Non-Interactive Anonymous Router

Elaine Shi
CMU
runting@cs.cmu.edu

Ke Wu
CMU
kew2@andrew.cmu.edu

April 2, 2021

Abstract

Anonymous routing is one of the most fundamental online privacy problems and has been studied extensively for decades. Almost all known approaches for anonymous routing (e.g., mix-nets, DC-nets, and others) rely on multiple servers or routers to engage in some *interactive* protocol; and anonymity is guaranteed in the *threshold* model, i.e., if one or more of the servers/routers behave honestly.

Departing from all prior approaches, we propose a novel *non-interactive* abstraction called a Non-Interactive Anonymous Router (NIAR), which works even with a *single untrusted router*. In a NIAR scheme, suppose that n senders each want to talk to a distinct receiver. A one-time trusted setup is performed such that each sender obtains a sending key, each receiver obtains a receiving key, and the router receives a *token* that "encrypts" the permutation mapping the senders to receivers. In every time step, each sender can encrypt its message using its sender key, and the router can use its token to convert the n ciphertexts received from the senders to n *transformed ciphertexts*. Each transformed ciphertext is delivered to the corresponding receiver, and the receiver can decrypt the message using its receiver key. Imprecisely speaking, security requires that the untrusted router, even when colluding with a subset of corrupt senders and/or receivers, should not be able to compromise the privacy of honest parties, including who is talking to who, and the message contents.

We show how to construct a communication-efficient NIAR scheme with provable security guarantees based on the standard Decisional Linear assumption in suitable bilinear groups. We show that a compelling application of NIAR is to realize a Non-Interactive Anonymous Shuffler (NIAS), where an untrusted server or data analyst can only decrypt a permuted version of the messages coming from n senders where the permutation is hidden. NIAS can be adopted to construct privacy-preserving surveys, differentially private protocols in the shuffle model, and pseudonymous bulletin boards.

Besides this main result, we also describe a variant that achieves fault tolerance when a subset of the senders may crash. Finally, we further explore a paranoid notion of security called full insider protection, and show that if we additionally assume sub-exponentially secure Indistinguishability Obfuscation and as sub-exponentially secure one-way functions, one can construct a NIAR scheme with paranoid security.

Contents

1 Introduction

The Internet has become a platform that billions of users rely on in their daily lives, and protecting users' online privacy is a significant challenge we face. Anonymous communication systems provide a way for users to communicate without leaking their identities or message contents. There has been several decades of work dedicated to the design, implementation, and deployment of anonymous communication systems [Cha81, Abe99, BG12, Cha88, CGF10, DMS04, GRS99, CBM15, ZZZR05, vdHLZZ15, TGL+17, SBS02], and numerous abstractions and techniques have been explored, including mix-nets [Cha81, Abe99, BG12], the Dining Cryptographers' nets [Cha88, CGF10, APY20], onion routing [DMS04, GRS99, DS18, CL05], multi-party-computation-based approaches [AKTZ17], multi-server PIR-write [CBM15, OS97, GIKM00], as well as variants/improvements of the above [ZZZR05, vdHLZZ15, TGL+17]. We refer the readers to several excellent surveys on this rich line of work [DD08, EY09, SSA+18].

To the best of our knowledge, almost all known anonymous routing schemes rely on *multiple* routers or servers to engage in an *interactive protocol*, and moreover, security is guaranteed in the *threshold* model, i.e., assuming that one or more of the routers remain honest. For example, the mix-net family of schemes typically require each router along the way to shuffle the input ciphertexts and remove a layer of encryption; the DC-net family of schemes require multiple parties to engage in a cryptographic protocol, and so on.

Departing from all prior approaches which are interactive and rely on some form of threshold cryptography, we ask the following natural question:

Can we achieve anonymous routing *non-interactively* on a *single untrusted* router?

1.1 Defining Non-Interactive Anonymous Router (NIAR)

Our first contribution is a conceptual one: we formulate a new abstraction called a non-interactive anonymous router (NIAR). The abstraction is in fact quite natural, and in hindsight, it may even be a little surprising why it has not been considered before.

Non-interactive anonymous router. Imagine that there are n senders and n receivers, and each sender wishes to speak with a distinct receiver. Henceforth let π denote the permutation that maps each sender to its intended receiver, i.e., each sender $i \in [n]$ wants to speak to receiver $\pi(i)$. A NIAR scheme has the following syntax:

- $(\{\mathsf{ek}_i, \mathsf{rk}_i\}_{i\in[n]}, \mathsf{tk}) \leftarrow \mathbf{Setup}(1^\kappa, n, \pi)$: First, we run a one-time trusted setup procedure that takes the security parameter 1^κ, the number of senders/receivers n, and the routing permutation π, and produces a *sender key* ek_i for each sender $i \in [n]$, and a *receiver key* rk_i for each receiver $i \in [n]$. Moreover, the setup procedure also produces a *token* tk for the router which encodes the secret permutation π. Note that the trusted setup can be decentralized using standard multi-party computation techniques.
- $\mathsf{ct}_{i,t} \leftarrow \mathbf{Enc}(\mathsf{ek}_i, x_{i,t}, t)$: With this one-time setup, we can allow the n senders to anonymously send T number of packets to their intended receivers. In every time step $t \in [T]$, each sender $i \in [n]$ encrypts its message $x_{i,t}$ using its secret key ek_i by calling $\mathbf{Enc}(\mathsf{ek}_i, x_{i,t}, t)$, and sends the resulting ciphertext $\mathsf{ct}_{i,t}$ to the router.
- $\{\mathsf{ct}'_{i,t}\}_{i\in[n]} \leftarrow \mathbf{Rte}(\mathsf{tk}, \{\mathsf{ct}_{i,t}\}_{i\in[n]})$: The untrusted router uses its token to convert the n ciphertexts collected from the senders into n *transformed ciphertexts*. This is accomplished by calling $\mathbf{Rte}(\mathsf{tk}, \{\mathsf{ct}_{i,t}\}_{i\in[n]})$. The router then forwards each transformed ciphertext $\mathsf{ct}'_{i,t}$ to the corresponding recipient i.

- $x_i \leftarrow \mathbf{Dec}(\mathsf{rk}_i, \mathsf{ct}'_{i,t})$: Finally, the recipients use their respective secret keys to decrypt the plaintexts by calling $\mathbf{Dec}(\mathsf{rk}_i, \mathsf{ct}'_{i,t})$.

At a very high level, we want that the untrusted router learns no information about the routing permutation π as well as the messages exchanged. Moreover, the scheme should offer robustness even when a (potentially majority) subset of the senders and/or receivers collude with the untrusted router. It turns out that defining robustness under collusion is non-trivial and the security requirements can vary from application to application — we will discuss the security definitions in more detail later.

Communication efficiency. The first naïve idea is to let each sender-receiver pair share a freshly and randomly chosen secret key during the setup. During each time step, each sender encrypts its messages using its secret key, and sends the ciphertext to the router. The router then forwards all n ciphertexts to each of the n receivers; and each receiver's secret key allows it to decrypt exactly one among the n ciphertexts received. This scheme protects the plaintext messages as well as the routing permutation π from the untrusted router; unfortunately, it incurs *quadratic* communication overhead in each time step[1].

Throughout the rest of the paper, we will require that the NIAR scheme preserve *communication efficiency*, that is, the communication blowup relative to sending the messages in the clear must be upper bounded by $\mathsf{poly}(\kappa)$ where κ is the security parameter. In other words, suppose, without loss of generality, that in each time step, each sender has one bit to send, then the total communication (among all senders and receivers) per time step must be upper bounded by $O(n) \cdot \mathsf{poly}(\kappa)$.

Non-interactive anonymous shuffler. One important application and special case of NIAR is to realize a non-interactive anonymous shuffler (NIAS). To understand what is a non-interactive anonymous shuffler, it helps to think of the following application. Suppose that during a pandemic, University X wants to implement a privacy-preserving daily check mechanism, where students and faculty each send a short message to report their health conditions every day, and whether they could have been exposed to the virus. To protect each individual's privacy, we want to shuffle the messages according to some randomly chosen permutation π, such that the history of an individual's reports is pseudonymous. In this scenario, we can employ a NIAR scheme, and give the data analyst the token tk as well as *all* n receiver keys. This ensures that the data analyst can decrypt only a shuffled list of the plaintexts, and moreover the permutation is hidden from the data analyst.

In other words, a Non-Interactive Anonymous Shuffler (NIAS) is a special case of NIAR where the router and all the receivers are a single party. In Section 1.4, we will present numerous applications of NIAR and NIAS. We point out that the NIAS special case in fact imposes some extra security requirements on top of our basic security notion for NIAR, in the sense that even a receiver cannot know which sender it is paired up with — we will discuss how to define security next.

1.2 Defining Security Requirements

If all receivers were fully trusted, then another naïve idea would be to have every sender encrypt its message along with its respective destination using a Fully Homomorphic Encryption (FHE) scheme. In this way, the untrusted router can accomplish the routing through homomorphic evaluation. However, all receivers must be given the FHE's secret key to decrypt the messages. Therefore, if even a single receiver colludes with the untrusted router, then all other honest players' anonymity

[1]Furthermore, while this naïve scheme works for a private-messaging scenario, and does not work for the non-interactive anonymous shuffler application to be described later, due to the fact that a receiver colluding with the router can learn which sender it is paired with. We will elaborate on this point when we define security.

would be broken. This is clearly unacceptable since in most applications of anonymous routing, anyone can become a sender or a receiver, including the owner of the router. Approaches that construct special-purpose homomorphic encryption schemes optimized for shuffling suffer from the same drawback [AW07].

We therefore require a security notion that provides robustness even when a subset of the senders and receivers can be corrupt, and potentially colluding with the untrusted router. It turns out that how to define robustness against collusion requires some careful thinking, since the security requirements can vary from application to application.

Basic notion. Our basic security notion is motivated by a private-messaging scenario, e.g., members of a secret society wish to send private emails without revealing their identities and their correspondence to the public. In this case, each player (i.e., either sender or receiver) knows who it is talking to. Therefore, if the adversary who controls the router additionally corrupts a subset of the senders and receivers, the adversary can learn who the corrupt senders and receivers are paired up with, as well as the messages received by corrupt receivers (from honest senders) in every time step. Our basic security notion requires that besides this natural leakage, the adversary should not learn any additional information. Observe that our communication-inefficient naïve solution that forwards all ciphertexts to every receiver would satisfy this basic notion.

Receiver-insider protection. The basic security notion, however, turns out to be insufficient for the NIAS application. In the NIAS application, a single entity acts as the router and all n receivers — for example, in our earlier "anonymous daily check-in" application, the data analyst has all receiver keys $\{\mathsf{rk}_i\}_{i\in[n]}$ as well as the token tk. To protect the users' pseudonymity, it is important that the data analyst does not learn which decrypted report corresponds to which user. In Section 1.4, we present more applications for NIAS, and all of them have the same security requirement.

We therefore propose a strengthened security notion, called receiver-insider protection, that is suitable and sufficient for NIAS-type applications. Here, we require that even a receiver does not learn which sender it is speaking with; however, a sender may learn which receiver it is speaking with. Now, if the adversary who controls the router additionally corrupts a subset of the senders and receivers, the adversary can learn the *corrupt-to-*$*$ part[2] of the permutation π as well as the messages received by corrupt receivers in every time step. Besides this natural leakage, the adversary should not learn anything else.

Full insider protection. While receiver-insider protection seems sufficient for most applications including NIAS, we additionally explore a *paranoid* notion of security. Here, we want that every player has no idea who it is speaking with, including both senders and receivers. Nonetheless, the *corrupt-to-corrupt* part of the permutation is *inherently* leaked to the adversary and this leakage cannot be avoided: since a corrupt sender can always try encrypting some message and check whether any corrupt receiver received the corresponding message. Therefore, our most paranoid notion, which we call *full insider protection*, requires that an adversary controlling the router and a subset of corrupt senders and receivers learns only the corrupt-to-corrupt part of the permutation π, as well as the messages received by corrupt receivers in every time step, but nothing else.

In Section 1.4, we describe more applications of NIAR and NIAS, and at that point, the reader can see how different applications require different notions. Of course, one can always go for the most paranoid notion; but the weaker notions suffice for a wide range of natural applications.

[2]Here, $*$ denotes a wildcard; thus the corrupt-to-$*$ part of the permutation includes who every corrupt sender is speaking with.

Therefore, differentiating between these notions can lead to more efficient constructions.

Equivalence between simulation- and indistinguishability-based notions. Later in the paper, we shall formalize the above security notions using two definitional approaches: *simulation-based* notions and *indistinguishability-based* notions. We then prove that in fact, *each simulation-based notion* (without insider protection, with receiver-insider protection, or with full insider protection) *is equivalent to the corresponding indistinguishability-based notion.* While the simulation-based notion more naturally captures the security requirements we want to express, the indistinguishability-based notions are often easier to work with in proofs.

Remark 1 (NIAR/NIAS requires no network-layer anonymity protection)**.** We point out that whenever a NIAR or NIAS scheme is deployed, one advantage is that *we would no longer need any network-layer anonymity protection* (e.g., Tor [DMS04] or DC nets [Cha88]). This is in contrast to a vast line of works that leverage cryptographic techniques such as zero-knowledge proofs for anonymity protection, e.g., E-Cash [CFN90, Cha82], e-voting [SK95, Adi08], anonymous credentials [BCKL08, BL13], ZCash [BCG+14], and others [HAB+17, HMPS14] — in these cases, an Internet Service Provider controlling the network routers can completely break anonymity despite the cryptographic techniques employed.

1.3 Our Results

1.3.1 Main Construction: NIAR with Receiver-Insider Protection and NIAS

To situate our results in context, it helps to first think of the following naïve construction based on a virtual-blackbox (VBB) obfuscator. During setup, we publish the public key pk of a public-key encryption (PKE) scheme, and moreover, we give each sender-receiver pair a symmetric encryption key. During each routing step, each sender uses its symmetric key to encrypt its respective message, resulting what we henceforth call an *inner ciphertext*. The sender then encrypts the inner ciphertext with the public key encryption scheme, resulting in an *outer ciphertext*. During the setup, we give the router a VBB obfuscation of the following program: use the PKE's secret key to decrypt each sender's outer ciphertext, obtain a list of n inner ciphertexts, and then apply the permutation π to the n inner ciphertexts and output the result. Now, during each routing step, the router can simply apply its VBB obfuscated program to the list of n outer ciphertexts collected from the senders, and the result would be n permuted inner ciphertexts. The i-th inner ciphertext is then forwarded to the i-th receiver where $i \in [n]$. Note that in this VBB-based solution, the program obfuscation hides the secret key of the PKE scheme as well as the secret permutation π. One can verify that indeed, this VBB-based construction satisfies security with *receiver-insider protection*; but it does *not* provide full insider protection. Specifically, a corrupt sender $i^* \in [n]$ colluding with the router can simply plant a random inner ciphertext c, and see which of the receivers receives c at the end — this must be the receiver i^* is speaking with[3].

The drawback with this naïve solution is obvious: it is well-known that VBB obfuscation is impossible to attain for general functions if one-way functions exist [BGI+01]. We therefore ask,

Can we construct a NIAR scheme from standard cryptographic assumptions?

We construct a NIAR scheme that achieves security with receiver-insider protection, relying on the Decisional Linear assumption in suitable bilinear groups. Our scheme satisfies communication efficiency: in each time step, each player sends or receives only $\mathsf{poly}(\kappa)$ bits of data (assuming,

[3]In general, achieving full insider security appears much more challenging than our basic notion or receiver-only insider protection. Indeed, we will discuss this in further detail later on.

without loss of generality, that each sender wants to send one bit during each time step). Furthermore, the public and secret key sizes are $\mathsf{poly}(n, \kappa)$; and yet the scheme can support an *unbounded* number of time steps.

At a high level, in our construction, each sender creates an inner encryption of its message using a symmetric key shared with its receiver, and then encrypts the inner ciphertext again using a special outer encryption scheme. With an appropriately constructed token, the router can output a permuted list of inner ciphertexts. We state the aforementioned result in the following theorem:

Theorem 1.1 (NIAR with receiver-insider protection)**.** *Assume that the Decisional Linear assumption holds in certain bilinear groups. Then, there exists a NIAR scheme with receiver-insider protection, where the public key and secret key sizes are at most* $\mathsf{poly}(n, \kappa)$ *bits, and the per-player communication cost in each routing step is only* $\mathsf{poly}(\kappa)$ *assuming that each sender has one bit to send per time step. Further, the scheme supports an unbounded number of time steps.*

The above theorem also implies a NIAS scheme with the same performance bounds. Although our work should primarily be viewed as an initial exploration of NIAR, the constructions that led to Theorem 1.1 is potentially implementable.

1.3.2 NIAR with Full Insider Protection

The receiver-insider protection achieved by Theorem 1.1 is sufficient for most application scenarios including NIAS. Nonetheless, it is interesting to ask whether one can achieve full insider protection. As mentioned, full insider protection is the strongest security notion one can hope for in the context of NIAR, since here we leak only the inevitable. Achieving full insider security, however, appears much more challenging. The reason is that we do not even want a corrupt sender to learn which honest receiver it is talking to. However, in our schemes so far (even the aforementioned VBB-based construction), a corrupt sender $i^* \in [n]$ colluding with the router can choose a random inner ciphertext c and just check which receiver receives c. In this way, the adversary can learn the corrupt-to-$*$ part of the permutation π.

Again, it is instructive to first consider how to achieve full insider protection using VBB obfuscation. To achieve such paranoid security, one way is to modify the previous VBB-based scheme such that inside the VBB, we decrypt the n input ciphertexts, permute them, and then *reencrypt* them under the receivers' keys, respectively. To defeat the aforementioned attack, it is important that the reencryption step produces *random* transformed ciphertexts. In fact, one useful insight we can draw here is that for any scheme that provides full insider protection, if the adversary controlling a corrupt sender $\widetilde{i} \in [n]$ switches $\widetilde{i}$'s input ciphertext, the transformed ciphertexts corresponding to *all* receivers output by the **Rte** procedure must all change.

We show how to achieve full insider protection by additionally relying on sub-exponentially secure indistinguishability obfuscation and sub-exponentially secure one-way functions.

Theorem 1.2 (NIAR with full insider protection)**.** *Assume the existence of sub-exponentially secure indistinguishability obfuscator, sub-exponentially secure one-way functions, and that the Decisional Linear assumption (with standard polynomial security) holds in certain bilinear groups. Then, there exists a NIAR scheme with full insider protection, and whose key sizes and communication cost match those of Theorem 1.1.*

Notably, a flurry of very recent works [JLS20,GP20,WW20,BDGM20] show that sub-exponentially secure indistinguishability obfuscator can be constructed under a variety of assumptions some of which are considered well-founded.

1.3.3 Extension: Fault-Tolerant NIAR

Similar to the line of work on Multi-Client Functional Encryption (MCFE) [SCR+11, GGG+14, CSG+18, ABG19, CSG+18], a drawback with the present formulation is that a single crashed sender can hamper liveness. Basically, the router must collect ciphertexts from all senders in each time step to successfully evaluate the **Rte** procedure. To the best of our knowledge, fault tolerance has been little investigated in this line of work.

We therefore formulate a variation of our basic NIAR abstraction, called *fault-tolerant NIAR*. In a fault-tolerant NIAR, if a subset of the senders have crashed, the remaining set of senders can encrypt their messages in a way that is aware of the set of senders who are known to be still online (henceforth denoted $\mathcal{O}$). Similarly, the router will perform the **Rte** procedure in a way that is aware of $\mathcal{O}$, too. In this way, the router can continue to perform the routing, without being stalled by the crashed senders. Similar to our basic notion, we define receiver-insider protection and full-insider protection for our fault-tolerant NIAR abstraction, and show that the most natural simulation-based and indistinguishability-based notions are equivalent.

We show that our previous NIAR constructions of Theorem 1.1 and Theorem 1.2 can be extended to the fault-tolerant setting, and the result is stated in the following theorem.

Theorem 1.3 (Informal: fault-tolerant NIAR)**.** *Suppose that the Decisional Linear assumption holds in suitable bilinear groups. Then, there exists a fault-tolerant NIAR scheme that leaks only the (corrupt+crashed)-to-$*$ part of the permutation as well as messages received by corrupt receivers, but nothing else (see Section 8 for formal security definitions).*

Suppose that the Decisional Linear assumption (with standard polynomial security) holds in suitable bilinear groups, and assume the existence of sub-exponentially secure indistinguishability obfuscation and one-way functions. Then, there there exists a fault-tolerant NIAR scheme that leaks only the inherent leakage, that is, the (corrupt+crashed)-to-corrupt part of the permutation as well as messages received by corrupt receivers, but nothing else (see Section 8 for formal security definitions).

Furthermore, both schemes achieve the same key sizes and communication efficiency as in Theorem 1.1.

1.4 Applications of NIAR and NIAS

NIAR adds to the existing suite of primitives [Cha81, Abe99, BG12, Cha88, CGF10, DMS04, GRS99, CBM15, ZZZR05, vdHLZZ15, TGL+17] that enable anonymous routing. In comparison with prior works, NIAR adopts a different trust model since it does not rely on threshold cryptography. Arguably it also has a somewhat simpler abstraction than most existing primitives, partly due to the non-interactive nature.

We discuss two flavors of applications for NIAR, including 1) using NIAR in private messaging, which is the more classical type of application; and 2) using NIAR as a non-interactive anonymous shuffler (NIAS). We will use these applications to motivate the need for the different security notions, without insider protection, with receiver-insider protection, or with full insider protection. We shall begin with NIAS-type applications since some of these applications are of emerging interest.

1.4.1 Using NIAR as a Non-Interactive Anonymous Shuffler

NIAR can serve as *a non-interactive anonymous shuffler* (NIAS), which shuffles n senders' messages in a non-interactive manner, such that the messages become unlinkable to their senders. This allows the senders to publish messages under a pseudonym, and the pseudonymity does not have to rely

on the network layer being anonymous. In a non-interactive shuffler type of application, typically a single entity acts as the router and all n receiver — therefore, typically these applications require *receiver-insider* protection. To understand what is a non-interactive anonymous shuffler, it is most instructive to look at some example applications.

Anonymous bulletin board or forum. Imagine that a group of users want to post messages *pseudonymously* to a website every day, e.g., to discuss some sensitive issues. The users act as the NIAR senders and encrypt their messages every day. The server, which acts as both the router and all the receivers in NIAR, decrypts a permuted list of the messages and posts them on the website. In this way, the untrusted server can mix the n senders' messages, and the pseudonymity guarantee need not rely on additional network-layer anonymity protection. In other words, even a powerful attacker controlling all routers in the world as well as the server cannot break the pseudonymity guarantees.

Since the server takes the role of the router and all n receivers, we would need a NIAR scheme that provides *receiver-insider protection*. This way, even when all the receivers are in the control of the adversary, the adversary cannot deanonymize honest senders.

Distributed differential privacy in the shuffle model. There has been a growing appetite for large-scale, privacy-preserving federated learning, especially due to interest and investment from big players such as Google and Facebook. Unlike the classical "central model" where we have a trusted database curator [DMNS06], in a federated learning scenario, the data collector is not trusted, and yet it wants to learn interesting statistics and patterns over data collected by multiple users' mobile phones, web browsers, and so on. This model is often referred to the "local model". It is understood that without any additional assumptions and without cryptographic hardness, mechanisms in the local model incur a utility loss [CSS12, SCRS17, BNO08] that is significantly worse than the central model (given a fixed privacy budget).

Recently, an elegant line of work [CSU+19, BBGN19b, GPV19, BBGN19a, EFM+19, BEM+17] emerged, and showed that if there exists a shuffler that randomly shuffles the users' input data, then we can design (information-theoretic) distributed differential privacy mechanisms that are often competitive to the central model. This is commonly referred to as the "shuffle model".

NIAR can be potentially employed to implement a shuffler for the shuffle model. In particular, it is suited for a setting like Google's RAPPOR project [EPK14], where data was repeatedly collected from the users' Chrome browsers on a daily basis. In this scenario, the data collector acts as the NIAR router and all the receivers too; therefore, we also need the NIAR scheme to satisfy receiver-insider protection. Again, we do not need network-level anonymity protection.

Privacy-preserving "daily check" during a pandemic. This application was described earlier in this section. We additionally point out an interesting variation of the same application: we can *create the inner layer of encryption using not symmetric-key encryption, but rather, a predicate encryption scheme* [SW05, SBC+07, BW07, BSW11, GVW15, AGW20]. In this way, a data analyst can be granted special tokens that would permit her to decrypt the data, only if some predicate is satisfied over the user's encrypted daily report (e.g., the user has come in contact with an infected person and needs to be quarantined).

Pseudonymous survey systems. Another application is to build a pseudonymous survey system. For example, we can allow students to pseudonymously and regularly post course feedback to an instructor throughout the semester, or ask questions that they would otherwise feel embarrassed

to ask. We can also create periodic surveys and allow members of an underrepresented minority group to pseudonymously report if they have been the victims of discrimination or harassment. Similar applications have been considered and implemented in the past [HMPS14, Adi08]. However, in such existing mechanisms [HMPS14, Adi08], the cryptographic protection alone is insufficient, and one must additionally rely on the network layer to be anonymous too. By contrast, with NIAR, we no longer need the network layer to provide anonymity protection.

Other applications. Besides these aforementioned applications, it is also known that a shuffler can lend to the design of light-weight multi-party computation (MPC) protocols [IKOS06].

1.4.2 Private Messaging

NIAR can also be used to enable private messaging, which is the more traditional application of anonymous routing. We give a few scenarios to motivate the different security requirements.

In the first scenario, we may imagine that members of a secret society wish to send private messages or emails to one another without identified. To do so, pairs of members that wish to communicate regularly can join a NIAR group. In this scenario, each pair of communicating parties know each other's identities, and therefore we only need the basic security notion, i.e., without insider protection.

Another application is to build an anonymous mentor-mentee system, or an anonymous buddy or mutual-support system. For example, some scientists have relied on Slack to provide such functionalities [sys], where members can anonymously post questions, and others can anonymously provide advice. Currently, the anonymity guarantee is provided solely by the Slack server. However, one can easily imagine scenarios where trusting a centralized party for anonymity is undesirable. In these cases, we can rely on NIAR to build an anonymous buddy system. Each pair of buddies can regularly engage in conversations to provide mutual support, and the untrusted router (e.g., Slack) cannot learn the communication pattern or the messages being exchanged. Like the earlier mentor-mentee scenario, the buddies themselves may not wish to reveal their identities to each other. Therefore, in this scenario, we would need the NIAR scheme to provide *full insider protection*.

1.5 Open Questions

Partly, our work makes a conceptual contribution since we are the first to define the NIAR and NIAS abstractions. Our work should be viewed as an initial exploration of these natural abstractions, inspired by a fundamental and long-standing online privacy problem. Many open questions arise given our new abstractions, and our work lays the groundwork for further exploring exciting future directions. We present a list of open questions in Section 9.

1.6 Technical Highlight

Why existing approaches fail. A first strawman attempt is to rely on a Multi-Client Functional Encryption (MCFE) scheme for inner products, also known as Multi-Client Inner-Product Encryption (MCIPE) [GGG+14, ABG19, ABM+20, CSG+18]. Multi-Client Functional Encryption (MCFE) was originally proposed by Shi et al. [SCR+11] where they considered the simple summation operation, and showed a construction based on Decisional Diffie-Hellman. Subsequently, Goldwasser et al. [GGG+14] generalized the notion to arbitrary polynomially sized functions, and showed a construction based on Indistinguishability Obfuscation [GGH+13] in the random oracle

model. Several other works considered MCFE for inner products, also called Multi-Client Inner-Product Encryption (MCIPE) [GGG+14, ABG19, ABM+20, CSG+18]

In a Multi-Client Inner-Product Encryption scheme, each of the n clients obtains a secret encryption key during a setup phase. During every time step t, each client i uses its secret encryption key to encrypt a message $x_{i,t}$ — henceforth the i-th ciphertext is denoted $\mathsf{ct}_{i,t}$ for $i \in [n]$, and moreover, let $\mathbf{x}_t := (x_{1,t}, \ldots, x_{n,t})$. An authority with a master secret key can generate a functional key $\mathsf{sk}_{\mathbf{y}}$ for a vector $\mathbf{y}$ whose length is also n. Given the collection of ciphertexts $\{\mathsf{ct}_{i,t}\}_{i\in[n]}$ and the functional key $\mathsf{sk}_{\mathbf{y}}$, one can evaluate the function $\langle \mathbf{x}_t, \mathbf{y} \rangle$ of the encrypted plaintexts but nothing else is revealed.

Our idea is to express the permutation π as n selection vectors, and each is used to select what one receiver would receive from the vector of input messages. The router receives one functional key for each selection vector. A selection vector $\mathbf{y}$ has exactly one coordinate that is set to 1, whereas all other coordinates are set to 0. In this way, the inner product of $\mathbf{x}_t$ and $\mathbf{y}$ selects exactly one coordinate of $\mathbf{x}_t$. In our NIAR construction, the input messages $\mathbf{x}_t$ to the MCFE-for-selection scheme will be inner ciphertexts encrypted under keys shared between each pair of sender and receiver, such that the router cannot see to the plaintext message.

At first sight, an MCFE scheme for inner products may seem like a good match for our problem, but upon more careful examination, all known MCFE schemes, including those based on program obfuscation, fail in our context. To the best of our knowledge, *all existing MCFE schemes* (for evaluating any function, not just inner products) *are NOT function-hiding*. In our context, this means that the functional key $\mathsf{sk}_{\mathbf{y}}$ is allowed to reveal the selection vector $\mathbf{y}$. This unfortunately means that the token could leak the routing permutation π and thus violate anonymity. Not only so, in fact, it appears that *no prior work has attempted to define or construct function-hiding MCFE* [SCR+11, ABG19, ABM+20, CSG+18, LT19], likely because we currently lack techniques to get function privacy for MCFE schemes, even allowing RO and program obfuscation [GGG+14]. The known techniques for upgrading Functional Encryption and Multi-Input Functional Encryption to have function privacy [SSW09, ACF+18, BJK15, LV16, Lin17] do not apply to MCFE, because they are fundamentally *incompatible with the scenario where some clients can be corrupt*.

Finally, we point out that a related line of work called *Multi-Input* Inner-Product Encryption [ACF+18, AGRW17, BKS18, GGG+14] also fails to solve our problem, because its security definition is too permissive: specifically, mix-and-matching ciphertexts from multiple time steps is allowed during evaluation, and this could be exploited by an adversary to break anonymity in our context.

Key insights and roadmap. We are the first to define function-hiding MCFE[4], and demonstrate a construction for a meaningful functionality, i.e., *selection*. Selection is a special case of inner product computation, and is structurally simpler than inner product. Leveraging this structural simplicity, we develop new construction and proof techniques that allow us to prove function-hiding security even when some of the clients can be corrupted. We use the resulting "function-hiding MCFE for selection" as a core building block to realize NIAR.

At a very high level, the structural simplicity of selection helps us in the following way. First, in a more general MCIPE scheme, even without function privacy, one must prevent mix-and-match attacks — in other words, the adversary should not be able to take clients' ciphertext from different time steps and combine them in the same inner-product evaluation. When it comes to the special case of selection, however, we can *defer* the handling of such mix-and-match attacks. Specifically, if

[4]The concurrent and independent work of Agrawal et al. [AGT20] also define function-hiding MCFE, and they show a construction for inner-product queries. However, their construction relies on random oracles which we do not need in this paper.

we were not concerned about function privacy, then mix-and-match attacks turned out to be a non-issue in an MCFE-for-selection scheme. With this observation, we first construct a conceptually simple MCFE-for-selection scheme *without* function privacy. Essentially, the construction runs n independent instances of semantically secure public-key encryption (PKE), one for each client. The functional key for selecting one client's plaintext is simply the corresponding PKE's secret key.

Next, we perform a function-privacy upgrade — during this function-privacy upgrade, we do need to take care and prevent the aforementioned *mix-and-match* attacks. The function-privacy upgrade is technically much more involved, and we will give an informal overview in Section 3. What lends to the function-privacy upgrade is the fact that the underlying MCFE scheme (without function privacy) is essentially "decomposable" into n independent components. This is an important reason why we can accomplish the function privacy upgrade *even when some of the clients can be corrupted*. In comparison, prior MCFE schemes for general inner-products [ABG19, CSG+18, ABM+20] need more structurally complicated techniques to prevent mix-and-match, even without function privacy. For this reason, our techniques in the current form are not capable of getting a *function-private* MCFE scheme for general inner-products — this remains a challenging open question.

Once we construct a function-hiding MCFE-for-selection scheme, we then use it to construct two NIAR schemes: one with receiver-insider protection, and one with full insider protection. The scheme with receiver-insider protection can be constructed without introducing additional assumptions — and this notion of security suffices for most applications including NIAS. As explained in Section 1.3.2, full insider protection seems much more challenging and a natural class of approaches fail. To get a paranoid construction with full insider protection, we additionally rely on sub-exponentially secure indistinguishability obfuscation and sub-exponentially one-way functions.

2 New Definitions: Non-Interactive Anonymous Router

We now define the syntax and security requirements of NIAR. Since our approach relies on a single untrusted router and is non-interactive, both the syntax and security definitions are incomparable to the formal definitions of anonymous routing in prior works, all of which involve multiple routers and interactive protocols [CL05, DS18, AKTZ17].

2.1 Syntax

Suppose that there are n senders and n receivers, and each sender wants to talk to a distinct receiver. They would like to route their messages anonymously to hide who is talking to who. The routing is performed by a single router non-interactively.

Let $\mathsf{Perm}([n])$ denote the set of all permutations on the set $[n]$. Let $\pi \in \mathsf{Perm}([n])$ be a permutation that represents the mapping between the sender and the receivers. For example, $\pi(1) = 3$ means that sender 1 wants to talk to receiver 3.

A Non-Interactive Anonymous Router (NIAR) is a cryptographic scheme consisting of the following, possibly randomized algorithms:

- $(\{\mathsf{ek}_i\}_{i\in[n]}, \{\mathsf{rk}_i\}_{i\in[n]}, \mathsf{tk}) \leftarrow \mathbf{Setup}(1^\kappa, n, \pi)$: the trusted **Setup** algorithm takes the security parameter 1^κ, the number of senders/receivers n, and a permuation $\pi \in \mathsf{Perm}([n])$ that represents the mapping between the senders and the receivers. The **Setup** algorithm outputs a sender key for each sender denoted $\{\mathsf{ek}_i\}_{i\in[n]}$, a receiver key for each receiver denoted $\{\mathsf{rk}_i\}_{i\in[n]}$, and a token for the router denoted tk.

- $\mathsf{ct}_{i,t} \leftarrow \mathbf{Enc}(\mathsf{ek}_i, x_{i,t}, t)$: sender i uses its sender key ek_i to encrypt the message $x_{i,t}$ where $t \in \mathbb{N}$ denotes the current time step. The **Enc** algorithm produces a ciphertext $\mathsf{ct}_{i,t}$.
- $(\mathsf{ct}'_{1,t}, \mathsf{ct}'_{2,t}, \ldots, \mathsf{ct}'_{n,t}) \leftarrow \mathbf{Rte}(\mathsf{tk}, \mathsf{ct}_{1,t}, \mathsf{ct}_{2,t}, \ldots, \mathsf{ct}_{n,t})$: the routing algorithm **Rte** takes its token tk (which encodes some permutation π), and n ciphertexts received from the n senders denoted $\mathsf{ct}_{1,t}, \mathsf{ct}_{2,t}, \ldots, \mathsf{ct}_{n,t}$, and produces *transformed ciphertexts* $\mathsf{ct}'_{1,t}, \mathsf{ct}'_{2,t}, \ldots, \mathsf{ct}'_{n,t}$ where $\mathsf{ct}'_{i,t}$ is destined for the receiver $i \in [n]$.
- $x \leftarrow \mathbf{Dec}(\mathsf{rk}_i, \mathsf{ct}'_{i,t})$: the decryption algorithm **Dec** takes a receiver key rk_i, a transformed ciphertext $\mathsf{ct}'_{i,t}$, and outputs a decrypted message x.

In our formulation above, the permutation π is known a-priori at **Setup** time. Once **Setup** has been run, the senders can communicate with the receivers over multiple time steps t.

Correctness. Without loss of generality, we may assume that each plaintext message is a single bit — if the plaintext contains multiple bits, we can always split it bit by bit and encrypt it over multiple time steps. Correctness requires that with probability 1, the following holds for any $\kappa \in \mathbb{N}$, any $(x_1, x_2, \ldots, x_n) \in \{0,1\}^n$ and any $t \in \mathbb{N}$: let $(\{\mathsf{ek}_i\}_{i\in[n]}, \{\mathsf{rk}_i\}_{i\in[n]}, \mathsf{tk}) \leftarrow \mathbf{Setup}(1^\kappa, n, \pi)$, let $\mathsf{ct}_{i,t} \leftarrow \mathbf{Enc}(\mathsf{ek}_i, x_i, t)$ for $i \in [n]$, let $(\mathsf{ct}'_{1,t}, \mathsf{ct}'_{2,t}, \ldots, \mathsf{ct}'_{n,t}) \leftarrow \mathbf{Rte}(\mathsf{tk}, \mathsf{ct}_{1,t}, \mathsf{ct}_{2,t}, \ldots, \mathsf{ct}_{n,t})$, and let $x'_i \leftarrow \mathbf{Dec}(\mathsf{rk}_i, \mathsf{ct}'_{i,t})$ for $i \in [n]$; it must be that

$$x'_{\pi(i)} = x_i \text{ for every } i \in [n].$$

Communication compactness. We require our NIAR scheme to have *compact communication*, that is, the total communication cost per time step should be upper bounded by $\mathsf{poly}(\kappa) \cdot O(n)$. Furthermore, we would like that the token tk, and every sender and receiver's secret key ek_i and rk_i respectively, are all upper bounded by a fixed polynomial in n.

2.2 Simulation-Based Security

We consider static corruption where the set of corrupt players are chosen prior to the **Setup** algorithm.

Real-world experiment $\mathsf{Real}^{\mathcal{A}}(1^\kappa)$. The real-world experiment is described below where $\mathcal{K}_S \subseteq [n]$ denotes the set of corrupt senders, and $\mathcal{K}_R \subseteq [n]$ denotes the set of corrupt receivers. Let $\mathcal{H}_S = [n] \setminus \mathcal{K}_S$ be the set of honest senders and $\mathcal{H}_R = [n] \setminus \mathcal{K}_R$ be the set of honest receivers. Let $\mathcal{A}$ be a *stateful* adversary:

- $n, \pi, \mathcal{K}_S, \mathcal{K}_R \leftarrow \mathcal{A}(1^\kappa)$
- $(\{\mathsf{ek}_i\}_{i\in[n]}, \{\mathsf{rk}_i\}_{i\in[n]}, \mathsf{tk}) \leftarrow \mathbf{Setup}(1^\kappa, n, \pi)$
- For $t = 1, 2, \ldots$:
  - if $t = 1$ then $\{x_{i,t}\}_{i\in\mathcal{H}_S} \leftarrow \mathcal{A}(\mathsf{tk}, \{\mathsf{ek}_i\}_{i\in\mathcal{K}_S}, \{\mathsf{rk}_i\}_{i\in\mathcal{K}_R})$; else $\{x_{i,t}\}_{i\in\mathcal{H}_S} \leftarrow \mathcal{A}(\{\mathsf{ct}_{i,t-1}\}_{i\in\mathcal{H}_S})$;
  - for $i \in \mathcal{H}_S$, $\mathsf{ct}_{i,t} \leftarrow \mathbf{Enc}(\mathsf{ek}_i, x_{i,t}, t)$

Ideal-world experiment $\mathsf{Ideal}^{\mathcal{A},\mathsf{Sim}}(1^\kappa)$. The ideal-world experiment involves not just $\mathcal{A}$, but also a p.p.t. (stateful) simulator denoted Sim, who is in charge of simulating $\mathcal{A}$'s view knowing essentially only what corrupt senders and receivers know. Further, the $\mathsf{Ideal}^{\mathcal{A},\mathsf{Sim}}(1^\kappa)$ experiment is parametrized by a leakage function denoted Leak to be defined later. Henceforth for $\mathcal{C} \subseteq [n]$, we use $\pi(\mathcal{C})$ to denote the set $\{\pi(i) : i \in \mathcal{C}\}$.

- $n, \pi, \mathcal{K}_S, \mathcal{K}_R \leftarrow \mathcal{A}(1^\kappa)$
- $(\{\mathsf{ek}_i\}_{i\in[n]}, \{\mathsf{rk}_i\}_{i\in[n]}, \mathsf{tk}) \leftarrow \mathsf{Sim}(1^\kappa, n, \mathcal{K}_S, \mathcal{K}_R, \mathsf{Leak}(\pi, \mathcal{K}_S, \mathcal{K}_R))$
- For $t = 1, 2, \ldots$:
  - if $t = 1$ then $\{x_{i,t}\}_{i\in\mathcal{H}_S} \leftarrow \mathcal{A}(\mathsf{tk}, \{\mathsf{ek}_i\}_{i\in\mathcal{K}_S}, \{\mathsf{rk}_i\}_{i\in\mathcal{K}_R})$; else $\{x_{i,t}\}_{i\in\mathcal{H}_S} \leftarrow \mathcal{A}(\{\mathsf{ct}_{i,t-1}\}_{i\in\mathcal{H}_S})$;
  - $\{\mathsf{ct}_{i,t}\}_{i\in\mathcal{H}_S} \leftarrow \mathsf{Sim}\left(\{\forall i \in \mathcal{K}_R \cap \pi(\mathcal{H}_S) : (i, x_{j,t}) \text{ for } j = \pi^{-1}(i)\}\right)$. In other words, the simulator Sim is allowed to see for each corrupt receiver talking to an honest sender, what message it receives.

Defining the insider information $\mathsf{Leak}(\pi, \mathcal{K}_S, \mathcal{K}_R)$ known to corrupt players. We require that *when no sender or receiver is corrupt, the adversary should not learn anything about the routing permutation π.* When some senders and receivers are corrupt, the adversary *may learn the insider information about π known to the corrupt players, but nothing else.* We use the function $\mathsf{Leak}(\pi, \mathcal{K}_S, \mathcal{K}_R)$ to describe the insider information known to corrupt senders and receivers about the routing permutation π. We define three natural notions of insider information:

1. *Every player knows who it is talking to.* The first natural notion is to assume that each sender or receiver knows whom the player itself is talking to, but it is not aware who others are talking to. By corrupting some senders and receivers, the adversary should not learn more about the routing permutation π beyond what the corrupt senders and receivers know. In other words, the part of the permutation π containing "corrupt $\rightarrow *$" and "$* \rightarrow$ corrupt" is leaked. More formally, we can define leakage as below:

$$\mathsf{Leak}^{\mathrm{SR}}(\pi, \mathcal{K}_S, \mathcal{K}_R) := \{\forall i \in \mathcal{K}_S : (i, \pi(i))\} \cup \{\forall i \in \mathcal{K}_R : (\pi^{-1}(i), i)\}$$

2. *Every sender knows who it is talking to.* Another natural notion is when a sender knows which receiver it is talking to, but a receiver may not know who it is receiving from. By corrupting a subset of the senders and receivers, the adversary should not learn more than what those corrupt players know. In other words, the "corrupt $\rightarrow *$" part of the permutation π is leaked. More formally, we can formally define leakage as below:

$$\mathsf{Leak}^{\mathrm{S}}(\pi, \mathcal{K}_S, \mathcal{K}_R) := \{\forall i \in \mathcal{K}_S : (i, \pi(i))\}$$

3. *Inherent leakage.* The least possible leakage is when only the "corrupt $\rightarrow$ corrupt" part of the permutation π is leaked. Note that this leakage is inherent because a corrupt sender can always encrypt multiple random messages in the same time slot, and observe whether any corrupt receiver received this message. In the minimum, inherent leakage scenario, we require that only this is leaked about the permutation π and nothing else. More formally, we can formally define leakage as below:

$$\mathsf{Leak}^{\min}(\pi, \mathcal{K}_S, \mathcal{K}_R) := \{\forall i \in \mathcal{K}_S \cap \pi^{-1}(\mathcal{K}_R) : (i, \pi(i))\}$$

Remark 2. Note that even in the minimum, inherent leakage scenario, knowing the leaked information $\mathsf{Leak}^{\min}(\pi, \mathcal{K}_S, \mathcal{K}_R) := \{\forall i \in \mathcal{K}_S \cap \pi^{-1}(\mathcal{K}_R) : (i, \pi(i))\}$ as well as $\mathcal{K}_R$, one can efficiently compute the set $\mathcal{K}_R \cap \pi(\mathcal{H}_S)$. Therefore, during the encryption phase, by learning $\{\forall i \in \mathcal{K}_R \cap \pi(\mathcal{H}_S) : (i, x_j) \text{ for } j = \pi^{-1}(i)\}$, i.e., the set of leaked messages received by corrupt receivers from honest senders, the simulator Sim does not learn anything extra about the routing permutation π beyond what it already learned earlier in the experiment, that is, $\mathsf{Leak}^{\min}(\pi, \mathcal{K}_S, \mathcal{K}_R)$.

Definition 1 (NIAR simulation security)**.** We define simulation security of a NIAR scheme as below depending on which leakage function is used in the $\mathsf{Ideal}^{\mathcal{A},\mathsf{Sim}}$ experiment:

1. We say that a Non-Interactive Anonymous Routing (NIAR) scheme is *SIM-secure* iff the following holds when using $\mathsf{Leak} := \mathsf{Leak}^{\mathrm{SR}}$ in the $\mathsf{Ideal}^{\mathcal{A},\mathsf{Sim}}$ experiment: there exists a p.p.t. simulator Sim such that for any non-uniform p.p.t. adversary $\mathcal{A}$, $\mathcal{A}$'s view in $\mathsf{Real}^{\mathcal{A}}(1^\kappa)$ and $\mathsf{Ideal}^{\mathcal{A},\mathsf{Sim}}(1^\kappa)$ are computationally indistinguishable.
2. We say that a NIAR scheme is *SIM-secure with receiver-insider protection*, iff the above holds when using $\mathsf{Leak} := \mathsf{Leak}^{\mathrm{S}}$ in the $\mathsf{Ideal}^{\mathcal{A},\mathsf{Sim}}$ experiment.
3. We say that a NIAR scheme is *SIM-secure with full insider protection*, iff the above holds when using $\mathsf{Leak} := \mathsf{Leak}^{\min}$ in the $\mathsf{Ideal}^{\mathcal{A},\mathsf{Sim}}$ experiment.

2.3 Indistinguishability-Based Security

We give an alternative, indistinguishability-based security notion. Similar to our simulation security notions, we consider three variants, basic IND-security, IND-security with receiver-insider protection, and IND-security with full insider protection. We will prove that each indistinguishability-based notion is in fact equivalent to the corresponding simulation-based notion in Definition 1. The indistinguishability-based notions are easier to work with in our security proofs so they can serve as convenient operational notions.

Consider the following experiment $\mathsf{NIAR\text{-}Expt}^{b,\mathcal{A}}(1^\kappa)$ parametrized by a bit $b \in \{0,1\}$:

- $n, \mathcal{K}_S, \mathcal{K}_R, \pi^{(0)}, \pi^{(1)} \leftarrow \mathcal{A}(1^\kappa)$
- $(\{\mathsf{ek}_i\}_{i\in[n]}, \{\mathsf{rk}_i\}_{i\in[n]}, \mathsf{tk}) \leftarrow \mathbf{Setup}(1^\kappa, n, \pi^{(b)})$
- For $t = 1, 2, \ldots$:
  - if $t = 1$ then $\{x^{(0)}_{i,t}\}_{i\in\mathcal{H}_S}, \{x^{(1)}_{i,t}\}_{i\in\mathcal{H}_S} \leftarrow \mathcal{A}(\mathsf{tk}, \{\mathsf{ek}_i\}_{i\in\mathcal{K}_S}, \{\mathsf{rk}_i\}_{i\in\mathcal{K}_R})$;
    else $\{x^{(0)}_{i,t}\}_{i\in\mathcal{H}_S}, \{x^{(1)}_{i,t}\}_{i\in\mathcal{H}_S} \leftarrow \mathcal{A}(\{\mathsf{ct}_{i,t-1}\}_{i\in\mathcal{H}_S})$;
  - for $i \in \mathcal{H}_S$, $\mathsf{ct}_{i,t} \leftarrow \mathbf{Enc}(\mathsf{ek}_i, x^{(b)}_{i,t}, t)$

We say that $\mathcal{A}$ is *admissible* iff with probability 1, it guarantees that

1. $\mathsf{Leak}(\pi^{(0)}, \mathcal{K}_S, \mathcal{K}_R) = \mathsf{Leak}(\pi^{(1)}, \mathcal{K}_S, \mathcal{K}_R)$ where Leak can be $\mathsf{Leak}^{\mathrm{SR}}$, $\mathsf{Leak}^{\mathrm{S}}$, or $\mathsf{Leak}^{\min}$; and
2. for any $i \in \mathcal{K}_R \cap \pi^{(0)}(\mathcal{H}_S) = \mathcal{K}_R \cap \pi^{(1)}(\mathcal{H}_S)$, $x^{(0)}_{j_0,t} = x^{(1)}_{j_1,t}$ where for $b \in \{0,1\}$, $j_b := (\pi^{(b)})^{-1}(i)$. In other words, here we require that in the two alternate worlds $b = 0$ or 1, every corrupt receiver receiving from an honest sender must receive the same message.

Definition 2 (NIAR indistinguishable security)**.** We define indistinguishability-based security notions for NIAR, depending on which leakage function is adopted:

1. We say that a NIAR scheme is *IND-secure* iff when $\mathsf{Leak} := \mathsf{Leak}^{\mathrm{SR}}$ is used in the above experiment, the following holds: for any non-uniform p.p.t. admissible $\mathcal{A}$, its views in the experiments $\mathsf{NIAR\text{-}Expt}^{0,\mathcal{A}}(1^\kappa)$ and $\mathsf{NIAR\text{-}Expt}^{1,\mathcal{A}}(1^\kappa)$ are computationally indistinguishable.
2. We say that a NIAR scheme is *IND-secure with receiver-insider protection*, iff the above holds when $\mathsf{Leak} := \mathsf{Leak}^{\mathrm{S}}$ is used in the above experiment.
3. We say that a NIAR scheme is *IND-secure with full insider protection*, iff the above holds when $\mathsf{Leak} := \mathsf{Leak}^{\min}$ is used in the above experiment.

2.4 Equivalence between Indistinguishability- and Simulation-Based Notions

We now prove the equivalence of each simulation-based notion and the corresponding indistinguishability-based notion.

Lemma 2.1. *A NIAR scheme is SIM-secure iff it is IND-secure. Similarly, a NIAR scheme is SIM-secure with receiver-insider protection (or full insider protection, respectively) iff it is IND-secure with receiver-insider protection (of full insider protection, respectively).*

Proof. The proof is deferred to Section A in the appendices. □

3 Informal Overview of Our Construction

We now give an informal overview of our constructions.

3.1 Notations and Building Block

We will concretely instantiate a scheme using a cyclic group $\mathbb{G}$ of prime order q. Therefore, we introduce some notations for group elements and group operations.

Group notation and implicit notation for group exponentiation. Throughout the paper, we use the notation $[\![x]\!]$ to denote a group element $g^x \in \mathbb{G}$ where $g \in \mathbb{G}$ is the generator of an appropriate cyclic group of prime order q where $x \in \mathbb{Z}_q$. Similarly, $[\![\mathbf{x}]\!]$ denotes a vector of group elements where $\mathbf{x} \in \mathbb{Z}_q^{|\mathbf{x}|}$ is the exponent vector. If we know $[\![x]\!] \in \mathbb{G}$ and $y \in \mathbb{Z}_p$, we can compute $[\![xy]\!] \in \mathbb{G}$. Therefore, whenever an algorithm needs to compute $[\![xy]\!]$, it only needs to know one of the exponents x or y. The same implicit notation is used for vectors too.

Correlated pseudorandom functions. We will need a building block which we call a correlated pseudorandom function, denoted CPRF. A CPRF scheme has the following possibly randomized algorithms:

- $(K_1, K_2, \ldots, K_n) \leftarrow \mathbf{Gen}(1^\kappa, n, q)$: takes a security parameter 1^κ and the number of users n, some prime q, and outputs the user secret key K_i for each $i \in [n]$.
- $v \leftarrow \mathbf{Eval}(K_i, x)$: given a user secret key K_i and an input $x \in \{0,1\}^\kappa$, output an evaluation result $v \in \mathbb{Z}_q$.

For correctness, we require that the following always holds if $\{K_i\}_{i\in[n]}$ is in the support of $\mathbf{Gen}(1^\kappa, n, q)$:

$$\sum_{i\in[n]} \mathsf{CPRF}.\mathbf{Eval}(K_i, x) = 0 \mod q \tag{1}$$

For security, we require that even when a subset of the keys $\mathcal{K} \subset [n]$ can be corrupted by the adversary, it must be that for every fresh x, all honest evaluations $\{\mathsf{CPRF}.\mathbf{Eval}(K_i, x)_{i\notin\mathcal{K}}\}$ are computationally indistinguishable from random terms subject to the constraint $\sum_{i\notin\mathcal{K}} \mathsf{CPRF}.\mathbf{Eval}(K_i, x) = -\sum_{i\in\mathcal{K}} \mathsf{CPRF}.\mathbf{Eval}(K_i, x) \mod q$ — note that the adversary can compute the right-hand-side of the equation.

Intuitively, such a correlated PRF guarantees that even when some players' keys can be corrupt, honest players' evaluations for any fresh input x must appear random, except that they are subject to the constraint in Equation 1. A couple earlier works [ABG19, BIK+17] showed how to construct such a CPRF from ordinary PRFs. We will present more formal definitions and construction in Section 4.3.

3.2 A Simple, Function-Revealing MCFE Scheme for Selection

We consider MCFE for "selection", which can be viewed as a special case of inner-product computation. An MCFE-for-selection scheme has four possibly randomized algorithms (**Setup**, **KGen**, **Enc**, **Dec**) — in our definition below, we allow each client to encrypt a vector $\mathbf{x}_{i,t} \in \{0,1\}^m$ of length m, and the selection vector $\mathbf{y} \in \{0,1\}^{mn}$ selects exactly one coordinate from one client's plaintext vector[5]:

- The $\mathbf{Setup}(1^\kappa, m, n)$ algorithm[6] outputs a secret key for each of the n clients where the i-th client's key is denoted ek_i, and a master public- and secret-key pair $(\mathsf{mpk}, \mathsf{msk})$.
- The $\mathbf{KGen}(\mathsf{mpk}, \mathsf{msk}, \mathbf{y})$ algorithm takes the master public-key mpk and the master secret-key msk, and outputs a functional key $\mathsf{sk}_\mathbf{y}$ for the selection vector $\mathbf{y} \in \{0,1\}^{mn}$. It is promised that the input $\mathbf{y}$ has only one coordinate set to 1, and the rest are set to 0.
- The $\mathbf{Enc}(\mathsf{mpk}, \mathsf{ek}_i, \mathbf{x}_{i,t}, t)$ algorithm lets client $i \in [n]$ use its secret key ek_i to encrypt a plaintext $\mathbf{x}_{i,t} \in \{0,1\}^m$ for the time step t.
- Finally, given the n ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_n$ collected from all clients pertaining to the same time step, as well as the functional key $\mathsf{sk}_\mathbf{y}$, one can call $\mathbf{Dec}(\mathsf{mpk}, \mathsf{sk}_\mathbf{y}, \{\mathsf{ct}_i\}_{i\in[n]})$ to evaluate the selection outcome $\langle \mathbf{x}, \mathbf{y} \rangle$ where $\mathbf{x}$ denotes the concatenation of the plaintexts encrypted under $\mathsf{ct}_1, \ldots, \mathsf{ct}_n$.

If we did not care about function privacy, it turns out that we can construct a very simple MCFE-for-selection scheme as follows. Basically, for each of the n clients, there is a separate symmetric-key encryption instance. During **Setup**, client i obtains the secret keys $\mathsf{sk}_{i,1}, \ldots, \mathsf{sk}_{i,m}$ corresponding to m independent encryption instances. For client i to encrypt a message of m bits during some time step t, it simply encrypts each bit $j \in [m]$ using $\mathsf{sk}_{i,j}$, and output the union of the ciphertexts. To generate a functional key for selection vector $\mathbf{y}$ that selects the j-th coordinate of the client i's message, simply output $(\mathbf{y}, \mathsf{sk}_{i,j})$, and decryption can be completed, i.e., using $\mathsf{sk}_{i,j}$ to decrypt the coordinate in the ciphertext that is being selected.

3.3 Preparing the MCFE Scheme for Function Privacy Upgrade

The next challenge is how to upgrade the above MCFE-for-selection scheme to have function privacy. Function privacy in inner-product functional encryption (FE) was first studied by Shen, Shi, and Waters [SSW09], who considered single-input FE and a weaker notion of function privacy than what we will need. Subsequent works have generalized and improved the techniques of Shen, Shi, and Waters [SSW09] to achieve stronger notions of function privacy [Lin17], and have extended the techniques to a multi-input FE context [ACF+18].

Our function privacy upgrade techniques are inspired by these earlier works [SSW09, Lin17, ACF+18], but we need non-trivial new techniques to make it work in our context. Specifically, previous function privacy techniques assume the encryptor to be trusted, and thus they are not directly applicable to the MCFE setting in which some of the clients may be corrupted, and their secret keys become known to the adversary.

To enable the function-private upgrade, let us first understand where the above MCFE-for-selection scheme in Section 3.2 leaks information about the selection vector $\mathbf{y}$. First, the scheme

[5] Our scheme can support the case where each coordinate of the plaintext vector $\mathbf{x}_{i,t}$ comes from a polynomially sized space, but we simply assume each coordinate is a bit for simplicity.

[6] In our subsequent formal sections, for notational reasons needed to make our presentation formal, we shall separate the **Setup** algorithm into a parameter generation algorithm **Gen** and a **Setup** algorithm, respectively.

blantantly embeds the selection vector $\mathbf{y}$ in cleartext in the functional key $\mathsf{sk}_{\mathbf{y}}$. Second, the decryption process itself also reveals $\mathbf{y}$ because decryption works on only the coordinate being selected. To fix the above problems, we would like to first modify the idea in Section 3.2 to satisfy the following two requirements:

1. We change the decryption process such that decryption involves all coordinates, and not just the coordinate being selected.
2. Further, we want to randomize the partial decryption outcome corresponding to every client such that from the partial decryptions alone, one cannot tell which coordinate is being selected.

We can instantiate an MCFE-for-selection scheme satisfying the above requirements in a cyclic group $\mathbb{G}$ of prime order q. The resulting scheme is still *function-revealing* — at this point, we have merely "prepared" the scheme for the function privacy upgrade described later in Section 3.4. We describe this scheme below where we use $\mathsf{CPRF}(K_i, t)$ as an abbreviation for $\mathsf{CPRF}.\mathbf{Eval}(K_i, t)$:

MCFE: function-revealing MCFE for selection, w/ randomized partial decryptions

$$\mathsf{msk} = \{\mathbf{S}_i, a_i\}_{i\in[n]}, \quad \mathsf{ek}_i = (K_i, \mathbf{S}_i, a_i) \text{ where each } \mathbf{S}_i \in \mathbb{Z}_q^{m\times 2}$$

Ciphertext for t where each $\mathbf{x}_{i,t} \in \{0,1\}^m$

$$\forall i \in [n]: \quad \Big([\![\mathbf{x}_{i,t} + \mathbf{S}_i\mathbf{r}_i]\!], \ [\![\mathbf{r}_i]\!], \ [\![\mathsf{CPRF}(K_i, t) + a_i\mu_i]\!], \ [\![\mu_i]\!] \Big)$$

where $\mathbf{r}_i$ and μ_i are chosen at random

Functional key for $\mathbf{y} := (\mathbf{y}_1, \ldots, \mathbf{y}_n)$ where each $\mathbf{y}_i \in \{0,1\}^m$

$$\forall i \in [n]: \quad \Big(\mathbf{y}_i, \ -\mathbf{S}_i^\top \mathbf{y}_i, \ \rho, \ -\rho a_i \Big)$$

where ρ is chosen at random

Henceforth, we will name $[\![\mathbf{c}_{i,1}]\!] := [\![\mathbf{x}_{i,t} + \mathbf{S}_i\mathbf{r}_i]\!]$, $[\![\mathbf{c}_{i,2}]\!] := [\![r_i]\!]$, and $[\![\widetilde{\mathbf{c}}]\!] := [\![\mathsf{CPRF}(K_i, t) + a_i\mu_i, \mu_i]\!]$. Additionally, let $\mathbf{k}_{i,1} := \mathbf{y}_i$, $\mathbf{k}_{i,2} := \mathbf{S}_i^\top \mathbf{y}_i$, and $\widetilde{\mathbf{k}}_i := (\rho, -\rho a_i)$.

For the above scheme to be a correct function-revealing MCFE-for-selection, we only need the first two terms of the ciphertext and functional keys, i.e., $([\![\mathbf{c}_{i,1}]\!], [\![\mathbf{c}_{i,2}]\!])$ and $(\mathbf{k}_{i,1}, \mathbf{k}_{i,2})$. Essentially, these terms can be viewed as a concrete instantiation of the ideas mentioned in Section 3.2: the j-th row of $\mathbf{S}_i$ is used to encrypt the j-th coordinate of $x_{i,t}$; further, to compute a functional key for $\mathbf{y}$ which is selecting the j-th coordinate of the i-th client's message, simply output $\mathbf{y}$ and the j-th row of $\mathbf{S}_i$ (which is equal to $\mathbf{S}_i^\top \mathbf{y}_i$). Security of the encryption follows from the Decisional Linear assumption. The extra terms in the ciphertexts and functional keys, denoted $\widetilde{\mathbf{c}}_i$ and $\widetilde{\mathbf{k}}_i$ are randomizing terms added to satisfy the aforementioned randomized partial decryption requirement as we explain below.

We now explain how decryption works. Given a ciphertext vector for n all clients $[\![\mathbf{c}]\!] := \big(([\![\mathbf{c}_{1,1}]\!], [\![\mathbf{c}_{1,2}]\!], [\![\widetilde{\mathbf{c}}_1]\!]), \ \ldots, \ ([\![\mathbf{c}_{n,1}]\!], \ [\![\mathbf{c}_{n,2}]\!], \ [\![\widetilde{\mathbf{c}}_n]\!])\big)$, and a key vector $\mathbf{k} := \big((\mathbf{k}_{1,1}, \ \mathbf{k}_{1,2}, \ \widetilde{\mathbf{k}}_1), \ \ldots, \ (\mathbf{k}_{n,1}, \mathbf{k}_{n,2}, \widetilde{\mathbf{k}}_n)\big)$, decryption computes the "inner-product-in-the-exponent" of the ciphertext vector and the token vector, i.e.,

$$[\![\langle \mathbf{c}, \mathbf{k}\rangle]\!] = \prod_{i\in[n]} \Big([\![\langle \mathbf{c}_{i,1}, \mathbf{k}_{i,1}\rangle]\!] \cdot [\![\langle \mathbf{c}_{i,2}, \mathbf{k}_{i,2}\rangle]\!] \cdot [\![\langle \widetilde{\mathbf{c}}_i, \widetilde{\mathbf{k}}_i\rangle]\!] \Big).$$

Finally, we output the discrete logarithm of the above expression as the decrypted message[7].

[7] Note that because decryption involves computing a discrete logarithm, we require the plaintext space to be small.

The decryption can alternatively be viewed as computing the partial decryption of each client and then multiplying the partial decryptions together. Henceforth, let $\mathsf{MCFE.Dec}^i$ denote the function that computes the partial decryption corresponding to client i, and let $p_{i,t}$ denote the i-th partial decryption:

$$p_{i,t} := \mathsf{MCFE.Dec}^i(\mathsf{sk}_i, \mathsf{ct}_{i,t}) = \left([\![\langle \mathbf{c}_{i,1}, \mathbf{k}_{i,1} \rangle]\!] \cdot [\![\langle \mathbf{c}_{i,2}, \mathbf{c}_{i,2} \rangle]\!] \cdot [\![\langle \widetilde{\mathbf{c}}_i, \widetilde{\mathbf{k}}_i \rangle]\!] \right),$$

and then multiplying all the randomized partially decrypted results. Note that the partial decryption function $\mathsf{MCFE.Dec}^i(\mathsf{sk}_i, \mathsf{ct}_{i,t})$ also evaluates an inner-product in the exponent. One can verify the following: let $\mathbf{x}_{i,t} := (x_{i,1,t}, \ldots, x_{i,m,t})$ be the plaintext message encrypted under $\mathsf{ct}_{i,t}$, we have that

$$p_{i,t} = \begin{cases} [\![\mathsf{CPRF}(K_i, t) \cdot \rho]\!] & \text{if client } i\text{'s vector is not being selected} \\ [\![x_{i,j,t} + \mathsf{CPRF}(K_i, t) \cdot \rho]\!] & \text{if the } j\text{-th coordinate of the } i\text{-th client is being selected} \end{cases}$$

Thus, the above decryption indeed involves all coordinates, and moreover, the partial decryption results $\{p_{i,t}\}_{i \in [n]}$ are randomized due to the use of the CPRF.

Remark 3 (Technical condition needed for the function privacy upgrade)**.** Informally speaking, we want the following (necessary but not sufficient) condition to hold for our function privacy upgrade to work. Let $\mathcal{H} \subseteq [n]$ be the set of honest clients. Assume that the Decisional Linear assumption holds. We want that even after having seen the public key, honest ciphertexts in all time steps other than t, honest ciphertexts in time step t, i.e., $\{\mathsf{ct}_{i,t}\}_{i \in \mathcal{H}}$, as well as $[\![\rho]\!]$ for a fresh random $\rho \in \mathbb{Z}_q$, the terms $\{[\![\mathsf{CPRF}(K_i, t) \cdot \rho]\!]\}_{i \in \mathcal{H}}$ must be computationally indistinguishable from random, except that their product is equal to some fixed term known to the adversary. This condition is needed in the proof of Lemma 5.6 which is arguably the most subtle lemma in the function privacy upgrade proof.

3.4 Function Privacy Upgrade

Since we do not want the functional key to leak the selection vector $\mathbf{y}$, we want to encrypt the functional key $\mathsf{sk}_\mathbf{y}$; but how can we use the encrypted $\mathsf{sk}_\mathbf{y}$ for correct decryption? Inspired by earlier works [Lin17, ACF+18], our idea is to adopt n instances (single-input) functional encryption henceforth denoted FE, such that the i-th client obtains the master secret key of the i-th instance, henceforth denoted msk_i. During **KGen**, we encrypt the the i-th coordinate of $\mathsf{sk}_\mathbf{y}$ using the i-th FE, and let the result be $\overline{\mathsf{sk}}_i$. To encrypt its message $\mathbf{x}_{i,t}$, the i-th client first encrypts $\mathbf{x}_{i,t}$ using the MCFE-for-selection scheme and obtains the ciphertext $\mathsf{ct}_{i,t}$; then it calls $\overline{\mathsf{ct}}_{i,t} := \mathsf{FE.KGen}(\mathsf{msk}_i, f^{\mathsf{ct}_{i,t}})$ to transform $\mathsf{ct}_{i,t}$ into an FE token for the function $f^{\mathsf{ct}_{i,t}}(\star) := \mathsf{MCFE.Dec}^i(\star, \mathsf{ct}_{i,t})$. Recall that $\mathsf{MCFE.Dec}^i$ computes the MCFE's partial decryption for the i-th coordinate. In this way, an evaluator can invoke $\mathsf{FE.Dec}$ on the pair $\overline{\mathsf{ct}}_{i,t}$ and $\overline{\mathsf{sk}}_i$ to obtain the i-th partial decryption.

To make this idea work, in fact, we do not even need FE for general circuits. Recall that in our MCFE-for-selection scheme above, each partial decryption function $\mathsf{MCFE.Dec}^i$ computes an inner-product in the exponent. We therefore only need an FE scheme capable of computing an inner-product in the exponent. Several earlier works [ABCP16, ALS16, Wee16] showed how to construct inner-product function encryption based on the DDH assumption. By slightly modifying these constructions, one can construct an FE scheme for evaluating "inner-product-in-the-exponent" as long as the Decisional Linear assumption holds in certain bilinear groups. For completeness, we shall present this special FE scheme for computing "inner-product-in-the-exponent" in Section D in the appendices.

From weak to full function privacy. Although intuitively, the above idea seems like it should work, it turns out for technical reasons, we can only prove that it satisfies a weak form of function privacy henceforth called *weak function hiding*. We defer its detailed technical definition to Section 5.1. Fortunately, we can borrow a two-slot trick from various prior works on Functional Encryption [SSW09, BJK15, ACF+18] and Indistinguishability Obfuscation [LV16, Lin17], and upgrade a weakly function-hiding MCFE-for-selection scheme to a fully function-hiding one. At a very high level, to achieve this, instead of having each client $i \in [n]$ encrypt its plaintext $\mathbf{x}_i \in \{0,1\}^m$, we have each client i encrypt the expanded vector $(\mathbf{x}_i, \mathbf{0})$ instead where $\mathbf{0}$ is also of length m. Similarly, the selection vector's length will need to be doubled accordingly too, i.e., to compute a functional key for $\mathbf{y} = (\mathbf{y}_1, \ldots, \mathbf{y}_n)$ where each $\mathbf{y}_i \in \{0,1\}^m$, we instead compute a functional key for the expanded vector $((\mathbf{y}_1, \mathbf{0}), \ldots, (\mathbf{y}_n, \mathbf{0}))$.

By expanding the plaintext and selection vectors, we gain some spare slots which can serve as "wiggle room" during our security proofs. This way, in our security proofs, we can make incremental modifications in every step of the hybrid sequence and make progress with the proof.

Our exposition above is geared towards understandability and is sometimes informal. The actual details and proofs are somewhat more involved and we refer the reader to Section 5 for a formal exposition.

Summarizing the above, we can construct an MCFE-for-selection scheme with (full) function privacy, henceforth denoted $\mathsf{MCFE}^{\mathrm{ffh}}$, presented more formally below. In the description below, MCFE is the aforementioned function-revealing MCFE for selection, augmented to have randomized partial decryptions; FE is a single-input functional encryption scheme for computing inner-products in exponents, formally defined in Section 4.2.

$\mathsf{MCFE}^{\mathrm{ffh}}$: **function-hiding MCFE for selection**

- **Gen**(1^κ): Sample a suitable prime q, and generate a suitable bilinear group of order q, with the pairing function $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Let $H : \{0,1\}^* \to \mathbb{G}_1$ be a random oracle. The public parameter pp contains the prime q, and the description of the bilinear group; the parameters pp' contains a description of $\mathbb{G}_1$, its order q, and a description of H.
- **Setup**(pp, m, n): Call $(\mathsf{mpk}', \mathsf{msk}', \{\mathsf{ek}'_i\}_{i\in[n]}) \leftarrow \mathsf{MCFE}.\mathbf{Setup}(\mathsf{pp}', 2m, n)$. For $i \in [n]$, call $(\mathsf{mpk}_i, \mathsf{msk}_i) \leftarrow \mathsf{FE}.\mathbf{Setup}(\mathsf{pp}, 2m+2)$. Output:

$$\mathsf{mpk} := (\mathsf{pp}, \mathsf{mpk}', \{\mathsf{mpk}_i\}_{i\in[n]}), \quad \mathsf{msk} := (\mathsf{msk}', \{\mathsf{msk}_i, \mathsf{ek}_i\}_{i\in[n]}),$$
$$\forall i \in [n] : \mathsf{ek}_i := (\mathsf{msk}_i, \mathsf{ek}'_i)$$

- **Enc**$(\mathsf{mpk}, \mathsf{ek}_i, \mathbf{x}, t)$:
  1. Let $\mathsf{ct} := \mathsf{MCFE}.\mathbf{Enc}(\mathsf{mpk}', \mathsf{ek}'_i, (\mathbf{x}, \mathbf{0}), t) \in \mathbb{G}_1^{2m+2}$.
  2. Let $\overline{\mathsf{ct}} := \mathsf{FE}.\mathbf{KGen}(\mathsf{msk}_i, \mathsf{ct})$.
  3. Output $\mathsf{CT} := (\mathsf{ct}, \overline{\mathsf{ct}})$.
- **KGen**$(\mathsf{mpk}, \mathsf{msk}, \mathbf{y})$:
  1. Parse $\mathbf{y} := (\mathbf{y}_1, \ldots, \mathbf{y}_n)$ where each $\mathbf{y}_i \in \{0,1\}^m$.
  2. Let $\widetilde{\mathbf{y}} = ((\mathbf{y}_1, \mathbf{0}), \ldots, (\mathbf{y}_n, \mathbf{0})) \in \{0,1\}^{2mn}$.
  3. Call $(\mathbf{k}_1, \ldots, \mathbf{k}_n) := \mathsf{MCFE}.\mathbf{KGen}(\mathsf{mpk}', \mathsf{msk}', \widetilde{\mathbf{y}})$ where each $\mathbf{k}_i \in \mathbb{Z}_q^{2m+2}$ for $i \in [n]$.

4. For $i \in [n]$, call $\overline{\mathbf{k}}_i := \mathsf{FE}.\mathbf{Enc}(\mathsf{mpk}_i, \mathbf{k}_i)$.
5. Output $\mathsf{sk}_{\mathbf{y}} := (\overline{\mathbf{k}}_1, \ldots, \overline{\mathbf{k}}_n)$.

- $\mathbf{Dec}(\mathsf{mpk}, \mathsf{sk}_{\mathbf{y}}, \{\mathsf{CT}_i\}_{i\in[n]})$: Parse each $\mathsf{CT}_i := (\mathsf{ct}_i, \overline{\mathsf{ct}}_i)$. Parse $\mathsf{sk}_{\mathbf{y}} := (\overline{\mathbf{k}}_1, \ldots, \overline{\mathbf{k}}_n)$. For $i \in [n]$, call $v_i := \mathsf{FE}.\mathbf{Dec}(\overline{\mathsf{ct}}_i, \mathsf{ct}_i, \overline{\mathbf{k}}_i)$. Output $\log(\prod_{i=1}^{n} v_i)$.

Our $\mathsf{MCFE}^{\mathsf{ffh}}$ scheme will be at the core of both our NIAR schemes, the one with receiver-insider protection, and the one with full insider protection.

Proof roadmap for $\mathsf{MCFE}^{\mathsf{ffh}}$. To prove our $\mathsf{MCFE}^{\mathsf{ffh}}$ scheme secure, a critical stepping stone is to prove that it satisfies a weak notion of function privacy — afterwards we can rely on known techniques [LV16, Lin17, BJK15, ACF+18] to prove full function privacy. Roughly speaking, we say that an MCFE scheme for selection satisfies weak function privacy iff no p.p.t. *admissible* adversary $\mathcal{A}$ can distinguish two worlds indexed by $b \in \{0, 1\}$. In world b:

- the adversary $\mathcal{A}$ first specifies a set of corrupt clients, and obtains the public parameters as well as secret keys for corrupt clients;
- the adversary $\mathcal{A}$ now submits multiple **KGen** queries, each time specifying $\mathbf{y}^{(0)}$ and $\mathbf{y}^{(1)}$; and the challenger computes and returns tokens for $\mathbf{y}^{(b)}$;
- then $\mathcal{A}$ makes **Enc** queries for each time step t by specifying $\{\mathbf{x}_{i,t}^{(0)}\}_{i\in\mathcal{H}}$ and $\{\mathbf{x}_{i,t}^{(1)}\}_{i\in\mathcal{H}}$ where $\mathcal{H} \subseteq [n]$ denotes the set of honest clients; and the challenger computes and returns encryptions for $\{\mathbf{x}_{i,t}^{(b)}\}_{i\in\mathcal{H}}$.

Moreover, an *admissible* adversary $\mathcal{A}$ must respect the following constraints:

1. for $i \in [n]\backslash\mathcal{H}$, $\mathbf{y}_i^{(0)} = \mathbf{y}_i^{(1)}$.
2. for any $\{\mathbf{x}_{i,t}^{(0)}, \mathbf{x}_{i,t}^{(1)}\}_{i\in\mathcal{H}}$ submitted in an **Enc** query,

$$\left\langle (\mathbf{x}_{i,t}^{(0)})_{i\in\mathcal{H}}, (\mathbf{y}_i^{(0)})_{i\in\mathcal{H}} \right\rangle = \left\langle (\mathbf{x}_{i,t}^{(1)})_{i\in\mathcal{H}}, (\mathbf{y}_i^{(0)})_{i\in\mathcal{H}} \right\rangle = \left\langle (\mathbf{x}_{i,t}^{(1)})_{i\in\mathcal{H}}, (\mathbf{y}_i^{(1)})_{i\in\mathcal{H}} \right\rangle$$

In our proof, we start from world 0, and through a sequence of hybrids, we switch to world 1; and every adjacent pair of hybrids are computationally indistinguishable. First, we use the function-revealing privacy of the underlying MCFE scheme to switch the encrypted vectors from $\{\mathbf{x}_{i,t}^{(0)}\}_{i\in\mathcal{H}}$ to $\{\mathbf{x}_{i,t}^{(1)}\}_{i\in\mathcal{H}}$ — this step is possible due to the aforementioned admissibility rule $\mathcal{A}$ must respect. Next, we want to switch to using $\mathbf{y}^{(1)}$ in each **KGen** query. To accomplish this, we rely on a hybrid sequence over the multiple **KGen** queries one by one. Essentially, in the ℓ-th hybrid, the first ℓ **KGen** queries are answered with $\mathbf{y}^{(1)}$, and the rest of the **KGen** queries are answered using $\mathbf{y}^{(0)}$. It suffices to argue that the $(\ell-1)$-th hybrid and the ℓ-th hybrid are computationally indistinguishable, and this turns out to be the most subtle step in our proof. To achieve this, let us consider the following modification of the $(\ell-1)$-th hybrid. Henceforth the ℓ-th **KGen** query is also called the *challenge* **KGen** query, and the two vectors submitted during this query are denoted $\mathbf{y}_*^{(0)}$ and $\mathbf{y}_*^{(1)}$ respectively:

1. During the ℓ-th **KGen** quer, for computing components of the key corresponding to honest players, the challenger switches the $\mathsf{FE}.\mathbf{Enc}$ inside the challenge **KGen** query to a simulated encryption which does not use the underlying MCFE's functional key as input. Corrupt players' key components are still computed honestly.

Correspondingly, in every time step, the challenger answers **Enc** queries by calling the a simulated $\mathsf{FE}.\mathbf{KGen}$ for every honest client i's ciphertext component: the i-th simulated $\mathsf{FE}.\mathbf{KGen}$ embeds the i-th partial decryption when paired with the challenge key for $\mathbf{y}_*^{(0)}$ of the underlying MCFE scheme. This step relies on the 1-SEL-SIM security of the single-input FE scheme (defined in Section 4.2).

2. At this moment, due to the randomizing terms, and the aforementioned admissibility rule, we argue that during each **Enc** query, instead of encoding in the simulated $\mathsf{FE}.\mathbf{KGen}$ the partial decryptions when paired with the challenge key for $\mathbf{y}_*^{(0)}$ of the underlying MCFE scheme, we could use $\mathbf{y}_*^{(1)}$ instead. This step is more involved and requires the technical condition in Remark 3.

From this point onwards, we can use a symmetric argument to switch all the way to the aforementioned ℓ-th hybrid, in which the first ℓ **KGen** queries are answered with $\mathbf{y}^{(1)}$, and the remaining answered with $\mathbf{y}^{(0)}$. We defer the detailed proof to the subsequent formal sections.

3.5 Constructing NIAR with Receiver-Insider Protection

Construction. With a function-hiding MCFE-for-selection scheme henceforth denoted $\mathsf{MCFE}^{\mathsf{ffh}}$, we can construct a NIAR scheme in a natural fashion informally described below:

- **Setup**: The idea is to use the $\mathsf{MCFE}^{\mathsf{ffh}}$ scheme to generate functional keys for n selection vectors, denoted $\mathsf{tk}_1, \ldots, \mathsf{tk}_n$, where tk_i is for selecting the message received by receiver $i \in [n]$. The colletion $\{\mathsf{tk}_i\}_{i\in[n]}$ is given to the router as the token. The $\mathsf{MCFE}^{\mathsf{ffh}}$ also generates n secret encryption keys denoted $\{\mathsf{ek}_i\}_{i\in[n]}$, one for each sender. Finally, the setup procedure generates n symmetric encryption keys, one for each sender-receiver pair.
- **Enc**: During each time step t, to encrypt a message $x_{i,t}$, the i-th sender first encrypts $x_{i,t}$ with its symmetric key shared with its receiver — let $c_{i,t}$ denote the resulting ciphertext. Now, call $\mathsf{ct}_{i,t} := \mathsf{MCFE}^{\mathsf{ffh}}.\mathbf{Enc}(\mathsf{mpk}, \mathsf{ek}_i, c_{i,t})$ to further encrypt $c_{i,t}$ and obtain a final ciphertext $\mathsf{ct}_{i,t}$. Here, we abuse notation slightly and use $\mathsf{MCFE}^{\mathsf{ffh}}.\mathbf{Enc}(\mathsf{mpk}, \mathsf{ek}_i, c_{i,t})$ to mean encrypting $c_{i,t}$ bit by bit with the $\mathsf{MCFE}^{\mathsf{ffh}}$ scheme.
- **Rte**: Using the n functional keys $\{\mathsf{tk}_i\}_{i\in[n]}$, a router can call $\mathsf{MCFE}^{\mathsf{ffh}}.\mathbf{Dec}$ to obtain the n inner ciphertexts encrypted under the symmetric keys, and send the corresponding inner ciphertext to each receiver.
- **Dec**: Finally, each receiver uses its symmetric key to decrypt the final outcome.

Proof roadmap. In Section 6, we shall prove that as long as $\mathsf{MCFE}^{\mathsf{ffh}}$ satisfies function-hiding security and the symmetric-key encryption scheme employed is secure, then, the above NIAR construction satisfies *receiver-insider protection*. To prove this, we use the indistinguishability security notion for NIAR, which is shown to be equivalent to the simulation-based notion. Rouhgly speaking, the indistinguishability game for NIAR, denoted $\mathsf{NIAR\text{-}Expt}^b$ is indexed by a bit $b \in \{0, 1\}$: imagine the adversary $\mathcal{A}$ chooses two permutations $\pi^{(0)}$ and $\pi^{(1)}$, and specifies two sets of messages $\{x_{i,t}^{(0)}\}_{i\in\mathcal{H}_S}$ and $\{x_{i,t}^{(1)}\}_{i\in\mathcal{H}_S}$ to query in each time step t. The challenger gives $\mathcal{A}$ a token for $\pi^{(b)}$, and ciphertexts for $\{x_{i,t}^{(b)}\}_{i\in\mathcal{H}_S}$ in each time step t. An *admissible* adversary must choose the permutations and messages such that the leakage in the two worlds are the same, where the leakage contains the corrupt-to-$*$ part of the permutation and the messages received by corrupt receivers in every time step. We want to prove that any efficient, admissible $\mathcal{A}$ cannot distinguish whether it is playing $\mathsf{NIAR\text{-}Expt}^0$ or $\mathsf{NIAR\text{-}Expt}^1$.

To prove this, we first modify $\mathsf{NIAR\text{-}Expt}^b$ slightly to obtain a hybrid Hyb^b for $b \in \{0,1\}$: in Hyb^b, we replace the inner symmetric-key encryption from honest senders to honest receivers with simulated ciphertexts. We can easily show that Hyb^b is computationally indistinguishable from $\mathsf{NIAR\text{-}Expt}^b$ by reducing to the security of the symmetric encryption scheme.

To complete the proof, the more challenging step is to show that Hyb^0 is computationally indistinguishable from Hyb^1 for any efficient, admissible adversary $\mathcal{A}$. Here, we want to leverage $\mathcal{A}$ to create an efficient reduction $\mathcal{B}$ that breaks the function-hiding security of the underlying $\mathsf{MCFE}^{\mathsf{ffh}}$ scheme. The subtlety is to make sure that $\mathcal{B}$ indeed respects the $\mathsf{MCFE}^{\mathsf{ffh}}$'s admissibility rules. In our formal proofs in Section 6, we fix the randomness ψ consumed by the SE instances corresponding to each receiver in the set $\pi^{(0)}(\mathcal{H}_S) = \pi^{(1)}(\mathcal{H}_S)$, and prove that the two experiments are indistinguishability for every choice of fixed ψ. We then define the reduction $\mathcal{B}$ in a natural manner, and make a careful argument that if $\mathcal{A}$ satisfies the NIAR game's admissibility rule (for the receiver-insider protection notion), then $\mathcal{B}$ will indeed respect the admissibility rules of the underlying $\mathsf{MCFE}^{\mathsf{ffh}}$.

We defer the formal description and proofs to Section 6.

3.6 Achieving Full Insider Protection

To upgrade our NIAR scheme to have full insider protection turns out to be more involved. As explained earlier in Section 1.3.2, for such a scheme to work, *all* the transformed ciphertexts output by **Rte** must change when a single sender's input ciphertext changes.

Construction (sketch). To accomplish this, we leverage a indistinguishability obfuscator for probabilistic circuits (piO) whose existence is implied by sub-exponentially secure indistinguishability obfuscation and sub-exponentially secure one-way functions [CLTV15].

- **Setup**: during the trusted setup, each receiver i receives the secret key of a PKE scheme (with special properties mentioned later); and each sender receives the encryption key generated by an $\mathsf{MCFE}^{\mathsf{ffh}}$ scheme.

  The router's token tk is a piO which encodes the $\mathsf{MCFE}^{\mathsf{ffh}}$ scheme's functional keys for all n selection vectors. Inside the piO, the following probabilistic program is evaluated:

  1. first, use the $\mathsf{MCFE}^{\mathsf{ffh}}$ functional keys to decrypt the messages that each receiver should receive;
  2. next, encrypt the messages under each receiver's respective public keys, and output the encrypted ciphertexts — note that the encryption scheme is randomized.

- **Enc**: in every time step, senders encrypt their messages using $\mathsf{MCFE}^{\mathsf{ffh}}$.

- **Rte**: in each time step, the router applies its token tk, which is an obfuscated program, to the n ciphertexts collected from senders. The outcome will be n transformed ciphertexts.

- **Dec**: When a receiver receives a transformed ciphertext, it simply uses its secret key to decrypt it.

Observe that in this construction, indeed, if a single sender's input ciphertext changes, *all* transformed ciphertexts output by the **Rte** procedure will change.

Proof roadmap and subtleties. We encounter some more subtleties when we attempt to prove the above construction secure. First, it turns out that for technical reasons, to prove the above

scheme secure, we need the public-key encryption (PKE) scheme used by the piO to reencrypt output messages to satisfy a special property: the PKE must be a special trapdoor mode in which encryptions of 0 and 1 are identically distributed. Obviously, the trapdoor mode loses information and cannot support correct decryption. In fact, in the real world, we will never use the trapdoor mode — it is used only inside our security proofs. We henceforth call a PKE scheme with this special property a perfectly hiding trapdoor encryption (tPKE). Such a tPKE scheme can be constructed assuming DDH [CLTV15].

Informally, our proof strategy is the following: First, we modify the real-world experiment (in which $\pi^{(0)}$ and $x_{i,t}^{(0)}$ are used), and switch the tPKE instances corresponding to honest receivers' to use a trapdoor setup. This step can be reduced to the tPKE's security, since the adversary does not have the tPKE instances' secret keys corresponding to honest receivers. Next, we modify the obfuscated program to *no longer use the functional keys corresponding to the honest receivers*; instead, the obfuscated program will simply output encryptions of 0 under the trapdoor public keys for honest receivers. For corrupt receivers, the obfuscated program still behaves like the real world: use the $\mathsf{MCFE}^{\mathrm{ffh}}$ scheme's **Dec** procedure to decrypt the messages they ought to receive, and output encryptions of these messages under each corrupt receiver's public keys, respectively. This step relies on the security of the piO and the fact that the modified program is "distributionally equivalent" to the original program. At this moment, the obfuscated program *no longer uses the functional keys for honest receivers*, and only at this point can we rely on the $\mathsf{MCFE}^{\mathrm{ffh}}$'s security and switch from using $\pi^{(0)}$ in the setup and encrypting $x_{i,t}^{(0)}$ to using $\pi^{(1)}$ in the setup and encrypting $x_{i,t}^{(1)}$. The remaining hybrids are symmetric to the above, such that eventually we arrive at an experiment that is the same as the real-world experiment in which $\pi^{(1)}$ and $x_{i,t}^{(1)}$ are used by the challenger.

Notice that in our construction, we use the piO to obfuscate the $\mathsf{MCFE}^{\mathrm{ffh}}$ scheme's **Dec** procedure using all n functional keys. One natural question is why we did not directly use the piO to obfuscate a program that calls the **Rte** procedure of our earlier NIAR scheme (with receiver-insider protection) and then encrypts the n outcomes using n instances of tPKE. It turns out that our proof strategy would not have worked for the latter, exactly because in our proofs, we needed an intermediate hybrid to completely stop using functional keys for honest receivers — intuitively, this is how we can prove the privacy of messages received by honest receivers. This explains why in our construction and proofs, we need to open up the NIAR scheme and directly manipulate the functional keys of the underlying $\mathsf{MCFE}^{\mathrm{ffh}}$.

3.7 Achieving Fault Tolerance

So far in our constructions, unless all senders send their encryption during a certain time step, the router would fail to perform the **Rte** operation. Such a scheme relies all senders to be online all the time, and thus is not fault-tolerant.

We modify our earlier NIAR abstraction to one that is fault-tolerant. The idea is to let **Enc** and **Rte** take an extra parameter $\mathcal{O} \subseteq [n]$ which denotes the set of senders that remain online. Additionally, **Rte** now takes in only ciphertexts from those in $\mathcal{O}$. In fact, our fault-tolerant NIAR abstraction can be viewed as a generalization of the non-fault-tolerant version.

To achieve fault tolerance, we observe that the underlying CPRF construction we use has a nice fault-tolerance property. In fact, we can modify the CPRF's evaluation function to take in $\mathcal{O}$, such that the following is satisfied:

$$\forall t \in \mathbb{N}: \quad \sum_{i \in \mathcal{O}} \mathsf{CPRF}.\mathbf{Eval}(K_i, t, \mathcal{O}) = 0$$

This way, for every receiver whose corresponding sender is in the online set $\mathcal{O}$, the router can correctly perform the $\mathsf{MCFE}^{\mathsf{ffh}}$'s decryption procedure using only ciphertexts from those in $\mathcal{O}$. Note that the recent elegant work of Bonawitz et al. [BIK+17] also made a similar observation of the fault-tolerance of the CPRF, and leveraged it to enable fault-tolerant, privacy-preserving federated learning — this is not explicitly stated in their paper but implicit in their constructions.

If a receiver i's corresponding sender is no longer online, however, then the $\mathsf{MCFE}^{\mathsf{ffh}}$'s decryption procedure will output an inner ciphertext of $\mathbf{0}$ for receiver i. Since receiver i cannot decrypt the $\mathbf{0}$ ciphertext using its symmetric key, it will simply output $\perp$ — this is inevitable since the corresponding sender is no longer around. However, the router can also observe that receiver i received an inner-ciphertext $\mathbf{0}$. In this way, if the adversary is able to drop the senders one by one and check which receiver starts to receive an inner ciphertext of $\mathbf{0}$, it can learn the receivers paired up with crashed senders. In our subsequent formal sections, we shall prove that in this fault-tolerant NIAR scheme, indeed the adversary can learn only the (corrupt+crashed)-to-$*$ part of the permutation π, as well as the messages received by corrupt receivers every time step, and nothing else.

Finally, using techniques similar to those sketched in Section 3.6, we can upgrade the security of the above fault-tolerant scheme to full insider protection, i.e., only the (corrupt + crashed)-to-corrupt part of the permutation is leaked as well as the messages received by corrupt receivers, but nothing else. As mentioned earlier, this leakage is inherent and unavoidable for any fault-tolerant NIAR scheme, since the adversary can always make the senders crash one by one and check which corrupt receiver now starts to receive $\perp$.

Of course, the above description is a gross simplification omitting various subtleties both in definitions and constructions. We refer the reader to Sections 8 and B for the detailed definitions, constructions, and proofs.

4 Preliminaries

Throughout the paper, we will use κ to denote the security parameter. We will use boldface letters such as $\mathbf{x}$ to denote vectors, and use normal-font letters such as x to denote scalars. Given two vectors $\mathbf{x} \in \mathbb{Z}_q^\ell$ and $\mathbf{y} \in \mathbb{Z}_q^\ell$ of the same dimension ℓ, we use the notation $\langle \mathbf{x}, \mathbf{y} \rangle \in \mathbb{Z}_q$ to denote their inner-product where the operations are performed modulo q.

4.1 Group Notations and the Decisional Linear Assumption

Notation for group elements. Given a cyclic group $\mathbb{G}$ of prime order q, and a generator $g \in \mathbb{G}$, we use the notation $[\![x]\!]$ to denote $g^x \in \mathbb{G}$ where $x \in \mathbb{Z}_q$. For a vector $\mathbf{x} := (x_1, x_2, \ldots, x_\ell) \in \mathbb{Z}_q^\ell$, the notation $[\![\mathbf{x}]\!]$ means the vector of group elements $(g^{x_1}, g^{x_2}, \ldots, g^{x_\ell})$.

Sometimes we will employ a bilinear group $(\mathbb{G}, \mathbb{G}_T)$ of prime order q with a pairing operator $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, and a random generator $g \in \mathbb{G}$. In this case, the notation $[\![x]\!]$ means g^x, and the notation $[\![x]\!]_T$ means $e(g, g)^x$. The notations for vectors are similarly defined.

Implicit notation for group operations. If a party knows $[\![x]\!] \in \mathbb{G}$ and $y \in \mathbb{Z}_q$, it is able to compute $[\![xy]\!] := [\![x]\!]^y$. Therefore, without risking ambiguity, often times when we write "compute $[\![xy]\!]$" when describing an algorithm, we actually mean compute the group exponentiation $[\![x]\!]^y$, or conversely, $[\![y]\!]^x$ if the party knows $[\![y]\!] \in \mathbb{G}$ and $x \in \mathbb{Z}_q$. The same rule also extends to vectors as well as bilinear groups.

The Decisional Linear assumption. We say that the Decisional Linear assumption holds for the group generator $\mathcal{G}$, iff the following two experiments are computationally indistinguishable:

1. Sample $\mathsf{pp} := (q, \mathbb{G}, g) \xleftarrow{\$} \mathcal{G}(1^\kappa)$ where $\mathbb{G}$ is a cyclic group of order q with a random generator $g = [\![1]\!]$. Sample random $\beta, \gamma, u, v \xleftarrow{\$} \mathbb{Z}_q$. Output the tuple $(\mathsf{pp}, [\![1]\!], [\![\beta]\!], [\![\gamma]\!], [\![u]\!], [\![\beta v]\!], [\![\gamma(u+v)]\!])$.
2. Sample $\mathsf{pp} := (q, \mathbb{G}, g) \xleftarrow{\$} \mathcal{G}(1^\kappa)$ where $\mathbb{G}$ is a cyclic group of order q with a random generator $g = [\![1]\!]$. Sample random $\beta, \gamma, u, v, z \xleftarrow{\$} \mathbb{Z}_q$. Output the tuple $(\mathsf{pp}, [\![1]\!], [\![\beta]\!], [\![\gamma]\!], [\![u]\!], [\![\beta v]\!], [\![z]\!])$.

Without risk of ambiguity, we sometimes say that the Decisional Linear assumption holds for the group $\mathbb{G}$ where $\mathbb{G}$ is the group sampled by the group generator $\mathcal{G}$.

4.2 1-SEL-SIM-Secure, Single-Input Functional Encryption

Syntax. We now define the syntax of a single-input functional encryption (FE) scheme capable of evaluating "inner-product in the exponent"; and if the evaluation outcome is promised to be from a small space, one can then compute the discrete logarithm efficiently and output the inner product that is encoded in the exponent. Such an FE scheme consists of the following, possibly randomized algorithms:

- $\mathsf{pp} \leftarrow \mathbf{Gen}(1^\kappa)$: takes in a security parameter κ and samples public parameters pp. We will assume that pp contains the description of a bilinear group $(\mathbb{G}, \mathbb{G}_T)$ of prime order q, a random generator $g \in \mathbb{G}$, and the description of the pairing operator $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$.
- $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathbf{Setup}(\mathsf{pp}, m)$: takes in the public parameters pp and the dimension m of the plaintext vector, outputs a public key mpk and a master secret key msk.
- $\mathsf{sk}_{\mathbf{y}} \leftarrow \mathbf{KGen}(\mathsf{msk}, [\![\mathbf{y}]\!])$: takes in the master secret key msk, and a vector of group elements $[\![\mathbf{y}]\!] \in \mathbb{G}^m$ which represents the group encoding of the vector $\mathbf{y} \in \mathbb{Z}_q^m$, outputs a functional (secret) key $\mathsf{sk}_{\mathbf{y}}$.
- $\mathsf{ct} \leftarrow \mathbf{Enc}(\mathsf{mpk}, [\![\mathbf{x}]\!])$: takes in the master public key mpk, a plaintext vector $[\![\mathbf{x}]\!] \in \mathbb{G}^m$ represented in group encoding, and outputs a ciphertext ct.
- $[\![v]\!]_T \leftarrow \mathbf{Dec}(\mathsf{sk}_{\mathbf{y}}, [\![\mathbf{y}]\!], \mathsf{ct})$: takes in the functional key $\mathsf{sk}_{\mathbf{y}}$, the group encoding $[\![\mathbf{y}]\!]$ of $\mathbf{y}$, and a ciphertext ct, outputs a decrypted outcome $[\![v]\!]_T$.

Correctness. Correctness requires that for any $\kappa, m \in \mathbb{N}, \mathbf{x}, \mathbf{y} \in \mathbb{Z}_q^m$, the following holds with probability 1: let $\mathsf{pp} \leftarrow \mathbf{Gen}(1^\kappa)$, $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathbf{Setup}(\mathsf{pp}, m)$, $\mathsf{sk}_{\mathbf{y}} \leftarrow \mathbf{KGen}(\mathsf{msk}, [\![\mathbf{y}]\!])$, $\mathsf{ct} \leftarrow \mathbf{Enc}(\mathsf{mpk}, [\![\mathbf{x}]\!])$, $[\![v]\!]_T \leftarrow \mathbf{Dec}(\mathsf{sk}_{\mathbf{y}}, [\![\mathbf{y}]\!], \mathsf{ct})$, then, it must be that $v := \langle \mathbf{x}, \mathbf{y} \rangle$.

Security definitions. We now define one-selective, simulation (1-SEL-SIM) security for FE.

Definition 3 (1-SEL-SIM-secure FE)**.** A single-input functional encryption scheme $\mathcal{FE}$ for computing inner product in the exponent is said to be 1-SEL-SIM-secure iff there exists a p.p.t. simulator $(\widetilde{\mathbf{Setup}}, \widetilde{\mathbf{Enc}}, \widetilde{\mathbf{KGen}})$ such that for any non-uniform p.p.t. adversary $\mathcal{A}$ and every $\kappa \in \mathbb{N}$, for every m, the following two distributions are computationally indistinguishable:

Experiment $\mathsf{Real}(1^\kappa, m)$:

$\mathsf{pp} := (q, e, \mathbb{G}, \mathbb{G}_T, g) \xleftarrow{\$} \mathcal{G}(1^\kappa)$

$(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathbf{Setup}(\mathsf{pp}, m)$

$\mathbf{x} \leftarrow \mathcal{A}(1^\kappa, \mathsf{mpk})$ where $\mathbf{x} \in \mathbb{Z}_q^m$

$\mathsf{ct} \leftarrow \mathbf{Enc}(\mathsf{mpk}, [\![\mathbf{x}]\!])$

$b \leftarrow \mathcal{A}^{\mathbf{KGen}(\mathsf{msk}, \cdot)}(\mathsf{ct})$

Output b

Experiment $\mathsf{Ideal}(1^\kappa, m)$:

$\mathsf{pp} := (q, e, \mathbb{G}, \mathbb{G}_T, g) \xleftarrow{\$} \mathcal{G}(1^\kappa)$

$(\mathsf{mpk}, \mathsf{msk}) \leftarrow \widetilde{\mathbf{Setup}}(\mathsf{pp}, m)$

$\mathbf{x} \leftarrow \mathcal{A}(1^\kappa, \mathsf{mpk})$ where $\mathbf{x} \in \mathbb{Z}_q^m$

$\mathsf{ct} \leftarrow \widetilde{\mathbf{Enc}}(\mathsf{msk})$

$b \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(\mathsf{ct})$

Output b

The oracle $\mathcal{O}(\cdot)$ in the Ideal experiment is defined as below: upon receiving a **KGen** query for $[\![\mathbf{y}]\!] \in \mathbb{G}^m$, return $\widetilde{\mathbf{KGen}}(\mathsf{msk}, [\![\mathbf{y}]\!], [\![\langle \mathbf{x}, \mathbf{y} \rangle]\!])$ to $\mathcal{A}$.

Theorem 4.1 (1-SEL-SIM-secure FE for evaluating "inner-product in the exponent")**.** *Suppose that the Decisional Linear assumption holds in appropriate cyclic groups of prime order. Then, there exists a 1-SEL-SIM-secure, single-input FE scheme for evaluating "inner-product in the exponent". Furthermore, the master public key* mpk *and master secret key* msk *have size at most* $O(m) \cdot \mathsf{poly}(\kappa)$*; each functional key has size* $\mathsf{poly}(\kappa)$*; and each ciphertext has size* $O(m) \cdot \mathsf{poly}(\kappa)$*.*

Proof. We adopt a variantion of the construction suggested in several prior works [Wee16, ACF+18, ABCP16, ALS16]. For completeness, in Section D in the appendices, we present the detailed construction and proofs. □

4.3 Correlated Pseudorandom Function

A correlated pseudorandom function family consists of the following randomized algorithms:

- $(K_1, K_2, \ldots, K_n) \leftarrow \mathbf{Gen}(1^\kappa, n, q)$: takes a security parameter 1^κ and the number of users n, some prime q, and outputs the user secret key K_i for each $i \in [n]$.
- $v \leftarrow \mathbf{Eval}(K_i, x)$: given a user secret key K_i and an input $x \in \{0,1\}^\kappa$, output an evaluation result $v \in \mathbb{Z}_q$.

Correctness. For correctness, we require that for any $\kappa \in \mathbb{N}$, any $(K_1, \ldots, K_n)$ in the support of $\mathbf{Gen}(1^\kappa)$, any input $x \in \{0,1\}^\kappa$, the following holds:

$$\sum_{i \in [n]} \mathsf{CPRF}.\mathbf{Eval}(K_i, x) = 0 \mod q$$

Correlated pseudorandomness. We require that for any non-uniform p.p.t. adversary $\mathcal{A}$ who is allowed corrupt $f \leq n-2$ users and obtain their user secret keys, for any subset U of at most $n-f-1$ honest users, for any input x, the evaluations $\{\mathsf{CPRF}.\mathbf{Eval}(K_i, x)\}_{i \in U}$ are computationally indistinguishable from random values, as long as the adversary has not made a query on the input x.

More formally, correlated pseudorandomness is defined as below. Consider a game denoted $\mathsf{CPRF\text{-}Expt}^b(1^\kappa, n, q)$ between $\mathcal{A}$ and a challenger $\mathcal{C}$, parametrized by a bit $b \in \{0,1\}$.

- **Setup.** $\mathcal{A}$ submits a set of corrupt nodes $\mathcal{K} \subset [n]$ of size at most $n-2$. Henceforth let $\mathcal{H} := [n] \backslash \mathcal{K}$. Now, $\mathcal{C}$ runs the honest $(K_1, \ldots, K_n) := \mathsf{CPRF}.\mathbf{Gen}(1^\kappa, n, q)$ algorithm, and gives $\{K_i\}_{i \in \mathcal{K}}$ to $\mathcal{A}$.

- **Queries.** $\mathcal{A}$ can adaptively make queries: for each query, $\mathcal{A}$ submits an input x. If $b = 0$, the challenger $\mathcal{C}$ chooses random $\{v_i\}_{i\in\mathcal{H}} \xleftarrow{\$} \mathbb{Z}_q^{|\mathcal{H}|}$ subject to the condition that $\sum_{i\in\mathcal{H}} v_i + \sum_{j\in\mathcal{K}} \mathsf{CPRF}.\mathbf{Eval}(K_j, x) = 0$, and returns $\{v_i\}_{i\in\mathcal{H}}$ to $\mathcal{A}$. Else if $b = 1$, the challenger gives $\{\mathsf{CPRF}.\mathbf{Eval}(K_i, x)\}_{i\in\mathcal{H}}$ to $\mathcal{A}$.

We say that a correlated pseudorandom function family CPRF satisfies correlated pseudorandomness, iff for any n and q, any non-uniform p.p.t. adversary $\mathcal{A}$'s views in $\mathsf{CPRF\text{-}Expt}^0(1^\kappa, n, q)$ and $\mathsf{CPRF\text{-}Expt}^1(1^\kappa, n, q)$ are computationally indistinguishable.

Construction. We can construct a correlated PRF scheme as follows — the idea was described by Abadalla, Benhamouda, and Gay [ABG19].

- $\mathbf{Gen}(1^\kappa, n, q)$: generate random secret keys $\{k_{ij}\}_{i\in[n],j\in[n],i<j}$ for some pseudorandom function family that maps κ-bit strings to $\mathbb{Z}_q$. For $i > j$, let $k_{ij} := k_{ji}$.

  Let $K_i := (i, \{k_{ij}\}_{j\neq i})$ for $i \in [n]$, and output $K_1, \ldots, K_n$.
- $\mathbf{Eval}(K_i, x)$: Output $\sum_{j\neq i}(-1)^{j<i} \cdot \mathsf{PRF}_{k_{ij}}(x)$ — here we use PRF to denote the evaluation function of the pseudorandom function family.

Theorem 4.2 (Correlated pseudorandom function family)**.** *Suppose that the pseudorandom function family employed is secure. Then, the above construction is correct and satisfies correlated pseudorandomness as defined above.*

Proof. First, we need to verify correctness. $\sum_{i\in[n]} \mathsf{CPRF}.\mathbf{Eval}(K_i, x) = \sum_i \sum_{j\neq i}(-1)^{j<i} \cdot \mathsf{PRF}_{k_{ij}}(x)$. In the above summation, we can pair up the terms $(-1)^{j<i} \cdot \mathsf{PRF}_{k_{ij}}(x)$ and $(-1)^{i<j} \cdot \mathsf{PRF}_{k_{ij}}(x)$ for every pair i and j where $i < j$; and the sum of the two terms is 0. Therefore, the entire sum is 0 too.

Next, we prove correlated pseudo-randomness. We can consider a hybrid experiment in which every honest user's PRF is replaced with a random function in the challenger's answers to the adversary. Specifically, we will use the random function $\mathsf{RO}_{ij}(\cdot)$ to replace $\mathsf{PRF}_{k_{ij}}(\cdot)$. This hybrid experiment is computationally indistinguishable from the experiment $\mathsf{CPRF\text{-}Expt}^1$ due to the security of the PRF.

Without loss of generality, we can number the honest users in $\mathcal{H}$ as $i_1 < i_2, \ldots, < i_h$ where $h := |\mathcal{H}|$. We claim that the hybrid experiment is identically distributed as $\mathsf{CPRF\text{-}Expt}^0$. To show this, it suffices to check that except for the last honest user i_h, when answering queries for any other honest user $j \in \{i_1, \ldots, i_{h-1}\}$, there is some independent randomness contained in the answer. Fix some query x, the answer for user i_1 contains the fresh random term $\mathsf{RO}_{i_1,i_2}(x)$, the answer for user i_2 contains the fresh random term $\mathsf{RO}_{i_2,i_3}(x)$, and so on. □

4.4 Indistinguishability Obfuscation for Probabilistic Circuits

We define the notion of indistinguishability obfuscation for probabilistic circuits, henceforth denoted piO, first proposed by Canetti et al. [CLTV15]. In our paper, we only need piO for probabilistic circuits that are functionally equivalent (i.e., result in the same distribution on any input) whereas the original Canetti et al. work [CLTV15] defined piO for a broader class of samplers.

Let $\mathcal{C} := \{\mathcal{C}_\kappa\}_{\kappa\in\mathbb{N}}$ be an ensemble of possibly randomized circuit families. Let $\mathcal{D}(1^\kappa) : \emptyset \to \mathcal{C}_\kappa \times \mathcal{C}_\kappa \times \{0,1\}^*$ be a randomized algorithm that samples a circuit from $\mathcal{C}_\kappa$. We say that $\mathcal{D}$ is a perfectly indistinguishable sampler over $\mathcal{C}$, iff for any $\kappa \in \mathbb{N}$, with probability 1 over the choice of $(C_0, C_1, z) \xleftarrow{\$} \mathcal{D}(1^\kappa)$, it must be that for any input x, the tuples $(C_0(x), C_0, C_1, z)$ and $(C_1(x), C_0, C_1, z)$ are identically distributed.

Definition 4 (piO for perfectly indistinguishable circuit samplers)**.** A uniform p.p.t. machine piO is an indistinguishability obfuscator for perfectly indistinguishable samplers over the possibly randomized circuit ensemble $\mathcal{C} = \{\mathcal{C}_\kappa\}_{\kappa\in\mathbb{N}}$, iff the following two conditions hold:

- **Correctness.** For any non-uniform p.p.t. *admissible* adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$, such that for any $\kappa \in \mathbb{N}$, for every probabilistic circuit $C \in \mathcal{C}_\kappa$,

$$\left|\Pr\left[\mathcal{A}^{C(\cdot)}(1^\kappa) = 1\right] - \Pr\left[\overline{C} \leftarrow \mathsf{piO}(1^\kappa, C) : \quad \mathcal{A}^{\overline{C}(\cdot)}(1^\kappa) = 1\right]\right| \leq \mathsf{negl}(\kappa)$$

  In the above, we say that $\mathcal{A}$ is *admissible* if it never asks the same query to its oracle more than once. We stress while the circuit C samples fresh random coins for answering each query, the output of the (randomized) obfuscator piO, namely $\overline{C}$, is a *deterministic* circuit.

- **Security.** For any perfectly indistinguishable sampler $\mathcal{D}$ over $\mathcal{C}$, for every non-uniform p.p.t. adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that

$$\begin{aligned}&\Big|\Pr\left[(C_0, C_1, z)\xleftarrow{\$}\mathcal{D}(1^\kappa) : \ \mathcal{A}(1^\kappa, C_0, C_1, z, \mathsf{piO}(1^\kappa, C_0)) = 1\right]\\ &- \Pr\left[(C_0, C_1, z)\xleftarrow{\$}\mathcal{D}(1^\kappa) : \ \mathcal{A}(1^\kappa, C_0, C_1, z, \mathsf{piO}(1^\kappa, C_1)) = 1\right]\Big| \leq \mathsf{negl}(\kappa)\end{aligned}$$

Theorem 4.3 (piO for perfectly indistinguishable samplers)**.** *Suppose that sub-exponentially secure indistinguishability obfuscator for deterministic circuits and sub-exponentially secure one-way functions exist. Then, there exists an indistinguishability obfuscator for perfectly indistinguishable samplers over general circuit families.*

4.5 Perfectly Hiding Trapdoor Encryption

A perfectly hiding trapdoor encryption scheme is a public-key encryption system with an additional trapdoor mode. In the normal mode, we run the normal key generation, and the encrypted ciphertexts encode actual information and can be correctly decrypted using an appropriate secret key. In the trapdoor mode, we run a trapdoor key generation procedure, and the encrypted ciphertexts lose all information about the plaintext message, and there is no secret key for decryption. More formally, a perfectly hiding trapdoor encryption scheme is defined as follows.

Definition 5 (Perfectly hiding trapdoor encryption scheme)**.** We say that $\mathsf{tPKE} := (\mathbf{Gen}, \mathbf{Enc}, \mathbf{Dec}, \widetilde{\mathbf{Gen}})$ is a perfectly hiding trapdoor encryption scheme, iff $(\mathbf{Gen}, \mathbf{Enc}, \mathbf{Dec})$ is a semantically secure public-key encryption scheme, and the trapdoor key generation procedure $\widetilde{\mathbf{Gen}}$ satisfies the following additional properties:

1. *Indistinguishability of the modes*: the following ensembles are computationally indistinguishable:

$$\left\{(\mathsf{pk}, \mathsf{sk})\xleftarrow{\$}\mathbf{Gen}(1^\kappa) : \mathsf{pk}\right\}_\kappa \stackrel{c}{\equiv} \left\{\widetilde{\mathsf{pk}}\xleftarrow{\$}\widetilde{\mathbf{Gen}}(1^\kappa) : \widetilde{\mathsf{pk}}\right\}_\kappa$$

2. *Perfectly hiding encryption under trapdoor mode*: the following ensembles are identically distributed:

$$\left\{\widetilde{\mathsf{pk}}\xleftarrow{\$}\widetilde{\mathbf{Gen}}(1^\kappa) : \mathbf{Enc}(\widetilde{\mathsf{pk}}, 0)\right\}_\kappa = \left\{\widetilde{\mathsf{pk}}\xleftarrow{\$}\widetilde{\mathbf{Gen}}(1^\kappa) : \mathbf{Enc}(\widetilde{\mathsf{pk}}, 1)\right\}_\kappa$$

Henceforth whenever we write $\mathbf{Enc}(\mathsf{pk}, x)$ for a string $x \in \{0,1\}^*$ that is multiple bits, we mean encrypt the plaintext x bit by bit.

Theorem 4.4 (Perfectly hiding trapdoor encryption)**.** *Assume that the Decisional Linear assumption holds in appropriate groups, then there exists a perfectly hiding trapdoor encryption scheme.*

Proof. The idea is similar to Canetti et al. [CLTV15] but we can replace their DDH assumption with Decisional Linear easily. Recall that Boneh, Boyen, and Shacham [BBS04] described the Linear Encryption system using the Decisional Linear assumption. It is not hard to see that the Linear Encryption system is randomizable. Therefore, we can build an encryption scheme by publishing an encryption of 0 and an encryption of 1 in the public key using the Linear Enryption scheme, and encrypting either 0 or 1 could be accomplished by rerandomizing the corresponding ciphertext in the public key. In the trapdoor mode, we could publish a fake public key that contain two encryptions of 0. Therefore, in the trapdoor mode, encryptions of 0 and 1 are identically distributed. □

5 Multi-Client Functional Encryption for Selection

5.1 Definition

Henceforth, we use m to denote the number of coorindates encrypted by each client, and use n to denote the number of clients. In a Multi-Client Functional Encryption (MCFE) scheme for selection, in every time step, each client $i \in [n]$ encrypts a vector $\mathbf{x}_i \in \mathbb{Z}_q^m$ using its private key ek_i. An authority holding a master secret key msk can generate a functional key $\mathsf{sk}_{\mathbf{y}}$ for a vector $\mathbf{y} \in \{0,1\}^{mn} = (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n)$ where each $\mathbf{y}_i \in \{0,1\}^m$. It is promised that at most one coordinate in $\mathbf{y}$ is set to 1 and all other coordinates are set to 0 — henceforth such vectors are called *selection vectors*. One can now apply the functional key $\mathsf{sk}_{\mathbf{y}}$ to the collection of all n clients' ciphertexts belonging to the same time step, and an evaluation procedure gives the result $\langle \mathbf{x}, \mathbf{y} \rangle$ where $\mathbf{x} := (\mathbf{x}_1, \ldots, \mathbf{x}_n)$. In other words, the functional key allows one to select one coordinate in one client's plaintext vector.

A multi-client functional encryption (MCFE) scheme for selection consists of the following algorithms:

- $\mathsf{pp} \leftarrow \mathbf{Gen}(1^\kappa)$: the parameter generation algorithm **Gen** takes in a security parameter κ and chooses parameters pp — we will assume that pp contains a prime number $q \in \mathbb{N}$ and the description of a suitable cyclic group $\mathbb{G}$ of prime order q.
- $(\mathsf{mpk}, \mathsf{msk}, \{\mathsf{ek}_i\}_{i\in[n]}) \leftarrow \mathbf{Setup}(\mathsf{pp}, m, n)$: takes in the parameters q, $\mathbb{G}$, m, and n, and outputs a public key mpk, a master secret key msk, and n user secret keys needed for encryption, denoted $\mathsf{ek}_1, \ldots, \mathsf{ek}_n$, respectively. Without loss of generality, henceforth we may assume that mpk encodes pp so we need not write the parameters pp explicitly below.
- $\mathsf{sk}_{\mathbf{y}} \leftarrow \mathbf{KGen}(\mathsf{mpk}, \mathsf{msk}, \mathbf{y})$: takes in the public key mpk, the master secret key msk, and a vector $\mathbf{y} \in \{0,1\}^{mn}$ which is promised to be a selection vector, and outputs a functional secret key $\mathsf{sk}_{\mathbf{y}}$.
- $\mathsf{ct}_{i,t} \leftarrow \mathbf{Enc}(\mathsf{mpk}, \mathsf{ek}_i, \mathbf{x}_i, t)$: takes in the public key mpk, a user secret key ek_i, a plaintext $\mathbf{x}_i \in \mathbb{Z}_q^m$, and a time step label $t \in \mathbb{N}$, outputs a ciphertext $\mathsf{ct}_{i,t}$.
- $v \leftarrow \mathbf{Dec}(\mathsf{mpk}, \mathsf{sk}_{\mathbf{y}}, \{\mathsf{ct}_{i,t}\}_{i\in[n]})$: takes in the public key mpk, the functional secret key $\mathsf{sk}_{\mathbf{y}}$, and a collection of ciphertexts $\{\mathsf{ct}_{i,t}\}_{i\in[n]}$, outputs a decrypted outcome $v \in \mathbb{Z}_q$.

Correctness. For correctness, we require that the following holds with probability 1 for any $\kappa, m, n \in \mathbb{N}$, any $\mathbf{y} \in \{0,1\}^{mn}$ in which only one coordinate is 1 and all other coordinates are 0, and any $\mathbf{x} := (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ from an appropriate domain determined by pp, and any $t \in \mathbb{N}$: let

$\mathsf{pp} \leftarrow \mathbf{Gen}(1^\kappa)$, let $(\mathsf{mpk}, \mathsf{msk}, \{\mathsf{ek}_i\}_{i\in[n]}) \leftarrow \mathbf{Setup}(\mathsf{pp}, m, n)$, let $\mathsf{sk}_{\mathbf{y}} \leftarrow \mathbf{KGen}(\mathsf{mpk}, \mathsf{msk}, \mathbf{y})$, let $\mathsf{ct}_{i,t} \leftarrow \mathbf{Enc}(\mathsf{mpk}, \mathsf{ek}_i, \mathbf{x}_i, t)$ for $i \in [n]$, and let $v \leftarrow \mathbf{Dec}(\mathsf{mpk}, \mathsf{sk}_{\mathbf{y}}, \{\mathsf{ct}_{i,t}\}_{i\in[n]})$, it must be that $v = \langle \mathbf{x}, \mathbf{y} \rangle$.

IND-security for MCFE. Consider the following experiment $\mathsf{Expt}^b(1^\kappa)$ between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

Experiment $\mathsf{Expt}^b(1^\kappa)$:

- **Setup.** $\mathcal{A}(1^\kappa)$ outputs a set of corrupted parties $\mathcal{K} \subset [n]$, as well as the parameters m and n to the challenger $\mathcal{C}$. The challenger $\mathcal{C}$ runs $\mathsf{pp} \leftarrow \mathbf{Gen}(1^\kappa)$, and $(\mathsf{mpk}, \mathsf{msk}, \{\mathsf{ek}_i\}_{i\in[n]}) \leftarrow \mathbf{Setup}(\mathsf{pp}, m, n)$; it gives mpk and $\{\mathsf{ek}_i\}_{i\in\mathcal{K}}$ to $\mathcal{A}$. Initially, $t = 1$.
- **Query.** The adversary can make the following types of queries; moreover, all **KGen** queries must be made *before* any **Enc** query:
  - **KGen queries.** Whenever the adversary $\mathcal{A}$ makes a **KGen** query with a selection vector $\mathbf{y} \in \{0,1\}^{mn}$, the challenger $\mathcal{C}$ calls $\mathsf{sk}_{\mathbf{y}} := \mathbf{KGen}(\mathsf{mpk}, \mathsf{msk}, \mathbf{y})$ and returns $\mathsf{sk}_{\mathbf{y}}$ to $\mathcal{A}$;
  - **Enc queries.** Whenever $\mathcal{A}$ makes an **Enc** query with the plaintext vectors $\{\mathbf{x}_{i,t}^{(0)}, \mathbf{x}_{i,t}^{(1)}\}_{i\in\mathcal{H}}$ where $\mathcal{H} := [n]\backslash\mathcal{K}$, the challenger $\mathcal{C}$ calls $\mathsf{ct}_i := \mathbf{Enc}(\mathsf{mpk}, \mathsf{ek}_i, \mathbf{x}_i^{(b)}, t)$ for $i \in \mathcal{H}$ and returns $\{\mathsf{ct}_i\}_{i\in\mathcal{H}}$ to $\mathcal{A}$; moreover it sets $t := t + 1$.

An adversary $\mathcal{A}$ is said to be *admissible* iff the following holds with probability 1: for any $\mathbf{y} := (\mathbf{y}_1, \ldots, \mathbf{y}_n)$ submitted in a **KGen** query where each $\mathbf{y}_i \in \{0,1\}^m$, for any $\{\mathbf{x}_{i,t}^{(0)}, \mathbf{x}_{i,t}^{(1)}\}_{i\in\mathcal{H}}$ submitted in an **Enc** query,

$$\left\langle (\mathbf{x}_{i,t}^{(0)})_{i\in\mathcal{H}}, (\mathbf{y}_i)_{i\in\mathcal{H}} \right\rangle = \left\langle (\mathbf{x}_{i,t}^{(1)})_{i\in\mathcal{H}}, (\mathbf{y}_i)_{i\in\mathcal{H}} \right\rangle$$

Definition 6 (IND-security of MCFE)**.** We say that an MCFE scheme for selection is IND-secure iff for any non-uniform p.p.t. admissible adversary $\mathcal{A}$, its views in $\mathsf{Expt}^0(1^\kappa)$ and $\mathsf{Expt}^1(1^\kappa)$ are computationally indistinguishable.

Function-hiding IND-security for MCFE. We now define a function-hiding notion of security, that is, MCFE scheme should not only hide the messages encrypted by honest clients, but also hide the selection vector $\mathbf{y}$ embedded in the functional key, as long as the selection vector $\mathbf{y}$ is selecting a coordinate in an honest client's plaintext vector. On the other hand, if $\mathbf{y}$ is selecting a coordinate in a corrupt client's plaintext vector, we need not hide which coordinate is being selected. We can formalize the intuition by considering the following experiment between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

Experiment $\mathsf{FH\text{-}Expt}^b(1^\kappa)$:

- **Setup.** Same as the eariler $\mathsf{Expt}^b(1^\kappa)$.
- **Query.** The adversary can make the following types of queries; moreover, all **KGen** queries must be made *before* any **Enc** query:
  - **KGen queries.** Whenever the adversary $\mathcal{A}$ makes a **KGen** query with two selection vectors $\mathbf{y}^{(0)} \in \{0,1\}^{mn}$ and $\mathbf{y}^{(1)} \in \{0,1\}^{mn}$: $\mathcal{C}$ calls $\mathsf{sk}_{\mathbf{y}^{(b)}} := \mathbf{KGen}(\mathsf{mpk}, \mathsf{msk}, \mathbf{y}^{(b)})$

and returns $\mathsf{sk}_{\mathbf{y}^{(b)}}$ to $\mathcal{A}$;

– **Enc queries.** Same as the eariler $\mathsf{Expt}^b(1^\kappa)$.

An adversary $\mathcal{A}$ is said to be *admissible* iff the following hold with probability 1: for any pair $(\mathbf{y}^{(0)}, \mathbf{y}^{(1)})$ submitted in a **KGen** query where for $b \in \{0,1\}$, $\mathbf{y}^{(b)} := (\mathbf{y}_1^{(b)}, \ldots, \mathbf{y}_n^{(b)}) \in \{0,1\}^{mn}$, it must be that

1. for $i \in \mathcal{K}$, $\mathbf{y}_i^{(0)} = \mathbf{y}_i^{(1)}$.
2. for any $\{\mathbf{x}_{i,t}^{(0)}, \mathbf{x}_{i,t}^{(1)}\}_{i \in \mathcal{H}}$ submitted in an **Enc** query,

$$\left\langle (\mathbf{x}_{i,t}^{(0)})_{i \in \mathcal{H}}, (\mathbf{y}_i^{(0)})_{i \in \mathcal{H}} \right\rangle = \left\langle (\mathbf{x}_{i,t}^{(1)})_{i \in \mathcal{H}}, (\mathbf{y}_i^{(1)})_{i \in \mathcal{H}} \right\rangle \tag{2}$$

Definition 7 (Function-hiding IND-security of MCFE)**.** We say that an MCFE scheme is function-hiding IND-secure iff for any non-uniform p.p.t. admissible adversary $\mathcal{A}$, its views in $\mathsf{FH\text{-}Expt}^0(1^\kappa)$ and $\mathsf{FH\text{-}Expt}^1(1^\kappa)$ are computationally indistinguishable.

Weak-function-hiding IND-security for MCFE. Weak-function-hiding IND-security is defined in almost the same manner as (full) function-hiding IND-security, except that we modify the admissibility rule to the following. An adversary $\mathcal{A}$ is now said to be *admissible* iff the following holds with probability 1: for any pair $(\mathbf{y}^{(0)}, \mathbf{y}^{(1)})$ submitted in a **KGen** query where for $b \in \{0,1\}$, $\mathbf{y}^{(b)} := (\mathbf{y}_1^{(b)}, \ldots, \mathbf{y}_n^{(b)}) \in \{0,1\}^{mn}$, it must be that

1. for $i \in \mathcal{K}$, $\mathbf{y}_i^{(0)} = \mathbf{y}_i^{(1)}$.
2. for any $\{\mathbf{x}_{i,t}^{(0)}, \mathbf{x}_{i,t}^{(1)}\}_{i \in \mathcal{H}}$ submitted in an **Enc** query,

$$\left\langle (\mathbf{x}_{i,t}^{(0)})_{i \in \mathcal{H}}, (\mathbf{y}_i^{(0)})_{i \in \mathcal{H}} \right\rangle = \left\langle (\mathbf{x}_{i,t}^{(1)})_{i \in \mathcal{H}}, (\mathbf{y}_i^{(0)})_{i \in \mathcal{H}} \right\rangle = \left\langle (\mathbf{x}_{i,t}^{(1)})_{i \in \mathcal{H}}, (\mathbf{y}_i^{(1)})_{i \in \mathcal{H}} \right\rangle \tag{3}$$

5.2 Special Function-Revealing MCFE for Selection

We now describe a special, function-revealing MCFE scheme for selection. The scheme essentially instantiates n independent instances of secret-key vector linear encryption (see Appendix C), and each vector linear encryption encrypts the m coordinates belonging to the same client. Each coordinate has a corresponding secret key. **KGen** essentially gives away the secret key corresponding to the coordinate being selected. We also add randomizing terms to each client's ciphertext to rerandomize the n partial decryptions corresponding to the n clients. The rerandomizing terms do *not* contribute to the function-revealing security of the scheme, but will become useful later during the function privacy upgrade.

Our MCFE scheme has a couple nice structural properties:

1. *Inner-product-in-the-exponent decryption.* The decryption procedure performs an inner-production computation in the exponent. Specifically, we compute the inner product (in the exponent) of each client i's ciphertext vector and a corresponding piece in the functional key; this gives the i-th partial decryption. Finally, all partial decryptions are multiplied together, and taking the discrete logarithm of the product gives the decrypted result. Note that since decryption needs to take a discrete logarithm in the end, we will apply this scheme to small plaintext spaces.

2. *Randomized partial decryptions.* Each partial decryption is randomized in a correlated manner. During an honest decryption, the decryptor uses clients' ciphertexts pertaining to the same time step. In this case, when multiplying all partial decryptions together, the randomizing terms all cancel out. On the other hand, if the decryptor mixes and matches ciphertexts from multiple time steps, the randomizing terms will not cancel out, and decryption just gives random-looking garbage.

In our scheme, the $\mathbf{Gen}(1^\kappa)$ algorithm chooses an appropriate group $\mathbb{G}$ of order q, and sets the parameters $\mathsf{pp} := (q, \mathbb{G})$, and the remaining algorithms are described below.

MCFE: special function-revealing MCFE for selection

$\mathbf{Setup}(\mathsf{pp}, m, n)$:

$\forall i \in [n] : \mathbf{S}_i \xleftarrow{\$} \mathbb{Z}_q^{m\times 2}, a_i \xleftarrow{\$} \mathbb{Z}_q$

$(K_1, \ldots, K_n) := \mathsf{CPRF}.\mathbf{Gen}(1^\kappa, n, q)$

$\mathsf{mpk} := \mathsf{pp}$

$\mathsf{msk} := \{\mathbf{S}_i, a_i\}_{i\in[n]}, \quad \mathsf{ek}_i := (K_i, a_i, \mathbf{S}_i)$

return $(\mathsf{mpk}, \mathsf{msk}, \{\mathsf{ek}_i\}_{i\in[n]})$

$\mathbf{Enc}(\mathsf{mpk}, \mathsf{ek}_i, \mathbf{x}_i, t)$:

$\mathbf{r} \xleftarrow{\$} \mathbb{Z}_q^2, \quad \mu \xleftarrow{\$} \mathbb{Z}_q$

$[\![\mathbf{c}_1]\!] := [\![\mathbf{x}_i + \mathbf{S}_i \cdot \mathbf{r}]\!] \in \mathbb{G}^m$

$[\![\mathbf{c}_2]\!] := [\![\mathbf{r}]\!] \in \mathbb{G}^2$

$[\![\widetilde{\mathbf{c}}]\!] := [\![\mathsf{CPRF}.\mathbf{Eval}(K_i, t) + a_i\mu, \ \mu]\!]$

return $\mathsf{ct}_i := ([\![\mathbf{c}_1]\!], [\![\mathbf{c}_2]\!], [\![\widetilde{\mathbf{c}}]\!])$

$\mathbf{KGen}(\mathsf{mpk}, \mathsf{msk}, \mathbf{y})$:

parse $\mathbf{y} := (\mathbf{y}_1, \ldots, \mathbf{y}_n)$ where each $\mathbf{y}_i \in \mathbb{Z}_q^m$

$\rho \xleftarrow{\$} \mathbb{Z}_q$

$\forall i \in [n] : \mathbf{k}_{i,1} := \mathbf{y}_i \in \mathbb{Z}_q^m$

$\mathbf{k}_{i,2} := -\mathbf{S}_i^\top \mathbf{y}_i \in \mathbb{Z}_q^2$

$\widetilde{\mathbf{k}}_i := (\rho, -\rho a_i) \in \mathbb{Z}_q^2$

return $\mathsf{sk}_\mathbf{y} := \{\mathbf{k}_{i,1}, \mathbf{k}_{i,2}, \widetilde{\mathbf{k}}_i\}_{i\in[n]}$

$\mathbf{Dec}(\mathsf{mpk}, \mathsf{sk}_\mathbf{y}, \{\mathsf{ct}_i\}_{i\in[n]})$

$\forall i \in [n]$: parse $\mathsf{ct}_i := [\![\mathbf{c}_{i,1}, \mathbf{c}_{i,2}, \widetilde{\mathbf{c}}_i]\!]$

parse $\mathsf{sk}_\mathbf{y} := \{\mathbf{k}_{i,1}, \mathbf{k}_{i,2}, \widetilde{\mathbf{k}}_i\}_{i\in[n]}$

$[\![v]\!] := \prod_{i\in[n]} [\![\langle \mathbf{c}_{i,1}, \mathbf{k}_{i,1}\rangle + \langle \mathbf{c}_{i,2}, \mathbf{k}_{i,2}\rangle + \langle \widetilde{\mathbf{c}}_i, \widetilde{\mathbf{k}}_i\rangle]\!]$

return $v \in \mathbb{Z}_q$ by computing $\log([\![v]\!])$

Fact 5.1. *The above* MCFE *scheme satisfies correctness.*

Proof. We can check correctness mechanically. Decryption computes the following where we use $\mathbf{r}_i$ and μ_i to differentiate between the different $\mathbf{r}$ and μ terms chosen by each client:

$$
\begin{aligned}
&\prod_{i\in[n]} [\![\langle \mathbf{c}_{i,1}, \mathbf{k}_{i,1}\rangle + \langle \mathbf{c}_{i,2}, \mathbf{k}_{i,2}\rangle + \langle \widetilde{\mathbf{c}}_i, \widetilde{\mathbf{k}}_i\rangle]\!] \\
=&\prod_{i\in[n]} [\![\mathbf{y}_i^\top(\mathbf{x}_i + \mathbf{S}_i \cdot \mathbf{r}_i) - \mathbf{y}_i^\top \cdot \mathbf{S}_i \cdot \mathbf{r}_i + (\mathsf{CPRF}.\mathbf{Eval}(K_i, t) + a_i\mu_i) \cdot \rho - \rho a_i \cdot \mu_i]\!] \\
=&\prod_{i\in[n]} [\![\langle \mathbf{x}_i, \mathbf{y}_i\rangle + \mathsf{CPRF}.\mathbf{Eval}(K_i, t) \cdot \rho]\!] \\
=&[\![\langle \mathbf{x}, \mathbf{y}\rangle]\!] \cdot [\![\sum_{i\in[n]} \mathsf{CPRF}.\mathbf{Eval}(K_i, t) \cdot \rho]\!] = [\![\langle \mathbf{x}, \mathbf{y}\rangle]\!]
\end{aligned}
$$

□

Theorem 5.2 (Special MCFE for selection, no function privacy)**.** *Assume that the Decisional Linear assumption holds in the group $\mathbb{G}$. The above* MCFE *scheme is IND-secure by Definition 6.*

Proof. In the above MCFE scheme, in each time step t, let $\mathsf{ct}_i := ([\![\mathbf{c}_{i,1}]\!], [\![\mathbf{c}_{i,2}]\!], [\![\widetilde{\mathbf{c}}_i]\!])$ be the ciphertext corresponding to the i-th client. Observe that the part $([\![\mathbf{c}_{i,1}]\!], [\![\mathbf{c}_{i,2}]\!])$ corresponds to a vector linear encryption (see Appendix C) of its input $\mathbf{x}_i \in \mathbb{Z}_q^m$. Moreover, the $\mathbf{S}_i$ term in the msk corresponds to the secret key of this vector linear encryption instance.

Now, consider a sequence of hybrid games. Hyb_0 is the same as Expt^0. In Hyb_ℓ for $\ell \in [n]$: during any **Enc** query, for any $i \in \mathcal{H}$, let $\mathbf{x}_i^{(0)}$ and $\mathbf{x}_i^{(1)}$ be the components corresponding to the i-th client in the plaintext vectors submitted by $\mathcal{A}$. For $i \leq \ell$, the challenger computes i's ciphertext components using $\mathbf{x}_i^{(1)}$; and for all other i, the challenger computes i's ciphertext components using $\mathbf{x}_i^{(0)}$.

We now argue that $\mathcal{A}$'s views in any adjacent pair $\mathsf{Hyb}_{\ell-1}$ and Hyb_ℓ are computationally indistinguishable for $\ell \in [n]$. To show this we argue that if $\mathcal{A}$ can distinguish any adjacent pairs of hybrids with non-negligible probability, we can build a reduction $\mathcal{B}$ that leverages $\mathcal{A}$ to distinguish between $\mathsf{LE\text{-}Expt}^0(1^\kappa)$ and $\mathsf{LE\text{-}Expt}^1(1^\kappa)$ as defined in Lemma C.4 of Appendix C.

The reduction $\mathcal{B}$ interacts with a vector linear encryption challenger $\mathcal{C}$ and tries to distinguish whether it is in $\mathsf{LE\text{-}Expt}^0$ or $\mathsf{LE\text{-}Expt}^1$; further, it acts as an MCFE challenger with $\mathcal{A}$.

- **Setup.**
  - $\mathcal{A}$ outputs a set of corrupt clients $\mathcal{K} \subseteq [n]$.
  - $\mathcal{B}$ interacts with a vector linear encryption challenger $\mathcal{C}$, and obtains the public parameters pp that contains the description of a suitable bilinear group. $\mathcal{C}$ has internally choose some secret key $\mathbf{S}^* \in \mathbb{Z}_q^{m \times 2}$ but $\mathcal{B}$ does not know it. $\mathcal{B}$ now runs $\mathsf{MCFE}.\mathbf{Setup}(\mathsf{pp}, m, n)$, but it will implicitly replace $\mathbf{S}_i := \mathbf{S}^*$.
  - If the adversary $\mathcal{A}$ has corrupted the client ℓ, $\mathcal{B}$ asks $\mathcal{C}$ to reveal the secret keys of all coordinates in the challenge vector linear encryption instance, i.e., the entire secret key $\mathbf{S}^*$, and returns $\mathbf{S}^*$ to $\mathcal{A}$. $\mathcal{B}$ gives mpk and $\{\mathsf{ek}_i\}_{i \in \mathcal{K}}$ to $\mathcal{A}$.
- **KGen queries**. If $\mathcal{A}$ makes a **KGen** query for selecting the i-th coordinate of the ℓ-th client's plaintext vector, $\mathcal{B}$ asks $\mathcal{C}$ to reveal the secret key for the corresponding coordinate, i.e., the i-th row of $\mathbf{S}^*$, and it can answer the **KGen** query using the i-th row of $\mathbf{S}^*$.
- **Enc queries**. Suppose that during some time step, $\mathcal{A}$ submits the two plaintext vectors $\{\mathbf{x}_i^{(0)}, \mathbf{x}_i^{(1)}\}_{i \in \mathcal{H}}$.
  - $\mathcal{B}$ computes $\{\mathsf{ct}_i\}_{i \in \mathcal{H}, i \neq \ell}$ just like in Hyb_ℓ. It can compute all these terms since it knows all these clients' secret keys.
  - If client ℓ is not corrupted: during any **Enc** query, let $\mathbf{x}_\ell^{(0)}$ and $\mathbf{x}_\ell^{(1)}$ be the components corresponding to the ℓ-th client in the two plaintext vectors submitted by $\mathcal{A}$. $\mathcal{B}$ simply forwards $\mathbf{x}_\ell^{(0)}$ and $\mathbf{x}_\ell^{(1)}$ to $\mathcal{C}$, and embeds its response in the $[\![\mathbf{c}_{\ell,1}]\!]$ and $[\![\mathbf{c}_{\ell,2}]\!]$ components of the ℓ-th client's ciphertext $\mathsf{ct}_\ell := ([\![\mathbf{c}_{\ell,1}]\!], [\![\mathbf{c}_{\ell,2}]\!], [\![\widetilde{\mathbf{c}}_\ell]\!])$. $\mathcal{B}$ can compute $[\![\widetilde{\mathbf{c}}_\ell]\!]$ on its own.

    Notice that by the admissibility rule for $\mathcal{A}$, if client ℓ is not corrupt but $\mathcal{A}$ has made **KGen** queries for selecting some coordinates $U \subseteq [m]$ in client ℓ's plaintext vector, then for any coordinate $j \in U$, it must be that $\mathbf{x}_{\ell,j}^{(0)} = \mathbf{x}_{\ell,j}^{(1)}$.

Notice that if $\mathcal{B}$ is playing $\mathsf{LE\text{-}Expt}^0$ then $\mathcal{A}$'s view is identically distributed as $\mathsf{Hyb}_{\ell-1}$; else $\mathcal{A}$'s view is identically distributed as Hyb_ℓ. □

5.3 Upgrade to Weak Function Privacy

We now describe how to upgrade the function-revealing MCFE scheme in Section 5.2 to have weak function privacy.

5.3.1 Construction

Let $\mathsf{FE} := (\mathbf{Setup}, \mathbf{KGen}, \mathbf{Enc}, \mathbf{Dec})$ denote a 1-SEL-SIM-secure single-input functional encryption scheme for computing inner-products in the exponent (see Section 4.2). Let $\mathsf{MCFE} := (\mathbf{Gen}, \mathbf{Setup}, \mathbf{KGen}, \mathbf{Enc}, \mathbf{Dec})$ denote the special multi-client inner-product encryption scheme for selection presented in Section 5.2 — recall that the ciphertext length of MCFE for each client is $m + 4$. Henceforth, we will assume that MCFE's parameter generation algorithm **Gen** chooses a suitable asymmetric bilinear group of prime order q with a pairing function $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, and we shall assume that the Decisional Linear assumption holds in $\mathbb{G}$.

We now construct a new MCFE scheme for selection, henceforth denoted $\mathsf{MCFE}^{\mathrm{wfh}}$, which satisfies weak-function-hiding IND-security.

$\mathsf{MCFE}^{\mathrm{wfh}}$: **weakly function-hiding MCFE for selection**

- $\mathbf{Gen}(1^\kappa)$: Sample a suitable prime q, and generate a suitable bilinear group of order q, with the pairing function $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. The public parameter pp contains the prime q, and the description of the bilinear group.
- $\mathbf{Setup}(\mathsf{pp}, m, n)$: Call $(\mathsf{mpk}', \mathsf{msk}', \{\mathsf{ek}'_i\}_{i\in[n]}) \leftarrow \mathsf{MCFE}.\mathbf{Setup}(\mathsf{pp}, m, n)$. For $i \in [n]$, call $(\mathsf{mpk}_i, \mathsf{msk}_i) \leftarrow \mathsf{FE}.\mathbf{Setup}(\mathsf{pp}, m + 4)$. Output the following:

$$\mathsf{mpk} := (\mathsf{pp}, \mathsf{mpk}', \{\mathsf{mpk}_i\}_{i\in[n]}), \quad \mathsf{msk} := (\mathsf{msk}', \{\mathsf{msk}_i, \mathsf{ek}_i\}_{i\in[n]}),$$
$$\forall i \in [n] : \mathsf{ek}_i := (\mathsf{msk}_i, \mathsf{ek}'_i)$$

- $\mathbf{Enc}(\mathsf{mpk}, \mathsf{ek}_i, \mathbf{x}, t)$: Let $\mathsf{ct} := \mathsf{MCFE}.\mathbf{Enc}(\mathsf{mpk}', \mathsf{ek}'_i, \mathbf{x}, t) \in \mathbb{G}_1^{m+4}$, and $\overline{\mathsf{ct}} := \mathsf{FE}.\mathbf{KGen}(\mathsf{msk}_i, \mathsf{ct})$. Output $\mathsf{CT} := (\mathsf{ct}, \overline{\mathsf{ct}})$.
- $\mathbf{KGen}(\mathsf{mpk}, \mathsf{msk}, \mathbf{y})$: Call $(\mathbf{k}_1, \ldots, \mathbf{k}_n) := \mathsf{MCFE}.\mathbf{KGen}(\mathsf{mpk}', \mathsf{msk}', \mathbf{y})$ where each $\mathbf{k}_i \in \mathbb{Z}_q^{m+4}$ for $i \in [n]$. For $i \in [n]$, call $\overline{\mathbf{k}}_i := \mathsf{FE}.\mathbf{Enc}(\mathsf{mpk}_i, [\![\mathbf{k}_i]\!])$. Output $\mathsf{sk}_\mathbf{y} := (\overline{\mathbf{k}}_1, \ldots, \overline{\mathbf{k}}_n)$.
- $\mathbf{Dec}(\mathsf{mpk}, \mathsf{sk}_\mathbf{y}, \{\mathsf{CT}_i\}_{i\in[n]})$: Parse each $\mathsf{CT}_i := (\mathsf{ct}_i, \overline{\mathsf{ct}}_i)$. Parse $\mathsf{sk}_\mathbf{y} := (\overline{\mathbf{k}}_1, \ldots, \overline{\mathbf{k}}_n)$. For $i \in [n]$, call $v_i := \mathsf{FE}.\mathbf{Dec}(\overline{\mathsf{ct}}_i, \mathsf{ct}_i, \overline{\mathbf{k}}_i)$. Output $\log(\prod_{i=1}^n v_i)$.

Theorem 5.3 (Weakly function-hiding MCFE for selection)**.** *Assume that the Decisional Linear assumption holds in $\mathbb{G}$, and suppose that in the construction in Section 5.3.1, we employ the FE scheme in Section 4.2 and the MCFE scheme in Section 5.2. Also, suppose that the CPRF function satisfies correlated pseudorandomness as defined in Section 4.3. Then, the construction in Section 5.3.1 is weakly-function-hiding IND-secure.*

Proof. The proof is presented in Section 5.3.2. □

5.3.2 Proof of Theorem 5.3

Experiment $\mathsf{FH\text{-}Expt}^0(1^\kappa)$. We start with the $\mathsf{FH\text{-}Expt}^0(1^\kappa)$ experiment described in Section 5.2. In this experiment, whenever answering either **Enc** queries or **KGen** queries from $\mathcal{A}$, the vectors

$\{\mathbf{x}_{i,t}^{(0)}\}_{i\in\mathcal{H}}$ and $\mathbf{y}^{(0)}$ are used. Recall also that $\mathcal{A}$ must satisfy the weak-function-hiding admissibility rules defined in Equation (3).

Experiment Hyb_0. In Hyb_0, when answering the t-th **Enc** query, the challenger $\mathcal{C}$ uses the vector $\{\mathbf{x}_{i,t}^{(1)}\}_{i\in\mathcal{H}}$, i.e., it calls $\mathsf{CT}_i := \mathsf{MCFE}^{\text{wfh}}.\mathbf{Enc}(\mathsf{mpk}, \mathsf{ek}_i, \mathbf{x}_i^{(1)}, t)$ for each $i \in \mathcal{H}$ and returns $\{\mathsf{CT}_i\}_{i\in\mathcal{H}}$ to $\mathcal{A}$.

Claim 5.4. *Suppose that the underlying* MCFE *scheme satisfies IND-security as defined in Definition 6, then, any non-uniform p.p.t. $\mathcal{A}$'s views in $\mathsf{FH\text{-}Expt}^0(1^\kappa)$ and Hyb_0 are computationally indistinguishable.*

Proof. Follows from a straightforward reduction to the IND-security of the underlying MCFE scheme. Note that the reduction relies on the fact that $\mathcal{A}$ must respect the admissibility rules specified in Equation (3). □

Let Q_{KGen} be the maximum number of **KGen** queries made by $\mathcal{A}$ and let Q_{Enc} be the maximum number of **Enc** queries made by $\mathcal{A}$. Without loss of generality, we may assume that $\mathcal{A}$ always makes exactly Q_{KGen} number of **KGen** queries — if not, we can consider an adversary $\mathcal{A}'$ that acts as an intermediary between $\mathcal{A}$ and the challenger, and if $\mathcal{A}$ does not make Q_{KGen} queries, $\mathcal{A}'$ will pad the number of queries to Q_{KGen}. Similarly, we may assume that $\mathcal{A}$ makes exactly Q_{Enc} number of **Enc** queries.

Experiment Hyb_ℓ. For $\ell \in [Q_{\text{KGen}}]$, Hyb_ℓ is defined as below: for the first ℓ number of **KGen** queries made by $\mathcal{A}$, generate the functional key by calling the honest $\mathsf{MCFE}^{\text{wfh}}.\mathbf{KGen}$ algorithm using $\mathbf{y}^{(1)}$ as the input vector; for all other **KGen** queries, generate the functional key by calling the honest $\mathsf{MCFE}^{\text{wfh}}.\mathbf{KGen}$ algorithm using $\mathbf{y}^{(0)}$ as the input vector. Otherwise, Hyb_ℓ is defined in the same way as Hyb_0.

Experiment $\widetilde{\mathsf{Hyb}}_\ell$. Let Q_{KGen} be the maximum number of **KGen** queries made by $\mathcal{A}$. For $\ell \in [Q_{\text{KGen}}]$, $\widetilde{\mathsf{Hyb}}_\ell$ is defined as below:

- **Setup.** For $i \in \mathcal{H}$: instead of calling $(\mathsf{mpk}_i, \mathsf{msk}_i) \leftarrow \mathsf{FE}.\mathbf{Setup}(\mathsf{pp}, m)$, the challenger $\mathcal{C}$ calls $(\mathsf{mpk}_i, \mathsf{msk}_i) \leftarrow \widetilde{\mathsf{FE}.\mathbf{Setup}}(\mathsf{pp}, m+4)$. The challenger $\mathcal{C}$ performs the remainder of the **Setup** following the honest algorithm, and gives the resulting mpk and $\{\mathsf{ek}_i\}_{i\in\mathcal{K}}$ to $\mathcal{A}$.
- **KGen queries.** The first $(\ell-1)$ **KGen** queries will be answered with the honest **KGen** algorithm using $\mathbf{y}^{(1)}$ as the input, any **KGen** query after the first ℓ queries will be answered with the honest **KGen** algorithm using $\mathbf{y}^{(0)}$ as input. For the ℓ-th **KGen** query:
  - let $\mathbf{y}^{(0)}$ and $\mathbf{y}^{(1)}$ be the two vectors submitted by $\mathcal{A}$ during the ℓ-th **KGen** query, let $\mathbf{y}^* := \mathbf{y}^{(0)} \in \mathbb{Z}_q^{mn}$;
  - let $(\mathbf{k}_1^*, \ldots, \mathbf{k}_n^*) := \mathsf{MCFE}.\mathbf{KGen}(\mathsf{mpk}', \mathsf{msk}', \mathbf{y}^*)$; where each $\mathbf{k}_i^* \in \mathbb{Z}_q^{m+4}$ for $i \in [n]$;
  - for $i \in \mathcal{H}$, the challenger $\mathcal{C}$ calls $\overline{\mathbf{k}}_i^* := \widetilde{\mathsf{FE}.\mathbf{Enc}}(\mathsf{msk}_i)$;
  - for $i \in \mathcal{K}$, the challenger $\mathcal{C}$ calls $\overline{\mathbf{k}}_i^* := \mathsf{FE}.\mathbf{Enc}(\mathsf{msk}_i, [\![\mathbf{k}_i^*]\!])$;
  - the challenger $\mathcal{C}$ returns the functional key $\mathsf{sk}_{\mathbf{y}^*} := (\overline{\mathbf{k}}_1^*, \ldots, \overline{\mathbf{k}}_n^*)$ to $\mathcal{A}$.

- **Enc queries.** For any **Enc** query with the submitted vectors $\{\mathbf{x}_i^{(0)}\}_{i\in\mathcal{H}}$ and $\{\mathbf{x}_i^{(1)}\}_{i\in\mathcal{H}}$, do the following. For $i \in \mathcal{H}$:
  - let $\mathsf{ct}_i = [\![\mathbf{c}_i]\!] := \mathsf{MCFE.Enc}(\mathsf{mpk}', \mathsf{ek}_i', \mathbf{x}_i^{(1)}, t) \in \mathbb{G}^{m+4}$, and
  - let $\overline{\mathsf{ct}_i} = [\![\overline{\mathbf{c}}_i]\!] := \mathsf{FE.}\widetilde{\mathbf{KGen}}(\mathsf{msk}_i, \mathsf{ct}_i, [\![\langle \mathbf{c}_i, \mathbf{k}_i^* \rangle]\!])$;

  Return $\{(\mathsf{ct}_i, \overline{\mathsf{ct}}_i)\}_{i\in\mathcal{H}}$ to $\mathcal{A}$.

Lemma 5.5. *Suppose that the* FE *scheme is 1-SEL-SIM-secure by Definition 3. Then, any non-uniform p.p.t. $\mathcal{A}$'s views in* Hyb_ℓ *and* $\widetilde{\mathsf{Hyb}}_{\ell+1}$ *are computationally indistinguishable for* $\ell \in [Q_{\mathrm{KGen}} - 1] \cup \{0\}$.

Proof. In Appendix D.3, we described a concurrent variant of the 1-SEL-SIM security notion that says that no non-uniform p.p.t. adversary can distinguish $\mathsf{FE\text{-}Real}[d]$ and $\mathsf{FE\text{-}Ideal}[d]$, where in $\mathsf{FE\text{-}Real}[d]$ and $\mathsf{FE\text{-}Ideal}[d]$, the 1-SEL-SIM adversary is allowed to invoke and interact with $L = \mathsf{poly}(\kappa)$ instances of the (real or simulated) FE scheme. We also prove Appendix D.3 through a standard hybrid argument that if the FE scheme is 1-SEL-SIM secure, than no non-uniform p.p.t. adversary can distinguish between $\mathsf{FE\text{-}Real}[d]$ and $\mathsf{FE\text{-}Ideal}[d]$.

We can prove the claim through a reduction to the computational indistinguishability of $\mathsf{FE\text{-}Real}[d](1^\kappa, m+4)$ and $\mathsf{FE\text{-}Ideal}[d](1^\kappa, m+4)$ for $d := |\mathcal{H}|$. Essentially, we can create a non-uniform p.p.t. reduction $\mathcal{B}$ that interacts with $\mathsf{FE\text{-}Expt}[d]$ where $\mathsf{FE\text{-}Expt}[d]$ is either $\mathsf{FE\text{-}Real}[d](1^\kappa, m+4)$ or $\mathsf{FE\text{-}Ideal}[d](1^\kappa, m+4)$. $\mathcal{B}$ wants to distinguish which experiment it is interacting with by leveraging the adversary $\mathcal{A}$ which tries to distinguish between Hyb_ℓ and $\widetilde{\mathsf{Hyb}}_{\ell+1}$.

- **Setup**. $\mathcal{B}$ embeds the terms from FE parameters obtained from $\mathsf{FE\text{-}Expt}[d]$ into the honest clients' FE public keys, that is, $\{\mathsf{mpk}_i\}_{i\in\mathcal{H}}$.
- **KGen queries**. During the $(\ell+1)$-th **KGen** query from $\mathcal{A}$, let $\mathbf{y}^{(0)}$ and $\mathbf{y}^{(1)}$ be the two vectors submitted by $\mathcal{A}$: $\mathcal{B}$ computes $(\mathbf{k}_1^*, \ldots, \mathbf{k}_n^*) := \mathsf{MCFE.KGen}(\mathsf{mpk}', \mathsf{msk}', \mathbf{y}^{(0)})$. Now, $\mathcal{B}$ forwards $\{\mathbf{k}_i^*\}_{i\in\mathcal{H}}$ to $\mathsf{FE\text{-}Expt}[d]$ as the challenge plaintext; $\mathcal{B}$ receives from $\mathsf{FE\text{-}Expt}[d]$ a ciphertext and uses the returned ciphertext as $\{\overline{\mathbf{k}}_i^*\}_{i\in\mathcal{H}}$ in its interaction with $\mathcal{A}$.

  During all other **KGen** queries from $\mathcal{A}$ except the $(\ell+1)$-th **KGen** query, $\mathcal{B}$ computes $\overline{\mathbf{k}}_i := \mathsf{FE.Enc}(\mathsf{mpk}_i, \mathbf{k}_i)$ for $i \in \mathcal{H}$ by itself, and uses the results $\{\overline{\mathbf{k}}_i\}_{i\in\mathcal{H}}$ in its interaction with $\mathcal{A}$.
- **Enc queries**. During any **Enc** query from $\mathcal{A}$, whenever $\mathcal{B}$ needs to call $\mathsf{FE.}\widetilde{\mathbf{KGen}}$ on some vector $\mathsf{ct}_i = [\![\mathbf{c}_i]\!]$, it forwards ct_i to $\mathsf{FE\text{-}Expt}[d]$ as a **KGen** query for the corresponding FE instance. The returned result will be used as $\overline{\mathsf{ct}_i}$ in $\mathcal{B}$'s interaction with $\mathcal{A}$. Since any **Enc** query must be made after all **KGen** queries are made, at the time $\mathcal{A}$ makes an **Enc** query, $\mathcal{B}$ must have submitted a challenge plaintext to $\mathsf{FE\text{-}Expt}[d]$.

Besides the above, $\mathcal{B}$ interacts with $\mathcal{A}$ just like in $\widetilde{\mathsf{Hyb}}_{\ell+1}$, and it outputs the same guess that is output by $\mathcal{A}$.

Observe that if $\mathsf{FE\text{-}Expt}[d] = \mathsf{FE\text{-}Real}[d]$, then $\mathcal{A}$'s view is identically distributed as in Hyb_ℓ; else $\mathcal{A}$'s view is identically distributed as $\widetilde{\mathsf{Hyb}}_{\ell+1}$. □

Next, we would like to show that no non-uniform p.p.t. adversary can distinguish $\widetilde{\mathsf{Hyb}}_\ell$ and Hyb_ℓ for $\ell \in [Q_{\mathrm{KGen}}]$ (Lemma 5.6). To show this, we will rely on a sequence of hybrid experiments.

Lemma 5.6. *Suppose that the Decisional Linear assumption holds in $\mathbb{G}$, and that the* CPRF *scheme satisfies correlated pseudorandomness. Further, suppose that we employ the* MCFE *scheme described in Section 5.2. Then, any non-uniform p.p.t. adversary's views in $\widetilde{\mathsf{Hyb}}_\ell$ and Hyb_ℓ are computationally indistinguishable for $\ell \in [Q_{\mathrm{KGen}}]$.*

Proof. We will prove the lemma with a sequence of hybrid experiments.

Experiment $\mathsf{Hyb}^{\blacktriangledown}_\ell$. Recall that in $\widetilde{\mathsf{Hyb}}_\ell$,

- During the ℓ-th **KGen** query, the challenger computes the following terms (among other things):
$$(\mathbf{k}^*_1, \ldots, \mathbf{k}^*_n) := \mathsf{MCFE}.\mathbf{KGen}(\mathsf{mpk}', \mathsf{msk}', \mathbf{y}^*)$$
Henceforth we will parse each $\mathbf{k}^*_i$ as $\mathbf{k}^*_i := (\mathbf{k}^*_{i,1}, \mathbf{k}^*_{i,2}, \widetilde{\mathbf{k}}^*_i)$. Recall that using our MCFE construction in Section 5.2, $\widetilde{\mathbf{k}}^*_i$ is of the form $(\rho^*, -\rho^* \cdot a_i)$ for some random choice of ρ^* sampled in the ℓ-th **KGen** query.
- During each **Enc** query, the challenger computes for each honest $i \in \mathcal{H}$:
$$\overline{\mathsf{ct}_i} = [\![\overline{\mathbf{c}}_i]\!] := \mathsf{FE}.\widetilde{\mathbf{KGen}}(\mathsf{msk}_i, \mathsf{ct}_i, [\![\langle \mathbf{c}_i, \mathbf{k}^*_i \rangle]\!])$$

The experiment $\mathsf{Hyb}^{\blacktriangledown}_\ell$ is almost the same as $\widetilde{\mathsf{Hyb}}_\ell$, except with the following modification. During the t-th **Enc** query for $t = 1, 2, \ldots$:

1. Choose random $\{\widetilde{T}_i\}_{i \in \mathcal{H}}$ from $\mathbb{G}$ subject to the constraint that
$$\prod_{i \in \mathcal{H}} \widetilde{T}_i \cdot \prod_{j \in \mathcal{K}} [\![\mathsf{CPRF}.\mathbf{Eval}(K_j, t) \cdot \rho^*]\!] = 1 \tag{4}$$
where K_j is j's CPRF key in the underlying MCFE scheme.
2. When $[\![\langle \mathbf{c}_i, \mathbf{k}^*_i \rangle]\!] := [\![\langle \mathbf{c}_{i,1}, \mathbf{k}^*_{i,1} \rangle + \langle \mathbf{c}_{i,2}, \mathbf{k}^*_{i,2} \rangle + \langle \widetilde{\mathbf{c}}_i, \widetilde{\mathbf{k}}^*_i \rangle]\!]$ is computed and passed as input to $\mathsf{FE}.\widetilde{\mathbf{KGen}}$, replace the term $[\![\langle \widetilde{\mathbf{c}}_i, \widetilde{\mathbf{k}}^*_i \rangle]\!]$ with $\widetilde{T}_i$ where $i \in \mathcal{H}$.

Henceforth, let $\{i_1, i_2, \ldots, i_d\}$ denote the set of honest clients where $i_1 < i_2 < \ldots < i_d$ and $d = |\mathcal{H}|$. We can equivalently view $\mathsf{Hyb}^{\blacktriangledown}_\ell$ as follows — for each time step t:

- For all but the last honest client $i \in \{i_1, \ldots, i_{d-1}\}$, generate $\widetilde{T}_i$ at random from $\mathbb{G}$;
- For the last honest client $i = i_d$, compute $\widetilde{T}_i$ using Equation (4).
- Instead of computing $\mathsf{FE}.\widetilde{\mathbf{KGen}}(\mathsf{msk}_i, \mathsf{ct}_i, [\![\langle \mathbf{c}_{i,1}, \mathbf{k}^*_{i,1} \rangle + \langle \mathbf{c}_{i,2}, \mathbf{k}^*_{i,2} \rangle + \langle \widetilde{\mathbf{c}}_i, \widetilde{\mathbf{k}}^*_i \rangle]\!])$ where $i \in \mathcal{H}$, call $\mathsf{FE}.\widetilde{\mathbf{KGen}}(\mathsf{msk}_i, \mathsf{ct}_i, [\![\langle \mathbf{c}_{i,1}, \mathbf{k}^*_{i,1} \rangle + \langle \mathbf{c}_{i,2}, \mathbf{k}^*_{i,2} \rangle + \widetilde{T}_i]\!])$. Observe that the second parameter ct_i depends on $\mathsf{CPRF}.\mathbf{Eval}(K_i, t)$, but the third parameter $[\![\langle \mathbf{c}_{i,1}, \mathbf{k}^*_{i,1} \rangle + \langle \mathbf{c}_{i,2}, \mathbf{k}^*_{i,2} \rangle + \widetilde{T}_i]\!]$ does not depend on $\mathsf{CPRF}.\mathbf{Eval}(K_i, t)$ any more.

Lemma 5.7. *Suppose that the Decisional Linear assumption holds in $\mathbb{G}$, and that* CPRF *satisfies correlated pseudorandomness. Then, any non-uniform p.p.t. adversary's views in $\widetilde{\mathsf{Hyb}}_\ell$ and $\mathsf{Hyb}^{\blacktriangledown}_\ell$ are computationally indistinguishable.*

Proof. We consider a sequence of hybrid experiments, where in the j-th hybrid H^j where $j \in [d-1] \cup \{0\}$,

1. Regardless of j, we make the following modification. For all but one honest client $i \in \{i_1, \ldots, i_{d-1}\}$, whenever the experiment queries $\mathsf{CPRF}.\mathbf{Eval}(K_i, \cdot)$, we replace the answer with a call to a random function $\mathsf{RO}_i(\cdot)$. For the last honest client, a call to $\mathsf{CPRF}.\mathbf{Eval}(K_{i_d}, t)$ is computed using the constraint

$$\sum_{i \in \mathcal{H}} \mathsf{RO}_i(t) + \sum_{j \in \mathcal{K}} \mathsf{CPRF}.\mathbf{Eval}(K_j, t) = 0$$

2. If i is among the first j honest clients, choose $\widetilde{T}_i \xleftarrow{\$} \mathbb{G}$ at random.
3. If i is honest but not among the first j honest clients and also not the last honest client, then the experiment chooses the $\widetilde{T}_i$ value in each time step t as follows: $\widetilde{T}_i := [\![\mathsf{RO}_i(t) \cdot \rho^*]\!]$.
4. For the last honest client i_d, its $\widetilde{T}_{i_d}$ value is computed using the constraint in Equation (4).

Besides the above modifications, every H^j would otherwise behave just like $\mathsf{Hyb}^{\blacktriangledown}_\ell$.

Observe that H^0 is computationally indistinguishable from $\widetilde{\mathsf{Hyb}}_\ell$ through a straightforward reduction to the correlated pseudorandomness of CPRF. Further, H^{d-1} the same as $\mathsf{Hyb}^{\blacktriangledown}_\ell$. Therefore, it suffices to show that no non-uniform p.p.t. adversary can distinguish between any two adjacent hybrids H^{j^*} and H^{j^*+1} except with negligible probability, for $j^* \in \{0, 1, \ldots, d-2\}$. Suppose there is an efficient adversary $\mathcal{A}$ that can distinguish H^{j^*} and H^{j^*+1} with non-negligible probability, we show how to construct an efficient reduction $\mathcal{B}$ that can break the Decisional Linear assumption.

Suppose that $\mathcal{B}$ obtains an instance $([\![1]\!], [\![\beta]\!], [\![\gamma]\!], [\![\mathbf{u}]\!], [\![\beta\mathbf{v}]\!], [\![\mathbf{z}]\!])$ from a Vector Decisional Linear challenger (see Section C.2), where $\mathbf{u}, \mathbf{v}, \mathbf{z} \in \mathbb{Z}_q^{Q_{\mathrm{enc}}}$ and $\beta, \gamma \in \mathbb{Z}_q$. $\mathcal{B}$'s task is to distinguish whether $[\![\mathbf{z}]\!] = [\![\gamma(\mathbf{u} + \mathbf{v})]\!]$ or random. $\mathcal{B}$ will now interact with $\mathcal{A}$ and embed this Decisional Linear instance in its answers.

Let i^* be the index of the (j^*+1)-th honest client. Recall that i_d denotes the index of the last honest client.

- **Setup.** When running $(\mathsf{mpk}', \mathsf{msk}', \{\mathsf{ek}'_i\}_{i \in [n]}) \leftarrow \mathsf{MCFE}.\mathbf{Setup}(\mathsf{pp}, m, n)$, $\mathcal{B}$ chooses $\xi \in \mathbb{Z}_q$ at random, and implicitly sets the terms $a_{i^*} := \beta^{-1}$ and $a_{i_d} = \xi \cdot \beta^{-1}$ without actually computing them — note that $\beta \neq 0$ with all but negligible probability, so we can ignore the event that $\beta = 0$. $\mathcal{B}$ runs the rest of the $\mathsf{MCFE}.\mathbf{Setup}$ honestly. Note that all $\mathcal{B}$ can compute all terms of mpk' and $\{\mathsf{ek}'_i\}_{i \in \mathcal{K}}$.
- **KGen queries.**
  1. The first $(\ell - 1)$ **KGen** queries will be answered with the honest **KGen** algorithm using $\mathbf{y}^{(1)}$ as the input. However, we need to explain how to still compute the correct **KGen** algorithm without knowing a_{i^*} and a_{i_d}. Recall that $\mathsf{MCFE}.\mathbf{KGen}(\mathsf{mpk}', \mathsf{msk}', \mathbf{y}^{(1)})$ gives a key of the form $\{\mathbf{k}_i := (\mathbf{k}_{i,1}, \mathbf{k}_{i,2}, \widetilde{\mathbf{k}}_i)\}_{i \in [n]}$.
     - For all indices $i \neq i^*$ and $i \neq i_d$, $\mathcal{B}$ knows all necessary terms to compute $\mathbf{k}_i := (\mathbf{k}_{i,1}, \mathbf{k}_{i,2}, \widetilde{\mathbf{k}}_i)$ normally.
     - For i^*, the terms $\mathbf{k}_{i^*,1}, \mathbf{k}_{i^*,2}$ need not use a_{i^*} and can be computed normally. We thus focus on the term $\widetilde{\mathbf{k}}_{i^*}$ which is supposed to be of the form $(\rho, -\rho \cdot a_{i^*})$ where ρ is chosen at random for each **KGen**. $\mathcal{B}$ implicitly use the terms $[\![\beta\rho', -\rho']\!]$ in place of $[\![\rho, -\rho \cdot a_{i^*}]\!]$. Note that $\mathcal{B}$ does not know the exponents but it can nonetheless complete the remainder of the computation, since it will next compute $\mathsf{FE}.\mathbf{Enc}(\mathsf{mpk}_i, [\![\mathbf{k}_i]\!])$ and this step only needs the group encoding of these elements.

- For i_d, $\mathcal{B}$ computes the response in a similar fashion as for i^* since it knows ξ.

2. Any **KGen** query after the first ℓ queries will be answered with the honest **KGen** algorithm using $\mathbf{y}^{(0)}$ as input. Using the same argument as above, although $\mathcal{B}$ does not know a_{i^*} and a_{i_d}, it can still compute these keys.
3. We now focus on the ℓ-th **KGen** query. For the ℓ-th **KGen** query:
   - Let $\mathbf{y}^{(0)}$ and $\mathbf{y}^{(1)}$ be the two vectors submitted by $\mathcal{A}$ during the ℓ-th **KGen** query, let $\mathbf{y}^* := \mathbf{y}^{(0)} \in \mathbb{Z}_q^{mn}$;
   - For $i \in \mathcal{H}$, $\mathcal{B}$ calls $\overline{\mathbf{k}}_i^* := \mathsf{FE}.\widetilde{\mathbf{Enc}}(\mathsf{msk}_i)$;
   - We now focus on how to compute the corrupt components $\overline{\mathbf{k}}_i^*$ where $i \in \mathcal{K}$. $\mathcal{B}$ wants to implicitly embed the γ term from the Decisional Linear challenge into the ρ term in the ℓ-th **KGen** query. However, $\mathcal{B}$ knows only $[\![\gamma]\!]$ but not the exponent γ. However, this is not a problem because $\mathcal{A}$ knows the a_i terms for $i \in \mathcal{K}$, and thus it can compute the term $[\![\widetilde{\mathbf{k}}_i]\!] := [\![\rho, -\rho a_i]\!] = [\![\gamma, -\gamma a_i]\!]$. Moreover, the following step $\mathsf{FE}.\mathbf{Enc}(\mathsf{mpk}_i, [\![\mathbf{k}_i]\!])$ only needs knowledge of the group encoding of $\mathbf{k}_i$.

- **Enc queries.** For any **Enc** query with the submitted vectors $\{\mathbf{x}_i^{(0)}\}_{i\in\mathcal{H}}$ and $\{\mathbf{x}_i^{(1)}\}_{i\in\mathcal{H}}$, do the following.
  - For every honest client that is neither the last honest client nor i^*, compute $\mathsf{ct}_i = [\![\mathbf{c}_i]\!] := [\![\mathbf{c}_{i,1}, \mathbf{c}_{i,2}, \widetilde{\mathbf{c}}_i]\!] := \mathsf{MCFE}.\mathbf{Enc}(\mathsf{mpk}', \mathsf{ek}_i', \mathbf{x}_i^{(1)}, t) \in \mathbb{G}^{m+4}$ honestly except that the $\mathsf{CPRF}.\mathbf{Eval}(K_i, t)$ term is replaced with $\mathsf{RO}_i(t)$. Let $\overline{\mathsf{ct}}_i := \mathsf{FE}.\widetilde{\mathbf{KGen}}(\mathsf{msk}_i,\ \mathsf{ct}_i,\ [\![\langle \mathbf{c}_{i,1}, \mathbf{k}_{i,1}^*\rangle + \langle \mathbf{c}_{i,2}, \mathbf{k}_{i,2}^*\rangle + \widetilde{T}_i]\!])$ where $\widetilde{T}_i$ is chosen like H^{j^*}, that is, if i is among the first j^* clients, $\widetilde{T}_i \xleftarrow{\$} \mathbb{G}$; else, $\widetilde{T}_i := [\![\mathsf{RO}_i(t) \cdot \gamma]\!]$ which can be computed knowing $\mathsf{RO}_i(t)$ and $[\![\gamma]\!]$.
  - For the honest client i^*, recall that $\mathcal{B}$ does not know a_{i^*}. $\mathcal{B}$ will compute the terms $[\![\mathbf{c}_{i^*,1}]\!], [\![\mathbf{c}_{i^*,2}]\!]$ honestly. The term $[\![\widetilde{\mathbf{c}}_{i^*}]\!]$ is generated as follows: let u_t, v_t, z_t be the t-th coordinate of $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{z}$, respectively. We will let
$$[\![\widetilde{\mathbf{c}}_{i^*}]\!] := [\![u_t, -\beta v_t]\!],$$
    Now, let $\overline{\mathsf{ct}}_{i^*} := \mathsf{FE}.\widetilde{\mathbf{KGen}}(\mathsf{msk}_{i^*}, \mathsf{ct}_{i^*}, [\![\langle \mathbf{c}_{i^*,1}, \mathbf{k}_{i^*,1}^*\rangle + \langle \mathbf{c}_{i^*,2}, \mathbf{k}_{i^*,2}^*\rangle + z_t]\!])$.
    In other words, we are implicitly letting
$$\mathsf{RO}_{i^*}(t) + a_{i^*}\mu_{i^*,t} = u_t, \quad \mu_{i^*,t} = -\beta v_t$$
    where $\mu_{i^*,t}$ is the μ term chosen by i^* in the t-th **Enc** query. Thus $\mathsf{RO}_{i^*}(t)\cdot\gamma = (u_t - a_{i^*}\mu_{i^*,t})\gamma = (u_t - \beta^{-1}\mu_{i^*,t})\gamma = (u_t + \beta^{-1}\beta v_t)\gamma = (u_t + v_t)\gamma$.
  - For the last honest client henceforth denoted i_d, $\mathcal{B}$ will compute the terms $[\![\mathbf{c}_{i_d,1}]\!], [\![\mathbf{c}_{i_d,2}]\!]$ honestly. We now focus on how to compute $[\![\widetilde{\mathbf{c}}_{i_d}]\!]$. $\mathcal{B}$ samples a random $\phi \in \mathbb{Z}_q$ and implicitly chooses
$$\mu_{i_d,t} = -\mu_{i^*,t}\cdot\xi^{-1} + a_{i^*}^{-1}\cdot\phi, \quad \mathsf{RO}_{i_d}(t) := -\left(\sum_{j\in\mathcal{H}, j\neq i_d} \mathsf{RO}_j(t) + \sum_{j\in\mathcal{K}} \mathsf{CPRF}.\mathbf{Eval}(K_j, t)\right)$$
    We may write $\mathsf{RO}_{i_d}(t)$ as $\mathsf{RO}_{i_d}(t) := \nu - \mathsf{RO}_{i^*}(t)$ where
$$\nu := -\left(\sum_{j\in\mathcal{H}, j\neq i_d, j\neq i^*} \mathsf{RO}_j(t) + \sum_{j\in\mathcal{K}} \mathsf{CPRF}.\mathbf{Eval}(K_j, t)\right) \in \mathbb{Z}_q \text{ is known to } \mathcal{B}.$$

Now, $\mathcal{B}$ will compute $[\![\widetilde{\mathbf{c}}_{i_d}]\!] := [\![\mathsf{RO}_{i_d}(t) + a_{i_d}\mu_{i_d,t}, \mu_{i_d,t}]\!]$ as follows:

$$\begin{aligned}[\![\mathsf{RO}_{i_d}(t) + a_{i_d}\mu_{i_d,t}]\!] &= [\![\nu - \mathsf{RO}_{i^*}(t) + \xi a_{i^*} \cdot (-\mu_{i^*,t} \cdot \xi^{-1} + a_{i^*}^{-1} \cdot \phi)]\!] \\ &= [\![\nu - \mathsf{RO}_{i^*}(t) - a_{i^*}\mu_{i^*,t} + \xi\phi]\!] \\ &= [\![\nu - u_t + \xi\phi]\!]\end{aligned}$$

which can be computed knowing $\nu, [\![u_t]\!]$, and $\xi\phi$. Further,

$$[\![\mu_{i_d,t}]\!] = [\![-\mu_{i^*,t} \cdot \xi^{-1} + a_{i^*}^{-1} \cdot \phi]\!] = [\![\beta v_t \cdot \xi^{-1} + \phi \cdot a_{i^*}^{-1}]\!] = [\![\beta v_t \cdot \xi^{-1} + \phi \cdot \beta]\!]$$

which can be computed knowing $[\![\beta v_t]\!]$, $[\![\beta]\!]$ and the exponents ξ and ϕ.

Next, let $\mathsf{ct}_{i_d} := ([\![\mathbf{c}_{i_d}, 1]\!], [\![\mathbf{c}_{i_d}, 2]\!], [\![\widetilde{\mathbf{c}}_{i_d}]\!])$. $\mathcal{B}$ calls $\mathsf{FE}.\widetilde{\mathbf{KGen}}(\mathsf{msk}_{i_d}, \mathsf{ct}_{i_d}, \widetilde{T}_{i_d})$ where $\widetilde{T}_{i_d}$ is chosen such that $\prod_{i\in\mathcal{H}} \widetilde{T}_i \cdot \prod_{j\in\mathcal{K}} [\![\mathsf{CPRF}.\mathbf{Eval}(K_j, t) \cdot \gamma]\!] = 1$ — we let $\widetilde{T}_{i^*} := [\![z_t]\!]$; and for $i \in \mathcal{H}, i \neq i_d, i \neq i^*$, let $\widetilde{T}_i$ be chosen like in H^{j^*}.

– Finally, and return $\{(\mathsf{ct}_i, \overline{\mathsf{ct}}_i)\}_{i\in\mathcal{H}}$ to $\mathcal{A}$.

If the Vector Decisional Linear tuple $([\![1]\!], [\![\beta]\!], [\![\gamma]\!], [\![\mathbf{u}]\!], [\![\beta\mathbf{v}]\!], [\![\mathbf{z}]\!])$ satisfies $[\![\mathbf{z}]\!] = [\![\gamma(\mathbf{u} + \mathbf{v})]\!]$, then $\mathcal{A}$'s view in the above experiment is identical to H^{j^*}. Else, if $[\![\mathbf{z}]\!]$ is randomly chosen, then $\mathcal{A}$'s view in the above experiment is identical to H^{j^*+1}. □

Experiment $\mathsf{Hyb}_\ell^\triangledown$. The hybrid $\mathsf{Hyb}_\ell^\triangledown$ is defined almost identically as $\mathsf{Hyb}_\ell^\blacktriangledown$, except that in the ℓ-th **KGen** query, the challenger switches to using $\mathbf{y}^{(1)}$ instead of $\mathbf{y}^{(0)}$.

Claim 5.8. *$\mathsf{Hyb}_\ell^\blacktriangledown$ and $\mathsf{Hyb}_\ell^\triangledown$ are identically distributed.*

Proof. Observe that in $\mathsf{Hyb}_\ell^\blacktriangledown$, there are only two places that depend on the two vectors $\mathbf{y}^{(0)} := (\mathbf{y}_1^{(0)}, \ldots, \mathbf{y}_n^{(0)})$ and $\mathbf{y}^{(1)} := (\mathbf{y}_1^{(1)}, \ldots, \mathbf{y}_n^{(1)})$ submitted during the ℓ-th **KGen** query:

1. During the ℓ-th **KGen** query, when $\overline{\mathbf{k}}_i^* := \mathsf{FE}.\mathbf{Enc}(\mathsf{mpk}_i, \mathbf{k}_i^*)$ is called for $i \in \mathcal{K}$. Here the term $\mathbf{k}_i^*$ depends on $\mathbf{y}_i^{(0)}$, but we know that $\mathbf{y}_i^{(0)} = \mathbf{y}_i^{(1)}$ for $i \in \mathcal{K}$ by the admissibility rule of weakly function-hiding IND-security notion. Therefore, here, switching to $\mathbf{y}_i^{(1)}$ does not alter the distribution.

2. During each **Enc** query, the last input parameter in the call to $\mathsf{FE}.\widetilde{\mathbf{KGen}}$ depends on $\mathbf{y}_i^{(0)}$ where $i \in \mathcal{H}$ in $\mathsf{Hyb}_\ell^\blacktriangledown$. By the construction of our MCFE scheme in Section 5.2, for every honest client i, the last parameter is equal to

$$[\![\langle \mathbf{x}_i^{(1)}, \mathbf{y}_i^{(0)} \rangle]\!] \cdot \widetilde{T}_i$$

where $\widetilde{T}_i$ is defined as in $\mathsf{Hyb}_\ell^\blacktriangledown$. As long as the admissibility rule for weak function hiding holds (see Equation 3), switching $\{\mathbf{y}_i^{(0)}\}_{i\in\mathcal{H}}$ to $\{\mathbf{y}_i^{(1)}\}_{i\in\mathcal{H}}$ does not alter the joint distribution.

□

Completing the proof of Lemma 5.6. At this moment, using a symmetric argument, we can show through a sequence of hybrids that $\mathsf{Hyb}_\ell^\triangledown$ is computationally indistinguishable from Hyb_ℓ.

□

Proof of Theorem 5.3. Suppose that the Decisional Linear assumption holds in $\mathbb{G}$. Then, by Theorem 4.1 and Theorem 5.2, FE is 1-SEL-SIM-secure and MCFE is IND-secure. Given this, the theorem follows directly from Claim 5.4, Lemma 5.5, and Lemma 5.6 — notice also that $\mathsf{Hyb}_{Q_{\mathrm{KGen}}}$ is the same as $\mathsf{FH\text{-}Expt}^1$.

5.4 Upgrade to Full Function Privacy

We now describe how to upgrade the $\mathsf{MCFE}^{\mathrm{wfh}}$ construction in Section 5.3 to have full function privacy. Here we use a two-slot trick used in prior works on Functional Encryption and Indistinguishability Obfuscation [SSW09, BJK15, GGG+14, LV16, Lin17, ACF+18]. Let $\mathsf{MCFE}^{\mathrm{wfh}}$ be a weakly function-hiding MCFE scheme for selection such as the one described in Section 5.3.1. We will now construct a fully function-hiding MCFE scheme for selection, henceforth denoted employing $\mathsf{MCFE}^{\mathrm{ffh}}$.

$\mathsf{MCFE}^{\mathrm{ffh}}$: fully function-hiding MCFE for selection

- **Gen**(1^κ): call $\mathsf{pp} := \mathsf{MCFE}^{\mathrm{wfh}}.\mathbf{Gen}(1^\kappa)$ and output the resulting parameters pp.
- **Setup**(pp, m, n): call $(\mathsf{mpk}, \mathsf{msk}, \{\mathsf{ek}_i\}_{i\in[n]}) := \mathsf{MCFE}^{\mathrm{wfh}}.\mathbf{Setup}(\mathsf{pp}, 2m, n)$. and output the resulting terms $\mathsf{mpk}, \mathsf{msk}$, and $\{\mathsf{ek}_i\}_{i\in[n]}$.
- **Enc**$(\mathsf{mpk}, \mathsf{ek}_i, \mathbf{x}, t)$: call $\mathsf{CT} := \mathsf{MCFE}^{\mathrm{wfh}}.\mathbf{Enc}(\mathsf{mpk}, \mathsf{ek}_i, \mathbf{x}||\mathbf{0}, t)$ where $\mathbf{0} \in \mathbb{Z}_q^m$, and output CT.
- **KGen**$(\mathsf{mpk}, \mathsf{msk}, \mathbf{y})$: parse $\mathbf{y} := (\mathbf{y}_1, \ldots, \mathbf{y}_n)$ where each $\mathbf{y}_i \in \{0,1\}^m$. Let $\widetilde{\mathbf{y}} = ((\mathbf{y}_1, \mathbf{0}), \ldots, (\mathbf{y}_n, \mathbf{0})) \in \{0,1\}^{2mn}$. Call $\mathsf{sk}_{\mathbf{y}} := \mathsf{MCFE}^{\mathrm{wfh}}.\mathbf{KGen}(\mathsf{mpk}, \mathsf{msk}, \widetilde{\mathbf{y}})$ where $\mathbf{0} \in \mathbb{Z}_q^m$.
- **Dec**$(\mathsf{mpk}, \mathsf{sk}_{\mathbf{y}}, \{\mathsf{CT}_i\}_{i\in[n]})$: call $x := \mathsf{MCFE}^{\mathrm{wfh}}.\mathbf{Dec}(\mathsf{mpk}, \mathsf{sk}_{\mathbf{y}}, \{\mathsf{CT}_i\}_{i\in[n]})$ and output x.

Theorem 5.9 (Fully function-hiding MCFE for selection)**.** *Assume that* $\mathsf{MCFE}^{\mathrm{wfh}}$ *is weakly-function-hiding IND-secure. Then, the above construction is function-hiding IND-secure.*

Proof. We can prove the theorem with a sequence of hybrid experiments as described by the following table.

Experiment	**Enc** query for $i \in \mathcal{H}$	**KGen** query for $i \in \mathcal{H}$	**KGen** query for $i \in \mathcal{K}$
$\mathsf{FH\text{-}Expt}^0$	$\mathbf{x}_i^{(0)}\|\|\mathbf{0}$	$\mathbf{y}_i^{(0)}\|\|\mathbf{0}$	$\mathbf{y}_i^{(0)}\|\|\mathbf{0} = \mathbf{y}_i^{(1)}\|\|\mathbf{0}$
Hyb_1	$\mathbf{x}_i^{(0)}\|\|\mathbf{x}_i^{(1)}$	$\mathbf{0}\|\|\mathbf{y}_i^{(1)}$	
Hyb_2	$\mathbf{x}_i^{(1)}\|\|\mathbf{x}_i^{(1)}$	$\mathbf{y}_i^{(1)}\|\|\mathbf{0}$	
$\mathsf{FH\text{-}Expt}^1$	$\mathbf{x}_i^{(1)}\|\|\mathbf{0}$	$\mathbf{y}_i^{(1)}\|\|\mathbf{0}$	

Essentially, the table shows:

1. whenever $\mathcal{A}$ submits an **Enc** query with two vectors $\mathbf{x}^{(0)} := \{\mathbf{x}_i^{(0)}\}_{i\in\mathcal{H}}$ and $\mathbf{x}^{(1)} := \{\mathbf{x}_i^{(1)}\}_{i\in\mathcal{H}}$ what is the plaintext the challenger $\mathcal{C}$ encrypts by calling the underlying $\mathsf{MCFE}^{\mathrm{wfh}}.\mathbf{Enc}$ for every $i \in \mathcal{H}$;
2. whenever $\mathcal{A}$ submits an **KGen** query with two vectors $\mathbf{y}^{(0)}$ and $\mathbf{y}^{(1)}$, what is the key vector the challenger $\mathcal{C}$ passes to the underlying $\mathsf{MCFE}^{\mathrm{wfh}}.\mathbf{KGen}$ call.

Any non-uniform p.p.t. adversary $\mathcal{A}$'s views in any pair of adjacent hybrids are computationally indistinguishable due to the fact that $\mathsf{MCFE}^{\mathrm{wfh}}$ is weakly-function-hiding IND-secure. Therefore, the theorem follows. □

6 NIAR Construction

Let $\mathsf{SE} := (\mathbf{Gen}, \mathbf{Enc}, \mathbf{Dec})$ denote a symmetric-key encryption scheme. Let $\mathsf{MCFE}^{\mathrm{ffh}}$ denote a fully function-hiding MCFE scheme for selection. We can construct a NIAR scheme as below.

NIAR with receiver-insider protection

- $\mathbf{Setup}(1^\kappa, n, \pi)$:
  - For $i \in [n]$, let $\mathsf{sk}_i := \mathsf{SE}.\mathbf{Gen}(1^\kappa)$. Let $\mathsf{pp} := \mathsf{MCFE}^{\mathrm{ffh}}.\mathbf{Gen}(1^\kappa)$, and $(\mathsf{mpk}, \mathsf{msk}, \{\mathsf{ek}'_i\}_{i\in[n]}) := \mathsf{MCFE}^{\mathrm{ffh}}.\mathbf{Setup}(\mathsf{pp}, 1, n)$.
  - For $i \in [n]$, let $j := \pi^{-1}(i)$, and let $\mathbf{e}_j$ be the j-th selection vector, i.e., except the j-th coordinate which is 1, all other coordinates are 0; call $\mathsf{tk}_i := \mathsf{MCFE}^{\mathrm{ffh}}.\mathbf{KGen}(\mathsf{mpk}, \mathsf{msk}, \mathbf{e}_j)$.
  - Output
$$\mathsf{tk} := \{\mathsf{tk}_i\}_{i\in[n]}, \quad \left\{\forall i \in [n] : \quad \mathsf{ek}_i := (\mathsf{mpk}, \mathsf{sk}_i, \mathsf{ek}'_i),\ \mathsf{rk}_i := \mathsf{sk}_{\pi(i)}\right\}$$
- $\mathbf{Enc}(\mathsf{ek}_i, x_i, t)$: Let $c := \mathsf{SE}.\mathbf{Enc}(\mathsf{sk}_i, x_i)$, and henceforth let $(c)_j$ denote the j-th bit of c. For $j \in [L]$ where $L := |c|$, let $(\mathsf{ct}_{i,t})_j := \mathsf{MCFE}^{\mathrm{ffh}}.\mathbf{Enc}(\mathsf{mpk}, \mathsf{ek}'_i, (c)_j, (t-1)L + j)$. Output $\mathsf{ct}_{i,t} := \{(\mathsf{ct}_{i,t})_j\}_{j\in[L]}$.
- $\mathbf{Rte}(\mathsf{tk}, \mathsf{ct}_{1,t}, \ldots, \mathsf{ct}_{n,t})$: For $i \in [n]$, for $j \in [L]$ where L is the length of an SE ciphertext, let $(\mathsf{ct}'_{i,t})_j := \mathsf{MCFE}^{\mathrm{ffh}}.\mathbf{Dec}(\mathsf{mpk}, \mathsf{tk}_i, (\mathsf{ct}_{1,t})_j, \ldots, (\mathsf{ct}_{n,t})_j)$, and let $\mathsf{ct}'_{i,t} := \{(\mathsf{ct}'_{i,t})_j\}_{j\in[L]}$. Output $\{\mathsf{ct}'_{i,t}\}_{i\in[n]}$.
- $\mathbf{Dec}(\mathsf{rk}_i, \mathsf{ct}'_{i,t})$: Output $x := \mathsf{SE}.\mathbf{Dec}(\mathsf{rk}_i, \mathsf{ct}'_{i,t})$.

Theorem 6.1 (NIAR with receiver-insider protection)**.** *Suppose that the* SE *scheme satisfies semantic security as defined in Appendix C.1, and that* $\mathsf{MCFE}^{\mathrm{ffh}}$ *is function-hiding IND-secure by Definition 2. Then, the NIAR scheme above is SIM-secure with receiver-insider protection by Definition 1.*

If the above theorem holds, and observing that the building blocks SE, and $\mathsf{MCFE}^{\mathrm{ffh}}$ (including the underlying CPRF) can all be instantiated with the Decisional Linear assumption in bilinear groups, we immediately have the following corollary.

Corollary 6.2 (NIAR with receiver-insider protection)**.** *Assume that the Decisional Linear assumption holds in suitable bilinear groups. Then, there exists an NIAR scheme that is SIM-secure with receiver-insider protection by Definition 1. Furthermore, the scheme has* $\mathsf{poly}(n, \kappa)$ *key size; moreover, in every time step, each sender and receiver's communication complexity is* $\mathsf{poly}(\kappa)$ *assuming each sender sends one bit per step.*

Proof of Theorem 6.1: Due to Lemma 2.1, it suffices to prove that the NIAR scheme above is IND-secure with receiver-insider protection by Definition 2. Recall that $\mathsf{NIAR\text{-}Expt}^b(1^\kappa)$ for $b \in \{0, 1\}$ were defined in Section 2.3. We will consider a following sequence of hybrid experiments to show that any non-uniform p.p.t. *admissible* adversary's views in $\mathsf{NIAR\text{-}Expt}^0(1^\kappa)$ and $\mathsf{NIAR\text{-}Expt}^1(1^\kappa)$ are computationally indistinguishable.

Experiment Hyb^b for $b \in \{0,1\}$. For $b \in \{0,1\}$, we define Hyb^b just like $\mathsf{NIAR\text{-}Expt}^b(1^\kappa)$ except with the following modification: Henceforth let $\mathsf{HH}^b \subseteq \mathcal{H}_S$ denote the set of senders who communicate with honest receivers as defined by $\pi^{(b)}$, and let $\mathsf{HK}^b = \mathcal{H}_S \backslash \mathsf{HH}^b$ denote the set of honest senders who talk to corrupt receivers as defined by $\pi^{(b)}$. Now, during the t-th **Enc** query made by the adversary $\mathcal{A}$ for $t = 1, 2, \ldots$, the query is answered as follows — below let $\mathsf{SE.Sim}$ be the simulator for the SE scheme, and let $\{x_{i,t}^{(0)}\}_{i \in \mathcal{H}}$ and $\{x_{i,t}^{(1)}\}_{i \in \mathcal{H}}$ be the two plaintext vectors submitted by $\mathcal{A}$ during time step t:

- For $i \in \mathsf{HH}^b$: let $c_i := \mathsf{SE.Sim}(1^\kappa)$, and for $j \in [L]$, let $(\mathsf{ct}_{i,t})_j := \mathsf{MCFE}^{\mathrm{ffh}}.\mathbf{Enc}(\mathsf{mpk}, \mathsf{ek}_i', (c_i)_j, (t-1)L + j)$.
- For $i \in \mathsf{HK}^b$: compute the ciphertext $\mathsf{ct}_{i,t}$ honestly using $x_{i,t}^{(b)}$ as input.

Claim 6.3. *Suppose that* SE *satisfies semantic security (and let* $\mathsf{SE.Sim}$ *be the corresponding simulator for semantic security), then for* $b \in \{0,1\}$*, any non-uniform p.p.t. adversary's views in* Hyb^b *and* $\mathsf{NIAR\text{-}Expt}^b$ *are computationally indistinguishable.*

Proof. Follows from a straightforward reduction. □

Lemma 6.4. *Suppose that the* $\mathsf{MCFE}^{\mathrm{ffh}}$ *scheme satisfies function-hiding IND-security, then no non-uniform p.p.t. adversary* $\mathcal{A}$ *that respects the admissibility rules of the NIAR IND security game (with receiver-insider protection) can distinguish* Hyb^0 *and* Hyb^1 *except with negligible probability.*

Proof. In experiment Hyb^b, let $\psi := \{\psi_i\}_{i \in \pi^{(b)}(\mathcal{H}_S)}$ where ψ_i is defined as the following:

- if $i \in \mathcal{K}_R$, ψ_i is the random coins consumed by the $\mathsf{SE.Gen}$ and $\mathsf{SE.Enc}$ algorithms pertaining to the j-th sender where $j := (\pi^{(b)})^{-1}(i)$.
- if $i \in \mathcal{H}_R$, then ψ_i is the random coins consumed by the $\mathsf{SE.Sim}$ algorithms pertaining to the j-th sender where $j := (\pi^{(b)})^{-1}(i)$.

Due to the admissibility rule $\mathcal{A}$ must abide by, $\pi^{(0)}(\mathcal{H}_S) = \pi^{(1)}(\mathcal{H}_S)$, therefore, ψ has the same format regardless of b. Let $\mathsf{Hyb}^b(\psi)$ be the experiment Hyb^b but when the aforementioned random coins are fixed to ψ. It suffices to show that for any fixed ψ, $\mathsf{Hyb}^0(\psi)$ and $\mathsf{Hyb}^1(\psi)$ are computationally indistinguishable as long as the non-uniform p.p.t.$\mathcal{A}$ respects the NIAR game's admissibility rules. We show that if a non-uniform p.p.t.$\mathcal{A}$ that respects the NIAR game's admissibility rules can distinguish $\mathsf{Hyb}^0(\psi)$ and $\mathsf{Hyb}^1(\psi)$ with non-negligible probability, we can build a reduction $\mathcal{B}$ that breaks the fully function-hiding IND-security of the $\mathsf{MCFE}^{\mathrm{ffh}}$ scheme.

- First, $\mathcal{A}$ outputs $n, \mathcal{K}_S, \mathcal{K}_R, \pi^{(0)}, \pi^{(1)}$.
- Next, $\mathcal{B}$ needs to simulate the NIAR scheme's **Setup** phase for $\mathcal{A}$. $\mathcal{B}$ obtains mpk and $\{\mathsf{ek}_i'\}_{i \in \mathcal{K}_S}$ from its own challenger, and embeds these parameters into the **Setup**.

  Next, for every $i \in [n]$, $\mathcal{B}$ makes **KGen** queries to its own challenger to generate functional secret keys for the pair of vectors $\mathbf{e}_{j_0}$ and $\mathbf{e}_{j_1}$ where for $b \in \{0,1\}$, $j_b := (\pi^{(b)})^{-1}(i)$; let the result be $\mathsf{tk} := \{\mathsf{tk}_i\}_{i \in [n]}$. $\mathcal{B}$ uses these tokens in the setup.

  For a corrupt receiver $j \in \pi^{(0)}(\mathcal{H}_S) = \pi^{(1)}(\mathcal{H}_S)$, $\mathcal{B}$ uses the appropriate part of ψ_j as the random coins to run $\mathsf{SE.Gen}$, and let the outcome be rk_j.

  For all corrupt senders $i \in \mathcal{H}_S$, run the honest $\mathsf{SE.Gen}$ and let the outcome be sk_i. If $j := \pi^{(0)}(i) = \pi^{(1)}(i)$ is corrupt, then let $\mathsf{rk}_j := \mathsf{sk}_i$.

  Now, return $(\mathsf{tk}, \{\mathsf{ek}_i := (\mathsf{mpk}, \mathsf{sk}_i, \mathsf{ek}_i')\}_{i \in \mathcal{K}_S}, \{\mathsf{rk}_i\}_{i \in \mathcal{K}_R})$ to $\mathcal{A}$.

- During each time step $t \in \mathbb{N}$, $\mathcal{A}$ submits two plaintext vectors $\{x_{i,t}^{(0)}\}_{i \in \mathcal{H}_S}$ and $\{x_{i,t}^{(1)}\}_{i \in \mathcal{H}_S}$.
  - For $i \in \mathsf{HH}^b$, and for $b \in \{0,1\}$, it computes $c_i^b := \mathsf{SE.Sim}(1^\kappa)$ using the appropriate random coins contained in $\psi_{\pi^b(i)}$.
  - For $i \in \mathsf{HK}^b$ and for $b \in \{0,1\}$, it computes $c_i^b := \mathsf{SE}.\mathbf{Enc}(\mathsf{sk}_{j_b}, x_i)$ where $j_b := \pi^{(b)}(i)$ using the appropriate random coins contained in $\psi_{\pi^{(b)}(i)}$.
  - Now, $\mathcal{B}$ forwards the two vectors $\{c_i^b\}_{i \in \mathcal{H}_S}$ for $b \in \{0,1\}$ to its own challenger bit by bit in a total of L queries. It returns the concatenated result to $\mathcal{A}$.

We verify that $\mathcal{B}$ indeed respects the admissibility rules of $\mathsf{MCFE}^{\mathsf{ffh}}$.

1. First, since $\mathcal{A}$ respects the admissiblity rules of the NIAR security game (with receiver-insider protection), the "corrupt $\to *$" part of the permutation must match for $\pi^{(0)}$ and $\pi^{(1)}$. Thus, for every pair of selection vector denoted $(\mathbf{e}_{j_0}, \mathbf{e}_{j_1})$ that $\mathcal{B}$ submits in a $\mathbf{KGen}$ query to its own challenger, if the coordinate j_0 being selected by $\mathbf{e}_{j_0}$ is corrupt, then the other selection vector must be selecting $j_1 = j_0$.
2. Checking the second admissibility rule for $\mathsf{MCFE}^{\mathsf{ffh}}$ boils down to checking that in the two alternate worlds, every receiver, no matter honest or corrupt, that receives from an honest sender must receive the same SE ciphertext. Recall that by the admissibility rule imposed on $\mathcal{A}$, if $i \in \mathcal{K}_R \cap \mathcal{H}_S$ denotes a corrupt receiver receiving from some honest sender, then i must receive the same message in the two alternate worlds. Given this, the second admissibility rule for $\mathcal{B}$ is guaranteed by construction due to our definition of ψ.

If the $\mathsf{MCFE}^{\mathsf{ffh}}$ challenger used the internal bit $b = 0$, then $\mathcal{A}$'s view is identically distributed as in $\mathsf{Hyb}^0(\psi)$; else its view is identically distributed as in $\mathsf{Hyb}^1(\psi)$. □

The proof of Theorem 6.1 now follows due to Claim 6.3 and Lemma 6.4 and the hybrid argument. □

7 Achieving Full Insider Protection

In this section, we describe how to construct a NIAR scheme with full insider protection. As mentioned earlier, to achieve full insider protection, it must be that all transformed ciphertexts output by $\mathbf{Rte}$ must change, even when only a single input ciphertext changes. This also means that a natural class of schemes like the techniques we used so far do not work.

Our idea is to rely on an indistinguishability obfuscator for probabilistic circuits (piO), whose existence is implied by sub-exponentially secure indistinguishability obfuscation [CLTV15] and sub-exponentially secure one-way functions. We will use the piO to obfuscate the following probabilistic program:

1. first, use the $\mathsf{MCFE}^{\mathsf{ffh}}$'s n functional keys to evaluate the $\mathbf{Dec}$ procedure, resulting in n messages to be received by the n receivers respectively;
2. next, encrypt the n messages under the n receivers' public keys, using a special public-key encryption scheme denoted tPKE that satisfies a "perfect trapdoor hiding" property.

Note that due to technical reasons needed in our proofs, it would not have worked had we used piO to directly obfuscate a program that calls the $\mathbf{Rte}$ procedure of our earlier NIAR scheme (with receiver-insider protection) and then encrypts the outcome messages using n instances of tPKE, respectively. Specifically, during our sequence of hybrids, we will need to at some point, remove the $\mathsf{MCFE}^{\mathsf{ffh}}$ functional keys corresponding to honest receivers from the piO.

7.1 Construction

Let $\mathsf{tPKE} := (\mathbf{Gen}, \mathbf{Enc}, \mathbf{Dec}, \widetilde{\mathbf{Gen}})$ denote a perfectly hiding trapdoor encryption scheme. Let piO be an indistinguishability obfuscator for perfectly indistinguishable samplers over general circuit families. Let $\mathsf{MCFE}^{\text{ffh}} := (\mathbf{Gen}, \mathbf{Setup}, \mathbf{KGen}, \mathbf{Enc}, \mathbf{Dec})$ denote a multi-client functional encryption scheme for selection. Specifically, we can instantiate the $\mathsf{MCFE}^{\text{ffh}}$ scheme by using the the MCFE scheme in Section 5.2, and then applying the function privacy upgrade in Sections 5.3 and 5.4 respectively. We can construct a NIAR scheme as below.

NIAR scheme with full insider security

- $\mathbf{Setup}(1^\kappa, n, \pi)$:
  - Let $\mathsf{pp} := \mathsf{MCFE}^{\text{ffh}}.\mathbf{Gen}(1^\kappa)$, and $(\mathsf{mpk}, \mathsf{msk}, \{\mathsf{ek}'_i\}_{i\in[n]}) := \mathsf{MCFE}^{\text{ffh}}.\mathbf{Setup}(\mathsf{pp}, 1, n)$.
  - For $i \in [n]$, let $(\mathsf{epk}_i, \mathsf{esk}_i) \leftarrow \mathsf{tPKE}.\mathbf{Gen}(1^\kappa)$.
  - For $i \in [n]$, let $j := \pi^{-1}(i)$, and let $\mathbf{e}_j$ be the j-th selection vector, i.e., except the j-th coordinate which is 1, all other coordinates are 0; call $\mathsf{tk}_i := \mathsf{MCFE}^{\text{ffh}}.\mathbf{KGen}(\mathsf{mpk}, \mathsf{msk}, \mathbf{e}_j)$.
  - Output

$$\mathsf{tk} := \mathsf{piO}(1^\kappa, P^{\mathsf{mpk}, \{\mathsf{epk}_i, \mathsf{tk}_i\}_{i\in[n]}}), \ \{\forall i \in [n] : \mathsf{ek}_i := (\mathsf{mpk}, \mathsf{ek}'_i), \ \mathsf{rk}_i := \mathsf{esk}_i\}$$

    where the program P is defined below.

- $\mathbf{Enc}(\mathsf{ek}_i, x_i, t)$: output $\mathsf{ct}_{i,t} := \mathsf{MCFE}^{\text{ffh}}.\mathbf{Enc}(\mathsf{mpk}, \mathsf{ek}'_i, x_i, t)$.
- $\mathbf{Rte}(\mathsf{tk}, \mathsf{ct}_{1,t}, \mathsf{ct}_{2,t}, \ldots, \mathsf{ct}_{n,t})$: output $(\mathsf{ct}'_{1,t}, \mathsf{ct}'_{2,t}, \ldots, \mathsf{ct}'_{n,t}) := \mathsf{tk}(\mathsf{ct}_{1,t}, \mathsf{ct}_{2,t}, \ldots, \mathsf{ct}_{n,t})$.
- $\mathbf{Dec}(\mathsf{rk}_i, \mathsf{ct}'_{i,t})$: output $\mathsf{tPKE}.\mathbf{Dec}(\mathsf{esk}_i, \mathsf{ct}'_{i,t})$.

Probabilistic program $P^{\mathsf{mpk}, \{\mathsf{epk}_i, \mathsf{tk}_i\}_{i\in[n]}}(\mathsf{ct}_1, \ldots, \mathsf{ct}_n)$

Hardwired: $\mathsf{mpk}, \{\mathsf{epk}_i, \mathsf{tk}_i\}_{i\in[n]}$

- For $i \in [n]$, let $x_i := \mathsf{MCFE}^{\text{ffh}}.\mathbf{Dec}(\mathsf{mpk}, \mathsf{tk}_i, \mathsf{ct}_1, \ldots, \mathsf{ct}_n)$.
- For $i \in [n]$, let $\widehat{\mathsf{ct}}_i := \mathsf{tPKE}.\mathbf{Enc}(\mathsf{epk}_i, x_i)$.
- Output $\{\widehat{\mathsf{ct}}_i\}_{i\in[n]}$.

Theorem 7.1 (NIAR scheme with full insider security)**.** *Suppose that* tPKE *is a perfectly hiding trapdoor encryption scheme,* piO *is an indistinguishability obfuscator for perfectly indistinguishable samplers over general circuit families, and moreover,* $\mathsf{MCFE}^{\text{ffh}}$ *is fully function-hiding IND-secure. Then, the above construction is SIM-secure with full insider protection.*

Proof. The proof is presented in Section 7.2. □

Corollary 7.2 (NIAR scheme with full insider security)**.** *Assume the existence of sub-exponentially secure indistinguishability obfuscator for general circuit families, sub-exponentially secure one-way functions, and that the Decisional Linear assumption (with standard polynomial security) holds in suitable bilinear groups. Then, there exists a SIM-secure NIAR scheme with full insider protection, in which the sizes of all keys are upper bounded by* $\mathsf{poly}(n, \kappa)$*, and moreover, in each time step, every sender or receiver's communication overhead is only* $\mathsf{poly}(\kappa)$*.*

Proof. We can instantiate the $\mathsf{MCFE}^{\mathrm{ffh}}$ scheme employed by the NIAR construction earlier in this section as follows: use the the MCFE scheme in Section 5.2, and then applying the function privacy upgrade in Sections 5.3 and 5.4 respectively. Note that the existence of tPKE is implied by the Decisional Linear assumption as mentioned in Theorem 4.4. Now, the corollary follows directly from Theorem 7.1, and the efficiency claims can be easily verified. □

7.2 Proof of Theorem 7.1

Due to Lemma 2.1, it suffices to prove that the construction in Section 7.1 is IND-secure with full insider protection. We will prove this through a sequence of hybrid experiments.

Experiment Hyb_1. $\mathsf{Hyb}_1(1^\kappa)$ is almost the same as $\mathsf{NIAR\text{-}Expt}^0(1^\kappa)$ except that during **Setup**, for every $i \in \mathcal{H}_R$, we run the trapdoor key generation of the tPKE scheme to generate its public key. In other words, for $i \in \mathcal{H}_R$, let $\mathsf{epk}_i \leftarrow \mathsf{tPKE}.\widetilde{\mathbf{Gen}}(1^\kappa)$.

Claim 7.3. *Suppose that* tPKE *is a perfectly hiding trapdoor encryption scheme. Then, any non-uniform p.p.t. adversary* $\mathcal{A}$*'s views in* $\mathsf{NIAR\text{-}Expt}^0$ *and* Hyb_1 *are computationally indistinguishable.*

Proof. Through a straightforward reduction to the tPKE's security, and specifically, the indistinguishability of the normal and the trapdoor modes. □

Experiment Hyb_2. Hyb_2 is almost the same as Hyb_1 except with the following modification: during **Setup**, instead of calling $\mathsf{tk} := \mathsf{piO}(1^\kappa, P^{\mathsf{mpk},\{\mathsf{epk}_i,\mathsf{tk}_i\}_{i\in[n]}})$, we call

$$\mathsf{tk} := \mathsf{piO}\left(1^\kappa, \widetilde{P}^{\mathsf{mpk},\{\mathsf{epk}_i\}_{i\in[n]},\{\mathsf{tk}_i\}_{i\in\mathcal{K}_R}}\right)$$

where the program $\widetilde{P}^{\mathsf{mpk},\{\mathsf{epk}_i\}_{i\in[n]},\{\mathsf{tk}_i\}_{i\in\mathcal{K}_R}}$ is defined below.

Probabilistic program $\widetilde{P}^{\mathsf{mpk},\{\mathsf{epk}_i\}_{i\in[n]},\{\mathsf{tk}_i\}_{i\in\mathcal{K}_R}}(\mathsf{ct}_1,\ldots,\mathsf{ct}_n)$

Hardwired: $\mathsf{mpk}, \{\mathsf{epk}_i\}_{i\in[n]}, \{\mathsf{tk}_i\}_{i\in\mathcal{K}_R}$

- For $i \in \mathcal{K}_R$, let $x_i := \mathsf{MCFE}^{\mathrm{ffh}}.\mathbf{Dec}(\mathsf{mpk}, \mathsf{tk}_i, \mathsf{ct}_1, \ldots, \mathsf{ct}_n)$; and let $\widehat{\mathsf{ct}}_i := \mathsf{tPKE}.\mathbf{Enc}(\mathsf{epk}_i, x_i)$.
- For $i \in \mathcal{H}_R$: let $\widehat{\mathsf{ct}}_i := \mathsf{tPKE}.\mathbf{Enc}(\mathsf{epk}_i, 0)$.
- Output $\{\widehat{\mathsf{ct}}_i\}_{i\in[n]}$.

Observe that in Hyb_2, during **Setup**, in fact, for any $i \in \mathcal{H}_R$, there is in fact no need to call $\mathsf{tk}_i := \mathsf{MCFE}^{\mathrm{ffh}}.\mathbf{KGen}(\mathsf{mpk}, \mathsf{msk}, \mathbf{e}_j)$ where $j := \pi^{-1}(i)$, since $\{\mathsf{tk}_i\}_{i\in\mathcal{H}_R}$ will never be used later.

Claim 7.4. *Suppose that* tPKE *is a perfectly hiding trapdoor encryption scheme, and that* piO *is an indistinguishability obfuscator for perfectly indistinguishable samplers over general circuit families. Then, any non-uniform p.p.t. adversary* $\mathcal{A}$*'s views in* Hyb_1 *and* Hyb_2 *are computationally indistinguishable.*

Proof. Due to the security of the tPKE scheme, specifically, due to the perfectly hiding property of the trapdoor mode, for any $i \in \mathcal{H}_R$, a ciphertext generated by creating a fresh encryption of 0 is identically distributed as a ciphertext generated by encrypting x_i. The indistinguishability of Hyb_1 and Hyb_2 therefore follows from the security of piO. □

Experiment Hyb_3. Hyb_3 is defined almost identically as Hyb_2 except that now, the NIAR challenger switches to using $\pi^{(1)}$ during **Setup** and to using $\{x_{i,t}^{(1)}\}_{i\in\mathcal{H}_S}$ for answering the online routing queries.

Claim 7.5. *Suppose that* $\mathsf{MCFE}^{\mathsf{ffh}}$ *is fully function-hiding IND-secure. Then, no non-uniform p.p.t. adversary* $\mathcal{A}$ *that respects the admissibility rules of the NIAR IND-security game can distinguish* Hyb_2 *and* Hyb_3 *except with negligible probability.*

Proof. If there is a non-uniform p.p.t. adversary $\mathcal{A}$ that respects the admissibility rules of the NIAR IND-security game who can distinguish Hyb_2 and Hyb_3 with more than negligible probability, we can build a reduction $\mathcal{B}$ that breaks the fully function-hiding IND-security of the $\mathsf{MCFE}^{\mathsf{ffh}}$ scheme.

- First, the adversary $\mathcal{A}$ outputs $n, \mathcal{K}_S, \mathcal{K}_R, \pi^{(0)}$, and $\pi^{(1)}$.
- Now, $\mathcal{B}$ needs to run **Setup**: to do so, it obtains mpk and $\{\mathsf{ek}_i'\}_i \in \mathcal{K}_S$ from its own challenger, and embeds these parameters into the **Setup**. $\mathcal{B}$ runs the tPKE key generation honestly for $i \in \mathcal{K}_R$ but calls the trapdoor key generation for $i \in \mathcal{H}_R$, just like in Hyb_2.

  Next, for every $i \in \mathcal{K}_R$, $\mathcal{B}$ makes **KGen** queries to its own challenger to generate functional secret keys for the pair of vectors $\mathbf{e}_{j_0}$ and $\mathbf{e}_{j_1}$ where for $b \in \{0,1\}$, $j_b := (\pi^{(b)})^{-1}$; let the result be $\{\mathsf{tk}_i\}_{i\in\mathcal{K}_R}$.

  The rest of **Setup** is performed just like in Hyb_2.
- Next, for each time step $t = 1, 2, \ldots$, $\mathcal{A}$ submits two plaintext vectors $\{x_{i,t}^{(0)}\}_{i\in\mathcal{H}_S}$ and $\{x_{i,t}^{(1)}\}_{i\in\mathcal{H}_S}$. $\mathcal{B}$ forwards the two vectors to its own challenger, and forwards the answer to $\mathcal{A}$.

Below, we verify that $\mathcal{B}$ indeed respects the admissibility rules of the $\mathsf{MCFE}^{\mathsf{ffh}}$'s security game:

1. First, since $\mathcal{A}$ respects the admissibility rules of the NIAR game, it must be that $\pi^{(0)}$ and $\pi^{(1)}$ are consistent when restricting to the "corrupt $\to$ corrupt" part of the permutation. This means that for every $i \in \mathcal{K}_R$ — consider the corresponding pair of selection vectors $\mathbf{e}_{j_0}$ and $\mathbf{e}_{j_1}$ where for $b \in \{0,1\}$, $j_b := (\pi^{(b)})^{-1}$ — it must be that either 1) $\mathbf{e}_{j_0}$ and $\mathbf{e}_{j_1}$ both do not select from any coordinate belonging to $\mathcal{K}_S$; or 2) they both select the same coordinate belonging to $\mathcal{K}_S$.
2. By the NIAR game's admissibility rule imposed on $\mathcal{A}$, during any time step $t \in \mathbb{N}$, for any $i \in \mathcal{K}_R \cap \pi^{(0)}(\mathcal{H}_S) = \mathcal{K}_R \cap \pi^{(1)}(\mathcal{H}_S)$, $x_{j_0,t}^{(0)} = x_{j_1,t}^{(1)}$ where for $b \in \{0,1\}$, $j_b := (\pi^{(b)})^{-1}(i)$. In other words, in the two alternate worlds $b = 0$ or 1, every corrupt receiver receiving from an honest sender must receive the same message. This means that for every selection vector $\mathcal{B}$ submitted as a **KGen** query that selects one of the corrupt receiver's output, it must produce the same output in the two alternate worlds.

Now, if the $\mathsf{MCFE}^{\mathsf{ffh}}$ challenger used $b = 0$, then $\mathcal{A}$'s view is identically distributed as in Hyb_2; else $\mathcal{A}$'s view is identically distributed as in Hyb_3. □

Completing the proof of Theorem 7.1. Finally, using a symmetric argument as the above, we can prove that no non-uniform p.p.t. adversary that respects the admissibility rules of the NIAR game can distinguish Hyb_3 and $\mathsf{NIAR\text{-}Expt}^1$ except with negligible probability.

8 Fault-Tolerant NIAR

In real-world applications, if some senders fail to show up, we would like the router to nonetheless be able to make a best effort at routing the remaining messages. We therefore propose a fault-tolerant variant of our earlier NIAR abstraction, which is in fact also a generalization of our earlier NIAR abstraction.

8.1 Definitions

Suppose that at some point, some senders crash and fail to show up henceforth. Let $\mathcal{O}$ denote the set of senders that remain online. Henceforth we assume that there is a mechanism in place for the senders to discover the set $\mathcal{O}$. For example, in the anonymous bulletin board application (see Section 1.4), everyone can easily observe which senders are still online. In a "distributed differential privacy in the shuffle-model" scenario such as RAPPOR [EPK14] (see Section 1.4), the data collector is voluntarily implementing the privacy mechanism due to compliance, regulation, or to waive liability; and therefore the data collector can be entrusted to distribute the online list $\mathcal{O}$. Note also that if a sender that crashed comes back online, it can also be added back to set $\mathcal{O}$ again.

Once the set $\mathcal{O}$ of online senders is known, the online senders will encrypt messages in every time step to the set $\mathcal{O}$. In this way, the router can perform successful decryption using only ciphertexts from those in $\mathcal{O}$.

Before jumping into formal definitions and constructions, we point out that our fault-tolerant NIAR abstraction can also be combined with other external mechanisms to provide resilience against potential faults. For example, if the scheme is deployed on a blockchain (e.g., imagine a pseudonymous bulletin board on a blockchain), then a promising approach is to use reward/penalty mechanisms to incentivize every sender to speak in every time step.

8.1.1 Syntax

We can formally define a fault-tolerant NIAR by modifying our original definition slightly, where the **Setup** and **Dec** algorithms are still as before, but the **Enc** and **Rte** algorithms now additionally take a set $\mathcal{O} \subseteq [n]$; and moreover, the **Rte** algorithm now takes in ciphertexts created by only those in $\mathcal{O}$.

Correctness. In the new formulation, obviously receivers paired with senders not in $\mathcal{O}$ do not have any messages to receive. We therefore modify our correctness notion to require that receivers speaking with those in $\mathcal{O}$ receive the correct messages, whereas receivers paired with those not in $\mathcal{O}$ receive a canonical message, e.g., either $\perp$ or 0.

Our new correctness requirement stipulates that with probability 1, the following holds: for any $\kappa \in \mathbb{N}$, any $(x_1, x_2, \ldots, x_n) \in \{0,1\}^n$, any $t \in \mathbb{N}$, and any $\mathcal{O} \subseteq [n]$: let $(\{\mathsf{ek}_i\}_{i\in[n]}, \{\mathsf{rk}_i\}_{i\in[n]}, \mathsf{tk}) \leftarrow \mathbf{Setup}(1^\kappa, n, \pi)$, let $\mathsf{ct}_{i,t} \leftarrow \mathbf{Enc}(\mathsf{ek}_i, x_i, t, \mathcal{O})$ for $i \in \mathcal{O}$, let $(\mathsf{ct}'_{1,t}, \mathsf{ct}'_{2,t}, \ldots, \mathsf{ct}'_{n,t}) \leftarrow \mathbf{Rte}(\mathsf{tk}, \mathcal{O}, \{\mathsf{ct}_{i,t}\}_{i\in\mathcal{O}})$, and let $x'_i \leftarrow \mathbf{Dec}(\mathsf{rk}_i, \mathsf{ct}'_{i,t})$, it must be that

$$x'_{\pi(i)} = \begin{cases} x_i & \text{if } i \in \mathcal{O} \\ \perp & \text{o.w.} \end{cases}$$

8.1.2 Security Definitions

It is interesting to consider how to define the security of a fault-tolerant NIAR. If an honest sender $i \in [n]$ is speaking with a corrupt receiver, then the adversary can make i go offline in some round, and observe if any corrupt receiver j starts to receive the canonical value $\perp$ henceforth. In this way, the adversary can discover that i is speaking with j. Therefore, in a fault-tolerant NIAR scheme, if an honest sender i speaking with a corrupt receiver ever goes offline, leaking $\pi(i)$ is inherent and cannot be avoided.

We now modify our earlier security notion to the fault tolerant setting. Henceforth, let $\mathcal{C}_S \subseteq \mathcal{H}_S$ denote the set of senders who are honest but can potentially crash at some point. To allow fault tolerance, the strongest possible notion is the following, which reveals only the inherent leakage to an adversary controlling the router and a subset of senders and receivers: such an adversary learns only the "$\mathcal{K}_S \cup \mathcal{C}_S \to \mathcal{K}_R$" part of the permutation and the messages received by corrupt receivers in every time step, but nothing else. In our definitions below, we will also consider a couple relaxed notions.

More formally, we can consider the following experiments:

Real-world experiment $\mathsf{Real}^{\mathcal{A}}(1^\kappa)$. Recall that $\mathcal{K}_S \subseteq [n]$ denotes the set of corrupt senders, and let $\mathcal{H}_S = [n] \setminus \mathcal{K}_S$ be the honest senders. The set $\mathcal{C}_S \subseteq \mathcal{H}_S$ denotes the set of honest but occasionally offline senders.

- $n, \pi, \mathcal{C}_S, \mathcal{K}_S, \mathcal{K}_R \leftarrow \mathcal{A}(1^\kappa)$
- $(\{\mathsf{ek}_i\}_{i\in[n]}, \{\mathsf{rk}_i\}_{i\in[n]}, \mathsf{tk}) \leftarrow \mathbf{Setup}(1^\kappa, n, \pi)$
- For $t = 1, 2, \ldots$:
  - if $t = 1$ then $(\mathcal{O}_t, \{x_{i,t}\}_{i\in\mathcal{O}_t\cap\mathcal{H}_S}) \leftarrow \mathcal{A}(\mathsf{tk}, \{\mathsf{ek}_i\}_{i\in\mathcal{K}_S}, \{\mathsf{rk}_i\}_{i\in\mathcal{K}_R})$;
    else $(\mathcal{O}_t, \{x_{i,t}\}_{i\in\mathcal{O}_t\cap\mathcal{H}_S}) \leftarrow \mathcal{A}(\{\mathsf{ct}_{i,t-1}\}_{i\in\mathcal{O}_{t-1}\cap\mathcal{H}_S})$;
  - for $i \in \mathcal{O}_t \cap \mathcal{H}_S$, $\mathsf{ct}_{i,t} \leftarrow \mathbf{Enc}(\mathsf{ek}_i, x_{i,t}, t, \mathcal{O}_t)$

Ideal-world experiment $\mathsf{Ideal}^{\mathcal{A},\mathsf{Sim}}(1^\kappa)$. The ideal-world experiment involves not just $\mathcal{A}$, but also a p.p.t. (stateful) simulator denoted Sim, who is in charge of simulation $\mathcal{A}$'s view knowing essentially only what corrupt senders and receivers know.

- $n, \pi, \mathcal{C}_S, \mathcal{K}_S, \mathcal{K}_R \leftarrow \mathcal{A}(1^\kappa)$
- $(\{\mathsf{ek}_i\}_{i\in[n]}, \{\mathsf{rk}_i\}_{i\in[n]}, \mathsf{tk}) \leftarrow \mathsf{Sim}(1^\kappa, n, \mathcal{K}_S, \mathcal{K}_R, \mathsf{Leak}(\pi, \mathcal{C}_S, \mathcal{K}_S, \mathcal{K}_R))$
- For $t = 1, 2, \ldots$:
  - if $t = 1$ then $(\mathcal{O}_t, \{x_{i,t}\}_{i\in\mathcal{O}_t\cap\mathcal{H}_S}) \leftarrow \mathcal{A}(\mathsf{tk}, \{\mathsf{ek}_i\}_{i\in\mathcal{K}_S}, \{\mathsf{rk}_i\}_{i\in\mathcal{K}_R})$;
    else $(\mathcal{O}_t, \{x_{i,t}\}_{i\in\mathcal{O}_t\cap\mathcal{H}_S}) \leftarrow \mathcal{A}(\{\mathsf{ct}_{i,t-1}\}_{i\in\mathcal{O}_{t-1}\cap\mathcal{H}_S})$;
  - $\{\mathsf{ct}_{i,t}\}_{i\in\mathcal{O}_t} \leftarrow \mathsf{Sim}\left(\mathcal{O}_t, \{\forall i \in \mathcal{K}_R \cap \pi(\mathcal{O}_t \cap \mathcal{H}_S) : (i, x_{j,t}) \text{ for } j = \pi^{-1}(i)\}\right)$.

We say that $\mathcal{A}$ is *admissible* iff with probability 1, it holds that $\mathcal{C}_S \cap \mathcal{K}_S = \emptyset$, and moreover, for every $t \in \mathbb{N}$, $\mathcal{H}_S \setminus \mathcal{O}_t \subseteq \mathcal{C}_S$. In other words, the adversary must promise ahead of time which subset of honest senders $\mathcal{C}_S$ may crash, and in each round, only those in $\mathcal{C}_S$ (but not necessarily all of them) can crash.

We present a few natural ways to define the leakage function:

1. As mentioned, the inherent, unavoidable leakage is the following:

$$\mathsf{Leak}^{\min}(\pi, \mathcal{C}_S, \mathcal{K}_S, \mathcal{K}_R) := \{\forall i \in \mathcal{K}_S \cup \mathcal{C}_S \text{ and } \pi(i) \in \mathcal{K}_R : (i, \pi(i))\}$$

   This notion protects against both sender- and receiver-insiders.

2. We consider a relaxation that provides only receiver-insider protection, i.e., each sender knows who it is speaking with, and the sender becomes corrupt, the adversary learns what the sender knows:

$$\mathsf{Leak}^{\mathrm{S}}(\pi, \mathcal{C}_S, \mathcal{K}_S, \mathcal{K}_R) := \{\forall i \in \mathcal{K}_S : (i, \pi(i))\} \cup \{\forall i \in \mathcal{C}_S \text{ and } \pi(i) \in \mathcal{K}_R : (i, \pi(i))\}$$

3. We give another natural relaxation that essentially provides receiver-insider protection, but additionally leaking who each crashed sender is speaking to, i.e.,

$$\mathsf{Leak}^{\mathrm{S}^*}(\pi, \mathcal{C}_S, \mathcal{K}_S, \mathcal{K}_R) := \{\forall i \in \mathcal{K}_S \cup \mathcal{C}_S : (i, \pi(i))\}$$

Definition 8 (Fault-tolerant NIAR)**.** We say that a fault-tolerant NIAR scheme is SIM-secure w.r.t. the leakage Leak iff, assuming that Leak is the leakage function used in the experiments, there exists a p.p.t. simulator Sim such that for any non-uniform p.p.t. admissible adversary $\mathcal{A}$, $\mathcal{A}$'s view in $\mathsf{Real}^{\mathcal{A}}(1^\kappa)$ and $\mathsf{Ideal}^{\mathcal{A},\mathsf{Sim}}(1^\kappa)$ are computationally indistinguishable.

Just like before, we can alternatively define security using an indistinguishability-based approach. In this case, we consider the following experiment $\mathsf{NIAR\text{-}Expt}^{b,\mathcal{A}}(1^\kappa)$ parametrized by a bit $b \in \{0, 1\}$:

- $n, \mathcal{C}_S, \mathcal{K}_S, \mathcal{K}_R, \pi^{(0)}, \pi^{(1)} \leftarrow \mathcal{A}(1^\kappa)$
- $(\{\mathsf{ek}_i\}_{i\in[n]}, \{\mathsf{rk}_i\}_{i\in[n]}, \mathsf{tk}) \leftarrow \mathbf{Setup}(1^\kappa, n, \pi^{(b)})$
- For $t = 1, 2, \ldots$:
  - if $t = 1$ then $(\mathcal{O}_t, \{x_{i,t}^{(0)}\}_{i\in\mathcal{O}_t\cap\mathcal{H}_S}, \{x_{i,t}^{(1)}\}_{i\in\mathcal{O}_t\cap\mathcal{H}_S}) \leftarrow \mathcal{A}(\mathsf{tk}, \{\mathsf{ek}_i\}_{i\in\mathcal{K}_S}, \{\mathsf{rk}_i\}_{i\in\mathcal{K}_R})$;
    else $(\mathcal{O}_t, \{x_{i,t}^{(0)}\}_{i\in\mathcal{O}_t\cap\mathcal{H}_S}, \{x_{i,t}^{(1)}\}_{i\in\mathcal{O}_t\cap\mathcal{H}_S}) \leftarrow \mathcal{A}(\{\mathsf{ct}_{i,t-1}\}_{i\in\mathcal{O}_{t-1}\cap\mathcal{H}_S})$;
  - for $i \in \mathcal{O}_t \cap \mathcal{H}_S$, $\mathsf{ct}_{i,t} \leftarrow \mathbf{Enc}(\mathsf{ek}_i, x_{i,t}^{(b)}, t, \mathcal{O}_t)$

In the above, Leak can be $\mathsf{Leak}^{\min}$, $\mathsf{Leak}^{\mathrm{S}}$, or $\mathsf{Leak}^{\mathrm{S}^*}$ as explained earlier. We say that $\mathcal{A}$ is *admissible* iff with probability 1, it guarantees that

1. $\mathcal{C}_S \cap \mathcal{K}_S = \emptyset$, and moreover, for every $t \in \mathbb{N}$, $\mathcal{H}_S \backslash \mathcal{O}_t \subseteq \mathcal{C}_S$;
2. $\mathsf{Leak}(\pi^{(0)}, \mathcal{C}_S, \mathcal{K}_S, \mathcal{K}_R) = \mathsf{Leak}(\pi^{(1)}, \mathcal{C}_S, \mathcal{K}_S, \mathcal{K}_R)$;
3. for any $i \in \mathcal{K}_R \cap \pi^{(0)}(\mathcal{O}_t \cap \mathcal{H}_S) = \mathcal{K}_R \cap \pi^{(1)}(\mathcal{O}_t \cap \mathcal{H}_S)$, it holds that $x_{j_0,t}^{(0)} = x_{j_1,t}^{(1)}$ where for $b \in \{0, 1\}$, $j_b := (\pi^{(b)})^{-1}(i)$.

Definition 9 (Alternative definition of fault-tolerant security with receiver-insider protection)**.** We say that a fault-tolerant NIAR scheme is IND-secure w.r.t. the leakage Leak, iff, assuming that Leak is used in the experiments, the following holds: for any non-uniform p.p.t. admissible $\mathcal{A}$, its views in the above experiments $\mathsf{NIAR\text{-}Expt}^{0,\mathcal{A}}(1^\kappa)$ and $\mathsf{NIAR\text{-}Expt}^{1,\mathcal{A}}(1^\kappa)$ are computationally indistinguishable.

Lemma 8.1 (Equivalence of the two notions)**.** *For $a \in \{\mathrm{S}, \mathrm{S}^*, \min\}$, A fault-tolerant NIAR scheme is SIM-secure w.r.t. Leak^a iff it is IND-secure w.r.t. Leak^a.*

Proof. □

8.2 Fault-Tolerant NIAR with Receiver-Insider Protection

We modify the scheme in Section 6 to one that is fault tolerant. Below we sketch the scheme, and defer a formal description and proofs to Appendix B. In our fault-tolerant scheme, **Setup** and **Dec** work in the same manner as in Section 6, except that we now additionally assume that decrypting an SE ciphertext of $\mathbf{0}$ gives $\perp$. The only modification needed is to the **Enc** and **Rte** algorithms to account for the online set $\mathcal{O}$:

- $\mathbf{Enc}(\mathsf{ek}_i, x_i, t, \mathcal{O})$: the encryption algorithm is the same as in Section 6, except that we replace the randomizing term $\mathsf{CPRF}.\mathbf{Eval}(K_i, t) := \sum_{j \neq i} (-1)^{j<i} \cdot \mathsf{PRF}_{k_{ij}}(t)$ with the following:

$$\mathsf{CPRF}.\mathbf{Eval}(K_i, t, \mathcal{O}) := \sum_{j \neq i, j \in \mathcal{O}} (-1)^{j<i} \cdot \mathsf{PRF}_{k_{ij}}(t)$$

- $\mathbf{Rte}(\mathsf{tk}, \mathcal{O}, \{\mathsf{ct}_{i,t}\}_{i \in \mathcal{O}})$: When evaluating the **Rte** function, for each of the n tokens: the router now uses only the components corresponding to those in $\mathcal{O}$ to perform the decryption. Recall that to decrypt the transformed ciphertext for each receiver, the router previously had to compute n partial decryptions by calling $\mathsf{FE}.\mathbf{Dec}$, and then multiply all the partial decryptions together. Now, the router computes the partial decryptions only for the set $\mathcal{O}$, and multiplies the partial decryptions together.

One can verify that the above scheme still preserves correctness: if a receiver is speaking with a sender who is still online, then it will decrypt the correct message; however, if a receiver is speaking with a sender no longer in $\mathcal{O}$, it will receive an SE ciphertext $\mathbf{0}$, and decrypting this ciphertext will give $\perp$. For this reason, this scheme also leaks the receivers that are paired with crashed senders, because those receivers would hear an SE ciphertext of $\mathbf{0}$.

Theorem 8.2 (Fault-tolerant NIAR with receiver-insider protection)**.** *Assume that the Decisional Linear assumption holds in suitable bilinear groups. Then, there exists a fault-tolerant NIAR scheme that is SIM-secure w.r.t. the leakage function* $\mathsf{Leak}^{\mathrm{S}^*}$*. Furthermore, the scheme has* $\mathsf{poly}(n, \kappa)$ *key size; moreover, in every time step, each sender and receiver's communication complexity is* $\mathsf{poly}(\kappa)$ *assuming each sender sends one bit per step.*

Proof. In the above, we sketched our fault-tolerant NIAR scheme slightly informally. The detailed constructions and proofs of this theorem are deferred to Section B in the appendices. □

8.3 Fault-Tolerant NIAR with Full Insider Protection

The fault-tolerant construction in Section 8.2 provides only receiver-insider protection, and it additionally leaks the receivers corresponding to crashed senders. We now discuss how to upgrade the protocol's security to full insider protection. In other words, we want to prove security with respect to the minimum, inherent leakage $\mathsf{Leak}^{\mathrm{min}}$ defined in Section 8.1.2.

Our idea is to leverage an indistinguishable obfuscator for probabilistic circuits (piO) in a similar fashion as in Section 7. Let $\mathsf{tPKE} := (\mathbf{Gen}, \mathbf{Enc}, \mathbf{Dec}, \widetilde{\mathbf{Gen}})$ denote a perfectly hiding trapdoor encryption scheme. Let piO be an indistinguishability obfuscator for perfectly indistinguishable samplers over general circuit families. Let $\mathsf{MCFE}^{\mathrm{ffh}} := (\mathbf{Gen}, \mathbf{Setup}, \mathbf{KGen}, \mathbf{Enc}, \mathbf{Dec})$ denote the MCFE-for-selection scheme with full function privacy described in Section B.1.2. We can construct a fault-tolerant NIAR scheme as below.

Fault-tolerant NIAR scheme with full insider security

- **Setup**$(1^\kappa, n, \pi)$:
  - Let $\mathsf{pp} := \mathsf{MCFE}^{\mathrm{ffh}}.\mathbf{Gen}(1^\kappa)$, and $(\mathsf{mpk}, \mathsf{msk}, \{\mathsf{ek}'_i\}_{i\in[n]}) := \mathsf{MCFE}^{\mathrm{ffh}}.\mathbf{Setup}(\mathsf{pp}, 1, n)$.
  - For $i \in [n]$, let $(\mathsf{epk}_i, \mathsf{esk}_i) \leftarrow \mathsf{tPKE}.\mathbf{Gen}(1^\kappa)$.
  - For $i \in [n]$, let $j := \pi^{-1}(i)$, and let $\mathbf{e}_j$ be the j-th selection vector, i.e., except the j-th coordinate which is 1, all other coordinates are 0; call $\mathsf{tk}_i := \mathsf{MCFE}^{\mathrm{ffh}}.\mathbf{KGen}(\mathsf{mpk}, \mathsf{msk}, \mathbf{e}_j)$.
  - Output

$$\mathsf{tk} := \mathsf{piO}(1^\kappa, P^{\mathsf{mpk}, \{\mathsf{epk}_i, \mathsf{tk}_i\}_{i\in[n]}}), \quad \{\forall i \in [n] : \mathsf{ek}_i := (\mathsf{mpk}, \mathsf{ek}'_i),\ \mathsf{rk}_i := \mathsf{esk}_i\}$$

    where the program P is defined below.

- **Enc**$(\mathsf{ek}_i, x_i, t, \mathcal{O})$: output $\mathsf{ct}_{i,t} := \mathsf{MCFE}^{\mathrm{ffh}}.\mathbf{Enc}(\mathsf{mpk}, \mathsf{ek}'_i, x_i, t, \mathcal{O})$.
- **Rte**$(\mathsf{tk}, \mathcal{O}, \{\mathsf{ct}_{i,t}\}_{i\in\mathcal{O}})$: for $i \notin \mathcal{O}$, let $\mathsf{ct}_{i,t} = \bot$, and output

$$(\mathsf{ct}'_{1,t}, \mathsf{ct}'_{2,t}, \ldots, \mathsf{ct}'_{n,t}) := \mathsf{tk}(\mathsf{ct}_{1,t}, \mathsf{ct}_{2,t}, \ldots, \mathsf{ct}_{n,t}, \mathcal{O})$$

- **Dec**$(\mathsf{rk}_i, \mathsf{ct}'_{i,t})$: output $\mathsf{tPKE}.\mathbf{Dec}(\mathsf{esk}_i, \mathsf{ct}'_{i,t})$.

Probabilistic program $P^{\mathsf{mpk}, \{\mathsf{epk}_i, \mathsf{tk}_i\}_{i\in[n]}}(\mathsf{ct}_1, \ldots, \mathsf{ct}_n, \mathcal{O})$

Hardwired: $\mathsf{mpk}, \{\mathsf{epk}_i, \mathsf{tk}_i\}_{i\in[n]}$

- For $i \in [n]$, let $x_i := \mathsf{MCFE}^{\mathrm{ffh}}.\mathbf{Dec}(\mathsf{mpk}, \mathsf{tk}_i, \mathcal{O}, \{\mathsf{ct}_i\}_{i\in\mathcal{O}})$.
- For $i \in [n]$, let $\widehat{\mathsf{ct}}_i := \mathsf{tPKE}.\mathbf{Enc}(\mathsf{epk}_i, x_i)$.
- Output $\{\widehat{\mathsf{ct}}_i\}_{i\in[n]}$.

Theorem 8.3 (Fault-tolerant NIAR scheme with full insider security)**.** *Suppose that* tPKE *is a perfectly hiding trapdoor encryption scheme,* piO *is an indistinguishability obfuscator for perfectly indistinguishable samplers over general circuit families, and moreover,* $\mathsf{MCFE}^{\mathrm{ffh}}$ *is fully function-hiding IND-secure. Then, the above fault-tolerant NIAR construction is SIM-secure with full insider protection, i.e., it is SIM-secure w.r.t. the leakage* $\mathsf{Leak}^{\min}$ *defined in Section 8.1.2.*

Proof. The proof ideas are similar to those of Section 7 but become more involved now due to the need to reason about the online set $\mathcal{O}$. We defer the full proofs to Section B.3 in the appendices. □

9 Conclusion and Open Questions

Our paper is an initial exploration of the NIAR abstraction, and we reveal various subtleties in terms of definitions as well as construction. Many open questions arise given our new abstractions, for example:

1. Can we construct NIAR with *full* insider protection from standard assumptions? Is full insider protection inherently more challenging than receiver-insider protection?

2. Can we improve or get rid of the $\mathsf{poly}(\kappa)$ blowup in communication, and construct a constant-rate or rate-1 NIAR scheme, similar to the notion of rate-1 in the context of fully homomorphic encryption [BDGM19, GH19]?
3. Can we design a concretely efficient multi-party computation protocol to realize the one-time trusted setup? Can we make this decentralized setup procedure less interactive?
4. As mentioned, our schemes with receiver-insider protection are potentially implementable; so another interesting question is whether we can further improve the concrete performance of NIAR and make it practical in real-world applications.

Acknowledgments

Elaine Shi would like to thank Isaac Sheff for insightful discussions which inspired her to formulate and work on this problem. She also would like to thank Kai-Min Chung and Hoeteck Wee for helpful technical discussions during an early stage of the project. This work is in part supported by a Packard Fellowship, a DARPA SIEVE grant under a sub-contract from SRI, and NSF grants under award numbers 2001026 and 1601879.

References

[ABCP16] Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval. Better security for functional encryption for inner product evaluations. Cryptology ePrint Archive, Report 2016/011, 2016.

[Abe99] Masayuki Abe. Mix-networks on permutation networks. In *ASIACRYPT*, 1999.

[ABG19] Michel Abdalla, Fabrice Benhamouda, and Romain Gay. From single-input to multi-client inner-product functional encryption. In *Asiacrypt*, 2019.

[ABM+20] Michel Abdalla, Florian Bourse, Hugo Marival, David Pointcheval, Azam Soleimanian, and Hendrik Waldner. Multi-client inner-product functional encryption in the random-oracle model. Cryptology ePrint Archive, Report 2020/788, 2020. https://eprint.iacr.org/2020/788.

[ACF+18] Michel Abdalla, Dario Catalano, Dario Fiore, Romain Gay, and Bogdan Ursu. Multi-input functional encryption for inner products: Function-hiding realizations and constructions without pairings. In *CRYPTO*, 2018.

[Adi08] Ben Adida. Helios: Web-based open-audit voting. In *Proceedings of the 17th Conference on Security Symposium*, 2008.

[AGRW17] Michel Abdalla, Romain Gay, Mariana Raykova, and Hoeteck Wee. Multi-input inner-product functional encryption from pairings. In *Advances in Cryptology – EUROCRYPT 2017*, pages 601–626, 2017.

[AGT20] Shweta Agrawal, Rishab Goyal, and Junichi Tomida. Multi-party functional encryption. Cryptology ePrint Archive, Report 2020/1266, 2020. https://eprint.iacr.org/2020/1266.

[AGW20] Michel Abdalla, Junqing Gong, and Hoeteck Wee. Functional encryption for attribute-weighted sums from k-lin. In *CRYPTO*, volume 12170, pages 685–716, 2020.

[AKTZ17] Nikolaos Alexopoulos, Aggelos Kiayias, Riivo Talviste, and Thomas Zacharias. Mcmix: Anonymous messaging via secure multiparty computation. In *Usenix Security*, 2017.

[ALS16] Shweta Agrawal, Benoît Libert, and Damien Stehlé. Fully secure functional encryption for inner products, from standard assumptions. In *CRYPTO*, volume 9816, pages 333–362, 2016.

[APY20] Ittai Abraham, Benny Pinkas, and Avishay Yanai. Blinder: Mpc based scalable and robust anonymous committed broadcast. In *ACM CCS*, 2020.

[AW07] Ben Adida and Douglas Wikström. How to shuffle in public. In *TCC*, 2007.

[BBGN19a] Borja Balle, James Bell, Adrià Gascón, and Kobbi Nissim. Differentially private summation with multi-message shuffling. *CoRR*, abs/1906.09116, 2019.

[BBGN19b] Borja Balle, James Bell, Adrià Gascón, and Kobbi Nissim. The privacy blanket of the shuffle model. In *CRYPTO*, 2019.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *Advances in Cryptology – CRYPTO 2004*, pages 41–55, 2004.

[BCG+14] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *IEEE S & P*, 2014.

[BCKL08] Mira Belenkiy, Melissa Chase, Markulf Kohlweiss, and Anna Lysyanskaya. P-signatures and noninteractive anonymous credentials. In *TCC*, 2008.

[BDGM19] Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Leveraging linear decryption: Rate-1 fully-homomorphic encryption and time-lock puzzles. In *TCC*, 2019.

[BDGM20] Zvika Brakerski, Nico Dottling, Sanjam Garg, and Giulio Malavolta. Factoring and pairings are not necessary for io: Circular-secure lwe suffices. Cryptology ePrint Archive, Report 2020/1024, 2020.

[BEM+17] Andrea Bittau, Úlfar Erlingsson, Petros Maniatis, Ilya Mironov, Ananth Raghunathan, David Lie, Mitch Rudominer, Ushasree Kode, Julien Tinnes, and Bernhard Seefeld. Prochlo: Strong privacy for analytics in the crowd. In *SOSP*, 2017.

[BG12] Stephanie Bayer and Jens Groth. Efficient zero-knowledge argument for correctness of a shuffle. In *Eurocrypt*, volume 7237, pages 263–280, 2012.

[BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In *CRYPTO*, 2001.

[BIK+17] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191. ACM, 2017.

[BJK15] Allison Bishop, Abhishek Jain, and Lucas Kowalczyk. Function-hiding inner product encryption. In *Asiacrypt*, 2015.

[BKS18] Zvika Brakerski, Ilan Komargodski, and Gil Segev. Multi-input functional encryption in the private-key setting: Stronger security from weaker assumptions. *J. Cryptology*, 31(2):434–520, 2018.

[BL13] Foteini Baldimtsi and Anna Lysyanskaya. Anonymous credentials light. In *ACM CCS*, 2013.

[BNO08] Amos Beimel, Kobbi Nissim, and Eran Omri. Distributed private data analysis: Simultaneously solving how and what. In *CRYPTO*, page 451–468, 2008.

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

[BW07] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In *Theoretical Cryptography Conference (TCC)*, 2007.

[CBM15] Henry Corrigan-Gibbs, Dan Boneh, and David Mazières. Riposte: An anonymous messaging system handling millions of users. In *S & P*, 2015.

[CFN90] D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In *CRYPTO*, 1990.

[CGF10] Henry Corrigan-Gibbs and Bryan Ford. Dissent: Accountable anonymous group messaging. In *CCS*, page 340–350, 2010.

[Cha81] David L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90, February 1981.

[Cha82] David Chaum. Blind signatures for untraceable payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *CRYPTO*, 1982.

[Cha88] David L. Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1(1):65–75, March 1988.

[CL05] Jan Camenisch and Anna Lysyanskaya. A formal treatment of onion routing. In *CRYPTO*, 2005.

[CLTV15] Ran Canetti, Huijia Lin, Stefano Tessaro, and Vinod Vaikuntanathan. Obfuscation of probabilistic circuits and applications. In *Theory of Cryptography*, pages 468–497, 2015.

[CSG+18] Jeremy Chotard, Edouard Dufour Sans, Romain Gay, Duong Hieu Phan, and David Pointcheval. Decentralized multi-client functional encryption for inner product. In *Asiacrypt*, 2018.

[CSS12] T-H. Hubert Chan, Elaine Shi, and Dawn Song. Optimal lower bound for differentially private multi-party aggregation. In *European Symposium on Algorithms (ESA)*, 2012.

[CSU+19] Albert Cheu, Adam Smith, Jonathan Ullman, David Zeber, and Maxim Zhilyaev. Distributed differential privacy via shuffling. In *Eurocrypt*, 2019.

[DD08] George Danezis and Claudia Diaz. A survey of anonymous communication channels. Technical Report MSR-TR-2008-35, Microsoft Research, 2008.

[DMNS06] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, 2006.

[DMS04] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *USENIX Security Symposium*, 2004.

[DS18] Jean Paul Degabriele and Martijn Stam. Untagging tor: A formal treatment of onion encryption. In *EUROCRYPT*, 2018.

[EFM+19] Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Abhradeep Thakurta. Amplification by shuffling: From local to central differential privacy via anonymity. In *SODA*, 2019.

[EHK+13] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Rafols, and Jorge Luis Villar. An algebraic framework for diffie-hellman assumptions. In *CRYPTO*, 2013.

[EPK14] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *CCS*, 2014.

[EY09] Matthew Edman and Bülent Yener. On anonymity in an electronic society: A survey of anonymous communication systems. *ACM Comput. Surv.*, 42(1), December 2009.

[GGG+14] Shafi Goldwasser, S. Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. Multi-input functional encryption. In *Eurocrypt*, volume 8441, pages 578–602, 2014.

[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 *IEEE Symposium on Foundations of Computer Science (FOCS)*, 2013.

[GH19] Craig Gentry and Shai Halevi. Compressible FHE with applications to PIR. In *Theory of Cryptography (TCC)*, volume 11892 of *Lecture Notes in Computer Science*, pages 438–464, 2019.

[GIKM00] Yael Gertner, Yuval Ishai, Eyal Kushilevitz, and Tal Malkin. Protecting data privacy in private information retrieval schemes. *J. Comput. Syst. Sci.*, 60(3), 2000.

[GP20] Romain Gay and Rafael Pass. Indistinguishability obfuscation from circular security. Cryptology ePrint Archive, Report 2020/1010, 2020.

[GPV19] Badih Ghazi, Rasmus Pagh, and Ameya Velingker. Scalable and differentially private distributed aggregation in the shuffled model. *CoRR*, abs/1906.08320, 2019.

[GRS99] David Goldschlag, Michael Reed, and Paul Syverson. Onion routing for anonymous and private internet connections. *Communications of the ACM*, 42:39–41, 1999.

[GVW15] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate encryption for circuits from LWE. In *CRYPTO*, 2015.

[HAB+17] Ethan Heilman, Leen Alshenibr, Foteini Baldimtsi, Alessandra Scafuro, and Sharon Goldberg. Tumblebit: An untrusted bitcoin-compatible anonymous payment hub. In *NDSS*, 2017.

[HMPS14] Susan Hohenberger, Steven A. Myers, Rafael Pass, and Abhi Shelat. ANONIZE: A large-scale anonymous survey system. In *IEEE S & P*, 2014.

[IKOS06] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography from anonymity. In *FOCS*, 2006.

[JLS20] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. Cryptology ePrint Archive, Report 2020/1003, 2020.

[Lin17] Huijia Lin. Indistinguishability obfuscation from sxdh on 5-linear maps and locality-5 prgs. In *CRYPTO*, 2017.

[LT19] Benoit Libert and Radu Titiu. Multi-client functional encryption for linear functions in the standard model from lwe. 11923:520–551, 2019.

[LV16] Huijia Lin and Vinod Vaikuntanathan. Indistinguishability obfuscation from DDH-like assumptions on constant-degree graded encodings. In *FOCS*, pages 11–20, 2016.

[OS97] Rafail Ostrovsky and Victor Shoup. Private information storage (extended abstract). In *STOC*, pages 294–303, 1997.

[SBC+07] Elaine Shi, John Bethencourt, T-H. Hubert Chan, Dawn Song, and Adrian Perrig. Multi-dimensional range query over encrypted data. In *IEEE Symposium on Security and Privacy*, pages 350–364, 2007.

[SBS02] Rob Sherwood, Bobby Bhattacharjee, and Aravind Srinivasan. P5: A protocol for scalable anonymous communication. In *IEEE S & P*, 2002.

[SCR+11] Elaine Shi, T-H. Hubert Chan, Eleanor Rieffel, Richard Chow, and Dawn Song. Privacy-preserving aggregation of time-series data. In *Network and Distributed System Security Symposium (NDSS)*, 2011.

[SCRS17] Elaine Shi, T.-H. Hubert Chan, Eleanor Rieffel, and Dawn Song. Distributed private data analysis: Lower bounds and practical constructions. *ACM Trans. Algorithms*, 13(4), December 2017.

[SK95] Kazue Sako and Joe Kilian. Receipt-free mix-type voting scheme: A practical solution to the implementation of a voting booth. In *EUROCRYPT*, 1995.

[SSA+18] Fatemeh Shirazi, Milivoj Simeonovski, Muhammad Rizwan Asghar, Michael Backes, and Claudia Diaz. A survey on routing in anonymous communication protocols. 2018.

[SSW09] Emily Shen, Elaine Shi, and Brent Waters. Predicate privacy in encryption systems. In *Theory of Cryptography Conference (TCC)*, pages 457–473, Berlin, Heidelberg, 2009. Springer-Verlag.

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

[sys] Computer science research and practice on slack.

[TGL+17] Nirvan Tyagi, Yossi Gilad, Derek Leung, Matei Zaharia, and Nickolai Zeldovich. Stadium: A distributed metadata-private messaging system. In *SOSP*, 2017.

[vdHLZZ15] Jelle van den Hooff, David Lazar, Matei Zaharia, and Nickolai Zeldovich. Vuvuzela: Scalable private messaging resistant to traffic analysis. In *SOSP*, 2015.

[Wee16] Hoeteck Wee. New techniques for attribute-hiding in prime-order bilinear groups. Manuscript, 2016.

[WW20] Hoeteck Wee and Daniel Wichs. Candidate obfuscation via oblivious LWE sampling. Cryptology ePrint Archive, Report 2020/1042, 2020.

[ZZZR05] Li Zhuang, Feng Zhou, Ben Y. Zhao, and Antony Rowstron. Cashmere: Resilient anonymous routing. In *NSDI*, 2005.

A Equivalence between Simulation- and Indistinguishability-Based Notions

We now prove Lemma 2.1. We restate the lemma for the reader's convenience.

Lemma A.1 (Restatement of Lemma 2.1)**.** *A NIAR scheme is SIM-secure iff it is IND-secure. Similarly, a NIAR scheme is SIM-secure with receiver-insider protection (or full insider protection, respectively) iff it is IND-secure with receiver-insider protection (of full insider protection, respectively).*

Proof. Let SIM^a-secure, for $a \in \{\text{SR}, \text{S}, \text{min}\}$, denote SIM-security with $\mathsf{Leak} = \mathsf{Leak}^a$. Similarly let IND^a-secure denote IND-security with $\mathsf{Leak} = \mathsf{Leak}^a$. The corresponding experiment $\mathsf{NIAR\text{-}Expt}^{b,\mathcal{A}}(1^\kappa)$ with $\mathsf{Leak} = \mathsf{Leak}^a$ in the admissibility rule is denoted as $\mathsf{NIAR\text{-}Expt}_a^{b,\mathcal{A}}(1^\kappa)$. Now we show that for any $a \in \{\text{SR}, \text{S}, \text{min}\}$, a NIAR scheme is SIM^a-secure iff IND^a-secure.

SIM^a-secure $\Rightarrow$ IND^a-secure If a NIAR scheme is SIM^a-secure, then according to Definition 1, there exists a simulator Sim^a such that for any non-uniform p.p.t. adversary $\mathcal{A}$, $\mathcal{A}$'s views in $\mathsf{Real}^{\mathcal{A}}(1^\kappa)$ and $\mathsf{Ideal}^{\mathcal{A},\mathsf{Sim}^a}(1^\kappa)$ are computationally indistinguishable. Now consider the following experiment $\mathsf{Hyb}_a^b(1^\kappa)$ for $b \in \{0,1\}$:

$n, \mathcal{K}_S, \mathcal{K}_R, \pi^{(0)}, \pi^{(1)} \leftarrow \mathcal{A}(1^\kappa)$

$(\{\mathsf{ek}_i\}_{i\in[n]}, \{\mathsf{rk}_i\}_{i\in[n]}, \mathsf{tk}) \leftarrow \mathsf{Sim}^a(1^\kappa, n, \mathcal{K}_S, \mathcal{K}_R, \mathsf{Leak}^a(\pi^{(b)}, \mathcal{K}_S, \mathcal{K}_R))$

For $t = 1, 2, \ldots$:

- If $t = 1$, then $\{x_{i,t}^{(0)}\}_{i\in\mathcal{H}_S}, \{x_{i,t}^{(1)}\}_{i\in\mathcal{H}_S} \leftarrow \mathcal{A}(\mathsf{tk}, \{\mathsf{ek}_i\}_{i\in\mathcal{K}_S}, \{\mathsf{rk}_i\}_{i\in\mathcal{K}_R})$;
- else $\{x_{i,t}^{(0)}\}_{i\in\mathcal{H}_S}, \{x_{i,t}^{(1)}\}_{i\in\mathcal{H}_S} \leftarrow \mathcal{A}(\{\mathsf{ct}_{i,t-1}\}_{i\in\mathcal{H}_S})$;
- $\{\mathsf{ct}_{i,t}\}_{i\in\mathcal{H}_S} \leftarrow \mathsf{Sim}^a\left(\left\{\forall i \in \mathcal{K}_R \cap \pi^{(b)}(\mathcal{H}_S) : (i, x_{j,t}) \text{ for } j = (\pi^{(b)})^{-1}(i)\right\}\right)$

According to the definition of SIM^a-secure, for $b \in \{0,1\}$, any non-uniform p.p.t. adversary's views in $\mathsf{NIAR\text{-}Expt}_a^{b,\mathcal{A}}(1^\kappa)$ and $\mathsf{Hyb}_a^b(1^\kappa)$ are computationally indistinguishable. Moreover, any admissible $\mathcal{A}$'s views in $\mathsf{Hyb}_a^0(1^\kappa)$ and $\mathsf{Hyb}_a^1(1^\kappa)$ are identical due to the admissibility conditions: 1) $\mathsf{Leak}^a(\pi^{(0)}, \mathcal{K}_S, \mathcal{K}_R) = \mathsf{Leak}^a(\pi^{(1)}, \mathcal{K}_S, \mathcal{K}_R)$; 2) for any $i \in \mathcal{K}_R \cap \pi^{(b)}(\mathcal{H}_S)$, $x_{j_0,t}^{(0)} = x_{j_1,t}^{(1)}$ where

for $b \in \{0,1\}$, $j_b := (\pi^{(b)})^{-1}(i)$. Further, $\mathsf{NIAR\text{-}Expt}_a^{1,\mathcal{A}}(1^\kappa)$ and $\mathsf{Hyb}_a^1(1^\kappa)$ are also computationally indistinguishable due to the SIM^a-security.

To summarize, any admissible, non-uniform p.p.t. adversary $\mathcal{A}$'s views in $\mathsf{NIAR\text{-}Expt}_a^{0,\mathcal{A}}(1^\kappa)$ and $\mathsf{NIAR\text{-}Expt}_a^{1,\mathcal{A}}(1^\kappa)$ are computationally indistinguishable, i.e., the NIAR scheme is IND^a-secure.

IND^a-secure $\Rightarrow$ SIM^a-secure A NIAR scheme is IND^a-secure iff any admissible, non-uniform p.p.t. adversary $\mathcal{A}$'s views in $\mathsf{NIAR\text{-}Expt}_a^{0,\mathcal{A}}(1^\kappa)$ and $\mathsf{NIAR\text{-}Expt}_a^{1,\mathcal{A}}(1^\kappa)$ are computationally indistinguishable. Construct a p.p.t. simulator Sim^a as follows:

Description of Sim^a

- Upon receiving $1^\kappa, n, \mathcal{K}_S, \mathcal{K}_R, \mathsf{Leak}^*$, Choose the lexicographically smallest $\widetilde{\pi}$ such that $\mathsf{Leak}^a(\widetilde{\pi}, \mathcal{K}_S, \mathcal{K}_R) = \mathsf{Leak}^*$. Note that for any $a \in \{\mathrm{S}, \mathrm{SR}, \min\}$, all π's satisfying $\mathsf{Leak}^a(\widetilde{\pi}, \mathcal{K}_S, \mathcal{K}_R) = \mathsf{Leak}^*$ result in the same $\mathcal{K}_R \cap \pi(\mathcal{H}_S)$ set — henceforth let Γ be this set. Output $(\{\mathsf{ek}_i\}_{i\in[n]}, \{\mathsf{rk}_i\}_{i\in[n]}, \mathsf{tk}) \leftarrow \mathbf{Setup}(1^\kappa, n, \widetilde{\pi})$.
- For $t = 1, 2, \ldots$: receive $\{(i, \widetilde{x}_{i,t})\}_{i\in\Gamma}$, and let

$$x_{j,t} := \begin{cases} \widetilde{x}_{i,t} & \text{if } j = \widetilde{\pi}^{-1}(i) \text{ for some } i \in \Gamma \\ 0 & \text{otherwise} \end{cases}$$

  For $i \in \mathcal{H}_S$, let $\mathsf{ct}_{i,t} \leftarrow \mathbf{Enc}(\mathsf{ek}_i, \{x_{i,t}\}_{i\in\mathcal{H}_S}, t)$, and output $\{\mathsf{ct}_{i,t}\}_{i\in\mathcal{H}_S}$.

If a non-uniform p.p.t. adversary $\mathcal{A}$ can distinguish $\mathsf{Real}^{\mathcal{A}}(1^\kappa)$ from $\mathsf{Ideal}^{\mathcal{A},\mathsf{Sim}^a}(1^\kappa)$, then we can construct a non-uniform p.p.t. admissible adversary $\mathcal{B}^a$ that can distinguish $\mathsf{NIAR\text{-}Expt}_a^{0,\mathcal{A}}(1^\kappa)$ and $\mathsf{NIAR\text{-}Expt}_a^{1,\mathcal{A}}(1^\kappa)$. $\mathcal{B}$ is interacting with a challenger $\mathcal{C}$ who is either running $\mathsf{NIAR\text{-}Expt}_a^{0,\mathcal{A}}(1^\kappa)$ or $\mathsf{NIAR\text{-}Expt}_a^{1,\mathcal{A}}(1^\kappa)$.

Description of $\mathcal{B}^a(1^\kappa)$

- $(n, \pi, \mathcal{K}_S, \mathcal{K}_R) \leftarrow \mathcal{A}(1^\kappa)$;
- Choose the smallest $\widetilde{\pi}$ in lexicographic order such that $\mathsf{Leak}^a(\widetilde{\pi}, \mathcal{K}_S, \mathcal{K}_R) = \mathsf{Leak}^a(\pi, \mathcal{K}_S, \mathcal{K}_R)$; and send $(n, \mathcal{K}_S, \mathcal{K}_R, \pi, \widetilde{\pi})$ to $\mathcal{C}$;
- For $t = 1, 2, \cdots$:
  - If $t = 1$, pass $(\mathsf{tk}, \{\mathsf{ek}_i\}_{i\in\mathcal{K}_S}, \{\mathsf{rk}_i\}_{i\in\mathcal{K}_R})$ received from $\mathcal{C}$ to $\mathcal{A}$; else, pass $\{\mathsf{ct}_{i,t-1}\}_{i\in\mathcal{H}_S}$ received from $\mathcal{C}$ to $\mathcal{A}$;
  - receive $\{x_{i,t}^{(0)}\}_{i\in\mathcal{H}_S}$ from $\mathcal{A}$;
  - $\forall j \in \mathcal{H}_S$, let $x_{j,t}^{(1)} = \begin{cases} x_{j',t}^{(0)} \text{ where } j' = \pi^{-1}(\widetilde{\pi}(j)) & \text{if } \widetilde{\pi}(j) \in \mathcal{K}_R \\ 0 & \text{otherwise} \end{cases}$ $\quad (*)$
  - send $\{x_{i,t}^{(0)}\}_{i\in\mathcal{H}_S}, \{x_{i,t}^{(1)}\}_{i\in\mathcal{H}_S}$ to $\mathcal{C}$;
- $b' \leftarrow \mathcal{A}$, output b'.

Note that for any $a \in \{\mathrm{S}, \mathrm{SR}, \min\}$, $\widetilde{\pi}(\mathcal{H}_S) \cap \mathcal{K}_R = \pi(\mathcal{H}_S) \cap \mathcal{K}_R$. Therefore, $\mathcal{B}^a$ is able to compute $\{x_{i,t}^{(1)}\}_{i\in\mathcal{H}_S}$ in the expression $(*)$. Further, $\mathcal{B}^a$ is an admissible adversary since one can check that

the following conditions hold by construction: 1) $\mathsf{Leak}^a(\widetilde{\pi}, \mathcal{K}_S, \mathcal{K}_R) = \mathsf{Leak}^a(\pi, \mathcal{K}_S, \mathcal{K}_R)$; 2) for any $i \in \mathcal{K}_R \cap \pi(\mathcal{H}_S) = \mathcal{K}_R \cap \widetilde{\pi}(\mathcal{H}_S)$, $x^{(0)}_{j_0,t} = x^{(1)}_{j_1,t}$ where $j_0 = \pi^{-1}(i)$ and $j_1 = \widetilde{\pi}^{-1}(i)$.

If $\mathcal{C}$ runs $\mathsf{NIAR\text{-}Expt}^{0,\mathcal{A}}_a(1^\kappa)$, then $\mathcal{A}$'s view is identically distributed as $\mathsf{Real}^{\mathcal{A}}(1^\kappa)$; otherwise, $\mathcal{A}$'s view is identically distributed as $\mathsf{Ideal}^{\mathcal{A},\mathsf{Sim}^a}(1^\kappa)$ according to the construction. Hence, $\mathcal{B}^a$ is an admissible adversary that can distinguish $\mathsf{NIAR\text{-}Expt}^{0,\mathcal{A}}_a(1^\kappa)$ and $\mathsf{NIAR\text{-}Expt}^{1,\mathcal{A}}_a(1^\kappa)$ with same advantage as $\mathcal{A}$ can distinguish $\mathsf{Real}^{\mathcal{A}}(1^\kappa)$ from $\mathsf{Ideal}^{\mathcal{A},\mathsf{Sim}^a}(1^\kappa)$.

Summarizing, if a NIAR scheme is IND^a-secure, then it is SIM^a-secure. □

B Details of Our Fault-Tolerant Scheme

Earlier in Section 8, we sketched our fault-tolerant NIAR constructions. In this section, we shall provide the full details of the constructions and prove them secure. Specifically, we will first present the details our fault-tolerant NIAR scheme with receiver-insider protection. Then, in Section B.3, we shall present the proofs for our construction with full insider protection.

B.1 Fault-Tolerant, Multi-Client Functional Encryption for Selection

First, we modify the underlying MCFE-for-selection scheme to be fault-tolerant. In terms of abstraction, the **Gen**, **Setup**, and **KGen** algorithms have the same syntax as before; but the $\mathbf{Enc}(\mathsf{mpk}, \mathsf{ek}_i, \mathbf{x}_i^{(b)}, t, \mathcal{O})$ and $\mathbf{Dec}(\mathsf{mpk}, \mathsf{sk}_{\mathbf{y}}, \mathcal{O}, \{\mathsf{ct}_i\}_{i\in\mathcal{O}})$ algorithms now take an additional parameter $\mathcal{O} \subseteq [n]$ as input where $\mathcal{O}$ denotes the currently online set of clients. For correctness, given a correctly constructed functional secret key $\mathsf{sk}_{\mathbf{y}}$ for the vector $\mathbf{y} := (\mathbf{y}_1, \ldots, \mathbf{y}_n)$ and a collection of correctly computed ciphertexts $\{\mathsf{ct}_i\}_{i\in\mathcal{O}}$ for the plaintext messages $\mathbf{x}_\mathcal{O} := \{\mathbf{x}_i\}_{i\in\mathcal{O}}$ respectively, we require that $\mathbf{Dec}(\mathsf{mpk}, \mathsf{sk}_{\mathbf{y}}, \mathcal{O}, \{\mathsf{ct}_i\}_{i\in\mathcal{O}})$ results in $\langle \mathbf{x}_\mathcal{O}, \mathbf{y}_\mathcal{O}\rangle$ where $\mathbf{y}_\mathcal{O} = \{\mathbf{y}_i\}_{i\in\mathcal{O}}$. In other words, if $\mathbf{y}$ is selecting a coordinate for a client not in $\mathcal{O}$, then the **Dec** should output 0.

B.1.1 Function-Revealing MCFE for Selection

Security definition. We modify the previous IND-secure notion (Definition 6) slightly to account for the online set $\mathcal{O}$.

Experiment $\mathsf{Expt}^b(1^\kappa)$:

- **Setup.** Same as before.
- **Query.** The adversary can make the following types of queries; moreover, all **KGen** queries must be made *before* any **Enc** query:
  - **KGen queries.** Same as before.
  - **Enc queries.** Whenever $\mathcal{A}$ makes an **Enc** query by specifying $(\mathcal{O}, \{\mathbf{x}^{(0)}_{i,t}, \mathbf{x}^{(1)}_{i,t}\}_{i\in\mathcal{H}\cap\mathcal{O}})$ where $\mathcal{H} := [n]\backslash\mathcal{K}$, the challenger $\mathcal{C}$ calls $\mathsf{ct}_i := \mathbf{Enc}(\mathsf{mpk}, \mathsf{ek}_i, \mathbf{x}_i^{(b)}, t, \mathcal{O})$ for $i \in \mathcal{H}\cap\mathcal{O}$ and returns $\{\mathsf{ct}_i\}_{i\in\mathcal{H}\cap\mathcal{O}}$ to $\mathcal{A}$; moreover it sets $t := t + 1$.

An adversary $\mathcal{A}$ is said to be *admissible* iff the following holds with probability 1: for any $\mathbf{y} := (\mathbf{y}_1, \ldots, \mathbf{y}_n)$ submitted in a **KGen** query where each $\mathbf{y}_i \in \{0,1\}^m$, for any tuple $(\mathcal{O}, \{\mathbf{x}^{(0)}_{i,t}, \mathbf{x}^{(1)}_{i,t}\}_{i\in\mathcal{H}})$ submitted in an **Enc** query,

$$\left\langle (\mathbf{x}^{(0)}_{i,t})_{i\in\mathcal{H}\cap\mathcal{O}}, (\mathbf{y}_i)_{i\in\mathcal{H}\cap\mathcal{O}} \right\rangle = \left\langle (\mathbf{x}^{(1)}_{i,t})_{i\in\mathcal{H}\cap\mathcal{O}}, (\mathbf{y}_i)_{i\in\mathcal{H}\cap\mathcal{O}} \right\rangle$$

We say that a fault-tolerant MCFE-for-selection scheme is IND-secure iff for any non-uniform p.p.t. admissible adversary $\mathcal{A}$, its views in the above $\mathsf{Expt}^0(1^\kappa)$ and $\mathsf{Expt}^1(1^\kappa)$ are computationally indistinguishable.

Construction. We now modify our previous MCFE construction in Section 5.2 as below.

- **Setup** and **KGen**: same as in Section 5.2.
- **Enc**$(\mathsf{mpk}, \mathsf{ek}_i, \mathbf{x}_i, t, \mathcal{O})$: same as Section 5.2 except that $\mathsf{CPRF.Eval}(K_i, t)$ is replaced with the following which takes into account the online clients $\mathcal{O}$:

$$\mathsf{CPRF.Eval}(K_i, t, \mathcal{O}) := \sum_{j \neq i, j \in \mathcal{O}} (-1)^{j<i} \cdot \mathsf{PRF}_{k_{ij}}(t)$$

- **Dec**$(\mathsf{mpk}, \mathsf{sk}_\mathbf{y}, \mathcal{O}, \{\mathsf{ct}_i\}_{i \in \mathcal{O}})$: same as Section 5.2 except that only the clients in $\mathcal{O}$ are involved in the decryption, i.e., the equation $[\![v]\!] := \prod_{i \in [n]} [\![\langle \mathbf{c}_{i,1}, \mathbf{k}_{i,1} \rangle + \langle \mathbf{c}_{i,2}, \mathbf{k}_{i,2} \rangle + \langle \widetilde{\mathbf{c}}_i, \widetilde{\mathbf{k}}_i \rangle]\!]$ is replaced with $[\![v]\!] := \prod_{i \in \mathcal{O}} [\![\langle \mathbf{c}_{i,1}, \mathbf{k}_{i,1} \rangle + \langle \mathbf{c}_{i,2}, \mathbf{k}_{i,2} \rangle + \langle \widetilde{\mathbf{c}}_i, \widetilde{\mathbf{k}}_i \rangle]\!]$.

Theorem B.1 (Fault-tolerant, function-revealing MCFE for selection)**.** *Assume that the Decisional Linear assumption holds in the group $\mathbb{G}$. The above fault-tolerant* MCFE *scheme is IND-secure.*

Proof. The proof can be carried out in almost an identical manner as that of Theorem 5.2. Particularly, note that the term $[\![\widetilde{\mathbf{c}}]\!]$ in the ciphertext does not affect the security of the scheme at this point. □

B.1.2 Upgrade for Function Privacy

Security definitions. The new function-hiding security experiment $\mathsf{FH\text{-}Expt}^b(1^\kappa)$ is defined just like in Section 5.1, except that **Enc** queries are defined like in Section B.1.1, where $\mathcal{A}$ additionally specifies an online set $\mathcal{O}$, and the challenger encrypts $\{\mathbf{x}_i^{(b)}\}_{i \in \mathcal{H} \cap \mathcal{O}}$.

An adversary $\mathcal{A}$ is said to be *admissible* iff the following hold with probability 1: for any pair $(\mathbf{y}^{(0)}, \mathbf{y}^{(1)})$ submitted in a **KGen** query where for $b \in \{0,1\}$, $\mathbf{y}^{(b)} := (\mathbf{y}_1^{(b)}, \ldots, \mathbf{y}_n^{(b)}) \in \{0,1\}^{mn}$, it must be that

1. for $i \in \mathcal{K}$, $\mathbf{y}_i^{(0)} = \mathbf{y}_i^{(1)}$.
2. for any $(\mathcal{O}, \{\mathbf{x}_{i,t}^{(0)}, \mathbf{x}_{i,t}^{(1)}\}_{i \in \mathcal{H} \cap \mathcal{O}})$ submitted in an **Enc** query,

$$\left\langle (\mathbf{x}_{i,t}^{(0)})_{i \in \mathcal{H} \cap \mathcal{O}}, (\mathbf{y}_i^{(0)})_{i \in \mathcal{H} \cap \mathcal{O}} \right\rangle = \left\langle (\mathbf{x}_{i,t}^{(1)})_{i \in \mathcal{H} \cap \mathcal{O}}, (\mathbf{y}_i^{(1)})_{i \in \mathcal{H} \cap \mathcal{O}} \right\rangle \tag{5}$$

We say that a fault-tolerant MCFE scheme is function-hiding IND-secure iff for any non-uniform p.p.t. admissible adversary $\mathcal{A}$, its views in the above $\mathsf{FH\text{-}Expt}^0(1^\kappa)$ and $\mathsf{FH\text{-}Expt}^1(1^\kappa)$ are computationally indistinguishable.

We can also define a relaxed notion called weakly-function-hiding IND-security. The only modification is in the admissibility rule: an adversary $\mathcal{A}$ is now said to be *admissible* iff the following holds with probability 1: for any pair $(\mathbf{y}^{(0)}, \mathbf{y}^{(1)})$ submitted in a **KGen** query where for $b \in \{0,1\}$, $\mathbf{y}^{(b)} := (\mathbf{y}_1^{(b)}, \ldots, \mathbf{y}_n^{(b)}) \in \{0,1\}^{mn}$, it must be that

1. for $i \in \mathcal{K}$, $\mathbf{y}_i^{(0)} = \mathbf{y}_i^{(1)}$.
2. for any $(\mathcal{O}, \{\mathbf{x}_{i,t}^{(0)}, \mathbf{x}_{i,t}^{(1)}\}_{i \in \mathcal{H} \cap \mathcal{O}})$ submitted in an **Enc** query,

$$\left\langle (\mathbf{x}_{i,t}^{(0)})_{i \in \mathcal{H} \cap \mathcal{O}}, (\mathbf{y}_i^{(0)})_{i \in \mathcal{H} \cap \mathcal{O}} \right\rangle = \left\langle (\mathbf{x}_{i,t}^{(1)})_{i \in \mathcal{H} \cap \mathcal{O}}, (\mathbf{y}_i^{(0)})_{i \in \mathcal{H} \cap \mathcal{O}} \right\rangle = \left\langle (\mathbf{x}_{i,t}^{(1)})_{i \in \mathcal{H} \cap \mathcal{O}}, (\mathbf{y}_i^{(1)})_{i \in \mathcal{H} \cap \mathcal{O}} \right\rangle \tag{6}$$

Weakly function-hiding construction $\mathsf{MCFE}^{\mathrm{wfh}}$. We can upgrade our MCFE scheme earlier in Section B.1.1 to have weak function privacy like below. The idea is the same as in Section 5.3, and the only change is that the new **Enc** and **Dec** are aware of the online set $\mathcal{O}$.

$\mathsf{MCFE}^{\mathrm{wfh}}$: **fault-tolerant, weakly function-hiding MCFE for selection**

- **Gen**(1^κ), **Setup**(pp, m, n), and **KGen**$(\mathsf{mpk}, \mathsf{msk}, \mathbf{y})$: Same as in Section 5.3.
- **Enc**$(\mathsf{mpk}, \mathsf{ek}_i, \mathbf{x}, t, \mathcal{O})$: Let $\mathsf{ct} := \mathsf{MCFE}.\mathbf{Enc}(\mathsf{mpk}', \mathsf{ek}'_i, \mathbf{x}, t, \mathcal{O}) \in \mathbb{G}_1^{m+4}$, and $\overline{\mathsf{ct}} := \mathsf{FE}.\mathbf{KGen}(\mathsf{msk}_i, \mathsf{ct})$. Output $\mathsf{CT} := (\mathsf{ct}, \overline{\mathsf{ct}})$.
- **Dec**$(\mathsf{mpk}, \mathsf{sk}_{\mathbf{y}}, \mathcal{O}, \{\mathsf{CT}_i\}_{i\in\mathcal{O}})$: Parse each $\mathsf{CT}_i := (\mathsf{ct}_i, \overline{\mathsf{ct}}_i)$ for $i \in \mathcal{O}$. Parse $\mathsf{sk}_{\mathbf{y}} := (\overline{\mathbf{k}}_1, \ldots, \overline{\mathbf{k}}_n)$. For $i \in \mathcal{O}$, call $v_i := \mathsf{FE}.\mathbf{Dec}(\overline{\mathsf{ct}}_i, \mathsf{ct}_i, \overline{\mathbf{k}}_i)$. Output $\prod_{i\in\mathcal{O}} v_i$.

Theorem B.2 (Fault-tolerant, weakly function-hiding MCFE for selection)**.** *Assume that the Decisional assumption holds in $\mathbb{G}$, and suppose that in the $\mathsf{MCFE}^{\mathrm{wfh}}$ construction above, we employ the FE scheme in Section 4.2 and the fault-tolerant MCFE scheme in Section B.1.1. Also, suppose that the PRF used in the construction of CPRF satisfies pseudorandomness. Then, the fault-tolerant $\mathsf{MCFE}^{\mathrm{wfh}}$ construction above is weakly-function-hiding IND-secure.*

Proof. The proof is very similar to that of Theorem 5.3. Nonetheless, we present a modified proof below for completeness.

Experiment $\mathsf{FH\text{-}Expt}^0(1^\kappa)$. We start with the $\mathsf{FH\text{-}Expt}^0(1^\kappa)$ experiment. In this experiment, whenever answering either **Enc** queries or **KGen** queries from $\mathcal{A}$, the vectors $\{\mathbf{x}^{(0)}_{i,t}\}_{i\in\mathcal{H}\cap\mathcal{O}}$ and $\mathbf{y}^{(0)}$ are used. Recall also that $\mathcal{A}$ must satisfy the weak-function-hiding admissibility rules defined in Equation (6).

Experiment Hyb_0. In Hyb_0, when answering the t-th **Enc** query, the challenger $\mathcal{C}$ uses the vector $\{\mathbf{x}^{(1)}_{i,t}\}_{i\in\mathcal{H}\cap\mathcal{O}}$, i.e., it calls $\mathsf{CT}_i := \mathsf{MCFE}^{\mathrm{wfh}}.\mathbf{Enc}(\mathsf{mpk}, \mathsf{ek}_i, \mathbf{x}^{(1)}_i, t, \mathcal{O})$ for each $i \in \mathcal{H}\cap\mathcal{O}$ and returns $\{\mathsf{CT}_i\}_{i\in\mathcal{H}\cap\mathcal{O}}$ to $\mathcal{A}$.

Any non-uniform p.p.t. adversary's views in $\mathsf{FH\text{-}Expt}^0(1^\kappa)$ and Hyb_0 are computationally indistinguishable as long as the faul-tolerant MCFE employed is IND-secure as defined in Section B.1.1.

Experiment Hyb_ℓ. For $\ell \in [Q_{\mathrm{KGen}}]$, Hyb_ℓ is defined as below: for the first ℓ number of **KGen** queries made by $\mathcal{A}$, generate the functional key by calling the honest $\mathsf{MCFE}^{\mathrm{wfh}}.\mathbf{KGen}$ algorithm using $\mathbf{y}^{(1)}$ as the input vector; for all other **KGen** queries, generate the functional key by calling the honest $\mathsf{MCFE}^{\mathrm{wfh}}.\mathbf{KGen}$ algorithm using $\mathbf{y}^{(0)}$ as the input vector. Otherwise, Hyb_ℓ is defined in the same way as Hyb_0.

Experiment $\widetilde{\mathsf{Hyb}}_\ell$. Let Q_{KGen} be the maximum number of **KGen** queries made by $\mathcal{A}$. For $\ell \in [Q_{\mathrm{KGen}}]$, $\widetilde{\mathsf{Hyb}}_\ell$ is defined as below:

- **Setup.** For $i \in \mathcal{H}$: instead of calling $(\mathsf{mpk}_i, \mathsf{msk}_i) \leftarrow \mathsf{FE}.\mathbf{Setup}(\mathsf{pp}, m)$, the challenger $\mathcal{C}$ calls $(\mathsf{mpk}_i, \mathsf{msk}_i) \leftarrow \mathsf{FE}.\widetilde{\mathbf{Setup}}(\mathsf{pp}, m+4)$. The challenger $\mathcal{C}$ performs the remainder of the **Setup** following the honest algorithm, and gives the resulting mpk and $\{\mathsf{ek}_i\}_{i\in\mathcal{K}}$ to $\mathcal{A}$.

- **KGen queries.** The first $(\ell-1)$ **KGen** queries will be answered with the honest **KGen** algorithm using $\mathbf{y}^{(1)}$ as the input, any **KGen** query after the first ℓ queries will be answered with the honest **KGen** algorithm using $\mathbf{y}^{(0)}$ as input. For the ℓ-th **KGen** query:
  - let $\mathbf{y}^{(0)}$ and $\mathbf{y}^{(1)}$ be the two vectors submitted by $\mathcal{A}$ during the ℓ-th **KGen** query, let $\mathbf{y}^* := \mathbf{y}^{(0)} \in \mathbb{Z}_q^n$;
  - let $(\mathbf{k}_1^*, \ldots, \mathbf{k}_n^*) := \mathsf{MCFE.KGen}(\mathsf{mpk}', \mathsf{msk}', \mathbf{y}^*)$; where each $\mathbf{k}_i^* \in \mathbb{Z}_q^{m+4}$ for $i \in [n]$.
  - for $i \in \mathcal{H}$, the challenger $\mathcal{C}$ calls $\overline{\mathbf{k}}_i^* := \widetilde{\mathsf{FE.Enc}}(\mathsf{msk}_i)$;
  - for $i \in \mathcal{K}$, the challenger $\mathcal{C}$ calls $\overline{\mathbf{k}}_i^* := \mathsf{FE.Enc}(\mathsf{mpk}_i, \mathbf{k}_i^*)$;
  - the challenger $\mathcal{C}$ returns the functional key $\mathsf{sk}_{\mathbf{y}^*} := (\overline{\mathbf{k}}_1^*, \ldots, \overline{\mathbf{k}}_n^*)$ to $\mathcal{A}$.
- **Enc queries.** For any **Enc** query with the submitted tuple $(\mathcal{O}, \{\mathbf{x}_i^{(0)}\}_{i\in\mathcal{H}\cap\mathcal{O}}, \{\mathbf{x}_i^{(1)}\}_{i\in\mathcal{H}\cap\mathcal{O}})$, do the following. For $i \in \mathcal{H} \cap \mathcal{O}$:
  - let $\mathsf{ct}_i = [\![\mathbf{c}_i]\!]_1 := \mathsf{MCFE.Enc}(\mathsf{mpk}', \mathsf{ek}_i', \mathbf{x}_i^{(1)}, t) \in \mathbb{G}_1^{m+4}$, and
  - let $\overline{\mathsf{ct}_i} = [\![\overline{\mathbf{c}}_i]\!]_1 := \widetilde{\mathsf{FE.KGen}}(\mathsf{msk}_i, \mathsf{ct}_i, [\![\langle \mathbf{c}_i, \mathbf{k}_i^* \rangle]\!]_1)$;

  Return $\{(\mathsf{ct}_i, \overline{\mathsf{ct}}_i)\}_{i\in\mathcal{H}\cap\mathcal{O}}$ to $\mathcal{A}$.

Using a proof that is essentially identical to that of Lemma 5.5, we can prove that as long as the FE scheme is 1-SEL-SIM-secure, then, any non-uniform p.p.t. $\mathcal{A}$'s views in Hyb_ℓ and $\widetilde{\mathsf{Hyb}}_{\ell+1}$ are computationally indistinguishable for $\ell \in [Q_{\mathrm{KGen}}] \cup \{0\}$.

Below we state and prove the counterpart of Lemma 5.6.

Lemma B.3. *Suppose that the Decisional Linear assumption holds in $\mathbb{G}$, and that the* PRF *employed by the* CPRF *scheme satisfies pseudorandomness. Further, suppose that we employ the fault-tolerant* MCFE *scheme described in Section B.1. Then, any non-uniform p.p.t. adversary's views in $\widetilde{\mathsf{Hyb}}_\ell$ and Hyb_ℓ are computationally indistinguishable for $\ell \in [Q_{\mathrm{KGen}}]$.*

Proof. We will prove the theorem with a sequence of hybrid experiments.

Experiment $\mathsf{Hyb}_\ell^\blacktriangledown$. The experiment $\mathsf{Hyb}_\ell^\blacktriangledown$ is almost the same as $\widetilde{\mathsf{Hyb}}_\ell$, except with the following modification. During the t-th **Enc** query for $t = 1, 2, \ldots$, let $\mathcal{O}_t$ be the online set specified by $\mathcal{A}$ during time step t:

1. Choose random $\{\widetilde{T}_i\}_{i\in\mathcal{H}}$ from $\mathbb{G}$ subject to the constraint that

$$\prod_{i\in\mathcal{H}\cap\mathcal{O}_t} \widetilde{T}_i \cdot \prod_{j\in\mathcal{K}\cap\mathcal{O}_t} [\![\mathsf{CPRF.Eval}(K_j, t, \mathcal{O}_t) \cdot \rho^*]\!] = 1 \tag{7}$$

2. When $[\![\langle \mathbf{c}_i, \mathbf{k}_i^* \rangle]\!] := [\![\langle \mathbf{c}_{i,1}, \mathbf{k}_{i,1}^* \rangle + \langle \mathbf{c}_{i,2}, \mathbf{k}_{i,2}^* \rangle + \langle \widetilde{\mathbf{c}}_i, \widetilde{\mathbf{k}}_i^* \rangle]\!]$ is computed and passed as input to $\widetilde{\mathsf{FE.KGen}}$, replace the term $[\![\langle \widetilde{\mathbf{c}}_i, \widetilde{\mathbf{k}}_i^* \rangle]\!]$ with $\widetilde{T}_i$ where $i \in \mathcal{H} \cap \mathcal{O}_t$.

Lemma B.4. *Suppose that the Decisional Linear assumption holds in $\mathbb{G}$, and that the* PRF *employed by the* CPRF *satisfies pseudorandomness. Then, any non-uniform p.p.t. adversary's views in $\widetilde{\mathsf{Hyb}}_\ell$ and $\mathsf{Hyb}_\ell^\blacktriangledown$ are computationally indistinguishable.*

Proof. Henceforth, let $\{i_1, i_2, \ldots, i_d\}$ denote the set of honest clients where $i_1 < i_2 < \ldots < i_d$ and $d = |\mathcal{H}|$. We consider a sequence of hybrid experiments, where in the j-th hybrid H^j where $j \in [d-1] \cup \{0\}$,

1. Regardless of j, we make the following modification. In every time step t, for every $i \in \mathcal{H} \cap \mathcal{O}_t$, replace the call to $\mathsf{CPRF}.\mathbf{Eval}(K_i, t, \mathcal{O}_t)$ with a group element $R_{i,t}$ chosen at random subject to the following constraint:
$$\sum_{i \in \mathcal{H} \cap \mathcal{O}_t} R_{i,t} + \sum_{j \in \mathcal{K} \cap \mathcal{O}_t} \mathsf{CPRF}.\mathbf{Eval}(K_j, t, \mathcal{O}_t) = 0$$
2. If i is among the first j clients in $\mathcal{H} \cap \mathcal{O}_t$ and is not the last client in $\mathcal{H} \cap \mathcal{O}_t$ (assuming lexicographically ordering), choose $\widetilde{T}_i \xleftarrow{\$} \mathbb{G}$ at random.
3. If $i \in \mathcal{H} \cap \mathcal{O}_t$, but is not among the first j clients in $\mathcal{H} \cap \mathcal{O}_t$ and is not the last client in $\mathcal{H} \cap \mathcal{O}_t$, then the experiment chooses the $\widetilde{T}_i$ value in each time step t as follows: $\widetilde{T}_i := [\![R_{i,t} \cdot \rho^*]\!]$.
4. If i is the last client in $\mathcal{H} \cap \mathcal{O}_t$, then choose $\widetilde{T}_i$ such that the following constraint is satisfied:
$$\prod_{i \in \mathcal{H} \cap \mathcal{O}_t} \widetilde{T}_i \cdot \prod_{i \in \mathcal{K}} [\![\mathsf{CPRF}.\mathbf{Eval}(K_i, t, \mathcal{O}_t) \cdot \rho^*]\!] = 1$$

Observe that H^0 is computationally indistinguishable from $\widetilde{\mathsf{Hyb}}_\ell$, and the argument is similar to to the proof of Theorem 4.2. Further, H^{d-1} the same as $\mathsf{Hyb}^{\blacktriangledown}_\ell$. Therefore, it suffices to show that no non-uniform p.p.t. adversary can distinguish between any two adjacent hybrids H^{j^*} and H^{j^*+1} except with negligible probability, for $j^* \in \{0, 1, \ldots, d-2\}$. Suppose there is an efficient adversary $\mathcal{A}$ that can distinguish H^{j^*} and H^{j^*+1} with non-negligible probability, we show how to construct an efficient reduction $\mathcal{B}$ that can break the Decisional Linear assumption.

Suppose that $\mathcal{B}$ obtains an instance $([\![1]\!], [\![\beta]\!], [\![\gamma]\!], [\![\mathbf{u}]\!], [\![\beta\mathbf{v}]\!], [\![\mathbf{z}]\!])$ from a Vector Decisional Linear challenger, where $\mathbf{u}, \mathbf{v}, \mathbf{z} \in \mathbb{Z}_q^{Q_{\mathrm{enc}}}$ and $\beta, \gamma \in \mathbb{Z}_q$. $\mathcal{B}$'s task is to distinguish whether $[\![\mathbf{z}]\!] = [\![\gamma(\mathbf{u} + \mathbf{v})]\!]$ or random. $\mathcal{B}$ will now interact with $\mathcal{A}$ and embed this Decisional Linear instance in its answers.

- **Setup.** When running $(\mathsf{mpk}', \mathsf{msk}', \{\mathsf{ek}'_i\}_{i \in [n]}) \leftarrow \mathsf{MCFE}.\mathbf{Setup}(\mathsf{pp}, m, n)$, for every $i \in \mathcal{H}$, $\mathcal{B}$ chooses $\xi_i \in \mathbb{Z}_q$ at random, and implicitly sets the term $a_i := \xi_i \cdot \beta^{-1}$ without actually computing them — note that $\beta \neq 0$ with all but negligible probability, so we can ignore the event that $\beta = 0$. $\mathcal{B}$ runs the rest of the $\mathsf{MCFE}.\mathbf{Setup}$ honestly. Note that all $\mathcal{B}$ can compute all terms of mpk' and $\{\mathsf{ek}'_i\}_{i \in \mathcal{K}}$.
- **KGen queries.**
  1. The first $(\ell - 1)$ **KGen** queries will be answered with the honest **KGen** algorithm using $\mathbf{y}^{(1)}$ as the input. However, we need to explain how to still compute the correct **KGen** algorithm without knowing a_i for $i \in \mathcal{H}$. Recall that $\mathsf{MCFE}.\mathbf{KGen}(\mathsf{mpk}', \mathsf{msk}', \mathbf{y}^{(1)})$ gives a key of the form $\{\mathbf{k}_i := (\mathbf{k}_{i,1}, \mathbf{k}_{i,2}, \widetilde{\mathbf{k}}_i)\}_{i \in [n]}$.
     - For all indices $i \in \mathcal{K}$, $\mathcal{B}$ knows all necessary terms to compute $\mathbf{k}_i := (\mathbf{k}_{i,1}, \mathbf{k}_{i,2}, \widetilde{\mathbf{k}}_i)$ normally.
     - For every $i \in \mathcal{H}$, the terms $\mathbf{k}_{i,1}, \mathbf{k}_{i,2}$ need not use a_i and can be computed normally. We thus focus on the term $\widetilde{\mathbf{k}}_i$ which is supposed to be of the form $(\rho, -\rho \cdot a_i)$ where ρ is chosen at random for each **KGen**. $\mathcal{B}$ implicitly use the terms $[\![\beta\rho', -\xi_i\rho']\!]$ in place of $[\![\rho, -\rho \cdot a_i]\!]$. Note that $\mathcal{B}$ does not know the exponents but it can nonetheless complete the remainder of the computation, since it will next compute $\mathsf{FE}.\mathbf{Enc}(\mathsf{mpk}_i, [\![\mathbf{k}_i]\!])$ and this step only needs the group encoding of these elements.

2. Any **KGen** query after the first ℓ queries will be answered with the honest **KGen** algorithm using $\mathbf{y}^{(0)}$ as input. Using the same argument as above, although $\mathcal{B}$ does not know $\{a_i\}_{i\in\mathcal{H}}$, it can still compute these keys.
3. We now focus on the ℓ-th **KGen** query. For the ℓ-th **KGen** query:
   - Let $\mathbf{y}^{(0)}$ and $\mathbf{y}^{(1)}$ be the two vectors submitted by $\mathcal{A}$ during the ℓ-th **KGen** query, let $\mathbf{y}^* := \mathbf{y}^{(0)} \in \mathbb{Z}_q^{mn}$;
   - For $i \in \mathcal{H}$, $\mathcal{B}$ calls $\overline{\mathbf{k}}_i^* := \mathsf{FE}.\widetilde{\mathbf{Enc}}(\mathsf{msk}_i)$;
   - We now focus on how to compute the corrupt components $\overline{\mathbf{k}}_i^*$ where $i \in \mathcal{K}$. $\mathcal{B}$ wants to implicitly embed the γ term from the Decisional Linear challenge into the ρ term in the ℓ-th **KGen** query. However, $\mathcal{B}$ knows only $[\![\gamma]\!]$ but not the exponent γ. However, this is not a problem because $\mathcal{A}$ knows the a_i terms for $i \in \mathcal{K}$, and thus it can compute the term $[\![\widetilde{\mathbf{k}}_i]\!] := [\![\rho, -\rho a_i]\!] = [\![\gamma, -\gamma a_i]\!]$. Moreover, the following step $\mathsf{FE}.\mathbf{Enc}(\mathsf{mpk}_i, [\![\mathbf{k}_i]\!])$ only needs knowledge of the group encoding of $\mathbf{k}_i$.

- **Enc queries.** For the t-th **Enc** query with the submitted tuple $(\mathcal{O}, \{\mathbf{x}_i^{(0)}\}_{i\in\mathcal{H}\cap\mathcal{O}}, \{\mathbf{x}_i^{(1)}\}_{i\in\mathcal{H}\cap\mathcal{O}})$, do the following. Let i^* be the (j^*+1)-th client in $\mathcal{H}\cap\mathcal{O}$. Note that i^* may change between different time steps t.

  - If $|\mathcal{H}\cap\mathcal{O}| > i^*$, then $\mathcal{B}$ will compute $\mathsf{ct}_{i^*} := ([\![\mathbf{c}_{i^*,1}]\!], [\![\mathbf{c}_{i^*,2}]\!], [\![\widetilde{\mathbf{c}}_{i^*}]\!])$ as follows. The terms $[\![\mathbf{c}_{i^*,1}]\!], [\![\mathbf{c}_{i^*,2}]\!]$ are computed honestly. The term $[\![\widetilde{\mathbf{c}}_{i^*}]\!]$ is generated as follows: let u_t, v_t, z_t be the t-th coordinate of $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{z}$, respectively. We will let

    $$[\![\widetilde{\mathbf{c}}_{i^*}]\!] := [\![u_t, -\xi_{i^*}^{-1}\beta v_t]\!],$$

    Now, let $\overline{\mathsf{ct}}_{i^*} := \mathsf{FE}.\widetilde{\mathbf{KGen}}(\mathsf{msk}_{i^*}, \mathsf{ct}_{i^*}, [\![\langle \mathbf{c}_{i^*,1}, \mathbf{k}_{i^*,1}^*\rangle + \langle \mathbf{c}_{i^*,2}, \mathbf{k}_{i^*,2}^*\rangle + z_t]\!])$.
    In other words, we are implicitly letting

    $$R_{i^*,t} + a_{i^*}\mu_{i^*,t} = u_t, \quad \mu_{i^*,t} = -\xi_{i^*}^{-1}\beta v_t$$

    where $\mu_{i^*,t}$ is the μ term chosen by i^* in the t-th **Enc** query. Thus $R_{i^*,t}\cdot\gamma = (u_t - a_{i^*}\mu_{i^*,t})\gamma = (u_t - \xi_{i^*}\beta^{-1}\cdot\mu_{i^*,t})\gamma = (u_t + \xi_{i^*}\beta^{-1}\cdot\xi_{i^*}^{-1}\beta v_t)\gamma = (u_t + v_t)\gamma$.
  - For every honest client $i \in (\mathcal{H}\cap\mathcal{O})\backslash\{i^*\}$ and i is not the last client in $\mathcal{H}\cap\mathcal{O}$, $\mathcal{B}$ will compute $\mathsf{ct}_i := ([\![\mathbf{c}_{i,1}]\!], [\![\mathbf{c}_{i,2}]\!], [\![\widetilde{\mathbf{c}}_i]\!])$ as follows. The terms $[\![\mathbf{c}_{i,1}]\!], [\![\mathbf{c}_{i,2}]\!]$ are computed honestly. We now focus on the term $[\![\widetilde{\mathbf{c}}_i]\!]$. We implicitly choose $\mu_{i,t} := a_i^{-1}\phi_{i,t}$ where $\phi_{i,t} \xleftarrow{\$} \mathbb{Z}_q$ is chosen at random. Now, we compute $[\![\widetilde{\mathbf{c}}_i]\!] := [\![R_{i,t} + a_i\mu_{i,t}, \mu_{i,t}]\!]$ as follows:

    $$[\![R_{i,t} + a_i\mu_{i,t}]\!] = [\![R_{i,t} + a_i a_i^{-1}\phi_{i,t}]\!] = [\![R_{i,t} + \phi_{i,t}]\!]$$

    $$[\![\mu_{i,t}]\!] = [\![a_i^{-1}\phi_{i,t}]\!] = [\![\xi_i\phi_{i,t}\cdot\beta]\!]$$

    Notice that both terms can be computed since the exponents $R_{i,t}, w, \phi_{i,t}, \xi_i$ are known, and $[\![\beta]\!]$ is known.

    Next, let $\mathsf{ct}_i := ([\![\mathbf{c}_{i,1}]\!], [\![\mathbf{c}_{i,2}]\!], [\![\widetilde{\mathbf{c}}_i]\!])$. $\mathcal{B}$ calls $\mathsf{FE}.\widetilde{\mathbf{KGen}}(\mathsf{msk}_i, \mathsf{ct}_i, \widetilde{T}_i)$ where $\widetilde{T}_i$ is chosen like in H^{j^*}. Note that $\mathcal{B}$ does not know $\rho^* = \gamma$, but since it knows $R_{i,t}$, it will be able to compute $\widetilde{T}_i$ whether it is chosen at random or chosen as $[\![R_{i,t}\cdot\gamma]\!]$.
  - For the last honest client in $\mathcal{H}\cap\mathcal{O}$ henceforth denoted i', $\mathcal{B}$ will compute the terms $[\![\mathbf{c}_{i',1}]\!], [\![\mathbf{c}_{i',2}]\!]$ honestly. We now focus on how to compute $[\![\widetilde{\mathbf{c}}_{i'}]\!]$. There are two cases. First, if $|\mathcal{H}\cap\mathcal{O}| \le i^*$,

in this case, $\mathcal{B}$ can compute $R_{i',t}$ because it knows all other $\{R_{i,t}\}_{i\in\mathcal{H}\cap\mathcal{O},i\neq i'}$. Therefore, it can compute $[\![\widetilde{\mathbf{c}}_{i'}]\!]$ just like it did for $i \in (\mathcal{H}\cap\mathcal{O})\backslash\{i^*\}$ earlier.

Second, if $|\mathcal{H}\cap\mathcal{O}| > i^*$, in this case, $\mathcal{B}$ does not know $R_{i^*,t}$. $\mathcal{B}$ computes $[\![\widetilde{\mathbf{c}}_{i'}]\!]$ as below. $\mathcal{B}$ samples a random $\phi \in \mathbb{Z}_q$ and implicitly chooses

$$\mu_{i',t} = -\xi_{i'}^{-1}\cdot\xi_{i^*}\cdot\mu_{i^*,t} + a_{i^*}^{-1}\cdot\phi, \quad R_{i',t} := -\left(\sum_{j\in\mathcal{H}\cap\mathcal{O},j\neq i'} R_{j,t} + \sum_{j\in\mathcal{K}} \mathsf{CPRF.Eval}(K_j,t,\mathcal{O})\right)$$

We may write $R_{i',t}$ as $R_{i',t} := \nu - R_{i^*,t}$ where

$$\nu := -\left(\sum_{j\in\mathcal{H}\cap\mathcal{O},j\neq i',j\neq i^*} R_{j,t} + \sum_{j\in\mathcal{K}} \mathsf{CPRF.Eval}(K_j,t,\mathcal{O})\right) \in \mathbb{Z}_q \text{ is known to } \mathcal{B}.$$

Now, $\mathcal{B}$ will compute $[\![\widetilde{\mathbf{c}}_{i_d}]\!]$ as follows:

$$\begin{aligned}
[\![R_{i',t} + a_{i'}\mu_{i',t}]\!] &= [\![\nu - R_{i^*,t} + \xi_{i'}\cdot\xi_{i^*}^{-1}\cdot a_{i^*}\cdot(-\xi_{i'}^{-1}\cdot\xi_{i^*}\mu_{i^*,t} + a_{i^*}^{-1}\cdot\phi)]\!] \\
&= [\![\nu - R_{i^*,t} - a_{i^*}\mu_{i^*,t} + \xi_{i'}\cdot\xi_{i^*}^{-1}\cdot\phi]\!] \\
&= [\![\nu - u_t + \xi_{i'}\xi_{i^*}^{-1}\cdot\phi]\!]
\end{aligned}$$

which can be computed knowing $\nu, [\![u_t]\!]$, and $\xi_{i'}\xi_{i^*}^{-1}\phi$.

Further,

$$\begin{aligned}
[\![\mu_{i',t}]\!] &= [\![\cdot(-\mu_{i^*,t}\cdot\xi_{i'}^{-1}\cdot\xi_{i^*} + a_{i^*}^{-1}\cdot\phi)]\!] \\
&= [\![\xi_{i^*}^{-1}\cdot\beta v_t\cdot\xi_{i'}^{-1}\cdot\xi_{i^*} + \phi\cdot a_{i^*}^{-1}]\!] \\
&= [\![\beta v_t\cdot\xi_{i'}^{-1} + \phi\cdot(\xi_{i^*})^{-1}\beta]\!]
\end{aligned}$$

which can be computed knowing $[\![\beta v_t]\!]$, $[\![\beta]\!]$ and the exponents $\xi_{i'}$, ξ_{i^*}, and ϕ.

Next, let $\mathsf{ct}_{i'} := ([\![\mathbf{c}_{i',1}]\!], [\![\mathbf{c}_{i',2}]\!], [\![\widetilde{\mathbf{c}}_{i'}]\!])$. $\mathcal{B}$ calls $\widetilde{\mathsf{FE.KGen}}(\mathsf{msk}_{i'}, \mathsf{ct}_{i'}, \widetilde{T}_{i'})$ where $\widetilde{T}_{i'}$ is chosen such that $\prod_{i\in\mathcal{H}\cap\mathcal{O}}\widetilde{T}_i \cdot \prod_{j\in\mathcal{K}}[\![\mathsf{CPRF.Eval}(K_j,t,\mathcal{O})\cdot\gamma]\!] = 1$ — we let $\widetilde{T}_{i^*} := [\![z_t]\!]$; and for $i \in \mathcal{H}\cap\mathcal{O}, i\neq i', i\neq i^*$, let $\widetilde{T}_i$ be chosen like in H^{j^*}.

- Finally, and return $\{(\mathsf{ct}_i, \overline{\mathsf{ct}}_i)\}_{i\in\mathcal{H}\cap\mathcal{O}}$ to $\mathcal{A}$.

If the Vector Decisional Linear tuple $([\![1]\!], [\![\beta]\!], [\![\gamma]\!], [\![\mathbf{u}]\!], [\![\beta\mathbf{v}]\!], [\![\mathbf{z}]\!])$ satisfies $[\![\mathbf{z}]\!] = [\![\gamma(\mathbf{u}+\mathbf{v})]\!]$, then $\mathcal{A}$'s view in the above experiment is identical to H^{j^*}. Else, if $[\![\mathbf{z}]\!]$ is randomly chosen, then $\mathcal{A}$'s view in the above experiment is identical to H^{j^*+1}. □

Experiment $\mathsf{Hyb}_\ell^\triangledown$. The hybrid $\mathsf{Hyb}_\ell^\triangledown$ is defined almost identically as $\mathsf{Hyb}_\ell^\blacktriangledown$, except that in the ℓ-th **KGen** query, the challenger switches to using $\mathbf{y}^{(1)}$ instead of $\mathbf{y}^{(0)}$.

Using the same idea as the proof of Claim 5.8, we can show that $\mathsf{Hyb}_\ell^\triangledown$ and and $\mathsf{Hyb}_\ell^\blacktriangledown$ are identically distributed.

Now, using a symmetric argument, we can complete the proof of Lemma B.3. □

At this moment, it is not hard to see that Theorem B.2 follows in the same way as the proof of Theorem 5.3. □

Fully function-hiding construction $\mathsf{MCFE}^{\mathrm{ffh}}$. We can now upgrade from weak function privacy to full function privacy just like in Section 5.4. For completeness, we present the construction below, and the only difference here is that **Enc** and **Dec** are now aware of $\mathcal{O}$:

$\mathsf{MCFE}^{\mathrm{ffh}}$: **fault-tolerant, fully function-hiding MCFE for selection**

- $\mathbf{Gen}(1^\kappa)$ and $\mathbf{Setup}(\mathsf{pp}, m, n)$: Same as in Section 5.4.
- $\mathbf{Enc}(\mathsf{mpk}, \mathsf{ek}_i, \mathbf{x}, t, \mathcal{O})$: call $\mathsf{CT} := \mathsf{MCFE}^{\mathrm{wfh}}.\mathbf{Enc}(\mathsf{mpk}, \mathsf{ek}_i, \mathbf{x}||\mathbf{0}, t, \mathcal{O})$ where $\mathbf{0} \in \mathbb{Z}_q^m$, and output CT.
- $\mathbf{KGen}(\mathsf{mpk}, \mathsf{msk}, \mathbf{y})$: Same as in Section 5.4.
- $\mathbf{Dec}(\mathsf{mpk}, \mathsf{sk}_{\mathbf{y}}, \mathcal{O}, \{\mathsf{CT}_i\}_{i\in\mathcal{O}})$: call $x := \mathsf{MCFE}^{\mathrm{wfh}}.\mathbf{Dec}(\mathsf{mpk}, \mathsf{sk}_{\mathbf{y}}, \mathcal{O}, \{\mathsf{CT}_i\}_{i\in\mathcal{O}})$ and output x.

Theorem B.5 (Fault-tolerant, fully function-hiding MCFE for selection)**.** *Assume that* $\mathsf{MCFE}^{\mathrm{wfh}}$ *is weakly-function-hiding IND-secure. Then, the above construction is function-hiding IND-secure.*

Proof. We can prove the theorem with a sequence of hybrid experiments as described by the following table.

Experiment	**Enc** query for $i \in \mathcal{H} \cap \mathcal{O}_t$	**KGen** query for $i \in \mathcal{H}$	**KGen** query for $i \in \mathcal{K}$
$\mathsf{FH\text{-}Expt}^0$	$\mathbf{x}_i^{(0)}\|\|\mathbf{0}$	$\mathbf{y}_i^{(0)}\|\|\mathbf{0}$	$\mathbf{y}_i^{(0)}\|\|\mathbf{0} = \mathbf{y}_i^{(1)}\|\|\mathbf{0}$
Hyb_1	$\mathbf{x}_i^{(0)}\|\|\mathbf{x}_i^{(1)}$	$\mathbf{0}\|\|\mathbf{y}_i^{(1)}$	
Hyb_2	$\mathbf{x}_i^{(1)}\|\|\mathbf{x}_i^{(1)}$	$\mathbf{y}_i^{(1)}\|\|\mathbf{0}$	
$\mathsf{FH\text{-}Expt}^1$	$\mathbf{x}_i^{(1)}\|\|\mathbf{0}$	$\mathbf{y}_i^{(1)}\|\|\mathbf{0}$	

Any non-uniform p.p.t. adversary $\mathcal{A}$'s views in any pair of adjacent hybrids are computationally indistinguishable due to the fact that $\mathsf{MCFE}^{\mathrm{wfh}}$ is weakly-function-hiding IND-secure. Therefore, the theorem follows. □

B.2 Fault-Tolerant NIAR with Receiver-Insider Protection

We can construct a fault-tolerant NIAR scheme from a fault-tolerant, function-hiding MCFE-for-selection scheme denoted $\mathsf{MCFE}^{\mathrm{ffh}}$. We sketched the construction earlier in Section 8.2. For completeness, we present the full construction below.

Fault-tolerant NIAR with receiver-insider protection

- $\mathbf{Setup}(1^\kappa, n, \pi)$: Same as in Section 6.
- $\mathbf{Enc}(\mathsf{ek}_i, x_i, t, \mathcal{O})$: Let $c := \mathsf{SE}.\mathbf{Enc}(\mathsf{sk}_i, x_i)$, and henceforth let $(c)_j$ denote the j-th bit of c. For $j \in [L]$ where $L := |c|$, let $(\mathsf{ct}_{i,t})_j := \mathsf{MCFE}^{\mathrm{ffh}}.\mathbf{Enc}(\mathsf{mpk}, \mathsf{ek}'_i, (c)_j, (t-1)L + j, \mathcal{O})$. Output $\mathsf{ct}_{i,t} := \{(\mathsf{ct}_{i,t})_j\}_{j\in[L]}$.
- $\mathbf{Rte}(\mathsf{tk}, \mathcal{O}, \{\mathsf{ct}_{i,t}\}_{i\in\mathcal{O}})$: For $i \in [n]$, for $j \in [L]$ where L is the length of an SE ciphertext, let $(\mathsf{ct}'_{i,t})_j := \mathsf{MCFE}^{\mathrm{ffh}}.\mathbf{Dec}(\mathsf{mpk}, \mathsf{tk}_i, \mathcal{O}, \{(\mathsf{ct}_{1,t})_j\}_{i\in\mathcal{O}})$ and let $\mathsf{ct}'_{i,t} := \{(\mathsf{ct}'_{i,t})_j\}_{j\in[L]}$. Output $\{\mathsf{ct}'_{i,t}\}_{i\in[n]}$.

- $\mathbf{Dec}(\mathsf{rk}_i, \mathsf{ct}'_{i,t})$: Output $x := \mathsf{SE}.\mathbf{Dec}(\mathsf{rk}_i, \mathsf{ct}'_{i,t})$. We may assume that $\mathsf{SE}.\mathbf{Dec}$ always outputs $\perp$ upon receiving the ciphertext $\mathsf{ct}'_{i,t} = \mathbf{0}$.

Theorem B.6 (Fault-tolerant NIAR with receiver-insider protection)**.** *Suppose that the* SE *scheme satisfies semantic security as defined in Appendix C.1, and that* $\mathsf{MCFE}^{\mathrm{ffh}}$ *is function-hiding IND-secure. Then, the above fault-tolerant NIAR scheme above is SIM-secure w.r.t. the leakage function* $\mathsf{Leak}^{\mathrm{S}^*}$.

Proof. Due to Lemma 8.1, it suffices to prove that the NIAR scheme above is IND-secure w.r.t. the leakage function $\mathsf{Leak}^{\mathrm{S}^*}$. Henceforth we use the security experiments $\mathsf{NIAR\text{-}Expt}^b(1^\kappa)$ defined in Section 8.1. We will consider a following sequence of hybrid experiments to show that any non-uniform p.p.t.*admissible* adversary's views in $\mathsf{NIAR\text{-}Expt}^0(1^\kappa)$ and $\mathsf{NIAR\text{-}Expt}^1(1^\kappa)$ are computationally indistinguishable.

Experiment Hyb^b **for** $b \in \{0,1\}$**.** For $b \in \{0,1\}$, we define Hyb^b just like $\mathsf{NIAR\text{-}Expt}^b(1^\kappa)$ except with the following modification: Henceforth let $\mathsf{HH}^b \subseteq \mathcal{H}_S$ denote the set of senders who communicate with honest receivers as defined by $\pi^{(b)}$, and let $\mathsf{HK}^b = \mathcal{H}_S \backslash \mathsf{HH}^b$ denote the set of honest senders who talk to corrupt receivers as defined by $\pi^{(b)}$. Now, during the t-th **Enc** query made by the adversary $\mathcal{A}$ for $t = 1, 2, \ldots$, the query is answered as follows — below let $\mathsf{SE.Sim}$ be the simulator for the SE scheme, and let $(\mathcal{O}_t, \{x^{(0)}_{i,t}\}_{i\in\mathcal{H}}, \{x^{(1)}_{i,t}\}_{i\in\mathcal{H}})$ be the tuple submitted by $\mathcal{A}$ during time step t:

- For $i \in \mathsf{HH}^b \cap \mathcal{O}_t$: let $c_i := \mathsf{SE.Sim}(1^\kappa)$, and for $j \in [L]$, let $(\mathsf{ct}_{i,t})_j := \mathsf{MCFE}^{\mathrm{ffh}}.\mathbf{Enc}(\mathsf{mpk}, \mathsf{ek}'_i, (c_i)_j, (t-1)L + j, \mathcal{O}_t)$.
- For $i \in \mathsf{HK}^b \cap \mathcal{O}_t$: compute the ciphertext $\mathsf{ct}_{i,t}$ honestly using $x^{(b)}_{i,t}$ as input.

Claim B.7. *Suppose that* SE *satisfies semantic security (and let* $\mathsf{SE.Sim}$ *be the corresponding simulator for semantic security), then for* $b \in \{0,1\}$*, any non-uniform p.p.t. adversary's views in* Hyb^b *and* $\mathsf{NIAR\text{-}Expt}^b$ *are computationally indistinguishable.*

Proof. Follows from a straightforward reduction. □

Fact B.8. *Due to the admissibility rules imposed on* $\mathcal{A}$ *(and recall that we are using* $\mathsf{Leak}^{\mathrm{S}^*}$*), we have the following:* $\pi^{(0)}(\mathcal{H}_S) = \pi^{(1)}(\mathcal{H}_S)$*, and* $\pi^{(0)}(\mathcal{H}_S \cap \mathcal{O}_t) = \pi^{(1)}(\mathcal{H}_S \cap \mathcal{O}_t)$.

Lemma B.9. *Suppose that the* $\mathsf{MCFE}^{\mathrm{ffh}}$ *scheme is function-hiding IND-secure, then no non-uniform p.p.t. adversary* $\mathcal{A}$ *that respects the admissibility rules of the NIAR IND-security game can distinguish* Hyb^0 *and* Hyb^1 *except with negligible probability.*

Proof. In experiment Hyb^b, let $\psi := \{\psi_i\}_{i\in\pi^{(b)}(\mathcal{H}_S)}$ where ψ_i is defined as the following:

- if $i \in \mathcal{K}_R$, ψ_i is the random coins consumed by the $\mathsf{SE}.\mathbf{Gen}$ and $\mathsf{SE}.\mathbf{Enc}$ algorithms pertaining to the j-th sender where $j := (\pi^{(b)})^{-1}(i)$.
- if $i \in \mathcal{H}_R$, then ψ_i is the random coins consumed by the $\mathsf{SE.Sim}$ algorithms pertaining to the j-th sender where $j := (\pi^{(b)})^{-1}(i)$.

Due to Fact B.8, $\pi^{(0)}(\mathcal{H}_S) = \pi^{(1)}(\mathcal{H}_S)$, therefore, ψ has the same format regardless of b. Let $\mathsf{Hyb}^b(\psi)$ be the experiment Hyb^b but when the aforementioned random coins are fixed to ψ. It suffices to show that for any fixed ψ, $\mathsf{Hyb}^0(\psi)$ and $\mathsf{Hyb}^1(\psi)$ are computationally indistinguishable as long as the non-uniform p.p.t.$\mathcal{A}$ respects the NIAR game's admissibility rules. We show that if

a non-uniform p.p.t.$\mathcal{A}$ that respects the NIAR game's admissibility rules can distinguish $\mathsf{Hyb}^0(\psi)$ and $\mathsf{Hyb}^1(\psi)$ with non-negligible probability, we can build a reduction $\mathcal{B}$ that breaks the fully function-hiding IND-security of the $\mathsf{MCFE}^{\mathsf{ffh}}$ scheme.

- First, $\mathcal{A}$ outputs $n, \mathcal{K}_S, \mathcal{K}_R, \pi^{(0)}, \pi^{(1)}$.
- Next, $\mathcal{B}$ needs to simulate the NIAR scheme's **Setup** phase for $\mathcal{A}$. $\mathcal{B}$ obtains mpk and $\{\mathsf{ek}'_i\}_i \in \mathcal{K}_S$ from its own challenger, and embeds these parameters into the **Setup**.

  Next, for every $i \in [n]$, $\mathcal{B}$ makes **KGen** queries to its own challenger to generate functional secret keys for the pair of vectors $\mathbf{e}_{j_0}$ and $\mathbf{e}_{j_1}$ where for $b \in \{0,1\}$, $j_b := (\pi^{(b)})^{-1}(i)$; let the result be $\{\mathsf{tk}_i\}_{i\in[n]}$. $\mathcal{B}$ uses these tokens in the setup.

  Due to Fact B.8, $\mathcal{K}_R \cap \pi^{(0)}(\mathcal{H}_S) = \mathcal{K}_R \cap \pi^{(1)}(\mathcal{H}_S)$. For an honest sender communicating with a corrupt receiver, $\mathcal{B}$ uses the appropriate coins inside ψ to run their $\mathsf{SE}.\mathbf{Gen}$.
- During each time step $t \in \mathbb{N}$, $\mathcal{A}$ submits the tuple $(\mathcal{O}_t, \{x^{(0)}_{i,t}\}_{i\in\mathcal{H}_S}, \{x^{(1)}_{i,t}\}_{i\in\mathcal{H}_S})$.
  - For $i \in \mathsf{HH}^b \cap \mathcal{O}_t$, and for $b \in \{0,1\}$, it computes $c^b_i := \mathsf{SE}.\mathsf{Sim}(1^\kappa)$ using the random coins inside ψ for both choices of b.
  - For $i \in \mathsf{HK}^b \cap \mathcal{O}_t$ and for $b \in \{0,1\}$, it computes $c^b_i := \mathsf{SE}.\mathbf{Enc}(\mathsf{sk}_{j_b}, x_i)$ where $j_b := \pi^{(b)}(i)$ using the random coins inside ψ for both choices of b.
  - Now, $\mathcal{B}$ forwards the two vectors $\{c^b_i\}_{i\in\mathcal{H}_S\cap\mathcal{O}_t}$ for $b \in \{0,1\}$ to its own challenger bit by bit in a total of L queries. It returns the concatenated result to $\mathcal{A}$.

  We verify that $\mathcal{B}$ indeed respects the admissibility rules of $\mathsf{MCFE}^{\mathsf{ffh}}$.

1. First, since $\mathcal{A}$ respects the admissiblity rules of the NIAR security game (with receiver-insider protection), the "corrupt $\to *$" part of the permutation must match for $\pi^{(0)}$ and $\pi^{(1)}$. Thus, for every pair of selection vector $\mathcal{B}$ submits to its own challenger, if one selection vector is selecting a corrupt sender's message, the other must be consistent.
2. By Fact B.8, we have that $\pi^{(0)}(\mathcal{H}_S\cap\mathcal{O}_t) = \pi^{(1)}(\mathcal{H}_S\cap\mathcal{O}_t)$. Checking the second admissibility rule for $\mathsf{MCFE}^{\mathsf{ffh}}$ boils down to checking that in the two alternate worlds, every receiver in $\pi^{(0)}(\mathcal{H}_S \cap \mathcal{O}_t) = \pi^{(1)}(\mathcal{H}_S \cap \mathcal{O}_t)$ must receive the same SE ciphertext. Recall that by the admissibility rule imposed on $\mathcal{A}$, the corrupt receivers receiving from $\mathcal{H}_S \cap \mathcal{O}_t$ must receive the same message in the two alternate worlds. Given this, the second admissibility rule for $\mathcal{B}$ is guaranteed by construction due to our definition of ψ.

If the $\mathsf{MCFE}^{\mathsf{ffh}}$ challenger used the bit $b = 0$, then $\mathcal{A}$'s view is identically distributed as in $\mathsf{Hyb}^0(\psi)$; else its view is identically distributed as in $\mathsf{Hyb}^1(\psi)$. □

The proof of Theorem B.6 now follows due to Claim B.7 and Lemma B.9 and the hybrid argument. □

B.3 Fault-Tolerant NIAR with Full Insider Protection

We presented our full insider protection scheme with fault tolerance in Section 8.3. In this section, we prove the security of this construction, that is, Theorem 8.3. We restate the theorem below for the reader's convenience.

Theorem B.10 (Fault-tolerant NIAR scheme with full insider security)**.** *Suppose that* tPKE *is a perfectly hiding trapdoor encryption scheme,* piO *is an indistinguishability obfuscator for perfectly indistinguishable samplers over general circuit families, and moreover,* $\mathsf{MCFE}^{\mathrm{ffh}}$ *is fully function-hiding IND-secure. Then, the fault-tolerant NIAR construction in Section 8.3 is SIM-secure with full insider protection, i.e., it is SIM-secure w.r.t. the leakage* $\mathsf{Leak}^{\min}$ *defined in Section 8.1.2.*

Proof. Due to Lemma 8.1, it suffices to prove that the construction is IND-secure with full insider protection. We will prove this through a sequence of hybrid experiments.

Experiment Hyb_1**.** $\mathsf{Hyb}_1(1^\kappa)$ is almost the same as $\mathsf{NIAR\text{-}Expt}^0(1^\kappa)$ except that during **Setup**, for every $i \in \mathcal{H}_R$, we run the trapdoor key generation of the tPKE scheme to generate its public key. In other words, for $i \in \mathcal{H}_R$, let $\mathsf{epk}_i \leftarrow \mathsf{tPKE}.\widetilde{\mathbf{Gen}}(1^\kappa)$.

With a straightforward reduction to tPKE's security, and specifically, the indistinguishability of the normal and the trapdoor modes, we can show that any non-uniform p.p.t. adversary $\mathcal{A}$'s views in $\mathsf{NIAR\text{-}Expt}^0$ and Hyb_1 are computationally indistinguishable.

Experiment Hyb_2**.** Hyb_2 is almost the same as Hyb_1 except with the following modification: during **Setup**, instead of calling $\mathsf{tk} := \mathsf{piO}(1^\kappa, P^{\mathsf{mpk},\{\mathsf{epk}_i,\mathsf{tk}_i\}_{i\in[n]}})$, we call

$$\mathsf{tk} := \mathsf{piO}\left(1^\kappa, \widetilde{P}^{\mathsf{mpk},\{\mathsf{epk}_i\}_{i\in[n]},\{\mathsf{tk}_i\}_{i\in\mathcal{K}_R}}\right)$$

where the program $\widetilde{P}^{\mathsf{mpk},\{\mathsf{epk}_i\}_{i\in[n]},\{\mathsf{tk}_i\}_{i\in\mathcal{K}_R}}$ is defined below.

Probabilistic program $\widetilde{P}^{\mathsf{mpk},\{\mathsf{epk}_i\}_{i\in[n]},\{\mathsf{tk}_i\}_{i\in\mathcal{K}_R}}(\mathsf{ct}_1,\ldots,\mathsf{ct}_n,\mathcal{O})$

Hardwired: $\mathsf{mpk}, \{\mathsf{epk}_i\}_{i\in[n]}, \{\mathsf{tk}_i\}_{i\in\mathcal{K}_R}$

- For $i \in \mathcal{K}_R$, let $x_i := \mathsf{MCFE}^{\mathrm{ffh}}.\mathbf{Dec}(\mathsf{mpk}, \mathsf{tk}_i, \mathcal{O}, \{\mathsf{ct}_i\}_{i\in\mathcal{O}})$; and let $\widehat{\mathsf{ct}}_i := \mathsf{tPKE}.\mathbf{Enc}(\mathsf{epk}_i, x_i)$.
- For $i \in \mathcal{H}_R$: let $\widehat{\mathsf{ct}}_i := \mathsf{tPKE}.\mathbf{Enc}(\mathsf{epk}_i, 0)$.
- Output $\{\widehat{\mathsf{ct}}_i\}_{i\in[n]}$.

Observe that in Hyb_2, during **Setup**, in fact, for any $i \in \mathcal{H}_R$, there is in fact no need to call $\mathsf{tk}_i := \mathsf{MCFE}^{\mathrm{ffh}}.\mathbf{KGen}(\mathsf{mpk}, \mathsf{msk}, \mathbf{e}_j)$ where $j := \pi^{-1}(i)$, since $\{\mathsf{tk}_i\}_{i\in\mathcal{H}_R}$ will never be used later.

In the same fashion as in Claim 7.4, we can show the following: suppose that tPKE is a perfectly hiding trapdoor encryption scheme, and that piO is an indistinguishability obfuscator for perfectly indistinguishable samplers over general circuit families, then any non-uniform p.p.t. adversary $\mathcal{A}$'s views in Hyb_1 and Hyb_2 are computationally indistinguishable.

Experiment Hyb_3**.** Hyb_3 is defined almost identically as Hyb_2 except that now, the NIAR challenger switches to using $\pi^{(1)}$ during **Setup** and to using $\{x^{(1)}_{i,t}\}_{i\in\mathcal{H}_S\cap\mathcal{O}_t}$ for answering the online routing queries.

Claim B.11. *Suppose that* $\mathsf{MCFE}^{\mathrm{ffh}}$ *is fully function-hiding IND-secure. Then, no non-uniform p.p.t. adversary* $\mathcal{A}$ *that respects the admissibility rules of the NIAR IND-security game can distinguish* Hyb_2 *and* Hyb_3 *except with negligible probability.*

Proof. If there is a non-uniform p.p.t. adversary $\mathcal{A}$ that respects the admissibility rules of the NIAR IND-security game who can distinguish Hyb_2 and Hyb_3 with more than negligible probability, we can build a reduction $\mathcal{B}$ that breaks the fully function-hiding IND-security of the $\mathsf{MCFE}^{\mathsf{ffh}}$ scheme.

- First, the adversary $\mathcal{A}$ outputs $n, \mathcal{C}_S, \mathcal{K}_S, \mathcal{K}_R, \pi^{(0)}$, and $\pi^{(1)}$.
- Now, $\mathcal{B}$ needs to run **Setup**: to do so, it obtains mpk and $\{\mathsf{ek}'_i\}_i \in \mathcal{K}_S$ from its own challenger, and embeds these parameters into the **Setup**. $\mathcal{B}$ runs the tPKE key generation honestly for $i \in \mathcal{K}_R$ but calls the trapdoor key generation for $i \in \mathcal{H}_R$, just like in Hyb_2.

  Next, for every $i \in \mathcal{K}_R$, $\mathcal{B}$ makes **KGen** queries to its own challenger to generate functional secret keys for the pair of vectors $\mathbf{e}_{j_0}$ and $\mathbf{e}_{j_1}$ where for $b \in \{0,1\}$, $j_b := (\pi^{(b)})^{-1}$; let the result be $\{\mathsf{tk}_i\}_{i \in \mathcal{K}_R}$.

  The rest of **Setup** is performed just like in Hyb_2.
- Next, for each time step $t = 1, 2, \ldots$, $\mathcal{A}$ submits a tuple $(\mathcal{O}, \{x^{(0)}_{i,t}\}_{i \in \mathcal{H}_S \cap \mathcal{O}})$ and $\{x^{(1)}_{i,t}\}_{i \in \mathcal{H}_S \cap \mathcal{O}}$. $\mathcal{B}$ forwards the two vectors to its own challenger, and forwards the answer to $\mathcal{A}$.

Below, we verify that $\mathcal{B}$ indeed respects the admissibility rules of the $\mathsf{MCFE}^{\mathsf{ffh}}$'s security game:

1. First, since $\mathcal{A}$ respects the admissibility rules of the NIAR game, it must be that $\pi^{(0)}$ and $\pi^{(1)}$ are consistent when restricting to the "corrupt $\rightarrow$ corrupt" part of the permutation. This means that for every $i \in \mathcal{K}_R$ — consider the corresponding pair of selection vectors $\mathbf{e}_{j_0}$ and $\mathbf{e}_{j_1}$ where for $b \in \{0,1\}$, $j_b := (\pi^{(b)})^{-1}$ — it must be that either 1) $\mathbf{e}_{j_0}$ and $\mathbf{e}_{j_1}$ both do not select from any coordinate belonging to $\mathcal{K}_S$; or 2) they both select the same coordinate belonging to $\mathcal{K}_S$.
2. By the NIAR game's admissibility rule imposed on $\mathcal{A}$, we have that for any $t \in \mathbb{N}$, it must be that $\mathcal{H}_S \cap \mathcal{O}_t \cap (\pi^{(0)})^{-1}(\mathcal{K}_R) = \mathcal{H}_S \cap \mathcal{O}_t \cap (\pi^{(1)})^{-1}(\mathcal{K}_R)$, and moreover, $\mathcal{K}_R \cap \pi^{(0)}(\mathcal{H}_S \cap \mathcal{O}_t) = \mathcal{K}_R \cap \pi^{(1)}(\mathcal{H}_S \cap \mathcal{O}_t)$. Due to the admissibility rule $\mathcal{A}$ must abide by, we have that during any time step $t \in \mathbb{N}$, for any $i \in \mathcal{K}_R \cap \pi^{(0)}(\mathcal{H}_S \cap \mathcal{O}_t) = \mathcal{K}_R \cap \pi^{(1)}(\mathcal{H}_S \cap \mathcal{O}_t)$, $x^{(0)}_{j_0,t} = x^{(1)}_{j_1,t}$ where for $b \in \{0,1\}$, $j_b := (\pi^{(b)})^{-1}(i)$. In other words, in the two alternate worlds $b = 0$ or 1, every corrupt receiver receiving from an honest and online sender must receive the same message. This means that for every selection vector $\mathcal{B}$ submitted as a **KGen** query that selects one of the corrupt receiver's output, it must produce the same output in the two alternate worlds.

Now, if the $\mathsf{MCFE}^{\mathsf{ffh}}$ challenger used $b = 0$, then $\mathcal{A}$'s view is identically distributed as in Hyb_2; else $\mathcal{A}$'s view is identically distributed as in Hyb_3.

□

Completing the proof of Theorem 8.3. Finally, using a symmetric argument as the above, we can prove that no non-uniform p.p.t. adversary that respects the admissibility rules of the NIAR game can distinguish Hyb_3 and $\mathsf{NIAR\text{-}Expt}^1$ except with negligible probability.

□

C Additional Background

C.1 Symmetric-Key Encryption

A symmetric-key encryption scheme $\mathsf{SE} := (\mathbf{Gen}, \mathbf{Enc}, \mathbf{Dec})$ has three possibly randomized algorithms: $\mathsf{sk} \leftarrow \mathbf{Gen}(1^\kappa)$ generates a secret key sk; $c \leftarrow \mathbf{Enc}(\mathsf{sk}, x)$ encrypts a plaintext message

$x \in \{0,1\}^\ell$ where $\ell(\cdot)$ is a fixed polynomial function in κ. $x \leftarrow \mathbf{Dec}(\mathsf{sk}, c)$ decrypts the ciphertext and returns a plaintext message x.

Correctness is defined in the most obvious manner: for any κ, the following holds with probability 1 for any $x \in \{0,1\}^\ell$: let $\mathsf{sk} \leftarrow \mathbf{Gen}(1^\kappa)$ and $c \leftarrow \mathbf{Enc}(\mathsf{sk}, x)$, then, it must be that $\mathbf{Dec}(\mathsf{sk}, c)$ gives x.

Semantic security requires that as long as the non-uniform p.p.t. adversary does know the honestly generated secret key, it cannot distinguish an honestly generated ciphertext of an arbitrary message x from a simulated ciphertext generated by a simulator that does not know either sk or the plaintext message x. More precisely, there exists a p.p.t. simulator Sim, such that the following two experiments are computationally indistinguishable for any $x \in \{0,1\}^\ell$:

- Real: $\mathsf{sk} \leftarrow \mathbf{Gen}(1^\kappa)$ and $c \leftarrow \mathbf{Enc}(\mathsf{sk}, x)$, output c.
- Ideal: output $\mathsf{Sim}(1^\kappa)$.

C.2 Decisional Linear and Vector Linear Encryption

Recall that the Decisional Linear assumption was defined earlier in Section 4.1. We now describe some straightforward applications of the Decisional Linear assumption, and these are the operational versions we use to prove our scheme.

Vector Decisional Linear assumption. For convenience, we use the Vector Decisional Linear assumption as an operational assumption, which posits that the following two distributions are computationally indistinguishable:

1. Sample $\mathsf{pp} := (q, \mathbb{G}, g) \xleftarrow{\$} \mathcal{G}(1^\kappa)$ where $\mathbb{G}$ is a cyclic group of order q with a random generator $g = [\![1]\!]$. Sample random $\beta, \gamma \xleftarrow{\$} \mathbb{Z}_q$, and random $\mathbf{u}, \mathbf{v} \xleftarrow{\$} \mathbb{Z}_q^n$. Output the tuple $(\mathsf{pp}, [\![1]\!], [\![\beta]\!], [\![\gamma]\!], [\![\mathbf{u}]\!], [\![\beta\mathbf{v}]\!], [\![\gamma(\mathbf{u}+\mathbf{v})]\!])$.
2. Sample $\mathsf{pp} := (q, \mathbb{G}, g) \xleftarrow{\$} \mathcal{G}(1^\kappa)$ where $\mathbb{G}$ is a cyclic group of order q with a random generator $g = [\![1]\!]$. Sample random $\beta, \gamma \xleftarrow{\$} \mathbb{Z}_q$, and random $\mathbf{u}, \mathbf{v}, \mathbf{z} \xleftarrow{\$} \mathbb{Z}_q^n$. Output the tuple $(\mathsf{pp}, [\![1]\!], [\![\beta]\!], [\![\gamma]\!], [\![\mathbf{u}]\!], [\![\beta\mathbf{v}]\!], [\![\mathbf{z}]\!])$.

Fact C.1. *Assume that the Decisional Linear assumption holds in $\mathbb{G}$, then the above Vector Decisional Linear assumption holds in $\mathbb{G}$ as well.*

Proof. The proof can be achieved through a straightforward hybrid argument. Consider a sequence of hybrid experiment indexed by $\{0, 1, \ldots, n\}$. In the i-th hybrid where $i \in [n] \cup \{0\}$, the adversary is given the tuple $([\![1]\!], [\![\beta]\!], [\![\gamma]\!], [\![\mathbf{u} := (u_1, \ldots, u_n)]\!], [\![\beta\mathbf{v} := (\beta v_1, \ldots, \beta v_n)]\!], [\![\mathbf{r} := (r_1, \ldots, r_n)]\!])$ where the first i coordinates of $\mathbf{r}$ are generated at random, and for any $j > i$, $r_j = \gamma(u_j + v_j)$. It suffices to show that any two adjacent hybrids $i^* - 1$ and i^* are computationally distinguishable. Consider a reduction $\mathcal{B}$ that receives a Decisional Linear instance $[\![1]\!], [\![\beta]\!], [\![\gamma]\!], [\![u^*]\!], [\![\beta v^*]\!], [\![z^*]\!]$) and it wants to distinguish whether $[\![z^*]\!]$ is random or $[\![z^*]\!] = [\![\gamma(u^* + v^*)]\!]$. It will interact with the adversary $\mathcal{A}$ who is trying to break the Vector Decisional Linear assumption. $\mathcal{B}$ implicitly chooses $u_{i^*} := u^*$ and $v_{i^*} := v^*$, and all other coordinates of $\mathbf{u}$ and $\mathbf{v}$ are chosen at random. $\mathcal{B}$ also chooses $r_1, \ldots, r_{i^*-1}$ at random. Observe that $\mathcal{B}$ can compute the tuple $([\![1]\!], [\![\beta]\!], [\![\gamma]\!], \mathbf{u}, \beta\mathbf{v}, [\![r_1, r_2, \ldots, r_{i^*-1}, z^*, \gamma(u_{i^*+1} + v_{i^*+1}), \ldots, \gamma(u_{i^*+2} + v_{i^*+2}), \ldots, \gamma(u_n + v_n)]\!])$ and it gives $\mathcal{A}$ this tuple. If $z = \gamma(u^* + v^*)$, $\mathcal{A}$'s view is identical as the $(i^* - 1)$-th hybrid; else $\mathcal{A}$'s view is identical as in the i^*-th hybrid. □

Matrix-DDH assumption (MDDH). Suppose that a group generator $\mathsf{pp} \xleftarrow{\$} \mathcal{G}(1^\kappa)$ samples a cyclic group $\mathbb{G}$ of prime order q, and a generator $[\![1]\!] \in \mathbb{G}$ — henceforth let pp denote the description of the group, including its order q and the sampled generator.

Let $\ell > d$. We say that the (ℓ, d)-MDDH assumption holds for $\mathcal{G}$, iff any non-uniform p.p.t. adversary's views in the following experiments are computationally indistinguishable:

- Sample $\mathsf{pp} \xleftarrow{\$} \mathcal{G}(1^\kappa)$. Next, sample a random matrix $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{\ell \times d}$ and a random vector $\mathbf{w} \xleftarrow{\$} \mathbb{Z}_q^d$, and output $([\![\mathbf{A}]\!], [\![\mathbf{A}\mathbf{w}]\!])$.
- Sample $\mathsf{pp} \xleftarrow{\$} \mathcal{G}(1^\kappa)$. Next, sample a random matrix $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{\ell \times d}$, and a random vector $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_q^\ell$, and $([\![\mathbf{A}]\!], [\![\mathbf{u}]\!])$.

Henceforth, *we also call the $(d+1, d)$-MDDH assumption d-MDDH for short.*

Fact C.2 (Escala et al. [EHK+13])**.** *Suppose that the Decisional Linear assumption holds in $\mathbb{G}$, then the d-MDDH assumption holds in $\mathbb{G}$ for $d = 2$.*

Fact C.3. *Suppose that 2-MDDH holds in $\mathbb{G}$, then for any $\ell > d$ such that ℓ is polynomially bounded in the security parameter, the (ℓ, d)-MDDH assumption holds in $\mathbb{G}$.*

Proof. Suppose that an adversary $\mathcal{A}$ is given $[\![\mathbf{A}]\!], [\![\mathbf{u}]\!]$, and it wants to distinguish whether $[\![\mathbf{u}]\!] = [\![\mathbf{A}\mathbf{w}]\!]$ for some $\mathbf{w}$ or if $\mathbf{u}$ is random. We consider a sequence of hybrid experiments indexed by $\{2, 3, \ldots, \ell\}$. In the i-th hybrid, we compute $\mathbf{u}' := (u'_1, u'_2, \ldots, u'_\ell) = \mathbf{A}\mathbf{w}$, and we give the tuple

$$[\![u'_1, u'_2, r_3, \ldots, r_i, u'_{i+1}, u'_{i+2}, u'_\ell]\!]$$

to $\mathcal{A}$, where $r_3, \ldots, r_i$ are independent random exponents. Therefore, it suffices to prove that any two adjacent hybrids $i^* - 1$ and i^* are computationally indistinguishable.

Suppose that a reduction $\mathcal{B}$ receives a 2-MDDH instance $[\![\mathbf{A}]\!] := (\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}^*)^\top \in \mathbb{G}^{3\times 2}, [\![\mathbf{u} := (u_1, u_2, u^*)]\!] \in \mathbb{G}^3$. For $j \geq i^* + 1$, pick $\mathbf{a}_j := \phi_j \mathbf{a}_1 + \phi'_j \mathbf{a}_2$ where $\phi_j, \phi'_j \xleftarrow{\$} \mathbb{Z}_q$, and let $u_j := \phi_j u_1 + \phi'_j u_2$. For i^*, let $u_{i^*} := u^*$, and let $\mathbf{a}_{i^*} := \mathbf{a}^*$. For $j \in [3, i^* - 1]$, let $u_j \xleftarrow{\$} \mathbb{Z}_q$, and let $\mathbf{a}_j \xleftarrow{\$} \mathbb{Z}_q^2$. Let $\mathbf{A}' := (\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_\ell)^\top$, let $\mathbf{u}' := (u_1, u_2, \ldots, u_\ell)^\top$. Give $\mathcal{A}$ the tuple $(\mathbf{A}', \mathbf{u}')$.

Observe that if $\mathbf{u} = \mathbf{A}\mathbf{w}$, then $\mathcal{A}$'s view is identical to the $(i^* - 1)$-th hybrid; else $\mathcal{A}$'s view is identical to the i^*-th hybrid.

□

Secret-key vector linear encryption. Suppose that a group generator $\mathcal{G}(1^\kappa)$ takes in a security parameter 1^κ and outputs a suitable cyclic group $\mathbb{G}$ of prime order q, and a random generator of the group $g := [\![1]\!] \in \mathbb{G}$. Henceforth, let pp be the description of the group as well as the chosen generator g. A secret-key vector linear encryption scheme for a polynomially bounded message space works as follows:

- $\mathbf{Setup}(\mathsf{pp}, m)$: pick $\mathbf{S} \xleftarrow{\$} \mathbb{Z}_q^{m\times 2}$, let the secret key $\mathsf{sk} := (\mathsf{pp}, \mathbf{S})$.
- $\mathbf{Enc}(\mathsf{sk}, \mathbf{x})$: given the public key pk and the plaintext $\mathbf{x} \in \mathbb{Z}_q^m$, choose a random $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_q^2$, and then compute and output the following ciphertext: $\mathsf{ct} := ([\![\mathbf{r}]\!], [\![\mathbf{x} + \mathbf{S}\mathbf{r}]\!])$.
- $\mathbf{Dec}(\mathsf{sk}, \mathsf{ct})$: parse $\mathsf{sk} := \mathbf{S} \in \mathbb{Z}_q^{m\times 2}$, and parse $\mathsf{ct} := ([\![\mathbf{r}]\!], [\![\mathbf{c}]\!])$ where $\mathbf{c} := (c_1, c_2, \ldots, c_m) \in \mathbb{Z}_q^m$. To decrypt the i-th coordinate for $i \in [m]$, let $\mathbf{s}_i \in \mathbb{Z}_q^{1\times 2}$ be the i-th row of $\mathbf{S}$, compute $[\![c_i - \mathbf{s}_i \cdot \mathbf{r}]\!]$ and output its discrete logarithm.

Correctness of decryption can be verified mechanically.

Lemma C.4 (Security of vector linear encryption)**.** *Assume that the Decisional Linear assumption holds in the group $\mathbb{G}$, then given the above vector linear encryption scheme, any non-uniform p.p.t. adversary $\mathcal{A}$'s views in the following experiments $\mathsf{LE\text{-}Expt}^0(1^\kappa)$ and $\mathsf{LE\text{-}Expt}^1(1^\kappa)$ are computationally indistinguishable.*

$\underline{\mathsf{LE\text{-}Expt}^b(1^\kappa)}$:

- **Setup.** First, run the honest group generation algorithm $\mathcal{G}(1^\kappa)$ to generate a suitable group $\mathbb{G}$ of prime order q; let pp be the description of the group and its generator. Next, a challenger $\mathcal{C}$ calls $\mathsf{sk} := \mathbf{Setup}(\mathsf{pp}, m)$.

  $\mathcal{C}$ gives pp to $\mathcal{A}$, and $\mathcal{A}$ chooses a subset of coordinates $\mathcal{K} \subseteq [m]$ to corrupt, and $\mathcal{C}$ gives $\mathcal{A}$ the secret keys $\{\mathbf{s}_i\}_{i\in\mathcal{K}}$ where $\mathbf{s}_i$ denotes the i-th row of the secret key $\mathbf{S}$.

- **Queries.** In each encryption query, it submits two plaintexts $\mathbf{x}^{(0)} = (x_1^{(0)}, \ldots, x_m^{(0)}) \in \mathbb{Z}_q^m$ and $\mathbf{x}^{(1)} = (x_1^{(1)}, \ldots, x_m^{(1)}) \in \mathbb{Z}_q^m$. It is required that for any $i \in \mathcal{K}$, $x_i^{(0)} = x_i^{(1)}$. Now, $\mathcal{C}$ computes $\mathsf{ct} := \mathbf{Enc}(\mathsf{sk}, \mathbf{x}^{(b)})$ and returns ct to $\mathcal{A}$.

Proof. Let Q be the maximum number of encryption queries made by $\mathcal{A}$. Without loss of generality, we may assume that $Q > 2$. We may consider a sequence of hybrids indexed by $i \in [Q+1]$. In the i-th hybrid, let $\mathcal{H} := [n]\backslash\mathcal{K}$ be the uncorrupted coordinates. For the j-th coordinate in $\mathcal{H}$ where $j < i$, the challenger uses $x_j^{(0)}$ in answering encryption queries. For the j-th coordinate in $\mathcal{H}$ where $j \geq i$, the challenger uses $x_j^{(1)}$ in answering encryption queries. We now show that if there exists a pair of adjacent hybrids i^* and i^*+1 such that a non-uniform p.p.t.$\mathcal{A}$ can distinguish between the two with non-negligible probability, we can construct a non-uniform p.p.t. reduction $\mathcal{B}$ that breaks $(Q, 2)$-MDDH assumption — recall that in Facts C.2 and C.3, we showed that the Decisional Linear assumption implies the $(Q, 2)$-MDDH assumption.

The reduction $\mathcal{B}$ obtains the tuple $([\![\mathbf{A}]\!], [\![\mathbf{u}]\!])$ from the $(Q, 2)$-MDDH challenger, and it wants to tell whether $\mathbf{u} = \mathbf{Aw}$ or $\mathbf{u}$ is random. Henceforth we write $\mathbf{A} := (\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_Q)^\top$, and $\mathbf{u} := (u_1, u_2, \ldots, u_Q)^\top$.

$\mathcal{B}$ guesses at random which coordinate is going to be the i^*-th coordinate, and implicitly lets $\mathbf{s}_{i^*}^\top := \mathbf{w}$ without actually computing it — here we use $\mathbf{s}_{i^*} \in \mathbb{Z}_p^{1\times 2}$ to denote the i^*-th row of the secret key $\mathbf{S}$. For all $i \neq i^*$, it chooses $\mathbf{s}_i$ (i.e., the i-th row of the secret key $\mathbf{S}$) at random such that it knows the exponent. After the adversary chooses $\mathcal{K}$, if it turns out that $\mathcal{B}$'s guess of i^* is wrong, $\mathcal{B}$ simply aborts. Otherwise, it gives the secret keys $\{\mathbf{s}_i\}_{i\in\mathcal{K}}$ to $\mathcal{A}$.

During the j-th encryption query, $\mathcal{B}$ will implicitly let the random coins $\mathbf{r} := \mathbf{a}_j$. Now, $\mathcal{B}$ computes all terms in the ciphertext $([\![\mathbf{r}]\!], [\![c_1, c_2, \ldots, c_m]\!])$ just like in hybrid i^*, except that it replaces $[\![c_{i^*}]\!]$ with the term $[\![u_j]\!]$ from the $(Q, 2)$-MDDH instance. Note that $\mathcal{B}$ can compute all other terms c_i in the ciphertext for $i \neq i^*$ since it knows $\mathbf{s}_i$.

Conditioned on $\mathcal{B}$ not aborting which happens with $1/Q$ probability, $\mathcal{A}$'s view is either identically distributed as the i^*-th hybrid or identically distributed as the (i^*+1)-th hybrid depending on what choice the $(Q, 2)$-MDDH challenger made. □

D Additional Preliminaries for 1-SEL-SIM-Secure FE

D.1 Construction: Single-Input FE for Computing Inner-Product in the Exponent

We present a single-input FE scheme for computing inner-product in the exponent. Our construction is essentially the same as in earlier works [ACF+18, ALS16, Wee16, ABCP16], except that 1) we use a bilinear group; 2) in our construction, **KGen** takes in a vector encoded in the group $\mathbb{G}_1$ rather than in $\mathbb{Z}_q$; and 3) decryption performs pairing operations to compute multiplications in the exponent.

In our construction, the $\mathbf{Gen}(1^\kappa)$ algorithm calls the group generator $\mathcal{G}(1^\kappa)$ and samples a bilinear group[8] $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ of prime order q, along with generators $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$, and the description of a pairing operation $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Henceforth, we will use $[\![v]\!]_1, [\![v]\!]_2, [\![v]\!]_T$ to denote g_1^v, g_2^v, and $e(g_1, g_2)^v$, respectively. Henceforth, we assume that the public parameters pp contains a description of the bilinear group including its order q, the generators g_1, g_2, and a description of the pairing operation e. The remaining algorithms, including **Setup**, **KGen**, **Enc**, and **Dec** are described below where d is a parameter — later we will show that the scheme satisfies our strengthened notion of 1-SEL-SIM security, i.e., Definition 3 assuming that the d-MDDH assumption (see Section C.2) holds in $\mathbb{G}_2$. In the special case when $d = 1$, the d-MDDH assumption degenerates to the DDH assumption; therefore the reader can also assume that $d = 1$.

Single-input FE for computing inner product in the exponent

$\mathbf{Setup}(\mathsf{pp}, m)$:

$\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{(d+1)\times d}$, $\mathbf{W} \xleftarrow{\$} \mathbb{Z}_q^{m\times(d+1)}$

$\mathsf{mpk} := ([\![\mathbf{A}]\!]_2, [\![\mathbf{WA}]\!]_2), \mathsf{msk} := (\mathbf{W}, \mathbf{A})$

return $(\mathsf{mpk}, \mathsf{msk})$

$\mathbf{KGen}(\mathsf{msk}, [\![\mathbf{y}]\!]_1)$:

return $[\![\mathsf{sk}_\mathbf{y}]\!]_1 := [\![\mathbf{W}^\top \mathbf{y}]\!]_1 \in \mathbb{G}_1^{d+1}$

$\mathbf{Enc}(\mathsf{mpk}, [\![\mathbf{x}]\!]_2 \in \mathbb{G}_2^m)$:

$\mathbf{r} \xleftarrow{\$} \mathbb{Z}_q^d$

return $([\![\mathbf{c}]\!]_2, [\![\mathbf{c}']\!]_2) := ([\![\mathbf{Ar}]\!]_2, [\![\mathbf{x} + \mathbf{WAr}]\!]_2)$

$\mathbf{Dec}([\![\mathsf{sk}_\mathbf{y}]\!]_1, [\![\mathbf{y}]\!]_1, ([\![\mathbf{c}]\!]_2, [\![\mathbf{c}']\!]_2))$:

return $[\![\langle \mathbf{y}_1, \mathbf{c}' \rangle - \langle \mathsf{sk}_\mathbf{y}, \mathbf{c} \rangle]\!]_T$

Note that the **Dec** algorithm is actually computed as $e([\![\mathbf{y}]\!]_1, [\![\mathbf{c}']\!]_2)/e([\![\mathsf{sk}_\mathbf{y}]\!]_1, [\![\mathbf{c}]\!]_2)$.

D.2 Proof of Theorem 4.1

To prove Theorem 4.1, it suffices to show that the above FE construction is 1-SEL-SIM-secure by Definition 3, assuming that the d-MDDH assumption holds in the group $\mathbb{G}_2$. For completeness, we present almost the same proof as Abdalla et al. [ACF+18], but now encoding the elements in the appropriate groups.

Consider the following simulator construction; note that this also specifies the Ideal experiment.

[8] In our NIAR construction, we use the symmetric case where $\mathbb{G}_1 = \mathbb{G}_2 = \mathbb{G}$; however, in this section, we present the more general version without assuming $\mathbb{G}_1 = \mathbb{G}_2$.

$\widetilde{\mathbf{Setup}}(\mathsf{pp}, m)$:

$\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{(d+1)\times d}, \widetilde{\mathbf{W}} \xleftarrow{\$} \mathbb{Z}_q^{m\times(d+1)}, \mathbf{u} \xleftarrow{\$} \mathbb{Z}_q^{d+1} \setminus \mathsf{span}(\mathbf{A})$, compute $\mathbf{0} \neq \mathbf{a} \in \mathbb{Z}_q^{d+1}$ s.t. $\mathbf{A}^\top \mathbf{a} = 0$

$\mathsf{mpk} = ([\![\mathbf{A}]\!]_2, [\![\widetilde{\mathbf{W}}\mathbf{A}]\!]_2), \mathsf{msk} = (\widetilde{\mathbf{W}}, \mathbf{A}, \mathbf{a}, \mathbf{u})$

return $(\mathsf{mpk}, \mathsf{msk})$

$\widetilde{\mathbf{KGen}}(\mathsf{msk}, [\![\mathbf{y}]\!]_1, [\![\langle \mathbf{x}, \mathbf{y} \rangle]\!]_1)$:

return $[\![\mathsf{sk}_\mathbf{y}]\!]_1 := [\![\widetilde{\mathbf{W}}^\top \mathbf{y} - \frac{\langle \mathbf{x}, \mathbf{y} \rangle}{\langle \mathbf{u}, \mathbf{a} \rangle} \mathbf{a}]\!]_1 \in \mathbb{G}_1^{d+1}$

$\widetilde{\mathbf{Enc}}(\mathsf{msk})$:

return $([\![\mathbf{c}]\!]_2, [\![\mathbf{c}']\!]_2) := ([\![\mathbf{u}]\!]_2, [\![\widetilde{\mathbf{W}}\mathbf{u}]\!]_2)$

Note that the input $[\![\mathbf{y}]\!]_1, [\![\langle \mathbf{x}, \mathbf{y} \rangle]\!]_1$ to $\widetilde{\mathbf{KGen}}$ are in $\mathbb{G}_1$; but because the $\widetilde{\mathbf{KGen}}$ algorithm knows $\widetilde{\mathbf{W}} \in \mathbb{Z}_q^{m\times(d+1)}$ and $\frac{1}{\langle \mathbf{u}, \mathbf{a} \rangle} \mathbf{a} \in \mathbb{Z}_q^{d+1}$, it can compute $\mathsf{sk}_\mathbf{y}$.

We now show that any non-uniform p.p.t. $\mathcal{A}$'s views in Real and Ideal are computationally indistinguishable. Define a hybrid experiment $\mathsf{Hyb}^{\mathcal{A}}(1^\kappa, m)$ to be the following:

- $\mathsf{pp} := (q, e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2) \xleftarrow{\$} \mathcal{G}(1^\kappa)$,
- Sample the following parameters $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{(d+1)\times d}$, $\mathbf{W} \xleftarrow{\$} \mathbb{Z}_q^{m\times(d+1)}$, $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_q^{d+1} \setminus \mathsf{span}(\mathbf{A})$, $\mathsf{mpk} := ([\![\mathbf{A}]\!]_2, [\![\mathbf{W}\mathbf{A}]\!]_2)$, $\mathsf{msk} := (\mathbf{W}, \mathbf{A}, \mathbf{u})$, and give mpk to $\mathcal{A}$, who outputs $\mathbf{x} \in \mathbb{Z}_q^m$.
- Compute $\mathsf{ct} := ([\![\mathbf{c}]\!]_2, [\![\mathbf{c}']\!]_2) = ([\![\mathbf{u}]\!]_2, [\![\mathbf{x} + \mathbf{W}\mathbf{u}]\!]_2)$ and return ct to $\mathcal{A}$.
- $\mathcal{A}$ is allowed to make queries to $\mathbf{KGen}(\mathsf{msk}, \cdot)$, and it finally outputs a bit b.

$\mathsf{Hyb}^{\mathcal{A}}(1^\kappa, m)$ is the same as $\mathsf{Real}(1^\kappa, m)$ except that $\mathbf{Ar}$ in the ciphertext is replaced with $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_q^{d+1} \setminus \mathsf{span}(\mathbf{A})$.

Lemma D.1. *Suppose that the d-MDDH assumption holds in $\mathbb{G}_2$. Then, any non-uniform p.p.t. adversary $\mathcal{A}$'s views in $\mathsf{Real}(1^\kappa, m)$ and $\mathsf{Hyb}(1^\kappa, m)$ are computationally indistinguishable.*

Proof. First, imagine an experiment Hyb' which is the same as Hyb except that $\mathbf{u}$ is chosen at random from $\mathbb{Z}_q^{d+1}$ instead. In this case, due to the d-MDDH assumption, no non-uniform p.p.t. adversary can distinguish $([\![\mathbf{A}]\!]_2, [\![\mathbf{A}\mathbf{r}]\!]_2)$ from $([\![\mathbf{A}]\!]_2, [\![\mathbf{u}]\!]_2)$ with non-negligible probability. Therefore, the computational indistinguishability of Real and Hyb' follows from a straightforward reduction to the d-MDDH assumption.

Finally, observe that the uniform distribution over $\mathbb{Z}_q^{d+1}$ and $\mathbb{Z}_q^{d+1} \setminus \mathsf{span}(\mathbf{A})$ have statistical distance at most $1/q$ which is a negligible function in κ. Therefore, $\mathcal{A}$'s views in Hyb and Hyb' have negligible statistical distance. The lemma now follows through a hybrid argument. □

To complete the proof of the 1-SEL-SIM security of the FE scheme, it suffices to prove the following lemma.

Lemma D.2. *For any adversary $\mathcal{A}$, its views in $\mathsf{Ideal}(1^\kappa, m)$ and $\mathsf{Hyb}(1^\kappa, m)$ are identically distributed.*

Proof. The lemma can be observed by performing a variable substitution by letting

$$\mathbf{W} := \widetilde{\mathbf{W}} - \frac{1}{\langle \mathbf{u}, \mathbf{a} \rangle} \mathbf{x}\mathbf{a}^\top$$

where $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_q^{d+1} \setminus \mathsf{span}(\mathbf{A}), \mathbf{0} \neq \mathbf{a} \in \mathbb{Z}_q^{d+1}$ s.t. $\mathbf{A}^\top \mathbf{a} = 0$. $\mathbf{W}$ can be computed because $\mathbf{u}$ is not in the span of A, indicating that $\langle \mathbf{u}, \mathbf{a} \rangle \neq 0$. Observe that

$$\begin{aligned} \mathbf{W}\mathbf{A} &= \widetilde{\mathbf{W}}\mathbf{A} \\ \mathbf{x} + \mathbf{W}\mathbf{u} &= \widetilde{\mathbf{W}}\mathbf{u} \\ \mathbf{W}^\top \mathbf{y} &= \widetilde{\mathbf{W}}^\top \mathbf{y} - \frac{\langle \mathbf{x}, \mathbf{y} \rangle}{\langle \mathbf{u}, \mathbf{a} \rangle} \mathbf{a} \end{aligned}$$

Therefore, the two experiments Ideal and Hyb are identically distributed. □

D.3 Alternative Operational Security Definition

We give an n-concurrent version of the security definition which is easier to use in our proofs. Consider an n-concurrent version of the security experiments in which the adversary $\mathcal{A}$ is allowed to invoke n instances of the FE scheme. Specifically, let $\mathsf{FE\text{-}Real}[n](1^\kappa, m)$ and $\mathsf{FE\text{-}Ideal}[n](1^\kappa, m)$ denote the n-current versions of the real-world and ideal-world experiments, as defined below:

Experiment $\mathsf{FE\text{-}Real}[n](1^\kappa, m)$:	**Experiment** $\mathsf{FE\text{-}Ideal}[n](1^\kappa, m)$:
$\mathsf{pp} := (q, e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2) \xleftarrow{\$} \mathcal{G}(1^\kappa)$	$\mathsf{pp} := (q, e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2) \xleftarrow{\$} \mathcal{G}(1^\kappa)$
$\forall i \in [n] : (\mathsf{mpk}_i, \mathsf{msk}_i) \leftarrow \mathbf{Setup}(\mathsf{pp}, m)$	$\forall i \in [n] : (\mathsf{mpk}_i, \mathsf{msk}_i) \leftarrow \widetilde{\mathbf{Setup}}(\mathsf{pp}, m)$
$\{\mathbf{x}_i\}_{i \in [n]} \leftarrow \mathcal{A}(1^\kappa, \{\mathsf{mpk}_i\}_{i \in [n]})$ where $\mathbf{x}_i \in \mathbb{Z}_q^m$	$\{\mathbf{x}_i\}_{i \in [n]} \leftarrow \mathcal{A}(1^\kappa, \{\mathsf{mpk}_i\}_{i \in [n]})$ where $\mathbf{x}_i \in \mathbb{Z}_q^m$
$\forall i \in [n] : \mathsf{ct}_i \leftarrow \mathbf{Enc}(\mathsf{mpk}_i, \mathbf{x}_i)$	$\forall i \in [n] : \mathsf{ct}_i \leftarrow \widetilde{\mathbf{Enc}}(\mathsf{msk}_i)$
$b \leftarrow \mathcal{A}^{\{\mathbf{KGen}(\mathsf{msk}_i, \cdot)\}_{i \in [n]}}(\{\mathsf{ct}_i\}_{i \in [n]})$	$b \leftarrow \mathcal{A}^{\{\mathcal{O}_i(\cdot)\}_{i \in [n]}}(\{\mathsf{ct}_i\}_{i \in [n]})$
Output b	Output b

For $i \in [n]$, the oracle $\mathcal{O}_i(\cdot)$ in the above $\mathsf{FE\text{-}Ideal}$ experiment is defined as below: upon receiving a **KGen** query with $[\mathbf{y}]_1 \in \mathbb{G}_1^m$ return $\widetilde{\mathbf{KGen}}(\mathsf{msk}_i, [\mathbf{y}]_1, [\langle \mathbf{x}_i, \mathbf{y} \rangle]_1)$ to $\mathcal{A}$.

Lemma D.3. *Let n be bounded by a fixed polynomial in the security parameter κ. If FE is 1-SEL-SIM-secure, then any non-uniform p.p.t. $\mathcal{A}$'s views in $\mathsf{FE\text{-}Real}[n](1^\kappa, m)$ and $\mathsf{FE\text{-}Ideal}[n](1^\kappa, m)$ are computationally indistinguishable.*

Proof. The proof can be accomplished through a standard hybrid argument. Starting from $\mathsf{FE\text{-}Real}[n](1^\kappa, m)$ in which all instances of FE are instantiated with the real scheme, one by one, we shall replace each instance of FE that $\mathcal{A}$ is interacting with with a simulated instance. Due to the 1-SEL-SIM-security of FE, $\mathcal{A}$'s views in each adjacent pair of hybrids are computationally indistinguishable. □