scheme $\mathsf{FE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$ is a dynamic-bounded-collusion simulation-secure FE scheme (as per Definition 3.5).

Proof. The proof follows from a composition of the static-bounded-collusion simulation-security property of $\mathsf{Static\text{-}FE}$. Recall that in the static setting, we require the scheme to provide a stronger form of real world vs. ideal world indistinguishability. Where typically the simulation security states that the distribution of secret keys and ciphertext are simulatable as long as the adversary does not make more key queries than what is specified by the static collusion bound. That is, if the adversary makes more key queries then no guarantee is provided. However, in our formalization of simulation security for static-bounded-collusion FE schemes, we require that the real and ideal worlds are also indistinguishable even when the adversary makes more key queries than that specified by the collusion bound as long as the adversary does not make any challenge message queries. That is, either the collusion bound is not crossed, or no challenge ciphertext is queried. Also, the running time of the simulator algorithms $\mathsf{S}_0, \mathsf{S}_1$ and S_3 (all except the ciphertext simulator S_2) grow only poly-logarithmically in the collusion bound.

Thus, the simulator for the dynamic-bounded-collusion FE scheme simply runs the S_0 algorithms for all collusion bounds $q = 1, \ldots, 2^\lambda$ to simulate the individual master public keys. It then also runs the S_1 algorithms for simulating the individual function keys for each of these static-bounded-collusion FE systems for answering each adversarial key query. Note that since the running time of S_0 and S_1 is also $\mathsf{poly}(\lambda, n, \log q)$ where $q = 1, \ldots, 2^\lambda$, thus this is efficient.

Now when the adversary makes the challenge query for message m, it also specifies the target collusion bound Q^*. The dynamic-bounded-collusion simulator then runs only the ciphertext simulator algorithm S_2 for the static FE system corresponding to collusion bound $\log q = \lceil \log Q^* \rceil$. Note that the simulator does not run S_2 for the underlying FE schemes with lower (and even higher) collusion bounds. This is important for two reasons: (1) we want to invoke the simulation security of the i-th static FE scheme for $i < \lceil \log Q^* \rceil$ but we can only do this if the ciphertext simulator S_2 is not run for these static FE schemes, (2) the running time of S_2 could grow polynomically with the collusion bound q, thus we should not invoke simulator algorithm S_2 for $i > \lceil \log Q^* \rceil$ as well (since for say $i = \lambda$, the running time would be exponential in λ which would make the dynamic simulator inefficient). Thus, even the ciphertext simulation is efficient and the dynamic simulator is an admissible adversary with respect to static FE challenger, therefore our dynamic FE simulator is both efficient and can rely on simulation security of the underlying static FE schemes. The last phase of simulation (i.e., post-challenge key generation phase) works the same as the second phase simulator (i.e., pre-challenge key generation phase) which is by running S_3 for all collusion bounds. This completes a high level sketch. A complete proof will be provided in the full version.

□

Remark 3.2 (non-adaptive simulation-security)**.** If the underlying static-bounded-collusion FE scheme only provides security against non-adaptive attackers Definition 3.4, then the resulting dynamic-bounded-collusion FE scheme is also secure only against non-adaptive attackers Definition 3.6.

4 Statically Bounded Collusion FE for NC1

We construct $\mathsf{Static\text{-}FE} = (\mathsf{S\text{-}FE.Setup}, \mathsf{S\text{-}FE.Enc}, \mathsf{S\text{-}FE.KeyGen}, \mathsf{S\text{-}FE.Dec})$, a weakly-optimal static-bounded-collusion FE scheme for the family of function classes $\{\mathcal{F}_n\}_{n\in\mathbb{N}}$ computable by $\mathsf{NC1}$ circuits. Each function $\mathcal{F}_n$ takes as input $x \in \{0,1\}^\ell$ where the circuit computing the function has size upper bounded by λ and degree upper bounded by D. The ingredients for our construction are an identity-based-encryption scheme IBE and garbled circuits GC. We show that our construction is non-adaptive simulation secure Definition 3.4 and weakly optimal. Let κ be the security parameter for the remainder of this paper.

$\mathsf{S\text{-}FE.Setup}(1^\kappa, 1^\lambda, 1^{\mathsf{D}}, \mathsf{q}) \to (\mathsf{mpk}, \mathsf{msk})$

- Set the parameters $\mathsf{p} = \kappa \cdot \mathsf{D} + 1$, $\mathsf{p}' = \kappa$, $\Gamma = 2\mathsf{pq}^2$, $\Delta = 2\mathsf{q}$, $\mathsf{t} = \kappa$, $\mathsf{N} = \mathsf{p} \cdot \Gamma$, $\mathsf{N}' = \mathsf{p}' \cdot \Delta$. Let $\mathsf{pp} = (\mathsf{p}, \mathsf{p}', \Gamma, \Delta, \mathsf{t}, \mathsf{N}, \mathsf{N}', (1^\lambda, 1^{\mathsf{D}}, \mathsf{q}))$ be implicitly known to each algorithm. Let $\mathbb{F}$ be a finite field of size $> \mathsf{N}$.
- $\mathsf{IBE.Setup}(1^\kappa, ([\mathsf{p}] \times [\Gamma]) \times ([\mathsf{p}'] \times [\Delta])) \to (\mathsf{IBE.pk}_1, \mathsf{IBE.msk}_1)$[5].
- $\mathsf{IBE.Setup}(1^\kappa, ([\mathsf{p}] \times [\Gamma]) \times ([\lambda] \times \{0,1\})) \to (\mathsf{IBE.pk}_2, \mathsf{IBE.msk}_2)$.
- Output $(\mathsf{mpk} = (\mathsf{IBE.pk}_1, \mathsf{IBE.pk}_2), \mathsf{msk} = (\mathsf{IBE.msk}_1, \mathsf{IBE.msk}_2))$.

$\mathsf{S\text{-}FE.KeyGen}(\mathsf{msk}, C) \to \mathsf{sk}_C$

- Sample $a_1, \ldots, a_{\mathsf{p}} \xleftarrow{R} [\Gamma]$.
- Sample $b_1, \ldots, b_{\mathsf{p}'} \xleftarrow{R} [\Delta]$.
- Let $C = C_1 \ldots C_\lambda$ be the bits of circuit C.
- For $i \in [\mathsf{p}]$, let $\mathsf{tag}^1{}_i = (i, a_i)$ and for $i \in [\mathsf{p}']$ let $\mathsf{tag}^2{}_i = (i, b_i)$.
- For $i \in [\lambda]$, let $\mathsf{tag}^3{}_i = (i, C_i)$.
- Let $\mathsf{taglist}^1 = (\mathsf{tag}^1{}_1, \ldots, \mathsf{tag}^1{}_{\mathsf{p}})$, $\mathsf{taglist}^2 = (\mathsf{tag}^2{}_1, \ldots, \mathsf{tag}^2{}_{\mathsf{p}'})$, and $\mathsf{taglist}^3 = (\mathsf{tag}^3{}_1, \ldots, \mathsf{tag}^3{}_\lambda)$.
- Let $\mathsf{sk}^1 = \mathsf{sk}^2 = \emptyset$. We assume that these sets are ordered sets.
  For $i_1 \in [\mathsf{p}]$,
  - For $i_2 \in [\mathsf{p}']$,
    - $\mathsf{sk}^1_{i_1,i_2} \leftarrow \mathsf{IBE.KeyGen}(\mathsf{IBE.msk}_1, (\mathsf{tag}^1{}_{i_1}, \mathsf{tag}^2{}_{i_2}))$.
    - Add $\mathsf{sk}^1_{i_1,i_2}$ to the set sk^1.
  - For $i_3 \in [\lambda]$,
    - $\mathsf{sk}^2_{i_1,i_3} \leftarrow \mathsf{IBE.KeyGen}(\mathsf{IBE.msk}_2, (\mathsf{tag}^1{}_{i_1}, \mathsf{tag}^3{}_{i_3}))$.
    - Add $\mathsf{sk}^2_{i_1,i_3}$ to the set sk^2.
- Output $\mathsf{sk}_C = (\mathsf{sk}^1, \mathsf{sk}^2, \mathsf{taglist}^1, \mathsf{taglist}^2, \mathsf{taglist}^3)$.

$\mathsf{S\text{-}FE.Enc}(\mathsf{mpk}, x = x_1 \ldots x_\ell, 1^{\mathsf{q}}) \to \mathsf{ct}$

- Pick ℓ random degree t polynomials $\mu_1(\cdot), \ldots, \mu_\ell(\cdot)$ where $\mu_i(\cdot)$ has constant term x_i.
- Pick N unique nonzero field elements f_{tag^1} where $\mathsf{tag}^1 \in [\mathsf{p}] \times [\Gamma]$.

[5] Our IBE scheme takes identity space $\mathcal{I}_n$ as input where we can imagine $\mathcal{I}_n$ is a binary vector of length $n = \log(\mathsf{N} \cdot \mathsf{N}')$

- Pick N' random degree $\mathsf{p}-1$ polynomials $\zeta_{\mathsf{tag}^2}(\cdot)$ where $\mathsf{tag}^2 \in [\mathsf{p}'] \times [\Delta]$ with constant term 0.
- Define $U[x_1, \ldots, x_\ell](C = c_1, \ldots c_\lambda, Z_1, \ldots, Z_{\mathsf{p}'})$ to be the universal circuit which takes as input the description of a binary circuit C of length λ and p' elements of $\mathbb{F} - Z_1, \ldots, Z_{\mathsf{p}'}$ and does the following:
  - Interpret C as an arithmetic circuit over $\mathbb{F}$. [6]
  - Compute and output $C(x_1, \ldots, x_\ell) + \sum_{i=1}^{\mathsf{p}'} Z_i$ where $x_1, \ldots, x_\ell$ are hardcoded inside the circuit.
- Let $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}_3, \mathsf{ct}_4)$ be $(\emptyset, \emptyset, \emptyset, \emptyset)$. We assume that these sets are ordered sets.
- For every possible $\mathsf{tag}^1 \in [\mathsf{p}] \times [\Gamma]$,
  - Garble the universal circuit $\hat{U} = U[\mu_1(f_{\mathsf{tag}^1}), \ldots, \mu_\ell(f_{\mathsf{tag}^1})]$ for each tag^1, i.e. $(\tilde{U}^{\mathsf{tag}^1}, \mathcal{L}^{\mathsf{tag}^1}) \leftarrow \mathsf{GC.Garble}(1^\kappa, \hat{U})$ where $\tilde{U}^{\mathsf{tag}^1}$ denotes the garbled circuit and $\mathcal{L}^{\mathsf{tag}^1}$ indicates the set of labels for each wire in this circuit.
  - For every possible $\mathsf{tag}^2 = (i, \delta) \in [\mathsf{p}'] \times [\Delta]$,
    - Let $\mathsf{input}_2 = \mathcal{L}^{\mathsf{tag}^1}_{\{Z_i\},\{\zeta_{\mathsf{tag}^2}(f_{\mathsf{tag}^1})\}}$ where recall the $\mathcal{L}_{S,T}$ notation Definition 2.1 is used to choose the input labels for a garbled circuit for position S corresponding to the real input T.
    - $\mathsf{ct}^1_{\mathsf{tag}^1,\mathsf{tag}^2} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_2))$.
    - Add $\mathsf{ct}^1_{\mathsf{tag}^1,\mathsf{tag}^2}$ to the set ct^1.
  - For every possible $\mathsf{tag}^3 = (i, b) \in [\lambda] \times \{0, 1\}$,
    - Let $\mathsf{input}_3 = \mathcal{L}^{\mathsf{tag}^1}_{\{c_i\},\{b\}}$.
    - $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_2, (\mathsf{tag}^1, \mathsf{tag}^3), (\mathsf{input}_3))$.
    - Add $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3}$ to the set ct^2.
  - Add the garbled circuit $\tilde{U}^{\mathsf{tag}^1}$ to the set ct^3.
  - Add f_{tag^1} to the set ct^4.
- Output $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}_3, \mathsf{ct}_4)$.

$\mathsf{S\text{-}FE.Dec}(\mathsf{sk}_C, \mathsf{ct}, 1^{\mathsf{q}}) \to y$

- Parse ct as $(\mathsf{ct}^1, \mathsf{ct}^2, \mathsf{ct}^3, \mathsf{ct}^4)$.
  - ct^1 as $\{\mathsf{ct}^1_{\mathsf{tag}^1,\mathsf{tag}^2}\}_{\mathsf{tag}^1 \in [\mathsf{p}] \times [\Gamma], \mathsf{tag}^2 \in [\mathsf{p}'] \times [\Delta]}$.
  - ct^2 as $\{\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3}\}_{\mathsf{tag}^1 \in [\mathsf{p}] \times [\Gamma], \mathsf{tag}^3 \in [\lambda] \times \{0,1\}}$.
  - ct^3 as $\{\tilde{U}^{\mathsf{tag}^1}\}_{\mathsf{tag}^1 \in [\mathsf{p}] \times [\Gamma]}$.
  - ct^4 as $\{f_{\mathsf{tag}^1}\}_{\mathsf{tag}^1 \in [\mathsf{p}] \times [\Gamma]}$.
- Parse sk_C as $(\mathsf{sk}^1, \mathsf{sk}^2, \mathsf{taglist}^1, \mathsf{taglist}^2, \mathsf{taglist}^3)$.
  - sk^1 as $\{\mathsf{sk}^1_{i_1,i_2}\}_{i_1 \in [\mathsf{p}], i_2 \in [\mathsf{p}']}$.
  - sk^2 as $\{\mathsf{sk}^2_{i_1,i_2}\}_{i_1 \in [\mathsf{p}], i_2 \in [\lambda]}$.

[6] We emphasize U is still a boolean circuit which computes field additions and multiplications as an arithmitization of the boolean circuit C

- For $i_1 \in [\mathsf{p}]$,
  - For $i_2 \in [\mathsf{p}']$,
    - $\mathsf{input}^2_{i_2} = \mathsf{IBE.Dec}(\mathsf{sk}^1_{i_1,i_2}, \mathsf{ct}^1_{\mathsf{tag}^1{}_{i_1},\mathsf{tag}^2{}_{i_2}})$
  - For $i_3 \in [\lambda]$
    - $\mathsf{input}^3_{i_3} = \mathsf{IBE.Dec}(\mathsf{sk}^2_{i_1,i_3}, \mathsf{ct}^2_{\mathsf{tag}^1{}_{i_1},\mathsf{tag}^3{}_{i_3}})$
  - Combine input labels for garbled circuit $\tilde{U}^{\mathsf{tag}^1{}_{i_1}}$ as $L = \{\mathsf{input}^2_{i_2}\}_{i_2 \in [\mathsf{p}']} \cup \{\mathsf{input}^3_{i_3}\}_{i_3 \in [\lambda]}$.
  - Let $x_{\mathsf{tag}^1{}_{i_1}} = \mathsf{GC.Eval}(\tilde{U}^{\mathsf{tag}^1{}_{i_1}}, L)$
- Let π be a degree $\mathsf{p}-1$ polynomial defined as follows. For each $i_1 \in [\mathsf{p}]$, let the evaluation at input $f_{\mathsf{tag}^1{}_{i_1}}$ be defined as $\pi(f_{\mathsf{tag}^1{}_{i_1}}) = x_{\mathsf{tag}^1{}_{i_1}}$. Interpolate the polynomial π.
- Output $\pi(0)$.

Efficiency

Claim 4.1. If $\mathsf{IBE}, \mathsf{GC}$ consists of PPT algorithms, then our $\mathsf{Static\text{-}FE} = (\mathsf{S\text{-}FE.Setup}, \mathsf{S\text{-}FE.Enc}, \mathsf{S\text{-}FE.KeyGen}, \mathsf{S\text{-}FE.Dec})$ scheme consists of PPT algorithms and is weakly optimal according to Definition 3.2.

Proof. We analyze the different algorithms.

- $\mathsf{S\text{-}FE.Setup}$ runs $\mathsf{IBE.Setup}$ twice for different identity spaces. We have that the first setup runs in time $\mathsf{poly}(\kappa, \log(\mathsf{N} \cdot \mathsf{N}')) = \mathsf{poly}(\kappa, \log(\mathsf{q}, \mathsf{D}))$. The second setup runs in time $\mathsf{poly}(\kappa, \log(\mathsf{N} \cdot \lambda \cdot 2)) = \mathsf{poly}(\kappa, \log(\mathsf{q}, \lambda))$. Thus $\mathsf{S\text{-}FE.Setup}$ is efficient according to Definition 3.2.
- $\mathsf{S\text{-}FE.KeyGen}$ runs $\mathsf{IBE.KeyGen}$ with $\mathsf{IBE.msk}_1$ for $\mathsf{p} \cdot \mathsf{p}'$ times and $\mathsf{IBE.KeyGen}$ with $\mathsf{IBE.msk}_2$ for $\mathsf{p} \cdot \lambda$ times. One invocation of $\mathsf{IBE.KeyGen}$ with $\mathsf{IBE.msk}_1$ takes $\mathsf{poly}(\kappa, \log(\mathsf{q}, \mathsf{D}))$ time and one invocation of $\mathsf{IBE.KeyGen}$ with $\mathsf{IBE.msk}_2$ takes $\mathsf{poly}(\kappa, \log(\mathsf{q}, \lambda))$ time. The whole algorithm thus runs in time $\mathsf{poly}(\kappa, \lambda, \mathsf{D}, \log(\mathsf{q}))$.
- $\mathsf{S\text{-}FE.Enc}$ runs $\mathsf{IBE.Enc}$ with $\mathsf{IBE.pk}_1$, $\mathsf{N} \cdot \mathsf{N}'$ times, $\mathsf{IBE.Enc}$ with $\mathsf{IBE.pk}_2$, $\mathsf{N} \cdot \lambda \cdot 2$ times and $\mathsf{GC.Garble}$ N times on a circuit that takes inputs of size $\lambda + \mathsf{p}'$ (and hardcoded ℓ values). Since these are PPT algorithms in their input, the resulting algorithm is $\mathsf{poly}(\kappa, \ell, \lambda, \mathsf{D}, \mathsf{q})$.
- $\mathsf{S\text{-}FE.Dec}$ runs $\mathsf{IBE.Dec}$ on ct's encrypted with $\mathsf{IBE.pk}_1$ $\mathsf{p} \cdot \mathsf{p}'$ times, $\mathsf{IBE.Dec}$ on ct's encrypted with $\mathsf{IBE.pk}_2$ $\mathsf{p} \cdot \lambda$ times and $\mathsf{GC.Eval}$ p times on a garbled circuit that garbled a circuit that takes inputs of size $\lambda + \mathsf{p}'$ (hardcoded ℓ values). Since these are PPT algorithms in their input, the resulting algorithm is $\mathsf{poly}(\kappa, \ell, \lambda, \mathsf{D}, \mathsf{q})$.

□

Correctness

Claim 4.2. If IBE is a correct identity based encryption scheme, GC is a correct garbling scheme, then our $\mathsf{S\text{-}FE}$ scheme is correct weakly-optimal static-bounded-collusion FE scheme.

Proof. We make the following observations for each $\kappa, \mathsf{q}, x \in \{0,1\}^\ell$, for any polynomials (in κ) λ (the maximum size of circuit), D (degree of the circuit) and any circuit C with size $\leq \lambda$ and degree $\leq \mathsf{D}$.

$\mathsf{S\text{-}FE.Dec}(\mathsf{sk}_C, \mathsf{ct}, 1^\mathsf{q})$, computes $\forall i_1 \in [\mathsf{p}], \forall i_2 \in [\mathsf{p}']$, $\mathsf{IBE.Dec}(\mathsf{sk}^1_{i_1,i_2}, \mathsf{ct}^1_{\mathsf{tag}^1{}_{i_1},\mathsf{tag}^2{}_{i_2}})$. We compute $\mathsf{sk}^1_{i_1,i_2}$ during $\mathsf{S\text{-}FE.KeyGen}$ as $\mathsf{IBE.KeyGen}(\mathsf{msk}_1, (\mathsf{tag}^1{}_{i_1}, \mathsf{tag}^2{}_{i_2}))$ and $\mathsf{ct}^1_{\mathsf{tag}^1{}_{i_1},\mathsf{tag}^2{}_{i_2}}$ during $\mathsf{IBE.Enc}$ as encryption of $\mathcal{L}^{\mathsf{tag}^1{}_{i_1}}_{\{Z_{i_2}\},\{\zeta_{\mathsf{tag}^2{}_{i_2}}(f_{\mathsf{tag}^1{}_{i_1}})\}}$ on $\mathsf{id} = (\mathsf{tag}^1{}_{i_1}, \mathsf{tag}^2{}_{i_2})$. If IBE is a correct identity based encryption scheme, then the decryption proceeds correctly i.e. the labels for garbled circuit $\tilde{U}^{\mathsf{tag}^1{}_{i_1}}$ where the input label for each Z_{i_2} is set according to the input $\zeta_{\mathsf{tag}^2{}_{i_2}}(f_{\mathsf{tag}^1{}_{i_1}})$.

Similarly, decryption computes $\forall i_1 \in [\mathsf{p}], \forall i_3 \in [\lambda]$, $\mathsf{IBE.Dec}(\mathsf{sk}^2_{i_1,i_3}, \mathsf{ct}^2_{\mathsf{tag}^1{}_{i_1},\mathsf{tag}^3{}_{i_3}})$. During keygen, $\mathsf{tag}^3{}_{i_3}$ is set as (i_3, C_i) where C is the circuit being computed and $\mathsf{sk}^2_{i_1,i_3}$ is computed as $\mathsf{IBE.KeyGen}(\mathsf{msk}_2, (\mathsf{tag}^1{}_{i_1}, \mathsf{tag}^3{}_{i_3}))$ and $\mathsf{ct}^2_{\mathsf{tag}^1{}_{i_1},\mathsf{tag}^3{}_{i_3}}$ during $\mathsf{IBE.Enc}$ as encryption of $\mathcal{L}^{\mathsf{tag}^1{}_{i_1}}_{\{c_i\},\{C_i\}}$ on $\mathsf{id} = (\mathsf{tag}^1{}_{i_1}, \mathsf{tag}^3{}_{i_3})$. If IBE is a correct identity based encryption scheme, then the decryption proceeds correctly i.e. the labels for garbled circuit $\tilde{U}^{\mathsf{tag}^1{}_{i_1}}$ where the input label for each c_i is set according to the input C_i.

Thus decryption through the procedure $\mathsf{GC.Eval}$ computes $\forall i_1 \in [\mathsf{p}]$,

$$C(\mu_1(f_{\mathsf{tag}^1{}_{i_1}}), \ldots, \mu_\ell(f_{\mathsf{tag}^1{}_{i_1}})) + \sum_{i_2 \in [\mathsf{p}']} \zeta_{\mathsf{tag}^2{}_{i_2}}(f_{\mathsf{tag}^1{}_{i_1}}).$$

As we know for each $i_1 \in [\mathsf{p}]$, $f_{\mathsf{tag}^1{}_{i_1}}$ from parsing of ct^4. We can compute the degree $\mathsf{p}-1$ polynomial π using polynomial interpolation.

Note that $\pi(0) = C(\mu_1(0), \ldots, \mu_\ell(0)) + \sum_{i_2 \in [\mathsf{p}']} \zeta_{\mathsf{tag}^2{}_{i_2}}(0)$. As polynomials $\mu_1, \ldots, \mu_\ell$ have constant terms $x_1, \ldots, x_\ell$ and the rest of the constant terms are 0, we compute $C(x)$.

□

4.1 Security

Theorem 4.1. Let IBE be a secure encryption scheme according to Definition 2.3 and GC be a secure garbled circuits scheme according to Definition 2.2. Then the above $\mathsf{S\text{-}FE}$ scheme is a statically-bounded-collusion *non-adaptive* strong simulation-secure encryption scheme according to Definition 3.4.

Proof. We describe the simulators below. For *non-adaptive* simulation security, we focus on simulators $\mathsf{S}_0, \mathsf{S}_1, \mathsf{S}_2$.

$\mathsf{S}_0(1^\kappa, 1^\ell, 1^\lambda, 1^\mathsf{D}, \mathsf{q}) \to (\mathsf{mpk}, \mathsf{st}_0)$

- Run $\mathsf{S\text{-}FE.Setup}(1^\kappa, 1^\ell, 1^\lambda, 1^\mathsf{D}, \mathsf{q}) \to (\mathsf{mpk}, \mathsf{msk})$. Let st_0 be msk and output $(\mathsf{mpk}, \mathsf{st}_0)$.

$\mathsf{S}_1(\mathsf{st}_0, C) \to \mathsf{sk}_C$.

- Run $\mathsf{S\text{-}FE.KeyGen}(\mathsf{st}_0, C) \to \mathsf{sk}_C$.

$\mathsf{S}_2(\mathsf{st}_1, \Pi^x) \to (\mathsf{ct}, \mathsf{st}_2)$.

- **Parsing:**
  - Let $|\Pi^x| = \mathsf{q}$ (wlog the cardinality is the maximum number of queries allowed).
  - Parse $\Pi^x = ((C_1, C_1(x)), \ldots, (C_{\mathsf{q}}, C_{\mathsf{q}}(x)))$.
  - $\forall j \in [\mathsf{q}]$, let sk_{C_j} be the reply to query C_j to S_1.
  - Parse sk_{C_j} as $(\mathsf{sk}^{1,(j)}, \mathsf{sk}^{2,(j)}, \mathsf{taglist}^{1,(j)}, \mathsf{taglist}^{2,(j)}, \mathsf{taglist}^{3,(j)})$.
  - Parse $\mathsf{taglist}^{1,(j)} = (\mathsf{tag}_1^{1,(j)}, \ldots, \mathsf{tag}_{\mathsf{p}}^{1,(j)})$.
  - Parse $\mathsf{taglist}^{2,(j)} = (\mathsf{tag}_1^{2,(j)}, \ldots, \mathsf{tag}_{\mathsf{p}'}^{2,(j)})$.
  - Parse $\mathsf{taglist}^{3,(j)} = (\mathsf{tag}_1^{3,(j)}, \ldots, \mathsf{tag}_{\lambda}^{3,(j)})$.
- **Computing colluding identities:**
  - Let $\mathcal{I} \subseteq ([\mathsf{p}] \times [\Gamma])$ be a set of identites,

$$\cup_{j \in \mathsf{q}, j' \in \mathsf{q}, i_1 \in [\mathsf{p}] \ \wedge \ j \neq j'} \left((\mathsf{tag}_{i_1}^{1,(j)}) \cap (\mathsf{tag}_{i_1}^{1,(j')}) \right).$$

  - If $|\mathcal{I}| > \mathsf{t}$, output $\perp$.
- **Computing cover free elements:**
  For $j \in \mathsf{q}$, let $\mathcal{K} = \mathcal{K}^* = \emptyset$. Let $\mathcal{K}^*$ be an ordered set.
  - Let $\mathcal{K}_1 = \cup_{i_2 \in [\mathsf{p}']} \mathsf{tag}_{i_2}^{2,(j)}$.
  - Pick any element $\mathsf{tag}_{i_2^*}^{2,(j)} \in \mathcal{K}_1 \setminus \mathcal{K} \in ([\mathsf{p}'] \times [\Delta])$. If no such element exists, then output $\perp$, otherwise, add $\mathsf{tag}_{i_2^*}^{2,(j)}$ to $\mathcal{K}^*$ such that $\mathcal{K}_j^* = \mathsf{tag}_{i_2^*}^{2,(j)}$.
  - Add $\mathcal{K}_1$ to set $\mathcal{K}$. i.e. $\mathcal{K} = \mathcal{K} \cup \mathcal{K}_1$.
- **Sampling polynomials:**
  - Pick ℓ random degree t polynomials $\mu_1(\cdot), \ldots, \mu_\ell(\cdot)$ where $\mu_i(\cdot)$ has constant term 0.
  - Pick N unique nonzero field elements f_{tag^1} where $\mathsf{tag}^1 \in [\mathsf{p}] \times [\Gamma]$.
  - For $j \in \mathsf{q}$,
    - Sample a random degree $\mathsf{p} - 1$ polynomials $\eta_j(\cdot)$ with constant term $C_j(x)$.
  - For $\mathsf{tag}^2 \in ([\mathsf{p}'] \times [\Delta]) \setminus \mathcal{K}^*$,
    - Pick a random degree $\mathsf{p} - 1$ polynomials $\zeta_{\mathsf{tag}^2}(\cdot)$ with constant term 0.
  - For $j \in \mathsf{q}$,
    - Set $\zeta_{\mathcal{K}_j^*}(\cdot)$ such that the following equation is satisfied, $\forall \mathsf{tag}^1 \in \mathcal{I}$,

$$\eta_j(f_{\mathsf{tag}^1}) = C_j(\mu_1(f_{\mathsf{tag}^1}), \ldots, \mu_\ell(f_{\mathsf{tag}^1})) + \sum_{i_2 \in [\mathsf{p}']} \zeta_{\mathsf{tag}_{i_2}^{2,(j)}}(f_{\mathsf{tag}^1}).$$

      Since we know the values of η_j at points in $\mathcal{I}$ and only one polynomial is not picked from the construction of $\mathcal{K}^*$, this is a valid assignment.
- **Defining circuits:**
  - Define $U[x_1, \ldots, x_\ell](C = c_1, \ldots c_\lambda, Z_1, \ldots, Z_{\mathsf{p}'})$ to be the universal circuit which takes as input the description of a binary circuit C of length λ and p' elements of $\mathbb{F} - Z_1, \ldots, Z_{\mathsf{p}'}$ and does the following:

* Interpret C as an arithmetic circuit over $\mathbb{F}$.
* Compute and output $C(x_1, \ldots, x_\ell) + \sum_{i=1}^{\mathsf{p}'} Z_i$ where $x_1, \ldots, x_\ell$ are hardcoded inside the circuit.

- Let $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}_3, \mathsf{ct}_4)$ be $(\emptyset, \emptyset, \emptyset, \emptyset)$. We assume that these sets are ordered sets.
- **Computing Ciphertexts:**
  For every possible $\mathsf{tag}^1 \in [\mathsf{p}] \times [\Gamma]$,
  - If $\mathsf{tag}^1 \in \mathcal{I}$, proceed according to the normal encryption, i.e.
    * Garble the universal circuit $\hat{U} = U[\mu_1(f_{\mathsf{tag}^1}), \ldots, \mu_\ell(f_{\mathsf{tag}^1})]$ for each tag^1, i.e. $(\tilde{U}^{\mathsf{tag}^1}, \mathcal{L}^{\mathsf{tag}^1}) \leftarrow \mathsf{GC.Garble}(1^\kappa, \hat{U})$ where $\tilde{U}^{\mathsf{tag}^1}$ denotes the garbled circuit and $\mathcal{L}^{\mathsf{tag}^1}$ indicates the set of labels for each wire in this circuit.
    * For every possible $\mathsf{tag}^2 = (i, \delta) \in [\mathsf{p}'] \times [\Delta]$,
      · Let $\mathsf{input}_2 = \mathcal{L}^{\mathsf{tag}^1}_{\{Z_i\}, \{\zeta_{\mathsf{tag}^2}(f_{\mathsf{tag}^1})\}}$ where recall the $\mathcal{L}_{S,T}$ notation Definition 2.1 is used to choose the input labels for a garbled circuit for position S corresponding to the real input T.
      · $\mathsf{ct}^1_{\mathsf{tag}^1, \mathsf{tag}^2} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_2))$.
      · Add $\mathsf{ct}^1_{\mathsf{tag}^1, \mathsf{tag}^2}$ to the set ct^1.
    * For every possible $\mathsf{tag}^3 = (i, b) \in [\lambda] \times \{0, 1\}$,
      · Let $\mathsf{input}_3 = \mathcal{L}^{\mathsf{tag}^1}_{\{c_i\}, \{b\}}$.
      · $\mathsf{ct}^2_{\mathsf{tag}^1, \mathsf{tag}^3} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_2, (\mathsf{tag}^1, \mathsf{tag}^3), (\mathsf{input}_3))$.
      · Add $\mathsf{ct}^2_{\mathsf{tag}^1, \mathsf{tag}^3}$ to the set ct^2.
    * Add the garbled circuit $\tilde{U}^{\mathsf{tag}^1}$ to the set ct^3.
    * Add f_{tag^1} to the set ct^4.
  - Else if $\mathsf{tag}^1 \notin \mathcal{I}$, at most one of $\mathsf{tag}^{1,(1)}, \ldots, \mathsf{tag}^{1,(\mathsf{q})}$ contain tag^1, because of our construction of set $\mathcal{I}$. Let j^* be the set such that $\exists i_1^* \in [\mathsf{p}]$ where $\mathsf{tag}^{1,(j^*)}_{i_1^*} = \mathsf{tag}^1$. If no such j^* exists, then $j^* = \perp$.
  - If $j^* \leq \mathsf{q}$,
    * Garble the universal circuit $\hat{U} = U[\mu_1(f_{\mathsf{tag}^1}), \ldots, \mu_\ell(f_{\mathsf{tag}^1})]$.
    * Let $(\tilde{U}^{\mathsf{tag}^1}, L) \leftarrow \mathsf{GC.Sim}(1^\kappa, 1^{(\lambda+\mathsf{p}')\cdot \log|\mathbb{F}|}, 1^{|\hat{U}|}, \eta_{j^*}(f_{\mathsf{tag}^1}))$ where $\tilde{U}^{\mathsf{tag}^1}$ denotes the garbled circuit and L are the input wires for the circuit.
    * For every possible $\mathsf{tag}^2 = (i_2, \delta) \in [\mathsf{p}'] \times [\Delta]$,
      · Let $\mathsf{input}_2 = L_{\{Z_{i_2}\}}$ (Definition 2.1).
      · If $\mathsf{tag}^{2,(j^*)}_{i_2} = \mathsf{tag}^2$, then $\mathsf{ct}^1_{\mathsf{tag}^1, \mathsf{tag}^2} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_2))$.
      · Else, $\mathsf{ct}^1_{\mathsf{tag}^1, \mathsf{tag}^2} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), 0^{|\mathsf{input}_2|})$.
      · Add $\mathsf{ct}^1_{\mathsf{tag}^1, \mathsf{tag}^2}$ to the set ct^1.
    * For every possible $\mathsf{tag}^3 = (i, b) \in [\lambda] \times \{0, 1\}$,
      · Let $\mathsf{input}_3 = L_{\{c_i\}}$.
      · If $C_{j^*, i} = b$, then $\mathsf{ct}^2_{\mathsf{tag}^1, \mathsf{tag}^3} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_2, (\mathsf{tag}^1, \mathsf{tag}^3), (\mathsf{input}_3))$.
      · Else, $\mathsf{ct}^2_{\mathsf{tag}^1, \mathsf{tag}^3} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_2, (\mathsf{tag}^1, \mathsf{tag}^3), 0^{|\mathsf{input}_3|})$.

· Add $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3}$ to the set ct^2.

∗ Add the garbled circuit $\tilde{U}^{\mathsf{tag}^1}$ to the set ct^3.

∗ Add f_{tag^1} to the set ct^4.

– Else,

∗ Garble the universal circuit $\hat{U} = U[0,\ldots,0]$ for each tag^1, i.e. $(\tilde{U}^{\mathsf{tag}^1}, \mathcal{L}^{\mathsf{tag}^1}) \leftarrow \mathsf{GC.Garble}(1^\kappa, \hat{U})$ where $\tilde{U}^{\mathsf{tag}^1}$ denotes the garbled circuit and $\mathcal{L}^{\mathsf{tag}^1}$ indicates the set of labels for each wire in this circuit.

∗ For every possible $\mathsf{tag}^2 = (i,\delta) \in [\mathsf{p}'] \times [\Delta]$,

· Let $\mathsf{input}_2 = \mathcal{L}^{\mathsf{tag}^1}_{\{Z_i\},\{\zeta_{\mathsf{tag}^2}(f_{\mathsf{tag}^1})\}}$.

· $\mathsf{ct}^1_{\mathsf{tag}^1,\mathsf{tag}^2} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1,\mathsf{tag}^2), 0^{|\mathsf{input}_2|})$.

· Add $\mathsf{ct}^1_{\mathsf{tag}^1,\mathsf{tag}^2}$ to the set ct^1.

∗ For every possible $\mathsf{tag}^3 = (i,b) \in [\lambda] \times \{0,1\}$

· Let $\mathsf{input}_3 = \mathcal{L}^{\mathsf{tag}^1}_{\{c_i\},\{b\}}$.

· $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, \emptyset, (\mathsf{tag}^1,\mathsf{tag}^3), 0^{|\mathsf{input}_3|})$.

· Add $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3}$ to the set ct^2.

∗ Add the garbled circuit $\tilde{U}^{\mathsf{tag}^1}$ to the set ct^3.

∗ Add f_{tag^1} to the set ct^4.

- Output $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}_3, \mathsf{ct}_4)$ with an empty state st_2.

The security follows from a careful case analysis.

- **Case 1:** For identities where $\mathsf{tag}^1 \in \mathcal{I}$, observe that there exist at least two keygen queries where secret keys might be issued. Thus we cannot rely on IBE security for these queries.
- **Case 2:** For identities where $\mathsf{tag}^1 \notin \mathcal{I}$ and $\jmath^* \leq \mathsf{q}$, observe that here we only rely on security for identities where $\mathsf{tag}^{2,(j^*)}_{i_2} \neq \mathsf{tag}^2$ or $C_{j^*,i} \neq b$. Since the query j^* was made on C_{j^*} and these sets of identities revealed, we want to avoid these identities. The remaining labels for the circuit can be simulated using the garbled circuit simulator as we know the output of $C_{j^*}(x)$.
- **Case 3:** For identities where $\mathsf{tag}^1 \notin \mathcal{I}$ and $\jmath^* = \perp$, we can rely on IBE security as know no identity is revealed in any of the prior queries. Thus all the labels of the garbled circuit are computationally hidden to an adversary and no information is revealed.

To capture these changes formally, we define our sequence of games. For each adjacent set of games we prove that the advantage of any PPT attacker $\mathcal{A}$ in the two games must be negligibly close.

First we change how polynomials ζ is sampled. This change is made as the simulator only gets $C_j(x)$ as input and not the plaintext x. Thus we eventually want to sample η directly depending on the value of $C_j(x)$ and not have it computed from setting μ and ζ's. In the next game, we rely on IBE security and those identities which have not been shared in prior keygen queries according to the case analysis discussed above. Finally we simulate the garbled circuit using the evaluation $C_j(x)$ and directly not the input μ's. In the final game we observe that μ values are only evaluated for points in $\mathcal{I}$ corresponding to Case 1 and hence using Lemma A.3, we can change how μ's are sampled to hide the final lingering information about x.

Game 0. This is the real non-adaptive security game between a challenger and an attacker. The game is parameterized by a security parameter κ.

1. Adversary $\mathcal{A}$ outputs circuit parameters $1^\lambda, 1^{\mathsf{D}}$ and collusion bound q.
2. Challenger $\mathcal{C}$ runs $\mathsf{S\text{-}FE.Setup}(1^\kappa, 1^\lambda, 1^{\mathsf{D}}, \mathsf{q}) \to (\mathsf{mpk}, \mathsf{msk})$ and sends mpk to $\mathcal{A}$.
   Setup phase:
   - Set the parameters $\mathsf{p} = \kappa \cdot \mathsf{D} + 1$, $\mathsf{p}' = \kappa$, $\Gamma = 2\mathsf{pq}^2$, $\Delta = 2\mathsf{q}$, $\mathsf{t} = \kappa$, $\mathsf{N} = \mathsf{p} \cdot \Gamma$, $\mathsf{N}' = \mathsf{p}' \cdot \Delta$. Let $\mathsf{pp} = (\mathsf{p}, \mathsf{p}', \Gamma, \Delta, \mathsf{t}, \mathsf{N}, \mathsf{N}', (1^\lambda, 1^{\mathsf{D}}, \mathsf{q}))$ be implicitly known to each algorithm.
   - $\mathsf{IBE.Setup}(1^\kappa, ([\mathsf{p}] \times [\Gamma]) \times ([\mathsf{p}'] \times [\Delta])) \to (\mathsf{IBE.pk}_1, \mathsf{IBE.msk}_1)$[7].
   - $\mathsf{IBE.Setup}(1^\kappa, ([\mathsf{p}] \times [\Gamma]) \times ([\lambda] \times \{0,1\})) \to (\mathsf{IBE.pk}_2, \mathsf{IBE.msk}_2)$.
   - Output $(\mathsf{mpk} = (\mathsf{IBE.pk}_1, \mathsf{IBE.pk}_2), \mathsf{msk} = (\mathsf{IBE.msk}_1, \mathsf{IBE.msk}_2))$.
3. **Keygen query phase:** $\mathcal{A}$ queries the keygen oracle $\mathsf{S\text{-}FE.KeyGen}(\mathsf{msk}, \cdot)$ and makes q queries, where the oracle computes as follows, $\mathsf{S\text{-}FE.KeyGen}(\mathsf{msk}, C) \to \mathsf{sk}_C$.
   - Sample $a_1, \ldots, a_{\mathsf{p}} \xleftarrow{R} [\Gamma]$.
   - Sample $b_1, \ldots, b_{\mathsf{p}'} \xleftarrow{R} [\Delta]$.
   - Let $C = C_1 \ldots C_\lambda$ be the bits of circuit C.
   - For $i \in [\mathsf{p}]$, let $\mathsf{tag}^1{}_i = (i, a_i)$ and for $i \in [\mathsf{p}']$ let $\mathsf{tag}^2{}_i = (i, b_i)$.
   - For $i \in [\lambda]$, let $\mathsf{tag}^3{}_i = (i, C_i)$.
   - Let $\mathsf{taglist}^1 = (\mathsf{tag}^1{}_1, \ldots, \mathsf{tag}^1{}_{\mathsf{p}})$, $\mathsf{taglist}^2 = (\mathsf{tag}^2{}_1, \ldots, \mathsf{tag}^2{}_{\mathsf{p}'})$, and $\mathsf{taglist}^3 = (\mathsf{tag}^3{}_1, \ldots, \mathsf{tag}^3{}_\lambda)$.
   - Let $\mathsf{sk}^1 = \mathsf{sk}^2 = \emptyset$. We assume that these sets are ordered sets.
     For $i_1 \in [\mathsf{p}]$,
     - For $i_2 \in [\mathsf{p}']$,
       - $\mathsf{sk}^1_{i_1,i_2} \leftarrow \mathsf{IBE.KeyGen}(\mathsf{IBE.msk}_1, (\mathsf{tag}^1{}_{i_1}, \mathsf{tag}^2{}_{i_2}))$.
       - Add $\mathsf{sk}^1_{i_1,i_2}$ to the set sk^1.
     - For $i_3 \in [\lambda]$,
       - $\mathsf{sk}^2_{i_1,i_3} \leftarrow \mathsf{IBE.KeyGen}(\mathsf{IBE.msk}_2, (\mathsf{tag}^1{}_{i_1}, \mathsf{tag}^3{}_{i_3}))$.
       - Add $\mathsf{sk}^2_{i_1,i_3}$ to the set sk^2.
   - Output $\mathsf{sk}_C = (\mathsf{sk}^1, \mathsf{sk}^2, \mathsf{taglist}^1, \mathsf{taglist}^2, \mathsf{taglist}^3)$.
4. **Challenge message:** $\mathcal{A}$ outputs a challenge message $x \in \{0,1\}^\ell$.
5. **Encryption phase:** $\mathcal{C}$ computes $\mathsf{S\text{-}FE.Enc}(\mathsf{mpk}, x, 1^{\mathsf{q}}) \to \mathsf{ct}$. It sends this to $\mathcal{A}$.
   - **Sampling polynomials:**
     - Pick ℓ random degree t polynomials $\mu_1(\cdot), \ldots, \mu_\ell(\cdot)$ where $\mu_i(\cdot)$ has constant term x_i.
     - Pick N unique nonzero field elements f_{tag^1} where $\mathsf{tag}^1 \in [\mathsf{p}] \times [\Gamma]$.

[7] Our IBE scheme takes identity space $\mathcal{I}_n$ as input where we can imagine $\mathcal{I}_n$ is a binary vector of length $n = \log(\mathsf{N} \cdot \mathsf{N}')$

- Pick N' random degree $\mathsf{p}-1$ polynomials $\zeta_{\mathsf{tag}^2}(\cdot)$ where $\mathsf{tag}^2 \in [\mathsf{p}'] \times [\Delta]$ with constant term 0.

- **Defining circuits:**
  - Define $U[x_1, \ldots, x_\ell](C = c_1, \ldots c_\lambda, Z_1, \ldots, Z_{\mathsf{p}'})$ to be the universal circuit which takes as input the description of a binary circuit C of length λ and p' elements of $\mathbb{F} - Z_1, \ldots, Z_{\mathsf{p}'}$ and does the following:
    * Interpret C as an arithmetic circuit over $\mathbb{F}$.
    * Compute and output $C(x_1, \ldots, x_\ell) + \sum_{i=1}^{\mathsf{p}'} Z_i$ where $x_1, \ldots, x_\ell$ are hardcoded inside the circuit.
- Let $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}_3, \mathsf{ct}_4)$ be $(\emptyset, \emptyset, \emptyset, \emptyset)$. We assume that these sets are ordered sets.
- **Computing Ciphertexts:**
  For every possible $\mathsf{tag}^1 \in [\mathsf{p}] \times [\Gamma]$,
  - Garble the universal circuit $\hat{U} = U[\mu_1(f_{\mathsf{tag}^1}), \ldots, \mu_\ell(f_{\mathsf{tag}^1})]$ for each tag^1, i.e. $(\tilde{U}^{\mathsf{tag}^1}, \mathcal{L}^{\mathsf{tag}^1}) \leftarrow \mathsf{GC.Garble}(1^\kappa, \hat{U})$ where $\tilde{U}^{\mathsf{tag}^1}$ denotes the garbled circuit and $\mathcal{L}^{\mathsf{tag}^1}$ indicates the set of labels for each wire in this circuit.
  - For every possible $\mathsf{tag}^2 = (i, \delta) \in [\mathsf{p}'] \times [\Delta]$,
    * Let $\mathsf{input}_2 = \mathcal{L}^{\mathsf{tag}^1}_{\{Z_i\}, \{\zeta_{\mathsf{tag}^2}(f_{\mathsf{tag}^1})\}}$ where recall the $\mathcal{L}_{S,T}$ notation Definition 2.1 is used to choose the input labels for a garbled circuit for position S corresponding to the real input T.
    * $\mathsf{ct}^1_{\mathsf{tag}^1, \mathsf{tag}^2} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_2))$.
    * Add $\mathsf{ct}^1_{\mathsf{tag}^1, \mathsf{tag}^2}$ to the set ct^1.
  - For every possible $\mathsf{tag}^3 = (i, b) \in [\lambda] \times \{0, 1\}$,
    * Let $\mathsf{input}_3 = \mathcal{L}^{\mathsf{tag}^1}_{\{c_i\}, \{b\}}$.
    * $\mathsf{ct}^2_{\mathsf{tag}^1, \mathsf{tag}^3} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_2, (\mathsf{tag}^1, \mathsf{tag}^3), (\mathsf{input}_3))$.
    * Add $\mathsf{ct}^2_{\mathsf{tag}^1, \mathsf{tag}^3}$ to the set ct^2.
  - Add the garbled circuit $\tilde{U}^{\mathsf{tag}^1}$ to the set ct^3.
  - Add f_{tag^1} to the set ct^4.
- Output $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}_3, \mathsf{ct}_4)$.

6. **Guess:** $\mathcal{A}$ outputs a bit b.

Game 1. In this game, we change the order in how we sample polynomials ζ i.e.

1. Adversary $\mathcal{A}$ outputs circuit parameters $1^\lambda, 1^{\mathsf{D}}$ and collusion bound q.
2. **Setup Phase:...,Keygen query phase:...,Challenge message:....**
3. **Encryption phase:** $\mathcal{C}$ computes $\mathsf{S\text{-}FE.Enc}(\mathsf{mpk}, x, 1^{\mathsf{q}}) \to \mathsf{ct}$. It sends this to $\mathcal{A}$.

- **Parsing:**
  - $\forall j \in [\mathsf{q}]$, let sk_{C_j} be the reply to query C_j to S_1.
  - Parse sk_{C_j} as $(\mathsf{sk}^{1,(j)}, \mathsf{sk}^{2,(j)}, \mathsf{taglist}^{1,(j)}, \mathsf{taglist}^{2,(j)}, \mathsf{taglist}^{3,(j)})$.

- Parse $\mathsf{taglist}^{1,(j)} = (\mathsf{tag}_1^{1,(j)}, \ldots, \mathsf{tag}_{\mathsf{p}}^{1,(j)})$.
- Parse $\mathsf{taglist}^{2,(j)} = (\mathsf{tag}_1^{2,(j)}, \ldots, \mathsf{tag}_{\mathsf{p}'}^{1,(j)})$.
- Parse $\mathsf{taglist}^{3,(j)} = (\mathsf{tag}_1^{3,(j)}, \ldots, \mathsf{tag}_{\lambda}^{1,(j)})$.

- **Computing colluding identities:**
  - Let $\mathcal{I} \subseteq ([\mathsf{p}] \times [\Gamma])$ be a set of identites,

$$\cup_{j \in \mathsf{q}, j' \in \mathsf{q}, i_1 \in [\mathsf{p}] \ \wedge \ j \neq j'} \left((\mathsf{tag}_{i_1}^{1,(j)}) \cap (\mathsf{tag}_{i_1}^{1,(j')}) \right).$$

  - If $|\mathcal{I}| > \mathsf{t}$, output $\perp$.

- **Computing cover free elements:**
  For $j \in \mathsf{q}$, let $\mathcal{K} = \mathcal{K}^* = \emptyset$. Let $\mathcal{K}^*$ be an ordered set.
  - Let $\mathcal{K}_1 = \cup_{i_2 \in [\mathsf{p}']} \mathsf{tag}_{i_2}^{2,(j)}$.
  - Pick any element $\mathsf{tag}_{i_2^*}^{2,(j)} \in \mathcal{K}_1 \setminus \mathcal{K} \in ([\mathsf{p}'] \times [\Delta])$. If no such element exists, then output $\perp$, otherwise, add $\mathsf{tag}_{i_2^*}^{2,(j)}$ to $\mathcal{K}^*$ such that $\mathcal{K}_j^* = \mathsf{tag}_{i_2^*}^{2,(j)}$.
  - Add $\mathcal{K}_1$ to set $\mathcal{K}$. i.e. $\mathcal{K} = \mathcal{K} \cup \mathcal{K}_1$.

- **Sampling polynomials:**
  - Pick ℓ random degree t polynomials $\mu_1(\cdot), \ldots, \mu_\ell(\cdot)$ where $\mu_i(\cdot)$ has constant term x_i.
  - Pick N unique nonzero field elements f_{tag^1} where $\mathsf{tag}^1 \in [\mathsf{p}] \times [\Gamma]$.
  - For $j \in \mathsf{q}$,
    - Sample a random degree $\mathsf{p} - 1$ polynomials $\eta_j(\cdot)$ with constant term $C_j(x)$.
  - For $\mathsf{tag}^2 \in ([\mathsf{p}'] \times [\Delta]) \setminus \mathcal{K}^*$,
    - Pick a random degree $\mathsf{p} - 1$ polynomials $\zeta_{\mathsf{tag}^2}(\cdot)$ with constant term 0.
  - For $j \in \mathsf{q}$,
    - Set $\zeta_{\mathcal{K}_j^*}(\cdot)$ such that the following equation is satisfied, $\forall \mathsf{tag}^1 \in [\mathsf{p}] \times [\Gamma]$,

$$\eta_j(f_{\mathsf{tag}^1}) = C_j(\mu_1(f_{\mathsf{tag}^1}), \ldots, \mu_\ell(f_{\mathsf{tag}^1})) + \sum_{i_2 \in [\mathsf{p}']} \zeta_{\mathsf{tag}_{i_2}^{2,(j)}}(f_i).$$

- Let $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}_3, \mathsf{ct}_4)$ be $(\emptyset, \emptyset, \emptyset, \emptyset)$. We assume that these sets are ordered sets.
- **Computing Ciphertexts:**
  For every possible $\mathsf{tag}^1 \in [\mathsf{p}] \times [\Gamma]$,
  - Garble the universal circuit $\hat{U} = U[\mu_1(f_{\mathsf{tag}^1}), \ldots, \mu_\ell(f_{\mathsf{tag}^1})]$ for each tag^1, i.e. $(\tilde{U}^{\mathsf{tag}^1}, \mathcal{L}^{\mathsf{tag}^1}) \leftarrow \mathsf{GC.Garble}(1^\kappa, \hat{U})$ where $\tilde{U}^{\mathsf{tag}^1}$ denotes the garbled circuit and $\mathcal{L}^{\mathsf{tag}^1}$ indicates the set of labels for each wire in this circuit.
  - For every possible $\mathsf{tag}^2 = (i, \delta) \in [\mathsf{p}'] \times [\Delta]$,
    - Let $\mathsf{input}_2 = \mathcal{L}^{\mathsf{tag}^1}_{\{Z_i\}, \{\zeta_{\mathsf{tag}^2}(f_{\mathsf{tag}^1})\}}$ where recall the $\mathcal{L}_{S,T}$ notation Definition 2.1 is used to choose the input labels for a garbled circuit for position S corresponding to the real input T.

* $\mathsf{ct}^1_{\mathsf{tag}^1,\mathsf{tag}^2} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_2))$.
* Add $\mathsf{ct}^1_{\mathsf{tag}^1,\mathsf{tag}^2}$ to the set ct^1.

– For every possible $\mathsf{tag}^3 = (i, b) \in [\lambda] \times \{0, 1\}$,
* Let $\mathsf{input}_3 = \mathcal{L}^{\mathsf{tag}^1}_{\{c_i\},\{b\}}$.
* $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_2, (\mathsf{tag}^1, \mathsf{tag}^3), (\mathsf{input}_3))$.
* Add $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3}$ to the set ct^2.

– Add the garbled circuit $\tilde{U}^{\mathsf{tag}^1}$ to the set ct^3.

– Add f_{tag^1} to the set ct^4.

- Output $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}_3, \mathsf{ct}_4)$.

4. **Guess:** $\mathcal{A}$ outputs a bit b.

Game 2. In this game, we rely on IBE security based on a case analysis if $\mathsf{tag}^1 \notin \mathcal{I}$.

1. Adversary $\mathcal{A}$ outputs circuit parameters $1^\lambda, 1^{\mathsf{D}}$ and collusion bound q.
2. **Setup Phase:...,Keygen query phase:...,Challenge message:....**
3. **Encryption phase:** $\mathcal{C}$ computes $\mathsf{S\text{-}FE.Enc}(\mathsf{mpk}, x, 1^{\mathsf{q}}) \to \mathsf{ct}$. It sends this to $\mathcal{A}$.

- **Parsing: ...,Computing colluding identities:...,Computing cover free elements:...,Sampling polynomials:....**
- Let $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}_3, \mathsf{ct}_4)$ be $(\emptyset, \emptyset, \emptyset, \emptyset)$. We assume that these sets are ordered sets.
- **Computing Ciphertexts:**
For every possible $\mathsf{tag}^1 \in [\mathsf{p}] \times [\Gamma]$,

– If $\mathsf{tag}^1 \in \mathcal{I}$, proceed according to the normal encryption, i.e.
* Garble the universal circuit $\hat{U} = U[\mu_1(f_{\mathsf{tag}^1}), \ldots, \mu_\ell(f_{\mathsf{tag}^1})]$ for each tag^1, i.e. $(\tilde{U}^{\mathsf{tag}^1}, \mathcal{L}^{\mathsf{tag}^1}) \leftarrow \mathsf{GC.Garble}(1^\kappa, \hat{U})$ where $\tilde{U}^{\mathsf{tag}^1}$ denotes the garbled circuit and $\mathcal{L}^{\mathsf{tag}^1}$ indicates the set of labels for each wire in this circuit.
* For every possible $\mathsf{tag}^2 = (i, \delta) \in [\mathsf{p}'] \times [\Delta]$,
· Let $\mathsf{input}_2 = \mathcal{L}^{\mathsf{tag}^1}_{\{Z_i\},\{\zeta_{\mathsf{tag}^2}(f_{\mathsf{tag}^1})\}}$ where recall the $\mathcal{L}_{S,T}$ notation Definition 2.1 is used to choose the input labels for a garbled circuit for position S corresponding to the real input T.
· $\mathsf{ct}^1_{\mathsf{tag}^1,\mathsf{tag}^2} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_2))$.
· Add $\mathsf{ct}^1_{\mathsf{tag}^1,\mathsf{tag}^2}$ to the set ct^1.
* For every possible $\mathsf{tag}^3 = (i, b) \in [\lambda] \times \{0, 1\}$,
· Let $\mathsf{input}_3 = \mathcal{L}^{\mathsf{tag}^1}_{\{c_i\},\{b\}}$.
· $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_2, (\mathsf{tag}^1, \mathsf{tag}^3), (\mathsf{input}_3))$.
· Add $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3}$ to the set ct^2.
* Add the garbled circuit $\tilde{U}^{\mathsf{tag}^1}$ to the set ct^3.
* Add f_{tag^1} to the set ct^4.

- Else if $\mathsf{tag}^1 \notin \mathcal{I}$, at most one of $\mathsf{tag}^{1,(1)}, \ldots, \mathsf{tag}^{1,(\mathsf{q})}$ contain tag^1, because of our construction of set $\mathcal{I}$. Let j^* be the set such that $\exists i_1^* \in [\mathsf{p}]$ where $\mathsf{tag}_{i_1^*}^{1,(j^*)} = \mathsf{tag}^1$. If no such j^* exists, then $j^* = \perp$.
- If $j^* \leq \mathsf{q}$,
  - Garble the universal circuit $\hat{U} = U[\mu_1(f_{\mathsf{tag}^1}), \ldots, \mu_\ell(f_{\mathsf{tag}^1})]$ for each tag^1, i.e. $(\tilde{U}^{\mathsf{tag}^1}, \mathcal{L}^{\mathsf{tag}^1}) \leftarrow \mathsf{GC.Garble}(1^\kappa, \hat{U})$ where $\tilde{U}^{\mathsf{tag}^1}$ denotes the garbled circuit and $\mathcal{L}^{\mathsf{tag}^1}$ indicates the set of labels for each wire in this circuit.
  - For every possible $\mathsf{tag}^2 = (i_2, \delta) \in [\mathsf{p}'] \times [\Delta]$,
    - Let $\mathsf{input}_2 = \mathcal{L}^{\mathsf{tag}^1}_{\{Z_{i_2}\}, \{\zeta_{\mathsf{tag}^2}(f_{\mathsf{tag}^1})\}}$ where recall the $\mathcal{L}_{S,T}$ notation Definition 2.1.
    - If $\mathsf{tag}_{i_2}^{2,(j^*)} = \mathsf{tag}^2$, then $\mathsf{ct}^1_{\mathsf{tag}^1,\mathsf{tag}^2} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_2))$.
    - Else, $\mathsf{ct}^1_{\mathsf{tag}^1,\mathsf{tag}^2} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), 0^{|\mathsf{input}_2|})$.
    - Add $\mathsf{ct}^1_{\mathsf{tag}^1,\mathsf{tag}^2}$ to the set ct^1.
  - For every possible $\mathsf{tag}^3 = (i, b) \in [\lambda] \times \{0,1\}$
    - Let $\mathsf{input}_3 = \mathcal{L}^{\mathsf{tag}^1}_{\{c_i\}, \{b\}}$.
    - If $C_{j^*,i} = b$, then $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_2, (\mathsf{tag}^1, \mathsf{tag}^3), (\mathsf{input}_3))$.
    - Else, $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_2, (\mathsf{tag}^1, \mathsf{tag}^3), 0^{|\mathsf{input}_3|})$.
    - Add $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3}$ to the set ct^2.
  - Add the garbled circuit $\tilde{U}^{\mathsf{tag}^1}$ to the set ct^3.
  - Add f_{tag^1} to the set ct^4.
- Else,
  - Garble the universal circuit $\hat{U} = U[\mu_1(f_{\mathsf{tag}^1}), \ldots, \mu_\ell(f_{\mathsf{tag}^1})]$ for each tag^1, i.e. $(\tilde{U}^{\mathsf{tag}^1}, \mathcal{L}^{\mathsf{tag}^1}) \leftarrow \mathsf{GC.Garble}(1^\kappa, \hat{U})$ where $\tilde{U}^{\mathsf{tag}^1}$ denotes the garbled circuit and $\mathcal{L}^{\mathsf{tag}^1}$ indicates the set of labels for each wire in this circuit.
  - For every possible $\mathsf{tag}^2 = (i, \delta) \in [\mathsf{p}'] \times [\Delta]$,
    - Let $\mathsf{input}_2 = \mathcal{L}^{\mathsf{tag}^1}_{\{Z_i\}, \{\zeta_{\mathsf{tag}^2}(f_{\mathsf{tag}^1})\}}$.
    - $\mathsf{ct}^1_{\mathsf{tag}^1,\mathsf{tag}^2} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), 0^{|\mathsf{input}_2|})$.
    - Add $\mathsf{ct}^1_{\mathsf{tag}^1,\mathsf{tag}^2}$ to the set ct^1.
  - For every possible $\mathsf{tag}^3 = (i, b) \in [\lambda] \times \{0,1\}$
    - Let $\mathsf{input}_3 = \mathcal{L}^{\mathsf{tag}^1}_{\{c_i\}, \{b\}}$.
    - $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_2, (\mathsf{tag}^1, \mathsf{tag}^3), 0^{|\mathsf{input}_3|})$
    - Add $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3}$ to the set ct^2.
  - Add the garbled circuit $\tilde{U}^{\mathsf{tag}^1}$ to the set ct^3.
  - Add f_{tag^1} to the set ct^4.

4. **Guess:** $\mathcal{A}$ outputs a bit b.

Game 3. In this game, we rely on garbled circuit security based on a case analysis if $\mathsf{tag}^1 \notin \mathcal{I}$.

1. Adversary $\mathcal{A}$ outputs circuit parameters $1^\lambda, 1^\mathsf{D}$ and collusion bound q.
2. **Setup Phase:...,Keygen query phase:...,Challenge message:....**
3. **Encryption phase:** $\mathcal{C}$ computes $\mathsf{S\text{-}FE.Enc}(\mathsf{mpk}, x, 1^\mathsf{q}) \to \mathsf{ct}$. It sends this to $\mathcal{A}$.
   - **Parsing: ...,Computing colluding identities:...,Computing cover free elements:...,Sampling polynomials:....**
   - Let $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}_3, \mathsf{ct}_4)$ be $(\emptyset, \emptyset, \emptyset, \emptyset)$. We assume that these sets are ordered sets.
   - **Computing Ciphertexts:**
     For every possible $\mathsf{tag}^1 \in [\mathsf{p}] \times [\Gamma]$,
     - If $\mathsf{tag}^1 \in \mathcal{I}$, proceed according to the normal encryption, i.e. ....
     - Else if $\mathsf{tag}^1 \notin \mathcal{I}$, ...
     - If $j^* \leq \mathsf{q}$,
       * Garble the universal circuit $\hat{U} = U[\mu_1(f_{\mathsf{tag}^1}), \ldots, \mu_\ell(f_{\mathsf{tag}^1})]$.
       * Let $(\tilde{U}^{\mathsf{tag}^1}, L) \leftarrow \mathsf{GC.Sim}(1^\kappa, 1^{(\lambda+\mathsf{p}')\cdot\log|\mathbb{F}|}, 1^{|\hat{U}|}, \eta_{j^*}(f_{\mathsf{tag}^1}))$ where $\tilde{U}^{\mathsf{tag}^1}$ denotes the garbled circuit and L are the input wires for the circuit.
       * For every possible $\mathsf{tag}^2 = (i_2, \delta) \in [\mathsf{p}'] \times [\Delta]$,
         · Let $\mathsf{input}_2 = L_{\{Z_{i_2}\}}$.
         · If $\mathsf{tag}_{i_2}^{2,(j^*)} = \mathsf{tag}^2$, then $\mathsf{ct}^1_{\mathsf{tag}^1,\mathsf{tag}^2} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_2))$.
         · Else, $\mathsf{ct}^1_{\mathsf{tag}^1,\mathsf{tag}^2} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), 0^{|\mathsf{input}_2|})$.
         · Add $\mathsf{ct}^1_{\mathsf{tag}^1,\mathsf{tag}^2}$ to the set ct^1.
       * For every possible $\mathsf{tag}^3 = (i, b) \in [\lambda] \times \{0,1\}$,
         · Let $\mathsf{input}_3 = L_{\{c_i\}}$.
         · If $C_{j^*,i} = b$, then $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_2, (\mathsf{tag}^1, \mathsf{tag}^3), (\mathsf{input}_3))$.
         · Else, $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_2, (\mathsf{tag}^1, \mathsf{tag}^3), 0^{|\mathsf{input}_3|})$.
         · Add $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3}$ to the set ct^2.
       * Add the garbled circuit $\tilde{U}^{\mathsf{tag}^1}$ to the set ct^3.
       * Add f_{tag^1} to the set ct^4.
     - Else,
       * Garble the universal circuit $\hat{U} = U[0, \ldots, 0]$ for each tag^1, i.e. $(\tilde{U}^{\mathsf{tag}^1}, \mathcal{L}^{\mathsf{tag}^1}) \leftarrow \mathsf{GC.Garble}(1^\kappa, \hat{U})$ where $\tilde{U}^{\mathsf{tag}^1}$ denotes the garbled circuit and $\mathcal{L}^{\mathsf{tag}^1}$ indicates the set of labels for each wire in this circuit.
       * For every possible $\mathsf{tag}^2 = (i, \delta) \in [\mathsf{p}'] \times [\Delta]$,
         · Let $\mathsf{input}_2 = \mathcal{L}^{\mathsf{tag}^1}_{\{Z_i\},\{\zeta_{\mathsf{tag}^2}(f_{\mathsf{tag}^1})\}}$.
         · $\mathsf{ct}^1_{\mathsf{tag}^1,\mathsf{tag}^2} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), 0^{|\mathsf{input}_2|})$.
         · Add $\mathsf{ct}^1_{\mathsf{tag}^1,\mathsf{tag}^2}$ to the set ct^1.
       * For every possible $\mathsf{tag}^3 = (i, b) \in [\lambda] \times \{0,1\}$

· Let $\mathsf{input}_3 = \mathcal{L}^{\mathsf{tag}^1}_{\{c_i\},\{b\}}$.

· $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, \emptyset, (\mathsf{tag}^1, \mathsf{tag}^3), 0^{|\mathsf{input}_3|})$.

· Add $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3}$ to the set ct^2.

∗ Add the garbled circuit $\tilde{U}^{\mathsf{tag}^1}$ to the set ct^3.

∗ Add f_{tag^1} to the set ct^4.

4. **Guess:** $\mathcal{A}$ outputs a bit b.

Game 4. This is the final ouput of the simulator, i.e. modify how the polynomials $\mu_1, \ldots, \mu_\ell$ are sampled.

1. Adversary $\mathcal{A}$ outputs circuit parameters $1^\lambda, 1^{\mathsf{D}}$ and collusion bound q.
2. **Setup Phase:. . . ,Keygen query phase:. . . ,Challenge message:. . . .**
3. **Encryption phase:** $\mathcal{C}$ computes $\mathsf{S\text{-}FE.Enc}(\mathsf{mpk}, x, 1^{\mathsf{q}}) \to \mathsf{ct}$. It sends this to $\mathcal{A}$.
   - **Parsing: . . . ,Computing colluding identities:. . . ,Computing cover free elements:. . . .**
   - **Sampling polynomials:**
     - Pick ℓ random degree t polynomials $\mu_1(\cdot), \ldots, \mu_\ell(\cdot)$ where $\mu_i(\cdot)$ has constant term 0.
     - Pick . . . .
   - Let $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}_3, \mathsf{ct}_4)$ be $(\emptyset, \emptyset, \emptyset, \emptyset)$. We assume that these sets are ordered sets.
   - **Computing Ciphertexts:. . . .**
4. **Guess:** $\mathcal{A}$ outputs a bit b.

4.2 Analysis.

Next, we show by a sequence of lemmas that no adversary can distinguish between any two adjacent games with non-negligible advantage. In the last game, we show that the advantage of any such adversary is negligible. We will let $\mathsf{adv}^x_{\mathcal{A}}$ denote the probability that $\mathcal{A}$ outputs bit 1 in Game x.

Lemma 4.1. $\forall \kappa \in \mathbb{N}$, $|\mathsf{adv}^0_{\mathcal{A}} - \mathsf{adv}^1_{\mathcal{A}}| = \mathsf{negl}(\kappa)$.

Proof. In game 0, ζ's are picked as polynomials with constant 0 and $\forall j \in [\mathsf{q}], \eta_j(\cdot)$ is defined by the relation,

$$\eta_j(f_i) = C_j(\mu_1(f_i), \ldots, \mu_\ell(f_i)) + \sum_{i_2 \in [\mathsf{p}']} \zeta_{\mathsf{tag}^{2,(j)}_{i_2}}(f_i).$$

Thus $\eta_j(\cdot)$ is a random polynomial with constant term $C_j(x)$.

In game 1, we output $\perp$ if the cover free property is not satisfied or if the set $\mathcal{I}$ is of cardinality greater than t. This happens with negligible probability because of Lemma A.2 and Lemma A.1. Additionally, $\eta_j(\cdot)$ is a random with constant $C_j(x)$ and the cover free element $\zeta_{\mathcal{K}^*_j}(\cdot)$ is set so that,

$$\eta_j(f_i) = C_j(\mu_1(f_i), \ldots, \mu_\ell(f_i)) + \sum_{i_2 \in [\mathsf{p}']} \zeta_{\mathsf{tag}^{2,(j)}_{i_2}}(f_i).$$

Since we have simply reversed the order in which we choose the polynomials and we choose them from the same distribution. Thus the only place the two games differ is when Game 1 outputs $\perp$ which is $\mathsf{negl}(\kappa)$. $\square$

Lemma 4.2. Assuming that IBE be a secure encryption scheme according to Definition 2.3. $\forall \kappa \in \mathbb{N}$, $|\mathsf{adv}^1_{\mathcal{A}} - \mathsf{adv}^2_{\mathcal{A}}| \leq \mathsf{negl}(\kappa)$.

Proof. In Game 1, every ciphertext is IBE encrypted on the input labels. In Game 2, we encrypt some ciphertexts to the all 0's string. We define a sequence of sub-hybrids where we change each IBE encryption one-by-one based on IBE security.

The sub-hybrids exist for each $\mathsf{tag}^1_* \in [\mathsf{p}] \times [\Gamma + 1]$, $\mathsf{tag}^2_* \in [\mathsf{p}'] \times [\Delta + 1]$ and $\mathsf{tag}^3_* \in [\lambda] \times \{0, 1, 2\}$. If we say $\mathsf{tag}^1 > \mathsf{tag}^1_*$, then we compare the two elements in $[\mathsf{p}] \times [\Gamma + 1]$ lexicographically. The elements in increasing order are $(1, 1), \ldots, (1, \Gamma), (2, 1), \ldots, (\mathsf{p}, \Gamma), (\mathsf{p}, \Gamma + 1)$. Additionally define the addition operation $\mathsf{tag}^1_* + 1$ to pick the next lexicographic element and operation $\mathsf{tag}^1_* - 1$ to pick the next previous lexicographic element. We similarly define the operations for iterators tag^2_* and tag^3_*.

Sub-Game $2^{\mathsf{tag}^1_*, \mathsf{tag}^2_*, \mathsf{tag}^3_*}$.

1. Adversary $\mathcal{A}$ outputs circuit parameters $1^\lambda, 1^{\mathsf{D}}$ and collusion bound q.
2. **Setup Phase:. . . ,Keygen query phase:. . . ,Challenge message:. . . .**
3. **Encryption phase:** $\mathcal{C}$ computes $\mathsf{S\text{-}FE.Enc}(\mathsf{mpk}, x, 1^{\mathsf{q}}) \to \mathsf{ct}$. It sends this to $\mathcal{A}$.
   - **Parsing: . . . ,Computing colluding identities:. . . ,Computing cover free elements:. . . ,Sampling polynomials:. . . .**
   - Let $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}_3, \mathsf{ct}_4)$ be $(\emptyset, \emptyset, \emptyset, \emptyset)$. We assume that these sets are ordered sets.
   - **Computing Ciphertexts:**
     For every possible $\mathsf{tag}^1 \in [\mathsf{p}] \times [\Gamma]$,
     - If $\mathsf{tag}^1 \in \mathcal{I}$ or $\mathsf{tag}^1 \geq \mathsf{tag}^1_*$, proceed according to the normal encryption, i.e. . . . .
     - Else if $\mathsf{tag}^1 \notin \mathcal{I}$, at most one of $\mathsf{tag}^{1,(1)}, \ldots, \mathsf{tag}^{1,(\mathsf{q})}$ contain tag^1, because of our construction of set $\mathcal{I}$. Let j^* be the set such that $\exists i^*_1 \in [\mathsf{p}]$ where $\mathsf{tag}^{1,(j^*)}_{i^*_1} = \mathsf{tag}^1$. If no such j^* exists, then $j^* = \perp$.
     - If $j^* \leq \mathsf{q}$,
       * Garble the universal circuit $\hat{U} = U[\mu_1(f_{\mathsf{tag}^1}), \ldots, \mu_\ell(f_{\mathsf{tag}^1})]$ for each tag^1, i.e. $(\tilde{U}^{\mathsf{tag}^1}, \mathcal{L}^{\mathsf{tag}^1}) \leftarrow \mathsf{GC.Garble}(1^\kappa, \hat{U})$ where $\tilde{U}^{\mathsf{tag}^1}$ denotes the garbled circuit and $\mathcal{L}^{\mathsf{tag}^1}$ indicates the set of labels for each wire in this circuit.
       * For every possible $\mathsf{tag}^2 = (i_2, \delta) \in [\mathsf{p}'] \times [\Delta]$,
         · Let $\mathsf{input}_2 = \mathcal{L}^{\mathsf{tag}^1}_{\{Z_{i_2}\}, \{\zeta_{\mathsf{tag}^2}(f_{\mathsf{tag}^1})\}}$ where recall the $\mathcal{L}_{S,T}$ notation Definition 2.1.
         · If $\mathsf{tag}^{2,(j^*)}_{i_2} = \mathsf{tag}^2$ or $\mathsf{tag}^2 \geq \mathsf{tag}^2_*$, then $\mathsf{ct}^1_{\mathsf{tag}^1, \mathsf{tag}^2} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_2))$.
         · Else, $\mathsf{ct}^1_{\mathsf{tag}^1, \mathsf{tag}^2} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), 0^{|\mathsf{input}_2|})$.
         · Add $\mathsf{ct}^1_{\mathsf{tag}^1, \mathsf{tag}^2}$ to the set ct^1.

* For every possible $\mathsf{tag}^3 = (i, b) \in [\lambda] \times \{0,1\}$
  · Let $\mathsf{input}_3 = \mathcal{L}^{\mathsf{tag}^1}_{\{c_i\},\{b\}}$.
  · If $C_{j^*,i} = b$ or $\mathsf{tag}^3 \geq \mathsf{tag}^3_*$, then $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_2, (\mathsf{tag}^1, \mathsf{tag}^3), (\mathsf{input}_3))$.
  · Else, $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_2, (\mathsf{tag}^1, \mathsf{tag}^3), 0^{|\mathsf{input}_3|})$.
  · Add $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3}$ to the set ct^2.
* Add the garbled circuit $\tilde{U}^{\mathsf{tag}^1}$ to the set ct^3.
* Add f_{tag^1} to the set ct^4.

– Else,
* Garble the universal circuit $\hat{U} = U[\mu_1(f_{\mathsf{tag}^1}), \ldots, \mu_\ell(f_{\mathsf{tag}^1})]$ for each tag^1, i.e. $(\tilde{U}^{\mathsf{tag}^1}, \mathcal{L}^{\mathsf{tag}^1}) \leftarrow \mathsf{GC.Garble}(1^\kappa, \hat{U})$ where $\tilde{U}^{\mathsf{tag}^1}$ denotes the garbled circuit and $\mathcal{L}^{\mathsf{tag}^1}$ indicates the set of labels for each wire in this circuit.
* For every possible $\mathsf{tag}^2 = (i, \delta) \in [\mathsf{p}'] \times [\Delta]$,
  · Let $\mathsf{input}_2 = \mathcal{L}^{\mathsf{tag}^1}_{\{Z_i\},\{\zeta_{\mathsf{tag}^2}(f_{\mathsf{tag}^1})\}}$.
  · If $\mathsf{tag}^2 \geq \mathsf{tag}^2_*$, $\mathsf{ct}^1_{\mathsf{tag}^1,\mathsf{tag}^2} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_2))$.
  · Else $\mathsf{ct}^1_{\mathsf{tag}^1,\mathsf{tag}^2} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), 0^{|\mathsf{input}_2|})$.
  · Add $\mathsf{ct}^1_{\mathsf{tag}^1,\mathsf{tag}^2}$ to the set ct^1.
* For every possible $\mathsf{tag}^3 = (i, b) \in [\lambda] \times \{0,1\}$
  · Let $\mathsf{input}_3 = \mathcal{L}^{\mathsf{tag}^1}_{\{c_i\},\{b\}}$.
  · If $\mathsf{tag}^3 \geq \mathsf{tag}^3_*$, $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_2, (\mathsf{tag}^1, \mathsf{tag}^3), (\mathsf{input}_3))$.
  · Else $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_2, (\mathsf{tag}^1, \mathsf{tag}^3), 0^{|\mathsf{input}_3|})$
  · Add $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3}$ to the set ct^2.
* Add the garbled circuit $\tilde{U}^{\mathsf{tag}^1}$ to the set ct^3.
* Add f_{tag^1} to the set ct^4.

4. **Guess:** $\mathcal{A}$ outputs a bit b.

We make the following observations.

- Game 1 is equivalent to Game $2^{(1,1),(1,1),(1,0)}$. As we have chosen $\mathsf{tag}^1_*, \mathsf{tag}^2_*, \mathsf{tag}^3_*$ as the lexicographically smallest elements, all encryptions are same as in Game 1.
- Game 2 is equivalent to Game $2^{(\mathsf{p},\Gamma),(\mathsf{p}',\Delta+1),(\lambda,2)}$. As we have chosen $\mathsf{tag}^1_*, \mathsf{tag}^2_*, \mathsf{tag}^3_*$ as the lexicographically largest elements, all switches in encryptions are made according to Game 2.
- Game $2^{\mathsf{tag}^1_*,(\mathsf{p}',\Delta+1),(\lambda,2)}$ is equivalent to Game $2^{\mathsf{tag}^1_*+1,(1,1),(1,0)}$. The construction of the sub-hybrid loops can be observed to verify this.

We write two separate reductions for the two different iterations. The first is when we go from Game $2^{\mathsf{tag}^1_*,\mathsf{tag}^2_*,\mathsf{tag}^3_*-1}$ to Game $2^{\mathsf{tag}^1_*,\mathsf{tag}^2_*,\mathsf{tag}^3_*}$ and the second is when we go from Game $2^{\mathsf{tag}^1_*,\mathsf{tag}^2_*-1,\mathsf{tag}^3_*}$ to Game $2^{\mathsf{tag}^1_*,\mathsf{tag}^2_*,\mathsf{tag}^3_*}$.

Lemma 4.3. Assuming that IBE be a secure encryption scheme according to Definition 2.3. $\forall \kappa \in \mathbb{N}$, $|\mathsf{adv}_{\mathcal{A}}^{2^{\mathsf{tag}_*^1,\mathsf{tag}_*^2,\mathsf{tag}_*^3-1}} - \mathsf{adv}_{\mathcal{A}}^{2^{\mathsf{tag}_*^1,\mathsf{tag}_*^2,\mathsf{tag}_*^3}}| \leq \mathsf{negl}(\kappa)$.

Proof. We first consider the different cases,

1. **Case 1:** If $\mathsf{tag}_*^1 \in \mathcal{I}$. Then the normal encryption procedure is called on both games and the games are identical.
2. **Case 2:** Else if $\mathsf{tag}_*^1 \notin \mathcal{I}$ and $j^* \leq \mathsf{q}$ and $C_{j^*,i} = b$. Then the normal encryption procedure is called on both games and the games are identical.
3. **Case 3:** Else if $\mathsf{tag}_*^1 \notin \mathcal{I}$ and $j^* \leq \mathsf{q}$ and $C_{j^*,i} \neq b$. Then we rely on reduction $\mathcal{B}_1$ that is described below.
4. **Case 4:** Else, we rely on reduction $\mathcal{B}_1$ that is described below.

Let $\mathcal{A}$ be an adversary that has non-negligible advantage in distingushing between the two games. We describe a reduction $\mathcal{B}_1$ that has non-negligible advantage in breaking IBE security between challenger $\mathcal{C}$ and $\mathcal{B}_1$.

Reduction $\mathcal{B}_1(1^\kappa)$:

1. $\mathcal{B}_1$ receives $\mathsf{IBE.pk}_2$ from Challenger $\mathcal{C}$ where identity space was set to $([\mathsf{p}]\times[\Gamma])\times([\lambda]\times\{0,1\})$.
2. Adversary $\mathcal{B}_1$ outputs circuit parameters $1^\lambda, 1^{\mathsf{D}}$ and collusion bound q to $\mathcal{A}$.
3. **Setup phase:**
   - Set the parameters $\mathsf{p} = \kappa \cdot \mathsf{D} + 1$, $\mathsf{p}' = \kappa$, $\Gamma = 2\mathsf{pq}^2$, $\Delta = 2\mathsf{q}$, $\mathsf{t} = \kappa$, $\mathsf{N} = \mathsf{p}\cdot\Gamma$, $\mathsf{N}' = \mathsf{p}'\cdot\Delta$. Let $\mathsf{pp} = (\mathsf{p}, \mathsf{p}', \Gamma, \Delta, \mathsf{t}, \mathsf{N}, \mathsf{N}', (1^\lambda, 1^{\mathsf{D}}, \mathsf{q}))$ be implicitly known to each algorithm.
   - $\mathsf{IBE.Setup}(1^\kappa, ([\mathsf{p}]\times[\Gamma])\times([\mathsf{p}']\times[\Delta])) \to (\mathsf{IBE.pk}_1, \mathsf{IBE.msk}_1)$[8].
   - Output $(\mathsf{mpk} = (\mathsf{IBE.pk}_1, \mathsf{IBE.pk}_2), \mathsf{msk} = (\mathsf{IBE.msk}_1, \perp))$.
4. **Keygen query phase:** $\mathcal{B}_1$ queries the keygen oracle $\mathsf{S\text{-}FE.KeyGen}(\mathsf{msk}, \cdot)$ and makes q queries, where the oracle computes as follows, $\mathsf{S\text{-}FE.KeyGen}(\mathsf{msk}, C) \to \mathsf{sk}_C$.
   - Sample $a_1, \ldots, a_{\mathsf{p}} \xleftarrow{R} [\Gamma]$.
   - Sample $b_1, \ldots, b_{\mathsf{p}'} \xleftarrow{R} [\Delta]$.
   - Let $C = C_1 \ldots C_\lambda$ be the bits of circuit C.
   - For $i \in [\mathsf{p}]$, let $\mathsf{tag}^1{}_i = (i, a_i)$ and for $i \in [\mathsf{p}']$ let $\mathsf{tag}^2{}_i = (i, b_i)$.
   - For $i \in [\lambda]$, let $\mathsf{tag}^3{}_i = (i, C_i)$.
   - Let $\mathsf{taglist}^1 = (\mathsf{tag}^1{}_1, \ldots, \mathsf{tag}^1{}_{\mathsf{p}})$, $\mathsf{taglist}^2 = (\mathsf{tag}^2{}_1, \ldots, \mathsf{tag}^2{}_{\mathsf{p}'})$, and $\mathsf{taglist}^3 = (\mathsf{tag}^3{}_1, \ldots, \mathsf{tag}^3{}_\lambda)$.
   - Let $\mathsf{sk}^1 = \mathsf{sk}^2 = \emptyset$. We assume that these sets are ordered sets.
     For $i_1 \in [\mathsf{p}]$,
     - For $i_2 \in [\mathsf{p}']$,

[8] Our IBE scheme takes identity space $\mathcal{I}_n$ as input where we can imagine $\mathcal{I}_n$ is a binary vector of length $n = \log(\mathsf{N}\cdot\mathsf{N}')$

* $\mathsf{sk}^1_{i_1,i_2} \leftarrow \mathsf{IBE.KeyGen}(\mathsf{IBE.msk}_1, (\mathsf{tag}^1{}_{i_1}, \mathsf{tag}^2{}_{i_2}))$.
* Add $\mathsf{sk}^1_{i_1,i_2}$ to the set sk^1.

– For $i_3 \in [\lambda]$,
* Let $\mathsf{sk}^2_{i_1,i_3}$ be the response of IBE challenger $\mathcal{C}$ when a key query is made on on identity $(\mathsf{tag}^1_{i_1}, \mathsf{tag}^3_{i_3})$.
* Add $\mathsf{sk}^2_{i_1,i_3}$ to the set sk^2.

- Output $\mathsf{sk}_C = (\mathsf{sk}^1, \mathsf{sk}^2, \mathsf{taglist}^1, \mathsf{taglist}^2, \mathsf{taglist}^3)$.

5. **Challenge message:** $\mathcal{A}$ outputs a challenge message $x \in \{0,1\}^\ell$.

6. **Encryption phase:** $\mathcal{B}_1$ computes $\mathsf{S\text{-}FE.Enc}(\mathsf{mpk}, x, 1^\mathsf{q}) \to \mathsf{ct}$. It sends this to $\mathcal{A}$.

- **Parsing: . . . ,Computing colluding identities:. . . ,Computing cover free elements:. . . ,Sampling polynomials:. . . .**
- Let $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}_3, \mathsf{ct}_4)$ be $(\emptyset, \emptyset, \emptyset, \emptyset)$. We assume that these sets are ordered sets.
- **Computing Ciphertexts:**
For every possible $\mathsf{tag}^1 \in [\mathsf{p}] \times [\Gamma]$,
– If $\mathsf{tag}^1 \in \mathcal{I}$ or $\mathsf{tag}^1 \geq \mathsf{tag}^1_*$, proceed according to the normal encryption, i.e. . . . .
– Else if $\mathsf{tag}^1 \notin \mathcal{I}$, at most one of $\mathsf{tag}^{1,(1)}, \ldots, \mathsf{tag}^{1,(\mathsf{q})}$ contain tag^1, because of our construction of set $\mathcal{I}$. Let j^* be the set such that $\exists i^*_1 \in [\mathsf{p}]$ where $\mathsf{tag}^{1,(j^*)}_{i^*_1} = \mathsf{tag}^1$. If no such j^* exists, then $j^* = \perp$.
– If $j^* \leq \mathsf{q}$,
* Garble the universal circuit $\hat{U} = U[\mu_1(f_{\mathsf{tag}^1}), \ldots, \mu_\ell(f_{\mathsf{tag}^1})]$ for each tag^1, i.e. $(\tilde{U}^{\mathsf{tag}^1}, \mathcal{L}^{\mathsf{tag}^1}) \leftarrow \mathsf{GC.Garble}(1^\kappa, \hat{U})$ where $\tilde{U}^{\mathsf{tag}^1}$ denotes the garbled circuit and $\mathcal{L}^{\mathsf{tag}^1}$ indicates the set of labels for each wire in this circuit.
* For every possible $\mathsf{tag}^2 = (i_2, \delta) \in [\mathsf{p}'] \times [\Delta]$,
· Let $\mathsf{input}_2 = \mathcal{L}^{\mathsf{tag}^1}_{\{Z_{i_2}\},\{\zeta_{\mathsf{tag}^2}(f_{\mathsf{tag}^1})\}}$ where recall the $\mathcal{L}_{S,T}$ notation Definition 2.1.
· If $\mathsf{tag}^{2,(j^*)}_{i_2} = \mathsf{tag}^2$ or $\mathsf{tag}^2 \geq \mathsf{tag}^2_*$, then $\mathsf{ct}^1_{\mathsf{tag}^1,\mathsf{tag}^2} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_2))$.
· Else, $\mathsf{ct}^1_{\mathsf{tag}^1,\mathsf{tag}^2} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), 0^{|\mathsf{input}_2|})$.
· Add $\mathsf{ct}^1_{\mathsf{tag}^1,\mathsf{tag}^2}$ to the set ct^1.
* For every possible $\mathsf{tag}^3 = (i, b) \in [\lambda] \times \{0,1\}$
· Let $\mathsf{input}_3 = \mathcal{L}^{\mathsf{tag}^1}_{\{c_i\},\{b\}}$.
· If $C_{j^*,i} = b$ or $\mathsf{tag}^3 \geq \mathsf{tag}^3_*$, then $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_2, (\mathsf{tag}^1, \mathsf{tag}^3), (\mathsf{input}_3))$.
· Else, $\mathcal{B}_1$ sends $(\mathsf{input}_3, 0^{|\mathsf{input}_3|}$ as the challenge message to $\mathcal{C}$ and $(\mathsf{tag}^1, \mathsf{tag}^3)$ as the challenge identity. Let ct^* be the ciphertext received by $\mathcal{B}_1$, then, $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3} = \mathsf{ct}^*$.
· Add $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3}$ to the set ct^2.
* Add the garbled circuit $\tilde{U}^{\mathsf{tag}^1}$ to the set ct^3.
* Add f_{tag^1} to the set ct^4.

– Else,
  * Garble the universal circuit $\hat{U} = U[\mu_1(f_{\mathsf{tag}^1}), \ldots, \mu_\ell(f_{\mathsf{tag}^1})]$ for each tag^1, i.e. $(\tilde{U}^{\mathsf{tag}^1}, \mathcal{L}^{\mathsf{tag}^1}) \leftarrow \mathsf{GC.Garble}(1^\kappa, \hat{U})$ where $\tilde{U}^{\mathsf{tag}^1}$ denotes the garbled circuit and $\mathcal{L}^{\mathsf{tag}^1}$ indicates the set of labels for each wire in this circuit.
  * For every possible $\mathsf{tag}^2 = (i, \delta) \in [\mathsf{p}'] \times [\Delta]$,
    · Let $\mathsf{input}_2 = \mathcal{L}^{\mathsf{tag}^1}_{\{Z_i\},\{\zeta_{\mathsf{tag}^2}(f_{\mathsf{tag}^1})\}}$.
    · If $\mathsf{tag}^2 \geq \mathsf{tag}^2_*$, $\mathsf{ct}^1_{\mathsf{tag}^1,\mathsf{tag}^2} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_2))$.
    · Else $\mathsf{ct}^1_{\mathsf{tag}^1,\mathsf{tag}^2} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), 0^{|\mathsf{input}_2|})$.
    · Add $\mathsf{ct}^1_{\mathsf{tag}^1,\mathsf{tag}^2}$ to the set ct^1.
  * For every possible $\mathsf{tag}^3 = (i, b) \in [\lambda] \times \{0, 1\}$
    · Let $\mathsf{input}_3 = \mathcal{L}^{\mathsf{tag}^1}_{\{c_i\},\{b\}}$.
    · If $\mathsf{tag}^3 \geq \mathsf{tag}^3_*$, $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_2, (\mathsf{tag}^1, \mathsf{tag}^3), (\mathsf{input}_3))$.
    · Else, $\mathcal{B}_1$ sends $(\mathsf{input}_3, 0^{|\mathsf{input}_3|}$ as the challenge message to $\mathcal{C}$ and $(\mathsf{tag}^1, \mathsf{tag}^3)$ as the challenge identity. Let ct^* be the ciphertext received by $\mathcal{B}_1$, then, $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3} = \mathsf{ct}^*$.
    · Add $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3}$ to the set ct^2.
  * Add the garbled circuit $\tilde{U}^{\mathsf{tag}^1}$ to the set ct^3.
  * Add f_{tag^1} to the set ct^4.

7. **Guess:** $\mathcal{B}_1$ outputs a bit b which is output by the adversary $\mathcal{A}$.

We note that $\mathcal{B}_1$ is an admissable IBE adversary as when we are in Case 3, the identity given out in prior keygen's to $\mathcal{C}$ doesn't match the challenge identity as $C_{j^*,i} \neq b$. Similarly in Case 4 as $j^* = \perp$, the challenge identity has a different tag^1 than all the prior queries from consruction of set $\mathcal{I}$. It is easy to notice that if $\mathcal{A}$ has non-negligible advantage in distinguishing the games, then $\mathcal{B}_1$ has non-negligible advantage. This violates IBE security from Definition 2.3. □

Lemma 4.4. Assuming that IBE be a secure encryption scheme according to Definition 2.3. $\forall \kappa \in \mathbb{N}$, $|\mathsf{adv}^{2^{\mathsf{tag}^1_*,\mathsf{tag}^2_*-1,\mathsf{tag}^3_*}}_{\mathcal{A}} - \mathsf{adv}^{2^{\mathsf{tag}^1_*,\mathsf{tag}^2_*,\mathsf{tag}^3_*}}_{\mathcal{A}}| \leq \mathsf{negl}(\kappa)$.

Proof. We first consider the different cases,

1. **Case 1:** If $\mathsf{tag}^1_* \in \mathcal{I}$. Then the normal encryption procedure is called on both games and the games are identical.

2. **Case 2:** Else if $\mathsf{tag}^1_* \notin \mathcal{I}$ and $j^* \leq \mathsf{q}_1$ and $\mathsf{tag}^{2,(j^*)}_{i_2} = \mathsf{tag}^2_*$. Then the normal encryption procedure is called on both games and the games are identical.

3. **Case 3:** Else if $\mathsf{tag}^1_* \notin \mathcal{I}$ and $j^* \leq \mathsf{q}_1$ and $\mathsf{tag}^{2,(j^*)}_{i_2} \neq \mathsf{tag}^2_*$. Then we rely on reduction $\mathcal{B}_2$ that is described below.

4. **Case 4:** Else, we rely on reduction $\mathcal{B}_2$ that is described below.

Let $\mathcal{A}$ be an adversary that has non-negligible advantage in distingushing between the two games. We describe a reduction $\mathcal{B}_2$ that has non-negligible advantage in breaking IBE security between challenger $\mathcal{C}$ and $\mathcal{B}_2$.

Reduction $\mathcal{B}_2(1^\kappa)$:

1. $\mathcal{B}_2$ receives $\mathsf{IBE.pk}_1$ from Challenger $\mathcal{C}$ where identity space was set to $([\mathsf{p}] \times [\Gamma]) \times ([\mathsf{p}'] \times [\Delta])$.
2. Adversary $\mathcal{B}_2$ outputs circuit parameters $1^\lambda, 1^{\mathsf{D}}$ and collusion bound q to $\mathcal{A}$.
3. **Setup phase:**
   - Set the parameters $\mathsf{p} = \kappa \cdot \mathsf{D} + 1$, $\mathsf{p}' = \kappa$, $\Gamma = 2\mathsf{pq}^2$, $\Delta = 2\mathsf{q}$, $\mathsf{t} = \kappa$, $\mathsf{N} = \mathsf{p} \cdot \Gamma$, $\mathsf{N}' = \mathsf{p}' \cdot \Delta$. Let $\mathsf{pp} = (\mathsf{p}, \mathsf{p}', \Gamma, \Delta, \mathsf{t}, \mathsf{N}, \mathsf{N}', (1^\lambda, 1^{\mathsf{D}}, \mathsf{q}))$ be implicitly known to each algorithm.
   - $\mathsf{IBE.Setup}(1^\kappa, ([\mathsf{p}] \times [\Gamma]) \times ([\lambda] \times \{0,1\})) \to (\mathsf{IBE.pk}_2, \mathsf{IBE.msk}_2)$.
   - Output $(\mathsf{mpk} = (\mathsf{IBE.pk}_1, \mathsf{IBE.pk}_2), \mathsf{msk} = (\perp, \mathsf{IBE.msk}_2))$.
4. **Keygen query phase:** $\mathcal{B}_2$ queries the keygen oracle $\mathsf{S\text{-}FE.KeyGen}(\mathsf{msk}, \cdot)$ and makes q queries, where the oracle computes as follows, $\mathsf{S\text{-}FE.KeyGen}(\mathsf{msk}, C) \to \mathsf{sk}_C$.
   - Sample $a_1, \ldots, a_{\mathsf{p}} \xleftarrow{R} [\Gamma]$.
   - Sample $b_1, \ldots, b_{\mathsf{p}'} \xleftarrow{R} [\Delta]$.
   - Let $C = C_1 \ldots C_\lambda$ be the bits of circuit C.
   - For $i \in [\mathsf{p}]$, let $\mathsf{tag}^1{}_i = (i, a_i)$ and for $i \in [\mathsf{p}']$ let $\mathsf{tag}^2{}_i = (i, b_i)$.
   - For $i \in [\lambda]$, let $\mathsf{tag}^3{}_i = (i, C_i)$.
   - Let $\mathsf{taglist}^1 = (\mathsf{tag}^1{}_1, \ldots, \mathsf{tag}^1{}_{\mathsf{p}})$, $\mathsf{taglist}^2 = (\mathsf{tag}^2{}_1, \ldots, \mathsf{tag}^2{}_{\mathsf{p}'})$, and $\mathsf{taglist}^3 = (\mathsf{tag}^3{}_1, \ldots, \mathsf{tag}^3{}_\lambda)$.
   - Let $\mathsf{sk}^1 = \mathsf{sk}^2 = \emptyset$. We assume that these sets are ordered sets.
     For $i_1 \in [\mathsf{p}]$,
     - For $i_2 \in [\mathsf{p}']$,
       - Let $\mathsf{sk}^2_{i_1,i_2}$ be the response of IBE challenger $\mathcal{C}$ when a key query is made on on identity $(\mathsf{tag}^1_{i_1}, \mathsf{tag}^2_{i_2})$.
       - Add $\mathsf{sk}^1_{i_1,i_2}$ to the set sk^1.
     - For $i_3 \in [\lambda]$,
       - $\mathsf{sk}^2_{i_1,i_3} \leftarrow \mathsf{IBE.KeyGen}(\mathsf{IBE.msk}_2, (\mathsf{tag}^1{}_{i_1}, \mathsf{tag}^3{}_{i_3}))$.
       - Add $\mathsf{sk}^2_{i_1,i_3}$ to the set sk^2.
   - Output $\mathsf{sk}_C = (\mathsf{sk}^1, \mathsf{sk}^2, \mathsf{taglist}^1, \mathsf{taglist}^2, \mathsf{taglist}^3)$.
5. **Challenge message:** $\mathcal{A}$ outputs a challenge message $x \in \{0,1\}^\ell$.
6. **Encryption phase:** $\mathcal{B}_2$ computes $\mathsf{S\text{-}FE.Enc}(\mathsf{mpk}, x, 1^{\mathsf{q}}) \to \mathsf{ct}$. It sends this to $\mathcal{A}$.
   - **Parsing: . . . ,Computing colluding identities:. . . ,Computing cover free elements:. . . ,Sampling polynomials:. . . .**
   - Let $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}_3, \mathsf{ct}_4)$ be $(\emptyset, \emptyset, \emptyset, \emptyset)$. We assume that these sets are ordered sets.

- **Computing Ciphertexts:**
  For every possible $\mathsf{tag}^1 \in [\mathsf{p}] \times [\Gamma]$,
  - If $\mathsf{tag}^1 \in \mathcal{I}$ or $\mathsf{tag}^1 \geq \mathsf{tag}^1_*$, proceed according to the normal encryption, i.e. ....
  - Else if $\mathsf{tag}^1 \notin \mathcal{I}$, at most one of $\mathsf{tag}^{1,(1)}, \ldots, \mathsf{tag}^{1,(\mathsf{q})}$ contain tag^1, because of our construction of set $\mathcal{I}$. Let j^* be the set such that $\exists i_1^* \in [\mathsf{p}]$ where $\mathsf{tag}^{1,(j^*)}_{i_1^*} = \mathsf{tag}^1$. If no such j^* exists, then $j^* = \perp$.
  - If $j^* \leq \mathsf{q}$,
    * Garble the universal circuit $\hat{U} = U[\mu_1(f_{\mathsf{tag}^1}), \ldots, \mu_\ell(f_{\mathsf{tag}^1})]$ for each tag^1, i.e. $(\tilde{U}^{\mathsf{tag}^1}, \mathcal{L}^{\mathsf{tag}^1}) \leftarrow \mathsf{GC.Garble}(1^\kappa, \hat{U})$ where $\tilde{U}^{\mathsf{tag}^1}$ denotes the garbled circuit and $\mathcal{L}^{\mathsf{tag}^1}$ indicates the set of labels for each wire in this circuit.
    * For every possible $\mathsf{tag}^2 = (i_2, \delta) \in [\mathsf{p}'] \times [\Delta]$,
      · Let $\mathsf{input}_2 = \mathcal{L}^{\mathsf{tag}^1}_{\{Z_{i_2}\},\{\zeta_{\mathsf{tag}^2}(f_{\mathsf{tag}^1})\}}$ where recall the $\mathcal{L}_{S,T}$ notation Definition 2.1.
      · If $\mathsf{tag}^{2,(j^*)}_{i_2} = \mathsf{tag}^2$ or $\mathsf{tag}^2 \geq \mathsf{tag}^2_*$, then $\mathsf{ct}^1_{\mathsf{tag}^1,\mathsf{tag}^2} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_2))$.
      · Else, $\mathcal{B}_2$ sends $(\mathsf{input}_2, 0^{|\mathsf{input}_2|}$ as the challenge message to $\mathcal{C}$ and $(\mathsf{tag}^1, \mathsf{tag}^2)$ as the challenge identity. Let ct^* be the ciphertext received by $\mathcal{B}_2$, then, $\mathsf{ct}^1_{\mathsf{tag}^1,\mathsf{tag}^2} = \mathsf{ct}^*$.
      · Add $\mathsf{ct}^1_{\mathsf{tag}^1,\mathsf{tag}^2}$ to the set ct^1.
    * For every possible $\mathsf{tag}^3 = (i, b) \in [\lambda] \times \{0,1\}$
      · Let $\mathsf{input}_3 = \mathcal{L}^{\mathsf{tag}^1}_{\{c_i\},\{b\}}$.
      · If $C_{j^*,i} = b$ or $\mathsf{tag}^3 \geq \mathsf{tag}^3_*$, then $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_2, (\mathsf{tag}^1, \mathsf{tag}^3), (\mathsf{input}_3))$.
      · Else, $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_2, (\mathsf{tag}^1, \mathsf{tag}^3), 0^{|\mathsf{input}_3|})$.
      · Add $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3}$ to the set ct^2.
    * Add the garbled circuit $\tilde{U}^{\mathsf{tag}^1}$ to the set ct^3.
    * Add f_{tag^1} to the set ct^4.
  - Else,
    * Garble the universal circuit $\hat{U} = U[\mu_1(f_{\mathsf{tag}^1}), \ldots, \mu_\ell(f_{\mathsf{tag}^1})]$ for each tag^1, i.e. $(\tilde{U}^{\mathsf{tag}^1}, \mathcal{L}^{\mathsf{tag}^1}) \leftarrow \mathsf{GC.Garble}(1^\kappa, \hat{U})$ where $\tilde{U}^{\mathsf{tag}^1}$ denotes the garbled circuit and $\mathcal{L}^{\mathsf{tag}^1}$ indicates the set of labels for each wire in this circuit.
    * For every possible $\mathsf{tag}^2 = (i, \delta) \in [\mathsf{p}'] \times [\Delta]$,
      · Let $\mathsf{input}_2 = \mathcal{L}^{\mathsf{tag}^1}_{\{Z_i\},\{\zeta_{\mathsf{tag}^2}(f_{\mathsf{tag}^1})\}}$.
      · If $\mathsf{tag}^2 \geq \mathsf{tag}^2_*$, $\mathsf{ct}^1_{\mathsf{tag}^1,\mathsf{tag}^2} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_2))$.
      · Else, $\mathcal{B}_2$ sends $(\mathsf{input}_2, 0^{|\mathsf{input}_2|}$ as the challenge message to $\mathcal{C}$ and $(\mathsf{tag}^1, \mathsf{tag}^2)$ as the challenge identity. Let ct^* be the ciphertext received by $\mathcal{B}_2$, then, $\mathsf{ct}^1_{\mathsf{tag}^1,\mathsf{tag}^2} = \mathsf{ct}^*$.
      · Add $\mathsf{ct}^1_{\mathsf{tag}^1,\mathsf{tag}^2}$ to the set ct^1.
    * For every possible $\mathsf{tag}^3 = (i, b) \in [\lambda] \times \{0,1\}$
      · Let $\mathsf{input}_3 = \mathcal{L}^{\mathsf{tag}^1}_{\{c_i\},\{b\}}$.

- If $\mathsf{tag}^3 \geq \mathsf{tag}^3_*$, $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_2, (\mathsf{tag}^1, \mathsf{tag}^3), (\mathsf{input}_3))$.
- Else, $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_2, (\mathsf{tag}^1, \mathsf{tag}^3), 0^{|\mathsf{input}_3|})$.
- Add $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3}$ to the set ct^2.

* Add the garbled circuit $\tilde{U}^{\mathsf{tag}^1}$ to the set ct^3.
* Add f_{tag^1} to the set ct^4.

7. **Guess:** $\mathcal{B}_2$ outputs a bit b which is output by the adversary $\mathcal{A}$.

We note that $\mathcal{B}_2$ is an admissable IBE adversary as when we are in Case 3, the identity given out in prior keygen's to $\mathcal{C}$ doesn't match the challenge identity as $\mathsf{tag}^{2,(j^*)}_{i_2} \neq \mathsf{tag}^2_*$. Similarly in Case 4 as $j^* = \perp$, the challenge identity has a different tag^1 than all the prior queries from consruction of set $\mathcal{I}$. It is easy to notice that if $\mathcal{A}$ has non-negligible advantage in distinguishing the games, then $\mathcal{B}_2$ has non-negligible advantage. This violates IBE security from Definition 2.3. □

Following the sequence of polynomial in κ many sub-hybrid proofs, we have that any two adjacent games have negligible advantage. Thus the advantage for an adversary in distinguishing Games 1 and 2 is negligible. □

Lemma 4.5. Assuming that GC is a secure garbled circuits scheme according to Definition 2.2. $\forall \kappa \in \mathbb{N}$, $|\mathsf{adv}^2_{\mathcal{A}} - \mathsf{adv}^3_{\mathcal{A}}| \leq \mathsf{negl}(\kappa)$.

Proof. In Game 3, garble circuits are simulated . We define a sequence of sub-hybrids where we change each garbling procedure one-by-one based on garbled circuit security.

The sub-hybrids exist for each $\mathsf{tag}^1_* \in [\mathsf{p}] \times [\Gamma + 1]$.

Sub-Game $3^{\mathsf{tag}^1_*}$.

1. Adversary $\mathcal{A}$ outputs circuit parameters $1^\lambda, 1^{\mathsf{D}}$ and collusion bound q.
2. **Setup Phase:...,Keygen query phase:...,Challenge message:....**
3. **Encryption phase:** $\mathcal{C}$ computes $\mathsf{S\text{-}FE.Enc}(\mathsf{mpk}, x, 1^{\mathsf{q}}) \to \mathsf{ct}$. It sends this to $\mathcal{A}$.

- **Parsing: ...,Computing colluding identities:...,Computing cover free elements:...,Sampling polynomials:....**
- Let $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}_3, \mathsf{ct}_4)$ be $(\emptyset, \emptyset, \emptyset, \emptyset)$. We assume that these sets are ordered sets.
- **Computing Ciphertexts:**
  For every possible $\mathsf{tag}^1 \in [\mathsf{p}] \times [\Gamma]$,
  - If $\mathsf{tag}^1 \in \mathcal{I}$, proceed according to the normal encryption, i.e. ....
  - Else if $\mathsf{tag}^1 \notin \mathcal{I}$, ...
  - If $j^* \leq \mathsf{q}$,
    * If $\mathsf{tag}^1 \geq \mathsf{tag}^1_*$, Garble the universal circuit $\hat{U} = U[\mu_1(f_{\mathsf{tag}^1}), \ldots, \mu_\ell(f_{\mathsf{tag}^1})]$ for each tag^1, i.e. $(\tilde{U}^{\mathsf{tag}^1}, \mathcal{L}^{\mathsf{tag}^1}) \leftarrow \mathsf{GC.Garble}(1^\kappa, \hat{U})$ where $\tilde{U}^{\mathsf{tag}^1}$ denotes the garbled circuit and $\mathcal{L}^{\mathsf{tag}^1}$ indicates the set of labels for each wire in this circuit.

* Else Garble the universal circuit $\hat{U} = U[\mu_1(f_{\mathsf{tag}^1}), \ldots, \mu_\ell(f_{\mathsf{tag}^1})]$. Let $(\tilde{U}^{\mathsf{tag}^1}, L) \leftarrow \mathsf{GC.Sim}(1^\kappa, 1^{(\lambda+\mathsf{p}')(\log|\mathbb{F}|)}, 1^{|\hat{U}|}, \eta_{j^*}(f_{\mathsf{tag}^1}))$ where $\tilde{U}^{\mathsf{tag}^1}$ denotes the garbled circuit and L are the input wires for the circuit.
* For every possible $\mathsf{tag}^2 = (i_2, \delta) \in [\mathsf{p}'] \times [\Delta]$,
  - If $\mathsf{tag}^1 \geq \mathsf{tag}^1_*$, let $\mathsf{input}_2 = \mathcal{L}^{\mathsf{tag}^1}_{\{Z_{i_2}\},\{\zeta_{\mathsf{tag}^2}(f_{\mathsf{tag}^1})\}}$.
  - Else, let $\mathsf{input}_2 = L_{\{Z_{i_2}\}}$ (Definition 2.1).
  - If $\mathsf{tag}^{2,(j^*)}_{i_2} = \mathsf{tag}^2$, then $\mathsf{ct}^1_{\mathsf{tag}^1,\mathsf{tag}^2} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_2))$.
  - Else, $\mathsf{ct}^1_{\mathsf{tag}^1,\mathsf{tag}^2} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), 0^{|\mathsf{input}_2|})$.
  - Add $\mathsf{ct}^1_{\mathsf{tag}^1,\mathsf{tag}^2}$ to the set ct^1.
* For every possible $\mathsf{tag}^3 = (i, b) \in [\lambda] \times \{0, 1\}$,
  - If $\mathsf{tag}^1 \geq \mathsf{tag}^1_*$, let $\mathsf{input}_3 = \mathcal{L}^{\mathsf{tag}^1}_{\{c_i\},\{b\}}$.
  - Else let $\mathsf{input}_3 = L_{\{c_i\}}$.
  - If $C_{j^*,i} = b$, then $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_2, (\mathsf{tag}^1, \mathsf{tag}^3), (\mathsf{input}_3))$.
  - Else, $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_2, (\mathsf{tag}^1, \mathsf{tag}^3), 0^{|\mathsf{input}_3|})$.
  - Add $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3}$ to the set ct^2.
* Add the garbled circuit $\tilde{U}^{\mathsf{tag}^1}$ to the set ct^3.
* Add f_{tag^1} to the set ct^4.

– Else,
* If $\mathsf{tag}^1 \geq \mathsf{tag}^1_*$, Garble the universal circuit $\hat{U} = U[\mu_1(f_{\mathsf{tag}^1}), \ldots, \mu_\ell(f_{\mathsf{tag}^1})]$ for each tag^1, i.e. $(\tilde{U}^{\mathsf{tag}^1}, \mathcal{L}^{\mathsf{tag}^1}) \leftarrow \mathsf{GC.Garble}(1^\kappa, \hat{U})$ where $\tilde{U}^{\mathsf{tag}^1}$ denotes the garbled circuit and $\mathcal{L}^{\mathsf{tag}^1}$ indicates the set of labels for each wire in this circuit.
* Else, Garble the universal circuit $\hat{U} = U[0, \ldots, 0]$ for each tag^1, i.e. $(\tilde{U}^{\mathsf{tag}^1}, \mathcal{L}^{\mathsf{tag}^1}) \leftarrow \mathsf{GC.Garble}(1^\kappa, \hat{U})$ where $\tilde{U}^{\mathsf{tag}^1}$ denotes the garbled circuit and $\mathcal{L}^{\mathsf{tag}^1}$ indicates the set of labels for each wire in this circuit.
* For every possible $\mathsf{tag}^2 = (i, \delta) \in [\mathsf{p}'] \times [\Delta]$,
  - Let $\mathsf{input}_2 = \mathcal{L}^{\mathsf{tag}^1}_{\{Z_i\},\{\zeta_{\mathsf{tag}^2}(f_{\mathsf{tag}^1})\}}$.
  - $\mathsf{ct}^1_{\mathsf{tag}^1,\mathsf{tag}^2} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), 0^{|\mathsf{input}_2|})$.
  - Add $\mathsf{ct}^1_{\mathsf{tag}^1,\mathsf{tag}^2}$ to the set ct^1.
* For every possible $\mathsf{tag}^3 = (i, b) \in [\lambda] \times \{0, 1\}$
  - Let $\mathsf{input}_3 = \mathcal{L}^{\mathsf{tag}^1}_{\{c_i\},\{b\}}$.
  - $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, \emptyset, (\mathsf{tag}^1, \mathsf{tag}^3), 0^{|\mathsf{input}_3|})$.
  - Add $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3}$ to the set ct^2.
* Add the garbled circuit $\tilde{U}^{\mathsf{tag}^1}$ to the set ct^3.
* Add f_{tag^1} to the set ct^4.

4. **Guess:** $\mathcal{A}$ outputs a bit b.

We make the following observations.

- Game 2 is equivalent to Game $3^{(1,1)}$. As we have chosen tag^1_* as the lexicographically smallest elements, all encryptions are same as in Game 2.
- Game 3 is equivalent to Game $2^{(\mathsf{p},\Gamma+1)}$. As we have chosen tag^1_* as the lexicographically largest elements, all switches in encryptions are made according to Game 3.

Lemma 4.6. Assuming that GC is a secure garbled circuits scheme according to Definition 2.2. $\forall \kappa \in \mathbb{N}$, $|\mathsf{adv}_{\mathcal{A}}^{3^{\mathsf{tag}^1_*-1}} - \mathsf{adv}_{\mathcal{A}}^{3^{\mathsf{tag}^1_*}}| \leq \mathsf{negl}(\kappa)$.

Proof. We first consider the different cases,

1. **Case 1:** If $\mathsf{tag}^1_* \in \mathcal{I}$. Then the normal garbling procedure is called on both games and the games are identical.
2. **Case 2:** Else if $\mathsf{tag}^1_* \notin \mathcal{I}$ and $j^* \leq \mathsf{q}$. We rely on reduction $\mathcal{B}_3$ that is described below.
3. **Case 3:** Else, observe that all the labels of the garbled circuit aren't being used in encryption. Thus we can switch the circuits to garble with hardcoded $0's$ rather than $\mu_1, \ldots, \mu_\ell$. Here we rely on the fact that in Corollary 2.1, the hardcoded values are hidden if no labels are revealed.

Let $\mathcal{A}$ be an adversary that has non-negligible advantage in distingushing between the two games. We describe a reduction $\mathcal{B}_3$ that has non-negligible advantage in breaking garbled circuit security between challenger $\mathcal{C}$ and $\mathcal{B}_3$.

Reduction $\mathcal{B}_3(1^\kappa)$:

1. Adversary $\mathcal{B}_3$ outputs circuit parameters $1^\lambda, 1^{\mathsf{D}}$ and collusion bound q.
2. **Setup Phase:...,Keygen query phase:...,Challenge message:....**
3. **Encryption phase:** $\mathcal{C}$ computes $\mathsf{S\text{-}FE.Enc}(\mathsf{mpk}, x, 1^{\mathsf{q}}) \to \mathsf{ct}$. It sends this to $\mathcal{A}$.
   - **Parsing: ...,Computing colluding identities:...,Computing cover free elements:...,Sampling polynomials:....**
   - Let $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}_3, \mathsf{ct}_4)$ be $(\emptyset, \emptyset, \emptyset, \emptyset)$. We assume that these sets are ordered sets.
   - **Computing Ciphertexts:**
     For every possible $\mathsf{tag}^1 \in [\mathsf{p}] \times [\Gamma]$,
     - If $\mathsf{tag}^1 \in \mathcal{I}$, proceed according to the normal encryption, i.e. ....
     - Else if $\mathsf{tag}^1 \notin \mathcal{I}$, ...
     - If $j^* \leq \mathsf{q}$,
       * $\mathcal{B}_3$ receives $(\tilde{U}^{\mathsf{tag}^1}, L)$ from the challenger $\mathcal{C}$ where the garbling circuit input was $\hat{U} = U[\mu_1(f_{\mathsf{tag}^1}), \ldots, \mu_\ell(f_{\mathsf{tag}^1})]$.
       * For every possible $\mathsf{tag}^2 = (i_2, \delta) \in [\mathsf{p}'] \times [\Delta]$,
         · Let $\mathsf{input}_2 = L_{\{Z_{i_2}\}}$.
         · If $\mathsf{tag}_{i_2}^{2,(j^*)} = \mathsf{tag}^2$, then $\mathsf{ct}^1_{\mathsf{tag}^1,\mathsf{tag}^2} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_2))$.
         · Else, $\mathsf{ct}^1_{\mathsf{tag}^1,\mathsf{tag}^2} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), 0^{|\mathsf{input}_2|})$.

· Add $\mathsf{ct}^1_{\mathsf{tag}^1,\mathsf{tag}^2}$ to the set ct^1.

* For every possible $\mathsf{tag}^3 = (i, b) \in [\lambda] \times \{0, 1\}$,
  · Let $\mathsf{input}_3 = L_{\{c_i\}}$.
  · If $C_{j^*,i} = b$, then $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_2, (\mathsf{tag}^1, \mathsf{tag}^3), (\mathsf{input}_3))$.
  · Else, $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_2, (\mathsf{tag}^1, \mathsf{tag}^3), 0^{|\mathsf{input}_3|})$.
  · Add $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3}$ to the set ct^2.
* Add the garbled circuit $\tilde{U}^{\mathsf{tag}^1}$ to the set ct^3.
* Add f_{tag^1} to the set ct^4.

– Else,
* If $\mathsf{tag}^1 \geq \mathsf{tag}^1_*$, Garble the universal circuit $\hat{U} = U[\mu_1(f_{\mathsf{tag}^1}), \ldots, \mu_\ell(f_{\mathsf{tag}^1})]$ for each tag^1, i.e. $(\tilde{U}^{\mathsf{tag}^1}, \mathcal{L}^{\mathsf{tag}^1}) \leftarrow \mathsf{GC.Garble}(1^\kappa, \hat{U})$ where $\tilde{U}^{\mathsf{tag}^1}$ denotes the garbled circuit and $\mathcal{L}^{\mathsf{tag}^1}$ indicates the set of labels for each wire in this circuit.
* Else, Garble the universal circuit $\hat{U} = U[0, \ldots, 0]$ for each tag^1, i.e. $(\tilde{U}^{\mathsf{tag}^1}, \mathcal{L}^{\mathsf{tag}^1}) \leftarrow \mathsf{GC.Garble}(1^\kappa, \hat{U})$ where $\tilde{U}^{\mathsf{tag}^1}$ denotes the garbled circuit and $\mathcal{L}^{\mathsf{tag}^1}$ indicates the set of labels for each wire in this circuit.
* For every possible $\mathsf{tag}^2 = (i, \delta) \in [\mathsf{p}'] \times [\Delta]$,
  · Let $\mathsf{input}_2 = \mathcal{L}^{\mathsf{tag}^1}_{\{Z_i\},\{\zeta_{\mathsf{tag}^2}(f_{\mathsf{tag}^1})\}}$.
  · $\mathsf{ct}^1_{\mathsf{tag}^1,\mathsf{tag}^2} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), 0^{|\mathsf{input}_2|})$.
  · Add $\mathsf{ct}^1_{\mathsf{tag}^1,\mathsf{tag}^2}$ to the set ct^1.
* For every possible $\mathsf{tag}^3 = (i, b) \in [\lambda] \times \{0, 1\}$
  · Let $\mathsf{input}_3 = \mathcal{L}^{\mathsf{tag}^1}_{\{c_i\},\{b\}}$.
  · $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, \emptyset, (\mathsf{tag}^1, \mathsf{tag}^3), 0^{|\mathsf{input}_3|})$.
  · Add $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3}$ to the set ct^2.
* Add the garbled circuit $\tilde{U}^{\mathsf{tag}^1}$ to the set ct^3.
* Add f_{tag^1} to the set ct^4.

4. **Guess:** $\mathcal{B}_3$ outputs a bit b which is output by the adversary $\mathcal{A}$.

We note that $\mathcal{B}_3$ is an admissable GC adversary as when we are in Case 2, the reduction shows that the advantage for an adversary is negligible. Similarly, in Case 3 as the labels aren't being used, we simply switch the circuits. The advantage of an adversary noticing this is also negligible from garbled circuit security. □

Following the sequence of polynomial in κ many sub-hybrid proofs, we have that any two adjacent games have negligible advantage. Thus the advantage for an adversary in distinguishing Games 2 and 3 is negligible. □

Lemma 4.7. $\forall \kappa \in \mathbb{N}$, $\mathsf{adv}^3_{\mathcal{A}} = \mathsf{adv}^4_{\mathcal{A}}$.

Proof. In game 3, $\mu_1, \ldots, \mu_\ell$ are picked as polynomials with constants $x_1, \ldots, x_\ell$ respectively while in game 4, we pick these with constants $0, \ldots, 0$ respectively.

These changes affect how the polynomials μ and ζ are sampled, but not η (as it's chosen independently in game 3 and ζ's are set later because of the change made in the first game). Observe in games 3 and 4, that we only care about evaluations of these polynomial at points in $\mathcal{I}$ and from the games we know that $|\mathcal{I}| \leq \mathsf{t}$. Thus these are identitically distributed because of lemma Lemma A.3. Hence an adversary cannot notice a difference. □

Game 4 is the game with simulators $\mathsf{S}_0, \mathsf{S}_1, \mathsf{S}_2$, hence we have shown the scheme is a statically-bounded-collusion *non-adaptive* simulation-secure encryption scheme. We now argue that we additionally satisfy strong security.

Claim 4.3. The above S-FE scheme is a statically-bounded-collusion *non-adaptive* strong simulation-secure encryption scheme according to Definition 3.3.

Proof. We go through the two properties.

- If the number of key queries made by adversary is allowed to exceed the static collusion bound q, then as long as the adversary does not submit any challenge message, the security game defined in Definition 3.1 holds. This can be seen easily as the simulators S_0 and S_1 are the same as S-FE.Setup, S-FE.KeyGen. Independent of the collusion bound, there is no change in the two worlds.
- The running time of the simulator algorithms $\mathsf{S}_0, \mathsf{S}_1$ are upper bounded by a fixed polynomial in λ, n and $\log q$ as S-FE.Setup, S-FE.KeyGen are weakly optimal.

□

□

5 Statically Bounded Collusion FE for P/poly

We construct Static-FE = (S-FE.Setup, S-FE.Enc, S-FE.KeyGen, S-FE.Dec), a weakly-optimal static-bounded-collusion FE scheme for the family of function classes $\{\mathcal{F}_n\}_{n\in\mathbb{N}}$ computable by P/poly circuits. Each function $\mathcal{F}_n$ takes as input $x \in \{0,1\}^\ell$ where the circuit computing the function has size upper bounded by λ. The ingredients for our construction are an identity-based-encryption scheme IBE and garbled circuits GC. We show that our construction is non-adaptive simulation secure Definition 3.4 and weakly optimal. Let κ be the security parameter for our construction.

S-FE.Setup$(1^\kappa, 1^\lambda, \mathsf{q}) \to (\mathsf{mpk}, \mathsf{msk})$

1. Set the parameters $\mathsf{p} = \kappa \cdot \mathsf{c}$[9] $+ 2$, $\mathsf{p}' = \kappa$, $\Gamma = 2\mathsf{p}\mathsf{q}^2$, $\Delta = 2\mathsf{q}$, $\mathsf{t} = \kappa$, $\mathsf{N} = \mathsf{p} \cdot \Gamma$, $\mathsf{N}' = \mathsf{p}' \cdot \Delta$. Let $\mathsf{pp} = (\mathsf{p}, \mathsf{p}', \Gamma, \Delta, \mathsf{t}, \mathsf{N}, \mathsf{N}', (1^\lambda, \mathsf{q}))$ be implicitly known to each algorithm.
   Let $\mathbb{F}$ be a finite field of size $> \mathsf{N}$. Let $\mathsf{PRG} : \{0,1\}^\kappa \to \{0,1\}^{4\mathsf{p}'\kappa+4}$ be a pseudorandom generator (this is implied by the existence of an IBE scheme).
2. IBE.Setup$(1^\kappa, ([\mathsf{p}] \times [\Gamma]) \times ([\mathsf{p}'] \times [\Delta])) \to (\mathsf{IBE.pk}_1, \mathsf{IBE.msk}_1)$[10].
3. IBE.Setup$(1^\kappa, ([\mathsf{p}] \times [\Gamma]) \times ([\lambda] \times \{0,1\})) \to (\mathsf{IBE.pk}_2, \mathsf{IBE.msk}_2)$.

[9] Here c is a constant and refers to the degree of the circuit $\mathcal{V}$ which is defined later in encryption.
[10] Our IBE scheme takes identity space $\mathcal{I}_n$ as input where we can imagine $\mathcal{I}_n$ is a binary vector of length $n = \log(\mathsf{N} \cdot \mathsf{N}')$

$\mathsf{CorrGarb}(C, x, \{\mathsf{sd}^j_{w,b}, k^j_{w,b}\}_{j\in[\mathsf{p}'],w\in\mathcal{W},b\in\{0,1\}}, \{r^j_w\}_{j\in[\mathsf{p}'],w\in\mathcal{W}})$

Inputs: Circuit C

Input Message $x \in \{0,1\}^\ell$

PRG Seeds $\{\mathsf{sd}^j_{w,b}\}_{j\in[\mathsf{p}'],w\in\mathcal{W},b\in\{0,1\}}$

keys $\{k^j_{w,b}\}_{j\in[\mathsf{p}'],w\in\mathcal{W},b\in\{0,1\}}$

Point and permute random bit $\{r^j_{w,b}\}_{j\in[\mathsf{p}'],w\in\mathcal{W},b\in\{0,1\}}$

Output: Garbled Cicuit $\mathsf{GC} = \tilde{C}$

Input Labels K

- For every wire $w \in \mathcal{W}$, bit $b \in \{0,1\}$, set $k_{w,b} = \oplus_{j=1}^{\mathsf{p}'} k^j_{w,b}$. Set $r_w = \oplus_{j=1}^{\mathsf{p}'} r^j_w$.
- Consider a universal circuit that takes in $\mathcal{U}(\cdot,\cdot)$ i.e. a circuit and an input and outputs the computation of the circuit. Let G be a gate in this circuit that takes in wires w_1, w_2 and outputs wires w_3, w_4. For every $b_1, b_2 \in \{0,1\}$.
  - Set $\mathsf{sd}_{w_3,G(b_1,b_2)\oplus r_{w_3}} = \left(\mathsf{sd}^1_{w_3,G(b_1,b_2)\oplus r_{w_3}} \circ \ldots \circ \mathsf{sd}^{\mathsf{p}'}_{w_3,G(b_1,b_2)\oplus r_{w_3}}\right)$.
  - Similarly, set $\mathsf{sd}_{w_4,G(b_1,b_2)\oplus r_{w_4}} = \left(\mathsf{sd}^1_{w_4,G(b_1,b_2)\oplus r_{w_4}} \circ \ldots \circ \mathsf{sd}^{\mathsf{p}'}_{w_4,G(b_1,b_2)\oplus r_{w_4}}\right)$.
  - Let $k_{w,0} = k^{(0)}_{w,0} \circ k^{(1)}_{w,0}$ and $k_{w,1} = k^{(0)}_{w,1} \circ k^{(1)}_{w,1}$ for every wire $w \in \mathcal{W}$.
  - The $(b_1,b_2)^{th}$ entry for gate G is set to be,

$$k^{(b_2)}_{w_1,b_1} \oplus k^{(b_1)}_{w_2,b_2} \oplus (\mathsf{sd}_{w_3,G(b_1\oplus r_{w_1},b_2\oplus r_{w_2})\oplus r_{w_3}} \circ G(b_1\oplus r_{w_1}, b_2\oplus r_{w_2}) \oplus r_{w_3} \circ \mathsf{sd}_{w_4,G(b_1\oplus r_{w_1},b_2\oplus r_{w_2})\oplus r_{w_4}} \circ G(b_1\oplus r_{w_1}, b_2\oplus r_{w_2}) \oplus r_{w_4})$$

    where $k_{w_1,b_1} = k^0_{w_1,b_1} \circ k^1_{w_1,b_1}$ and $k_{w_2,b_2} = (k^0_{w_2,b_2} \circ k^1_{w_2,b_2})$
- The translation table for the circuit $\mathcal{U}$ is the mapping $\mathcal{M}$ function that maps $\{0,1\}^{4v\kappa+4} \to \{0,1\}$ where $\mathcal{M}(k_{\mathsf{ow},r_{\mathsf{ow}}}) = 0$ and $\mathcal{M}(k_{\mathsf{ow},r_{\mathsf{ow}}\oplus 1}) = 1$ for each output wire ow.
- The complete garbled circuit with the description of each garbled gate is contained in GC. We generate the input wire keys corresponding to the inputs $\mathsf{inp} = (C \circ x)$ and call that K.

$$K = \left((\mathsf{sd}_{\mathsf{iw}_1,\mathsf{inp}_1\oplus r_{\mathsf{iw}_1}}; \mathsf{inp}_1 \oplus r_{\mathsf{iw}_1}) \circ \cdots \circ (\mathsf{sd}_{\mathsf{iw}_\mathsf{L},\mathsf{inp}_\mathsf{L}\oplus r_{\mathsf{iw}_\mathsf{L}}}; \mathsf{inp}_\mathsf{L} \oplus r_{\mathsf{iw}_\mathsf{L}})\right),$$

  where (i) $\mathsf{sd}_{\mathsf{iw}_i,b} = \left(\mathsf{sd}_{\mathsf{iw}^1_i,b\circ\cdots\circ\mathsf{sd}^{\mathsf{p}'}_{\mathsf{iw}_i,b}}\right)$, (ii) $\mathsf{L} = |C| + \ell$ and, (iii) $\mathsf{iw}_1, \ldots, \mathsf{iw}_\mathsf{L}$ are the input wire labels for the circuit $\mathcal{U}(\cdot,\cdot)$.
- Output (GC, K) where $\mathsf{s}'' = |\mathsf{GC}| + |K|$.

Figure 1: Routine Correlated Garbling

4. Output $(\mathsf{mpk} = (\mathsf{IBE.pk}_1, \mathsf{IBE.pk}_2), \mathsf{msk} = (\mathsf{IBE.msk}_1, \mathsf{IBE.msk}_2))$.

$\mathsf{S\text{-}FE.KeyGen}(\mathsf{msk}, C) \to \mathsf{sk}_C$.

$\mathsf{CorrEval}(\mathsf{GC}, K)$

Inputs: Garbled Circuit $\mathsf{GC} = \tilde{C}$
Input Labels K_{inp}
Outputs: Output Value y

On input the garbled circuit GC, wire labels K_{inp}, for every gate G with input wires w_1, w_2 and output wires w_3, w_4, do the following: suppose $(k_{w_1, b_1 \oplus r_{w_1}}, b_1 \oplus r_{w_1})$ and $(k_{w_2}, b_2 \oplus r_{w_2}, b_2 \oplus r_{w_2})$ be the wire labels respectively corresponding to the wires w_1 and w_2 obtained during the evaluation process. Using these keys and the $(b_1 \oplus r_{w_1}, b_2 \oplus r_{w_2})^{th}$ entry of the garbled table, we get the key $\mathsf{sd}_{w_3}, G(b_1, b_2) \oplus r_{w_3} \circ G(b_1, b_2) \oplus r_{w_3}$ for the w_3^{th} wire and the key $\mathsf{sd}_{w_4}, G(b_1, b_2) \oplus r_{w_4} \circ G(b_1, b_2) \oplus r_{w_4}$ for the w_4^{th} wire. Continuing this way, we finally obtain the labels associated with the output wire. Let the wire labels obtained corresponding to the output wires be $k_{\mathsf{ow}_1}, y_1 \oplus r_{\mathsf{ow}_1}, \ldots, k_{\mathsf{ow}_{L'}, y_{L'} \oplus r_{\mathsf{ow}_{L'}}}$. Using the translation table, output $y = (y_1 \circ \ldots \circ y_{L'})$.

Figure 2: Routine Correlated Evaluation

1. Let $C = C_1 \ldots C_\lambda$ be the bits of circuit C.
2. For each $i_1 \in [\mathsf{p}]$, sample $\gamma \xleftarrow{R} [\Gamma]$, let $\mathsf{tag}^1{}_{i_1} = (i_1, \gamma)$.
3. For each $i_2 \in [\mathsf{p}']$, sample $\delta \xleftarrow{R} [\Delta]$, let $\mathsf{tag}^2{}_{i_2} = (i_2, \delta)$.
4. For $i \in [\lambda]$, let $\mathsf{tag}^3{}_i = (i, C_i)$.
5. Let $\mathsf{taglist}^1 = (\mathsf{tag}^1{}_1, \ldots, \mathsf{tag}^1{}_{\mathsf{p}})$, $\mathsf{taglist}^2 = (\mathsf{tag}^2{}_1, \ldots, \mathsf{tag}^2{}_{\mathsf{p}'})$, and $\mathsf{taglist}^3 = (\mathsf{tag}^3{}_1, \ldots, \mathsf{tag}^3{}_\lambda)$.
6. Let $\mathsf{sk}^1 = \mathsf{sk}^2 = \emptyset$. We assume that these sets are ordered sets.
   For $i_1 \in [\mathsf{p}]$,
   - For $i_2 \in [\mathsf{p}']$,
     - $\mathsf{sk}^1_{i_1, i_2} \leftarrow \mathsf{IBE.KeyGen}(\mathsf{IBE.msk}_1, (\mathsf{tag}^1{}_{i_1}, \mathsf{tag}^2{}_{i_2}))$.
     - Add $\mathsf{sk}^1_{i_1, i_2}$ to the set sk^1.
   - For $i_3 \in [\lambda]$,
     - $\mathsf{sk}^2_{i_1, i_3} \leftarrow \mathsf{IBE.KeyGen}(\mathsf{IBE.msk}_2, (\mathsf{tag}^1{}_{i_1}, \mathsf{tag}^3{}_{i_3}))$.
     - Add $\mathsf{sk}^2_{i_1, i_3}$ to the set sk^2.
7. Output $\mathsf{sk}_C = (\mathsf{sk}^1, \mathsf{sk}^2, \mathsf{taglist}^1, \mathsf{taglist}^2, \mathsf{taglist}^3)$.

$\mathsf{S\text{-}FE.Enc}(\mathsf{mpk}, x = x_1 \ldots x_\ell, 1^{\mathsf{q}}) \rightarrow \mathsf{ct}$

1. Let $\mathcal{U}(\cdot, \cdot)$ be a universal circuit that takes a circuit C and an input x and evaluates $C(x)$ where length of C is λ and length of x is ℓ bits. Let $\mathcal{W}$ refer to all wires of said universal circuit.
2. Pick N unique nonzero field elements f_{tag^1} where $\mathsf{tag}^1 \in [\mathsf{p}] \times [\Gamma]$.
3. For every possible $\mathsf{tag}^3 = (i_3, b) \in [\lambda] \times \{0, 1\}$,
   - Sample a degree t polynomial $\delta_{\mathsf{tag}^3}(\cdot)$ over $\mathbb{F}$ whose constant term is b.
4. For every wire $w \in \mathcal{W}, b \in \{0, 1\}, \mathsf{tag}^2 \in ([\mathsf{p}'] \times [\Delta])$,
   - Sample $\mathsf{sd}^{\mathsf{tag}^2}_{w,b} \xleftarrow{R} \{0, 1\}^\kappa$.

- Set $k_{w,b}^{\mathsf{tag}^2} = \mathsf{PRG}(\mathsf{sd}_{w,b}^{\mathsf{tag}^2})$.
- Sample $r_w^{\mathsf{tag}^2} \xleftarrow{R} \{0,1\}$.
- For $h \in [\kappa]$,
  - Sample a degree t polynomial $\zeta_{w,b,h}^{\mathsf{tag}^2}(\cdot)$ over $\mathbb{F}$ whose constant term is h^{th} bit of $\mathsf{sd}_{w,b}^{\mathsf{tag}^2}$.
- For $h \in [4 \cdot \mathsf{p}' \cdot \kappa + 4]$,
  - Sample a degree t polynomial $\xi_{w,b,h}^{\mathsf{tag}^2}(\cdot)$ over $\mathbb{F}$ whose constant term is h^{th} bit of $k_{w,b}^{\mathsf{tag}^2}$.
- Sample a degree t polynomial $\eta_w^{\mathsf{tag}^2}(\cdot)$ over $\mathbb{F}$ whose constant term is $r_w^{(\mathsf{tag}^2)}$.

5. For every $\mathsf{tag}^2 \in ([\mathsf{p}'] \times [\Delta]), \mathsf{h} \in [\mathsf{s}'']$[11],
   - Sample a degree $\mathsf{p}-1$ polynomial $\nu_{\mathsf{h}}^{\mathsf{tag}^2}(\cdot)$ over $\mathbb{F}$ whose constant term is 0.
6. For every input $i \in \ell, h \in [\log |\mathbb{F}|]$,
   - Sample a degree-t polynomial $\mu_{i,h}(\cdot)$ over $\mathbb{F}$ whose constant term is $x_{i,h}$ where x_i denotes the i^{th} coordinate of $x \in \mathbb{F}$ and is represented as bits of length $\log |\mathbb{F}|$.
7. For $\mathsf{tag}^1 \in [\mathsf{p}] \times [\Gamma], \mathsf{h} \in [\mathsf{s}'']$, let $\mathcal{V}^{\mathsf{tag}^1,\mathsf{h}}[\{\delta_{\mathsf{tag}^3}(f_{\mathsf{tag}^1})\}_{\mathsf{tag}^3 \in [\lambda] \times \{0,1\}}]$ be the circuit that takes as inputs a circuit C of size λ and $E^\mu, E^\zeta, E^\xi, E^\eta, E^\nu$ and computes an arithmetization of the following circuit.
   - Let $C_1, \ldots C_\lambda$ be the binary representation of C.
   - Let $E_C^\delta = \{\delta_{(i,C_i)}(f_{\mathsf{tag}^1})\}_{i \in [\lambda]}$.
   - Compute $G = \mathsf{CorrGarb}(E_C^\delta, E^\mu, E^\zeta, E^\xi, E^\eta) + \sum_{e \in E^\nu} e$ (see Figure 1).
   - Output the h^{th} bit of the G.
8. Let $\mathsf{ct} = (\mathsf{ct}^1, \mathsf{ct}^2, \mathsf{ct}^3, \mathsf{E}^\mu, \mathsf{E}^\zeta, \mathsf{E}^\xi, \mathsf{E}^\eta, \mathsf{E}^\nu)$ be $(\emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset)$. We assume that these sets are ordered sets.
9. For $\mathsf{tag}^1 \in ([\mathsf{p}] \times [\Gamma]), \mathsf{h} \in [\mathsf{s}'']$,
   - Garble the circuit i.e. $(\tilde{\mathcal{V}}^{\mathsf{tag}^1,\mathsf{h}}, \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}) \leftarrow \mathsf{GC.Garble}(1^\kappa, \mathcal{V}^{\mathsf{tag}^1,\mathsf{h}})$.
   - Add the garbled circuit $\tilde{\mathcal{V}}^{\mathsf{tag}^1,\mathsf{h}}$ to the set ct^1.
   - For every possible $\mathsf{tag}^2 = (i_2, \delta) \in ([\mathsf{p}'] \times [\Delta])$,
     - Let $\hat{E}^\zeta_{\mathsf{tag}^1,\mathsf{tag}^2} = \{\zeta_{w,b,h}^{\mathsf{tag}^2}(f_{\mathsf{tag}^1})\}_{w \in \mathcal{W}, b \in \{0,1\}, h \in [\kappa]}$.
       Let $\mathsf{input}_\zeta = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\zeta_{i_2}, \hat{E}^\zeta_{\mathsf{tag}^1,\mathsf{tag}^2}}$.
       Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_\zeta))$ to E^ζ.
     - Let $\hat{E}^\xi_{\mathsf{tag}^1,\mathsf{tag}^2} = \{\xi_{w,b,h}^{\mathsf{tag}^2}(f_{\mathsf{tag}^1})\}_{w \in \mathcal{W}, b \in \{0,1\}, h \in [4\mathsf{p}'\kappa+4]}$.
       Let $\mathsf{input}_\xi = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\xi_{i_2}, \hat{E}^\xi_{\mathsf{tag}^1,\mathsf{tag}^2}}$.
       Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_\xi))$ to E^ξ.

[11] Recall that s'' is the output length of algorithm $\mathsf{CorrGarb}$ on circuit of size λ, input length ℓ.

- Let $\hat{E}^{\eta}_{\mathsf{tag}^1,\mathsf{tag}^2} = \{\eta_w^{\mathsf{tag}^2}(f_{\mathsf{tag}^1})\}_{w\in\mathcal{W}}$.
  Let $\mathsf{input}_\eta = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\eta_{i_2},\hat{E}^\eta_{\mathsf{tag}^1,\mathsf{tag}^2}}$.
  Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_1,(\mathsf{tag}^1,\mathsf{tag}^2),(\mathsf{input}_\eta))$ to E^η.
- Let $\hat{E}^{\nu}_{\mathsf{tag}^1,\mathsf{tag}^2,\mathsf{h}} = \{\nu_{\mathsf{h}}^{\mathsf{tag}^2}(f_{\mathsf{tag}^1})\}$.
  Let $\mathsf{input}_\nu = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\nu_{i_2},\hat{E}^\nu_{\mathsf{tag}^1,\mathsf{tag}^2,\mathsf{h}}}$.
  Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_1,(\mathsf{tag}^1,\mathsf{tag}^2),(\mathsf{input}_\nu))$ to E^ν.

- For every possible $\mathsf{tag}^3 = (i,b) \in ([\lambda]\times\{0,1\})$,
  - Let $\mathsf{input}_2 = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{\{C_i\},\{b\}}$.
  - $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3,\mathsf{h}} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_2,(\mathsf{tag}^1,\mathsf{tag}^3),(\mathsf{input}_2))$.
  - Add $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3,\mathsf{h}}$ to the set ct^2.
- Let $\hat{E}^{\mu}_{\mathsf{tag}^1} = \{\mu_{i,h}(f_{\mathsf{tag}^1})\}_{i\in\ell,h\in[\log|\mathbb{F}|]}$.
- Let $\mathsf{input}_\mu = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\mu,\hat{E}^\mu_{\mathsf{tag}^1}}$.
- Add input_μ to the set E^μ.
- Add f_{tag^1} to the set ct^3.

10. Output $\mathsf{ct} = (\mathsf{ct}^1,\mathsf{ct}^2,\mathsf{ct}^3,\mathsf{E}^\mu,\mathsf{E}^\zeta,\mathsf{E}^\xi,\mathsf{E}^\eta,\mathsf{E}^\nu)$.

$\mathsf{S\text{-}FE.Dec}(\mathsf{sk}_C,\mathsf{ct},1^{\mathsf{q}}) \to y$

1. Parse ct as $(\mathsf{ct}^1,\mathsf{ct}^2,\mathsf{ct}^3,\mathsf{E}^\mu,\mathsf{E}^\zeta,\mathsf{E}^\xi,\mathsf{E}^\eta,\mathsf{E}^\nu)$.
   - ct^1 as $\{\tilde{\mathcal{V}}^{\mathsf{tag}^1,\mathsf{h}}\}_{\mathsf{tag}^1\in[\mathsf{p}]\times[\Gamma],\mathsf{h}\in[\mathsf{s}'']}$.
   - ct^2 as $\{\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3,\mathsf{h}}\}_{\mathsf{tag}^1\in[\mathsf{p}]\times[\Gamma],\mathsf{tag}^3\in[\lambda]\times\{0,1\},\mathsf{h}\in[\mathsf{s}'']}$.
   - ct^3 as $\{f_{\mathsf{tag}^1}\}_{\mathsf{tag}^1\in[\mathsf{p}]\times[\Gamma]}$.
   - E^μ as $\{\mathsf{E}^\mu_{\mathsf{tag}^1}\}_{\mathsf{tag}^1\in[\mathsf{p}]\times[\Gamma]}$.
   - E^ζ as $\{\mathsf{E}^\zeta_{\mathsf{tag}^1,\mathsf{tag}^2,\mathsf{h}}\}_{\mathsf{tag}^1\in[\mathsf{p}]\times[\Gamma],\mathsf{tag}^2\in[\mathsf{p}']\times[\Delta],\mathsf{h}\in[\mathsf{s}'']}$.
   - E^ξ as $\{\mathsf{E}^\xi_{\mathsf{tag}^1,\mathsf{tag}^2,\mathsf{h}}\}_{\mathsf{tag}^1\in[\mathsf{p}]\times[\Gamma],\mathsf{tag}^2\in[\mathsf{p}']\times[\Delta],\mathsf{h}\in[\mathsf{s}'']}$.
   - E^η as $\{\mathsf{E}^\eta_{\mathsf{tag}^1,\mathsf{tag}^2,\mathsf{h}}\}_{\mathsf{tag}^1\in[\mathsf{p}]\times[\Gamma],\mathsf{tag}^2\in[\mathsf{p}']\times[\Delta],\mathsf{h}\in[\mathsf{s}'']}$.
   - E^ν as $\{\mathsf{E}^\nu_{\mathsf{tag}^1,\mathsf{tag}^2,\mathsf{h}}\}_{\mathsf{tag}^1\in[\mathsf{p}]\times[\Gamma],\mathsf{tag}^2\in[\mathsf{p}']\times[\Delta],\mathsf{h}\in[\mathsf{s}'']}$.
2. Parse sk_C as $(\mathsf{sk}^1,\mathsf{sk}^2,\mathsf{taglist}^1,\mathsf{taglist}^2,\mathsf{taglist}^3)$.
   - sk^1 as $\{\mathsf{sk}^1_{i_1,i_2}\}_{i_1\in[\mathsf{p}],i_2\in[\mathsf{p}']}$.
   - sk^2 as $\{\mathsf{sk}^2_{i_1,i_2}\}_{i_1\in[\mathsf{p}],i_2\in[\lambda]}$.
3. For each $i\in[\mathsf{p}]$ and $\mathsf{h}\in[\mathsf{s}'']$,
   - For $i_3\in[\lambda]$
     - $\mathsf{input}^2_{i_3} = \mathsf{IBE.Dec}(\mathsf{sk}^2_{i_1,i_3},\mathsf{ct}^2_{\mathsf{tag}^1{}_{i_1},\mathsf{tag}^3{}_{i_3},\mathsf{h}})$
   - For $i_2\in[\mathsf{p}']$,
     - $\mathsf{input}^\zeta_{i_2} = \mathsf{IBE.Dec}(\mathsf{sk}^1_{i_1,i_2},\mathsf{E}^\zeta_{\mathsf{tag}^1{}_{i_1},\mathsf{tag}^2{}_{i_2},\mathsf{h}})$.
     - $\mathsf{input}^\xi_{i_2} = \mathsf{IBE.Dec}(\mathsf{sk}^1_{i_1,i_2},\mathsf{E}^\xi_{\mathsf{tag}^1{}_{i_1},\mathsf{tag}^2{}_{i_2},\mathsf{h}})$.

- $\mathsf{input}^{\eta}_{i_2} = \mathsf{IBE.Dec}(\mathsf{sk}^1_{i_1,i_2}, \mathsf{E}^{\eta}_{\mathsf{tag}^1{}_{i_1},\mathsf{tag}^2{}_{i_2},\mathsf{h}})$.
- $\mathsf{input}^{\nu}_{i_2} = \mathsf{IBE.Dec}(\mathsf{sk}^1_{i_1,i_2}, \mathsf{E}^{\nu}_{\mathsf{tag}^1{}_{i_1},\mathsf{tag}^2{}_{i_2},\mathsf{h}})$.

- Compute $y_{\mathsf{tag}^1{}_i,\mathsf{h}} \leftarrow \mathsf{GC.Eval}(\tilde{\mathcal{V}}_i^{\mathsf{tag}^1,\mathsf{h}}, (\mathsf{input}^2, \mathsf{E}^{\mu}, \mathsf{input}^{\zeta}, \mathsf{input}^x i, \mathsf{input}^{\eta}, \mathsf{input}^n u))$.

4. For each $\mathsf{h} \in [\mathsf{s}'']$, interpolate the sets of degree $\mathsf{p}-1$ polynomials which lie on $(f_{\mathsf{tag}^1{}_i}, y_{\mathsf{tag}^1{}_i,\mathsf{h}})$ to obtain π_{h}. Parse $\pi_1(0) \circ \ldots \circ \pi_{\mathsf{s}''}(0)$ as (GC, K).
5. Output $\mathsf{CorrEval}(\mathsf{GC}, K)$ (see Figure 2).

Efficiency

Claim 5.1. If $\mathsf{IBE}, \mathsf{GC}$ consists of PPT algorithms, then our $\mathsf{Static\text{-}FE} = (\mathsf{S\text{-}FE.Setup}, \mathsf{S\text{-}FE.Enc}, \mathsf{S\text{-}FE.KeyGen}, \mathsf{S\text{-}FE.Dec})$ scheme consists of PPT algorithms and is weakly optimal according to Definition 3.2.

Proof. We analyze the different algorithms.

- S-FE.Setup runs IBE.Setup twice for different identity spaces. We have that the first setup runs in time $\mathsf{poly}(\kappa, \log(\mathsf{N} \cdot \mathsf{N}')) = \mathsf{poly}(\kappa, \log(\mathsf{q}, \mathsf{D}, \kappa))$. The second setup runs in time $\mathsf{poly}(\kappa, \log(\mathsf{N} \cdot \lambda \cdot 2)) = \mathsf{poly}(\kappa, \log(\mathsf{q}, \lambda, \kappa))$. Thus S-FE.Setup is efficient according to Definition 3.2.

- S-FE.KeyGen runs IBE.KeyGen with $\mathsf{IBE.msk}_1$ for $\mathsf{p} \cdot \mathsf{p}'$ times and IBE.KeyGen with $\mathsf{IBE.msk}_2$ for $\mathsf{p} \cdot \lambda \cdot 2$ times. One invocation of IBE.KeyGen with $\mathsf{IBE.msk}_1$ takes $\mathsf{poly}(\kappa, \log(\mathsf{q}, \mathsf{D}))$ time and one invocation of IBE.KeyGen with $\mathsf{IBE.msk}_2$ takes $\mathsf{poly}(\kappa, \log(\mathsf{q}, \lambda))$ time. Thus, we can bound the runtime of the entire algorithm similar with $\mathsf{poly}(\kappa, \lambda, \mathsf{D}, \log(\mathsf{q}))$.

- S-FE.Enc runs GC.Garble on a circuit of size $\mathcal{V}$. Since from above we can see $|\mathcal{V}| \in \mathsf{poly}(\kappa, \lambda, \mathsf{q})$, we can see by efficiency of garbled circuits, this in total takes time in $\mathsf{poly}(\kappa, \lambda, \mathsf{q})$.

  Next, note that the labels of each input of $\mathcal{V}$ is encrypted at most $\max(\mathsf{N} \cdot \mathsf{s}'' \cdot \mathsf{N}', \mathsf{N} \cdot \mathsf{s}'' \cdot \lambda \cdot 2) \in \mathsf{poly}(\kappa, \lambda, \mathsf{q})$ times. Since GC.Garble is efficient, each labels is of length $\mathsf{poly}(\kappa, \lambda)$. Since IBE is efficient, the encryptions of all these labels can be generated in $\mathsf{poly}(\kappa, \lambda, \mathsf{q})$ time.

- S-FE.Dec runs IBE.Dec on a subset of ciphertexts produced by S-FE.Enc. Since S-FE.Enc is efficient, these are polynomially bounded in number and size. Then, it runs GC.Eval $\mathsf{p} \cdot \mathsf{s}''$ many times, which, by efficiency of GC, is also efficient.

□

Correctness The claim below will help us formalize correctness.

Claim 5.2. Circuit $\mathcal{V}$ has size $\mathsf{poly}(|C|, \kappa, \mathsf{q})$ and depth c.

Proof. $\mathcal{V}$ considers a universal circuit which takes in inputs C, x, which are both length bounded by $|C|$. We can bound the size of $|\mathcal{U}|$ by $\mathsf{poly}(|C|)$. We can observe that for each wire, the length of the following inputs are

- E^{δ} takes in a field element and a tag (which can be written in $\log(\mathsf{poly}(\mathsf{q} \cdot \Gamma)) \in O(\log(\mathsf{q}, \kappa, \mathsf{D}))$ bits) for each of of $2|C|$ tags, so we can bound the size of E^{δ} with $\mathsf{poly}(|C|, \log(\kappa, \mathsf{D}, \mathsf{q}))$.

- E^{μ} takes in two field element for each of $\mathsf{p} = \kappa \cdot \mathsf{D}$ tags, which bounds the size with $\mathsf{poly}(\kappa, \mathsf{D}, \log(\mathsf{q}))$
- E^{ζ} takes in $b \cdot h = 2\kappa$ field element for each of $\mathsf{p}' = \kappa$ tags, which bounds the size with $\mathsf{poly}(\kappa, \log(\mathsf{q}, \mathsf{D}))$.
- E^{ξ} takes in $b \cdot h \in O(\mathsf{p}' \cdot \kappa) \in \mathsf{poly}(\kappa)$ field element for each of $\mathsf{p}' = \kappa$ tags, which bounds the size with $\mathsf{poly}(\kappa, \log(\mathsf{q}, \mathsf{D}))$.
- E^{η} a field element and a tag for each for $\mathsf{p}' = \kappa$ tags, which bounds the size with $\mathsf{poly}(\kappa, \log(\mathsf{q}, \mathsf{D}))$.

Since all of these inputs are applied to at most $|\mathcal{U}|$ wires, we can bound the total size of all inputs of $\mathsf{CorrGarb}$ with $\mathsf{poly}(|C|, \kappa, \mathsf{D}, \log(\mathsf{q}))$. Since each inputs is involved in a constant amount of concatenation and xor operations, this bounds the total size of E_{C_i} with $\mathsf{poly}(|C|, \kappa, \mathsf{D}, \log(\mathsf{q}))$. This also bounds the size of E^{ν} with $\mathsf{poly}(|C|, \kappa, \mathsf{D}, \log(\mathsf{q}))$. Since E^{C_i} simply adds these values together on top of this, we can in turn bound its size similarly with $\mathsf{poly}(|C|, \kappa, \mathsf{D}, \log(\mathsf{q}))$.

□

Claim 5.3. If IBE is a correct identity based encryption scheme, GC is a correct garbling scheme, then our $\mathsf{S\text{-}FE}$ scheme is correct weakly-optimal static-bounded-collusion FE scheme.

Proof. We make the following observations for each $\kappa, \mathsf{q}, x \in \{0,1\}^{\ell}$, for any polynomials (in κ) λ (the maximum size of circuit), D (degree of the circuit) and any circuit C with size $\leq \lambda$ and degree $\leq \mathsf{D}$.

$\mathsf{S\text{-}FE.Dec}(\mathsf{sk}_C, \mathsf{ct}, 1^{\mathsf{q}})$, computes $\forall i_1 \in [\mathsf{p}], \mathsf{h} \in [\mathsf{s}''], \forall i_2 \in [\mathsf{p}']$, a bunch of decryptions on identities $(\mathsf{tag}^1{}_{i_1}, \mathsf{tag}^2{}_{i_2})$ under msk_1. As the ciphertexts are generated using $\mathsf{IBE.Enc}$ as encryption of $\mathcal{L}^{\mathsf{tag}^1{}_{i_1}, \mathsf{h}}_{E^{\zeta}_{i_2}, \hat{E}^{\zeta}_{\mathsf{tag}^1{}_{i_1}, \mathsf{tag}^2{}_{i_2}}}$ and others similarly. If IBE is a correct identity based encryption scheme, then the decryption proceeds correctly i.e. the labels for garbled circuit $\tilde{\mathcal{V}}^{\mathsf{tag}^1{}_{i_1}, \mathsf{h}}$ where the input label for each position is set accordingly.

Similarly, decryption computes $\forall i_1 \in [\mathsf{p}], \forall i_3 \in [\lambda]$, $\mathsf{IBE.Dec}(\mathsf{sk}^2_{i_1, i_3}, \mathsf{ct}^2_{\mathsf{tag}^1{}_{i_1}, \mathsf{tag}^3{}_{i_3}})$. During keygen, $\mathsf{tag}^3{}_{i_3}$ is set as (i_3, C_i) where C is the circuit being computed and $\mathsf{sk}^2_{i_1, i_3}$ is computed as $\mathsf{IBE.KeyGen}(\mathsf{msk}_2, (\mathsf{tag}^1{}_{i_1}, \mathsf{tag}^3{}_{i_3}))$ and $\mathsf{ct}^2_{\mathsf{tag}^1{}_{i_1}, \mathsf{tag}^3{}_{i_3}}$ during $\mathsf{IBE.Enc}$ as encryption of $\mathcal{L}^{\mathsf{tag}^1{}_{i_1}, \mathsf{h}}_{\{c_i\}, \{C_i\}}$ on $\mathsf{id} = (\mathsf{tag}^1{}_{i_1}, \mathsf{tag}^3{}_{i_3})$. If IBE is a correct identity based encryption scheme, then the decryption proceeds correctly i.e. the labels for garbled circuit $\tilde{\mathcal{V}}^{\mathsf{tag}^1{}_{i_1}, \mathsf{h}}$ where the input label for each c_i is set according to the input C_i.

Thus decryption through the procedure $\mathsf{GC.Eval}$ thus computes $\forall i_1 \in [\mathsf{p}]$,

$$\mathsf{CorrGarb}(\{\delta_{i,C_i}(f_{\mathsf{tag}^1{}_{i_1}})\}_{i \in \lambda}, \{\mu_{i,h}(f_{\mathsf{tag}^1{}_{i_1}})\}_{i \in \ell, h \in [\log \mathbb{F}]}, \{\zeta^{\mathsf{tag}^2}_{w,b,h}(f_{\mathsf{tag}^1{}_{i_1}})\}_{w \in \mathcal{W}, b \in \{0,1\}, h \in [\kappa], \mathsf{tag}^2 \in \mathsf{taglist}^2},$$

$$\{\zeta^{\mathsf{tag}^2}_{w,b,h}(f_{\mathsf{tag}^1{}_{i_1}})\}_{w \in \mathcal{W}, b \in \{0,1\}, h \in [\kappa], \mathsf{tag}^2 \in \mathsf{taglist}^2}, \{\eta^{\mathsf{tag}^2}_{w}(f_{\mathsf{tag}^1{}_{i_1}})\}_{w \in \mathcal{W}, \mathsf{tag}^2 \in \mathsf{taglist}^2}) + \sum_{\mathsf{tag}^2 \in \mathsf{taglist}^2} \nu^{\mathsf{tag}^2}_{\mathsf{h}}(f_{\mathsf{tag}^1{}_{i_1}})$$

As we know for each $i_1 \in [\mathsf{p}]$, $f_{\mathsf{tag}^1{}_{i_1}}$ from parsing of ct^3.

For each $\mathsf{h} \in [\mathsf{s}'']$, we can interpolate the sets of degree $\mathsf{p}-1$ polynomials which lie on $(f_{\mathsf{tag}^1{}_i}, y_{\mathsf{tag}^1{}_i,\mathsf{h}})$ to obtain π_{h}. $\pi_1(0) \circ \ldots \circ \pi_{\mathsf{s}''}(0)$ is thus set as (GC, K) which is then evaluated to reveal output $C(x)$ using $\mathsf{CorrEval}$.

□

5.1 Security for AV scheme

The simulator proceeds in roughly the same way as the Section 4 simulator, with the primary difference being that rather being able to directly simulate our circuit output, we need to simulate the output of the $\mathsf{CorrGarb}$ garbling procedure which produces the corresponding circuit output. To do this, the simulator, after receiving the pre-challenge phase functional queries and their outputs, simply needs to compose the simulator $\mathsf{Sim}_{\mathsf{CorrGarb}}$ (analogous to the one in [AV19]) below to produce a target output GC, K of the regular garbled circuit, which can be simulated using the Section 4 simulator. The caveat of the $\mathsf{Sim}_{\mathsf{CorrGarb}}$ simulator needs to be run on an independent random $\mathsf{sd}^j_{w,0}$, which we can argue via the cover-freeness of the tag^2 space in a similar way to arguing the independence of the $\sum Z_i$ random polynomials in Section 4.

More details are provided later in Appendix B.

References

[ABB10] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (h)ibe in the standard model. In *Proceedings of the 29th Annual international conference on Theory and Applications of Cryptographic Techniques*, EUROCRYPT'10, pages 553–572, Berlin, Heidelberg, 2010. Springer-Verlag.

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

[AIK06] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Computationally private randomizing polynomials and their applications. *Computational Complexity*, 15(2):115–162, 2006.

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

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

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

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

[AV19] Prabhanjan Ananth and Vinod Vaikuntanathan. Optimal bounded-collusion secure functional encryption. In Dennis Hofheinz and Alon Rosen, editors, *Theory of Cryptography - 17th International Conference, TCC 2019, Nuremberg, Germany, December 1-5, 2019, Proceedings, Part I*, volume 11891 of *Lecture Notes in Computer Science*, pages 174–198. Springer, 2019.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. In *Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '01, 2001.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, 1988.

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

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: definitions and challenges. In *Proceedings of the 8th conference on Theory of cryptography*, TCC'11, pages 253–273, Berlin, Heidelberg, 2011. Springer-Verlag.

[BV15] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. In *Proceedings of the 2015 IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 171–190, 2015.

[BW07] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In *Proceedings of the 4th conference on Theory of cryptography*, TCC'07, pages 535–554, Berlin, Heidelberg, 2007. Springer-Verlag.

[CHH+07] Ronald Cramer, Goichiro Hanaoka, Dennis Hofheinz, Hideki Imai, Eike Kiltz, Rafael Pass, Abhi Shelat, and Vinod Vaikuntanathan. Bounded cca2-secure encryption. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 502–518. Springer, 2007.

[CHKP10] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30 - June 3, 2010. Proceedings*, pages 523–552, 2010.

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

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

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

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

[GGH+13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS*, 2013.

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

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

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

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

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

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

[KMUW18] Lucas Kowalczyk, Tal Malkin, Jonathan Ullman, and Daniel Wichs. Hardness of non-interactive differential privacy from one-way functions, 2018.

[KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *Proceedings of the theory and applications of cryptographic techniques 27th annual international conference on Advances in cryptology*, EUROCRYPT'08, 2008.

[Sha85] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Proceedings of CRYPTO 84 on Advances in cryptology*, pages 47–53, New York, NY, USA, 1985. Springer-Verlag New York, Inc.

[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*, CCS '10, pages 463–472, New York, NY, USA, 2010. ACM.

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

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *STOC*, pages 475–484, 2014.

[Yao86] Andrew Yao. How to generate and exchange secrets. In *FOCS*, pages 162–167, 1986.

A Helpful Combinatoric Lemmas

We first discuss a few lemma's that will help us prove the security of the scheme.

Lemma A.1. Let $A_1, \dots A_{\mathsf{p}}$ be disjoint sets of size $\Gamma = \mathsf{pq}^2$. For $i \in [\mathsf{p}], j \in [\mathsf{q}]$, let each $a_{i,j}$ be sampled uniformly and independently at random from A_i. Let $\mathcal{A}$ be the set

$$\left\{ x \in \bigcup_{i=0}^{\mathsf{p}} A_i \mid \exists i, j_0, j_1 : j_0 \neq j_1 \wedge x = a_{i,j_0} = a_{i,j_1} \right\}$$

Then

$$\Pr[|\mathcal{A}| \geq \mathsf{q}] \leq \mathsf{negl}(\kappa)$$

Proof. Denote the elements of A_i as $x_{i,k}$ for $k \in [\Gamma]$. Let $X_{i,k}$ be the indicator variable which is one if $x_{i,k}$ is in $\mathcal{A}$ and 0 otherwise, and let $X = \sum X_{i,k}$, which is the cardinality of $\mathcal{A}$.

Claim A.1. $E[X] \leq 1$

Proof. Consider the set

$$\mathcal{A}' = \left\{ (i, j_0, j_1) \in [\mathsf{p}] \times [\mathsf{q}]^2 \mid j_0 < j_1 \wedge a_{i,j_0} = a_{i,j_1} \right\}$$

Clearly we can define a surjective mapping from $\mathcal{A}'$ to $\mathcal{A}$ by taking the mapping each $(i, j_0, j_1$ to a_{i,j_0} in $\mathcal{A}$, letting us upper bound $|\mathcal{A}| \leq |\mathcal{A}'|$. Now we can write $E[|\mathcal{A}'|]$ as the sum of of indicator variables X_{i,j_0,j_1}, which is 1 if $a_{i,j_0} = a_{i,j_1}$. We can see that since each $a_{i,j}$ are chosen independently and uniformly at random in a set of size S

$$\Pr[X_{i,j_0,j_1} = 1] \leq \frac{1}{\Gamma}$$

, which means by linearity of expectations we can write

$$E[|\mathcal{A}'|] = \frac{\mathsf{p} \cdot \binom{\mathsf{q}}{2}}{\Gamma} \leq \frac{\mathsf{p} \cdot \mathsf{q}^2}{2\mathsf{pq}^2} \leq 1$$

which transitively upper bounds $E[X]$. □

Claim A.2. The set of random variables $\{X_{i,k}\}_{i \in [\mathsf{p}], k \in [\Gamma]}$ is negatively associated.

Proof. $i \in [\mathsf{p}], j \in [\mathsf{q}], k \in [\Gamma]$, let $Y_{i,j,k}$ to be the random variable which is 1 if $a_{i,j} = k$ and 0 otherwise. For any fixed i^*, j^*, we can see that the collection $\{Y_{i^*,j^*,k}\}_{k \in [\Gamma]}$ is negatively associated by the Zero-One Lemma, since a_{i^*,j^*} can only take on one value. Since each $a_{i,j}$ are chosen independently, we can use the fact that the union of independent negatively associated sets of variables are negatively associated to see that $\{Y_{i,j,k}\}_{i \in [\mathsf{p}], j \in [\mathsf{q}], k \in [\Gamma]}$ is negatively associated. Finally, we observe that for any i^*, k^*, we can write X_{i^*,k^*} as the zero one function which is one if and only if $\sum_{j \in \mathsf{q}} Y_{i^*,j,k^*} \geq 2$. Since $\{X_{i,k}\}_{i \in [\mathsf{p}], k \in [\Gamma]}$ can be written as a monotonic function on disjoint index sets of $\{Y_{i,j,k}\}_{i \in [\mathsf{p}], j \in [\mathsf{q}], k \in [\Gamma]}$, it too is negatively assocaited. □

Thus, we can apply a Chernoff bound to

$$\Pr\left[\sum_{i\in[\mathsf{p}]\ k\in[\Gamma]} X_{i,k} \geq \kappa \cdot E\left[\sum_{i\in[\mathsf{p}]\ k\in[\Gamma]} X_{i,k}\right]\right] \leq \mathsf{exp}\left(-(\kappa-1)^2 E\left[\sum_{i\in[\mathsf{p}]\ k\in[\Gamma]} X_{i,k}\right]/(\kappa+1)\right)$$
$$\Leftrightarrow \Pr[|\mathcal{A}| \geq \kappa] \leq \mathsf{exp}(-\Omega(\kappa))$$

□

Lemma A.2. Let $\mathsf{p}' = \kappa$ and $q \in \mathsf{poly}(\kappa)$. Let $B_1, \ldots B_{\mathsf{p}'}$ be disjoint sets of size $\Delta \geq 2q$. For $i \in [\mathsf{p}'], j \in [\mathsf{q}]$, let each $b_{i,j}$ be sampled uniformly and independently at random from B_i.

$$\Pr\left[\exists j^* \in [\mathsf{q}]\ \forall i \in [\mathsf{p}']\ \exists j' \in [\mathsf{q}]\backslash\{j\}\ : b_{i,j^*} = b_{i,j'}\right] \leq \mathsf{negl}(\kappa)$$

Proof. Consider for each j, at in every B_i, observe the set $\{b_{i,j'} \in B_i \mid j' \neq j\}$ has size at most $q-1$. Since $b_{i,j}$ is chosen independently, the probability a j' exists which for which $b_{i,j'}$ equal to $b_{i,j}$ is $\frac{1}{2}$. By independence, the probability this is true for all i is at most $\left(\frac{1}{2}\right)^{\mathsf{p}'}$. We can union bound over all $j \in [\mathsf{q}]$ with $\mathsf{q} \cdot 2^{-\mathsf{p}'} \in negl(\kappa)$ □

Lemma A.3. Let $\mathbb{F}$ be a finite field. $\forall t \in [0, |\mathbb{F}|) c_0 \neq c_1 \in \mathbb{F}$, let $p_b(\cdot)$ be a random degree t polynomial where $p_b(0) = c_b$. Finally, let $S \subset \mathbb{F}\backslash\{0\} : |S| \leq t$. Define $\mathcal{D}_b = \{(x, p_b(x)) \mid x \in S\}$. The distributions $\mathcal{D}_0$ and $\mathcal{D}_1$ are identical.

Proof. We first prove the following claim.

Claim A.3. Consider any subset of distinct field elements $x_1, \ldots x_{t+1} \in \mathbb{F}$, then for every set of values $y_1, \ldots y_{t+1} \in \mathbb{F}$, there exists a unique degree t polynomial $p(\cdot)$ such that $\forall i \in [t+1]\ p(x_i) = y_i$.

Proof. We can construct said polynomial $p(\cdot)$ via lagrange interpolation as

$$\sum_{i\in[t+1]} y_i \cdot \prod_{j\in[t+1]\backslash\{i\}} \frac{x - x_j}{x_i - x_j}$$

We can verify that $p(x_i) = y_i$ as for any $i' \neq i$, the product term contains $\frac{x-x_i}{x_{i'}-x_i}$, which evaluates to 0. In addition, we can see the value of the product term at x_i is $y_i \cdot \prod_{j\neq i} \frac{x_i - x_j}{x_i - x_j} = y_i \cdot 1 = y_i$. Since each product is of t degree one terms, this is a degree t polynomial. Since the cardinality of degree t polynomials is $|\mathbb{F}|^{t+1}$, equal to the cardinality of possible $\{y_i\} + i \in [t+1]$ sets, this surjective relation must be bijective, so the polynomial is unique. □

Suppose $|S| = t$, fix a particular element $D = \{(x_i, y_i)\}_{i\in[t]}$ in the range of $\mathcal{D}_b$. We can compute the probability of D occuring as the probability the random polynomial p_b satisfies $\forall x \in S : p_b(x_i) = y_i$ and $p_b(0) = c_b$. By the claim above, there is a unique polynomial satisfying these relations. Thus, the probability of $D \xleftarrow{R} \mathcal{D}_b$ is 1 in the number of degree t polynomials over $\mathbb{F}$ with constant term c_b, which of course is $|\mathbb{F}|^t$, and independent of b, so these distributions are identical.

If $|S| < t$, we can consider an arbitrary subset $S' \subset \mathbb{F} : |S'| = t$ and $S \subset S'$. We can compute the observation of any given $D = \{(x_i, y_i)\}_{i\in[|S|]} \xleftarrow{R} \mathcal{D}_b$ from S as the union of $D' = D \cup \{(x_i^*, y_i^*)\}_{x_i^* \in S'\backslash S} \xleftarrow{R} \mathcal{D}'_b$ for all possible sets $\{y_i^*\}$, which has cardinality $\mathbb{F}^{t-|S|}$ so we can conclude the probability of D is $\frac{|\mathbb{F}|^{t-|S|}}{|\mathbb{F}|^t}$, which of course is independent of b.

□

B Security for AV scheme

We describe the simulators below. For *non-adaptive* simulation security, we focus on simulators $\mathsf{S}_0, \mathsf{S}_1, \mathsf{S}_2$.

$\mathsf{S}_0(1^\kappa, 1^\ell, 1^\lambda, 1^{\mathsf{D}}, \mathsf{q}) \to (\mathsf{mpk}, \mathsf{st}_0)$

- Run $\mathsf{S\text{-}FE.Setup}(1^\kappa, 1^\ell, 1^\lambda, 1^{\mathsf{D}}, \mathsf{q}) \to (\mathsf{mpk}, \mathsf{msk})$. Let st_0 be msk and output $(\mathsf{mpk}, \mathsf{st}_0)$.

$\mathsf{S}_1(\mathsf{st}_0, C) \to \mathsf{sk}_C$.

- Run $\mathsf{S\text{-}FE.KeyGen}(\mathsf{st}_0, C) \to \mathsf{sk}_C$.

$\mathsf{S}_2(\mathsf{st}_1, \Pi^x) \to (\mathsf{ct}, \mathsf{st}_2)$.

- **Parsing:**
  - Let $|\Pi^x| = \mathsf{q}$ (wlog the cardinality is the maximum number of queries allowed).
  - Parse $\Pi^x = ((C_1, C_1(x)), \ldots, (C_{\mathsf{q}}, C_{\mathsf{q}}(x)))$.
  - $\forall j \in [\mathsf{q}]$, let sk_{C_j} be the reply to query C_j to S_1.
  - Parse sk_{C_j} as $(\mathsf{sk}^{1,(j)}, \mathsf{sk}^{2,(j)}, \mathsf{taglist}^{1,(j)}, \mathsf{taglist}^{2,(j)}, \mathsf{taglist}^{3,(j)})$.
  - Parse $\mathsf{taglist}^{1,(j)} = (\mathsf{tag}_1^{1,(j)}, \ldots, \mathsf{tag}_{\mathsf{p}}^{1,(j)})$.
  - Parse $\mathsf{taglist}^{2,(j)} = (\mathsf{tag}_1^{2,(j)}, \ldots, \mathsf{tag}_{\mathsf{p}'}^{2,(j)})$.
  - Parse $\mathsf{taglist}^{3,(j)} = (\mathsf{tag}_1^{3,(j)}, \ldots, \mathsf{tag}_{\lambda}^{3,(j)})$.
- **Computing recieved identities:**
  - Let $\mathcal{I} \subseteq ([\mathsf{p}] \times [\Gamma])$ be a set of identities which are queried on two or more keys,

$$\bigcup_{j \in \mathsf{q}, j' \in \mathsf{q}, i_1 \in [\mathsf{p}] \ \wedge \ j \neq j'} \left((\mathsf{tag}_{i_1}^{1,(j)}) \cap (\mathsf{tag}_{i_1}^{1,(j')}) \right).$$

  - If $|\mathcal{I}| > \mathsf{t}$, output $\perp$.
- **Computing cover free elements:**
  For $j \in \mathsf{q}$, let $\mathcal{K} = \mathcal{K}^* = \emptyset$. Let $\mathcal{K}^*$ be an ordered set.
  - Let $\mathcal{K}_1 = \cup_{i_2 \in [\mathsf{p}']} \mathsf{tag}_{i_2}^{2,(j)}$.
  - Pick any element $\mathsf{tag}_{i_2^*}^{2,(j)} \in \mathcal{K}_1 \setminus \mathcal{K} \in ([\mathsf{p}'] \times [\Delta])$ and add it $(\mathsf{tag}_{i_2^*}^{2,(j)})$ to $\mathcal{K}^*$ such that $\mathcal{K}_j^* = \mathsf{tag}_{i_2^*}^{2,(j)}$. If no such element exists, output $\perp$
  - Add $\mathcal{K}_1$ to set $\mathcal{K}$. i.e. $\mathcal{K} = \mathcal{K} \cup \mathcal{K}_1$.
- Let $\mathcal{U}(\cdot, \cdot)$ be a universal circuit that takes a circuit C and an input x and evaluates $C(x)$ where length of C is λ and length of x is ℓ bits. Let $\mathcal{W}$ refer to all wires of said universal circuit.
- Pick N unique nonzero field elements f_{tag^1} where $\mathsf{tag}^1 \in [\mathsf{p}] \times [\Gamma]$.
- For every possible $\mathsf{tag}^3 = (i_3, b) \in [\lambda] \times \{0, 1\}$,
  - Sample a degree t polynomial $\delta_{\mathsf{tag}^3}(\cdot)$ over $\mathbb{F}$ whose constant term is b.

- For every wire $w \in \mathcal{W}, b \in \{0,1\}, \mathsf{tag}^2 \in ([\mathsf{p}'] \times [\Delta])$,
  - Sample $\mathsf{sd}_{w,b}^{\mathsf{tag}^2} \xleftarrow{R} \{0,1\}^\kappa$.
  - Set $k_{w,b}^{\mathsf{tag}^2} = \mathsf{PRG}(\mathsf{sd}_{w,b}^{\mathsf{tag}^2})$.
  - Sample $r_w^{\mathsf{tag}^2} \xleftarrow{R} \{0,1\}$.
  - For $h \in [\kappa]$,
    - Sample a degree t polynomial $\zeta_{w,b,h}^{\mathsf{tag}^2}(\cdot)$ over $\mathbb{F}$ whose constant term is h^{th} bit of $\mathsf{sd}_{w,b}^{\mathsf{tag}^2}$.
  - For $h \in [4 \cdot \mathsf{p}' \cdot \kappa + 4]$,
    - Sample a degree t polynomial $\xi_{w,b,h}^{\mathsf{tag}^2}(\cdot)$ over $\mathbb{F}$ whose constant term is h^{th} bit of $k_{w,b}^{\mathsf{tag}^2}$.
  - Sample a degree t polynomial $\eta_w^{\mathsf{tag}^2}(\cdot)$ over $\mathbb{F}$ whose constant term is $r_w^{(\mathsf{tag}^2)}$.
- For $\mathsf{tag}^1 \in [\mathsf{p}] \times [\Gamma], \mathsf{h} \in [\mathsf{s}'']$, let $\mathcal{V}^{\mathsf{tag}^1,\mathsf{h}}[\{\delta_{\mathsf{tag}^3}(f_{\mathsf{tag}^1})\}_{\mathsf{tag}^3 \in [\lambda] \times \{0,1\}}]$ be the circuit that takes as inputs a circuit C of size λ and $E^\mu, E^\zeta, E^\xi, E^\eta, E^\nu$ and computes an arithmetization of the following circuit.
  - Let $C_1, \ldots C_\lambda$ be the binary representation of C.
  - Let $E_C^\delta = \{\delta_{(i,C_i)}(f_{\mathsf{tag}^1})\}_{i \in [\lambda]}$.
  - Compute $G = \mathsf{CorrGarb}(E_C^\delta, E^\mu, E^\zeta, E^\xi, E^\eta) + \sum_{e \in E^\nu} e$.
  - Output the h^{th} bit of the G.
- For $\mathsf{tag}^1 \in [\mathsf{p}] \times [\Gamma], j \in \mathsf{q}, \mathsf{h} \in [\mathsf{s}'']$
  - Sample a random degree $\mathsf{p}-1$ polynomials $\tau_{j,\mathsf{h}}(\cdot)$ with constant term as the h^{th} bit of $\mathsf{Sim}_{\mathsf{CorrGarb}}(1^\kappa, C_j, C_j(x))$.
- For every $\mathsf{tag}^2 \in ([\mathsf{p}'] \times [\Delta]) \backslash \mathcal{K}^*, \mathsf{h} \in [\mathsf{s}'']$,
  - Sample a degree $\mathsf{p}-1$ polynomial $\nu_{\mathsf{h}}^{\mathsf{tag}^2}(\cdot)$ over $\mathbb{F}$ whose constant term is 0.
- For every $\mathsf{tag}^{2\prime} \in \mathcal{K}^*, \mathsf{h} \in [\mathsf{s}'']$,
  - Set $\nu_{\mathsf{h}}^{\mathsf{tag}^2}(\cdot)$ to be the polynomial interpolation which satisfies $\forall \mathsf{tag}^1 \in \mathcal{I}$,

$$\tau_{j,\mathsf{h}}(f_{\mathsf{tag}^1}) = \mathsf{Sim}_{\mathsf{CorrGarb}}(1^\kappa, C_j, C_j(x)).$$

- For every input $i \in \ell, h \in [\log |\mathbb{F}|]$,
  - Sample a degree-t polynomial $\mu_{i,h}(\cdot)$ over $\mathbb{F}$ whose constant term is 0.
- Let $\mathsf{ct} = (\mathsf{ct}^1, \mathsf{ct}^2, \mathsf{ct}^3, \mathsf{E}^\mu, \mathsf{E}^\zeta, \mathsf{E}^\xi, \mathsf{E}^\eta, \mathsf{E}^\nu)$ be $(\emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset)$. We assume that these sets are ordered sets.
- For $\mathsf{tag}^1 \in ([\mathsf{p}] \times [\Gamma]), \mathsf{h} \in [\mathsf{s}'']$,
  - If there exists a unique j^* index for which $\mathsf{tag}^1 \in \mathsf{taglist}^{1,(j^*)}$.
    - Simulate the garbled circuit $(\tilde{\mathcal{V}}^{\mathsf{tag}^1,\mathsf{h}}, \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}) \leftarrow \mathsf{GC.Sim}(1^\kappa, 1^{|\mathsf{input}|}, 1^{|\mathcal{V}^{\mathsf{tag}^1,\mathsf{h}}|}, \tau_{j^*}(f_{\mathsf{tag}^1}))$
    - Add the simulated garbled circuit $\tilde{\mathcal{V}}^{\mathsf{tag}^1,\mathsf{h}}$ to the set ct^1.

* For every possible $\mathsf{tag}^2 = (i_2, \delta) \in ([\mathsf{p}'] \times [\Delta])$
  Set $\mathsf{input}_\zeta = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\zeta_{i_2}}$
  Set $\mathsf{input}_\xi = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\xi_{i_2}}$
  Set $\mathsf{input}_\eta = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\eta_{i_2}}$
  Set $\mathsf{input}_\nu = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\nu_{i_2}}$
* For every possible $\mathsf{tag}^3 = (i, b) \in ([\lambda] \times \{0,1\})$,
  Set $\mathsf{input}_2 = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{\{C_i\}}$.
* Set $\mathsf{input}_\mu = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\mu}$

– Otherwise
* Garble the circuit i.e. $(\tilde{\mathcal{V}}^{\mathsf{tag}^1,\mathsf{h}}, \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}) \leftarrow \mathsf{GC.Garble}(1^\kappa, \mathcal{V}^{\mathsf{tag}^1,\mathsf{h}})$.
* Add the garbled circuit $\tilde{\mathcal{V}}^{\mathsf{tag}^1,\mathsf{h}}$ to the set ct^1.
* For every possible $\mathsf{tag}^2 = (i_2, \delta) \in ([\mathsf{p}'] \times [\Delta])$
  · If $\exists j : \mathsf{tag}^1 \in \mathsf{taglist}^{1,(j)} \wedge \mathsf{tag}^2 \in \mathsf{taglist}^{2,(j)}$
    Let $\hat{E}^\zeta_{\mathsf{tag}^1,\mathsf{tag}^2} = \{\zeta^{\mathsf{tag}^2}_{w,b,h}(f_{\mathsf{tag}^1})\}_{w\in\mathcal{W},b\in\{0,1\},h\in[\kappa]}$.
    Set $\mathsf{input}_\zeta = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\zeta_{i_2},\hat{E}^\zeta_{\mathsf{tag}^1,\mathsf{tag}^2}}$
    Let $\hat{E}^\xi_{\mathsf{tag}^1,\mathsf{tag}^2} = \{\xi^{\mathsf{tag}^2}_{w,b,h}(f_{\mathsf{tag}^1})\}_{w\in\mathcal{W},b\in\{0,1\},h\in[4\mathsf{p}'\kappa+4]}$.
    Set $\mathsf{input}_\xi = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\xi_{i_2},\hat{E}^\xi_{\mathsf{tag}^1,\mathsf{tag}^2}}$.
    Let $\hat{E}^\eta_{\mathsf{tag}^1,\mathsf{tag}^2} = \{\eta^{\mathsf{tag}^2}_w(f_{\mathsf{tag}^1})\}_{w\in\mathcal{W}}$.
    Set $\mathsf{input}_\eta = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\eta_{i_2},\hat{E}^\eta_{\mathsf{tag}^1,\mathsf{tag}^2}}$.
    Let $\hat{E}^\nu_{\mathsf{tag}^1,\mathsf{tag}^2,\mathsf{h}} = \{\nu^{\mathsf{tag}^2}_{\mathsf{h}}(f_{\mathsf{tag}^1})\}$.
    Set $\mathsf{input}_\nu = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\nu_{i_2},\hat{E}^\nu_{\mathsf{tag}^1,\mathsf{tag}^2,\mathsf{h}}}$.
  · Otherwise, if $\not\exists j : \mathsf{tag}^1 \in \mathsf{taglist}^{1,(j)} \wedge \mathsf{tag}^2 \in \mathsf{taglist}^{2,(j)}$, set input_ζ to an appropriately length of padded zeroes.
    Set $\mathsf{input}_\zeta = 0^{|\mathsf{input}_\zeta|}$
    Set $\mathsf{input}_\xi = 0^{|\mathsf{input}_\xi|}$
    Set $\mathsf{input}_\eta = 0^{|\mathsf{input}_\eta|}$
    Set $\mathsf{input}_\nu = 0^{|\mathsf{input}_\nu|}$
* For every possible $\mathsf{tag}^3 = (i, b) \in ([\lambda] \times \{0,1\})$,
  · If $\exists j : \mathsf{tag}^1 \in \mathsf{taglist}^{1,(j)} \wedge \mathsf{tag}^3 \in \mathsf{taglist}^{3,(j)}$
    Set $\mathsf{input}_2 = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{\{C_i\},\{b\}}$.
  · Otherwise, add encryptions of appropriately padded 0's
    Set $\mathsf{input}_2 = 0^{|\mathsf{input}_2|}$
  · Let $\hat{E}^\mu_{\mathsf{tag}^1} = \{\mu_{i,h}(f_{\mathsf{tag}^1})\}_{i\in\ell,h\in[\log|\mathbb{F}|]}$.
  · Let $\mathsf{input}_\mu = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\mu,\hat{E}^\mu_{\mathsf{tag}^1}}$.

– Add components to ciphertext

* For $\mathsf{tag}^1 \in ([\mathsf{p}] \times [\Gamma]), \mathsf{h} \in [\mathsf{s}'']$
  - Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_\zeta))$ to E^ζ.
  - Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_\xi))$ to E^ξ.
  - Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_\eta))$ to E^η.
  - Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_\nu))$ to E^ν.
* For every possible $\mathsf{tag}^3 = (i, b) \in ([\lambda] \times \{0,1\})$
  - Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_2, (\mathsf{tag}^1, \mathsf{tag}^3), (\mathsf{input}_2))$ to ct^2
* Add input_μ to the set E^μ.
* Add f_{tag^1} to the set ct^3

- Output $\mathsf{ct} = (\mathsf{ct}^1, \mathsf{ct}^2, \mathsf{ct}^3, \mathsf{E}^\mu, \mathsf{E}^\zeta, \mathsf{E}^\xi, \mathsf{E}^\eta, \mathsf{E}^\nu)$ with an empty state st_2.

Before moving to the security of the encryption scheme, we analyze the security of the $\mathsf{CorrGarb}$ algorithm in two experiments $\mathsf{CorrExpt}_0^{\mathcal{A},\mathsf{Ch}}(1^\lambda)$ and $\mathsf{CorrExpt}_1^{\mathcal{A},\mathsf{Sim}_{\mathsf{CorrGarb}}}(1^\lambda)$ which are very similar from their [AV19] counterparts.

$\mathsf{CorrExpt}_0^{\mathcal{A},\mathsf{Ch}}(1^\lambda)$:

- Challenge Ch Computes the following:
  - Let $\{\mathsf{taglist}^{2,(i)}\}_{i\in[\mathsf{q}]}$ be a cover free set of subsets of $([\mathsf{p}'] \times [\Delta])$ where $\mathsf{taglist}^{2,(i)} = (\mathsf{tag}_1^{2,(i)}, \ldots, \mathsf{tag}_{\mathsf{p}'}^{2,(i)})$.
  - For every wire $w \in \mathcal{W}$, index $\mathsf{tag}^2 \in ([\mathsf{p}'] \times [\Delta])$, bit $b \in \{0,1\}$, sample $\mathsf{sd}_{w,b}^{\mathsf{tag}^2} \xleftarrow{R} \{0,1\}^\lambda$ and sample $r_w^{\mathsf{tag}^2} \xleftarrow{R} \{0,1\}$
  - For every wire $w \in \mathcal{W}$, index $\mathsf{tag}^2 \in ([\mathsf{p}'] \times [\Delta])$, bit $b \in \{0,1\}$, let $k_{w,b}^{\mathsf{tag}^2} = \mathsf{PRG}(\mathsf{sd}_{w,b}^{\mathsf{tag}^2})$
- $\mathcal{A}$ submits its challenge input x. It then submits at most q circuit queries $C_1, \ldots, C_\mathsf{q}$.
- For the i^{th} circuit query C_i, Ch generates the following:

$$(\mathsf{GC}_i, K_i) \leftarrow \mathsf{CorrGarb}\left(C_i, x, \{\mathsf{sd}_{w,b}^{\mathsf{taglist}_j^{2,(i)}}, k_{w,b}^{\mathsf{taglist}_j^{2,(i)}}\}_{j\in[\mathsf{p}'],w\in\mathcal{W},b\in\{0,1\}}, \{r_w^j\}_{j\in[\mathsf{p}'],w\in\mathcal{W}}\right)$$

$\mathsf{CorrExpt}_1^{\mathcal{A},\mathsf{Sim}_{\mathsf{CorrGarb}}}(1^\lambda)$:

- $\mathcal{A}$ submits its challenge input x. It then submits at most Q circuit queries $C_1, \ldots, C_Q$ adaptively.
- On inputs $(1^\lambda, C_i, C_i(x))$, $\mathsf{Sim}_{\mathsf{CorrGarb}}$ generates $\left(\{\mathsf{GC}_i, K_i\}_{i\in[Q]}\right)$. The result is sent to $\mathcal{A}$.
- $\mathcal{A}$ outputs bit b. Output b.

Lemma B.1. For every PPT adversary $\mathcal{A}$, there exists a PPT simulator $\mathsf{Sim}_{\mathsf{CorrGarb}}$ such that the following holds:

$$\left|\Pr\left[0 \leftarrow \mathsf{CorrExpt}_0^{\mathcal{A},\mathsf{Ch}}(1^\lambda)\right] - \Pr\left[0 \leftarrow \mathsf{CorrExpt}_1^{\mathcal{A},\mathsf{Sim}_{\mathsf{CorrGarb}}}(1^\lambda)\right]\right| \leq \mathsf{negl}(\lambda)$$

Proof. The proof follows exactly from Lemma 2 in [AV19] where we've adjusted the simulator and notation according to the notation of our paper. □

$\mathsf{Sim}_{\mathsf{CorrGarb}}(1^\kappa, C, C(x))$

Inputs: Security Parameter1^κ
Circuit C
Circuit Output $C(x)$

Output: Garbled Circuit $\mathsf{GC} = \tilde{C}$
Input Labels K

- For every $i \in [\mathsf{q}]$, sample a p'-sized set $\mathsf{taglist}^{2,(i)} \subseteq ([\mathsf{p}'] \times [\Delta])$ uniformly at random.
- Check if this is cover free, i.e. for every $i^* \in [\mathsf{q}]$, there exists a unique index that is not present in other sets. $\mathsf{taglist}^{2,(i^*)} \setminus (\cup_{i \in [\mathsf{q}], i \neq i^*} \mathsf{taglist}^{2,(i)}) \neq \emptyset$. If this is not true, then output $\perp$.
- For every wire $w \in \mathcal{W}$, bit $b \in \{0,1\}$, sample the PRG seeds as follows: for every $\mathsf{tag}^2 \in ([\mathsf{p}'] \times [\Delta])$, sample $\mathsf{sd}_{w,b}^{\mathsf{tag}^2} \xleftarrow{R} \{0,1\}^\kappa$.
- $\mathcal{A}$ submits the input x. Then it adaptively submits the queries $C_1, \ldots, C_{\mathsf{q}}$.
- For every $i \in [\mathsf{q}]$, the i^{th} simulated garbling GC_i of C_i is generated as follows:
  - For every wire $w \in \mathcal{W}$ and bit b, set $\mathsf{sd}_{w,b} = \mathsf{sd}_{w,b}^{\mathsf{taglist}_1^{2,(i)}} \circ \ldots \circ \mathsf{sd}_{w,b}^{\mathsf{taglist}_{\mathsf{p}'}^{2,(i)}}$
  - For every wire $w \in \mathcal{W}$, sample $r_w \xleftarrow{R} \{0,1\}$ uniformly at random
  - Next, generate the input wire labels of GC_i as
$$K_i = \left((\mathsf{sd}_{\mathsf{iw}_1, r_{\mathsf{iw}_1}}; r_{\mathsf{iw}_1}) \circ \ldots \circ (\mathsf{sd}_{\mathsf{iw}_L, r_{\mathsf{iw}_L}}; r_{\mathsf{iw}_L}) \right)$$
  - For every gate G in universal circuit $\mathcal{U}(\cdot,\cdot)$, program the $(r_{w_1}, r_{w_2})^{th}$ entry of the garbled table as
$$k_{w_1, r_{w_1}}^{r_{w_2}} \oplus k_{w_2, r_{w_2}}^{r_{w_1}} \oplus \left(\mathsf{sd}_{w_3, r_{w_3}} \circ r_{w_3} \circ \mathsf{sd}_{w_4, r_{w_4}} \circ r_{w_4} \right)$$
    where each $k_{w,r_w}^0 \circ k_{w,r_w}^1$ is computed to be the xor of the PRG output on the corresponding seeds. All other entries in the garbled circuit are uniformly random.
    Construct GC_i to consist of garbled gates for every $\mathcal{U}(\cdot,\cdot)$.
  - Complete the translation table TT_i as follows, TT_i has the wire label $k_{\mathsf{ow}, r_{\mathsf{ow}}}$ mapped to $C_i(x)$ and a random string mapped to complement of $C_i(x)$, where ow is the output wire of C_i.
- Output $\{\mathsf{GC}_i, K_i\}_{i \in [\mathsf{q}]}$

Figure 3: Routine Correlated Garbling Simulator

Theorem B.1. Let IBE be a secure encryption scheme according to Definition 2.3 and GC be a secure garbled circuits scheme according to Definition 2.2. Then the above S-FE scheme is a statically-bounded-collusion *non-adaptive* strong simulation-secure encryption scheme according to Definition 3.4.

Game 0. This is the real non-adaptive security game between a challenger and an attacker. The game is parameterized by a security parameter κ.

1. Adversary $\mathcal{A}$ outputs circuit parameters $1^\lambda, 1^{\mathsf{D}}$ and collusion bound q.
2. Challenger $\mathcal{C}$ runs $\mathsf{S\text{-}FE.Setup}(1^\kappa, 1^\lambda, 1^{\mathsf{D}}, \mathsf{q}) \to (\mathsf{mpk}, \mathsf{msk})$ and sends mpk to $\mathcal{A}$.
   Setup phase:
   - Set the parameters $\mathsf{p} = \kappa \cdot \mathsf{c} + 2$, $\mathsf{p}' = \kappa$, $\Gamma = 2\mathsf{p}\mathsf{q}^2$, $\Delta = 2\mathsf{q}$, $\mathsf{t} = \kappa$, $\mathsf{N} = \mathsf{p} \cdot \Gamma$, $\mathsf{N}' = \mathsf{p}' \cdot \Delta$. Let $\mathbb{F}$ be a finite field of size $> \mathsf{N}$. Let $\mathsf{PRG} : \{0,1\}^\kappa \to \{0,1\}^{4\mathsf{p}'\kappa+4}$ be a pseudorandom generator.
   - $\mathsf{IBE.Setup}(1^\kappa, ([\mathsf{p}] \times [\Gamma]) \times ([\mathsf{p}'] \times [\Delta])) \to (\mathsf{IBE.pk}_1, \mathsf{IBE.msk}_1)$.
   - $\mathsf{IBE.Setup}(1^\kappa, ([\mathsf{p}] \times [\Gamma]) \times ([\lambda] \times \{0,1\})) \to (\mathsf{IBE.pk}_2, \mathsf{IBE.msk}_2)$.
   - Output $(\mathsf{mpk} = (\mathsf{IBE.pk}_1, \mathsf{IBE.pk}_2), \mathsf{msk} = (\mathsf{IBE.msk}_1, \mathsf{IBE.msk}_2))$.
3. **Keygen query phase:** $\mathcal{A}$ queries the keygen oracle $\mathsf{S\text{-}FE.KeyGen}(\mathsf{msk}, \cdot)$ and makes q queries, where the oracle computes as follows, $\mathsf{S\text{-}FE.KeyGen}(\mathsf{msk}, C) \to \mathsf{sk}_C$.
   - Let $C = C_1 \dots C_\lambda$ be the bits of circuit C.
   - For each $i_1 \in [\mathsf{p}]$, sample $\gamma \xleftarrow{R} [\Gamma]$, let $\mathsf{tag}^1{}_{i_1} = (i_1, \gamma)$.
   - For each $i_2 \in [\mathsf{p}']$, sample $\delta \xleftarrow{R} [\Delta]$, let $\mathsf{tag}^2{}_{i_2} = (i_2, \delta)$.
   - For $i \in [\lambda]$, let $\mathsf{tag}^3{}_i = (i, C_i)$.
   - Let $\mathsf{taglist}^1 = (\mathsf{tag}^1{}_1, \dots, \mathsf{tag}^1{}_{\mathsf{p}})$, $\mathsf{taglist}^2 = (\mathsf{tag}^2{}_1, \dots, \mathsf{tag}^2{}_{\mathsf{p}'})$, and $\mathsf{taglist}^3 = (\mathsf{tag}^3{}_1, \dots, \mathsf{tag}^3{}_\lambda)$.
   - Let $\mathsf{sk}^1 = \mathsf{sk}^2 = \emptyset$. We assume that these sets are ordered sets.
     For $i_1 \in [\mathsf{p}]$,
     - For $i_2 \in [\mathsf{p}']$,
       * $\mathsf{sk}^1_{i_1,i_2} \leftarrow \mathsf{IBE.KeyGen}(\mathsf{IBE.msk}_1, (\mathsf{tag}^1{}_{i_1}, \mathsf{tag}^2{}_{i_2}))$.
       * Add $\mathsf{sk}^1_{i_1,i_2}$ to the set sk^1.
     - For $i_3 \in [\lambda]$,
       * $\mathsf{sk}^2_{i_1,i_3} \leftarrow \mathsf{IBE.KeyGen}(\mathsf{IBE.msk}_2, (\mathsf{tag}^1{}_{i_1}, \mathsf{tag}^3{}_{i_3}))$.
       * Add $\mathsf{sk}^2_{i_1,i_3}$ to the set sk^2.
   - Output $\mathsf{sk}_C = (\mathsf{sk}^1, \mathsf{sk}^2, \mathsf{taglist}^1, \mathsf{taglist}^2, \mathsf{taglist}^3)$.
4. **Challenge message:** $\mathcal{A}$ outputs a challenge message $x \in \{0,1\}^\ell$.
5. **Encryption phase:** $\mathcal{C}$ computes $\mathsf{S\text{-}FE.Enc}(\mathsf{mpk}, x = x_1 x_2 \dots x_\ell, 1^{\mathsf{q}}) \to \mathsf{ct}$. It sends this to $\mathcal{A}$.
   - Let $\mathcal{U}(\cdot,\cdot)$ be a universal circuit that takes a circuit C and an input x and evaluates $C(x)$ where length of C is λ and length of x is ℓ bits. Let $\mathcal{W}$ refer to all wires of said universal circuit.
   - Pick N unique nonzero field elements f_{tag^1} where $\mathsf{tag}^1 \in [\mathsf{p}] \times [\Gamma]$.
   - For every possible $\mathsf{tag}^3 = (i_3, b) \in [\lambda] \times \{0,1\}$,
     - Sample a degree t polynomial $\delta_{\mathsf{tag}^3}(\cdot)$ over $\mathbb{F}$ whose constant term is b.

- For every wire $w \in \mathcal{W}, b \in \{0,1\}, \mathsf{tag}^2 \in ([\mathsf{p}'] \times [\Delta])$,
  - Sample $\mathsf{sd}_{w,b}^{\mathsf{tag}^2} \xleftarrow{R} \{0,1\}^\kappa$.
  - Set $k_{w,b}^{\mathsf{tag}^2} = \mathsf{PRG}(\mathsf{sd}_{w,b}^{\mathsf{tag}^2})$.
  - Sample $r_w^{\mathsf{tag}^2} \xleftarrow{R} \{0,1\}$.
  - For $h \in [\kappa]$,
    - Sample a degree t polynomial $\zeta_{w,b,h}^{\mathsf{tag}^2}(\cdot)$ over $\mathbb{F}$ whose constant term is h^{th} bit of $\mathsf{sd}_{w,b}^{\mathsf{tag}^2}$.
  - For $h \in [4 \cdot \mathsf{p}' \cdot \kappa + 4]$,
    - Sample a degree t polynomial $\xi_{w,b,h}^{\mathsf{tag}^2}(\cdot)$ over $\mathbb{F}$ whose constant term is h^{th} bit of $k_{w,b}^{\mathsf{tag}^2}$.
  - Sample a degree t polynomial $\eta_w^{\mathsf{tag}^2}(\cdot)$ over $\mathbb{F}$ whose constant term is $r_w^{(\mathsf{tag}^2)}$.
- For every $\mathsf{tag}^2 \in ([\mathsf{p}'] \times [\Delta]), \mathsf{h} \in [\mathsf{s}'']$,
  - Sample a degree $\mathsf{p}-1$ polynomial $\nu_{\mathsf{h}}^{\mathsf{tag}^2}(\cdot)$ over $\mathbb{F}$ whose constant term is 0.
- For every input $i \in \ell, h \in [\log |\mathbb{F}|]$,
  - Sample a degree-t polynomial $\mu_{i,h}(\cdot)$ over $\mathbb{F}$ whose constant term is $x_{i,h}$ where x_i denotes the i^{th} coordinate of $x \in \mathbb{F}$ and is represented as bits of length $\log |F|$.
- For $\mathsf{tag}^1 \in [\mathsf{p}] \times [\Gamma], \mathsf{h} \in [\mathsf{s}'']$, let $\mathcal{V}^{\mathsf{tag}^1,\mathsf{h}}[\{\delta_{\mathsf{tag}^3}(f_{\mathsf{tag}^1})\}_{\mathsf{tag}^3 \in [\lambda] \times \{0,1\}}]$ be the circuit that takes as inputs a circuit C of size λ and $E^\mu, E^\zeta, E^\xi, E^\eta, E^\nu$ and computes an arithmetization of the following circuit.
  - Let $C_1, \ldots C_\lambda$ be the binary representation of C.
  - Let $E_C^\delta = \{\delta_{(i,C_i)}(f_{\mathsf{tag}^1})\}_{i \in [\lambda]}$.
  - Compute $G = \mathsf{CorrGarb}(E_C^\delta, E^\mu, E^\zeta, E^\xi, E^\eta) + \sum_{e \in E^\nu} e$ (see Figure 1).
  - Output the h^{th} bit of the G.
- Let $\mathsf{ct} = (\mathsf{ct}^1, \mathsf{ct}^2, \mathsf{ct}^3, \mathsf{E}^\mu, \mathsf{E}^\zeta, \mathsf{E}^\xi, \mathsf{E}^\eta, \mathsf{E}^\nu)$ be $(\emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset)$. We assume that these sets are ordered sets.
- For $\mathsf{tag}^1 \in ([\mathsf{p}] \times [\Gamma]), \mathsf{h} \in [\mathsf{s}'']$,
  - Garble the circuit i.e. $(\tilde{\mathcal{V}}^{\mathsf{tag}^1,\mathsf{h}}, \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}) \leftarrow \mathsf{GC.Garble}(1^\kappa, \mathcal{V}^{\mathsf{tag}^1,\mathsf{h}})$.
  - Add the garbled circuit $\tilde{\mathcal{V}}^{\mathsf{tag}^1,\mathsf{h}}$ to the set ct^1.
  - For every possible $\mathsf{tag}^2 = (i_2, \delta) \in ([\mathsf{p}'] \times [\Delta])$,
    - Let $\hat{E}^\zeta_{\mathsf{tag}^1,\mathsf{tag}^2} = \{\zeta_{w,b,h}^{\mathsf{tag}^2}(f_{\mathsf{tag}^1})\}_{w \in \mathcal{W}, b \in \{0,1\}, h \in [\kappa]}$.
      Let $\mathsf{input}_\zeta = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\xi_{i_2}, \hat{E}^\zeta_{\mathsf{tag}^1,\mathsf{tag}^2}}$.
      Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_\zeta))$ to E^ζ.
    - Let $\hat{E}^\xi_{\mathsf{tag}^1,\mathsf{tag}^2} = \{\xi_{w,b,h}^{\mathsf{tag}^2}(f_{\mathsf{tag}^1})\}_{w \in \mathcal{W}, b \in \{0,1\}, h \in [4\mathsf{p}'\kappa+4]}$.
      Let $\mathsf{input}_\xi = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\xi_{i_2}, \hat{E}^\xi_{\mathsf{tag}^1,\mathsf{tag}^2}}$.
      Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_\xi))$ to E^ξ.

* Let $\hat{E}^{\eta}_{\mathsf{tag}^1,\mathsf{tag}^2} = \{\eta_w^{\mathsf{tag}^2}(f_{\mathsf{tag}^1})\}_{w\in\mathcal{W}}$.
  Let $\mathsf{input}_\eta = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^{\eta}_{i_2},\hat{E}^{\eta}_{\mathsf{tag}^1,\mathsf{tag}^2}}$.
  Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_1,(\mathsf{tag}^1,\mathsf{tag}^2),(\mathsf{input}_\eta))$ to E^η.
* Let $\hat{E}^{\nu}_{\mathsf{tag}^1,\mathsf{tag}^2,\mathsf{h}} = \{\nu_{\mathsf{h}}^{\mathsf{tag}^2}(f_{\mathsf{tag}^1})\}$.
  Let $\mathsf{input}_\nu = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^{\nu}_{i_2},\hat{E}^{\nu}_{\mathsf{tag}^1,\mathsf{tag}^2,\mathsf{h}}}$.
  Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_1,(\mathsf{tag}^1,\mathsf{tag}^2),(\mathsf{input}_\nu))$ to E^ν.

– For every possible $\mathsf{tag}^3 = (i,b) \in ([\lambda]\times\{0,1\})$,
* Let $\mathsf{input}_2 = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{\{C_i\},\{b\}}$.
* $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3,\mathsf{h}} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_2,(\mathsf{tag}^1,\mathsf{tag}^3),(\mathsf{input}_2))$.
* Add $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3,\mathsf{h}}$ to the set ct^2.

– Let $\hat{E}^{\mu}_{\mathsf{tag}^1} = \{\mu_{i,h}(f_{\mathsf{tag}^1})\}_{i\in\ell,h\in[\log|\mathbb{F}|]}$.
– Let $\mathsf{input}_\mu = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^{\mu},\hat{E}^{\mu}_{\mathsf{tag}^1}}$.
– Add input_μ to the set E^μ.
– Add f_{tag^1} to the set ct^3.

- Output $\mathsf{ct} = (\mathsf{ct}^1,\mathsf{ct}^2,\mathsf{ct}^3,\mathsf{E}^\mu,\mathsf{E}^\zeta,\mathsf{E}^\xi,\mathsf{E}^\eta,\mathsf{E}^\nu)$.

6. **Guess:** $\mathcal{A}$ outputs a bit b.

Game 1. In this game, we introduce some purely syntactic changes which make it easier to describe some of the following games

1. Adversary $\mathcal{A}$ outputs circuit parameters $1^\lambda, 1^{\mathsf{D}}$ and collusion bound q.
2. **Setup Phase:...,Keygen query phase:...,Challenge message:....**
3. **Encryption phase:** $\mathcal{C}$ computes $\mathsf{S\text{-}FE.Enc}(\mathsf{mpk}, x = x_1x_2\dots x_\ell, 1^{\mathsf{q}}) \to \mathsf{ct}$. It sends this to $\mathcal{A}$.

- **Parsing:**
  – Let $|\Pi^x| = \mathsf{q}$ (wlog the cardinality is the maximum number of queries allowed).
  – Parse $\Pi^x = ((C_1, C_1(x)),\dots,(C_{\mathsf{q}}, C_{\mathsf{q}}(x)))$.
  – $\forall j \in [\mathsf{q}]$, let sk_{C_j} be the reply to query C_j to S_1.
  – Parse sk_{C_j} as $(\mathsf{sk}^{1,(j)},\mathsf{sk}^{2,(j)},\mathsf{taglist}^{1,(j)},\mathsf{taglist}^{2,(j)},\mathsf{taglist}^{3,(j)})$.
  – Parse $\mathsf{taglist}^{1,(j)} = (\mathsf{tag}_1^{1,(j)},\dots,\mathsf{tag}_{\mathsf{p}}^{1,(j)})$.
  – Parse $\mathsf{taglist}^{2,(j)} = (\mathsf{tag}_1^{2,(j)},\dots,\mathsf{tag}_{\mathsf{p}'}^{2,(j)})$.
  – Parse $\mathsf{taglist}^{3,(j)} = (\mathsf{tag}_1^{3,(j)},\dots,\mathsf{tag}_{\lambda}^{3,(j)})$.
- **Computing recieved identities:**
  – Let $\mathcal{I} \subseteq ([\mathsf{p}]\times[\Gamma])$ be a set of identities which are queried on two or more keys,

$$\bigcup_{j\in\mathsf{q},j'\in\mathsf{q},i_1\in[\mathsf{p}]\ \wedge\ j\neq j'} \left((\mathsf{tag}_{i_1}^{1,(j)}) \cap (\mathsf{tag}_{i_1}^{1,(j')})\right).$$

- **Computing cover free elements:**
  For $j \in \mathsf{q}$, let $\mathcal{K} = \mathcal{K}^* = \emptyset$. Let $\mathcal{K}^*$ be an ordered set.
  - Let $\mathcal{K}_1 = \cup_{i_2 \in [\mathsf{p}']} \mathsf{tag}_{i_2}^{2,(j)}$.
  - Pick any element $\mathsf{tag}_{i_2^*}^{2,(j)} \in \mathcal{K}_1 \setminus \mathcal{K} \in ([\mathsf{p}'] \times [\Delta])$ and add it $(\mathsf{tag}_{i_2^*}^{2,(j)})$ to $\mathcal{K}^*$ such that $\mathcal{K}_j^* = \mathsf{tag}_{i_2^*}^{2,(j)}$.
  - Add $\mathcal{K}_1$ to set $\mathcal{K}$. i.e. $\mathcal{K} = \mathcal{K} \cup \mathcal{K}_1$.
- Let $\mathcal{U}(\cdot,\cdot)$ be a universal circuit that takes a circuit C and an input x and evaluates $C(x)$ where length of C is λ and length of x is ℓ bits. Let $\mathcal{W}$ refer to all wires of said universal circuit.
- Pick N unique nonzero field elements f_{tag^1} where $\mathsf{tag}^1 \in [\mathsf{p}] \times [\Gamma]$.
- For every possible $\mathsf{tag}^3 = (i_3, b) \in [\lambda] \times \{0,1\}$,
  - Sample a degree t polynomial $\delta_{\mathsf{tag}^3}(\cdot)$ over $\mathbb{F}$ whose constant term is b.
- For every wire $w \in \mathcal{W}, b \in \{0,1\}, \mathsf{tag}^2 \in ([\mathsf{p}'] \times [\Delta])$,
  - Sample $\mathsf{sd}_{w,b}^{\mathsf{tag}^2} \xleftarrow{R} \{0,1\}^\kappa$.
  - Set $k_{w,b}^{\mathsf{tag}^2} = \mathsf{PRG}(\mathsf{sd}_{w,b}^{\mathsf{tag}^2})$.
  - Sample $r_w^{\mathsf{tag}^2} \xleftarrow{R} \{0,1\}$.
  - For $h \in [\kappa]$,
    - Sample a degree t polynomial $\zeta_{w,b,h}^{\mathsf{tag}^2}(\cdot)$ over $\mathbb{F}$ whose constant term is h^{th} bit of $\mathsf{sd}_{w,b}^{\mathsf{tag}^2}$.
  - For $h \in [4 \cdot \mathsf{p}' \cdot \kappa + 4]$,
    - Sample a degree t polynomial $\xi_{w,b,h}^{\mathsf{tag}^2}(\cdot)$ over $\mathbb{F}$ whose constant term is h^{th} bit of $k_{w,b}^{\mathsf{tag}^2}$.
  - Sample a degree t polynomial $\eta_w^{\mathsf{tag}^2}(\cdot)$ over $\mathbb{F}$ whose constant term is $r_w^{(\mathsf{tag}^2)}$.
- For every $\mathsf{tag}^2 \in ([\mathsf{p}'] \times [\Delta]), \mathsf{h} \in [\mathsf{s}'']$,
  - Sample a degree $\mathsf{p}-1$ polynomial $\nu_{\mathsf{h}}^{\mathsf{tag}^2}(\cdot)$ over $\mathbb{F}$ whose constant term is 0.
- For every input $i \in \ell, h \in [\log|\mathbb{F}|]$,
  - Sample a degree-t polynomial $\mu_{i,h}(\cdot)$ over $\mathbb{F}$ whose constant term is $x_{i,h}$ where x_i denotes the i^{th} coordinate of $x \in \mathbb{F}$ and is represented as bits of length $\log|F|$.
- For $\mathsf{tag}^1 \in [\mathsf{p}] \times [\Gamma], \mathsf{h} \in [\mathsf{s}'']$, let $\mathcal{V}^{\mathsf{tag}^1,\mathsf{h}}[\{\delta_{\mathsf{tag}^3}(f_{\mathsf{tag}^1})\}_{\mathsf{tag}^3 \in [\lambda]\times\{0,1\}}]$ be the circuit that takes as inputs a circuit C of size λ and $E^\mu, E^\zeta, E^\xi, E^\eta, E^\nu$ and computes an arithmetization of the following circuit.
  - Let $C_1, \ldots C_\lambda$ be the binary representation of C.
  - Let $E_C^\delta = \{\delta_{(i,C_i)}(f_{\mathsf{tag}^1})\}_{i\in[\lambda]}$.
  - Compute $G = \mathsf{CorrGarb}(E_C^\delta, E^\mu, E^\zeta, E^\xi, E^\eta) + \sum_{e \in E^\nu} e$.
  - Output the h^{th} bit of the G.
- Let $\mathsf{ct} = (\mathsf{ct}^1, \mathsf{ct}^2, \mathsf{ct}^3, \mathsf{E}^\mu, \mathsf{E}^\zeta, \mathsf{E}^\xi, \mathsf{E}^\eta, \mathsf{E}^\nu)$ be $(\emptyset,\emptyset,\emptyset,\emptyset,\emptyset,\emptyset,\emptyset,\emptyset)$. We assume that these sets are ordered sets.

- For $\mathsf{tag}^1 \in ([\mathsf{p}] \times [\Gamma]), \mathsf{h} \in [\mathsf{s}'']$,
  - Garble the circuit i.e. $(\tilde{\mathcal{V}}^{\mathsf{tag}^1,\mathsf{h}}, \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}) \leftarrow \mathsf{GC.Garble}(1^\kappa, \mathcal{V}^{\mathsf{tag}^1,\mathsf{h}})$.
  - Add the garbled circuit $\tilde{\mathcal{V}}^{\mathsf{tag}^1,\mathsf{h}}$ to the set ct^1.
  - For every possible $\mathsf{tag}^2 = (i_2, \delta) \in ([\mathsf{p}'] \times [\Delta])$,
    - Let $\hat{E}^{\zeta}_{\mathsf{tag}^1,\mathsf{tag}^2} = \{\zeta^{\mathsf{tag}^2}_{w,b,h}(f_{\mathsf{tag}^1})\}_{w\in\mathcal{W},b\in\{0,1\},h\in[\kappa]}$.
      Let $\mathsf{input}_\zeta = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\zeta_{i_2},\hat{E}^\zeta_{\mathsf{tag}^1,\mathsf{tag}^2}}$.
      Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_\zeta))$ to E^ζ.
    - Let $\hat{E}^{\xi}_{\mathsf{tag}^1,\mathsf{tag}^2} = \{\xi^{\mathsf{tag}^2}_{w,b,h}(f_{\mathsf{tag}^1})\}_{w\in\mathcal{W},b\in\{0,1\},h\in[4\mathsf{p}'\kappa+4]}$.
      Let $\mathsf{input}_\xi = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\xi_{i_2},\hat{E}^\xi_{\mathsf{tag}^1,\mathsf{tag}^2}}$.
      Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_\xi))$ to E^ξ.
    - Let $\hat{E}^{\eta}_{\mathsf{tag}^1,\mathsf{tag}^2} = \{\eta^{\mathsf{tag}^2}_{w}(f_{\mathsf{tag}^1})\}_{w\in\mathcal{W}}$.
      Let $\mathsf{input}_\eta = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\eta_{i_2},\hat{E}^\eta_{\mathsf{tag}^1,\mathsf{tag}^2}}$.
      Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_\eta))$ to E^η.
    - Let $\hat{E}^{\nu}_{\mathsf{tag}^1,\mathsf{tag}^2,\mathsf{h}} = \{\nu^{\mathsf{tag}^2}_{\mathsf{h}}(f_{\mathsf{tag}^1})\}$.
      Let $\mathsf{input}_\nu = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\nu_{i_2},\hat{E}^\nu_{\mathsf{tag}^1,\mathsf{tag}^2,\mathsf{h}}}$.
      Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_\nu))$ to E^ν.
  - For every possible $\mathsf{tag}^3 = (i, b) \in ([\lambda] \times \{0,1\})$,
    - Let $\mathsf{input}_2 = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{\{C_i\},\{b\}}$.
    - $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3,\mathsf{h}} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_2, (\mathsf{tag}^1, \mathsf{tag}^3), (\mathsf{input}_2))$.
    - Add $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3,\mathsf{h}}$ to the set ct^2.
  - Let $\hat{E}^{\mu}_{\mathsf{tag}^1} = \{\mu_{i,h}(f_{\mathsf{tag}^1})\}_{i\in\ell,h\in[\log|\mathbb{F}|]}$.
  - Let $\mathsf{input}_\mu = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\mu,\hat{E}^\mu_{\mathsf{tag}^1}}$.
  - Add input_μ to the set E^μ.
  - Add f_{tag^1} to the set ct^3.
- Output $\mathsf{ct} = (\mathsf{ct}^1, \mathsf{ct}^2, \mathsf{ct}^3, \mathsf{E}^\mu, \mathsf{E}^\zeta, \mathsf{E}^\xi, \mathsf{E}^\eta, \mathsf{E}^\nu)$.

4. **Guess:** $\mathcal{A}$ outputs a bit b.

Game 2. In this game, we proceed with the game only if the randomly selected subsets satisfy some combinatorial properties required to allow us to properly program our ciphertexts.

1. Adversary $\mathcal{A}$ outputs circuit parameters $1^\lambda, 1^\mathsf{D}$ and collusion bound q.
2. **Setup Phase:. . . ,Keygen query phase:. . . ,Challenge message:. . . .**
3. **Encryption phase:** $\mathcal{C}$ computes $\mathsf{S\text{-}FE.Enc}(\mathsf{mpk}, x = x_1x_2\ldots x_\ell, 1^\mathsf{q}) \to \mathsf{ct}$. It sends this to $\mathcal{A}$.
   - **Parsing:**

- Let $|\Pi^x| = \mathsf{q}$ (wlog the cardinality is the maximum number of queries allowed).
- Parse $\Pi^x = ((C_1, C_1(x)), \ldots, (C_{\mathsf{q}}, C_{\mathsf{q}}(x)))$.
- $\forall j \in [\mathsf{q}]$, let sk_{C_j} be the reply to query C_j to S_1.
- Parse sk_{C_j} as $(\mathsf{sk}^{1,(j)}, \mathsf{sk}^{2,(j)}, \mathsf{taglist}^{1,(j)}, \mathsf{taglist}^{2,(j)}, \mathsf{taglist}^{3,(j)})$.
- Parse $\mathsf{taglist}^{1,(j)} = (\mathsf{tag}_1^{1,(j)}, \ldots, \mathsf{tag}_{\mathsf{p}}^{1,(j)})$.
- Parse $\mathsf{taglist}^{2,(j)} = (\mathsf{tag}_1^{2,(j)}, \ldots, \mathsf{tag}_{\mathsf{p}'}^{2,(j)})$.
- Parse $\mathsf{taglist}^{3,(j)} = (\mathsf{tag}_1^{3,(j)}, \ldots, \mathsf{tag}_{\lambda}^{3,(j)})$.

- **Computing recieved identities:**
  - Let $\mathcal{I} \subseteq ([\mathsf{p}] \times [\Gamma])$ be a set of identities which are queried on two or more keys,

$$\bigcup_{j \in \mathsf{q}, j' \in \mathsf{q}, i_1 \in [\mathsf{p}] \ \wedge \ j \neq j'} \left((\mathsf{tag}_{i_1}^{1,(j)}) \cap (\mathsf{tag}_{i_1}^{1,(j')}) \right).$$

  - If $|\mathcal{I}| > \mathsf{t}$, output $\perp$.
- **Computing cover free elements:**
  For $j \in \mathsf{q}$, let $\mathcal{K} = \mathcal{K}^* = \emptyset$. Let $\mathcal{K}^*$ be an ordered set.
  - Let $\mathcal{K}_1 = \cup_{i_2 \in [\mathsf{p}']} \mathsf{tag}_{i_2}^{2,(j)}$.
  - Pick any element $\mathsf{tag}_{i_2^*}^{2,(j)} \in \mathcal{K}_1 \setminus \mathcal{K} \in ([\mathsf{p}'] \times [\Delta])$ and add it $(\mathsf{tag}_{i_2^*}^{2,(j)})$ to $\mathcal{K}^*$ such that $\mathcal{K}_j^* = \mathsf{tag}_{i_2^*}^{2,(j)}$. If no such element exists, output $\perp$
  - Add $\mathcal{K}_1$ to set $\mathcal{K}$. i.e. $\mathcal{K} = \mathcal{K} \cup \mathcal{K}_1$.
- Let $\mathcal{U}(\cdot, \cdot)$ be a universal circuit that takes a circuit C and an input x and evaluates $C(x)$ where length of C is λ and length of x is ℓ bits. Let $\mathcal{W}$ refer to all wires of said universal circuit.
- Pick N unique nonzero field elements f_{tag^1} where $\mathsf{tag}^1 \in [\mathsf{p}] \times [\Gamma]$.
- For every possible $\mathsf{tag}^3 = (i_3, b) \in [\lambda] \times \{0, 1\}$,
  - Sample a degree t polynomial $\delta_{\mathsf{tag}^3}(\cdot)$ over $\mathbb{F}$ whose constant term is b.
- For every wire $w \in \mathcal{W}, b \in \{0, 1\}, \mathsf{tag}^2 \in ([\mathsf{p}'] \times [\Delta])$,
  - Sample $\mathsf{sd}_{w,b}^{\mathsf{tag}^2} \xleftarrow{R} \{0, 1\}^\kappa$.
  - Set $k_{w,b}^{\mathsf{tag}^2} = \mathsf{PRG}(\mathsf{sd}_{w,b}^{\mathsf{tag}^2})$.
  - Sample $r_w^{\mathsf{tag}^2} \xleftarrow{R} \{0, 1\}$.
  - For $h \in [\kappa]$,
    - Sample a degree t polynomial $\zeta_{w,b,h}^{\mathsf{tag}^2}(\cdot)$ over $\mathbb{F}$ whose constant term is h^{th} bit of $\mathsf{sd}_{w,b}^{\mathsf{tag}^2}$.
  - For $h \in [4 \cdot \mathsf{p}' \cdot \kappa + 4]$,
    - Sample a degree t polynomial $\xi_{w,b,h}^{\mathsf{tag}^2}(\cdot)$ over $\mathbb{F}$ whose constant term is h^{th} bit of $k_{w,b}^{\mathsf{tag}^2}$.
  - Sample a degree t polynomial $\eta_w^{\mathsf{tag}^2}(\cdot)$ over $\mathbb{F}$ whose constant term is $r_w^{(\mathsf{tag}^2)}$.

- For every $\mathsf{tag}^2 \in ([\mathsf{p}'] \times [\Delta]), \mathsf{h} \in [\mathsf{s}'']$,
  - Sample a degree $\mathsf{p}-1$ polynomial $\nu_{\mathsf{h}}^{\mathsf{tag}^2}(\cdot)$ over $\mathbb{F}$ whose constant term is 0.
- For every input $i \in \ell, h \in [\log|\mathbb{F}|]$,
  - Sample a degree-t polynomial $\mu_{i,h}(\cdot)$ over $\mathbb{F}$ whose constant term is $x_{i,h}$ where x_i denotes the i^{th} coordinate of $x \in \mathbb{F}$ and is represented as bits of length $\log|F|$.
- For $\mathsf{tag}^1 \in [\mathsf{p}] \times [\Gamma], \mathsf{h} \in [\mathsf{s}'']$, let $\mathcal{V}^{\mathsf{tag}^1,\mathsf{h}}[\{\delta_{\mathsf{tag}^3}(f_{\mathsf{tag}^1})\}_{\mathsf{tag}^3 \in [\lambda]\times\{0,1\}}]$ be the circuit that takes as inputs a circuit C of size λ and $E^\mu, E^\zeta, E^\xi, E^\eta, E^\nu$ and computes an arithmetization of the following circuit.
  - Let $C_1, \ldots C_\lambda$ be the binary representation of C.
  - Let $E_C^\delta = \{\delta_{(i,C_i)}(f_{\mathsf{tag}^1})\}_{i\in[\lambda]}$.
  - Compute $G = \mathsf{CorrGarb}(E_C^\delta, E^\mu, E^\zeta, E^\xi, E^\eta) + \sum_{e\in E^\nu} e$.
  - Output the h^{th} bit of the G.
- Let $\mathsf{ct} = (\mathsf{ct}^1, \mathsf{ct}^2, \mathsf{ct}^3, \mathsf{E}^\mu, \mathsf{E}^\zeta, \mathsf{E}^\xi, \mathsf{E}^\eta, \mathsf{E}^\nu)$ be $(\emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset)$. We assume that these sets are ordered sets.
- For $\mathsf{tag}^1 \in ([\mathsf{p}] \times [\Gamma]), \mathsf{h} \in [\mathsf{s}'']$,
  - Garble the circuit i.e. $(\tilde{\mathcal{V}}^{\mathsf{tag}^1,\mathsf{h}}, \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}) \leftarrow \mathsf{GC.Garble}(1^\kappa, \mathcal{V}^{\mathsf{tag}^1,\mathsf{h}})$.
  - Add the garbled circuit $\tilde{\mathcal{V}}^{\mathsf{tag}^1,\mathsf{h}}$ to the set ct^1.
  - For every possible $\mathsf{tag}^2 = (i_2, \delta) \in ([\mathsf{p}'] \times [\Delta])$,
    - Let $\hat{E}^\zeta_{\mathsf{tag}^1,\mathsf{tag}^2} = \{\zeta^{\mathsf{tag}^2}_{w,b,h}(f_{\mathsf{tag}^1})\}_{w\in\mathcal{W},b\in\{0,1\},h\in[\kappa]}$.
      Let $\mathsf{input}_\zeta = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\zeta_{i_2},\hat{E}^\zeta_{\mathsf{tag}^1,\mathsf{tag}^2}}$.
      Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_\zeta))$ to E^ζ.
    - Let $\hat{E}^\xi_{\mathsf{tag}^1,\mathsf{tag}^2} = \{\xi^{\mathsf{tag}^2}_{w,b,h}(f_{\mathsf{tag}^1})\}_{w\in\mathcal{W},b\in\{0,1\},h\in[4\mathsf{p}'\kappa+4]}$.
      Let $\mathsf{input}_\xi = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\xi_{i_2},\hat{E}^\xi_{\mathsf{tag}^1,\mathsf{tag}^2}}$.
      Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_\xi))$ to E^ξ.
    - Let $\hat{E}^\eta_{\mathsf{tag}^1,\mathsf{tag}^2} = \{\eta^{\mathsf{tag}^2}_w(f_{\mathsf{tag}^1})\}_{w\in\mathcal{W}}$.
      Let $\mathsf{input}_\eta = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\eta_{i_2},\hat{E}^\eta_{\mathsf{tag}^1,\mathsf{tag}^2}}$.
      Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_\eta))$ to E^η.
    - Let $\hat{E}^\nu_{\mathsf{tag}^1,\mathsf{tag}^2,\mathsf{h}} = \{\nu^{\mathsf{tag}^2}_{\mathsf{h}}(f_{\mathsf{tag}^1})\}$.
      Let $\mathsf{input}_\nu = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\nu_{i_2},\hat{E}^\nu_{\mathsf{tag}^1,\mathsf{tag}^2,\mathsf{h}}}$.
      Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_\nu))$ to E^ν.
  - For every possible $\mathsf{tag}^3 = (i, b) \in ([\lambda] \times \{0,1\})$,
    - Let $\mathsf{input}_2 = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{\{C_i\},\{b\}}$.
    - $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3,\mathsf{h}} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_2, (\mathsf{tag}^1, \mathsf{tag}^3), (\mathsf{input}_2))$.
    - Add $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3,\mathsf{h}}$ to the set ct^2.
  - Let $\hat{E}^\mu_{\mathsf{tag}^1} = \{\mu_{i,h}(f_{\mathsf{tag}^1})\}_{i\in\ell,h\in[\log|\mathbb{F}|]}$.

- Let $\mathsf{input}_\mu = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\mu,\hat{E}^\mu_{\mathsf{tag}^1}}$.
- Add input_μ to the set E^μ.
- Add f_{tag^1} to the set ct^3.

- Output $\mathsf{ct} = (\mathsf{ct}^1, \mathsf{ct}^2, \mathsf{ct}^3, \mathsf{E}^\mu, \mathsf{E}^\zeta, \mathsf{E}^\xi, \mathsf{E}^\eta, \mathsf{E}^\nu)$.

4. **Guess:** $\mathcal{A}$ outputs a bit b.

Game 3. In this game, we reorder the way we sample polynomials in the ciphertext.

1. Adversary $\mathcal{A}$ outputs circuit parameters $1^\lambda, 1^\mathsf{D}$ and collusion bound q.
2. **Setup Phase:. . . ,Keygen query phase:. . . ,Challenge message:. . . .**
3. **Encryption phase:** $\mathcal{C}$ computes $\mathsf{S\text{-}FE.Enc}(\mathsf{mpk}, x = x_1x_2\ldots x_\ell, 1^\mathsf{q}) \to \mathsf{ct}$. It sends this to $\mathcal{A}$.

- **Parsing:**
  - Let $|\Pi^x| = \mathsf{q}$ (wlog the cardinality is the maximum number of queries allowed).
  - Parse $\Pi^x = ((C_1, C_1(x)), \ldots, (C_\mathsf{q}, C_\mathsf{q}(x)))$.
  - $\forall j \in [\mathsf{q}]$, let sk_{C_j} be the reply to query C_j to S_1.
  - Parse sk_{C_j} as $(\mathsf{sk}^{1,(j)}, \mathsf{sk}^{2,(j)}, \mathsf{taglist}^{1,(j)}, \mathsf{taglist}^{2,(j)}, \mathsf{taglist}^{3,(j)})$.
  - Parse $\mathsf{taglist}^{1,(j)} = (\mathsf{tag}^{1,(j)}_1, \ldots, \mathsf{tag}^{1,(j)}_\mathsf{p})$.
  - Parse $\mathsf{taglist}^{2,(j)} = (\mathsf{tag}^{2,(j)}_1, \ldots, \mathsf{tag}^{2,(j)}_{\mathsf{p}'})$.
  - Parse $\mathsf{taglist}^{3,(j)} = (\mathsf{tag}^{3,(j)}_1, \ldots, \mathsf{tag}^{3,(j)}_\lambda)$.
- **Computing recieved identities:**
  - Let $\mathcal{I} \subseteq ([\mathsf{p}] \times [\Gamma])$ be a set of identities which are queried on two or more keys,
$$\bigcup_{j\in\mathsf{q}, j'\in\mathsf{q}, i_1\in[\mathsf{p}] \ \wedge \ j\neq j'} \left((\mathsf{tag}^{1,(j)}_{i_1}) \cap (\mathsf{tag}^{1,(j')}_{i_1})\right).$$
  - If $|\mathcal{I}| > \mathsf{t}$, output $\perp$.
- **Computing cover free elements:**
  For $j \in \mathsf{q}$, let $\mathcal{K} = \mathcal{K}^* = \emptyset$. Let $\mathcal{K}^*$ be an ordered set.
  - Let $\mathcal{K}_1 = \cup_{i_2\in[\mathsf{p}']}\mathsf{tag}^{2,(j)}_{i_2}$.
  - Pick any element $\mathsf{tag}^{2,(j)}_{i_2^*} \in \mathcal{K}_1 \setminus \mathcal{K} \in ([\mathsf{p}'] \times [\Delta])$ and add it $(\mathsf{tag}^{2,(j)}_{i_2^*})$ to $\mathcal{K}^*$ such that $\mathcal{K}^*_j = \mathsf{tag}^{2,(j)}_{i_2^*}$. If no such element exists, output $\perp$
  - Add $\mathcal{K}_1$ to set $\mathcal{K}$. i.e. $\mathcal{K} = \mathcal{K} \cup \mathcal{K}_1$.
- Let $\mathcal{U}(\cdot,\cdot)$ be a universal circuit that takes a circuit C and an input x and evaluates $C(x)$ where length of C is λ and length of x is ℓ bits. Let $\mathcal{W}$ refer to all wires of said universal circuit.
- Pick N unique nonzero field elements f_{tag^1} where $\mathsf{tag}^1 \in [\mathsf{p}] \times [\Gamma]$.
- For every possible $\mathsf{tag}^3 = (i_3, b) \in [\lambda] \times \{0,1\}$,

- Sample a degree t polynomial $\delta_{\mathsf{tag}^3}(\cdot)$ over $\mathbb{F}$ whose constant term is b.

- For every wire $w \in \mathcal{W}, b \in \{0,1\}, \mathsf{tag}^2 \in ([\mathsf{p}'] \times [\Delta])$,
  - Sample $\mathsf{sd}_{w,b}^{\mathsf{tag}^2} \xleftarrow{R} \{0,1\}^\kappa$.
  - Set $k_{w,b}^{\mathsf{tag}^2} = \mathsf{PRG}(\mathsf{sd}_{w,b}^{\mathsf{tag}^2})$.
  - Sample $r_w^{\mathsf{tag}^2} \xleftarrow{R} \{0,1\}$.
  - For $h \in [\kappa]$,
    - Sample a degree t polynomial $\zeta_{w,b,h}^{\mathsf{tag}^2}(\cdot)$ over $\mathbb{F}$ whose constant term is h^{th} bit of $\mathsf{sd}_{w,b}^{\mathsf{tag}^2}$.
  - For $h \in [4 \cdot \mathsf{p}' \cdot \kappa + 4]$,
    - Sample a degree t polynomial $\xi_{w,b,h}^{\mathsf{tag}^2}(\cdot)$ over $\mathbb{F}$ whose constant term is h^{th} bit of $k_{w,b}^{\mathsf{tag}^2}$.
  - Sample a degree t polynomial $\eta_w^{\mathsf{tag}^2}(\cdot)$ over $\mathbb{F}$ whose constant term is $r_w^{(\mathsf{tag}^2)}$.

- For $\mathsf{tag}^1 \in [\mathsf{p}] \times [\Gamma], \mathsf{h} \in [\mathsf{s}'']$, let $\mathcal{V}^{\mathsf{tag}^1,\mathsf{h}}[\{\delta_{\mathsf{tag}^3}(f_{\mathsf{tag}^1})\}_{\mathsf{tag}^3 \in [\lambda] \times \{0,1\}}]$ be the circuit that takes as inputs a circuit C of size λ and $E^\mu, E^\zeta, E^\xi, E^\eta, E^\nu$ and computes an arithmetization of the following circuit.
  - Let $C_1, \ldots C_\lambda$ be the binary representation of C.
  - Let $E_C^\delta = \{\delta_{(i,C_i)}(f_{\mathsf{tag}^1})\}_{i \in [\lambda]}$.
  - Compute $G = \mathsf{CorrGarb}(E_C^\delta, E^\mu, E^\zeta, E^\xi, E^\eta) + \sum_{e \in E^\nu} e$.
  - Output the h^{th} bit of the G.

- For $\mathsf{tag}^1 \in [\mathsf{p}] \times [\Gamma], j \in \mathsf{q}, \mathsf{h} \in [\mathsf{s}'']$
  - Sample a random degree $\mathsf{p}-1$ polynomials $\tau_{j,\mathsf{h}}(\cdot)$ with constant term as the h^{th} bit of $\mathsf{CorrGarb}(C, x, \zeta(0), \xi(0), \eta(0), \{\nu^{\mathsf{tag}^2}(0)\}_{\mathsf{tag}^2 \in \mathsf{taglist}^{2,(j)}})$.

- For every $\mathsf{tag}^2 \in ([\mathsf{p}'] \times [\Delta]) \backslash \mathcal{K}^*, \mathsf{h} \in [\mathsf{s}'']$,
  - Sample a degree $\mathsf{p}-1$ polynomial $\nu_{\mathsf{h}}^{\mathsf{tag}^2}(\cdot)$ over $\mathbb{F}$ whose constant term is 0.

- For every $\mathsf{tag}^{2\prime} \in \mathcal{K}^*, \mathsf{h} \in [\mathsf{s}'']$,
  - Set $\nu_{\mathsf{h}}^{\mathsf{tag}^2}(\cdot)$ to be the polynomial interpolation which satisfies $\forall \mathsf{tag}^1 \in \mathcal{I}$,

$$\tau_{j,\mathsf{h}}(f_{\mathsf{tag}^1}) = G(\delta_C(f_{\mathsf{tag}^1}), \mu(f_{\mathsf{tag}^1}), \zeta(f_{\mathsf{tag}^1}), \xi(f_{\mathsf{tag}^1}), \eta(f_{\mathsf{tag}^1}), \{\nu^{\mathsf{tag}^2}(f_{\mathsf{tag}^1})\}_{\mathsf{tag}^2 \in \mathsf{taglist}^{2,(j)}}).$$

- For every input $i \in \ell, h \in [\log |\mathbb{F}|]$,
  - Sample a degree-t polynomial $\mu_{i,h}(\cdot)$ over $\mathbb{F}$ whose constant term is $x_{i,h}$ where x_i denotes the i^{th} coordinate of $x \in \mathbb{F}$ and is represented as bits of length $\log |F|$.

- Let $\mathsf{ct} = (\mathsf{ct}^1, \mathsf{ct}^2, \mathsf{ct}^3, \mathsf{E}^\mu, \mathsf{E}^\zeta, \mathsf{E}^\xi, \mathsf{E}^\eta, \mathsf{E}^\nu)$ be $(\emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset)$. We assume that these sets are ordered sets.

- For $\mathsf{tag}^1 \in ([\mathsf{p}] \times [\Gamma]), \mathsf{h} \in [\mathsf{s}'']$,
  - Garble the circuit i.e. $(\tilde{\mathcal{V}}^{\mathsf{tag}^1,\mathsf{h}}, \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}) \leftarrow \mathsf{GC.Garble}(1^\kappa, \mathcal{V}^{\mathsf{tag}^1,\mathsf{h}})$.
  - Add the garbled circuit $\tilde{\mathcal{V}}^{\mathsf{tag}^1,\mathsf{h}}$ to the set ct^1.

- For every possible $\mathsf{tag}^2 = (i_2, \delta) \in ([\mathsf{p}'] \times [\Delta])$,
  - $*$ Let $\hat{E}^{\zeta}_{\mathsf{tag}^1,\mathsf{tag}^2} = \{\zeta^{\mathsf{tag}^2}_{w,b,h}(f_{\mathsf{tag}^1})\}_{w\in\mathcal{W},b\in\{0,1\},h\in[\kappa]}$.
    Let $\mathsf{input}_{\zeta} = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^{\zeta}_{i_2},\hat{E}^{\zeta}_{\mathsf{tag}^1,\mathsf{tag}^2}}$.
    Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_{\zeta}))$ to E^{ζ}.
  - $*$ Let $\hat{E}^{\xi}_{\mathsf{tag}^1,\mathsf{tag}^2} = \{\xi^{\mathsf{tag}^2}_{w,b,h}(f_{\mathsf{tag}^1})\}_{w\in\mathcal{W},b\in\{0,1\},h\in[4\mathsf{p}'\kappa+4]}$.
    Let $\mathsf{input}_{\xi} = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^{\xi}_{i_2},\hat{E}^{\xi}_{\mathsf{tag}^1,\mathsf{tag}^2}}$.
    Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_{\xi}))$ to E^{ξ}.
  - $*$ Let $\hat{E}^{\eta}_{\mathsf{tag}^1,\mathsf{tag}^2} = \{\eta^{\mathsf{tag}^2}_{w}(f_{\mathsf{tag}^1})\}_{w\in\mathcal{W}}$.
    Let $\mathsf{input}_{\eta} = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^{\eta}_{i_2},\hat{E}^{\eta}_{\mathsf{tag}^1,\mathsf{tag}^2}}$.
    Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_{\eta}))$ to E^{η}.
  - $*$ Let $\hat{E}^{\nu}_{\mathsf{tag}^1,\mathsf{tag}^2,\mathsf{h}} = \{\nu^{\mathsf{tag}^2}_{\mathsf{h}}(f_{\mathsf{tag}^1})\}$.
    Let $\mathsf{input}_{\nu} = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^{\nu}_{i_2},\hat{E}^{\nu}_{\mathsf{tag}^1,\mathsf{tag}^2,\mathsf{h}}}$.
    Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_{\nu}))$ to E^{ν}.
- For every possible $\mathsf{tag}^3 = (i, b) \in ([\lambda] \times \{0,1\})$,
  - $*$ Let $\mathsf{input}_2 = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{\{C_i\},\{b\}}$.
  - $*$ $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3,\mathsf{h}} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_2, (\mathsf{tag}^1, \mathsf{tag}^3), (\mathsf{input}_2))$.
  - $*$ Add $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3,\mathsf{h}}$ to the set ct^2.
- Let $\hat{E}^{\mu}_{\mathsf{tag}^1} = \{\mu_{i,h}(f_{\mathsf{tag}^1})\}_{i\in\ell,h\in[\log|\mathbb{F}|]}$.
- Let $\mathsf{input}_{\mu} = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^{\mu},\hat{E}^{\mu}_{\mathsf{tag}^1}}$.
- Add input_{μ} to the set E^{μ}.
- Add f_{tag^1} to the set ct^3.

- Output $\mathsf{ct} = (\mathsf{ct}^1, \mathsf{ct}^2, \mathsf{ct}^3, \mathsf{E}^{\mu}, \mathsf{E}^{\zeta}, \mathsf{E}^{\xi}, \mathsf{E}^{\eta}, \mathsf{E}^{\nu})$.

4. **Guess:** $\mathcal{A}$ outputs a bit b.

Game 4. In this game, we erase certain encryptions when the IBE ciphertext is not under a key given to the adversary.

1. Adversary $\mathcal{A}$ outputs circuit parameters $1^{\lambda}, 1^{\mathsf{D}}$ and collusion bound q.
2. **Setup Phase:. . . ,Keygen query phase:. . . ,Challenge message:. . . .**
3. **Encryption phase:** $\mathcal{C}$ computes $\mathsf{S\text{-}FE.Enc}(\mathsf{mpk}, x = x_1x_2\ldots x_{\ell}, 1^{\mathsf{q}}) \to \mathsf{ct}$. It sends this to $\mathcal{A}$.

- **Parsing:**
  - Let $|\Pi^x| = \mathsf{q}$ (wlog the cardinality is the maximum number of queries allowed).
  - Parse $\Pi^x = ((C_1, C_1(x)), \ldots, (C_{\mathsf{q}}, C_{\mathsf{q}}(x)))$.
  - $\forall j \in [\mathsf{q}]$, let sk_{C_j} be the reply to query C_j to S_1.

- Parse sk_{C_j} as $(\mathsf{sk}^{1,(j)}, \mathsf{sk}^{2,(j)}, \mathsf{taglist}^{1,(j)}, \mathsf{taglist}^{2,(j)}, \mathsf{taglist}^{3,(j)})$.
- Parse $\mathsf{taglist}^{1,(j)} = (\mathsf{tag}_1^{1,(j)}, \ldots, \mathsf{tag}_{\mathsf{p}}^{1,(j)})$.
- Parse $\mathsf{taglist}^{2,(j)} = (\mathsf{tag}_1^{2,(j)}, \ldots, \mathsf{tag}_{\mathsf{p}'}^{2,(j)})$.
- Parse $\mathsf{taglist}^{3,(j)} = (\mathsf{tag}_1^{3,(j)}, \ldots, \mathsf{tag}_{\lambda}^{3,(j)})$.

- **Computing recieved identities:**
  - Let $\mathcal{I} \subseteq ([\mathsf{p}] \times [\Gamma])$ be a set of identities which are queried on two or more keys,

$$\bigcup_{j \in \mathsf{q}, j' \in \mathsf{q}, i_1 \in [\mathsf{p}] \ \wedge \ j \neq j'} \left((\mathsf{tag}_{i_1}^{1,(j)}) \cap (\mathsf{tag}_{i_1}^{1,(j')}) \right).$$

  - If $|\mathcal{I}| > \mathsf{t}$, output $\perp$.
- **Computing cover free elements:**
  For $j \in \mathsf{q}$, let $\mathcal{K} = \mathcal{K}^* = \emptyset$. Let $\mathcal{K}^*$ be an ordered set.
  - Let $\mathcal{K}_1 = \cup_{i_2 \in [\mathsf{p}']} \mathsf{tag}_{i_2}^{2,(j)}$.
  - Pick any element $\mathsf{tag}_{i_2^*}^{2,(j)} \in \mathcal{K}_1 \setminus \mathcal{K} \in ([\mathsf{p}'] \times [\Delta])$ and add it $(\mathsf{tag}_{i_2^*}^{2,(j)})$ to $\mathcal{K}^*$ such that $\mathcal{K}_j^* = \mathsf{tag}_{i_2^*}^{2,(j)}$. If no such element exists, output $\perp$
  - Add $\mathcal{K}_1$ to set $\mathcal{K}$. i.e. $\mathcal{K} = \mathcal{K} \cup \mathcal{K}_1$.
- Let $\mathcal{U}(\cdot, \cdot)$ be a universal circuit that takes a circuit C and an input x and evaluates $C(x)$ where length of C is λ and length of x is ℓ bits. Let $\mathcal{W}$ refer to all wires of said universal circuit.
- Pick N unique nonzero field elements f_{tag^1} where $\mathsf{tag}^1 \in [\mathsf{p}] \times [\Gamma]$.
- For every possible $\mathsf{tag}^3 = (i_3, b) \in [\lambda] \times \{0, 1\}$,
  - Sample a degree t polynomial $\delta_{\mathsf{tag}^3}(\cdot)$ over $\mathbb{F}$ whose constant term is b.
- For every wire $w \in \mathcal{W}, b \in \{0, 1\}, \mathsf{tag}^2 \in ([\mathsf{p}'] \times [\Delta])$,
  - Sample $\mathsf{sd}_{w,b}^{\mathsf{tag}^2} \xleftarrow{R} \{0, 1\}^\kappa$.
  - Set $k_{w,b}^{\mathsf{tag}^2} = \mathsf{PRG}(\mathsf{sd}_{w,b}^{\mathsf{tag}^2})$.
  - Sample $r_w^{\mathsf{tag}^2} \xleftarrow{R} \{0, 1\}$.
  - For $h \in [\kappa]$,
    - Sample a degree t polynomial $\zeta_{w,b,h}^{\mathsf{tag}^2}(\cdot)$ over $\mathbb{F}$ whose constant term is h^{th} bit of $\mathsf{sd}_{w,b}^{\mathsf{tag}^2}$.
  - For $h \in [4 \cdot \mathsf{p}' \cdot \kappa + 4]$,
    - Sample a degree t polynomial $\xi_{w,b,h}^{\mathsf{tag}^2}(\cdot)$ over $\mathbb{F}$ whose constant term is h^{th} bit of $k_{w,b}^{\mathsf{tag}^2}$.
  - Sample a degree t polynomial $\eta_w^{\mathsf{tag}^2}(\cdot)$ over $\mathbb{F}$ whose constant term is $r_w^{(\mathsf{tag}^2)}$.
- For $\mathsf{tag}^1 \in [\mathsf{p}] \times [\Gamma], \mathsf{h} \in [\mathsf{s}'']$, let $\mathcal{V}^{\mathsf{tag}^1, \mathsf{h}}[\{\delta_{\mathsf{tag}^3}(f_{\mathsf{tag}^1})\}_{\mathsf{tag}^3 \in [\lambda] \times \{0,1\}}]$ be the circuit that takes as inputs a circuit C of size λ and $E^\mu, E^\zeta, E^\xi, E^\eta, E^\nu$ and computes an arithmetization of the following circuit.

- Let $C_1, \ldots C_\lambda$ be the binary representation of C.
- Let $E_C^\delta = \{\delta_{(i,C_i)}(f_{\mathsf{tag}^1})\}_{i\in[\lambda]}$.
- Compute $G = \mathsf{CorrGarb}(E_C^\delta, E^\mu, E^\zeta, E^\xi, E^\eta) + \sum_{e\in E^\nu} e$.
- Output the h^{th} bit of the G.

- For $\mathsf{tag}^1 \in [\mathsf{p}] \times [\Gamma], j \in \mathsf{q}, \mathsf{h} \in [\mathsf{s}'']$
  - Sample a random degree $\mathsf{p}-1$ polynomials $\tau_{j,\mathsf{h}}(\cdot)$ with constant term as the h^{th} bit of $\mathsf{CorrGarb}(C, x, \zeta(0), \xi(0), \eta(0), \{\nu^{\mathsf{tag}^2}(0)\}_{\mathsf{tag}^2\in\mathsf{taglist}^{2,(j)}})$.
- For every $\mathsf{tag}^2 \in ([\mathsf{p}'] \times [\Delta])\backslash\mathcal{K}^*, \mathsf{h} \in [\mathsf{s}'']$,
  - Sample a degree $\mathsf{p}-1$ polynomial $\nu_{\mathsf{h}}^{\mathsf{tag}^2}(\cdot)$ over $\mathbb{F}$ whose constant term is 0.
- For every $\mathsf{tag}^{2\prime} \in \mathcal{K}^*, \mathsf{h} \in [\mathsf{s}'']$,
  - Set $\nu_{\mathsf{h}}^{\mathsf{tag}^2}(\cdot)$ to be the polynomial interpolation which satisfies $\forall \mathsf{tag}^1 \in \mathcal{I}$,

$$\tau_{j,\mathsf{h}}(f_{\mathsf{tag}^1}) = G(\delta_C(f_{\mathsf{tag}^1}), \mu(f_{\mathsf{tag}^1}), \zeta(f_{\mathsf{tag}^1}), \xi(f_{\mathsf{tag}^1}), \eta(f_{\mathsf{tag}^1}), \{\nu^{\mathsf{tag}^2}(f_{\mathsf{tag}^1})\}_{\mathsf{tag}^2\in\mathsf{taglist}^{2,(j)}}).$$

- For every input $i \in \ell, h \in [\log|\mathbb{F}|]$,
  - Sample a degree-t polynomial $\mu_{i,h}(\cdot)$ over $\mathbb{F}$ whose constant term is $x_{i,h}$ where x_i denotes the i^{th} coordinate of $x \in \mathbb{F}$ and is represented as bits of length $\log|F|$.
- Let $\mathsf{ct} = (\mathsf{ct}^1, \mathsf{ct}^2, \mathsf{ct}^3, \mathsf{E}^\mu, \mathsf{E}^\zeta, \mathsf{E}^\xi, \mathsf{E}^\eta, \mathsf{E}^\nu)$ be $(\emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset)$. We assume that these sets are ordered sets.
- For $\mathsf{tag}^1 \in ([\mathsf{p}] \times [\Gamma]), \mathsf{h} \in [\mathsf{s}'']$,
  - Garble the circuit i.e. $(\tilde{\mathcal{V}}^{\mathsf{tag}^1,\mathsf{h}}, \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}) \leftarrow \mathsf{GC.Garble}(1^\kappa, \mathcal{V}^{\mathsf{tag}^1,\mathsf{h}})$.
  - Add the garbled circuit $\tilde{\mathcal{V}}^{\mathsf{tag}^1,\mathsf{h}}$ to the set ct^1.
  - For every possible $\mathsf{tag}^2 = (i_2, \delta) \in ([\mathsf{p}'] \times [\Delta])$,
    - If $\exists j : \mathsf{tag}^1 \in \mathsf{taglist}^{1,(j)} \wedge \mathsf{tag}^2 \in \mathsf{taglist}^{2,(j)}$
      - Let $\hat{E}^\zeta_{\mathsf{tag}^1,\mathsf{tag}^2} = \{\zeta^{\mathsf{tag}^2}_{w,b,h}(f_{\mathsf{tag}^1})\}_{w\in\mathcal{W},b\in\{0,1\},h\in[\kappa]}$.
        Let $\mathsf{input}_\zeta = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\zeta_{i_2}, \hat{E}^\zeta_{\mathsf{tag}^1,\mathsf{tag}^2}}$.
        Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_\zeta))$ to E^ζ.
      - Let $\hat{E}^\xi_{\mathsf{tag}^1,\mathsf{tag}^2} = \{\xi^{\mathsf{tag}^2}_{w,b,h}(f_{\mathsf{tag}^1})\}_{w\in\mathcal{W},b\in\{0,1\},h\in[4\mathsf{p}'\kappa+4]}$.
        Let $\mathsf{input}_\xi = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\xi_{i_2}, \hat{E}^\xi_{\mathsf{tag}^1,\mathsf{tag}^2}}$.
        Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_\xi))$ to E^ξ.
      - Let $\hat{E}^\eta_{\mathsf{tag}^1,\mathsf{tag}^2} = \{\eta^{\mathsf{tag}^2}_w(f_{\mathsf{tag}^1})\}_{w\in\mathcal{W}}$.
        Let $\mathsf{input}_\eta = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\eta_{i_2}, \hat{E}^\eta_{\mathsf{tag}^1,\mathsf{tag}^2}}$.
        Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_\eta))$ to E^η.
      - Let $\hat{E}^\nu_{\mathsf{tag}^1,\mathsf{tag}^2,\mathsf{h}} = \{\nu^{\mathsf{tag}^2}_{\mathsf{h}}(f_{\mathsf{tag}^1})\}$.
        Let $\mathsf{input}_\nu = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\nu_{i_2}, \hat{E}^\nu_{\mathsf{tag}^1,\mathsf{tag}^2,\mathsf{h}}}$.
        Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_\nu))$ to E^ν.

* Otherwise, add encryptions of appropriately padded 0's
  · Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), 0^{|\mathsf{input}_\zeta|})$ to E^ζ
  · Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), 0^{|\mathsf{input}_\xi|})$ to E^ξ
  · Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), 0^{|\mathsf{input}_\eta|})$ to E^η
  · Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), 0^{|\mathsf{input}_\nu|})$ to E^ν

– For every possible $\mathsf{tag}^3 = (i, b) \in ([\lambda] \times \{0,1\})$,
  * If $\exists j : \mathsf{tag}^1 \in \mathsf{taglist}^{1,(j)} \wedge \mathsf{tag}^3 \in \mathsf{taglist}^{3,(j)}$
    · Let $\mathsf{input}_2 = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{\{C_i\},\{b\}}$.
    · $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3,\mathsf{h}} = \mathsf{IBE.Enc}(\mathsf{IBE.pk}_2, (\mathsf{tag}^1, \mathsf{tag}^3), (\mathsf{input}_2))$.
    · Add $\mathsf{ct}^2_{\mathsf{tag}^1,\mathsf{tag}^3,\mathsf{h}}$ to the set ct^2.
  * Otherwise, add encryptions of appropriately padded 0's
    · Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_2, (\mathsf{tag}^1, \mathsf{tag}^3), 0^{|\mathsf{input}_2|})$ to ct^2

– Let $\hat{E}^\mu_{\mathsf{tag}^1} = \{\mu_{i,h}(f_{\mathsf{tag}^1})\}_{i\in\ell, h\in[\log|\mathbb{F}|]}$.

– Let $\mathsf{input}_\mu = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\mu,\hat{E}^\mu_{\mathsf{tag}^1}}$.

– Add input_μ to the set E^μ.

– Add f_{tag^1} to the set ct^3.

- Output $\mathsf{ct} = (\mathsf{ct}^1, \mathsf{ct}^2, \mathsf{ct}^3, \mathsf{E}^\mu, \mathsf{E}^\zeta, \mathsf{E}^\xi, \mathsf{E}^\eta, \mathsf{E}^\nu)$.

4. **Guess:** $\mathcal{A}$ outputs a bit b.

Game 5. In this game, we simulate the garbled circuits when only requested on one key.

1. Adversary $\mathcal{A}$ outputs circuit parameters $1^\lambda, 1^{\mathsf{D}}$ and collusion bound q.
2. **Setup Phase:...,Keygen query phase:...,Challenge message:....**
3. **Encryption phase:** $\mathcal{C}$ computes $\mathsf{S\text{-}FE.Enc}(\mathsf{mpk}, x = x_1x_2\dots x_\ell, 1^{\mathsf{q}}) \to \mathsf{ct}$. It sends this to $\mathcal{A}$.

- **Parsing:**
  – Let $|\Pi^x| = \mathsf{q}$ (wlog the cardinality is the maximum number of queries allowed).
  – Parse $\Pi^x = ((C_1, C_1(x)), \dots, (C_{\mathsf{q}}, C_{\mathsf{q}}(x)))$.
  – $\forall j \in [\mathsf{q}]$, let sk_{C_j} be the reply to query C_j to S_1.
  – Parse sk_{C_j} as $(\mathsf{sk}^{1,(j)}, \mathsf{sk}^{2,(j)}, \mathsf{taglist}^{1,(j)}, \mathsf{taglist}^{2,(j)}, \mathsf{taglist}^{3,(j)})$.
  – Parse $\mathsf{taglist}^{1,(j)} = (\mathsf{tag}^{1,(j)}_1, \dots, \mathsf{tag}^{1,(j)}_{\mathsf{p}})$.
  – Parse $\mathsf{taglist}^{2,(j)} = (\mathsf{tag}^{2,(j)}_1, \dots, \mathsf{tag}^{2,(j)}_{\mathsf{p}'})$.
  – Parse $\mathsf{taglist}^{3,(j)} = (\mathsf{tag}^{3,(j)}_1, \dots, \mathsf{tag}^{3,(j)}_{\lambda})$.
- **Computing recieved identities:**
  – Let $\mathcal{I} \subseteq ([\mathsf{p}] \times [\Gamma])$ be a set of identities which are queried on two or more keys,

$$\bigcup_{j\in\mathsf{q}, j'\in\mathsf{q}, i_1\in[\mathsf{p}] \ \wedge \ j\neq j'} \left((\mathsf{tag}^{1,(j)}_{i_1}) \cap (\mathsf{tag}^{1,(j')}_{i_1})\right).$$

- If $|\mathcal{I}| > \mathsf{t}$, output $\perp$.

- **Computing cover free elements:**
  For $j \in \mathsf{q}$, let $\mathcal{K} = \mathcal{K}^* = \emptyset$. Let $\mathcal{K}^*$ be an ordered set.
  - Let $\mathcal{K}_1 = \cup_{i_2 \in [\mathsf{p}']} \mathsf{tag}_{i_2}^{2,(j)}$.
  - Pick any element $\mathsf{tag}_{i_2^*}^{2,(j)} \in \mathcal{K}_1 \setminus \mathcal{K} \in ([\mathsf{p}'] \times [\Delta])$ and add it $(\mathsf{tag}_{i_2^*}^{2,(j)})$ to $\mathcal{K}^*$ such that $\mathcal{K}_j^* = \mathsf{tag}_{i_2^*}^{2,(j)}$. If no such element exists, output $\perp$
  - Add $\mathcal{K}_1$ to set $\mathcal{K}$. i.e. $\mathcal{K} = \mathcal{K} \cup \mathcal{K}_1$.

- Let $\mathcal{U}(\cdot,\cdot)$ be a universal circuit that takes a circuit C and an input x and evaluates $C(x)$ where length of C is λ and length of x is ℓ bits. Let $\mathcal{W}$ refer to all wires of said universal circuit.

- Pick N unique nonzero field elements f_{tag^1} where $\mathsf{tag}^1 \in [\mathsf{p}] \times [\Gamma]$.

- For every possible $\mathsf{tag}^3 = (i_3, b) \in [\lambda] \times \{0,1\}$,
  - Sample a degree t polynomial $\delta_{\mathsf{tag}^3}(\cdot)$ over $\mathbb{F}$ whose constant term is b.

- For every wire $w \in \mathcal{W}, b \in \{0,1\}, \mathsf{tag}^2 \in ([\mathsf{p}'] \times [\Delta])$,
  - Sample $\mathsf{sd}_{w,b}^{\mathsf{tag}^2} \xleftarrow{R} \{0,1\}^\kappa$.
  - Set $k_{w,b}^{\mathsf{tag}^2} = \mathsf{PRG}(\mathsf{sd}_{w,b}^{\mathsf{tag}^2})$.
  - Sample $r_w^{\mathsf{tag}^2} \xleftarrow{R} \{0,1\}$.
  - For $h \in [\kappa]$,
    - Sample a degree t polynomial $\zeta_{w,b,h}^{\mathsf{tag}^2}(\cdot)$ over $\mathbb{F}$ whose constant term is h^{th} bit of $\mathsf{sd}_{w,b}^{\mathsf{tag}^2}$.
  - For $h \in [4 \cdot \mathsf{p}' \cdot \kappa + 4]$,
    - Sample a degree t polynomial $\xi_{w,b,h}^{\mathsf{tag}^2}(\cdot)$ over $\mathbb{F}$ whose constant term is h^{th} bit of $k_{w,b}^{\mathsf{tag}^2}$.
  - Sample a degree t polynomial $\eta_w^{\mathsf{tag}^2}(\cdot)$ over $\mathbb{F}$ whose constant term is $r_w^{(\mathsf{tag}^2)}$.

- For $\mathsf{tag}^1 \in [\mathsf{p}] \times [\Gamma], \mathsf{h} \in [\mathsf{s}'']$, let $\mathcal{V}^{\mathsf{tag}^1,\mathsf{h}}[\{\delta_{\mathsf{tag}^3}(f_{\mathsf{tag}^1})\}_{\mathsf{tag}^3 \in [\lambda] \times \{0,1\}}]$ be the circuit that takes as inputs a circuit C of size λ and $E^\mu, E^\zeta, E^\xi, E^\eta, E^\nu$ and computes an arithmetization of the following circuit.
  - Let $C_1, \ldots C_\lambda$ be the binary representation of C.
  - Let $E_C^\delta = \{\delta_{(i,C_i)}(f_{\mathsf{tag}^1})\}_{i \in [\lambda]}$.
  - Compute $G = \mathsf{CorrGarb}(E_C^\delta, E^\mu, E^\zeta, E^\xi, E^\eta) + \sum_{e \in E^\nu} e$.
  - Output the h^{th} bit of the G.

- For $\mathsf{tag}^1 \in [\mathsf{p}] \times [\Gamma], j \in \mathsf{q}, \mathsf{h} \in [\mathsf{s}'']$
  - Sample a random degree $\mathsf{p}-1$ polynomials $\tau_{j,\mathsf{h}}(\cdot)$ with constant term as the h^{th} bit of $\mathsf{CorrGarb}(C, x, \zeta(0), \xi(0), \eta(0), \{\nu^{\mathsf{tag}^2}(0)\}_{\mathsf{tag}^2 \in \mathsf{taglist}^{2,(j)}})$.

- For every $\mathsf{tag}^2 \in ([\mathsf{p}'] \times [\Delta]) \backslash \mathcal{K}^*, \mathsf{h} \in [\mathsf{s}'']$,
  - Sample a degree $\mathsf{p}-1$ polynomial $\nu_{\mathsf{h}}^{\mathsf{tag}^2}(\cdot)$ over $\mathbb{F}$ whose constant term is 0.

- For every $\mathsf{tag}^{2'} \in \mathcal{K}^*, \mathsf{h} \in [\mathsf{s}'']$,
  - Set $\nu_{\mathsf{h}}^{\mathsf{tag}^2}(\cdot)$ to be the polynomial interpolation which satisfies $\forall \mathsf{tag}^1 \in \mathcal{I}$,

    $$\tau_{j,\mathsf{h}}(f_{\mathsf{tag}^1}) = G(\delta_C(f_{\mathsf{tag}^1}), \mu(f_{\mathsf{tag}^1}), \zeta(f_{\mathsf{tag}^1}), \xi(f_{\mathsf{tag}^1}), \eta(f_{\mathsf{tag}^1}), \{\nu^{\mathsf{tag}^2}(f_{\mathsf{tag}^1})\}_{\mathsf{tag}^2 \in \mathsf{taglist}^{2,(j)}}).$$

- For every input $i \in \ell, h \in [\log |\mathbb{F}|]$,
  - Sample a degree-t polynomial $\mu_{i,h}(\cdot)$ over $\mathbb{F}$ whose constant term is $x_{i,h}$ where x_i denotes the i^{th} coordinate of $x \in \mathbb{F}$ and is represented as bits of length $\log |F|$.
- Let $\mathsf{ct} = (\mathsf{ct}^1, \mathsf{ct}^2, \mathsf{ct}^3, \mathsf{E}^\mu, \mathsf{E}^\zeta, \mathsf{E}^\xi, \mathsf{E}^\eta, \mathsf{E}^\nu)$ be $(\emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset)$. We assume that these sets are ordered sets.
- For $\mathsf{tag}^1 \in ([\mathsf{p}] \times [\Gamma]), \mathsf{h} \in [\mathsf{s}'']$,
  - If there exists a unique j^* index for which $\mathsf{tag}^1 \in \mathsf{taglist}^{1,(j^*)}$.
    - Simulate the garbled circuit $(\tilde{\mathcal{V}}^{\mathsf{tag}^1,\mathsf{h}}, \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}) \leftarrow \mathsf{GC.Sim}(1^\kappa, 1^{|\mathsf{input}|}, 1^{|\mathcal{V}^{\mathsf{tag}^1,\mathsf{h}}|}, \tau_{j^*}(f_{\mathsf{tag}^1}))$
    - Add the simulated garbled circuit $\tilde{\mathcal{V}}^{\mathsf{tag}^1,\mathsf{h}}$ to the set ct^1.
    - For every $\mathsf{tag}^2 = (i_2, \delta) \in \mathsf{taglist}^{2,(j^*)}$
      Set $\mathsf{input}_\zeta = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\zeta_{i_2}}$
      Set $\mathsf{input}_\xi = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\xi_{i_2}}$
      Set $\mathsf{input}_\eta = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\eta_{i_2}}$
      Set $\mathsf{input}_\nu = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\nu_{i_2}}$
    - For other $\mathsf{tag}^2 = (i_2, \delta) \in [\mathsf{p}'] \times [\Gamma] \backslash \mathsf{taglist}^{2,(j^*)}$
      Set $\mathsf{input}_\zeta = 0^{|\mathsf{input}_\zeta|}$
      Set $\mathsf{input}_\xi = 0^{|\mathsf{input}_\xi|}$
      Set $\mathsf{input}_\eta = 0^{|\mathsf{input}_\eta|}$
      Set $\mathsf{input}_\nu = 0^{|\mathsf{input}_\nu|}$
    - Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_\zeta))$ to E^ζ.
    - Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_\xi))$ to E^ξ.
    - Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_\eta))$ to E^η.
    - Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_\nu))$ to E^ν.
    - For every possible $\mathsf{tag}^3 = (i, b) \in \mathsf{taglist}^{3,(j^*)}$,
      Set $\mathsf{input}_2 = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{\{C_i\}}$.
    - For other $\mathsf{tag}^3 = (i, b) \in [\lambda] \times \{0,1\} \backslash \mathsf{taglist}^{3,(j^*)}$
      Set $\mathsf{input}_2 = 0^{|\mathsf{input}_2|}$
    - Set $\mathsf{input}_\mu = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\mu}$
    - Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_2, (\mathsf{tag}^1, \mathsf{tag}^3), (\mathsf{input}_2))$ to ct^2
  - Otherwise
    - Garble the circuit i.e. $(\tilde{\mathcal{V}}^{\mathsf{tag}^1,\mathsf{h}}, \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}) \leftarrow \mathsf{GC.Garble}(1^\kappa, \mathcal{V}^{\mathsf{tag}^1,\mathsf{h}})$.
    - Add the garbled circuit $\tilde{\mathcal{V}}^{\mathsf{tag}^1,\mathsf{h}}$ to the set ct^1.
    - For every possible $\mathsf{tag}^2 = (i_2, \delta) \in ([\mathsf{p}'] \times [\Delta])$

- If $\exists j : \mathsf{tag}^1 \in \mathsf{taglist}^{1,(j)} \wedge \mathsf{tag}^2 \in \mathsf{taglist}^{2,(j)}$
  Let $\hat{E}^{\zeta}_{\mathsf{tag}^1,\mathsf{tag}^2} = \{\zeta^{\mathsf{tag}^2}_{w,b,h}(f_{\mathsf{tag}^1})\}_{w\in\mathcal{W},b\in\{0,1\},h\in[\kappa]}$.
  Set $\mathsf{input}_{\zeta} = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^{\zeta}_{i_2},\hat{E}^{\zeta}_{\mathsf{tag}^1,\mathsf{tag}^2}}$
  Let $\hat{E}^{\xi}_{\mathsf{tag}^1,\mathsf{tag}^2} = \{\xi^{\mathsf{tag}^2}_{w,b,h}(f_{\mathsf{tag}^1})\}_{w\in\mathcal{W},b\in\{0,1\},h\in[4\mathsf{p}'\kappa+4]}$.
  Set $\mathsf{input}_{\xi} = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^{\xi}_{i_2},\hat{E}^{\xi}_{\mathsf{tag}^1,\mathsf{tag}^2}}$.
  Let $\hat{E}^{\eta}_{\mathsf{tag}^1,\mathsf{tag}^2} = \{\eta^{\mathsf{tag}^2}_{w}(f_{\mathsf{tag}^1})\}_{w\in\mathcal{W}}$.
  Set $\mathsf{input}_{\eta} = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^{\eta}_{i_2},\hat{E}^{\eta}_{\mathsf{tag}^1,\mathsf{tag}^2}}$.
  Let $\hat{E}^{\nu}_{\mathsf{tag}^1,\mathsf{tag}^2,\mathsf{h}} = \{\nu^{\mathsf{tag}^2}_{\mathsf{h}}(f_{\mathsf{tag}^1})\}$.
  Set $\mathsf{input}_{\nu} = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^{\nu}_{i_2},\hat{E}^{\nu}_{\mathsf{tag}^1,\mathsf{tag}^2,\mathsf{h}}}$.
- Otherwise, if $\nexists j : \mathsf{tag}^1 \in \mathsf{taglist}^{1,(j)} \wedge \mathsf{tag}^2 \in \mathsf{taglist}^{2,(j)}$, set input_{ζ} to an appropriately length of padded zeroes.
  Set $\mathsf{input}_{\zeta} = 0^{|\mathsf{input}_{\zeta}|}$
  Set $\mathsf{input}_{\xi} = 0^{|\mathsf{input}_{\xi}|}$
  Set $\mathsf{input}_{\eta} = 0^{|\mathsf{input}_{\eta}|}$
  Set $\mathsf{input}_{\nu} = 0^{|\mathsf{input}_{\nu}|}$
- Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_{\zeta}))$ to E^{ζ}.
- Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_{\xi}))$ to E^{ξ}.
- Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_{\eta}))$ to E^{η}.
- Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_{\nu}))$ to E^{ν}.

* For every possible $\mathsf{tag}^3 = (i, b) \in ([\lambda] \times \{0, 1\})$,
  - If $\exists j : \mathsf{tag}^1 \in \mathsf{taglist}^{1,(j)} \wedge \mathsf{tag}^3 \in \mathsf{taglist}^{3,(j)}$
    Set $\mathsf{input}_2 = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{\{C_i\},\{b\}}$.
  - Otherwise, add encryptions of appropriately padded 0's
    Set $\mathsf{input}_2 = 0^{|\mathsf{input}_2|}$
  - Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_2, (\mathsf{tag}^1, \mathsf{tag}^3), (\mathsf{input}_2))$ to ct^2
* Let $\hat{E}^{\mu}_{\mathsf{tag}^1} = \{\mu_{i,h}(f_{\mathsf{tag}^1})\}_{i\in\ell,h\in[\log|\mathbb{F}|]}$.
* Let $\mathsf{input}_{\mu} = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^{\mu},\hat{E}^{\mu}_{\mathsf{tag}^1}}$.
* Add input_{μ} to the set E^{μ}.
* Add f_{tag^1} to the set ct^3

- Output $\mathsf{ct} = (\mathsf{ct}^1, \mathsf{ct}^2, \mathsf{ct}^3, \mathsf{E}^{\mu}, \mathsf{E}^{\zeta}, \mathsf{E}^{\xi}, \mathsf{E}^{\eta}, \mathsf{E}^{\nu})$.

4. **Guess:** $\mathcal{A}$ outputs a bit b.

Game 6. In this game, we simulate the correlated garbling.

1. Adversary $\mathcal{A}$ outputs circuit parameters $1^{\lambda}, 1^{\mathsf{D}}$ and collusion bound q.
2. **Setup Phase:...,Keygen query phase:...,Challenge message:....**

3. **Encryption phase:** $\mathcal{C}$ computes $\mathsf{S\text{-}FE.Enc}(\mathsf{mpk}, x = x_1 x_2 \dots x_\ell, 1^{\mathsf{q}}) \to \mathsf{ct}$. It sends this to $\mathcal{A}$.

- **Parsing:**
  - Let $|\Pi^x| = \mathsf{q}$ (wlog the cardinality is the maximum number of queries allowed).
  - Parse $\Pi^x = ((C_1, C_1(x)), \dots, (C_{\mathsf{q}}, C_{\mathsf{q}}(x)))$.
  - $\forall j \in [\mathsf{q}]$, let sk_{C_j} be the reply to query C_j to S_1.
  - Parse sk_{C_j} as $(\mathsf{sk}^{1,(j)}, \mathsf{sk}^{2,(j)}, \mathsf{taglist}^{1,(j)}, \mathsf{taglist}^{2,(j)}, \mathsf{taglist}^{3,(j)})$.
  - Parse $\mathsf{taglist}^{1,(j)} = (\mathsf{tag}_1^{1,(j)}, \dots, \mathsf{tag}_{\mathsf{p}}^{1,(j)})$.
  - Parse $\mathsf{taglist}^{2,(j)} = (\mathsf{tag}_1^{2,(j)}, \dots, \mathsf{tag}_{\mathsf{p}'}^{2,(j)})$.
  - Parse $\mathsf{taglist}^{3,(j)} = (\mathsf{tag}_1^{3,(j)}, \dots, \mathsf{tag}_{\lambda}^{3,(j)})$.
- **Computing recieved identities:**
  - Let $\mathcal{I} \subseteq ([\mathsf{p}] \times [\Gamma])$ be a set of identities which are queried on two or more keys,

$$\bigcup_{j \in \mathsf{q}, j' \in \mathsf{q}, i_1 \in [\mathsf{p}] \ \wedge \ j \neq j'} \left((\mathsf{tag}_{i_1}^{1,(j)}) \cap (\mathsf{tag}_{i_1}^{1,(j')}) \right).$$

  - If $|\mathcal{I}| > \mathsf{t}$, output $\perp$.
- **Computing cover free elements:**
  For $j \in \mathsf{q}$, let $\mathcal{K} = \mathcal{K}^* = \emptyset$. Let $\mathcal{K}^*$ be an ordered set.
  - Let $\mathcal{K}_1 = \cup_{i_2 \in [\mathsf{p}']} \mathsf{tag}_{i_2}^{2,(j)}$.
  - Pick any element $\mathsf{tag}_{i_2^*}^{2,(j)} \in \mathcal{K}_1 \setminus \mathcal{K} \in ([\mathsf{p}'] \times [\Delta])$ and add it $(\mathsf{tag}_{i_2^*}^{2,(j)})$ to $\mathcal{K}^*$ such that $\mathcal{K}_j^* = \mathsf{tag}_{i_2^*}^{2,(j)}$. If no such element exists, output $\perp$
  - Add $\mathcal{K}_1$ to set $\mathcal{K}$. i.e. $\mathcal{K} = \mathcal{K} \cup \mathcal{K}_1$.
- Let $\mathcal{U}(\cdot, \cdot)$ be a universal circuit that takes a circuit C and an input x and evaluates $C(x)$ where length of C is λ and length of x is ℓ bits. Let $\mathcal{W}$ refer to all wires of said universal circuit.
- Pick N unique nonzero field elements f_{tag^1} where $\mathsf{tag}^1 \in [\mathsf{p}] \times [\Gamma]$.
- For every possible $\mathsf{tag}^3 = (i_3, b) \in [\lambda] \times \{0, 1\}$,
  - Sample a degree t polynomial $\delta_{\mathsf{tag}^3}(\cdot)$ over $\mathbb{F}$ whose constant term is b.
- For every wire $w \in \mathcal{W}, b \in \{0, 1\}, \mathsf{tag}^2 \in ([\mathsf{p}'] \times [\Delta])$,
  - Sample $\mathsf{sd}_{w,b}^{\mathsf{tag}^2} \xleftarrow{R} \{0, 1\}^\kappa$.
  - Set $k_{w,b}^{\mathsf{tag}^2} = \mathsf{PRG}(\mathsf{sd}_{w,b}^{\mathsf{tag}^2})$.
  - Sample $r_w^{\mathsf{tag}^2} \xleftarrow{R} \{0, 1\}$.
  - For $h \in [\kappa]$,
    - Sample a degree t polynomial $\zeta_{w,b,h}^{\mathsf{tag}^2}(\cdot)$ over $\mathbb{F}$ whose constant term is h^{th} bit of $\mathsf{sd}_{w,b}^{\mathsf{tag}^2}$.
  - For $h \in [4 \cdot \mathsf{p}' \cdot \kappa + 4]$,

* Sample a degree t polynomial $\xi^{\mathsf{tag}^2}_{w,b,h}(\cdot)$ over $\mathbb{F}$ whose constant term is h^{th} bit of $k^{\mathsf{tag}^2}_{w,b}$.

– Sample a degree t polynomial $\eta^{\mathsf{tag}^2}_w(\cdot)$ over $\mathbb{F}$ whose constant term is $r^{(\mathsf{tag}^2)}_w$.

- For $\mathsf{tag}^1 \in [\mathsf{p}] \times [\Gamma], \mathsf{h} \in [\mathsf{s}'']$, let $\mathcal{V}^{\mathsf{tag}^1,\mathsf{h}}[\{\delta_{\mathsf{tag}^3}(f_{\mathsf{tag}^1})\}_{\mathsf{tag}^3 \in [\lambda]\times\{0,1\}}]$ be the circuit that takes as inputs a circuit C of size λ and $E^\mu, E^\zeta, E^\xi, E^\eta, E^\nu$ and computes an arithmetization of the following circuit.
  – Let $C_1, \ldots C_\lambda$ be the binary representation of C.
  – Let $E^\delta_C = \{\delta_{(i,C_i)}(f_{\mathsf{tag}^1})\}_{i\in[\lambda]}$.
  – Compute $G = \mathsf{CorrGarb}(E^\delta_C, E^\mu, E^\zeta, E^\xi, E^\eta) + \sum_{e\in E^\nu} e$.
  – Output the h^{th} bit of the G.
- For $\mathsf{tag}^1 \in [\mathsf{p}] \times [\Gamma], j \in \mathsf{q}, \mathsf{h} \in [\mathsf{s}'']$
  – Sample a random degree $\mathsf{p}-1$ polynomials $\tau_{j,\mathsf{h}}(\cdot)$ with constant term as the h^{th} bit of $\mathsf{Sim}_{\mathsf{CorrGarb}}(1^\kappa, C_j, C_j(x))$.
- For every $\mathsf{tag}^2 \in ([\mathsf{p}'] \times [\Delta])\backslash\mathcal{K}^*, \mathsf{h} \in [\mathsf{s}'']$,
  – Sample a degree $\mathsf{p}-1$ polynomial $\nu^{\mathsf{tag}^2}_{\mathsf{h}}(\cdot)$ over $\mathbb{F}$ whose constant term is 0.
- For every $\mathsf{tag}^{2\prime} \in \mathcal{K}^*, \mathsf{h} \in [\mathsf{s}'']$,
  – Set $\nu^{\mathsf{tag}^2}_{\mathsf{h}}(\cdot)$ to be the polynomial interpolation which satisfies $\forall \mathsf{tag}^1 \in \mathcal{I}$,

$$\tau_{j,\mathsf{h}}(f_{\mathsf{tag}^1}) = \mathsf{Sim}_{\mathsf{CorrGarb}}(1^\kappa, C_j, C_j(x)) + \sum_{\nu(\cdot)\in \mathsf{E}^\nu} \nu(f_{\mathsf{tag}^1}).$$

- For every input $i \in \ell, h \in [\log|\mathbb{F}|]$,
  – Sample a degree-t polynomial $\mu_{i,h}(\cdot)$ over $\mathbb{F}$ whose constant term is $x_{i,h}$ where x_i denotes the i^{th} coordinate of $x \in \mathbb{F}$ and is represented as bits of length $\log|F|$.
- Let $\mathsf{ct} = (\mathsf{ct}^1, \mathsf{ct}^2, \mathsf{ct}^3, \mathsf{E}^\mu, \mathsf{E}^\zeta, \mathsf{E}^\xi, \mathsf{E}^\eta, \mathsf{E}^\nu)$ be $(\emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset)$. We assume that these sets are ordered sets.
- For $\mathsf{tag}^1 \in ([\mathsf{p}] \times [\Gamma]), \mathsf{h} \in [\mathsf{s}'']$,
  – If there exists a unique j^* index for which $\mathsf{tag}^1 \in \mathsf{taglist}^{1,(j^*)}$.
    * Simulate the garbled circuit $(\tilde{\mathcal{V}}^{\mathsf{tag}^1,\mathsf{h}}, \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}) \leftarrow \mathsf{GC.Sim}(1^\kappa, 1^{|\mathsf{input}|}, 1^{|\mathcal{V}^{\mathsf{tag}^1,\mathsf{h}}|}, \tau_{j^*}(f_{\mathsf{tag}^1}))$
    * Add the simulated garbled circuit $\tilde{\mathcal{V}}^{\mathsf{tag}^1,\mathsf{h}}$ to the set ct^1.
    * For every $\mathsf{tag}^2 = (i_2, \delta) \in \mathsf{taglist}^{2,(j^*)}$
      Set $\mathsf{input}_\zeta = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\zeta_{i_2}}$
      Set $\mathsf{input}_\xi = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\xi_{i_2}}$
      Set $\mathsf{input}_\eta = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\eta_{i_2}}$
      Set $\mathsf{input}_\nu = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\nu_{i_2}}$
    * For other $\mathsf{tag}^2 = (i_2, \delta) \in [\mathsf{p}'] \times [\Gamma]\backslash\mathsf{taglist}^{2,(j^*)}$
      Set $\mathsf{input}_\zeta = 0^{|\mathsf{input}_\zeta|}$

Set $\mathsf{input}_\xi = 0^{|\mathsf{input}_\xi|}$

Set $\mathsf{input}_\eta = 0^{|\mathsf{input}_\eta|}$

Set $\mathsf{input}_\nu = 0^{|\mathsf{input}_\nu|}$

* Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_\zeta))$ to E^ζ.
* Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_\xi))$ to E^ξ.
* Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_\eta))$ to E^η.
* Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_\nu))$ to E^ν.
* For every possible $\mathsf{tag}^3 = (i, b) \in \mathsf{taglist}^{3,(j^*)}$,
  Set $\mathsf{input}_2 = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{\{C_i\}}$.
* For other $\mathsf{tag}^3 = (i, b) \in [\lambda] \times \{0,1\} \backslash \mathsf{taglist}^{3,(j^*)}$
  Set $\mathsf{input}_2 = 0^{|\mathsf{input}_2|}$
* Set $\mathsf{input}_\mu = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\mu}$
* Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_2, (\mathsf{tag}^1, \mathsf{tag}^3), (\mathsf{input}_2))$ to ct^2

– Otherwise

* Garble the circuit i.e. $(\tilde{\mathcal{V}}^{\mathsf{tag}^1,\mathsf{h}}, \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}) \leftarrow \mathsf{GC.Garble}(1^\kappa, \mathcal{V}^{\mathsf{tag}^1,\mathsf{h}})$.
* Add the garbled circuit $\tilde{\mathcal{V}}^{\mathsf{tag}^1,\mathsf{h}}$ to the set ct^1.
* For every possible $\mathsf{tag}^2 = (i_2, \delta) \in ([\mathsf{p}'] \times [\Delta])$
  · If $\exists j : \mathsf{tag}^1 \in \mathsf{taglist}^{1,(j)} \wedge \mathsf{tag}^2 \in \mathsf{taglist}^{2,(j)}$
    Let $\hat{E}^\zeta_{\mathsf{tag}^1,\mathsf{tag}^2} = \{\zeta^{\mathsf{tag}^2}_{w,b,h}(f_{\mathsf{tag}^1})\}_{w\in\mathcal{W},b\in\{0,1\},h\in[\kappa]}$.
    Set $\mathsf{input}_\zeta = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\zeta_{i_2},\hat{E}^\zeta_{\mathsf{tag}^1,\mathsf{tag}^2}}$
    Let $\hat{E}^\xi_{\mathsf{tag}^1,\mathsf{tag}^2} = \{\xi^{\mathsf{tag}^2}_{w,b,h}(f_{\mathsf{tag}^1})\}_{w\in\mathcal{W},b\in\{0,1\},h\in[4\mathsf{p}'\kappa+4]}$.
    Set $\mathsf{input}_\xi = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\xi_{i_2},\hat{E}^\xi_{\mathsf{tag}^1,\mathsf{tag}^2}}$.
    Let $\hat{E}^\eta_{\mathsf{tag}^1,\mathsf{tag}^2} = \{\eta^{\mathsf{tag}^2}_w(f_{\mathsf{tag}^1})\}_{w\in\mathcal{W}}$.
    Set $\mathsf{input}_\eta = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\eta_{i_2},\hat{E}^\eta_{\mathsf{tag}^1,\mathsf{tag}^2}}$.
    Let $\hat{E}^\nu_{\mathsf{tag}^1,\mathsf{tag}^2,\mathsf{h}} = \{\nu^{\mathsf{tag}^2}_{\mathsf{h}}(f_{\mathsf{tag}^1})\}$.
    Set $\mathsf{input}_\nu = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\nu_{i_2},\hat{E}^\nu_{\mathsf{tag}^1,\mathsf{tag}^2,\mathsf{h}}}$.
  · Otherwise, if $\not\exists j : \mathsf{tag}^1 \in \mathsf{taglist}^{1,(j)} \wedge \mathsf{tag}^2 \in \mathsf{taglist}^{2,(j)}$, set input_ζ to an appropriately length of padded zeroes.
    Set $\mathsf{input}_\zeta = 0^{|\mathsf{input}_\zeta|}$
    Set $\mathsf{input}_\xi = 0^{|\mathsf{input}_\xi|}$
    Set $\mathsf{input}_\eta = 0^{|\mathsf{input}_\eta|}$
    Set $\mathsf{input}_\nu = 0^{|\mathsf{input}_\nu|}$
  · Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_\zeta))$ to E^ζ.
  · Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_\xi))$ to E^ξ.
  · Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_\eta))$ to E^η.
  · Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_\nu))$ to E^ν.
* For every possible $\mathsf{tag}^3 = (i, b) \in ([\lambda] \times \{0,1\})$,

- If $\exists j : \mathsf{tag}^1 \in \mathsf{taglist}^{1,(j)} \wedge \mathsf{tag}^3 \in \mathsf{taglist}^{3,(j)}$
  Set $\mathsf{input}_2 = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{\{C_i\},\{b\}}$.
- Otherwise, add encryptions of appropriately padded 0's
  Set $\mathsf{input}_2 = 0^{|\mathsf{input}_2|}$
- Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_2, (\mathsf{tag}^1, \mathsf{tag}^3), (\mathsf{input}_2))$ to ct^2

* Let $\hat{E}^{\mu}_{\mathsf{tag}^1} = \{\mu_{i,h}(f_{\mathsf{tag}^1})\}_{i\in\ell, h\in[\log|\mathbb{F}|]}$.
* Let $\mathsf{input}_\mu = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\mu,\hat{E}^{\mu}_{\mathsf{tag}^1}}$.
* Add input_μ to the set E^μ.
* Add f_{tag^1} to the set ct^3

- Output $\mathsf{ct} = (\mathsf{ct}^1, \mathsf{ct}^2, \mathsf{ct}^3, \mathsf{E}^\mu, \mathsf{E}^\zeta, \mathsf{E}^\xi, \mathsf{E}^\eta, \mathsf{E}^\nu)$.

4. **Guess:** $\mathcal{A}$ outputs a bit b.

Game 7. In this game, we erase the final input information encoded in the input polynomials, allowing the circuit to be simulated.

1. Adversary $\mathcal{A}$ outputs circuit parameters $1^\lambda, 1^{\mathsf{D}}$ and collusion bound q.
2. **Setup Phase:. . . ,Keygen query phase:. . . ,Challenge message:. . . .**
3. **Encryption phase:** $\mathcal{C}$ computes $\mathsf{S\text{-}FE.Enc}(\mathsf{mpk}, x = x_1x_2\ldots x_\ell, 1^{\mathsf{q}}) \to \mathsf{ct}$. It sends this to $\mathcal{A}$.

- **Parsing:**
  - Let $|\Pi^x| = \mathsf{q}$ (wlog the cardinality is the maximum number of queries allowed).
  - Parse $\Pi^x = ((C_1, C_1(x)), \ldots, (C_{\mathsf{q}}, C_{\mathsf{q}}(x)))$.
  - $\forall j \in [\mathsf{q}]$, let sk_{C_j} be the reply to query C_j to S_1.
  - Parse sk_{C_j} as $(\mathsf{sk}^{1,(j)}, \mathsf{sk}^{2,(j)}, \mathsf{taglist}^{1,(j)}, \mathsf{taglist}^{2,(j)}, \mathsf{taglist}^{3,(j)})$.
  - Parse $\mathsf{taglist}^{1,(j)} = (\mathsf{tag}^{1,(j)}_1, \ldots, \mathsf{tag}^{1,(j)}_{\mathsf{p}})$.
  - Parse $\mathsf{taglist}^{2,(j)} = (\mathsf{tag}^{2,(j)}_1, \ldots, \mathsf{tag}^{2,(j)}_{\mathsf{p}'})$.
  - Parse $\mathsf{taglist}^{3,(j)} = (\mathsf{tag}^{3,(j)}_1, \ldots, \mathsf{tag}^{3,(j)}_{\lambda})$.
- **Computing recieved identities:**
  - Let $\mathcal{I} \subseteq ([\mathsf{p}] \times [\Gamma])$ be a set of identities which are queried on two or more keys,
  $$\bigcup_{j\in\mathsf{q}, j'\in\mathsf{q}, i_1\in[\mathsf{p}] \ \wedge \ j\neq j'} \left((\mathsf{tag}^{1,(j)}_{i_1}) \cap (\mathsf{tag}^{1,(j')}_{i_1})\right).$$
  - If $|\mathcal{I}| > \mathsf{t}$, output $\perp$.
- **Computing cover free elements:**
  For $j \in \mathsf{q}$, let $\mathcal{K} = \mathcal{K}^* = \emptyset$. Let $\mathcal{K}^*$ be an ordered set.
  - Let $\mathcal{K}_1 = \cup_{i_2\in[\mathsf{p}']}\mathsf{tag}^{2,(j)}_{i_2}$.
  - Pick any element $\mathsf{tag}^{2,(j)}_{i_2^*} \in \mathcal{K}_1 \setminus \mathcal{K} \in ([\mathsf{p}'] \times [\Delta])$ and add it $(\mathsf{tag}^{2,(j)}_{i_2^*})$ to $\mathcal{K}^*$ such that $\mathcal{K}^*_j = \mathsf{tag}^{2,(j)}_{i_2^*}$. If no such element exists, output $\perp$

- Add $\mathcal{K}_1$ to set $\mathcal{K}$. i.e. $\mathcal{K} = \mathcal{K} \cup \mathcal{K}_1$.

- Let $\mathcal{U}(\cdot,\cdot)$ be a universal circuit that takes a circuit C and an input x and evaluates $C(x)$ where length of C is λ and length of x is ℓ bits. Let $\mathcal{W}$ refer to all wires of said universal circuit.
- Pick N unique nonzero field elements f_{tag^1} where $\mathsf{tag}^1 \in [\mathsf{p}] \times [\Gamma]$.
- For every possible $\mathsf{tag}^3 = (i_3, b) \in [\lambda] \times \{0,1\}$,
  - Sample a degree t polynomial $\delta_{\mathsf{tag}^3}(\cdot)$ over $\mathbb{F}$ whose constant term is b.
- For every wire $w \in \mathcal{W}, b \in \{0,1\}, \mathsf{tag}^2 \in ([\mathsf{p}'] \times [\Delta])$,
  - Sample $\mathsf{sd}_{w,b}^{\mathsf{tag}^2} \xleftarrow{R} \{0,1\}^\kappa$.
  - Set $k_{w,b}^{\mathsf{tag}^2} = \mathsf{PRG}(\mathsf{sd}_{w,b}^{\mathsf{tag}^2})$.
  - Sample $r_w^{\mathsf{tag}^2} \xleftarrow{R} \{0,1\}$.
  - For $h \in [\kappa]$,
    - Sample a degree t polynomial $\zeta_{w,b,h}^{\mathsf{tag}^2}(\cdot)$ over $\mathbb{F}$ whose constant term is h^{th} bit of $\mathsf{sd}_{w,b}^{\mathsf{tag}^2}$.
  - For $h \in [4 \cdot \mathsf{p}' \cdot \kappa + 4]$,
    - Sample a degree t polynomial $\xi_{w,b,h}^{\mathsf{tag}^2}(\cdot)$ over $\mathbb{F}$ whose constant term is h^{th} bit of $k_{w,b}^{\mathsf{tag}^2}$.
  - Sample a degree t polynomial $\eta_w^{\mathsf{tag}^2}(\cdot)$ over $\mathbb{F}$ whose constant term is $r_w^{(\mathsf{tag}^2)}$.
- For $\mathsf{tag}^1 \in [\mathsf{p}] \times [\Gamma], \mathsf{h} \in [\mathsf{s}'']$, let $\mathcal{V}^{\mathsf{tag}^1,\mathsf{h}}[\{\delta_{\mathsf{tag}^3}(f_{\mathsf{tag}^1})\}_{\mathsf{tag}^3 \in [\lambda]\times\{0,1\}}]$ be the circuit that takes as inputs a circuit C of size λ and $E^\mu, E^\zeta, E^\xi, E^\eta, E^\nu$ and computes an arithmetization of the following circuit.
  - Let $C_1, \ldots C_\lambda$ be the binary representation of C.
  - Let $E_C^\delta = \{\delta_{(i,C_i)}(f_{\mathsf{tag}^1})\}_{i\in[\lambda]}$.
  - Compute $G = \mathsf{CorrGarb}(E_C^\delta, E^\mu, E^\zeta, E^\xi, E^\eta) + \sum_{e\in E^\nu} e$.
  - Output the h^{th} bit of the G.
- For $\mathsf{tag}^1 \in [\mathsf{p}] \times [\Gamma], j \in \mathsf{q}, \mathsf{h} \in [\mathsf{s}'']$
  - Sample a random degree $\mathsf{p}-1$ polynomials $\tau_{j,\mathsf{h}}(\cdot)$ with constant term as the h^{th} bit of $\mathsf{Sim}_{\mathsf{CorrGarb}}(1^\kappa, C_j, C_j(x))$.
- For every $\mathsf{tag}^2 \in ([\mathsf{p}'] \times [\Delta]) \backslash \mathcal{K}^*, \mathsf{h} \in [\mathsf{s}'']$,
  - Sample a degree $\mathsf{p}-1$ polynomial $\nu_{\mathsf{h}}^{\mathsf{tag}^2}(\cdot)$ over $\mathbb{F}$ whose constant term is 0.
- For every $\mathsf{tag}^{2\prime} \in \mathcal{K}^*, \mathsf{h} \in [\mathsf{s}'']$,
  - Set $\nu_{\mathsf{h}}^{\mathsf{tag}^2}(\cdot)$ to be the polynomial interpolation which satisfies $\forall \mathsf{tag}^1 \in \mathcal{I}$,

$$\tau_{j,\mathsf{h}}(f_{\mathsf{tag}^1}) = \mathsf{Sim}_{\mathsf{CorrGarb}}(1^\kappa, C_j, C_j(x)).$$

- For every input $i \in \ell, h \in [\log|\mathbb{F}|]$,
  - Sample a degree-t polynomial $\mu_{i,h}(\cdot)$ over $\mathbb{F}$ whose constant term is 0.

- Let $\mathsf{ct} = (\mathsf{ct}^1, \mathsf{ct}^2, \mathsf{ct}^3, \mathsf{E}^\mu, \mathsf{E}^\zeta, \mathsf{E}^\xi, \mathsf{E}^\eta, \mathsf{E}^\nu)$ be $(\emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset)$. We assume that these sets are ordered sets.
- For $\mathsf{tag}^1 \in ([\mathsf{p}] \times [\Gamma]), \mathsf{h} \in [\mathsf{s}'']$,
  - If there exists a unique j^* index for which $\mathsf{tag}^1 \in \mathsf{taglist}^{1,(j^*)}$.
    - Simulate the garbled circuit $(\tilde{\mathcal{V}}^{\mathsf{tag}^1,\mathsf{h}}, \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}) \leftarrow \mathsf{GC.Sim}(1^\kappa, 1^{|\mathsf{input}|}, 1^{|\mathcal{V}^{\mathsf{tag}^1,\mathsf{h}}|}, \tau_{j^*}(f_{\mathsf{tag}^1}))$
    - Add the simulated garbled circuit $\tilde{\mathcal{V}}^{\mathsf{tag}^1,\mathsf{h}}$ to the set ct^1.
    - For every $\mathsf{tag}^2 = (i_2, \delta) \in \mathsf{taglist}^{2,(j^*)}$
      Set $\mathsf{input}_\zeta = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\zeta_{i_2}}$
      Set $\mathsf{input}_\xi = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\xi_{i_2}}$
      Set $\mathsf{input}_\eta = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\eta_{i_2}}$
      Set $\mathsf{input}_\nu = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\nu_{i_2}}$
    - For other $\mathsf{tag}^2 = (i_2, \delta) \in [\mathsf{p}'] \times [\Gamma] \backslash \mathsf{taglist}^{2,(j^*)}$
      Set $\mathsf{input}_\zeta = 0^{|\mathsf{input}_\zeta|}$
      Set $\mathsf{input}_\xi = 0^{|\mathsf{input}_\xi|}$
      Set $\mathsf{input}_\eta = 0^{|\mathsf{input}_\eta|}$
      Set $\mathsf{input}_\nu = 0^{|\mathsf{input}_\nu|}$
    - Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_\zeta))$ to E^ζ.
    - Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_\xi))$ to E^ξ.
    - Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_\eta))$ to E^η.
    - Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_\nu))$ to E^ν.
    - For every possible $\mathsf{tag}^3 = (i, b) \in \mathsf{taglist}^{3,(j^*)}$,
      Set $\mathsf{input}_2 = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{\{C_i\}}$.
    - For other $\mathsf{tag}^3 = (i, b) \in [\lambda] \times \{0,1\} \backslash \mathsf{taglist}^{3,(j^*)}$
      Set $\mathsf{input}_2 = 0^{|\mathsf{input}_2|}$
    - Set $\mathsf{input}_\mu = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\mu}$
    - Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_2, (\mathsf{tag}^1, \mathsf{tag}^3), (\mathsf{input}_2))$ to ct^2
  - Otherwise
    - Garble the circuit i.e. $(\tilde{\mathcal{V}}^{\mathsf{tag}^1,\mathsf{h}}, \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}) \leftarrow \mathsf{GC.Garble}(1^\kappa, \mathcal{V}^{\mathsf{tag}^1,\mathsf{h}})$.
    - Add the garbled circuit $\tilde{\mathcal{V}}^{\mathsf{tag}^1,\mathsf{h}}$ to the set ct^1.
    - For every possible $\mathsf{tag}^2 = (i_2, \delta) \in ([\mathsf{p}'] \times [\Delta])$
      - If $\exists j : \mathsf{tag}^1 \in \mathsf{taglist}^{1,(j)} \wedge \mathsf{tag}^2 \in \mathsf{taglist}^{2,(j)}$
        Let $\hat{E}^\zeta_{\mathsf{tag}^1,\mathsf{tag}^2} = \{\zeta^{\mathsf{tag}^2}_{w,b,h}(f_{\mathsf{tag}^1})\}_{w \in \mathcal{W}, b \in \{0,1\}, h \in [\kappa]}$.
        Set $\mathsf{input}_\zeta = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\zeta_{i_2}, \hat{E}^\zeta_{\mathsf{tag}^1,\mathsf{tag}^2}}$
        Let $\hat{E}^\xi_{\mathsf{tag}^1,\mathsf{tag}^2} = \{\xi^{\mathsf{tag}^2}_{w,b,h}(f_{\mathsf{tag}^1})\}_{w \in \mathcal{W}, b \in \{0,1\}, h \in [4\mathsf{p}'\kappa+4]}$.
        Set $\mathsf{input}_\xi = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\xi_{i_2}, \hat{E}^\xi_{\mathsf{tag}^1,\mathsf{tag}^2}}$.
        Let $\hat{E}^\eta_{\mathsf{tag}^1,\mathsf{tag}^2} = \{\eta^{\mathsf{tag}^2}_w(f_{\mathsf{tag}^1})\}_{w \in \mathcal{W}}$.

Set $\mathsf{input}_\eta = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\eta_{i_2},\hat{E}^\eta_{\mathsf{tag}^1,\mathsf{tag}^2}}$.

Let $\hat{E}^\nu_{\mathsf{tag}^1,\mathsf{tag}^2,\mathsf{h}} = \{\nu^{\mathsf{tag}^2}_{\mathsf{h}}(f_{\mathsf{tag}^1})\}$.

Set $\mathsf{input}_\nu = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\nu_{i_2},\hat{E}^\nu_{\mathsf{tag}^1,\mathsf{tag}^2,\mathsf{h}}}$.

· Otherwise, if $\nexists j : \mathsf{tag}^1 \in \mathsf{taglist}^{1,(j)} \wedge \mathsf{tag}^2 \in \mathsf{taglist}^{2,(j)}$, set input_ζ to an appropriately length of padded zeroes.

Set $\mathsf{input}_\zeta = 0^{|\mathsf{input}_\zeta|}$

Set $\mathsf{input}_\xi = 0^{|\mathsf{input}_\xi|}$

Set $\mathsf{input}_\eta = 0^{|\mathsf{input}_\eta|}$

Set $\mathsf{input}_\nu = 0^{|\mathsf{input}_\nu|}$

· Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_\zeta))$ to E^ζ.

· Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_\xi))$ to E^ξ.

· Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_\eta))$ to E^η.

· Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_1, (\mathsf{tag}^1, \mathsf{tag}^2), (\mathsf{input}_\nu))$ to E^ν.

∗ For every possible $\mathsf{tag}^3 = (i, b) \in ([\lambda] \times \{0, 1\})$,

· If $\exists j : \mathsf{tag}^1 \in \mathsf{taglist}^{1,(j)} \wedge \mathsf{tag}^3 \in \mathsf{taglist}^{3,(j)}$

Set $\mathsf{input}_2 = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{\{C_i\},\{b\}}$.

· Otherwise, add encryptions of appropriately padded 0's

Set $\mathsf{input}_2 = 0^{|\mathsf{input}_2|}$

· Add $\mathsf{IBE.Enc}(\mathsf{IBE.pk}_2, (\mathsf{tag}^1, \mathsf{tag}^3), (\mathsf{input}_2))$ to ct^2

∗ Let $\hat{E}^\mu_{\mathsf{tag}^1} = \{\mu_{i,h}(f_{\mathsf{tag}^1})\}_{i\in\ell,h\in[\log|\mathbb{F}|]}$.

∗ Let $\mathsf{input}_\mu = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^\mu,\hat{E}^\mu_{\mathsf{tag}^1}}$.

∗ Add input_μ to the set E^μ.

∗ Add f_{tag^1} to the set ct^3

- Output $\mathsf{ct} = (\mathsf{ct}^1, \mathsf{ct}^2, \mathsf{ct}^3, \mathsf{E}^\mu, \mathsf{E}^\zeta, \mathsf{E}^\xi, \mathsf{E}^\eta, \mathsf{E}^\nu)$.

4. **Guess:** $\mathcal{A}$ outputs a bit b.

B.1 Analysis.

Next, we show by a sequence of lemmas that no adversary can distinguish between any two adjacent games with non-negligible advantage. In the last game, we show that the advantage of any such adversary is negligible. We will let $\mathsf{adv}^x_{\mathcal{A}}$ denote the probability that $\mathcal{A}$ outputs bit 1 in Game x.

Lemma B.2. $\forall\kappa \in \mathbb{N}$, $|\mathsf{adv}^0_{\mathcal{A}} - \mathsf{adv}^1_{\mathcal{A}}| = 0$.

Proof. This game simply introduces new syntactic notation and does not affect the execution of the game □

Lemma B.3. $\forall\kappa \in \mathbb{N}$, $|\mathsf{adv}^1_{\mathcal{A}} - \mathsf{adv}^2_{\mathcal{A}}| = \mathsf{negl}(\kappa)$.

Proof. In Game 2, we add two abort conditions when $\mathsf{taglist}^1$ is does not satisfy the small set intersection property or when $\mathsf{taglist}^2$ is not cover free. By Lemma A.2 and Lemma A.1, these both occur with negligible probability. □

Lemma B.4. $\forall \kappa \in \mathbb{N}$, $|\mathsf{adv}^2_{\mathcal{A}} - \mathsf{adv}^3_{\mathcal{A}}| = 0$.

Proof. Note in Game 2, the distribution of $\tau_{j,\mathsf{h}}$ is a uniformly random polynomial with the specified constant term by the randomness of $\nu(\cdot)$. Since the distribution of $\{\nu^{\mathsf{tag}^2}(\cdot)\}_{\mathsf{tag}^2 \in \mathsf{taglist}^2 \setminus \mathsf{tag}^{2\prime}}, \tau_{j,\mathsf{h}}$ is identical, and these fix the value of $\nu^{\mathsf{tag}^{2\prime}}(\cdot)$, the distribution of $\nu^{\mathsf{tag}^2}(\cdot)$ given to $\mathcal{A}$ is identical. □

Lemma B.5. $\forall \kappa \in \mathbb{N}$, $|\mathsf{adv}^3_{\mathcal{A}} - \mathsf{adv}^4_{\mathcal{A}}| = \mathsf{negl}(\kappa)$.

Proof. If there does not exist $j : \mathsf{tag}^1 \in \mathsf{taglist}^{1,(j)} \wedge \mathsf{tag}^2 \in \mathsf{taglist}^{2,(j)}$, then we know that by construction of S-FE.KeyGen, that IBE keygen is never given out for identity $(\mathsf{tag}^1, \mathsf{tag}^2)$ for $\mathsf{IBE.pk}_1$. Similarly, if there does not exist $j : \mathsf{tag}^1 \in \mathsf{taglist}^{1,(j)} \wedge \mathsf{tag}^3 \in \mathsf{taglist}^{3,(j)}$, then we know that by construction of S-FE.KeyGen, that IBE keygen is never given out for identity $(\mathsf{tag}^1, \mathsf{tag}^3)$ for $\mathsf{IBE.pk}_2$. Thus, we can provide a series of (polynomially many) subhybrids incrementally replacing each ciphertext with 0, where we can argue the indistinguishability of said subhybrids by reducing to IBE security.

Let $Ł = \mathsf{E}^{\zeta} \circ \mathsf{E}^{\xi} \circ \mathsf{E}^{\eta} \circ \mathsf{E}^{\nu} \circ \mathsf{ct}^2$ be a list of all IBE ciphertexts output by Game 3. For $a \in |Ł|$, we define Game 3^a to replace any ciphertext in the first a elements of Ł to under identity $(\mathsf{tag}^1, \mathsf{tag}^2) :/ \exists j\ \mathsf{tag}^1 \in \mathsf{taglist}^{1,(j)} \wedge \mathsf{tag}^2 \in \mathsf{taglist}^{2,(j)}$ or $(\mathsf{tag}^1, \mathsf{tag}^3) : \nexists j\ \mathsf{tag}^1 \in \mathsf{taglist}^{1,(j)} \wedge \mathsf{tag}^3 \in \mathsf{taglist}^{3,(j)}$ with an equal length encryption of 0 under the same ciphertext.

Claim B.1. $\forall \kappa \in \mathbb{N}, a \in [|Ł|]$, $|\mathsf{adv}^{3^{a-1}}_{\mathcal{A}} - \mathsf{adv}^{3^a}_{\mathcal{A}}| \leq \mathsf{negl}(\kappa)$

Proof. Note the only difference between Games 3^{a-1} and 3^a is the way ciphertext $Ł_a$ is constructed. We can break this down into two cases - for the tag which $Ł_a$ is encrypted under, either it is a tag $(\mathsf{tag}^1, \mathsf{tag}^2)$ for which $\exists j : \mathsf{tag}^1 \in \mathsf{taglist}^{1,(j)} \wedge \mathsf{tag}^2 \in \mathsf{taglist}^{2,(j)}$ or there is no such j (an analogous argument holds for ciphertexts encrypted under tags in space $(\mathsf{tag}^1, \mathsf{tag}^3)$, but for notational simplicity, we will refer to the former case). In the first case, the two games are identical, so we are done. In the second case, we can rely on IBE security.

Suppose $\mathcal{A}$ is a distinguisher between games 3^{a-1} and 3^a. Then we can construct an adversay $\mathcal{A}'$ which breaks the IBE security game as follows:

$\mathcal{A}'$

1. Run $\mathcal{A}$, receive circuit parameters
2. Receive mpk on identity space $([\mathsf{p}] \times [\Gamma]) \times ([\mathsf{p}'] \times [\Delta])$ from challenger and set as $\mathsf{IBE.pk}_1$
3. Generate $(\mathsf{IBE.pk}_2, \mathsf{IBE.sk}_2) \leftarrow \mathsf{IBE.Setup}(1^\kappa, ([\mathsf{p}] \times [\Gamma]) \times ([\lambda] \times \{0,1\})) \rightarrow$.
4. Pass $(\mathsf{IBE.pk}_1, \mathsf{IBE.pk}_2)$ to $\mathcal{A}$.
5. Receive keygen queries from $\mathcal{A}$, and run S-FE to generate any of required secret keys. Query the IBE.KeyGen oracle for any secret keys of $\mathsf{IBE.pk}_1 = \mathsf{mpk}$ required.
6. Run Game 3^{a-1}'s Encryption Phase.
   - When encrypting $Ł_a$, let m be the message to be encrypted under $(\mathsf{tag}^1, \mathsf{tag}^2)$. Set $m_0 = m$ and $m_1 = 0$ and receive IBE challenge ciphertext ct. Set $Ł_a = \mathsf{ct}$.
7. Receive bit b from $\mathcal{A}$. Return b.

Since the only difference between Game 3^{a-1} and 3^a on Ł_a, we can see the execution of $\mathcal{A}'$ above corresponds exactly to 3^{a-1} when $b = 0$ and 3^a when $b = 1$. Since there is not $j : \mathsf{tag}^1 \in \mathsf{taglist}^{1,(j)} \wedge \mathsf{tag}^2 \in \mathsf{taglist}^{2,(j)}$, IBE.KeyGen never needs to request on $(\mathsf{tag}^1, \mathsf{tag}^2)$. □

Since Ł is of polynomial size, the difference between Games $3 = 3^0$ and $3^{|\text{Ł}|} = 4$ is negligible as well.

□

Lemma B.6. $\forall \kappa \in \mathbb{N}$, $|\mathsf{adv}^4_{\mathcal{A}} - \mathsf{adv}^5_{\mathcal{A}}| = \mathsf{negl}(\kappa)$.

Proof. Much like the previous games, we can provide a series of subhybrids indexed by $\mathsf{tag}^1 \in ([\mathsf{p}] \times [\Gamma])$ which are indistinguishable this time by a reduction to the simulation security of garbled circuits.

Define some list $\mathsf{tag}^1{}_1, \ldots \mathsf{tag}^1{}_{\mathsf{p}\cdot\Gamma}$ of all elements in $([\mathsf{p}] \times [\Gamma])$. Define Game 4^a to be the game for which simulate garbling (run the red text in Game 5) only for tags $\mathsf{tag}^1{}_1, \ldots, \mathsf{tag}^1{}_a$.

Claim B.2. $\forall \kappa \in \mathbb{N}\ \forall a \in [\mathsf{p} \cdot \Gamma]$, $|\mathsf{adv}^{4^{a-1}}_{\mathcal{A}} - \mathsf{adv}^{4^a}_{\mathcal{A}}| = \mathsf{negl}(\kappa)$.

Proof. Observe that these games only differ on $\mathsf{tag}^1{}_a$. We can split this into two cases - either there does not exists a unique j^* for which $\mathsf{tag}^1{}_a \in \mathsf{taglist}^{1,(j^*)}$, in which case these games are identical, or there does exist such a unique $\mathsf{taglist}^{1,(j^*)}$. In the latter case, suppose $\mathcal{A}$ is a distinguisher between games 4^{a-1} and 4^a. Consider the following distinguisher $\mathcal{A}'$ for garbled circuits.

$\mathcal{A}'$

1. Run $\mathcal{A}$ and receive circuit parameters.
2. Run **Setup Phase**, **Keygen Phase**, **Challenge Phase** as the challenger using S-FE.Setup, S-FE.KeyGen.
3. Run Game 4^{a-1}'s Encryption Phase.
   - At $\mathsf{tag}^1{}_a$, request a sample $(\mathsf{GC}, \mathcal{L})$ from the garbled circuit security game for a circuit of size $|\mathcal{V}^{\mathsf{tag}^1,\mathsf{h}}|$ evaluated at $E^{\delta}_C(\mathsf{tag}^1{}_a), E^{\mu}(\mathsf{tag}^1{}_a), E^{\zeta}(\mathsf{tag}^1{}_a), E^{\xi}(\mathsf{tag}^1{}_a), E^{\eta}(\mathsf{tag}^1{}_a), E^{\nu}(\mathsf{tag}^1{}_a)$. For all $\mathsf{tag}^2 \in \mathsf{taglist}^{2,(j^*)}$,
     - Set $\mathsf{input}_{\zeta} = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^{\zeta}_{i_2}}$
     - Set $\mathsf{input}_{\xi} = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^{\xi}_{i_2}}$
     - Set $\mathsf{input}_{\eta} = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^{\eta}_{i_2}}$
     - Set $\mathsf{input}_{\nu} = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E^{\nu}_{i_2}}$

     Similarly, for all $\mathsf{tag}^3 \in \mathsf{taglist}^{3,(j^*)}$,
     - Set $\mathsf{input}_2 = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{\{C_i\}}$

     In addition, for all other $\mathsf{tag}^2, \mathsf{tag}^3$ not in the above lists, set the above mentioned values to appropriately padded 0. Finally, set $\mathsf{input}_{\mu} = \mathcal{L}^{\mathsf{tag}^1,\mathsf{h}}_{E_{\mu}}$ before proceeding with execution of Game 4^{a-1}.

4. Receive bit b from $\mathcal{A}$. Return b.

We can observe here that the distribution induced by the above is Game 4^{a-1} when $(\mathsf{GC}, \mathcal{L})$ is produced by $\mathsf{GC.Garble}$ and is Game 4^a when produced by $\mathsf{GC.Sim}$. We can verify that the latter is true as we set $\tau_{j^*}(f_{\mathsf{tag}^1}) = \mathcal{V}^{\mathsf{tag}^1,\mathsf{h}}(E_C^{\delta}(\mathsf{tag}^1{}_a), E^{\mu}(\mathsf{tag}^1{}_a), E^{\zeta}(\mathsf{tag}^1{}_a), E^{\xi}(\mathsf{tag}^1{}_a), E^{\eta}(\mathsf{tag}^1{}_a), E^{\nu}(\mathsf{tag}^1{}_a))$, while the former is true as we can see when $\mathsf{tag}^2 \notin \mathsf{taglist}^{2,(j^*)}$, since j^* is the *unique* list for which $\mathsf{tag}^1 \in \mathsf{taglist}^{1,(j^*)}$, we know that there cannot exist $j : \mathsf{tag}^1 \in \mathsf{taglist}^{1,(j)} \wedge \mathsf{tag}^2 \in \mathsf{taglist}^{2,(j)}$, which means that Game 4^{a-1} sets the inputs under those corresponding tags to 0. Since these two distributions are computationally indistinguishable by the security of garbled circuits, $\mathcal{A}'$ and thus $\mathcal{A}$ must have negligible advantage.

□

Since $\mathsf{p} \cdot \Gamma$ is polynomial, the difference between Games $4 = 4^0$ and $4^{\mathsf{p}\cdot\Gamma} = 5$ is negligible as well.

□

Lemma B.7. $\forall \kappa \in \mathbb{N}$, $|\mathsf{adv}^5_{\mathcal{A}} - \mathsf{adv}^6_{\mathcal{A}}| = \mathsf{negl}(\kappa)$.

Proof. The only difference between these games is the use of $\mathsf{Sim}_{\mathsf{CorrGarb}}$ rather than $\mathsf{CorrGarb}$ in generating $\tau_{j,\mathsf{h}}$. Let $\mathcal{A}$ be a distinguisher between Games 5 and 6. Then consider the following reduction $\mathcal{A}'$ against the security game of correlated garbling.

$\mathcal{A}'$

1. Run $\mathcal{A}$ and receive circuit parameters.
2. Run **Setup Phase**, **Keygen Phase**, **Challenge Phase** as the challenger using $\mathsf{S\text{-}FE.Setup}, \mathsf{S\text{-}FE.KeyGen}$.
3. Run Game 5's Encryption Phase.
   - Submit the input x and circuits $C_1, \ldots, C_{\mathsf{q}}$ received in Keygen phase to your challenger.
   - Receive encodings $\{\mathsf{GC}_i, K_i\}_{i\in[\mathsf{q}]}$
   - Set $\tau_{j,\mathsf{h}}(\cdot)$ to be a random polynomial with constant term equal to the h^{th} bit of (GC_j, K_j).
   - Set $\nu_{j,\mathsf{h}}(\cdot)$ to be the polynomial which satisfies

$$\tau_{j,\mathsf{h}}(f_{\mathsf{tag}^1}) = (\mathsf{GC}_j, K_j)_{\mathsf{h}} + \sum_{\nu(\cdot)\in\mathsf{E}^{\nu}} \nu(f_{\mathsf{tag}^1}).$$

4. Receive bit b from $\mathcal{A}$. Return b.

□

Lemma B.8. $\forall \kappa \in \mathbb{N}$, $|\mathsf{adv}^6_{\mathcal{A}} - \mathsf{adv}^7_{\mathcal{A}}| = 0$.

Lemma B.9. Observe that in game 6, μ is only evaluated on points where GC is not simulated - i.e. when $\mathsf{tag}^1 \in \mathcal{I}$. Since there are $< \mathsf{p}'$ such points, the evaluation of a random polynomial with constant terms x_i and 0 are information-theoretically identical.