University of California
Santa Barbara

Non-Interactive Private Set Intersection From Functional Encryption

A dissertation submitted in partial satisfaction
of the requirements for the degree

Master of Science
in
Computer Science

by

Saikumar Yadugiri

Committee in charge:

Professor Prabhanjan Ananth, Chair
Professor Amr El Abbadi
Professor Trinabh Gupta

June 2023

The Dissertation of Saikumar Yadugiri is approved.

Professor Amr El Abbadi

Professor Trinabh Gupta

Professor Prabhanjan Ananth, Committee Chair

May 2023

Non-Interactive Private Set Intersection From Functional Encryption

by

Saikumar Yadugiri

Dedicated to my Grandparents.

Acknowledgements

As my incredible journey filled with academic and personal enrichment comes to an end here at University of California Santa Barbara, I would like to thank a multitude of people who have helped me throughout this journey. I sincerely apologize if I left out anyone from this list.

First and foremost, I would like to thank Prof. Prabhanjan Ananth for being a wonderful mentor, an excellent professor and a caring guide throughout my graduate studies. He has always entertained my (often faulty) ideas, directed me towards the right path, and provided me a stage to explore various research ideas. I would like to particularly thank Achintya Desai for his insightful comments, valuable mentoring, and stress-reducing card games. Work presented in this dissertation is a joint collaboration with Prof. Prabhanjan Ananth and Achintya Desai.

Being my first tryst with multiple-disciplines in computer science, I like to express my gratitude towards all the professors of UC Santa Barbara who have expertly crafted their rich and comprehensive academic material. A special thank you to the graduate committee of the computer science department for providing me a fantastic platform to advance my academic career.

I would also like to thank Prof. Trinabh Gupta and Prof. Amr El Abbadi for agreeing to be a part of my thesis committee and reviewing my progress in my graduate students. Most significantly, I wish to acknowledge my family whose continuous and unconditional support gave me the strength to continue my graduate journey. Especially my uncles who have helped a lot to start this wonderful journey. I would like to dedicate this dissertation to my grandfather and my grandmother to whom I owe my life.

Abstract

Non-Interactive Private Set Intersection From Functional Encryption

by

Saikumar Yadugiri

In a traditional public-key encryption scheme, users who possess the secret key learn the entire message from the ciphertext whereas other users do not learn anything. With the advent of cloud computing and the increasing demand for privacy-preserving technologies a more sophisticated tool that provides fine-grained access to data is required. Functional encryption is one such tool which reveals the value of a function f acted upon x i.e, $f(x)$ for any user in the possession of the ciphertext corresponding to x and a secret key sk_f associated with f.

Private Set Intersection is a cryptographic scheme that helps two parties, Alice and Bob, to find the intersection of their input sets A and B while revealing nothing more than A $\cap$ B to each other. Efficient and practical implementations of this scheme are already in use in the industry in intra- and inter-organizational settings serving as a solution to many problems ranging from private digital marketing to international data privacy laws. However, most solutions are interactive and require both parties to be online and repeat the entire procedure if any changes to either party's sets are to be considered.

Eliminating this interactive nature of the problem is the key to overcoming inevitable obstacles. However, the non-interactive version of this problem presents many obstacles that require efficient utilization techniques from a multitude of cryptographic domains. We pose the problem as a version of an message-hiding functional encryption scheme in which the ciphertexts and the secret keys are associated with the two parties' inputs.

Although several functional encryption schemes are present for general functions in

various computational models, only a handful functional encryption schemes for specialized functions are present. The main contributions are two fold:

1. We construct an semi-adaptively secure collusion-resistant private-key functional encryption scheme for set intersection. The encryption time and secret key generation time grow linearly with the maximum set size. Moreover, the sizes of ciphertexts and secret-keys also grow linearly in the maximum set size.

2. We construct an adaptively secure bounded-collusion public-key functional encryption scheme for set intersection. The encryption time, secret key generation time, ciphertext size, and secret-key size grow linearly with the query bound Q.

We also present possible avenues to pursue for further improvements to enhance the security and efficiency of the aforementioned schemes.

Contents

Chapter 1

Introduction

Functional encryption [SW05, O'N10, BSW11] is one of the foundational concepts in the paradigm of computing on encrypted data. Functional encryption has also been used as a useful abstraction in several theoretical and practical avenues including, but not limited to, the construction of indistinguishability obfuscation [AJ15, BV18], succinct randomized encoding [AL18, GS18, AM18], watermarking schemes [GKM$^+$19], deep learning using semi-encrypted neural networks [RPB$^+$19], and proving lower bounds in differential privacy [KMUW18]. Functional encryption offers the excellent advantage of non-interactivity over other cryptographic constructs such as secure computation which typically entail interactive solutions.

Roughly speaking, a functional encryption scheme allows for the generation of special-purpose functional secret keys associated with the functions $f_1, \ldots, f_Q$ such that anyone, using these keys, can decrypt an encryption of x to recover the outputs $f_1(x), \ldots, f_Q(x)$, and nothing else. Although feasibility results for functional encryption for general functions exist [GVW12, AR17, CVW$^+$18, AV19], these results possess large asymptotic complexity making their usage for interesting and specialized functions impractical. A handful of avenues in specialized functions such as inner-product [ABCP15, BJK15, ALMT20,

ACF+18, AGRW17], quadratic functions [AGT22, AGT21, APS21], and predicated functions [AGT21] have been studied for functional encryption. However, several interesting and practical functions such as set intersection or functions utilized for machine learning such as the mean square function or sigmoid function have not been explored very much in the field of cryptography. Efficient functional encryption schemes for these functionalities pave the way for the utilization of privacy-focused computation in a multitude of domains.

Set Intersection. In this paper, we focus on the private set intersection problem where two parties, each possessing one set wishes to compute the intersection of the sets without revealing any other entries that are not included in the intersection. For instance, Alice holds set S and Bob holds set T. Both Alice and Bob wish to learn $S \cap T$. However, Alice wants to hide the elements in $S \setminus S \cap T$ from Bob and vice-versa. Private set intersection is known to be used in many practical scenarios such as mitigating privacy and legal concerns in mobile private contact systems [KRS+19]. Consider the following example—a large cybersecurity firm releases a set of compromised authentication details it found in several attacks, dark web, etc. A large bank wishes to learn if any of its users are part of this set and if so wants to warn them to change their login details immediately. In this case, neither the bank nor the cybersecurity firm wants to reveal their data but both of them want to learn the intersection of their sets. Similarly, two multi-national conglomerates wish to find the users who use both of their services indicating a mutually benificial data collection to both of them. Due to several data regulations present, neither of the parties can divulge their dataset but can make use of privacy preserving techniques. The private set intersection is one such technique that can find the common users without leaking the identity of the users to the conglomerate whose service they do not use.

Private set intersection has been widely studied with different settings in the ap-

plied cryptography domain with main approach being secure computation protocols [TYG22, CGH+21, LRSZ21]. In the celebrated series of works, secure computation protocols including oblivious transfer [TYG22, CGH+21, GPR+21, PRTY20, CO18], hashing schemes [CDG+21, LRSZ21, PRTY19, PSTY19, PSWW18, RR17, DCW13, PSZ14], oblivious pseudo-random functions [CM20, KRTW19, KLS+17, FIPR05], etc, have been used to develop several interactive private set intersection protocols. Furthermore, few of these protocols even achieve linear communication complexity [TYG22, CGS21, JL09, JL10, PSTY19] and scale excellently in the size of the input sets. However, these are interactive protocols that require the servers of all the parties involved in the computation to be online. This way, all the parties need to bear the communication, computation, and server operational costs. Moreover, if there is a change to the sets of one of these parties, the protocol needs to be executed from the beginning which is unconducive to the other party.

Functional Encryption for Set Intersection. We construct a functional encryption scheme that is non-interactive, linear in the query bound and low-degree polynomial in the size of the input sets. To the best of our knowledge, this is the first functional encryption scheme focused on set intersection functionality. In a functional encryption scheme for set intersection, Alice encrypts the set S and queries the functional key generation oracle for a functional key for the set T. When the decryption algorithm is executed using the functional key associated with set T and the ciphertext associated with set S, it outputs $S \cap T$ and nothing else.

Our Results. We provide both a private-key and public-key functional encryption schemes for set intersection which work under different assumptions. In terms of the security of our schemes, our private-key scheme is semi-adaptive simulation secure and public-key scheme is adaptive-simulation secure and reveal nothing more than the inter-

section of the sets S, T i.e, $S \cap T$. For public-key functional encryption schem, this is the highest level of security notion that can be satisfied. To be more precise, our private-key functional encryption scheme is collusion-resistant under the assumption that the one-way function exists. That is, for any polynomially-bounded adversary querying polynomially many functional key queries for sets $T_1, \ldots, T_Q$ and ciphertexts for sets $S_1, \ldots, S_M$, we divulge only the information $\{S_1 \cap T_1, \ldots, S_1 \cap T_Q\}, \ldots, \{S_M \cap T_1, \ldots, S_M \cap T_Q\}$. Our public-key functional encryption scheme is bounded-collusion secure and assumes the existence of public-key encryption schemes and the computational hardness of Decisional Diffie-Hellman(DDH) problem. That is, for a prior given query bound Q, for any polynomially-bounded adversary querying polynomially many functional key queries for sets $T_1, \ldots, T_Q$ and ciphertexts for set S, we divulge only the information $\{S \cap T_1, \ldots, S \cap T_Q\}$.

Motivation. Using a non-interactive functional encryption scheme, one of the parties can encrypt and publish their encrypted sets to the world through a website and remain offline for further computation. From the security of functional encryption, this ciphertext will be indistinguishable to any polynomially-bounded adversary. Any party who wishes to compute the intersection of their set with the publisher's set can acquire the functional key associated with its set and run the decryption algorithm to learn the intersection and nothing else. If there are any changes to the set, the party can acquire a fresh functional key and proceed with its computation all the while the publisher remains in an offline state. Any changes to the publisher's set can also be published without the involvement of the other party.

Furthermore, if any other party wishes to join the network, it need not require any communication with the publisher. To understand this further, consider the example of the cybersecurity firm which releases compromised authentication details. Everytime the firm finds new entries, it can add them to its set and publish a fresh ciphertext. Any

bank that wants to find the details of compromised to users, can acquire the functional key associated with its user dataset and run the decryption algorithm to find the list of its compromised users. This computation doesn't reveal any information about the compromised users to the cybersecurity firm as only bank possesses the output of the decryption algorithm. Moreover, if any other bank wants to do the same, all it has to do is to acquire a functional key for its user dataset all the while the cybersecurity firm's servers remain offline and oblivious.

The efficiency of our private-key funcional encryption scheme is linear in the sizes of input sets and the running time of the choice of PRF. The asymptotic efficiency of our public-key functional encryption scheme's encryption time and ciphertext size is linear in the query bound Q. In comparison with other public-key functional encryption schemes for general circuits, our construction is low-degree polynomial in security parameter and the sizes of input sets. For instance, consider the [AV19] construction which also provides a functional encryption whose ciphertext size grows linear in the query bound. They use a load balancing technique which transfers the exponent of the query bound to the security parameter. Prior to this, the asymptotic complexity of the scheme is Q^9 and λ^8 assuming the universal circuit is of size $\Theta(s \log(s))$ [ZYZL19]. This means the load balancing doubles the exponent for the security parameter. In our public-key functional encryption scheme, prior to the load balancing, we mainitain the size of the ciphertext and the encryption time to be quadratic in the query bound which yields a scheme that is low-order polynomial in the security parameter and the sizes of the input sets.

1.1 Permissions and Attributions

The contents of this dissertation are the result of a collaboration with Prof. Prabhanjan Ananth and Achintya Desai. It is reproduced with permission from both parties.

Chapter 2

Technical Overview

In this section, we will give an overview of our techniques used to compute functional encryption for set intersection problem. For private-key we discuss the nuances for constructing private-key functional encyption for set intersection assuming the existence of one-way functions. For public-key functional encryption, we discuss the observations that lead us to an efficient version of the functional encryption for set intersection starting from one-key functional encryption for P/Poly.

2.1 Private-Key FE for SI

Under the assumption of existence of one-way functions, we construct the private-key functional encryption for set intersection based on pseudo-random functions. In the private-key setting, master secret key is used in the encryption as well as the secret key generation algorithm which provides an edge over it's public-key counterpart. We can use the master secret key and the set element as input to the pseudo-random function and compute an identical input-hiding value in the encryption and the functional-key generation algorithm. This makes our private-key functional encryption scheme efficient

and collusion-resistant. However, constructing a scheme based on this idea is non-trivial. Unlike public-key functional encryption, we need to explicitly prove the security when adversary queries multiple sets in the encryption phase. Firstly, the obvious construction, that uses the master secret key as a key for the pseudo-random function, is correct but not secure. It is easy to see why this construction is not secure as the encryption is deterministic due to the determinism of pseudo-random functions which violates CPA security.

To randomize the encryption, we use an additional pseudo-random function that takes the output of the first pseudo-random function as a key and a random string as input to output a pseudo-random value. This ensures that the pseudo-random value is not the same for two distinct queries irrespective of the set element. We modify the rest of the construction to hold the correctness. A brief summary of this construction is as follows:

$\underline{\mathsf{Setup}(1^\lambda)}$: Output symmetric-key MSK.

$\underline{\mathsf{KeyGen}(\mathsf{MSK}, T = \{t_1, \ldots, t_\mu\})}$: Output $\mathsf{sk}_T = \Big(\Big(t_1, \mathsf{PRF}_1(\mathsf{MSK}, t_1)\Big), \ldots, \Big(t_\mu, \mathsf{PRF}_1(\mathsf{MSK}, t_\mu)\Big)\Big)$.

$\underline{\mathsf{Enc}(\mathsf{MSK}, S = \{s_1, \ldots, s_\gamma\})}$:

- Sample $\forall i \in [\gamma], r_i \xleftarrow{\$} \{0,1\}^p$.
- Compute $\forall i \in [\gamma], k_i \leftarrow \mathsf{PRF}_1(\mathsf{MSK}, s_i)$.
- Compute $\forall i \in [\gamma], ct_i \leftarrow \mathsf{PRF}_2(k_i, r_i)$.
- Output CT as $\Big(\Big(r_1, ct_1\Big), \ldots, \Big(r_\gamma, ct_\gamma\Big)\Big)$.

$\underline{\mathsf{Dec}(\mathsf{sk}_T, \mathsf{CT})}$: Output the set

$P = \{t_j | (t_j, x_j) \in \mathsf{sk}_T \ \wedge \exists i \in [\gamma] : (r_i, e_i) \in \mathsf{CT} \wedge \ e_i = \mathsf{PRF}_2(x_j, r_i)\}$.

The correctness of the above construction holds trivially. However, the construction is only selectively-secure. In the definition of adaptive security, there are two phases where the adversary queries the challenger for secret keys for sets. In the simulated experiment, the simulator can respond to the adversary's initial secret key query phase and the challenge ciphertext phase. Specifically, for the sets which are common in the initial secret key query phase and the challenge ciphertext phase, simulator replaces the output of $\mathsf{PRF}_1(\mathsf{MSK}, .)$ and the sets queried only in the challenge ciphertext phase, the output of $\mathsf{PRF}_2(., .)$ with uniform random strings. Now, during the second secret key query phase if the adversary queries a set element which is in common with the challenge ciphertext phase, then the adversary can distinguish between simulator and an honest challenge. In this case, the simulator needs to abort. As we replace the output of the ciphertext query phase to a uniformly random string, the $\mathsf{PRF}_1(\mathsf{msk}, .)$ computation cannot be reversed. This makes the construction selectively secure which is not desired. To overcome this, we notice that the adversary may not even query the same set element in both the challenge ciphertext phase and the second functional query phase. Hence, we can make the simulator flip a fair coin during the challenge ciphertext phase to decide between $\mathsf{PRF}_1(\mathsf{MSK}, .)$ and $\mathsf{PRF}_2(., .)$ to replace with uniform random string. However, the adversary can still distinguish between simulator and an honest challenge with non-negligible probability.

This leads us to our final construction. Here, we inherit the idea of two PRFs from the previous construction. However, we instantiate PRF_1 twice with an old and a new key. During the secret key generation algorithm, we output the PRF_1 values for the set elements with both the keys. In the encryption algorithm, we additionally perform an exclusive-or of the ciphertext from previous construction with the output of PRF_1 instantiated with new key. This pushes our construction to semi-adaptive security. This is because the simulator can now simply replace the ciphertext with a uniform random

string. If the same set element is queried during the second functional query phase, then, the simulator samples another uniform random string to replace the PRF_1 instantiated with the old key and computes the rest of the values such that it preserves the relationship between ciphertext and the two instantiations of PRF_1. However, if the adversary queries a set element in one of the challenge query phases which has not been queried as part of any of the previous functional query phases and queries the same element again in the next functional query phase, we cannot answer this using our construction. This is because the adversary only receives the intersection information in the challenge query phase which masks such a set element and hence, we can't reproduce the same randomness. The correctness and security analysis of this construction is present in chapter 6.

2.2 Public-Key FE for SI

The starting point for our public-key functional encryption for set intersection is the public-key adaptively-secure bounded-key functional encryption scheme for P/Poly general circuits([AV19]). This construction assumes the existence of public-key adaptively-secure single-key functional encryption scheme for P/Poly general circuits [GVW12] and a novel correlated garbling scheme for universal circuit. This is the first optimal construction in terms of functionality, assumption, efficiency, and ciphertext size in bounded-collusion functional encryption.

For our problem, it is sufficient that the circuit takes a PRF key as input and outputs PRF values for all the elements in the functional query set. Due to the simplistic nature of the circuit, using bounded-collusion functional encryption for P/Poly is expensive and we can do better in terms of efficiency. The main bottleneck here is the correlated garbling scheme although which is necessary for generalized circuits, presents a huge computational constraint for our goal. Using this scheme, the theoretical asymptotic

complexity for encryption time and ciphertext size is in the order of Q^9 where Q is the priori query bound. In our first attempt to overcome this, we replaced correlated garbling with Shamir Secret Sharing. In our scheme, we instantiate N single-key adaptively-secure functional encryption schemes from which n are chosen in the functional-key generation algorithm. N and n are chosen in accordance with the small-pairwise intersection lemma 10.

Since n out of N shares are chosen uniformly at random, we need N shares of the $N \cdot \gamma$ number of $\mathsf{PRF}(.,.)$ values for the challenge set. The overview of this construction is as follows:

$\underline{\mathsf{Setup}(1^\lambda, 1^Q, 1^{L_{max}})}$: Generate $\forall i \in [N]$, $(\mathsf{pk}_i, \mathsf{msk}_i) \leftarrow \mathsf{1FE.Setup}(1^\lambda)$.
Output $\mathsf{MSK} = (\mathsf{msk}_1, \ldots, \mathsf{msk}_N)$, $\mathsf{PK} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_N)$.
$\underline{\mathsf{KeyGen}(\mathsf{MSK}, T = \{t_1, \ldots, t_\mu\})}$:

- Sample a random subset $\Delta_T \subset [N]$ of size n.
- Set $\hat{T} \leftarrow ((1, t_1), \ldots, (\mu, t_\mu))$.
- Construct the function circuit c that takes $(K, \{x_{i,1}, \ldots, x_{i,\gamma}\}_{i \in [N]})$ and outputs $\left((\mathsf{PRF}(K, t_1), \ldots, \mathsf{PRF}(K, t_\mu)), \left(\bigoplus_{i \in \Delta_T} x_{i,1}, \ldots, \bigoplus_{i \in \Delta_T} x_{i,\gamma}\right)\right)$.
- $\forall u \in \Delta_T$, compute $\mathsf{sk}_u \leftarrow \mathsf{1FE.KeyGen}(\mathsf{msk}_u, c)$.
- Output $\mathsf{sk}_T = (\hat{T}, \Delta_T, \{\mathsf{sk}_u\}_{u \in \Delta_T})$.

$\underline{\mathsf{Enc}(\mathsf{PK}, S = \{s_1, \ldots, s_\gamma\})}$:

- $\forall u \in [N]$, sample $K_u \xleftarrow{\$} \{0,1\}^\lambda$.
- $\forall u \in [N], i \in [\gamma]$, compute $\mathsf{PRF}(K_u, s_i)$ and sample a degree-τ polynomial $\gamma_{u,i}(.)$ over a binary extension finite field of size m such that $\gamma_{u,i}(0) = \mathsf{PRF}(K_u, s_i)$.

- For $u \in [N], i \in [\gamma]$, let $SH_k = \{(\gamma_{u,i}(k), \ldots, \gamma_{u,i}(k))\}$.

- $\forall u \in [N]$, $ct_u = \mathsf{1FE.Enc}(\mathsf{pk}_u, (K_u, SH_u))$.

- Output $\mathsf{CT} = \{ct_u\}_{u \in [N]}$.

$\underline{\mathsf{Dec}(\mathsf{sk}_T, \mathsf{CT})}$:

- $\forall d \in \Delta_T$, $((w_{d,1}, \ldots, w_{d,\mu}), (x_{d,1}, \ldots, x_{d,\gamma})) \leftarrow \mathsf{1FE.Dec}(\mathsf{sk}_d, ct_d)$.

- $\forall j \in [\mu]$, Compute $\hat{P}_j = \bigoplus_{d \in \Delta_T} w_{d,j}$.

- $\forall\ i\ \in [\gamma]$, similarly compute P'_k using $\{x_{d,i}\}_{d \in \Delta_T}$ as secret shares from Shamir's Secret Sharing Scheme.

- Construct the set $\bar{P} = \left\{ j | \exists i \in [\gamma], \hat{P}_j = P'_i \right\}$.

- Output the set $P = \left\{ t_j | j \in \bar{P}, (j, t_j) \in \hat{T} \right\}$.

The correctness of this scheme follows intuitively. Like [AV19], this scheme is also bounded-collusion adaptively-simulation secure. However, as we generate $N^2 \cdot \gamma$ shares during the encryption algorithm, the theoretical asymptotic efficiency is in the order of Q^4 which is better but still impractical. Improvements involving the secret sharing scheme such as packed secret sharing [FY92] also do not improve the efficiency asymptotically. Now the bottleneck is secret sharing scheme. In order to eliminate secret sharing scheme, we used public-key encryption scheme to encrypt the $N \cdot \gamma$ pseudo-random function values. In the circuit c, before computing the exclusive-OR of $\mathsf{PRF}(.,.)$ values, we decrypt them. The advantage here is that the encryption protects the $\mathsf{PRF}(.,.)$ from being leaked to the adversary and, the correctness of the public-key encryption scheme ensures that the circuit evaluates to the correct values of the $\mathsf{PRF}(.,.)$. This scheme looks to be secure, however, a keen-eyed reader might have figured out that the lack of function-hiding property makes the scheme insecure.

A major observation in this scheme is that we do not require the $\mathsf{PRF}(.,.)$ values readily. We only need to be able to compare the $\mathsf{PRF}(.,.)$ values corresponding to the set elements. This takes us to our most efficient construction with the theoretical asymptotic efficiency in the order of Q^2, which is presented in chapter 7. In this final construction, we use an inner product functional encryption scheme that outputs the inner product between two vectors in the exponent of a group element. Therefore, if we construct a vector containing all the $\mathsf{PRF}(.,,)$ values of a set element associated with different keys $K_1, \ldots, K_N$ and a binary vector. We can obtain the sum of n out of N $\mathsf{PRF}(.,.)$ values in the exponent of a group generator. Since the $\mathsf{PRF}(.,.)$ values of the set associated with functional-key query are readily available in the functional key, we can always perform a summation and raise it to the same group generator. Comparing these values in a succinct fashion will yield the set intersection.

There are a few caveats to be considered for this scheme:

1. The inner product functional encryption limits the number of queries to the functional-key generation algorithm with the length of the vector.

2. If the exponentiation operation cycles the group, there is a possibility that it might produce a false positive which will yield an incorrect set intersection.

3. For our scheme to be adaptively-simulation secure, this inner product functional encryption needs to be adaptively-simulation secure.

We handle these caveats by choosing the appropriate inner-product functional encryption scheme and setting our parameters in the scheme appropriately in accordance with small-pairwise intersection lemma. The correctness and security analysis of this construction is detailed in the later chapter 7.

Chapter 3

Related Work

Functional encryption for general purpose functions [GVW12, AR17, CVW+18, AV19, AS16, AM18, AMVY21, AKM+22, ACFQ22] have been well studied in the past decades. This exciting line of research also produced several connections between functional encryption and indistinguishability obfuscation [AJ15, BV18]. A majority of the work is done in the circuit model of computation with constructions in NC^1 [GVW12], P/Poly [GVW12, AR17, CVW+18, AV19] circuit classes. The efficiency of these constructions have also gradually improved from quartic orders [GVW12] in the priori query bound to linear [AV19] with assumptions ranging from Learning with Errors(LWE) [GVW12, AR17, CVW+18] to minimal cryptographic assumptions [AV19]. Other models of computation such as turing machines have also seen constructions with varying security definitions in the works of [AS16, AM18, AMVY21, AKM+22]. Collusion-resistant functional encryption for the RAM computation model was recently constructed in [ACFQ22] building on top of the single-key single-input functional encryption of [GHRW14].

On the other hand, private set intersection has seen major strides in the development of special-purpose protocols. Many early protocols use Oblivious Polynomial Evaluation similar to Shamir's Secret Sharing schemes [CDSJ16, HN10, DSMRY09, KS05, FNP04]

which yield high computational costs but low communication costs. PSI protocols using Oblivious Transfer(OT) [TYG22, CGH+21, GPR+21, PRTY20, CO18] and primitives based on OT such as OPRF functions [TYG22, CGS21, CM20, HFNO19, KRTW19, PSTY19, KLS+17, DCT12, CT10, JL10, JL09, FIPR05] and typically exhibit linear communication complexity. Protocols that use hashing schemes including cuckoo hashing [CDG+21, PRTY19, PSTY19, PSWW18, PSZ18, PSSZ15, PSZ14] and bloom filters [LRSZ21, RR17, DCW13, MBD12] have also yielded asymptotically linear constructions for PSI. Other flavors of protocols utilize strong cryptographic tools such as homomorphic encryption [JWP22, CTX20, GS19, CHLR18, CLR17], pairing-based public-key cryptosystems [RA18], etc.

Functional encryption for specialized functions such as inner products [ABCP15, BJK15, DDM16, ALS16, TAO16, ALMT20, TT20, ABG19, ACF+18, DOT18, AGRW17] has also been studied extensively. Constructions in the single-input setting [ABCP15, BJK15, DDM16, ALS16, TAO16, ALMT20, TT20] and the multi-input setting [ABG19, ACF+18, DOT18, AGRW17] under various assumptions and security levels have been developed in the past decade. Constructions for variations such as mixed-group inner product functional encryption and predicated inner product functional encryption [AGT21] have also been developed. Functional encryption for quadratic functions in the multi-input and multi-client settings are seen in [AGT22, AGT21, APS21].

Chapter 4

Preliminaries

We denote the security parameter by λ. The universe set, which is the set of all possible values for the set elements is denoted by U. For practical purposes, we can consider $U = \{0,1\}^{poly(\lambda)}$. We denote the set of fist n positive integers by $[n]$, i.e, $[n] = \{1, \ldots, n\}$. If D is a probability distribution that has an efficient sampler, $x \xleftarrow{\$} D$ denotes the process of sampling some x uniformly at random. The statistical distance between two probability distributions D_0 and D_1 with support V is ϵ if:

$$\sum_{x \in V} \left| \Pr\left[x \xleftarrow{\$} D_0\right] - \Pr\left[x \xleftarrow{\$} D_1\right] \right| \leq 2\epsilon$$

Two distributions D_0 and D_1 are said to be computationally indistinguishable if for every probabilistic polynomial time (PPT) adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that:

$$\left| \Pr_{x \xleftarrow{\$} D_0} [0 \leftarrow \mathcal{A}(x)] - \Pr_{x \xleftarrow{\$} D_1} [0 \leftarrow \mathcal{A}(x)] \right| \leq \mathsf{negl}(\lambda)$$

Chapter 5

Definitions

Our private-key functional encryption scheme for set intersection consists of these algorithms:

- $\mathsf{Setup}\left(1^{\lambda}, 1^{L_{max}}\right)$: On input security parameter λ and maximum set size L_{max}, output the master secret key MSK.
- $\mathsf{KeyGen}(\mathsf{MSK}, T)$: On input master secret key MSK, and a set $T \subseteq U$, output the functional key sk_T.
- $\mathsf{Enc}(\mathsf{MSK}, S)$: On input master secret key MSK and input set S, output the ciphertext CT.
- $\mathsf{Dec}(\mathsf{sk}_T, \mathsf{CT})$: On input functional key sk_T, and ciphertext CT, outputs set P.

Similarly, our public-key functional encryption for set intersection consists of these algorithms:

- $\mathsf{Setup}\left(1^{\lambda}, 1^{Q}, 1^{L_{max}}\right)$: On input security parameter λ and query bound Q, and maximum set size L_{max} for the scheme, output the master secret key MSK, and master public key PK.

- $\mathsf{KeyGen}(\mathsf{MSK}, T)$: On input master secret key MSK, and a set $T \subseteq U$, output the functional key sk_T.
- $\mathsf{Enc}(\mathsf{PK}, S)$: On input master public key PK and input set S, output the ciphertext CT.
- $\mathsf{Dec}(\mathsf{sk}_T, \mathsf{CT})$: On input functional key sk_T, and ciphertext CT, outputs set P.

5.1 Correctness

Consider the input set $S = \{s_1, \ldots, s_\gamma\} \subseteq U$ of size γ and the functional set $T = \{t_1, \ldots, t_\mu\} \subseteq U$ of size μ. For the correctness of the private-key functional encryption scheme for set intersection, we require the following:

$$\Pr\left[T \cap S = P : \begin{array}{c} \mathsf{MSK} \leftarrow \mathsf{Setup}(1^\lambda, 1^{L_{max}}); \\ \mathsf{sk}_T \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, T); \\ \mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{MSK}, S); \\ P \leftarrow \mathsf{Dec}(\mathsf{sk}_T, \mathsf{CT}) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda)$$

Similarly, for the correctness of public-key functional encryption scheme for set intersection, we require for every $Q \geq 1$:

$$\Pr\left[T \cap S = P : \begin{array}{c} (\mathsf{MSK}, \mathsf{PK}) \leftarrow \mathsf{Setup}(1^\lambda, 1^Q, 1^{L_{max}}); \\ \mathsf{sk}_T \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, T); \\ \mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{PK}, S); \\ P \leftarrow \mathsf{Dec}(\mathsf{sk}_T, \mathsf{CT}) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda)$$

5.2 Security for Private-Key FE for SI

We define two experiments Expt_0(Real experiment) and Expt_1(Ideal experiment). Expt_0 is parameterized by a PPT adversary $\mathcal{A}$ and a real challenger Ch whereas Expt_1 is parameterized by a PPT adversary $\mathcal{A}$ and a stateful simulator Sim. The description of the experiments for $M = poly(\lambda)$ challenge ciphertext queries is as follows:

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

1. $\mathcal{A}$ outputs maximum set length L_{max} .
2. Ch executes $\mathsf{PrivKeySI.Setup}(1^\lambda, 1^{L_{max}})$ to obtain secret key MSK.
3. $\mathcal{A}$, with oracle access to $\mathsf{PrivKeySI.KeyGen}(\mathsf{MSK}, \cdot)$, outputs the challenge sets $S^{(1)}, \ldots, S^{(M)}$.
4. Ch outputs the corresponding challenge ciphertexts $\mathsf{CT}^{(1)} \leftarrow \mathsf{PrivKeySI.Enc}(\mathsf{MSK}, S^{(1)})$, …, $\mathsf{CT}^{(M)} \leftarrow \mathsf{PrivKeySI.Enc}(\mathsf{MSK}, S^{(M)})$.
5. Output bit b.

$\mathsf{Expt}_1^{\mathsf{PrivKeySI},\,\mathcal{A},\,\mathsf{Sim}}(1^\lambda)$:

1. $\mathcal{A}$ outputs maximum set length L_{max}.
2. $\mathcal{A}$, with oracle access to Sim which generates simulated functional keys, outputs challenge sets $S^{(1)}, \ldots, S^{(M)}$.
   - Let QSet be set of queries made by $\mathcal{A}$ to Sim.
   - Construct the sets $V^{(1)}, \ldots, V^{(M)}$ such that $V^{(z)} = \left\{(T, T \cap S^{(i)}) \mid \forall\ T \in \mathsf{QSet}\right\}$ where $z \in [M]$.
3. Sim takes input $\left\{1^{|S^{(z)}|}, V^{(z)}\right\}$ where $z \in [M]$ and outputs the challenge ciphertexts $\mathsf{CT}^{(z)}$.

4. Output bit b.

Definition 1 (Semi-Adaptive Security)**.** *A private-key functional encryption for set intersection scheme* ***PrivKeySI*** *is* ***semi-adaptively secure*** *if for every sufficiently large security parameter $\lambda \in \mathbb{N}$, every PPT adversary $\mathcal{A}$, there exists a PPT simulator* ***Sim*** *such that the following holds:*

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

5.3 Security for Public-Key FE for SI

We define two experiments Expt_0(Real experiment) and Expt_1(Ideal experiment). Expt_0 is parameterized by a PPT adversary $\mathcal{A}$ and a real challenger Ch whereas Expt_1 is parameterized by a PPT adversary $\mathcal{A}$ and a stateful simulator Sim. The description of the experiments is as follows:

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

1. $\mathcal{A}$ outputs a query bound Q and maximum set length L_{max} .
2. Ch executes $\mathsf{PubKeySI.Setup}(1^\lambda, 1^Q, 1^{L_{max}})$ to obtain master public key-master secret key pair $(\mathsf{MSK}, \mathsf{PK})$.
3. $\mathcal{A}$, with oracle access to $\mathsf{PubKeySI.KeyGen}(\mathsf{MSK}, \cdot)$, outputs challenge set S.
4. Ch outputs the challenge ciphertext $\mathsf{CT} \leftarrow \mathsf{PubKeySI.\ Enc}(\mathsf{PK},\ S)$
5. $\mathcal{A}$, with oracle access to $\mathsf{PubKeySI.KeyGen}(\mathsf{MSK}, \cdot)$, outputs bit b.
6. If the number of calls to the oracle are greater than Q, output $\perp$. Otherwise, output b.

$\underline{\mathsf{Expt}_1{}^{\mathsf{PubKeySI},\,\mathcal{A},\,\mathsf{Sim}}(1^\lambda)}$:

1. $\mathcal{A}$ outputs a query bound Q and maximum set length L_{max}.
2. Sim on input $(1^\lambda, 1^Q, 1^{L_{max}})$ outputs master public key PK.
3. $\mathcal{A}$, with oracle access to Sim which generates simulated functional keys, outputs challenge set S.
   - Let QSet be set of queries made by $\mathcal{A}$ to Sim.
   - Construct the set V such that $V = \Big\{(T, T \cap S) \mid \forall\ T \in \mathsf{QSet}\Big\}$.
4. Sim takes input $\left\{1^{|S|}, V\right\}$ and outputs the challenge ciphertext CT.
5. $\mathcal{A}$, with oracle access to Sim which generates simulated functional keys, outputs bit b.
6. If the number of calls to the oracle are greater than Q, output $\perp$. Otherwise, output b.

Definition 2 (Adaptive Security)**.** *A public-key functional encryption for set intersection scheme* $\mathsf{PubKeySI}$ *is* ***adpatively secure*** *if for every sufficiently large security parameter* $\lambda \in \mathbb{N}$*, every PPT adversary* $\mathcal{A}$*, there exists a PPT simulator* Sim *such that the following holds:*

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

5.4 Inner Product Functional Encryption

In this paper, we are concerned with the public-key version of the Inner Product Functional Encryption(ipFE), which is a functional encryption scheme defined using the

following four algorithms:

- $\mathsf{Setup}\left(1^\lambda, 1^L\right)$: On input security parameter λ and the maximum length of vectors L, output the master secret key MSK and the public key PK.
- $\mathsf{KeyGen}\left(\mathsf{MSK}, y\right)$: On input master secret key MSK and a vector y such that $|y| \leq L$, output the functional key sk_y.
- $\mathsf{Enc}\left(\mathsf{PK}, x\right)$: On input public key PK and a vector x such that $|x| \leq L$, output the ciphertext CT.
- $\mathsf{Dec}\left(\mathsf{sk}_y, \mathsf{CT}\right)$: On input functional key sk_y and ciphertext CT, output the value z.

5.4.1 Correctness

The ipFE scheme is said to be correct if the following expression holds for every λ and L and some negligible function $\mathsf{negl}(\cdot)$:

$$\Pr\left[z = \langle x, y\rangle : \begin{array}{c} (\mathsf{MSK}, \mathsf{PK}) \leftarrow \mathsf{Setup}(1^\lambda, 1^L); \\ \mathsf{sk}_y \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, y); \\ \mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{PK}, x); \\ z \leftarrow \mathsf{Dec}(\mathsf{sk}_y, \mathsf{CT}) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda)$$

5.4.2 Security

We define two experiments as follows:

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

1. $\mathcal{A}$ outputs the maximum vector length L
2. Ch executes $\mathsf{ipFE.Setup}(1^\lambda, 1^L)$ to obtain public key-master secret key pair $(\mathsf{MSK}, \mathsf{PK})$
3. $\mathcal{A}$, with oracle access to $\mathsf{ipFE.KeyGen}(\mathsf{MSK}, \cdot)$, outputs challenge vector x.

4. Ch outputs the challenge ciphertext

   $\mathsf{CT} \leftarrow \mathsf{ipFE.Enc}(\mathsf{PK}, x)$

5. $\mathcal{A}$, with oracle access to $\mathsf{ipFE.KeyGen}(\mathsf{MSK}, \cdot)$, outputs bit b.

$\mathsf{Expt}_1{}^{\mathsf{ipFE},\ \mathcal{A},\ \mathsf{Sim}}(1^\lambda)$:

1. $\mathcal{A}$ outputs the maximum vector length L

2. Sim on input $(1^\lambda, 1^L)$ outputs master public key PK

3. $\mathcal{A}$, with oracle access to Sim which generates simulated functional keys, outputs challenge vector x.

   - Let QSet be set of queries made by $\mathcal{A}$ to Sim.
   - Construct the set V such that $V = \big\{\big(y, \langle x, y\rangle\big) \mid \forall\ y \in \mathsf{QSet}\big\}$.

4. Sim takes input $\big\{1^{|x|}, V\big\}$ and outputs the challenge ciphertext CT.

5. $\mathcal{A}$, with oracle access to Sim which generates simulated functional keys, outputs bit b.

Definition 3 (Adaptive Security)**.** *A public-key inner product functional encryption scheme ipFE is* ***adpatively secure*** *if for every sufficiently large security parameter $\lambda \in \mathbb{N}$, every PPT adversary $\mathcal{A}$, there exists a PPT simulator Sim such that the following holds:*

$$\Big|\Pr\big[0 \leftarrow \mathit{Expt}_0{}^{\mathit{ipFE},\ \mathcal{A},\ \mathit{Ch}}(1^\lambda)\big] - \Pr\big[0 \leftarrow \mathit{Expt}_1{}^{\mathit{ipFE},\ \mathcal{A},\ \mathit{Sim}}(1^\lambda)\big]\Big| \leq \mathsf{negl}(\lambda)$$

Chapter 6

Private-Key FE for SI

In this section, we present the construction, correctness, efficiency, and security of our private-key functional encryption scheme for set intersection.

6.1 Construction

Theorem 4. *There exists a private-key semi-adaptively secure functional encryption for set intersection scheme* PrivKeySI *under the assumption that one-way functions exist.*

Proof: For the construction of PrivKeySI, we rely on two pseudorandom functions, $\mathsf{PRF}_1 : \{0,1\}^\lambda \times \{0,1\}^l \to \{0,1\}^m$, $\mathsf{PRF}_2 : \{0,1\}^m \times \{0,1\}^\lambda \to \{0,1\}^m$. The first input to PRF_1(and PRF_2) is the key and the second input is the string on which PRF_1 (and PRF_2) needs to be evaluated upon. The detailed description of the construction is as follows:

$\underline{\mathsf{Setup}}(1^\lambda, 1^{L_{max}})$: On input security parameter λ and the maximum set size L_{max}, generate $\mathsf{fk} \xleftarrow{\$} \{0,1\}^\lambda$ and $\mathsf{rk} \xleftarrow{\$} \{0,1\}^\lambda$. L_{max} is implicitly provided to KeyGen and Enc. Output $\mathsf{MSK} = (\mathsf{fk}, \mathsf{rk})$.

$\underline{\mathsf{KeyGen}(\mathsf{MSK}, T = \{t_1, \ldots, t_\mu\})}$: On input master secret key MSK and the set T, $\mu \leq L_{max}$, output the functional key as:

$$\mathsf{sk}_T = \Big(\Big(t_1, \mathsf{PRF}_1(\mathsf{fk}, t_1), \mathsf{PRF}_1(\mathsf{rk}, t_1)\Big), \ldots, \Big(t_\mu, \mathsf{PRF}_1(\mathsf{fk}, t_\mu), \mathsf{PRF}_1(\mathsf{rk}, t_\mu)\Big)\Big).$$

$\underline{\mathsf{Enc}(\mathsf{MSK}, S = \{s_1, \ldots, s_\gamma\})}$: We permute S before starting the procedure. On input the master secret key MSK and the set $S = \{s_1, ..., s_\gamma\}$, $\gamma \leq L_{max}$,

- Parse MSK as $(\mathsf{fk}, \mathsf{rk})$.
- Sample $\forall i \in [\gamma], r_i \xleftarrow{\$} \{0,1\}^\lambda$.
- Compute $\forall i \in [\gamma], k_i \leftarrow \mathsf{PRF}_1(\mathsf{fk}, s_i)$ and $mk_i \leftarrow \mathsf{PRF}_1(\mathsf{rk}, s_i)$.
- Compute $\forall i \in [\gamma], ct_i \leftarrow \mathsf{PRF}_2(k_i, r_i) \oplus mk_i$.
- Output $\mathsf{CT} = \Big(\Big(r_1, ct_1\Big), \ldots, \Big(r_\gamma, ct_\gamma\Big)\Big)$.

$\underline{\mathsf{Dec}(\mathsf{sk}_T, \mathsf{CT})}$: The deterministic decryption procedure takes the functional key associated with the set T, sk_T, and the ciphertext CT as input,

- Parse the ciphertext as $\mathsf{CT} = (\mathsf{CT}_1, \ldots, \mathsf{CT}_\gamma)$ such that $\mathsf{CT}_i = (r_i, e_i)$ where $i \in [\gamma]$.
- Parse the secret key as $\mathsf{sk}_T = (t_j, x_j, y_j)_{j=1}^{\mu}$.
- Compute set $P = \big\{t_j \mid \big(t_j, x_j, y_j\big) \in \mathsf{sk}_T \wedge \exists i \in [\gamma] : (r_i, e_i) \in \mathsf{CT} \wedge e_i = \mathsf{PRF}_2\big(x_j, r_i\big) \oplus y_j\big\}$.
- Output set P.

□

6.2 Correctness

Consider two sets $S = \{s_1, \ldots, s_\gamma\} \subseteq U$ and $T = \{t_1, \ldots, t_\mu\} \subseteq U$ such that $|S| = \gamma \leq L_{max}$ and $|T| = \mu \leq L_{max}$ and $\mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{MSK}, S)$ and $\mathsf{sk}_T \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, T)$. It follows that:

$$\mathsf{sk}_T = \Big(\Big(t_1, \mathsf{PRF}_1(\mathsf{fk}, t_1), \mathsf{PRF}_1(\mathsf{rk}, t_1)\Big), \ldots, \Big(t_\mu, \mathsf{PRF}_1(\mathsf{fk}, t_\mu), \mathsf{PRF}_1(\mathsf{rk}, t_\mu)\Big)\Big)$$

$$\mathsf{CT} = \Big(\Big(r_1, \mathsf{PRF}_2(\mathsf{PRF}_1(\mathsf{fk}, s_1), r_1) \oplus \mathsf{PRF}_1(\mathsf{rk}, s_1)\Big), \ldots, \Big(r_\gamma, \mathsf{PRF}_2(\mathsf{PRF}_1(\mathsf{fk}, s_\gamma), r_\gamma) \oplus \mathsf{PRF}_1(\mathsf{rk}, s_\gamma)\Big)\Big)$$

We show that the output of $\mathsf{Dec}\,(\mathsf{sk}_T, \mathsf{CT})$ is $T \cap S$ with overwhelming probability.

$$\mathsf{Dec}\,(\mathsf{sk}_T, \mathsf{CT}) = \big\{t_j \mid (t_j, x_j, y_j) \in \mathsf{sk}_T \ \wedge \exists i \in [\gamma] : (r_i, e_i) \in \mathsf{CT} \wedge \ e_i = \mathsf{PRF}_2(x_j, r_i) \oplus y_j\big\}$$

$$= \{t_j \mid (t_j, \mathsf{PRF}_1(\mathsf{fk}, t_j), \mathsf{PRF}_1(\mathsf{rk}, t_j)) \in \mathsf{sk}_T \ \wedge \exists i \in [\gamma] : (r_i, \mathsf{PRF}_2(\mathsf{PRF}_1(\mathsf{fk}, s_i), r_i)) \in \mathsf{CT}$$

$$\wedge\ \mathsf{PRF}_2(\mathsf{PRF}_1(\mathsf{fk}, s_i), r_i) \oplus \mathsf{PRF}_1(\mathsf{rk}, s_i) = \mathsf{PRF}_2(\mathsf{PRF}_1(\mathsf{fk}, t_j), r_i) \oplus \mathsf{PRF}_1(\mathsf{rk}, t_j)\}$$

As dictated by PRF security, the output distribution of the PRF evaluations is computationally indistinguishable from the uniform distribution. The probability that the two strings sampled uniformly at random are identical is negligible in the length of the string. Hence, the probability that the output of PRF on different inputs, under the same key, is identical is also negligible. Hence, with overwhelming probability we have

$$= \{t_j \mid (t_j, \mathsf{PRF}_1(\mathsf{fk}, t_j), \mathsf{PRF}_1(\mathsf{rk}, t_j)) \in \mathsf{sk}_T \ \wedge \exists i \in [\gamma] : \big(r_i, \mathsf{PRF}_2(\mathsf{PRF}_1(\mathsf{fk}, s_i), r_i) \oplus$$

$$\mathsf{PRF}_1(\mathsf{rk}, s_j)\big) \in \mathsf{CT} \wedge \ s_i = t_j\}$$

$$= T \cap S.$$

6.3 Efficiency

Assuming that the running times of PRF_1 and PRF_2 are polynomial in the security parameter λ,

- **Setup:** We are generating two keys of size λ. Hence, we take $O(\lambda)$ time and the output size is $O(\lambda)$.
- **Enc:** We use $O(L_{max})$ evaluations of PRF_2 and $O(2 \cdot L_{max})$ evaluations of PRF_1 respectively. The size of the ciphertext generated by Enc algorithm is $O(L_{max} \cdot (m + \lambda))$.
- **KeyGen:** We invoke $O(2 \cdot L_{max})$ evaluations of PRF_1. The size of the functional keys generated by KeyGen algorithm is $O(L_{max} \cdot (l + 2 \cdot m))$.
- **Dec:** $O(L_{max})$ instantiations of PRF_2 and $O(L_{max}^2)$ pairwise comparisons occur in this algorithm.

6.4 Security

We prove the semi-adaptive security of the $\mathsf{PrivKeySI}$ using a simulator which poses as a challenger against the adversary in the ideal world experiment as mentioned in 5.2. While the real challenger always responds to the queries by the adversary using the master secret key, MSK, the simulator on the other hand does not hold MSK. However, the simulator generates responses which are indistinguishable from challenger's responses to any polynomial-time adversary. The simulator does so by sampling uniformly random strings of appropriate length and substituting the outputs of PRF_1 and PRF_2 accordingly. Moreover, the simulator can only access the lengths of the challenge ciphertexts and the pairwise intersection of the challenge ciphertext set with every functional query set. Using

the security of the pseudorandom functions PRF_1 and PRF_2, we describe the simulator as follows:

Simulator

1. Sim receives maximum set length L_{max} from $\mathcal{A}$.

2. Let $\mathcal{A}$ make total Q queries throughout the experiment. For $q \in [Q]$, let $T^{(q)} = \left\{t_1^{(q)}, \ldots, t_{\mu_q}^{(q)}\right\}$ be the set associated with q-th query and $\mu_q \leq L_{max}$ be the number of set elements in $T^{(q)}$. Maintain a QSet across all the queries such that it should contain the set elements and their corresponding query responses. Initially, the QSet is empty.

   For every query $q \in [Q]$ and $j \in [\mu_q]$ do the following:

   - If $\left(t_j^{(q)}, \cdot, \cdot\right) \in \mathsf{QSet}$ then retrieve $\alpha_j^{(q)}$, $\beta_j^{(q)}$ from QSet.
   - Otherwise, sample $\alpha_j^{(q)} \xleftarrow{\$} \{0,1\}^m$, $\beta_j^{(q)} \xleftarrow{\$} \{0,1\}^m$ and add $\left(t_j^{(q)}, \alpha_j^{(q)}, \beta_j^{(q)}\right)$ to QSet.

   Sim sends $\mathsf{sk}_{T^{(q)}} = \Big(\left(t_1^{(q)}, \alpha_1^{(q)}, \beta_1^{(q)}\right), \ldots,$
   $\left(t_{\mu_q}^{(q)}, \alpha_{\mu_q}^{(q)}, \beta_{\mu_q}^{(q)}\right) \Big)$ to $\mathcal{A}$.

3. $\mathcal{A}$ outputs challenge sets as $S^{(1)}, \ldots, S^{(M)}$ where $S^{(z)} = \left\{s_1^{(z)}, \ldots, s_{\gamma_z}^{(z)}\right\}$, $\forall z \in [M]$ and $M = poly(\lambda)$.

4. For every challenge set $S^{(z)}$ of size $\gamma_z \leq L_{max}$, where $z \in [M]$, Sim takes $\left\{1^{\gamma_z}, \left\{ \left(T^{(1)}, S^{(z)} \cap T^{(1)}\right), \ldots, \left(T^{(Q)}, S^{(z)} \cap T^{(Q)}\right)\right\}\right\}$ as input and executes the following steps to compute the ciphertext for $S^{(z)}$. For every set element $s_i^{(z)}$ where $i \in [\gamma_z]$,

   - Let $\left\{s_1^{(z)}, \ldots, s_{\phi_z}^{(z)}\right\} = S^{(z)} \cap \left(T^{(1)} \cup \ldots \cup T^{(z)}\right)$.

- For every $i \in [\phi_z]$, $s_i^{(z)} \in \mathsf{QSet}$. Retrieve $\left(s_i^{(z)}, \alpha^{s_i^{(z)}}, \beta^{s_i^{(z)}}\right)$ from QSet. Sample $\rho^{s_i^{(z)}} \xleftarrow{\$} \{0,1\}^\lambda$ and compute $\theta^{s_i^{(z)}} \leftarrow \mathsf{PRF}_2(\alpha^{s_i^{(z)}}, \rho^{s_i^{(z)}}) \oplus \beta^{s_i^{(z)}}$.
- For $i \in [\gamma_z - \phi_z]$, sample $\eta^{s_i^{(z)}} \xleftarrow{\$} \{0,1\}^l, \rho^{s_i^{(z)}} \xleftarrow{\$} \{0,1\}^\lambda, \theta^{s_i^{(z)}}, \xleftarrow{\$} \{0,1\}^m$.
- Generate a random permutation of $\{s_1^{(z)}, \ldots, s_{\gamma_z}^{(z)}, \eta_1^{(z)}, \ldots, \eta_{\gamma_z}^{(z)}\}$ to obtain $\{\varsigma_1^{(z)}, \ldots, \varsigma_\gamma^{(z)}\}$.

Sim sends $\mathsf{CT}^{(z)} = \left(\left(\rho^{\varsigma_1^{(z)}}, \theta^{\varsigma_1^{(z)}}\right), \ldots, \left(\rho^{\varsigma_{\gamma_z}^{(z)}}, \theta^{\varsigma_{\gamma_z}^{(z)}}\right)\right)$ to $\mathcal{A}$.

5. $\mathcal{A}$ outputs bit b. Output bit b.

We show the computational indistinguishability between the real challenger and the simulator using a series of hybrids as mentioned below.

$\underline{\text{Hyb}_0}$: Same as Expt_0 from 5.2. The hybrid consists of the following steps:

1. Ch receives maximum set length L_{max}.
2. Ch generates master secret key $\mathsf{MSK} = (\mathsf{fk}, \mathsf{rk})$ as $\mathsf{fk} \xleftarrow{\$} \{0,1\}^\lambda$ and $\mathsf{rk} \xleftarrow{\$} \{0,1\}^\lambda$.
3. Let $\mathcal{A}$ make Q queries to Ch. For every query $q \in [Q]$, challenger Ch computes the secret key as follows: $\mathsf{sk}_{T^{(q)}} = \Big(\Big(t_1^{(q)}, \mathsf{PRF}_1\left(\mathsf{fk}, t_1^{(q)}\right), \mathsf{PRF}_1\left(\mathsf{rk}, t_1^{(q)}\right)\Big), \ldots, \Big(t_{\mu_q}^{(q)}, \mathsf{PRF}_1\left(\mathsf{fk}, t_{\mu_q}^{(q)}\right), \mathsf{PRF}_1\left(\mathsf{rk}, t_{\mu_q}^{(q)}\right)\Big)\Big)$.
   Send $\mathsf{sk}_{T^{(q)}}$ to $\mathcal{A}$.
4. Ch receives the challenge set queries as $S^{(1)}, \ldots, S^{(M)}$, where $S^{(z)} = \left\{s_1^{(z)}, \ldots, s_{\gamma_z}^{(z)}\right\}$ $\forall z \in [M]$, from $\mathcal{A}$.
5. For every challenge ciphertext set $S^{(z)}$ of size $\gamma_z \leq L_{max}$, where $z \in [M]$, Ch does the following :
   - Sample $\forall i \in [\gamma_z], r_i^{(z)} \xleftarrow{\$} \{0,1\}^\lambda$.
   - Compute $\forall i \in [\gamma_z], k_i^{(z)} \leftarrow \mathsf{PRF}_1(\mathsf{fk}, s_i^{(z)})$ and $mk_i^{(z)} \leftarrow \mathsf{PRF}_1(\mathsf{rk}, s_i^{(z)})$.

- Compute $\forall i \in [\gamma_z], ct_i^{(z)} \leftarrow \mathsf{PRF}_2\left(k_i^{(z)}, r_i^{(z)}\right) \oplus mk_i^{(z)}$.

Ch sends $\mathsf{CT}^{(z)} = \left(\left(r_1^{(z)}, ct_1^{(z)}\right), \ldots, \left(r_{\gamma_z}^{(z)}, ct_{\gamma_z}^{(z)}\right)\right)$ to $\mathcal{A}$.

6. $\mathcal{A}$ outputs bit b. Output b.

$\underline{\text{Hyb}_1}$: In this hybrid, we replace all outputs of $\mathsf{PRF}_1(\mathsf{fk}, \cdot)$ with a uniformly random string of appropriate length. The changes are marked in red. The hybrid consists of the following steps:

1. Ch receives maximum set length L_{max}.
2. Ch generates master secret key $\mathsf{MSK} = (\mathsf{fk}, \mathsf{rk})$ as $\mathsf{fk} \xleftarrow{\$} \{0,1\}^\lambda$ and $\mathsf{rk} \xleftarrow{\$} \{0,1\}^\lambda$.
3. Initialize QSet to empty set.
4. Let $\mathcal{A}$ make Q circuit queries to Ch. For every query $q \in [Q]$, Ch computes the secret key as follows:
   - If $\left(t_j^{(q)}, \cdot\right) \in \mathsf{QSet}$ then retrieve $\alpha_j^{(q)}$.
   - Otherwise, sample $\alpha_j^{(q)} \xleftarrow{\$} \{0,1\}^m$ and add $\left(t_j^{(q)}, \alpha_j^{(q)}\right)$ to QSet.

   Ch sends $\mathsf{sk}_{T^{(q)}} = \left(\left(t_1^{(q)}, \alpha_1^{(q)}, \mathsf{PRF}_1\left(\mathsf{rk}, t_1^{(q)}\right)\right), \ldots, \left(t_{\mu_q}^{(q)}, \alpha_{\mu_q}^{(q)}, \mathsf{PRF}_1\left(\mathsf{rk}, t_{\mu_q}^{(q)}\right)\right)\right)$ to $\mathcal{A}$.
5. Ch receives the challenge set queries as $S^{(1)}, \ldots, S^{(M)}$, where $S^{(z)} = \left\{s_1^{(z)}, \ldots, s_{\gamma_z}^{(z)}\right\}$ $\forall z \in [M]$, from $\mathcal{A}$.
6. For every challenge ciphertext set $S^{(z)}$ of size $\gamma_z \leq L_{max}$, where $z \in [M]$, Ch does the following :
   - Sample $\forall i \in [\gamma_z]$, $r_i^{(z)} \xleftarrow{\$} \{0,1\}^\lambda$.

- $\forall i \in [\gamma_z]$, if $\left(s_i^{(z)}, \cdot\right) \in \mathsf{QSet}$ then retrieve $\alpha^{s_i^{(z)}}$ and set $k_i^{(z)} = \alpha^{s_i^{(z)}}$. Otherwise, compute $k_i^{(z)} = \alpha^{s_i^{(z)}}$ where $\alpha^{s_i^{(z)}} \xleftarrow{\$} \{0,1\}^m$. Add $\left(s_i^{(z)}, \alpha^{s_i^{(z)}}\right)$ to QSet.
- Compute $\forall i \in [\gamma_z]$, $mk_i^{(z)} \leftarrow \mathsf{PRF}_1\left(\mathsf{rk}, s_i^{(z)}\right)$.
- Compute $\forall i \in [\gamma_z], ct_i^{(z)} \leftarrow \mathsf{PRF}_2\left(k_i^{(z)}, r_i^{(z)}\right) \oplus mk_i^{(z)}$.

Ch sends $\mathsf{CT}^{(z)} = \left(\left(r_1^{(z)}, ct_1^{(z)}\right), \ldots, \left(r_{\gamma_z}^{(z)}, ct_{\gamma_z}^{(z)}\right)\right)$ to $\mathcal{A}$.

7. $\mathcal{A}$ outputs bit b. Output b.

Claim 5. *Assuming the security of pseudorandom function* PRF_1*, the output distributions of the* Hyb_0 *and* Hyb_1 *are computationally indistinguishable.*

Proof: If there were a PPT adversary $\mathcal{A}$ that can distinguish between Hyb_0 and Hyb_1, that is,

$$\left|\Pr\left[0 \leftarrow \mathcal{A}^{\text{Hyb}_0}(1^\lambda)\right] - \Pr\left[0 \leftarrow \mathcal{A}^{\text{Hyb}_1}(1^\lambda)\right]\right| > \mathsf{negl}\ (\lambda)$$

We can use $\mathcal{A}$ to construct a reduction $\mathcal{B}$ that can break the security of PRF_1. In other words, $\mathcal{B}$ can act as the challenger interacting with $\mathcal{A}$ and $\mathcal{B}$ can distinguish between oracle access to $\mathsf{PRF}_1(\mathsf{fk}, \cdot) : \{0,1\}^l \to \{0,1\}^m$ and a random function $\mathcal{R}(\cdot) : \{0,1\}^l \to \{0,1\}^m$ with non-negligible probability. Formally,

$$|\Pr[\mathcal{B}^{\mathsf{PRF}_1(\mathsf{fk},\cdot)}(1^\lambda) = 1] - \Pr[\mathcal{B}^{R(\cdot)}(1^\lambda) = 1]| > \mathsf{negl}(\lambda)$$

The description of $\mathcal{B}$ is as follows:

Algorithm 1 $\mathcal{B}^{\mathcal{O}}\ (1^\lambda)$

1. For every query q by $\mathcal{A}$ where $q \in [Q]$ and $j \in [\mu_q]$: Receive $t_j^{(q)}$ from adversary $\mathcal{A}$. Query oracle function with $t_j^{(q)}$ to obtain $\alpha_j^{(q)}$. Forward $\left(t_j^{(q)}, \alpha_j^{(q)}\right)$ to $\mathcal{A}$ as the query response. Add $\left(t_j^{(q)}, \alpha_j^{(q)}\right)$ to QSet.

2. For every challenge ciphertext set query $S^{(z)}$ by $\mathcal{A}$ where $z \in [M]$: If $\left(s_i^{(z)}, \cdot\right) \in \mathsf{QSet}$, retrieve $\alpha^{s_i^{(z)}}$ from QSet. Otherwise, query the oracle function with $s_i^{(z)}$, where $i \in [\gamma_z]$ to obtain $\alpha^{s_i^{(z)}}$ and sample $r_i^{(z)} \xleftarrow{\$} \{0,1\}^\lambda$. Forward $\left(r_i^{(z)}, \mathsf{PRF}_2\left(\alpha^{s_i^{(z)}}, r_i^{(z)}\right) \oplus \mathsf{PRF}_1\left(\mathsf{rk}, s_i^{(z)}\right)\right)$ to $\mathcal{A}$ as a query response.

3. Simulate any other steps same as in Hyb_0.

4. $\mathcal{A}$ outputs the bit b. $\mathcal{B}$ outputs b.

If the oracle access given is a pseudorandom function then the $\mathcal{A}$ guesses Hybrid 0 with non-negligible probability. If the oracle access given is a random function then the $\mathcal{A}$ guesses Hybrid 1 with non-negligible probability. This comes from the fact that $\mathcal{A}$ has a non-negligible advantage in distinguishing between Hybrids 0 and 1. From the construction of $\mathcal{B}$, it follows that the advantage of $\mathcal{B}$ in breaking the PRF_1 security is same as the advantage of $\mathcal{A}$ which is non-negligible as per our assumption. However, from pseudorandom function definition, we know that the advantage of any PPT adversary in breaking PRF_1 security is negligible. This contradicts our initial assumption which now implies that no such adversary $\mathcal{A}$ exists. Hence, Hybrids 0 and 1 are computationally indistinguishable. □

$\underline{\text{Hyb}_2}$: In this hybrid, we replace all outputs of $\mathsf{PRF}_1(\mathsf{rk}, \cdot)$ with a uniformly random string of appropriate length. The rest of the steps remain same as Hyb_1:

1. Ch receives maximum set length L_{max}.

2. Ch generates master secret key $\mathsf{MSK} = (\mathsf{fk}, \mathsf{rk})$ as $\mathsf{fk} \xleftarrow{\$} \{0,1\}^\lambda$ and $\mathsf{rk} \xleftarrow{\$} \{0,1\}^\lambda$.

3. Initialize QSet to empty set.

4. Let $\mathcal{A}$ make Q circuit queries to Ch. For every query $q \in [Q]$, Ch computes the secret key as follows:

   - If $\left(t_j^{(q)}, \cdot, \cdot\right) \in \mathsf{QSet}$ then retrieve $\alpha_j^{(q)}$ and $\beta_j^{(q)}$.
   - Otherwise, sample $\alpha_j^{(q)} \xleftarrow{\$} \{0,1\}^m$, $\beta_j^{(q)} \xleftarrow{\$} \{0,1\}^m$ and add $\left(t_j^{(q)}, \alpha_j^{(q)}, \beta_j^{(q)}\right)$ to QSet.

   Ch sends $\mathsf{sk}_{T^{(q)}} = \left(\left(t_1^{(q)}, \alpha_1^{(q)}, \beta_1^{(q)}\right), \ldots, \left(t_{\mu_q}^{(q)}, \alpha_{\mu_q}^{(q)}, \beta_{\mu_q}^{(q)}\right)\right)$ to $\mathcal{A}$.

5. Ch receives the challenge set queries as $S^{(1)}, \ldots, S^{(M)}$, where $S^{(z)} = \{s_1^{(z)}, \ldots, s_{\gamma_z}^{(z)}\}$ $\forall z \in [M]$, from $\mathcal{A}$.

6. For every challenge ciphertext set $S^{(z)}$ of size $\gamma_z \leq L_{max}$, where $z \in [M]$, Ch does the following :

   - Sample $\forall i \in [\gamma_z]$, $r_i^{(z)} \xleftarrow{\$} \{0,1\}^\lambda$.
   - $\forall i \in [\gamma_z]$, if $\left(s_i^{(z)}, \cdot, \cdot\right) \in \mathsf{QSet}$ then retrieve $\alpha^{s_i^{(z)}}$, $\beta^{s_i^{(z)}}$ and set $k_i^{(z)} = \alpha^{s_i^{(z)}}$, $mk_i^{(z)} = \beta^{s_i^{(z)}}$. Otherwise, compute $k_i^{(z)} = \alpha^{s_i^{(z)}}$, $mk_i^{(z)} = \beta^{s_i^{(z)}}$ where $\alpha^{s_i^{(z)}} \xleftarrow{\$} \{0,1\}^m$, $\beta^{s_i^{(z)}} \xleftarrow{\$} \{0,1\}^m$. Add $\left(s_i^{(z)}, \alpha^{s_i^{(z)}}, \beta^{s_i^{(z)}}\right)$ to QSet.
   - Compute $\forall i \in [\gamma_z], ct_i^{(z)} \leftarrow \mathsf{PRF}_2\left(k_i^{(z)}, r_i^{(z)}\right) \oplus mk_i^{(z)}$.

   Ch sends $\mathsf{CT}^{(z)} = \left(\left(r_1^{(z)}, ct_1^{(z)})\right), \ldots, \left(r_{\gamma_z}^{(z)}, ct_{\gamma_z}^{(z)}\right)\right)$ to $\mathcal{A}$.

7. $\mathcal{A}$ outputs bit b. Output b.

Claim 6. *Assuming the security of the pseudorandom function* PRF_2*, the output distributions of the* Hyb_1 *and* Hyb_2 *are computationally indistinguishable.*

Proof: If there were a PPT adversary $\mathcal{A}$ that can distinguish between Hyb_1 and Hyb_2, that is,

$$\left|\Pr\left[0 \leftarrow \mathcal{A}^{\text{Hyb}_1}(1^\lambda)\right] - \Pr\left[0 \leftarrow \mathcal{A}^{\text{Hyb}_2}(1^\lambda)\right]\right| > \mathsf{negl}\ (\lambda)$$

We can use $\mathcal{A}$ to construct a reduction $\mathcal{B}$ that can break the security of PRF_1. In other words, $\mathcal{B}$ can act as the challenger interacting with $\mathcal{A}$ and $\mathcal{B}$ can distinguish between oracle access to $\mathsf{PRF}_1(\mathsf{rk}, \cdot) : \{0,1\}^l \to \{0,1\}^m$ and a random function $\mathcal{R}(\cdot) : \{0,1\}^l \to \{0,1\}^m$ with non-negligible probability. Formally,

$$|\Pr[\mathcal{B}^{\mathsf{PRF}_1(\mathsf{rk},\cdot)}(1^\lambda) = 1] - \Pr[\mathcal{B}^{R(\cdot)}(1^\lambda) = 1]| > \mathsf{negl}(\lambda)$$

The description of $\mathcal{B}$ is as follows:

Algorithm 2 $\mathcal{B}^{\mathcal{O}}\ (1^\lambda)$

1. For every query q by $\mathcal{A}$ where $q \in [Q]$ and $j \in [\mu_q]$: Receive $t_j^{(q)}$ from adversary $\mathcal{A}$. Sample $\alpha_j^{(q)} \xleftarrow{\$} \{0,1\}^m$. Query oracle function with $t_j^{(q)}$ to obtain $\beta_j^{(q)}$. Forward $\left(t_j^{(q)}, \alpha_j^{(q)}, \beta_j^{(q)}\right)$ to $\mathcal{A}$ as the query response. Add $\left(t_j^{(q)}, \alpha_j^{(q)}, \beta_j^{(q)}\right)$ to QSet.

2. For every challenge ciphertext set query $S^{(z)}$ by $\mathcal{A}$ where $z \in [M]$: If $\left(s_i^{(z)}, \cdot, \cdot\right)$ *in* QSet, retrieve $\alpha^{s_i^{(z)}}$ from QSet. Otherwise, sample $r_i^{(z)} \xleftarrow{\$} \{0,1\}^\lambda$, $\alpha^{s_i^{(z)}} \xleftarrow{\$} \{0,1\}^m$. Query the oracle function with $s_i^{(z)}$ where $i \in [\gamma_z]$ to obtain $\beta^{s_i^{(z)}}$. Forward $\left(r_i^{(z)}, \mathsf{PRF}_2\left(\alpha^{s_i^{(z)}}, r_i^{(z)}\right) \oplus \beta^{s_i^{(z)}}\right)$ to $\mathcal{A}$ as a query response. Add $\left(s_i^{(q)}, \alpha^{s_i^{(q)}}, \beta^{s_i^{(q)}}\right)$ to QSet.

3. Simulate any other steps same as in Hyb_1.

4. $\mathcal{A}$ outputs the bit b. $\mathcal{B}$ outputs b.

If the oracle access given is a pseudorandom function then the $\mathcal{A}$ guesses Hybrid 1 with non-negligible probability. If the oracle access given is a random function then the

$\mathcal{A}$ guesses Hybrid 2 with non-negligible probability. This comes from the fact that $\mathcal{A}$ has a non-negligible advantage in distinguishing between Hybrids 1 and 2. From the construction of $\mathcal{B}$, it follows that the advantage of $\mathcal{B}$ in breaking the PRF_1 security is same as the advantage of $\mathcal{A}$ which is non-negligible as per our assumption. However, from pseudorandom function definition, we know that the advantage of any PPT adversary in breaking PRF_1 security is negligible. This contradicts our initial assumption which now implies that no such adversary $\mathcal{A}$ exists. Hence, Hybrids 1 and 2 are computationally indistinguishable. □

Hyb$_3$: In this hybrid, we replace all outputs of PRF_2 with a uniformly random string of appropriate length. The rest of the steps remain same as Hyb$_2$:

1. Ch receives maximum set length L_{max}.
2. Ch generates master secret key $\mathsf{MSK} = (\mathsf{fk}, \mathsf{rk})$ as $\mathsf{fk} \xleftarrow{\$} \{0,1\}^\lambda$ and $\mathsf{rk} \xleftarrow{\$} \{0,1\}^\lambda$.
3. Initialize QSet to empty set.
4. Let $\mathcal{A}$ make Q circuit queries to Ch. For every query $q \in [Q]$, Ch computes the secret key as follows:
   - If $\left(t_j^{(q)}, \cdot, \cdot\right) \in \mathsf{QSet}$ then retrieve $\alpha_j^{(q)}$ and $\beta_j^{(q)}$.
   - Otherwise, sample $\alpha_j^{(q)} \xleftarrow{\$} \{0,1\}^m$, $\beta_j^{(q)} \xleftarrow{\$} \{0,1\}^m$ and add $\left(t_j^{(q)}, \alpha_j^{(q)}, \beta_j^{(q)}\right)$ to QSet.

   Ch sends $\mathsf{sk}_{T^{(q)}} = \left(\left(t_1^{(q)}, \alpha_1^{(q)}, \beta_1^{(q)}\right), \ldots, \left(t_{\mu_q}^{(q)}, \alpha_{\mu_q}^{(q)}, \beta_{\mu_q}^{(q)}\right)\right)$ to $\mathcal{A}$.
5. Ch receives the challenge set queries as $S^{(1)}, \ldots, S^{(M)}$, where $S^{(z)} = \{s_1^{(z)}, \ldots, s_{\gamma_z}^{(z)}\}$ $\forall z \in [M]$, from $\mathcal{A}$.
6. For every challenge ciphertext set $S^{(z)}$ of size $\gamma_z \leq L_{max}$, where $z \in [M]$, Ch does the following :

- Sample $\forall i \in [\gamma_z]$, $r_i^{(z)} \xleftarrow{\$} \{0,1\}^\lambda$.
- $\forall i \in [\gamma_z]$, if $\left(s_i^{(z)}, \cdot, \cdot\right) \in \mathsf{QSet}$ then retrieve $\alpha^{s_i^{(z)}}$, $\beta^{s_i^{(z)}}$ and set $k_i^{(z)} = \alpha^{s_i^{(z)}}$, $mk_i^{(z)} = \beta_i^{s^{(z)}}$. Compute $\forall i \in [\gamma_z], ct_i^{(z)} \leftarrow \mathsf{PRF}_2\left(k_i^{(z)}, r_i^{(z)}\right) \oplus mk_i^{(z)}$.
- Otherwise, sample $ct_i^{(z)} \xleftarrow{\$} \{0,1\}^m$.

Ch sends $\mathsf{CT}^{(z)} = \left(\left(r_1^{(z)}, ct_1^{(z)}\right), \ldots, \left(r_{\gamma_z}^{(z)}, ct_{\gamma_z}^{(z)}\right)\right)$ to $\mathcal{A}$.

7. $\mathcal{A}$ outputs bit b. Output b.

Claim 7. *Assuming the security of the pseudorandom function* PRF_2*, the output distributions of the* Hyb_2 *and* Hyb_3 *are computationally indistinguishable.*

Proof: If there were a PPT adversary $\mathcal{A}$ that can distinguish between Hyb_2 and Hyb_3, that is,

$$\left|\Pr\left[0 \leftarrow \mathcal{A}^{\text{Hyb}_2}(1^\lambda)\right] - \Pr\left[0 \leftarrow \mathcal{A}^{\text{Hyb}_3}(1^\lambda)\right]\right| > \mathsf{negl}\ (\lambda)$$

We can use $\mathcal{A}$ to construct a reduction $\mathcal{B}$ that can break the security of PRF_1. In other words, $\mathcal{B}$ can act as the challenger interacting with $\mathcal{A}$ and $\mathcal{B}$ can distinguish between oracle access to $\mathsf{PRF}_2(\kappa, \cdot) : \{0,1\}^\lambda \to \{0,1\}^m$, where $\kappa \xleftarrow{\$} \{0,1\}^m$ and a random function $\mathcal{R}(\cdot) : \{0,1\}^\lambda \to \{0,1\}^m$ with non-negligible probability. Formally,

$$|\Pr[\mathcal{B}^{\mathsf{PRF}_2(\kappa,\cdot)}(1^\lambda) = 1] - \Pr[\mathcal{B}^{R(\cdot)}(1^\lambda) = 1]| > \mathsf{negl}(\lambda)$$

The description of $\mathcal{B}$ is as follows:

Algorithm 3 $\mathcal{B}^{\mathcal{O}}\ (1^\lambda)$

1. For every query q by $\mathcal{A}$ where $q \in [Q]$ and $j \in [\mu_q]$: Receive $t_j^{(q)}$ from adversary $\mathcal{A}$. Sample $\alpha_j^{(q)} \xleftarrow{\$} \{0,1\}^m$ and $\beta_j^{(q)} \xleftarrow{\$} \{0,1\}^m$. Forward $\left(t_j^{(q)}, \alpha_j^{(q)}, \beta_j^{(q)}\right)$ to $\mathcal{A}$ as the query response. Add $\left(t_j^{(q)}, \alpha_j^{(q)}, \beta_j^{(q)}\right)$ to QSet.

2. For every challenge ciphertext set query $S^{(z)}$ by $\mathcal{A}$ where $z \in [M]$: Sample sample $r_i^{(z)} \xleftarrow{\$} \{0,1\}^\lambda$. If $\left(s_i^{(z)}, \cdot, \cdot\right)$ in QSet, retrieve $\alpha^{s_i^{(z)}}$, $\beta^{s_i^{(z)}}$ from QSet and forward $\left(r_i^{(z)}, \mathsf{PRF}_2\left(\alpha^{s_i^{(z)}}, r_i^{(z)}\right) \oplus \beta^{s_i^{(z)}}\right)$ to $\mathcal{A}$ as query response. Otherwise, query the oracle function with $s_i^{(z)}$ where $i \in [\gamma_z]$ to obtain $ct_j^{(z)}$. Forward $\left(s_i^{(z)}, r_i^{(z)}, ct_i^{(z)}\right)$ to $\mathcal{A}$ as a query response.

3. Simulate any other steps same as in Hybrid 2.

4. $\mathcal{A}$ outputs the bit b. $\mathcal{B}$ outputs b.

If the oracle access given is a pseudorandom function then the $\mathcal{A}$ guesses Hybrid 2 with non-negligible probability. If the oracle access given is a random function then the $\mathcal{A}$ guesses Hybrid 3 with non-negligible probability. This comes from the fact that $\mathcal{A}$ has a non-negligible advantage in distinguishing between Hybrids 2 and 3. From the construction of $\mathcal{B}$, it follows that the advantage of $\mathcal{B}$ in breaking the PRF_2 security is same as the advantage of $\mathcal{A}$ which is non-negligible as per our assumption. However, from pseudorandom function definition, we know that the advantage of any PPT adversary in breaking PRF_2 security is negligible. This contradicts our initial assumption which now implies that no such adversary $\mathcal{A}$ exists. Hence, Hybrids 2 and 3 are computationally indistinguishable. □

Hyb$_4$: Same as Expt$_1$ from 5.2.

Claim 8. *The output distributions of the Hyb$_3$ and Hyb$_4$ are identical.*

Proof: By definition of Exp_1, Hyb_3 is same as Hyb_4. The random permutation doesn't change the output distribution as we take the input to encryption as a random permuted order. □

Chapter 7

Public-Key FE for SI

In this section, we present the construction, correctness, efficiency, and security of our public-key functional encryption scheme for set intersection.

7.1 Construction

Theorem 9. *There exists a public-key adaptively secure functional encryption for set intersection scheme* PubKeySI *under the assumption that public key adaptively secure* 1FE *for P/Poly and public key adaptively simulation-based secure inner product functional encryption scheme exist.*

Proof: For the construction of PubKeySI, we rely on the pseudorandom function $\mathsf{PRF} : \{0,1\}^{\lambda} \times \{0,1\}^{l} \rightarrow \{0,1\}^{m}$, where l and m are appropriate polynomials in λ and the inner-product functional encryption scheme ipFE. The ipFE scheme should satisfy the following requirements:

1. ipFE scheme should be adaptively-simulation secure.

2. The Dec algorithm of the scheme should work in two phases. In the first phase,

the output is calculated in the exponent i.e, $g^{\langle x,y,\rangle}$, where x and y are the input vectors to the scheme. In the second phase, $\langle x, y\rangle$ is evaluated using polynomial search which is feasible when the norms of x and y are within a certain range.

3. The aforementioned second phase should be independent of the second phase.

Instantiation: To instantiate an inner product functional encryption scheme that satisfies the aforementioned properties, consider the scheme present in [ALMT20] which relies on the DDH assumption. We edit the decryption algorithm so that it no longer takes the master public key, pk as input. However, the decryption algorithm learns the master public key pk as part of its evaluation. As the master public key is revealed to everyone, this is a reasonable assumption.

Using parameters $t = \Theta(\lambda), n = \Theta(t)$, and $N = \Theta(Q^2 n)$ in accordance with the small-pairwise intersection lemma(10), the public key functional encryption scheme for set intersection is described as follows:

$\underline{\mathsf{Setup}(1^\lambda, 1^Q, 1^{L_{max}})}$: On input security parameter λ, query bound Q, and the maximum set size L_{max}, generate for every $u \in [N]$, $(\mathsf{pk}_u, \mathsf{msk}_u) \leftarrow \mathsf{1FE.Setup}(1^\lambda, 1^\psi)$ and $\forall\ k \in [L_{max}]$, $(\mathsf{ipPK_k}, \mathsf{ipMSK_k}) \leftarrow \mathsf{ipFE.Setup}(1^m, 1^N)$ where ψ is the size of the circuit used in KeyGen and depends on L_{max} and λ.

Output $\mathsf{MSK} = \big(\mathsf{ipMSK}_1, \ldots, \mathsf{ipMSK}_{L_{max}}, \mathsf{msk}_1, \ldots, \mathsf{msk}_N\big)$, $\mathsf{PK} = \big(\mathsf{ipPK}_1, \ldots, \mathsf{ipPK}_{L_{max}}, \mathsf{pk}_1, \ldots, \mathsf{pk}_N\big)$.

$\underline{\mathsf{KeyGen}(\mathsf{MSK}, T)}$: On input set $T = \{t_1, \ldots, t_\mu\}$, where $\mu \leq L_{max}$,

- Parse MSK as $\big(\mathsf{ipMSK}_1, \ldots, \mathsf{ipMSK}_{L_{max}}, \mathsf{msk}_1, \ldots, \mathsf{msk}_N\big)$.
- Sample a random subset $\Delta_T \subset [N]$ of size n.
- Set $\hat{T} = ((1, t_1), \ldots, (\mu, t_\mu))$

- Construct a functional circuit C that takes $K \in \{0,1\}^\lambda$ as input and outputs $(\mathsf{PRF}(K,t_1),\ldots,\mathsf{PRF}(K,t_\mu))$.
- For every $d \in \Delta_T$, $\mathsf{sk}_d \leftarrow \mathsf{1FE.KeyGen}(\mathsf{msk}_d, C)$.
- For every $k \in [L_{max}]$, compute $\mathsf{ipSK}_k \leftarrow \mathsf{ipFE.KeyGen}\left(\mathsf{ipMSK}_k, e^{\Delta_T}\right)$ where e^{Δ_T} is an indicator vector of length N such that $e_d^{\Delta_T} = 1$ if $d \in \Delta_T$ and 0 otherwise.
- Output $\mathsf{sk}_T = \left(\hat{T}, \Delta_T, \{\mathsf{ipSK}_k\}_{k\in[L_{max}]}, \{\mathsf{sk}_d\}_{d\in\Delta_T}\right)$.

$\underline{\mathsf{Enc}(\mathsf{PK}, S)}$: On input set $S = \{s_1,\ldots,s_\gamma\}$, where $\gamma \le L_{max}$. We assume that S is given in a random permuted order. If this is not true, we can permute them before $\mathsf{ipFE.Enc}$ phase.

- Parse PK as $\left(\mathsf{ipPK}_1,\ldots,\mathsf{ipPK}_{L_{max}},\mathsf{pk}_1,\ldots,\mathsf{pk}_N\right)$.
- For every $u \in [N]$, sample $K_u \xleftarrow{\$} \{0,1\}^\lambda$.
- For every $i \in [\gamma]$, compute $\mathsf{ipCT}_i \leftarrow \mathsf{ipFE.Enc}(\mathsf{ipPK}_i, (\mathsf{PRF}(K_1,s_i),\ldots,\mathsf{PRF}(K_N,s_i)))$.
- For every $u \in [N]$, compute $ct_u \leftarrow \mathsf{1FE.Enc}(\mathsf{pk}_u, K_u)$.
- Output $\mathsf{CT} = \left(\{ct_u\}_{u\in[N]}, \{\mathsf{ipCT}_i\}_{i\in[\gamma]}\right)$.

$\underline{\mathsf{Dec}(\mathsf{sk}_T, \mathsf{CT})}$:

- Parse sk_T as $\left(\hat{T}, \Delta_T, \{\mathsf{ipSK}_k\}_{k\in[L_{max}]}, \{\mathsf{sk}_d\}_{d\in\Delta_T}\right)$ and CT as $\left(\{ct_u\}_{u\in[N]}, \{\mathsf{ipCT}_i\}_{i\in[\gamma]}\right)$.
- For every $d \in \Delta_T$, compute $(w_{d,1},\ldots,w_{d,\mu}) \leftarrow \mathsf{1FE.Dec}\ (\mathsf{sk}_d, ct_d)$.
- For every $i \in [\gamma]$, compute $x_i \leftarrow \mathsf{ipFE.Dec}(\mathsf{ipSK}_i, \mathsf{ipCT}_i)$ and set $P' = \{x_i \mid \forall\ i \in [\gamma]\}$.
- For every $j \in [\mu]$ compute $\hat{P}_j = g^{\sum_{d\in\Delta_T} w_{d,j}}$ and set $\hat{P} = \{\hat{P}_j | \forall\ j \in [\mu]\}$, where g is obtained from the master public key of ipFE.

- Construct $\bar{P} = \left\{ j \mid \forall\ i \in [\gamma], j \in [\mu], \exists\ P_i^{'} = \hat{P}_j \right\}$ by pairwise comparision of group elements.

- Output $\{t_j \mid j \in \bar{P}, (j, t_j) \in \hat{T}\}$.

□

7.2 Correctness

We show that the public key FE scheme described above will output $T \cap S$ with overwhelming probability. Consider the set associated with input as S of length γ and the set associated with the function query f_T as T of length μ such that $f_T(S) = T \cap S$. We parse the ciphertext $\mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{PK}, S)$ and the secret key $\mathsf{sk}_T \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, T)$ as follows

$$\mathsf{sk}_T = \left(\hat{T}, \Delta_T, \{\mathsf{ipFE.KeyGen}(\mathsf{ipMSK}, e^{\Delta_T})\}_{k \in [L_{max}]}, \{\mathsf{1FE.KeyGen}(\mathsf{msk}_d, c)\}_{d \in \Delta_T}\right)$$
$$\mathsf{CT} = \left(\{\mathsf{1FE.Enc}(\mathsf{pk}_u, K_u)\}_{u \in [N]}, \{\mathsf{ipFE.Enc}(\mathsf{ipPK_i}, (\mathsf{PRF}(K_u, s_i))_{u \in [N]})\}_{i \in [\gamma]}\right)$$

From the correctness of 1FE, we see that for every $d \in \Delta_T$, $j \in [\mu]$, $w_{d,j} = \mathsf{PRF}(K_d, t_j)$. Hence, we have

$$\hat{P}_j = g^{\sum_{d \in \Delta_T} \mathsf{PRF}(K_d, t_j)}$$

From the correctness of ipFE, we have

$$x_j = g^{\sum_{d \in \Delta_T} \mathsf{PRF}(K_d, s_j)}$$
$$\implies \bar{P} = \left\{ i \mid i \in [\gamma], j \in [\mu], g^{\sum_{d \in \Delta_T} \mathsf{PRF}(K_d, t_j)} = g^{\sum_{d \in \Delta_T} \mathsf{PRF}(K_d, s_i)} \right\}$$

From the security of PRF scheme, as with only negligible probability, the PRF evaluation

of two values will be same, and from the correctness of hashing scheme, we have

$$\bar{P} = \{j \mid i \in [\gamma], j \in [\mu], t_j = s_i\}$$

Hence, the output of the $\mathsf{Dec}(\mathsf{sk}_T, \mathsf{CT})$ algorithm is $T \cap S$ with overwhelming probability. Next, we state and prove the small-pairwise intersection lemma which we will use in the security proof of our scheme. Our version of small-pairwise intersection lemma follows a similar overlay of [GVW12] version but optimizes the parameters by eliminating a factor of Q^2 from N.

Lemma 10 (Small-Pairwise Intersection)**.** *Let $Q \in \mathbb{Z}_{>0}$ be the query bound and λ be the security parameter. Set $t = \Theta(\lambda), n = \Theta(t)$, and $N = \Theta(Q^2\lambda)$. For every $q \in [Q]$, sample $\Delta_q \subset [N]$ of size n. The following holds with overwhelming probability in λ:*

$$\Pr\left[\left|\bigcup_{q_1 \neq q_2} \Delta_{q_1} \cap \Delta_{q_2}\right| \leq t\right] \geq 1 - 2^{-\Theta(\lambda)}$$

where the probability is taken over the randomness of $\Delta_1, \ldots, \Delta_Q$.

Proof: For every q_1, q_2 let $X_{q_1 q_2}$ be a random variable such that $X_{q_1 q_2} = |\Delta_{q_1} \cap \Delta_{q_2}|$, and $X = \Sigma_{q_1 \neq q_2} X_{q_1 q_2}$. Let $X_{q_1 q_2} = 0$ for $q_1 = q_2$. From the linearity of expectation, we have $\mathbb{E}[X] = \Sigma_{q_1 \neq q_2} \mathbb{E}\left[X_{q_1 q_2}\right]$. With a fixed q_1, the maximum value $|\Delta_{q_1} \cap \Delta_{q_2}|$ can take is $|\Delta_{q_1}| = n$. We are randomly choosing $|\Delta_{q_2}|$ samples from $[N]$ and checking for success. This forms a hypergeometric distribution as sets don't have duplicate elements, hence it can be thought of choosing without replacement, with number of trials and maximum number of successes as n. Hence,

$$\begin{aligned}\epsilon = \mathbb{E}[X] &= \Sigma_{q_1 \neq q_2} \mathbb{E}\left[X_{q_1 q_2}\right] \\ &= \frac{Q(Q-1)n^2}{N} \quad \text{(We consider duplicates too)}\end{aligned}$$

Consider n random variables X_d for $d \in [n]$, such that $X_d = 1$ if the d-th element of Δ_{q_1} is present in $\Delta_{q_1} \cap \Delta_{q_2}$. Then, we can invoke Chernoff's bound as $X = \Sigma_{d=1}^{n} X_d$. By setting $t = c\lambda, n = kt = kc\lambda$ for some constants $k, c \in \mathbb{R}^+$, and $N = kcQ(Q-1)\lambda$, we have $\mu = ck\lambda = kt$. For sufficently large k, any real $\sigma \geq 0$:

$$\Pr\left[X > (1+\sigma)\epsilon\right] < exp\left(\frac{-\sigma^2\epsilon}{2+\sigma}\right)$$

$$\implies \Pr\left[X > t\right] < 2^{-\Theta(\lambda)}$$

$$\implies \Pr\left[\left|\bigcup_{q_1 \neq q_2} \Delta_{q_1} \cap \Delta_{q_2}\right| \leq t\right] = 1 - \Pr\left[X > t\right] \geq 1 - 2^{-\Theta(\lambda)}$$

□

7.3 Efficiency

In the FE for SI scheme, using $N = \Theta(Q^2\lambda), n = \Theta(\lambda)$, and $t = \Theta(\lambda)$ in accordance with the small pair-wise intersection lemma,

- **Setup:** We invoke N instantiations of 1FE and L_{max} instantiations of ipFE. The output contains $L_{max} \cdot (3N + 2)$ group elements and $4N \cdot poly(\lambda, L_{max})$ public-key encryption system keys.

- **KeyGen:** We perform n evaluations of 1FE.KeyGen algorithm and L_{max} evaluations of ipFE.KeyGen algorithm. We also sample a random subset of size n from $[N]$. The output contains $2L_{max}$ group elements and $n \cdot poly(\lambda, L_{max})$ public-key encryption system keys.

- **Enc:** We need at most $N \cdot L_{max}$ PRF evaluations and at most L_{max} executions of ipFE.Enc. We also require N executions of 1FE.Enc. We also sample N random

strings from $\{0,1\}^\lambda$. The output contains $4N \cdot \psi$ public-key encryption ciphertexts and at most $L_{max} \cdot (2N + 2)$ group elements.

- **Dec:** We perform n executions of 1FE.Dec algorithm and at most L_{max} executions of ipFE.Dec. We also perform at most L_{max} group exponentiations. It takes $O(|S| \cdot |T|)$ time to find the set intersection using pairwise comparisons followed by $O(|S \cap T|)$ time to generate the output.

7.4 Security

We prove the adaptive security of PubKeySI scheme using a simulator. Similar to the operations of the simulator for PrivKeySI, simulator for PubKeySI poses as a challenger interacting with the adversary. The simulator does not posses the master secret key generated by the setup and has to respond to adversary's query such that any PPT adversary cannot distinguish between the real challenger's responses and simulator's responses. Simulator for public-key encryption scheme simulates the three main components namely, 1FE, ipFE and PRF. We use the simulators $\mathsf{sim}_{\mathsf{1FE}}$ and $\mathsf{sim}_{\mathsf{ipFE}}$ to simulate the responses for 1FE and ipFE respectively. We sample uniform random strings to replace the output of PRF. Also, the simulator does not have direct access to the challenge ciphertext queries. Instead, it has access to the length of the challenge ciphertext and pairwise intersection of the challenger ciphertext query set and every functional query set. Using the adaptive security of 1FE, ipFE, and the security of PRF, we describe the simulator as follows:

Simulator

1. Sim receives the query bound Q and the maximum set size L_{max} from $\mathcal{A}$.

2. Sample Q sets as follows: for every $q \in [Q]$, sample a uniformly random set $\Delta_q \subset$

$[N]$ of size n. Construct $\Delta_{Corr} = \bigcup_{q_1 \neq q_2}(\Delta_{q_1} \cap \Delta_{q_2})$, where Δ_{Corr} denotes the set of corrupted instantiations. If $|\Delta_{Corr}| > t$, abort and output $\perp$.

3. For every $u \in [N]$, sample $K_u \xleftarrow{\$} \{0,1\}^\lambda$ uniformly at random.

4. Sim computes the master public key PK as follows: for every $u \in [N]$,

   - If $u \in \Delta_{Corr}$: $(\mathsf{pk}_u, \mathsf{msk}_u) \leftarrow \mathsf{1FE.Setup}(1^\lambda, 1^\psi)$
   - Otherwise, $\mathsf{pk}_u \leftarrow \mathsf{sim}_{\mathsf{1FE}}^{(u)}.\mathsf{Setup}(1^\lambda, 1^\psi)$.

   For every $k \in [L_{max}]$, $(\mathsf{ipPK}_k, \mathsf{ipMSK}_k) \leftarrow \mathsf{sim}_{\mathsf{ipFE}}^{(k)}(1^m, 1^N)$. Sim sets $\mathsf{PK} = (\mathsf{ipPK}, \ldots, \mathsf{ipPK}_{L_{max}}, \mathsf{pk}_1, \ldots, \mathsf{pk}_N)$ and sends it to $\mathcal{A}$.

5. Maintain a PSet and CSet across all functional key queries and challenge set queries such that it contains the set elements corresponding query responses. Initially PSet and CSet are empty.

6. $\mathcal{A}$ makes Q_1 functional set queries $T^{(1)}, \ldots, T^{(Q_1)}$ to Sim. For every $q \in [Q_1]$, let $T^{(q)} = \left\{t_1^{(q)}, \ldots, t_{\mu_q}^{(q)}\right\}$, and $\mu_q \leq L_{max}$ be the length of the q-th functional set. Sim computes functional key for $T^{(q)}$ as follows:

   - Set $\hat{T}^{(q)} \leftarrow \left(\left(1, t_1^{(q)}\right), \ldots, \left(\mu_q, t_{\mu_q}^{(q)}\right)\right)$.
   - For every $j \in [\mu_q], d \in \Delta_q$,
     - If $d \in \Delta_{Corr} \cap \Delta_q$, if $\left(t_j^{(q)}, K_d, \cdot\right) \notin \mathsf{PSet}$: set $\rho_d^{t_j^{(q)}} = \mathsf{PRF}\left(K_d, t_j^{(q)}\right)$.
     - Otherwise if $d \in \Delta_q \setminus \Delta_{Corr}$, if $\left(t_j^{(q)}, K_d, \cdot\right) \notin \mathsf{PSet}$: sample $\rho_d^{t_j^{(q)}} \xleftarrow{\$} \{0,1\}^m$.

     Add $\left(t_j^{(q)}, K_d, \rho_d^{t_j^{(q)}}\right)$ to PSet.
   - Create a functional circuit C_q that takes $K \in \{0,1\}^\lambda$ as input and outputs $\left(\mathsf{PRF}\left(K, t_1^{(q)}\right), \ldots, \mathsf{PRF}\left(K, t_{\mu_q}^{(q)}\right)\right)$.
   - Construct an N length vector e^{Δ_q} where $e_d^{\Delta_q} = 1$ if $d \in \Delta_q$ and 0 otherwise.

- For every $k \in [L_{max}]$, compute the simulated functional key for e^{Δ_q}. That is $\mathsf{ipSK}_k^{(q)} \leftarrow \mathsf{sim}_{\mathsf{ipFE}}^{(k)}(\mathsf{ipMSK}_k, e^{\Delta_q})$.
- For every $d \in \Delta_q$,
  - If $d \in \Delta_{Corr} \cap \Delta_q$: compute $\mathsf{sk}_d^{(q)} \leftarrow \mathsf{1FE.KeyGen}(\mathsf{msk}_d, C_q)$
  - Otherwise, if $d \in \Delta_q \setminus \Delta_{Corr}$: compute the functional secret key for C_q. That is, $\mathsf{sk}_d^{(q)} \leftarrow \mathsf{sim}_{\mathsf{1FE}}^{(d)}(C_q)$.

Set $\mathsf{sk}_{T^{(q)}} = \left(\hat{T}_q, \Delta_q, \left\{\mathsf{ipSK}_z^{(q)}\right\}_{z\in[L_{max}]} \left\{\mathsf{sk}_d^{(q)}\right\}_{d\in\Delta_q}\right)$. Send $\mathsf{sk}_{T^{(q)}}$ to $\mathcal{A}$.

7. $\mathcal{A}$ submits the challenge set $S = \{s_1, \ldots, s_\gamma\}$. Sim takes $\left\{1^\gamma, \left(T^{(1)}, S \cap T^{(1)}\right), \ldots, \left(T^{(Q_1)}, S \cap T^{(Q_1)}\right)\right\}$ as input and does the following:
   - Let $\{s_1, \ldots, s_\phi\} = S \cap \left(T^{(1)} \cup \ldots \cup T^{(Q_1)}\right)$
   - For every $i \in [\phi], u \in [N]$, note that $(s_i, K_u, \cdot) \in \mathsf{PSet}$. Retrieve $(s_i, K_u, \rho_u^{s_i})$ from PSet.
   - For every $i \in [\gamma - \phi], u \in [N]$, add $(\eta_i, K_u, \rho_u^{\eta_i})$ to PSet, where $\eta_i \xleftarrow{\$} \{0,1\}^l$ and $\rho_u^{\eta_i} \xleftarrow{\$} \{0,1\}^m$.
   - Generate a random permutation of the set $\left\{s_1, \ldots, s_\phi, \eta_1, \ldots, \eta_{\gamma-\phi}\right\}$ to get $\{\varsigma_1, \ldots, \varsigma_\gamma\}$.
   - For every $i \in [\gamma]$, let $\mathcal{V}_i = \left\{\left(e^{\Delta_q}, \left\langle (\rho_u^{\varsigma_i})_{u\in[N]}, e^{\Delta_q}\right\rangle, \mathsf{ipSK}_i^{(q)}\right) \mid \forall\ q \in [Q_1]\right\}$. $\mathsf{ipCT}_i \leftarrow \mathsf{sim}_{\mathsf{ipFE}}^{(z)}\left(\mathsf{ipPK}_i, \mathsf{ipMSK}_i, \mathcal{V}_i, \left\{1^{|\rho_1^{\varsigma_i}|}, \ldots, 1^{|\rho_1^{\varsigma_i}|}\right\}\right)$, where g_i is obtained from ipPK_i. Store the state returned by the simulator separately in $\mathsf{ipFE.st}_i$.
   - For every $u \in [N]$,
     - If $u \in \Delta_{Corr}$: $\mathsf{CT}_u \leftarrow \mathsf{1FE.Enc}(\mathsf{pk}_u, K_u)$.
     - Otherwise if $u \notin \Delta_{Corr}$: for every $q \in [Q_1]$, let $\hat{y}_u^{(q)} = \left(\rho_u^{t_1^q}, \ldots, \rho_u^{t_{\mu_q}^{(q)}}\right)$ and $V_u = \left\{\hat{y}_u^{(q)} : \forall\ q \in [Q_1]\right\}$. Compute the simulated ciphertext

$$\mathsf{CT}_u \leftarrow \mathsf{sim}_{\mathsf{1FE}}^{(u)}\left(1^\lambda, V_u\right).$$

Set $\mathsf{CT} = (\mathsf{CT}_1, \ldots, \mathsf{CT}_N, \mathsf{ipCT}_1, \ldots, \mathsf{ipCT}_\gamma)$. Send CT to $\mathcal{A}$.

8. $\mathcal{A}$ makes $Q - Q_1$ functional set queries $T^{(Q_1+1)}, \ldots, T^{(Q)}$ to Sim. For every $q \in \{Q_1+1, \ldots, Q\}$, let $T^{(q)} = \left\{t_1^{(q)}, \ldots, t_{\mu_q}^{(q)}\right\}$, and $\mu_q \leq L_{max}$ be the length of the q-th functional set. Sim takes $\left\{T^{(q)}, S \cap T^{(q)}\right\}$ as input and computes functional key for $T^{(q)}$ as follows:

   - Set $\hat{T}^{(q)} \leftarrow \left(\left(1, t_1^{(q)}\right), \ldots, \left(\mu_q, t_{\mu_q}^{(q)}\right)\right)$.
   - For every $j \in [\mu_q], d \in \Delta_q$,
     - If $\left(t_j^{(q)}, \cdot, \cdot\right) \in \mathsf{PSet}$, Retrieve $\left(t_j^{(q)}, K_d, \rho_d^{t_j^{(q)}}\right)$ from PSet.
     - Otherwise, if $\left(t_j^{(q)}, \cdot, \cdot\right) \notin \mathsf{PSet}$ and $t_j^{(q)} \notin S \cap T^{(q)}$,
       - If $d \in \Delta_{Corr} \cap \Delta_q$: set $\rho_d^{t_j^{(q)}} = \mathsf{PRF}\left(K_d, t_j^{(q)}\right)$.
       - Otherwise if $d \in \Delta_q \setminus \Delta_{Corr}$: sample $\rho_d^{t_j^{(q)}} \xleftarrow{\$} \{0,1\}^m$.

       Add $\left(t_j^{(q)}, K_d, \rho_d^{t_j^{(q)}}\right)$ to PSet.
     - Otherwise, if $\left(t_j^{(q)}, \cdot, \cdot\right) \notin \mathsf{PSet}$ and $t_j^{(q)} \in S \cap T^{(q)}$,
       - If $\left(t_j^{(q)}, \cdot\right) \in \mathsf{CSet}$, retrieve (χ, K_d, ρ_d^χ).
       - Otherwise, sample $\chi \xleftarrow{\$} \{\eta_1, \ldots, \eta_{\gamma-\phi}\}$ such that $(\cdot, \chi) \notin \mathsf{CSet}$. Add $\left(t_j^{(q)}, \chi\right)$ to CSet.

       Select $d^* \xleftarrow{\$} \Delta_q \setminus \Delta_{Corr}$ and set $\rho_{d^*}^{t_j^{(q)}} = \left(\sum_{d \in \Delta_q} \rho_d^\chi - \sum_{d \in \Delta_q, d \neq d^*} \mathsf{PRF}\left(K_d, t_j^{(q)}\right)\right)$ mod p. For other $d \in \Delta_q$, $d \neq d^*$, set $\rho_d^{t_j^{(q)}} = \mathsf{PRF}\left(K_d, t_j^{(q)}\right)$.
   - Create a functional circuit C_q that takes $K \in \{0,1\}^\lambda$ as input and outputs $\left(\mathsf{PRF}\left(K, t_1^{(q)}\right), \ldots, \mathsf{PRF}\left(K, t_{\mu_q}^{(q)}\right)\right)$.
   - Construct an N length vector e^{Δ_q} where $e_d^{\Delta_q} = 1$ if $d \in \Delta_q$ and 0 otherwise.

- For every $i \in [\gamma]$, compute the simulated functional key for e^{Δ_q}. That is $\mathsf{ipSK}_i^{(q)} \leftarrow \mathsf{sim}_{\mathsf{ipFE}}^{(i)}\left(\mathsf{ipMSK}_i,\, e^{\Delta_q}, \left\langle \left(\rho_u^{\varsigma_i}\right)_{u\in[N]}, e^{\Delta_q} \right\rangle, \mathsf{ipFE.st}_i\right)$.
- For every $d \in \Delta_q$,
  - If $d \in \Delta_{Corr} \cap \Delta_q$: compute $\mathsf{sk}_d^{(q)} \leftarrow$ 1FE. $\mathsf{KeyGen}(\mathsf{msk}_d, C_q)$
  - Otherwise, if $d \in \Delta_q \setminus \Delta_{Corr}$: compute the functional secret key for C_q. That is, $\mathsf{sk}_d^{(q)} \leftarrow \mathsf{sim}_{\mathsf{1FE}}^{(d)}(C_q, \hat{y}_d^{(q)})$ where $\hat{y}_d^{(q)} = \left(\rho_d^{t_1^{(q)}}, \ldots, \rho_d^{t_{\mu_q}^{(q)}}\right)$.

Set $\mathsf{sk}_{T^{(q)}} = \left(\hat{T}^{(q)}, \Delta_q, \left\{\mathsf{ipSK}_j^{(q)}\right\}_{j\in[L_{max}]} \left\{\mathsf{sk}_d^{(q)}\right\}_{d\in\Delta_q}\right)$. Send $\mathsf{sk}_{T^{(q)}}$ to $\mathcal{A}$.

9. $\mathcal{A}$ outputs b. Output b.

We show the computational indistinguishability between the real challenger and the simulator using a series of hybrids as mentioned below.

$\underline{\text{Hyb}_0}$: Same as Expt0 from 5.3. This hybrid consists of the following steps:

1. Ch receives the query bound Q and the maximum set size L_{max} from $\mathcal{A}$.
2. Ch computes the master public key PK as follows: for every $u \in [N]$, Ch generates $(\mathsf{pk}_u, \mathsf{msk}_u) \leftarrow$ 1FE. $\mathsf{Setup}(1^\lambda, 1^\psi)$. For every $k \in [L_{max}]$, Ch generates $(\mathsf{ipPK}_k, \mathsf{ipMSK}_k) \leftarrow \mathsf{ipFE.Setup}(1^m, 1^N)$. Ch sets $\mathsf{PK} = (\mathsf{ipPK}, \ldots, \mathsf{ipPK}_{L_{max}}, \mathsf{pk}_1, \ldots, \mathsf{pk}_N)$ and sends it to $\mathcal{A}$.
3. $\mathcal{A}$ makes Q_1 functional set queries $T^{(1)}, \ldots, T^{(Q_1)}$ to Ch. For every $q \in [Q_1]$, let $T^{(q)} = \left\{t_1^{(q)}, \ldots, t_{\mu_q}^{(q)}\right\}$, and $\mu_q \leq L_{max}$ be the length of the q-th functional set. Ch computes functional key for $T^{(q)}$ as follows:
   - Set $\hat{T^{(q)}} \leftarrow \left(\left(1, T_1^{(q)}\right), \ldots, \left(\mu_q, T_{\mu_q}^{(q)}\right)\right)$.
   - Sample $\Delta_q \subset [N]$ of size n uniformly at random.

- Create a functional circuit C_q that takes K as input and outputs $\left(\mathsf{PRF}\left(K, t_1^{(q)}\right), \ldots, \mathsf{PRF}\left(K, t_{\mu^{(q)}}^{(q)}\right)\right)$
- Construct an N length vector e^{Δ_q} where $e_d^{\Delta_q} = 1$ if $d \in \Delta_q$ and 0 otherwise.
- For every $k \in [L_{max}]$, compute $\mathsf{ipSK}_k^{(q)} \leftarrow \mathsf{ipFE.KeyGen}(\mathsf{ipMSK}_k, e^{\Delta_q})$.
- For every $d \in \Delta_q$, compute $\mathsf{sk}_d^{(q)} \leftarrow \mathsf{1FE.KeyGen}\ (\mathsf{msk}_d, C_q)$

Set $\mathsf{sk}_{T^{(q)}} = \left(\hat{T}^{(q)}, \Delta_q, \{\mathsf{ipSK}_k^{(q)}\}_{k\in[L_{max}]}\{\mathsf{sk}_d^{(q)}\}_{d\in\Delta_q}\right)$. Send $\mathsf{sk}_{T^{(q)}}$ to $\mathcal{A}$.

4. $\mathcal{A}$ submits the challenge set $S = \{s_1, \ldots, s_\gamma\}$. Ch does the following:

   - For every $u \in [N]$, sample $K_u \xleftarrow{\$} \{0,1\}^\lambda$ uniformly at random.
   - For every $i \in [\gamma]$, compute $\mathsf{ipCT}_i \leftarrow \mathsf{ipFE.Enc}(\mathsf{ipPK}_i, (\mathsf{PRF}(K_1, s_i), \ldots, \mathsf{PRF}(K_N, s_i)))$.
   - For every $u \in [N], \mathsf{CT}_u \leftarrow \mathsf{1FE.Enc}\,(\mathsf{pk}_u, K_u)$

   Set $\mathsf{CT} = (\mathsf{CT}_1, \ldots, \mathsf{CT}_N, \mathsf{ipCT}_1, \ldots, \mathsf{ipCT}_\gamma)$. Send CT to $\mathcal{A}$.

5. $\mathcal{A}$ makes $Q - Q_1$ functional set queries $T^{(Q_1+1)}, \ldots, T^{(Q)}$ to Ch. For every $q \in \{Q_1+1, \ldots, Q\}$, let $T^{(q)} = \left\{t_1^{(q)}, \ldots, t_{\mu_q}^{(q)}\right\}$, and $\mu_q \leq L_{max}$ be the length of the q-th functional set. Ch computes functional key for $T^{(q)}$ as follows:

   - Set $\hat{T}^{(q)} \leftarrow \left(\left(1, T_1^{(q)}\right), \ldots, \left(\mu_q, T_{\mu_q}^{(q)}\right)\right)$.
   - Sample $\Delta_q \subset [N]$ of size n uniformly at random.
   - Create a functional circuit C_q that takes K as input and outputs $\left(\mathsf{PRF}\left(K, t_1^{(q)}\right), \ldots, \mathsf{PRF}\left(K, t_{\mu_q}^{(q)}\right)\right)$
   - Construct an N length vector e^{Δ_q} where $e_d^{\Delta_q} = 1$ if $d \in \Delta_q$ and 0 otherwise.
   - For every $i \in [\gamma]$, compute $\mathsf{ipSK}_i^{(q)} \leftarrow \mathsf{ipFE.KeyGen}(\mathsf{ipMSK}_k, e^{\Delta_q})$.

- For every $d \in \Delta_q$, compute $\mathsf{sk}_d^{(q)} \leftarrow \mathsf{1FE.KeyGen}(\mathsf{msk}_d, C_q)$

Set $\mathsf{sk}_{T^{(q)}} = \left(\hat{T}^{(q)}, \Delta_q, \left\{\mathsf{ipSK}_i^{(q)}\right\}_{i\in[\gamma]} \left\{\mathsf{sk}_d^{(q)}\right\}_{d\in\Delta_q}\right)$. Send $\mathsf{sk}_{T^{(q)}}$ to $\mathcal{A}$.

6. $\mathcal{A}$ outputs b. Output b.

$\underline{\text{Hyb}_1}$: In this hybrid, we will sample Δ_q uniformly at random in the beginning. The changes are marked in red. The rest of the steps remain the same as Hyb_0. The output distribution will remain identical.

2. Sample Q sets as follows: for every $q \in [Q]$, sample a uniformly random set $\Delta_q \subset [N]$ of size n.

$\underline{\text{Hyb}_2}$: We will find the corrupted instances of 1FE early in the experiment using $|\Delta_{Corr}| = |\cup_{q_1\neq q_2}\Delta_{q_1} \cap \Delta_{q_2}|$. If $|\Delta_{Corr} > t$, abort. The rest of the steps remain the same as Hyb_0. It follows from small-pairwise intersection lemma that the statistical distance between Hyb_1 and Hyb_2 is negligible.

2. Sample Q sets as follows: for every $q \in [Q]$, sample a uniformly random set $\Delta_q \subset [N]$ of size n. Construct $\Delta_{Corr} = \bigcup_{q_1\neq q_2}(\Delta_{q_1} \cap \Delta_{q_2})$. If $|\Delta_{Corr}| > t$, abort and output $\perp$.

$\underline{\text{Hyb}_{3,\ell} \text{ for } \ell \in [N]}$: The first ℓ-1 non-corrupted instantiations of 1FE are simulated while the rest of the steps remain as in Hyb_2.

1. Ch receives the query bound Q and the maximum set size L_{max} from $\mathcal{A}$.

2. Sample Q sets as follows: for every $q \in [Q]$, sample a uniformly random set $\Delta_q \subset [N]$ of size n. Construct $\Delta_{Corr} = \bigcup_{q_1\neq q_2}(\Delta_{q_1} \cap \Delta_{q_2})$. If $|\Delta_{Corr}| > t$, abort and output $\perp$.

3. Ch computes the master public key PK as follows: for every $u \in [N]$,

- If $u \geq \ell$ or $u \in \Delta_{Corr}$: $(\mathsf{pk}_u, \mathsf{msk}_u) \leftarrow \mathsf{1FE}.\mathsf{Setup}(1^\lambda, 1^\psi)$
- Otherwise, if $u < \ell$ and $u \notin \Delta_{Corr}$, $\mathsf{pk}_u \leftarrow \mathsf{sim}_{\mathsf{1FE}}^{(u)}.\mathsf{Setup}(1^\lambda, 1^\psi)$.

Ch generates $\forall k \in [L_{max}](\mathsf{ipPK}_k, \mathsf{ipMSK}_k) \leftarrow \mathsf{ipFE}.\mathsf{Setup}(1^m, 1^N)$. Ch sets $\mathsf{PK} = (\mathsf{ipPK}, \ldots, \mathsf{ipPK}_{L_{max}}, \mathsf{pk}_1, \ldots, \mathsf{pk}_N)$ and sends it to $\mathcal{A}$.

4. $\mathcal{A}$ makes Q_1 functional set queries $T^{(1)}, \ldots, T^{(Q_1)}$ to Ch. For every $q \in [Q_1]$, let $T^{(q)} = \left\{t_1^{(q)}, \ldots, t_{\mu_q}^{(q)}\right\}$, and $\mu_q \leq L_{max}$ be the length of the q-th functional set. Ch computes functional key for $T^{(q)}$ as follows:
   - Set $\hat{T}^{(q)} \leftarrow \left(\left(1, T_1^{(q)}\right), \ldots, \left(\mu_q, T_{\mu_q}^{(q)}\right)\right)$.
   - Create a functional circuit C_q that takes K as input and outputs $\left(\mathsf{PRF}\left(K, t_1^{(q)}\right), \ldots, \mathsf{PRF}\left(K, t_{\mu_q}^{(q)}\right)\right)$
   - Construct an N length vector e^{Δ_q} where $e_d^{\Delta_q} = 1$ if $d \in \Delta_q$ and 0 otherwise.
   - For every $k \in [L_{max}]$, compute $\mathsf{ipSK}_k^{(q)} \leftarrow \mathsf{ipFE}.\mathsf{KeyGen}(\mathsf{ipMSK}_k, e^{\Delta_q})$.
   - For every $d \in \Delta_q$,
     - If $d \in \Delta_{Corr} \cap \Delta_q$ or $d \geq \ell$: compute $\mathsf{sk}_d^{(q)} \leftarrow \mathsf{1FE}.\mathsf{KeyGen}(\mathsf{msk}_d, C_q)$
     - Otherwise, if $d \in \Delta_q \setminus \Delta_{Corr}$ and $d < \ell$: compute the functional secret key for C_q. That is, $\mathsf{sk}_d^{(q)} \leftarrow \mathsf{sim}_{\mathsf{1FE}}^{(i)}(C_q)$.

   Set $\mathsf{sk}_{T^{(q)}} = \left(\hat{T}^{(q)}, \Delta_q, \left\{\mathsf{ipSK}_k^{(q)}\right\}_{k \in [L_{max}]} \left\{\mathsf{sk}_d^{(q)}\right\}_{d \in \Delta_q}\right)$. Send $\mathsf{sk}_{T^{(q)}}$ to $\mathcal{A}$.
5. $\mathcal{A}$ submits the challenge set $S = \{s_1, \ldots, s_\gamma\}$. Ch does the following:
   - For every $u \in [N]$, sample $K_u \xleftarrow{\$} \{0,1\}^\lambda$ uniformly at random.
   - For every $i \in [\gamma]$, compute $\mathsf{ipCT}_i \leftarrow \mathsf{ipFE}.\mathsf{Enc}(\mathsf{ipPK}_i, (\mathsf{PRF}(K_1, s_i), \ldots, \mathsf{PRF}(K_N, s_i)))$.

- For every $u \in [N]$,
  - If $u \in \Delta_{Corr}$ or $u \geq \ell$: $\mathsf{CT}_u \leftarrow \mathsf{1FE.Enc}\left(\mathsf{pk}_u, K_u\right)$.
  - Otherwise if $u \notin \Delta_{Corr}$ and $u < \ell$: for every $q \in [Q_1]$, let $\hat{y}_u^{(q)} = C_q(K_u)$ and $V_u = \left\{\hat{y}_u^{(q)} \mid \forall\ q \in [Q_1]\right\}$. Compute the simulated ciphertext $\mathsf{CT}_u \leftarrow \mathsf{sim}_{\mathsf{1FE}}^{(u)}(1^\lambda, V_u)$.

Set $\mathsf{CT} = (\mathsf{CT}_1, \ldots, \mathsf{CT}_N, \mathsf{ipCT}_1, \ldots, \mathsf{ipCT}_\gamma)$. Send CT to $\mathcal{A}$.

6. $\mathcal{A}$ makes $Q - Q_1$ functional set queries $T^{(Q_1+1)}, \ldots, T^{(Q)}$ to Ch. For every $q \in \{Q_1+1, \ldots, Q\}$, let $T^{(q)} = \left\{t_1^{(q)}, \ldots, t_{\mu_q}^{(q)}\right\}$, and $\mu_q \leq L_{max}$ be the length of the q-th functional set. Ch computes functional key for $T^{(q)}$ as follows:
   - Set $\hat{T^{(q)}} \leftarrow \left(\left(1, T_1^{(q)}\right), \ldots, \left(\mu_q, T_{\mu_q}^{(q)}\right)\right)$.
   - Create a functional circuit C_q that takes K as input and outputs $\left(\mathsf{PRF}\left(K, t_1^{(q)}\right), \ldots, \mathsf{PRF}\left(K, t_{\mu_q}^{(q)}\right)\right)$
   - Construct an N length vector e^{Δ_q} where $e_d^{\Delta_q} = 1$ if $d \in \Delta_q$ and 0 otherwise.
   - For every $i \in [\gamma]$, compute $\mathsf{ipSK}_i^{(q)} \leftarrow \mathsf{ipFE.KeyGen}\left(\mathsf{ipMSK}_i, e^{\Delta_q}\right)$.
   - For every $d \in \Delta_q$,
     - If $d \in \Delta_{Corr} \cap \Delta_q$ or $d \geq \ell$: compute $\mathsf{sk}_d^{(q)} \leftarrow \mathsf{1FE.KeyGen}(\mathsf{msk}_d, C_q)$
     - Otherwise, if $d \in \Delta_q \setminus \Delta_{Corr}$ and $i < \ell$: compute the functional secret key for C_q. That is, $\mathsf{sk}_d^{(q)} \leftarrow \mathsf{sim}_{\mathsf{1FE}}^{(d)}(C_q, \hat{y}_d^{(q)})$ where $\hat{y}_d^{(q)} = C_q(K_d)$.

   Set $\mathsf{sk}_{T^{(q)}} = \left(\hat{T}^{(q)}, \Delta_q, \left\{\mathsf{ipSK}_i^{(q)}\right\}_{i\in[\gamma]}, \left\{\mathsf{sk}_d^{(q)}\right\}_{d\in\Delta_q}\right)$. Send $\mathsf{sk}_{T^{(q)}}$ to $\mathcal{A}$.
7. $\mathcal{A}$ outputs b. Output b.

Claim 11. *Assuming the security of* 1FE*, for every $\ell \in [N-1]$, $Hyb_{3,\ell}$ and $Hyb_{3,\ell+1}$ are computationally indistinguishable.*

Proof: Note that Hyb_2 and $\text{Hyb}_{3,1}$ are identically distributed. If $\ell \in \Delta_{Corr}$, we are not simulating this instantiation. Hence, the output distribution of $\text{Hyb}_{3,\ell}$ will be exactly same as $\text{Hyb}_{3,\ell+1}$. WLOG, assume that $\ell \notin \Delta_{Corr}$ i.e, ℓ-th instantiation is not corrupted and is only queried once for secret key. Assuming that there exists an adversary $\mathcal{A}$ that can distinguish the hybrids $\text{Hyb}_{3,\ell}$ and $\text{Hyb}_{3,\ell+1}$, we will construct an adversary $\mathcal{B}$ that can break the adaptively secure property of the ℓ-th instantiation of $\mathsf{1FE}$. More formally, if $\mathcal{A}$'s advantage is such that

$$\left|\Pr\left[0 \leftarrow \mathcal{A}^{\mathsf{Hyb}_{3,\ell}}(1^\lambda)\right] - \Pr\left[0 \leftarrow \mathcal{A}^{\mathsf{Hyb}_{3,\ell+1}}(1^\lambda)\right]\right| > \mathsf{negl}(\lambda)$$

We will construct a polynomial time reduction $\mathcal{B}$ which distinguishes between oracle access to the honest challenger Ch and a simulator $\mathsf{sim}_{\mathsf{1FE}}$ for the $\mathsf{1FE}$ scheme. More specifically, we will construct an adversary $\mathcal{B}$ whose advantage in distinguishing the real and ideal experiments for $\mathsf{1FE}$ is non-negligible.

$$\left|\Pr\left[0 \leftarrow \mathcal{B}^{\mathsf{Ch}}(1^\lambda)\right] - \Pr\left[0 \leftarrow \mathcal{B}^{\mathsf{sim}^{(\ell)}_{\mathsf{1FE}}}(1^\lambda)\right]\right| > \mathsf{negl}(\lambda)$$

The description of $\mathcal{B}$ is as follows:

1. $\mathcal{A}$ sends Q, L_{max} to $\mathcal{B}$. $\mathcal{B}$ computes ψ based on L_{max} and λ and sends s to oracle $\mathcal{O}$.

2. $\mathcal{B}$ computes $N = \Theta(Q^2\lambda), t = \Theta(\lambda)$, and $n = \Theta(t)$ such that $t < n$. It also computes $\Delta_q \subseteq [N]$ of size n for each $q \in [Q]$ and $\Delta_{Corr} = \cup_{q_1 \neq q_2}(\Delta_{q_1} \cap \Delta_{q_2})$. $\mathcal{B}$ aborts if $|\Delta_{Corr}| > t$ and outputs $\perp$.

3. $\mathcal{B}$ computes master public key as follows: for every $u \in [N]$:

   - If $u \in \Delta_{Corr}$ or $u > \ell$: $(\mathsf{pk}_u, \mathsf{msk}_u) \leftarrow \mathsf{1FE.Setup}\ (1^\lambda, 1^\psi)$.

- Otherwise, if $u = \ell$, set pk_u = output received from $\mathcal{O}$.
- Otherwise, if $u \notin \Delta_{Corr}$ and $u < \ell$: $\mathsf{pk}_u \leftarrow \mathsf{sim}_{\mathsf{1FE}}^{(u)}(1^\lambda, 1^\psi)$.

For every $k \in [L_{max}]$, $\mathcal{B}$ computes $(\mathsf{ipPK}_k, \mathsf{ipMSK}_k) \leftarrow \mathsf{ipFE.Setup}(1^m, 1^N)$. $\mathcal{B}$ sets $\mathsf{PK} \leftarrow \left(\mathsf{ipPK}_1, \ldots, \mathsf{ipPK}_{L_{max}}, \mathsf{pk}_1, \ldots, \mathsf{pk}_N\right)$ and sends it to $\mathcal{A}$.

4. Let Q_1 queries for functional secret keys be submitted. For every $q \in [Q_1]$, let $T^{(q)} = \left\{t_1^{(q)}, \ldots, t_{\mu_q}^{(q)}\right\}$ be the functional set associated with the query. $\mathcal{B}$ calculates $\hat{T}^{(q)}$, the functional circuit C_q, and the N-length vector e^{Δ_q}. For every $k \in [L_{max}]$, compute $\mathsf{ipSK}_k^{(q)} \leftarrow \mathsf{ipFE.KeyGen}\left(\mathsf{ipMSK}_k, e^{\Delta_q}\right)$. Then, for every $d \in \Delta_{Corr} \cap \Delta_q$ or $d > \ell$, $\mathcal{B}$ computes $\mathsf{sk}_d^{(q)} \leftarrow \mathsf{1FE.KeyGen}(\mathsf{msk}_d, C_q)$. Otherwise, if $d \in \Delta_q \setminus \Delta_{Corr}$ and $d < \ell$, $\mathcal{B}$ computes $\mathsf{sk}_d^{(q)} \leftarrow \mathsf{sim}_{\mathsf{1FE}}^{(d)}(C_q)$. If $d = \ell$, $sk_d^{(q)} \leftarrow \mathcal{O}(C_q)$. $\mathcal{B}$ sets $\mathsf{sk}_{T^{(q)}} \leftarrow \left(\hat{T}^{(q)}, \Delta_q, \left\{\mathsf{ipSK}_k^{(q)}\right\}_{k\in[L_{max}]}, \{\mathsf{sk}_d^{(q)}\}_{d\in\Delta_q}\right)$ and sends it to $\mathcal{A}$.

5. $\mathcal{A}$ submits the challenge set $S = \{s_1, \ldots, s_\gamma\}$. For each $u \in [N]$, $\mathcal{B}$ samples $K_u \xleftarrow{\$} \{0,1\}^\lambda$. For every $i \in [\gamma]$, compute $\mathsf{ipCT}_i \leftarrow \mathsf{ipFE.Enc}\Big(\mathsf{ipPK}_i, (\mathsf{PRF}(K_1, s_i), \ldots, \mathsf{PRF}(K_N, s_i))\Big)$. For each $u \in [N]$, $\mathcal{B}$ does the following:
   - If $u \in \Delta_{Corr}$ or $u > \ell$: $\mathsf{CT}_u \leftarrow \mathsf{1FE.Enc}(\mathsf{pk}_u, K_u)$.
   - Otherwise, if $u \notin \Delta_{Corr}$ and $u < \ell$: $\mathsf{CT}_u \leftarrow \mathsf{sim}_{\mathsf{1FE}}^{(u)}(1^\lambda, V_u)$ where $V_u = \left\{\hat{y}_u^{(q)} \mid \forall q \in [Q_1]\right\}$ and $\hat{y}_u^{(q)} = C_q(K_u)$.
   - Otherwise, if $u = \ell$, set $ct_u \leftarrow \mathcal{O}(K_u)$

   $\mathcal{B}$ sets $\mathsf{CT} = (\mathsf{CT}_1, \ldots, \mathsf{CT}_N, \mathsf{ipCT}_1, \ldots, \mathsf{ipCT}_\gamma)$ and sends it to $\mathcal{A}$.

6. Now $Q - Q_1$ queries for functional secret keys will be queried. For every $q \in \{Q_1 + 1, \ldots, Q\}$, let $T^{(q)} = \left\{t_1^{(q)}, \ldots, t_{\mu_q}^{(q)}\right\}$ be the functional set associated with the query. $\mathcal{B}$ calculates $\hat{T}^{(q)}$, the functional circuit C_q, and the N-length vector e^{Δ_q}. For every $i \in [\gamma]$, compute $\mathsf{ipSK}_i \leftarrow \mathsf{ipFE.KeyGen}\left(\mathsf{ipMSK}_i, e^{\Delta_q}\right)$. Then, for every

$d \in \Delta_{Corr} \cap \Delta_q$ or $d > \ell$, $\mathcal{B}$ computes $\mathsf{sk}_d^{(q)} \leftarrow \mathsf{1FE.KeyGen}(\mathsf{msk}_d, C_q)$. Otherwise, if $d \in \Delta_q \setminus \Delta_{Corr}$ and $d < \ell$, $\mathcal{B}$ computes $\mathsf{sk}_d^{(q)} \leftarrow \mathsf{sim}_{\mathsf{1FE}}^{(d)}(C_q, \hat{y}_d^{(q)})$ where $\hat{y}_d^{(q)} = C_q(K_d)$. If $d = \ell$, $sk_d^{(q)} \leftarrow \mathcal{O}(C_q)$. $\mathcal{B}$ sets $\mathsf{sk}_{T^{(q)}} \leftarrow \left(\hat{T}^{(q)}, \Delta_q, \left\{\mathsf{ipSK}_i^{(q)}\right\}_{i\in[\gamma]} \left\{\mathsf{sk}_d^{(q)}\right\}_{d\in\Delta_q}\right)$ and sends it to $\mathcal{A}$.

7. $\mathcal{A}$ outputs the bit b. $\mathcal{B}$ outputs b.

As we can see, $\mathcal{B}$ runs in polynomial time in the parameters for Q, λ, L_{max} as both Ch and sim for $\mathsf{1FE}$ (oracles) run in polynomial times too. If the oracle $\mathcal{O}$ is an honest challenger for $\mathsf{1FE}$, $\mathcal{B}$ behaves like $\mathrm{Hyb}_{3,\ell}$ and if $\mathcal{O}$ is sim for $\mathsf{1FE}$, $\mathcal{B}$ behaves like $\mathrm{Hyb}_{3,\ell+1}$. As $\mathcal{A}$ can distinguish between them with non-negligible advantage and the abort probability for $\mathcal{B}$ is negligible, we can see that $\mathcal{B}$ with non-negligible probability distinguished between an honest challenger and simulator for $\mathsf{1FE}$. This contradicts our assumption for adaptively secure $\mathsf{1FE}$. Hence, $\mathrm{Hyb}_{3,\ell}$ and $\mathrm{Hyb}_{3,\ell+1}$ are computationally indistinguishable. □

$\mathrm{Hyb}_{4,\ell}$ for $\ell \in [\gamma]$: We will simulate the first $\ell - 1$ ipFE instantiations while the rest are generated honestly in this experiment.

1. Ch receives the query bound Q and the maximum set size L_{max} from $\mathcal{A}$.

2. Sample Q sets as follows: for every $q \in [Q]$, sample a uniformly random set $\Delta_q \subset [N]$ of size n. Construct $\Delta_{Corr} = \bigcup_{q_1 \neq q_2}(\Delta_{q_1} \cap \Delta_{q_2})$. If $|\Delta_{Corr}| > t$, abort and output $\perp$.

3. Ch computes the master public key PK as follows: for every $u \in [N]$,

   - If $u \in \Delta_{Corr}$: $(\mathsf{pk}_u, \mathsf{msk}_u) \leftarrow \mathsf{1FE.Setup}(1^\lambda, 1^\psi)$
   - Otherwise, if $u \notin \Delta_{Corr}$, $\mathsf{pk}_u \leftarrow \mathsf{sim}_{\mathsf{1FE}}^{(u)}.\mathsf{Setup}\ (1^\lambda, 1^\psi)$.

   For every $k \in [L_{max}]$,

   - If $k \geq \ell$: $(\mathsf{ipPK}_k, \mathsf{ipMSK}_k) \leftarrow \mathsf{ipFE.Setup}(1^m, 1^N)$.

- Otherwise if $k < \ell$: $(\mathsf{ipPK}_k, \mathsf{ipMSK}_k) \leftarrow \mathsf{sim}_{\mathsf{ipFE}}^{(k)}(1^m, 1^N)$.

Ch sets $\mathsf{PK} = (\mathsf{ipPK}, \ldots, \mathsf{ipPK}_{L_{max}}, \mathsf{pk}_1, \ldots, \mathsf{pk}_N)$ and sends it to $\mathcal{A}$.

4. $\mathcal{A}$ makes Q_1 functional set queries $T^{(1)}, \ldots, T^{(Q_1)}$ to Ch. For every $q \in [Q_1]$, let $T^{(q)} = \left\{t_1^{(q)}, \ldots, t_{\mu_q}^{(q)}\right\}$, and $\mu_q \leq L_{max}$ be the length of the q-th functional set. Ch computes functional key for $T^{(q)}$ as follows:

   - Set $\hat{T}^{(q)} \leftarrow \left(\left(1, t_1^{(q)}\right), \ldots, \left(\mu^q, t_{\mu_q}^{(q)}\right)\right)$.
   - Create a functional circuit C_q that takes K as input and outputs $\left(\mathsf{PRF}\left(K, t_1^{(q)}\right), \ldots, \mathsf{PRF}\left(K, t_{\mu_q}^{(q)}\right)\right)$.
   - Construct an N length vector e^{Δ_q} where $e_d^{\Delta_q} = 1$ if $d \in \Delta_q$ and 0 otherwise.
   - For every $k \in [L_{max}]$,
     - If $k \geq \ell$: compute $\mathsf{ipSK}_k^{(q)} \leftarrow \mathsf{ipFE.KeyGen}\left(\mathsf{ipMSK}_k, e^{\Delta_q}\right)$.
     - Otherwise, if $k < \ell$: compute the simulated functional key for e^{Δ_q}. That is $\mathsf{ipSK}_k^{(q)} \leftarrow \mathsf{sim}_{\mathsf{ipFE}}^{(k)}\left(\mathsf{ipMSK}_k, e^{\Delta_q}\right)$.
   - For every $d \in \Delta_q$,
     - If $d \in \Delta_{Corr} \cap \Delta_q$: compute $\mathsf{sk}_d^{(q)} \leftarrow \mathsf{1FE.KeyGen}(\mathsf{msk}_d, C_q)$
     - Otherwise, if $d \in \Delta_q \setminus \Delta_{Corr}$: compute the functional secret key for C_q. That is, $\mathsf{sk}_d^{(q)} \leftarrow \mathsf{sim}_{\mathsf{1FE}}^{(d)}(C_q)$.

   Set $\mathsf{sk}_{T^{(q)}} = \left(\hat{T}^{(q)}, \Delta_q, \left\{\mathsf{ipSK}_k^{(q)}\right\}_{k \in [L_{max}]} \left\{\mathsf{sk}_d^{(q)}\right\}_{d \in \Delta_q}\right)$. Send $\mathsf{sk}_{T^{(q)}}$ to $\mathcal{A}$.

5. $\mathcal{A}$ submits the challenge set $S = \{s_1, \ldots, s_\gamma\}$. Ch does the following:

   - For every $u \in [N]$, sample $K_u \xleftarrow{\$} \{0,1\}^\lambda$ uniformly at random.
   - For every $i \in [\gamma]$,
     - If $i \geq \ell$: compute $\mathsf{ipCT}_i \leftarrow \mathsf{ipFE.Enc}(\mathsf{ipPK}_i, (\mathsf{PRF}(K_1, s_i), \ldots, \mathsf{PRF}(K_N, s_i)))$.

– Otherwise if $i < \ell$: Let $\mathcal{V}_i = \left\{ \left(e^{\Delta_q}, \left\langle \left(\mathsf{PRF}(K_u, s_i)\right)_{u\in[N]}, e^{\Delta_q} \right\rangle, \mathsf{ipSK}_i^{(q)} \right) \mid \forall\ q \in [Q_1] \right\}$. $\mathsf{ipCT}_i \leftarrow \mathsf{sim}_{\mathsf{ipFE}}^{(i)}\left(\mathsf{ipPK}_i, \mathsf{ipMSK}_i, \mathcal{V}_i, \left\{1^{|\mathsf{PRF}(K_1,s_i)|}, \ldots, 1^{|\mathsf{PRF}(K_N,s_i)|}\right\}\right)$. Store the state returned by the simulator separately in $\mathsf{ipFE.st}_i$.

- For every $u \in [N]$,
  - If $u \in \Delta_{Corr}$: $\mathsf{CT}_u \leftarrow \mathsf{1FE.Enc}\left(\mathsf{pk}_u, K_u\right)$.
  - Otherwise if $u \notin \Delta_{Corr}$: for every $q \in [Q_1]$, let $\hat{y}_u^{(q)} = C_q(K_u)$ and $V_u = \left\{ y_u^{(q)} : \forall\ q \in [Q_1] \right\}$. Compute the simulated ciphertext $\mathsf{CT}_u \leftarrow \mathsf{sim}_{\mathsf{1FE}}^{(u)}(1^\lambda, V_u)$.

Set $\mathsf{CT} = (\mathsf{CT}_1, \ldots, \mathsf{CT}_N, \mathsf{ipCT}_1, \ldots, \mathsf{ipCT}_\gamma)$. Send CT to $\mathcal{A}$.

6. $\mathcal{A}$ makes $Q - Q_1$ functional set queries $T^{(Q_1+1)}, \ldots, T^{(Q)}$ to Ch. For every $q \in \{Q_1+1, \ldots, Q\}$, let $T^{(q)} = \left\{ t_1^{(q)}, \ldots, t_{\mu_q}^{(q)} \right\}$, and $\mu_q \leq L_{max}$ be the length of the q-th functional set. Ch computes functional key for $T^{(q)}$ as follows:
   - Set $\hat{T}^{(q)} \leftarrow \left(\left(1, t_1^{(q)}\right), \ldots, \left(\mu_q, t_{\mu_q}^{(q)}\right) \right)$.
   - Create a functional circuit C_q that takes K as input and outputs $\left(\mathsf{PRF}\left(K, t_1^{(q)}\right), \ldots, \mathsf{PRF}\left(K, t_{\mu_q}^{(q)}\right) \right)$.
   - Construct an N length vector e^{Δ_q} where $e_d^{\Delta_q} = 1$ if $d \in \Delta_q$ and 0 otherwise.
   - For every $i \in [\gamma]$,
     - If $i \geq \ell$: compute $\mathsf{ipSK}_i^{(q)} \leftarrow \mathsf{ipFE.KeyGen}\left(\mathsf{ipMSK}_i, e^{\Delta_q}\right)$.
     - Otherwise, if $i < \ell$: compute the simulated functional key for e^{Δ_q}. That is $\mathsf{ipSK}_i^{(q)} \leftarrow \mathsf{sim}_{\mathsf{ipFE}}^{(i)}\left(\mathsf{ipMSK}_i, e^{\Delta_q}, \left\langle \left(\mathsf{PRF}(K_u, s_i)\right)_{u\in[N]}, e^{\Delta_q} \right\rangle, \mathsf{ipFE.st}_i\right)$.
   - For every $d \in \Delta_q$,

- If $d \in \Delta_{Corr} \cap \Delta_q$: compute $\mathsf{sk}_d^{(q)} \leftarrow \mathsf{1FE.KeyGen}(\mathsf{msk}_d, C_q)$
- Otherwise, if $d \in \Delta_q \setminus \Delta_{Corr}$: compute the functional secret key for C_q. That is, $\mathsf{sk}_d^{(q)} \leftarrow \mathsf{sim}_{\mathsf{1FE}}^{(d)}\left(C_q, \hat{y}_d^{(q)}\right)$ where $\hat{y}_d^{(q)} = C_q(K_d)$.

Set $\mathsf{sk}_{T^{(q)}} = \left(\hat{T}^{(q)}, \Delta_q, \left\{\mathsf{ipSK}_i^{(q)}\right\}_{i\in[\gamma]} \left\{\mathsf{sk}_d^{(q)}\right\}_{d\in\Delta_q}\right)$. Send $\mathsf{sk}_{T^{(q)}}$ to $\mathcal{A}$.

7. $\mathcal{A}$ outputs b. Output b.

Claim 12. *Assuming the security of* 1FE*, $Hyb_{3,N}$ and $Hyb_{4,1}$ are computationally indistinguishable.*

Proof: As we are only simulating the N-th instantiation of 1FE, proof is similar to claim 11. □

Claim 13. *Assuming the security of ipFE, for every $\ell \in [\gamma - 1]$, $Hyb_{4,\ell}$ and $Hyb_{4,\ell+1}$ are computationally indistinguishable.*

Proof: Assuming that there exists an adversary $\mathcal{A}$ who can distinguish between $\mathrm{Hyb}_{4,\ell}$ and $\mathrm{Hyb}_{4,\ell+1}$, we will construct a hybrid that can break the adaptively secure property of the ℓ-th instantiation of ipFE. More formally, if $\mathcal{A}$'s advantage is such that

$$\left|\Pr\left[0 \leftarrow \mathcal{A}^{\mathsf{Hyb}_{4,\ell}}(1^\lambda)\right] - \Pr\left[0 \leftarrow \mathcal{A}^{\mathsf{Hyb}_{4,\ell+1}}(1^\lambda)\right]\right| > \mathsf{negl}(\lambda)$$

We will construct a polynomial time reduction $\mathcal{B}$ which distinguishes between oracle access to the honest challenger Ch and a simulator $\mathsf{sim}_{\mathsf{ipFE}}^{(\ell)}$ for the ipFE scheme. More specifically, we will construct an adversary $\mathcal{B}$ whose advantage in distinguishing the real and ideal experiments for ipFE is non-negligible.

$$\left|\Pr\left[0 \leftarrow \mathcal{B}^{\mathsf{Ch}}(1^\lambda)\right] - \Pr\left[0 \leftarrow \mathcal{B}^{\mathsf{sim}_{\mathsf{ipFE}}^{(\ell)}}(1^\lambda)\right]\right| > \mathsf{negl}(\lambda)$$

The description of $\mathcal{B}$ is as follows:

1. $\mathcal{A}$ sends Q, L_{max} to $\mathcal{B}$. $\mathcal{B}$ calculates $m = poly(\lambda), t = \Theta(\lambda), n = \Theta(t)$, and $N = \Theta(Q^2 t)$. $\mathcal{B}$ sends m, N to the oracle $\mathcal{O}$.

2. $\mathcal{B}$ computes $\Delta_q \subseteq [N]$ of size n for each $q \in [Q]$ and $\Delta_{Corr} = \cup_{q_1 \neq q_2} \Delta_{q_1} \cap \Delta_{q_2}$. $\mathcal{B}$ aborts if $|\Delta_{Corr}| > t$ and outputs $\perp$.

3. $\mathcal{B}$ computes master public key as follows: for every $u \in [N]$:

   - If $u \in \Delta_{Corr}$: $(\mathsf{pk}_u, \mathsf{msk}_u) \leftarrow \mathsf{1FE.Setup}(1^\lambda, 1^\psi)$.
   - Otherwise, if $u \notin \Delta_{Corr}$: $\mathsf{pk}_u \leftarrow \mathsf{sim}^{(u)}_{\mathsf{1FE}}(1^\lambda, 1^\psi)$.

   For every $k \in [L_{max}]$, $\mathcal{B}$ computes

   - If $k > \ell$: $(\mathsf{ipPK}_k, \mathsf{ipMSK}_k) \leftarrow \mathsf{ipFE.Setup}(1^m, 1^N)$.
   - Otherwise if $k < \ell$: $(\mathsf{ipPK}_k, \mathsf{ipMSK}_k) \leftarrow \mathsf{sim}^{(k)}_{\mathsf{ipFE}}\ (1^m, 1^N)$.
   - Otherwise if $k = \ell$: Set ipPK_k as value returned by $\mathcal{O}$.

   $\mathcal{B}$ sets $\mathsf{PK} \leftarrow \left(\mathsf{ipPK}_1, \ldots, \mathsf{ipPK}_{L_{max}}, \mathsf{pk}_1, \ldots, \mathsf{pk}_N\right)$ and sends it to $\mathcal{A}$.

4. Let Q_1 queries for functional secret keys be submitted. For every $q \in [Q_1]$, let $T^{(q)} = \left\{t_1^{(q)}, \ldots, t_{\mu_q}^{(q)}\right\}$ be the functional set associated with the query. $\mathcal{B}$ calculates $\hat{T}^{(q)}$, the functional circuit C_q, and the N-length vector e^{Δ_q}. For every $k \in [L_{max}]$, if $k > \ell$, compute $\mathsf{ipSK}_k^{(q)} \leftarrow \mathsf{ipFE.KeyGen}\left(\mathsf{ipMSK}_k, e^{\Delta_q}\right)$. Otherwise if $k < \ell$, compute $\mathsf{ipSK}_k^{(q)} \leftarrow \mathsf{sim}^{(k)}_{\mathsf{ipFE}}\left(\mathsf{ipMSK}_z, e^{\Delta_q}\right)$. Otherwise, if $k = \ell$, set $\mathsf{ipSK}_k^{(q)} = \mathcal{O}(e^{\Delta_q})$. Then, for every $d \in \Delta_{Corr} \cap \Delta_q$, $\mathcal{B}$ computes $\mathsf{sk}_d^{(q)} \leftarrow \mathsf{1FE.KeyGen}(\mathsf{msk}_d, C_q)$. Otherwise, if $d \in \Delta_q \setminus \Delta_{Corr}$, $\mathcal{B}$ computes $\mathsf{sk}_d^{(q)} \leftarrow \mathsf{sim}^{(d)}_{\mathsf{1FE}}(C_q)$. $\mathcal{B}$ sets $\mathsf{sk}_{T^{(q)}} \leftarrow \left(\hat{T}^{(q)}, \Delta_q, \left\{\mathsf{ipSK}_k^{(q)}\right\}_{k \in [L_{max}]} \left\{\mathsf{sk}_d^{(q)}\right\}_{d \in \Delta_q}\right)$ and sends it to $\mathcal{A}$.

5. $\mathcal{A}$ submits the challenge set $S = \{s_1, \ldots, s_\gamma\}$. For each $u \in [N]$, $\mathcal{B}$ samples $K_u \xleftarrow{\$} \{0,1\}^\lambda$. For every $i \in [\gamma]$, compute

- If $i > \ell$: $\mathsf{ipCT}_i \leftarrow \mathsf{ipFE.Enc}\Big(\mathsf{ipPK}_i, \Big(\mathsf{PRF}(K_1, s_i), \ldots, \mathsf{PRF}(K_N, s_i)\Big)\Big)$
- Otherwise if $i < \ell$: $\mathsf{ipCT}_i \leftarrow \mathsf{sim}^{(i)}_{\mathsf{ipFE}}(\mathsf{ipPK}_i, \mathsf{ipMSK}_i, \mathcal{V}_i, \{1^{|\mathsf{PRF}(K_1,s_i)|}, \ldots, 1^{|\mathsf{PRF}(K_N,s_i)|}\})$ where $\mathcal{V}_i = \Big\{ \Big(e^{\Delta_q}, \Big\langle (\mathsf{PRF}(K_u, s_i))_{u\in[N]}, e^{\Delta_q} \Big\rangle, \mathsf{ipSK}^{(q)}_i\Big) \mid \forall q \in [Q_1]\Big\}$. Store the state in $\mathsf{ipFE.st}_i$.
- Otherwise if $i = \ell$: Set $\mathsf{ipCT}_i \leftarrow \mathcal{O}\left(\Big(\mathsf{PRF}(K_1, s_i), \ldots, \mathsf{PRF}(K_N, s_i)\Big)\right)$

For each $u \in [N]$, $\mathcal{B}$ does the following:

- If $u \in \Delta_{Corr}$: $\mathsf{CT}_u \leftarrow \mathsf{1FE.Enc}(\mathsf{pk}_u, K_u)$.
- Otherwise, if $u \notin \Delta_{Corr}$: $\mathsf{CT}_u \leftarrow \mathsf{sim}^{(u)}_{\mathsf{1FE}}(1^\lambda, V_u)$ where $V_u = \Big\{\hat{y}^{(q)}_u : \forall q \in [Q_1]\Big\}$ and $\hat{y}^{(q)}_u = C_q(K_u)$.

$\mathcal{B}$ sets $\mathsf{CT} = (\mathsf{CT}_1, \ldots, \mathsf{CT}_N, \mathsf{ipCT}_1, \ldots, \mathsf{ipCT}_\gamma)$ and sends it to $\mathcal{A}$.

6. Now $Q - Q_1$ queries for functional secret keys will be queried. For every $q \in \{Q_1 + 1, \ldots, Q\}$, let $T^{(q)} = \Big\{t^{(q)}_1, \ldots, t^{(q)}_{\mu_q}\Big\}$ be the functional set associated with the query. $\mathcal{B}$ calculates $\hat{T}^{(q)}$, the functional circuit C_q, and the N-length vector e^{Δ_q}. For every $i \in [\gamma]$, if $i > \ell$, compute $\mathsf{ipSK}^{(q)}_i \leftarrow \mathsf{ipFE.KeyGen}\left(\mathsf{ipMSK}_i, e^{\Delta_q}\right)$. Otherwise if $k < \ell$, compute $\mathsf{ipSK}^{(q)}_i \leftarrow \mathsf{sim}^{(i)}_{\mathsf{ipFE}}\Big(\mathsf{ipMSK}_i, e^{\Delta_q}, \Big\langle (\mathsf{PRF}(K_u, s_i))_{u\in[N]}, e^{\Delta_q}\Big\rangle, \mathsf{ipFE.st}_i\Big)$. Otherwise, if $i = \ell$, set $\mathsf{ipSK}^{(q)}_i = \mathcal{O}(e^{\Delta_q})$. Then, for every $d \in \Delta_{Corr} \cap \Delta_q$, $\mathcal{B}$ computes $\mathsf{sk}^{(q)}_d \leftarrow \mathsf{1FE.KeyGen}(\mathsf{msk}_d, C_q)$. Otherwise, if $d \in \Delta_q \setminus \Delta_{Corr}$, $\mathcal{B}$ computes $\mathsf{sk}^{(q)}_d \leftarrow \mathsf{sim}^{(d)}_{\mathsf{1FE}}(C_q)$. $\mathcal{B}$ sets $\mathsf{sk}_{T^{(q)}} \leftarrow \left(\hat{T}^{(q)}, \Delta_q, \Big\{\mathsf{ipSK}^{(q)}_i\Big\}_{i\in[\gamma]} \Big\{\mathsf{sk}^{(q)}_d\Big\}_{d\in\Delta_q}\right)$ and sends it to $\mathcal{A}$.

7. $\mathcal{A}$ outputs the bit b. $\mathcal{B}$ outputs b.

As we can see, $\mathcal{B}$ runs in polynomial time in the parameters for Q, λ, L_{max} as all the components used and the oracle $\mathcal{O}$ runs in polynomial time. If $\mathcal{O}$ is the honest challenger for ipFE scheme, $\mathcal{B}$'s output distributions is exactly $\mathrm{Hyb}_{4,\ell}$ and if $\mathcal{O}$ is a simulator for

ipFE, $\mathcal{B}$'s output distribution is same as $\text{Hyb}_{4,\ell+1}$. As $\mathcal{A}$ can distinguish between them with non-negligible advantage and the abort probability for $\mathcal{B}$ is negligible, we can see that $\mathcal{B}$ with non-negligible probability can distinguish between the honest challenger and simulator for ipFE scheme. This contradicts the adaptive security of ipFE. Hence, $\text{Hyb}_{4,\ell}$ and $\text{Hyb}_{4,\ell+1}$ are computationally indistinguishable. □

$\text{Hyb}_{5,\ell}$ for $\ell \in [N]$: We will use random strings instead of $\mathsf{PRF}(K_u, \cdot)$ for $u \leq \ell - 1$.

1. Ch receives the query bound Q and the maximum set size L_{max} from $\mathcal{A}$.

2. Sample Q sets as follows: for every $q \in [Q]$, sample a uniformly random set $\Delta_q \subset [N]$ of size n. Construct $\Delta_{Corr} = \bigcup_{q_1 \neq q_2} (\Delta_{q_1} \cap \Delta_{q_2})$. If $|\Delta_{Corr}| > t$, abort and output $\perp$.

3. For every $u \in [N]$, sample $K_u \xleftarrow{\$} \{0,1\}^\lambda$ uniformly at random.

4. Ch computes the master public key PK as follows: for every $u \in [N]$,

   - If $u \in \Delta_{Corr}$: $(\mathsf{pk}_u, \mathsf{msk}_u) \leftarrow \mathsf{1FE.Setup}(1^\lambda, 1^\psi)$
   - Otherwise, if $u \notin \Delta_{Corr}$, $\mathsf{pk}_u \leftarrow \mathsf{sim}^{(u)}_{\mathsf{1FE}}.\mathsf{Setup}\ (1^\lambda, 1^\psi)$.

   For every $k \in [L_{max}]$, $(\mathsf{ipPK}_k, \mathsf{ipMSK}_k) \leftarrow \mathsf{sim}^{(k)}_{\mathsf{ipFE}}\ (1^m, 1^N)$. Ch sets $\mathsf{PK} = (\mathsf{ipPK}, \ldots, \mathsf{ipPK}_{L_{max}}, \mathsf{pk}_1, \ldots, \mathsf{pk}_N)$ and sends it to $\mathcal{A}$.

5. Maintain a PSet across all functional key queries and challenge set queries such that it contains the set elements corresponding query responses. Initially PSet is empty.

6. $\mathcal{A}$ makes Q_1 functional set queries $T^{(1)}, \ldots, T^{(Q_1)}$ to Ch. For every $q \in [Q_1]$, let $T^{(q)} = \left\{t_1^{(q)}, \ldots, t_{\mu_q}^{(q)}\right\}$, and $\mu_q \leq L_{max}$ be the length of the q-th functional set. Ch computes functional key for $T^{(q)}$ as follows:

- Set $\hat{T}^{(q)} \leftarrow \left(\left(1, T^{(q)}\right), \ldots, \left(\mu_q, T^{(q)}_{\mu_q}\right)\right)$.
- For every $j \in [\mu_q], d \in \Delta_q$,
  - If $d \in \Delta_{Corr} \cap \Delta_q$ or $d \geq \ell$, if $\left(t^{(q)}_j, K_d, \cdot\right) \notin \mathsf{PSet}$: add $\left(t^{(q)}_j, K_d, \mathsf{PRF}\left(K_d, t^{(q)}_j\right)\right)$ to PSet.
  - Otherwise if $d \in \Delta_q \setminus \Delta_{Corr}$ and $d < \ell$, if $\left(t^{(q)}_j, K_d, \cdot\right) \notin \mathsf{PSet}$: sample $\rho_d^{t^{(q)}_j} \xleftarrow{\$} \{0,1\}^m$ and add $\left(t^{(q)}_j, K_d, \rho_d^{t^{(q)}_j}\right)$ to PSet.
- Create a functional circuit C_q that takes K as input and outputs $\left(\mathsf{PRF}(K, t^{(q)}_1), \ldots, \mathsf{PRF}\left(K, t^{(q)}_{\mu_q}\right)\right)$
- Construct an N length vector e^{Δ_q} where $e^{\Delta_q}_d = 1$ if $d \in \Delta_q$ and 0 otherwise.
- For every $k \in [L_{max}]$, compute the simulated functional key for e^{Δ_q}. That is $\mathsf{ipSK}^{(q)}_k \leftarrow \mathsf{sim}^{(k)}_{\mathsf{ipFE}}\left(\mathsf{ipMSK}_k, e^{\Delta_q}\right)$.
- For every $d \in \Delta_q$,
  - If $d \in \Delta_{Corr} \cap \Delta_q$: compute $\mathsf{sk}^{(q)}_d \leftarrow \mathsf{1FE.KeyGen}(\mathsf{msk}_d, C_q)$
  - Otherwise, if $d \in \Delta_q \setminus \Delta_{Corr}$: compute the functional secret key for C_q. That is, $\mathsf{sk}^{(q)}_d \leftarrow \mathsf{sim}^{(d)}_{\mathsf{1FE}}(C_q)$.

Set $\mathsf{sk}_{T^{(q)}} = \left(\hat{T}^{(q)}, \Delta_q, \left\{\mathsf{ipSK}^{(q)}_k\right\}_{k \in [L_{max}]}, \left\{\mathsf{sk}^{(q)}_d\right\}_{d \in \Delta_q}\right)$. Send $\mathsf{sk}_{T^{(q)}}$ to $\mathcal{A}$.

7. $\mathcal{A}$ submits the challenge set $S = \{s_1, \ldots, s_\gamma\}$. Ch does the following:
   - For every $i \in [\gamma], u \in [N]$,
     - If $u \in \Delta_{Corr}$ or $u \geq \ell$, if $(s_i, K_u, \cdot) \notin \mathsf{PSet}$: add $(s_i, K_u, \mathsf{PRF}(K_u, s_i))$ to PSet.
     - Otherwise, if $u \notin \Delta_{Corr}$ and $u < \ell$, if $(s_i, K_u, \cdot) \notin \mathsf{PSet}$: sample $\rho^{s_i}_u \xleftarrow{\$} \{0,1\}^m$ and add $(s_i, K_u, \rho^{s_i}_u)$ to PSet.

Retrieve $(s_i, K_u, \rho_u^{s_i})_{u\in[N],i\in[\gamma]}$ from PSet

- For every $i \in [\gamma]$, let $\mathcal{V}_i = \left\{ \left(e^{\Delta_q}, \left\langle (\rho_u^{s_i})_{u\in[N]}, e^{\Delta_q} \right\rangle, \mathsf{ipSK}_i^{(q)} \right) \mid \forall\ q \in [Q_1] \right\}$. $\mathsf{ipCT}_i \leftarrow \mathsf{sim}_{\mathsf{ipFE}}^{(i)}\left(\mathsf{ipPK}_i,\ \mathsf{ipMSK}_i, \mathcal{V}_i, \left\{1^{|\rho_1^{s_i}|}, \ldots, 1^{|\rho_N^{s_i}|}\right\}\right)$. Store the state returned by the simulator separately in $\mathsf{ipFE.st}_i$.
- For every $u \in [N]$,
  - If $u \in \Delta_{Corr}$: $\mathsf{CT}_u \leftarrow \mathsf{1FE.Enc}\left(\mathsf{pk}_u, K_u\right)$.
  - Otherwise if $u \notin \Delta_{Corr}$: for every $q \in [Q_1]$, let $\hat{y}_u^{(q)} = \left(\rho_u^{t_1^{(q)}}, \ldots, \rho_u^{t_{\mu_q}^{(q)}}\right)$ and $V_u = \left\{\hat{y}_u^{(q)} \mid \forall\ q \in [Q_1]\right\}$. Compute the simulated ciphertext $\mathsf{CT}_u \leftarrow \mathsf{sim}_{\mathsf{1FE}}^{(u)}(1^\lambda, V_u)$.

Set $\mathsf{CT} = (\mathsf{CT}_1, \ldots, \mathsf{CT}_N, \mathsf{ipCT}_1, \ldots, \mathsf{ipCT}_\gamma)$. Send CT to $\mathcal{A}$.

8. $\mathcal{A}$ makes $Q - Q_1$ functional set queries $T^{(Q_1+1)}, \ldots, T^{(Q)}$ to Ch. For every $q \in \{Q_1+1, \ldots, Q\}$, let $T^{(q)} = \left\{t_1^{(q)}, \ldots, t_{\mu_q}^{(q)}\right\}$, and $\mu_q \le L_{max}$ be the length of the q-th functional set. Ch computes functional key for $T^{(q)}$ as follows:
   - Set $\hat{T^{(q)}} \leftarrow \left(\left(1, t_1^{(q)}\right), \ldots, \left(\mu_q, t_{\mu_q}^{q}\right)\right)$.
   - For every $j \in [\mu_q], d \in \Delta_q$,
     - If $d \in \Delta_{Corr} \cap \Delta_q$ or $d \ge \ell$, if $\left(t_j^{(q)}, K_d, \cdot\right) \notin \mathsf{PSet}$: add $\left(t_j^{(q)}, K_d, \mathsf{PRF}\left(K_d, t_j^{(q)}\right)\right)$ to PSet.
     - Otherwise if $d \in \Delta_q \setminus \Delta_{Corr}$ and $d < \ell$, if $\left(t_j^{(q)}, K_d, \cdot\right) \notin \mathsf{PSet}$: sample $\rho_d^{t_j^{(q)}} \xleftarrow{\$} \{0,1\}^m$ and add $\left(t_j^{(q)}, K_d, \rho_d^{t_j^{(q)}}\right)$ to PSet.
   - Create a functional circuit C_q that takes K as input and outputs $\left(\mathsf{PRF}\left(K, t_1^{(q)}\right), \ldots, \mathsf{PRF}\left(K, t_{\mu_q}^{(q)}\right)\right)$.
   - Construct an N length vector e^{Δ_q} where $e_d^{\Delta_q} = 1$ if $d \in \Delta_q$ and 0 otherwise.

- For every $i \in [\gamma]$, compute the simulated functional key for e^{Δ_q}. That is $\mathsf{ipSK}_i^{(q)} \leftarrow \mathsf{sim}_{\mathsf{ipFE}}^{(i)}\left(\mathsf{ipMSK}_i, e^{\Delta_q}, \left\langle \left(\rho_u^{s_i}\right)_{u\in[N]}, e^{\Delta_q}\right\rangle, \mathsf{ipFE.st}_i\right)$.
- For every $d \in \Delta_q$,
  - If $d \in \Delta_{Corr} \cap \Delta_q$: compute $\mathsf{sk}_d^{(q)} \leftarrow \mathsf{1FE.KeyGen}(\mathsf{msk}_d, C_q)$.
  - Otherwise, if $d \in \Delta_q \setminus \Delta_{Corr}$: compute the functional secret key for C_q. That is, $\mathsf{sk}_d^{(q)} \leftarrow \mathsf{sim}_{\mathsf{1FE}}^{(d)}\left(C_q, \hat{y}_d^{(q)}\right)$ where $\hat{y}_d^{(q)} = \left(\rho_d^{t_1^{(q)}}, \ldots, \rho_d^{t_{\mu_q}^{(q)}}\right)$.

Set $\mathsf{sk}_{T^{(q)}} = \left(\hat{T}^{(q)}, \Delta_q, \left\{\mathsf{ipSK}_i^{(q)}\right\}_{i\in[\gamma]}, \left\{\mathsf{sk}_d^{(q)}\right\}_{d\in\Delta_q}\right)$. Send $\mathsf{sk}_{T^{(q)}}$ to $\mathcal{A}$.

9. $\mathcal{A}$ outputs b. Output b.

Claim 14. *Assuming the security of ipFE, $Hyb_{4,\gamma}$ and $Hyb_{5,1}$ are computationally indistinguishable.*

Proof: As the only difference is the early generation of PRF keys and initialization of PSet to empty set, the remaining distribution follows the proof of claim 13. □

Claim 15. *Assuming the security of* PRF *scheme, for every $\ell \in [N-1]$, $Hyb_{5,\ell}$ and $Hyb_{5,\ell+1}$ are computationally indistinguishable.*

Proof: Note that if $\ell \in \Delta_{Corr}$, we are not substituting $\mathsf{PRF}(K_\ell, \cdot)$. Hence, the output distribution of $\mathrm{Hyb}_{5,\ell}$ will be exactly same as $\mathrm{Hyb}_{5,\ell+1}$. WLOG, assume that $\ell \notin \Delta_{Corr}$ i.e, ℓ-th instantiation is not corrupted and is only queried once for functional secret key. Assuming that there exists an adversary $\mathcal{A}$ that can distinguish the hybrids $\mathrm{Hyb}_{5,\ell}$ and $\mathrm{Hyb}_{5,\ell+1}$, we will construct a hybrid that distinguish between the oracles $\mathsf{PRF}\left(K_\ell, \cdot\right) : \{0,1\}^\lambda \times \{0,1\}^l \to \{0,1\}^m$ and the random function $\mathcal{R}(\cdot) : \{0,1\}^l \to \{0,1\}^m$ from the same domain to co-domain. More formally, if $\mathcal{A}$'s advantage is such that

$$\left|\Pr\left[0 \leftarrow \mathcal{A}^{\mathsf{Hyb}_{5,\ell}}(1^\lambda)\right] - \Pr\left[0 \leftarrow \mathcal{A}^{\mathsf{Hyb}_{5,\ell+1}}(1^\lambda)\right]\right| > \mathsf{negl}(\lambda)$$

We will construct a polynomial time reduction $\mathcal{B}$ which distinguishes between oracle access to $\mathsf{PRF}(K_\ell, \cdot)$ and a random function $\mathcal{R}(\cdot)$. More specifically, we will construct an adversary $\mathcal{B}$ whose advantage in distinguishing the output of the pseudorandom function and a uniformly random function is non-negligible.

$$\left|\Pr\left[0 \leftarrow \mathcal{B}^{\mathsf{PRF}(K_\ell,\cdot)}(1^\lambda)\right] - \Pr\left[0 \leftarrow \mathcal{B}^{\mathcal{R}(\cdot)}(1^\lambda)\right]\right| > \mathsf{negl}(\lambda)$$

The description of $\mathcal{B}$ is as follows:

1. $\mathcal{A}$ sends Q, L_{max} to $\mathcal{B}$.

2. $\mathcal{B}$ calculates $t = \Theta(\lambda), n = \Theta(t)$, and $N = \Theta(Q^2 t)$. $\mathcal{B}$ computes $\Delta_q \subseteq [N]$ of size n for each $q \in [Q]$ and $\Delta_{Corr} = \cup_{q_1 \neq q_2} \Delta_{q_1} \cap \Delta_{q_2}$. $\mathcal{B}$ aborts if $|\Delta_{Corr}| > t$ and outputs $\perp$.

3. For every $u \in [N]$, sample $K_u \xleftarrow{\$} \{0,1\}^\lambda$.

4. $\mathcal{B}$ computes master public key as follows: for every $u \in [N]$, if $u \in \Delta_{Corr}$: $(\mathsf{pk}_u, \mathsf{msk}_u) \leftarrow \mathsf{1FE.Setup}(1^\lambda, 1^\psi)$. Otherwise, if $u \notin \Delta_{Corr}$: $\mathsf{pk}_u \leftarrow \mathsf{sim}^{(u)}_{\mathsf{1FE}}(1^\lambda, 1^\psi)$. For every $k \in [L_{max}]$, $\mathcal{B}$ computes $(\mathsf{ipPK}_k, \mathsf{ipMSK}_k) \leftarrow \mathsf{sim}^{(k)}_{\mathsf{ipFE}}(1^m, 1^N)$. $\mathcal{B}$ sets $\mathsf{PK} \leftarrow \left(\mathsf{ipPK}_1, \ldots, \mathsf{ipPK}_{L_{max}}, \mathsf{pk}_1, \ldots, \mathsf{pk}_N\right)$ and sends it to $\mathcal{A}$.

5. $\mathcal{B}$ initializes PSet to empty.

6. Let Q_1 queries for functional secret keys be submitted. For every $q \in [Q_1]$, let $T^{(q)} = \left\{t_1^{(q)}, \ldots, t_{\mu_q}^{(q)}\right\}$ be the functional set associated with the query. For every $j \in [\mu_q], d \in \Delta_q$,

   - If $d \in \Delta_{Corr} \cap \Delta_q$ or $d > \ell$, if $\left(t_j^{(q)}, K_d, \cdot\right) \notin \mathsf{PSet}$: add $\left(t_j^{(q)}, K_d, \mathsf{PRF}\left(K_d, t_j^{(q)}\right)\right)$ to PSet.

- Otherwise if $d \in \Delta_q \setminus \Delta_{Corr}$ and $d < \ell$, if $(t_j^{(q)}, K_d, \cdot) \notin \mathsf{PSet}$: sample $\rho_d^{t_j^{(q)}} \xleftarrow{\$} \{0,1\}^m$ and add $\left(t_j^{(q)}, K_d, \rho_d^{t_j^{(q)}}\right)$ to PSet.
- Otherwise if $d = \ell$, if $\left(t_j^{(q)}, K_d, \cdot\right) \notin \mathsf{PSet}$: set $\rho_d^{t_j^{(q)}} \leftarrow \mathcal{O}(t_j^{(q)})$ and add $\left(t_j^{(q)}, K_d, \rho_d^{t_j^{(q)}}\right)$ to PSet.

$\mathcal{B}$ calculates $\hat{T}^{(q)}$ and the N-length vector e^{Δ_q}. $\mathcal{B}$ creates the functional C_q that takes input $K \in \{0,1\}^\lambda$ and outputs $\left(\mathsf{PRF}\left(K, t_1^{(q)}\right), \ldots, \mathsf{PRF}\left(K, t_{\mu_q}^{(q)}\right)\right)$. For every $k \in [L_{max}]$, compute $\mathsf{ipSK}_k^{(q)} \leftarrow \mathsf{sim}_{\mathsf{ipFE}}^{(k)}(\mathsf{ipMSK}_k, e^{\Delta_q})$. Then, for every $d \in \Delta_{Corr} \cap \Delta_q$, $\mathcal{B}$ computes $\mathsf{sk}_d^{(q)} \leftarrow \mathsf{1FE.KeyGen}(\mathsf{msk}_d, C_q)$. Otherwise, if $d \in \Delta_q \setminus \Delta_{Corr}$, $\mathcal{B}$ computes $\mathsf{sk}_d^{(q)} \leftarrow \mathsf{sim}_{\mathsf{1FE}}^{(d)}(C_q)$. $\mathcal{B}$ sets $\mathsf{sk}_{T^{(q)}} \leftarrow \left(\hat{T}^{(q)}, \Delta_q, \left\{\mathsf{ipSK}_k^{(q)}\right\}_{k \in [L_{max}]} \left\{\mathsf{sk}_d^{(q)}\right\}_{d \in \Delta_q}\right)$ and sends it to $\mathcal{A}$.

7. $\mathcal{A}$ submits the challenge set $S = \{s_1, \ldots, s_\gamma\}$. For every $i \in [\gamma], u \in [N]$,

   - If $u \in \Delta_{Corr}$ or $u > \ell$, if $(s_i, K_u, \cdot) \notin \mathsf{PSet}$: add $(s_i, K_u, \mathsf{PRF}(K_u, s_i))$ to PSet.
   - Otherwise, if $u \notin \Delta_{Corr}$ and $u < \ell$, if $(s_i, K_u, \cdot) \notin \mathsf{PSet}$: sample $\rho_u^{s_i} \xleftarrow{\$} \{0,1\}^m$ and add $(s_i, K_u, \rho_u^{s_i})$ to PSet.
   - Otherwise if $u = \ell$, if $(s_i, K_u, \cdot) \notin \mathsf{PSet}$: set $\rho_u^{s_i} \leftarrow \mathcal{O}(s_i)$ and add $(s_i, K_u, \rho_u^{s_i})$ to PSet.

   Retrieve $(s_i, K_u, \alpha_u^{s_i})_{u \in [N], i \in [\gamma]}$. For every $i \in [\gamma]$, compute $\mathsf{ipCT}_i \leftarrow \mathsf{sim}_{\mathsf{ipFE}}^{(i)}\Big(\mathsf{ipPK}_i, \mathsf{ipMSK}_i, \mathcal{V}_i, \left\{1^{|\rho_1^{s_i}|}, \ldots, 1^{|\rho_N^{s_i}|}\right\}\Big)$ where $\mathcal{V}_i = \left\{ \left(e^{\Delta_q}, \left\langle (\rho_u^{s_i})_{u \in [N]}, e^{\Delta_q} \right\rangle, \mathsf{ipSK}_i^{(q)}\right) \,\middle|\, \forall q \in [Q_1] \right\}$. Store the state in $\mathsf{ipFE.st}_i$. For each $u \in [N]$, $\mathcal{B}$ does the following: if $u \in \Delta_{Corr}$: $\mathsf{CT}_u \leftarrow \mathsf{1FE.Enc}(\mathsf{pk}_u, K_u)$. Otherwise, if $u \notin \Delta_{Corr}$: $\mathsf{CT}_u \leftarrow \mathsf{sim}_{\mathsf{1FE}}^{(u)}(1^\lambda, V_u)$ where $V_u = \left\{\hat{y}_u^{(q)} \mid \forall q \in [Q_1]\right\}$ and $\hat{y}_u^{(q)} = \left(\rho_u^{t_1^{(q)}}, \ldots, \rho_u^{t_{\mu_q}^{(q)}}\right)$. $\mathcal{B}$ sets $\mathsf{CT} = (\mathsf{CT}_1, \ldots, \mathsf{CT}_N, \mathsf{ipCT}_1, \ldots, \mathsf{ipCT}_\gamma)$ and sends it to $\mathcal{A}$.

8. Now $Q - Q_1$ queries for functional secret keys will be queried. For every $q \in \{Q_1 + 1, \ldots, Q\}$,let $T^{(q)} = \left\{t_1^{(q)}, \ldots, t_{\mu_q}^{(q)}\right\}$ be the functional set associated with the query. $\mathcal{B}$ calculates $\hat{T}^{(q)}$ and the N-length vector e^{Δ_q}. For every $j \in [\mu_q], d \in \Delta_q$,

   - If $d \in \Delta_{Corr} \cap \Delta_q$ or $d > \ell$, if $\left(t_j^{(q)}, K_d, \cdot\right) \notin \mathsf{PSet}$: add $\left(t_j^{(q)}, K_d, \mathsf{PRF}\left(K_d, t_j^{(q)}\right)\right)$ to PSet.
   - Otherwise if $d \in \Delta_q \setminus \Delta_{Corr}$ and $d < \ell$, if $\left(t_j^{(q)}, K_d, \cdot\right) \notin \mathsf{PSet}$: sample $\rho_d^{t_j^{(q)}} \xleftarrow{\$} \{0,1\}^m$ and add $\left(t_j^{(q)}, K_d, \rho_d^{t_j^{(q)}}\right)$ to PSet.
   - Otherwise if $d = \ell$, if $\left(t_j^{(q)}, K_d, \cdot\right) \notin \mathsf{PSet}$: set $\rho_d^{t_j^{(q)}} \leftarrow \mathcal{O}(t_j^{(q)})$ and add $\left(t_j^{(q)}, K_d, \rho_d^{t_j^{(q)}}\right)$ to PSet.

   $\mathcal{B}$ creates the functional C_q that takes input $K \in \{0,1\}^\lambda$ and outputs $\big(\mathsf{PRF}(K, t_1^{(q)}), \ldots, \mathsf{PRF}(K, t_{\mu_q}^{(q)})\big)$. For every $i \in [\gamma]$, compute $\mathsf{ipSK}_i^{(q)} \leftarrow \mathsf{sim}_{\mathsf{ipFE}}^{(i)}\Bigg(\mathsf{ipMSK}_i, e^{\Delta_q}, \left\langle \left(\rho_u^{s_i}\right)_{u \in [N]}, e^{\Delta_q} \right\rangle, \mathsf{ipFE.st}_i\Bigg)$. Then, for every $d \in \Delta_{Corr} \cap \Delta_q$, $\mathcal{B}$ computes $\mathsf{sk}_d^{(q)} \leftarrow \mathsf{1FE.KeyGen}(\mathsf{msk}_d, C_q)$. Otherwise, if $d \in \Delta_q \setminus \Delta_{Corr}$, $\mathcal{B}$ computes $\mathsf{sk}_d^{(q)} \leftarrow \mathsf{sim}_{\mathsf{1FE}}^{(d)}(C_q, \hat{y}_d^{(q)})$ where $\hat{y}_d^{(q)} = \left(\rho_d^{t_1^{(q)}}, \ldots, \rho_d^{t_{\mu_q}^{(q)}}\right)$. $\mathcal{B}$ then computes $\mathsf{sk}_{T^{(q)}}$ as $\left(\hat{T}^{(q)}, \Delta_q, \left\{\mathsf{ipSK}_i^{(q)}\right\}_{i \in [\gamma]}, \left\{\mathsf{sk}_d^{(q)}\right\}_{d \in \Delta_q}\right)$ and sends it to $\mathcal{A}$.

9. $\mathcal{A}$ outputs the bit b. $\mathcal{B}$ outputs b.

As we can see, $\mathcal{B}$ runs in polynomial time in the parameters for Q, λ, L_{max} as both $\mathsf{PRF}(K_\ell, \cdot)$ and $\mathcal{R}(\cdot)$ run in polynomial times too. If the oracle $\mathcal{O}$ is $\mathsf{PRF}(K_\ell, \cdot), \mathcal{B}$ behaves like $\mathrm{Hyb}_{5,\ell}$ and if $\mathcal{O}$ is $\mathcal{R}(\cdot), \mathcal{B}$ behaves like $\mathrm{Hyb}_{5,\ell+1}$. As $\mathcal{A}$ can distinguish between them with non-negligible advantage and the abort probability for $\mathcal{B}$ is negligible, we can see that $\mathcal{B}$ with non-negligible probability can distinguish between $\mathsf{PRF}(K_\ell, \cdot)$ and $\mathcal{R}(\cdot)$ oracles. This contradicts our assumption for secure PRF. Hence, $\mathrm{Hyb}_{5,\ell}$ and $\mathrm{Hyb}_{5,\ell+1}$ are computationally indistinguishable. □

$\underline{\text{Hyb}_6}$: Same as the simulated experiment.

Claim 16. *Assuming the security of* PRF *scheme,* $Hyb_{5,N}$ *and* Hyb_6 *are computationally indistinguishable.*

Proof: As we are substituting random strings for the $\mathsf{PRF}(K_N, \cdot)$ oracle, the proof is similar to claim 15. Moreover, the simulator is identically distributed to $\text{Hyb}_{5,N}$. □

7.5 Linearizing Encryption

The current output size and running time of the encryption algorithm is linear in $N = \Theta(Q^2\lambda)$. We can linear the size and time in the query bound Q using the load balancing theorem, Theorem 4 from [AV19]. Thus, we get a bounded key functional encryption scheme for set intersection whose output size and running time are $O(Q \cdot poly(\lambda, L_{max}))$.

Chapter 8

Further Optimizations

- We can trivially optimize the decryption algorithms in the private- and public-key constructions by using an efficient hashing mechanism such as cuckoo hashing to decrease the complexity from quadratic to amortized linear time.

- The semi-adaptive security for private-key functional encryption scheme is sub-optimal. A desirable avenue that might lead to an adaptive collusion-resistant functional encryption scheme for set intersection might involve random oracles. This avenue is being pursued by us now.

- Although unbounded public-key functional encryption for general circuits imply indistinguishability obfuscation, for specific circuits such as the one we saw in chapter 7 is feasible. Research in this direction using bilinear maps is being pursued.

- One caveat in the public- and private-key schemes is that they are a two-party scheme. Can multi-input functional encryption schemes for set intersection be constructed from standard assumptions? If so, what are the security requirements?

Bibliography

[ABCP15] Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval. Simple functional encryption schemes for inner products. In *IACR International Workshop on Public Key Cryptography*, pages 733–751. Springer, 2015.

[ABG19] Michel Abdalla, Fabrice Benhamouda, and Romain Gay. From single-input to multi-client inner-product functional encryption. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 552–582. Springer, 2019.

[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 *Annual International Cryptology Conference*, pages 597–627. Springer, 2018.

[ACFQ22] Prabhanjan Ananth, Kai-Min Chung, Xiong Fan, and Luowen Qian. Collusion-resistant functional encryption for rams. *Cryptology ePrint Archive*, 2022.

[AGRW17] Michel Abdalla, Romain Gay, Mariana Raykova, and Hoeteck Wee. Multi-input inner-product functional encryption from pairings. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 601–626. Springer, 2017.

[AGT21] Shweta Agrawal, Rishab Goyal, and Junichi Tomida. Multi-input quadratic functional encryption from pairings. In *Annual International Cryptology Conference*, pages 208–238. Springer, 2021.

[AGT22] Shweta Agrawal, Rishab Goyal, and Junichi Tomida. Multi-input quadratic functional encryption: Stronger security, broader functionality. In *Theory of Cryptography Conference*, pages 711–740. Springer, 2022.

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

[AKM+22] Shweta Agrawal, Fuyuki Kitagawa, Anuja Modi, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Bounded functional encryption for turing machines: Adaptive security from general assumptions. *Cryptology ePrint Archive*, 2022.

[AL18] Prabhanjan Ananth and Alex Lombardi. Succinct garbling schemes from functional encryption through a local simulation paradigm. In *Theory of Cryptography Conference*, pages 455–472. Springer, 2018.

[ALMT20] Shweta Agrawal, Benoît Libert, Monosij Maitra, and Radu Titiu. Adaptive simulation security for inner product functional encryption. In *IACR International Conference on Public-Key Cryptography*, pages 34–64. Springer, 2020.

[ALS16] Shweta Agrawal, Benoît Libert, and Damien Stehlé. Fully secure functional encryption for inner products, from standard assumptions. In *Annual International Cryptology Conference*, pages 333–362. Springer, 2016.

[AM18] Shweta Agrawal and Monosij Maitra. Fe and io for turing machines from minimal assumptions. In *Theory of Cryptography Conference*, pages 473–512. Springer, 2018.

[AMVY21] Shweta Agrawal, Monosij Maitra, Narasimha Sai Vempati, and Shota Yamada. Functional encryption for turing machines with dynamic bounded collusion from lwe. In *Annual International Cryptology Conference*, pages 239–269. Springer, 2021.

[APS21] Michel Abdalla, David Pointcheval, and Azam Soleimanian. 2-step multi-client quadratic functional encryption from decentralized function-hiding inner-product. *Cryptology ePrint Archive*, 2021.

[AR17] Shweta Agrawal and Alon Rosen. Functional encryption for bounded collusions, revisited. In *Theory of Cryptography Conference*, pages 173–205. Springer, 2017.

[AS16] Prabhanjan Ananth and Amit Sahai. Functional encryption for turing machines. In *Theory of Cryptography Conference*, pages 125–153. Springer, 2016.

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

[BJK15] Allison Bishop, Abhishek Jain, and Lucas Kowalczyk. Function-hiding inner product encryption. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 470–491. Springer, 2015.

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *Theory of Cryptography Conference*, pages 253–273. Springer, 2011.

[BV18] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. *Journal of the ACM (JACM)*, 65(6):1–37, 2018.

[CDG+21] Nishanth Chandran, Nishka Dasgupta, Divya Gupta, Sai Lakshmi Bhavana Obbattu, Sruthi Sekar, and Akash Shah. Efficient linear multiparty psi and extensions to circuit/quorum psi. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 1182–1204, 2021.

[CDSJ16] Chongwon Cho, Dana Dachman-Soled, and Stanisław Jarecki. Efficient concurrent covert computation of string equality and set intersection. In *Cryptographers' Track at the RSA Conference*, pages 164–179. Springer, 2016.

[CGH+21] Melissa Chase, Sanjam Garg, Mohammad Hajiabadi, Jialin Li, and Peihan Miao. Amortizing rate-1 ot and applications to pir and psi. In *Theory of Cryptography Conference*, pages 126–156. Springer, 2021.

[CGS21] Nishanth Chandran, Divya Gupta, and Akash Shah. Circuit-psi with linear complexity via relaxed batch opprf. *Cryptology ePrint Archive*, 2021.

[CHLR18] Hao Chen, Zhicong Huang, Kim Laine, and Peter Rindal. Labeled psi from fully homomorphic encryption with malicious security. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1223–1237, 2018.

[CLR17] Hao Chen, Kim Laine, and Peter Rindal. Fast private set intersection from homomorphic encryption. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1243–1255, 2017.

[CM20] Melissa Chase and Peihan Miao. Private set intersection in the internet setting from lightweight oblivious prf. In *Annual International Cryptology Conference*, pages 34–63. Springer, 2020.

[CO18] Michele Ciampi and Claudio Orlandi. Combining private set-intersection with secure two-party computation. In *International Conference on Security and Cryptography for Networks*, pages 464–482. Springer, 2018.

[CT10] Emiliano De Cristofaro and Gene Tsudik. Practical private set intersection protocols with linear complexity. In *International Conference on Financial Cryptography and Data Security*, pages 143–159. Springer, 2010.

[CTX20] Yunlu Cai, Chunming Tang, and Qiuxia Xu. Two-party privacy-preserving set intersection with fhe. *Entropy*, 22(12):1339, 2020.

[CVW+18] Yilei Chen, Vinod Vaikuntanathan, Brent Waters, Hoeteck Wee, and Daniel Wichs. Traitor-tracing from lwe made simple and attribute-based. In *Theory of Cryptography Conference*, pages 341–369. Springer, 2018.

[DCT12] Emiliano De Cristofaro and Gene Tsudik. Experimenting with fast private set intersection. In *International Conference on Trust and Trustworthy Computing*, pages 55–73. Springer, 2012.

[DCW13] Changyu Dong, Liqun Chen, and Zikai Wen. When private set intersection meets big data: an efficient and scalable protocol. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 789–800, 2013.

[DDM16] Pratish Datta, Ratna Dutta, and Sourav Mukhopadhyay. Functional encryption for inner product with full function privacy. In *Public-Key Cryptography–PKC 2016*, pages 164–195. Springer, 2016.

[DOT18] Pratish Datta, Tatsuaki Okamoto, and Junichi Tomida. Full-hiding (unbounded) multi-input inner product functional encryption from the k-linear assumption. In *IACR International Workshop on Public Key Cryptography*, pages 245–277. Springer, 2018.

[DSMRY09] Dana Dachman-Soled, Tal Malkin, Mariana Raykova, and Moti Yung. Efficient robust private set intersection. In *International Conference on Applied Cryptography and Network Security*, pages 125–142. Springer, 2009.

[FIPR05] Michael J Freedman, Yuval Ishai, Benny Pinkas, and Omer Reingold. Keyword search and oblivious pseudorandom functions. In *Theory of Cryptography Conference*, pages 303–324. Springer, 2005.

[FNP04] Michael J Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In *International conference on the theory and applications of cryptographic techniques*, pages 1–19. Springer, 2004.

[FY92] Matthew Franklin and Moti Yung. Communication complexity of secure computation. In *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, pages 699–710, 1992.

[GHRW14] Craig Gentry, Shai Halevi, Mariana Raykova, and Daniel Wichs. Outsourcing private ram computation. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 404–413. IEEE, 2014.

[GKM+19] Rishab Goyal, Sam Kim, Nathan Manohar, Brent Waters, and David J Wu. Watermarking public-key cryptographic primitives. In *Annual International Cryptology Conference*, pages 367–398. Springer, 2019.

[GPR+21] Gayathri Garimella, Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. Oblivious key-value stores and amplification for private set intersection. In *Annual International Cryptology Conference*, pages 395–425. Springer, 2021.

[GS18] Sanjam Garg and Akshayaram Srinivasan. Adaptively secure garbling with near optimal online complexity. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 535–565. Springer, 2018.

[GS19] Satrajit Ghosh and Mark Simkin. The communication complexity of threshold private set intersection. In *Annual International Cryptology Conference*, pages 3–29. Springer, 2019.

[GVW12] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In *Annual Cryptology Conference*, pages 162–179. Springer, 2012.

[HFNO19] Brett Hemenway Falk, Daniel Noble, and Rafail Ostrovsky. Private set intersection with linear communication from general assumptions. In *Proceedings of the 18th ACM Workshop on Privacy in the Electronic Society*, pages 14–25, 2019.

[HN10] Carmit Hazay and Kobbi Nissim. Efficient set operations in the presence of malicious adversaries. In *International Workshop on Public Key Cryptography*, pages 312–331. Springer, 2010.

[JL09] Stanisław Jarecki and Xiaomin Liu. Efficient oblivious pseudorandom function with applications to adaptive ot and secure computation of set intersection. In *Theory of Cryptography Conference*, pages 577–594. Springer, 2009.

[JL10] Stanisław Jarecki and Xiaomin Liu. Fast secure computation of set intersection. In *International Conference on Security and Cryptography for Networks*, pages 418–435. Springer, 2010.

[JWP22] Yuting Jiang, Jianghong Wei, and Jing Pan. Publicly verifiable private set intersection from homomorphic encryption. In *International Symposium on Security and Privacy in Social Networks and Big Data*, pages 117–137. Springer, 2022.

[KLS+17] Ágnes Kiss, Jian Liu, Thomas Schneider, N Asokan, and Benny Pinkas. Private set intersection for unequal set sizes with mobile applications. *Cryptology ePrint Archive*, 2017.

[KMUW18] Lucas Kowalczyk, Tal Malkin, Jonathan Ullman, and Daniel Wichs. Hardness of non-interactive differential privacy from one-way functions. In *Annual International Cryptology Conference*, pages 437–466. Springer, 2018.

[KRS+19] Daniel Kales, Christian Rechberger, Thomas Schneider, Matthias Senker, and Christian Weinert. Mobile private contact discovery at scale. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 1447–1464, 2019.

[KRTW19] Vladimir Kolesnikov, Mike Rosulek, Ni Trieu, and Xiao Wang. Scalable private set union from symmetric-key techniques. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 636–666. Springer, 2019.

[KS05] Lea Kissner and Dawn Song. Privacy-preserving set operations. In *Annual International Cryptology Conference*, pages 241–257. Springer, 2005.

[LRSZ21] Bai Liu, Ou Ruan, Runhua Shi, and Mingwu Zhang. Quantum private set intersection cardinality based on bloom filter. *Scientific Reports*, 11(1):1–9, 2021.

[MBD12] Dilip Many, Martin Burkhart, and Xenofontas Dimitropoulos. Fast private set operations with sepia. *ETZ G93*, 2012.

[O'N10] Adam O'Neill. Definitional issues in functional encryption. *Cryptology ePrint Archive*, 2010.

[PRTY19] Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. Spot-light: Lightweight private set intersection from sparse ot extension. In *Annual International Cryptology Conference*, pages 401–431. Springer, 2019.

[PRTY20] Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. Psi from paxos: fast, malicious private set intersection. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 739–767. Springer, 2020.

[PSSZ15] Benny Pinkas, Thomas Schneider, Gil Segev, and Michael Zohner. Phasing: Private set intersection using permutation-based hashing. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 515–530, 2015.

[PSTY19] Benny Pinkas, Thomas Schneider, Oleksandr Tkachenko, and Avishay Yanai. Efficient circuit-based psi with linear communication. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 122–153. Springer, 2019.

[PSWW18] Benny Pinkas, Thomas Schneider, Christian Weinert, and Udi Wieder. Efficient circuit-based psi via cuckoo hashing. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 125–157. Springer, 2018.

[PSZ14] Benny Pinkas, Thomas Schneider, and Michael Zohner. Faster private set intersection based on {OT} extension. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 797–812, 2014.

[PSZ18] Benny Pinkas, Thomas Schneider, and Michael Zohner. Scalable private set intersection based on ot extension. *ACM Transactions on Privacy and Security (TOPS)*, 21(2):1–35, 2018.

[RA18] Amanda C Davi Resende and Diego F Aranha. Faster unbalanced private set intersection. In *International Conference on Financial Cryptography and Data Security*, pages 203–221. Springer, 2018.

[RPB+19] Théo Ryffel, David Pointcheval, Francis Bach, Edouard Dufour-Sans, and Romain Gay. Partially encrypted deep learning using functional encryption. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[RR17] Peter Rindal and Mike Rosulek. Improved private set intersection against malicious adversaries. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 235–259. Springer, 2017.

[SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 457–473. Springer, 2005.

[TAO16] Junichi Tomida, Masayuki Abe, and Tatsuaki Okamoto. Efficient functional encryption for inner-product values with full-hiding security. In *International Conference on Information Security*, pages 408–425. Springer, 2016.

[TT20] Junichi Tomida and Katsuyuki Takashima. Unbounded inner product functional encryption from bilinear maps. *Japan Journal of Industrial and Applied Mathematics*, 37(3):723–779, 2020.

[TYG22] Ni Trieu, Avishay Yanai, and Jiahui Gao. Multiparty private set intersection cardinality and its applications. *Cryptology ePrint Archive*, 2022.

[ZYZL19] Shuoyao Zhao, Yu Yu, Jiang Zhang, and Hanlin Liu. Valiant's universal circuits revisited: an overall improvement and a lower bound. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 401–425. Springer, 2019.