Differentially Private Functional Encryption

Jasmin Zalonis
University of Mannheim
Mannheim, Germany
zalonis@uni-mannheim.de

Frederik Armknecht
University of Mannheim
Mannheim, Germany
armknecht@uni-mannheim.de

Linda Scheu-Hachtel
University of Mannheim
Mannheim, Germany
lscheuha@mail.uni-mannheim.de

ABSTRACT

We address the question of realizing privacy preserving analysis of user data. The abstract scenario considered is that an analyst aims to evaluate a function f on some user data X. To achieve comprehensive privacy, it is necessary to protect the input X directly. However, it is known that $f(X)$ may leak too much information about X as well. A common approach to mitigate such risks is to make the computation differential private. In practice, this is often accomplished by replacing f by a noisy variant f^*.

We investigate the use of multi-input functional encryption (MIFE) for achieving input- and output-privacy in one cryptographic mechanism. In a MIFE scheme, a setup authority can generate restricted decryption keys which enable to learn specific functions of encrypted messages, without revealing any additional information. To achieve differential privacy in this process, we introduce as a new cryptographic primitive: noisy multi-input functional encryption (NMIFE). It extends the concept of MIFE such that the decryption key may also encode a *noisy* function where the noise value is secret.

While the change from MIFE to NMIFE is rather straightforward, the challenge is to come up with precise and workable definitions of correctness and security definition that we propose and explain in this work. Here, the security definition is tailored to the use case of differential privacy. As it is a special case of the established notion of full-hiding security, we present a generic transformation that allows to turn any full-hiding MIFE scheme into a secure NMIFE scheme that has practically the same performance as the initial MIFE scheme.

Moreover, we make use of the fact that the proposed security definition is less restrictive and present a new concrete NMIFE scheme for evaluating the inner product. It is dubbed DiffPIPE (short for DIFFerentially Private Inner Product Evaluation). DiffPIPE is not the result from the transformation and outperforms all from existing full-hiding MIFE schemes constructed NMIFE schemes. In experiments, we demonstrate its applicability for realizing privacy preserving counting queries on data sets.

KEYWORDS

Privacy preserving analysis, differential privacy, functional encryption

Proceedings on Privacy Enhancing Technologies 2024(2), 509–530
© 2024 Copyright held by the owner/author(s).
https://doi.org/10.56553/popets-2024-0061

1 INTRODUCTION

For numerous applications, the analysis of massive data sets leads to valuable improvement in various parts of research, for example in medicine. A typical scenario can be generalized as follows: An analyst is interested in evaluating a function f on data $X = (x_1, \ldots, x_n)$. For this purpose, there is a computation process Π that receives as inputs x_i from different users i and a function f from the analyst and returns $f^*(X)$ to the latter (see Figure 1).

A major challenge is to realize the trade-off between correctness and privacy. On the one hand, it is in the interest of the analyst that $f^* = f$ or at least as close as possible. On the other hand, this process should not compromise the privacy of the users. This is motivated by various use cases such as the census data survey or patients that want to participate in medical research, without revealing too much sensitive information.

Modern cryptography features several mechanisms for potentially resolving this conflict partially. These can roughly be divided into two categories: based on multi-party computation and based on encryption. Note that these mechanisms only protect the *inputs* to the process Π, i.e., X in our case. However, it is well known that also the *output* of this process, namely $f^*(X)$, may leak information about X as well. To mitigate this risk, a common practical solution is to make the computation differential private. A popular approach for realizing differential privacy (DP) is to replace the deterministic numeric function f by a noisy counterpart, i.e., $f^* = f + \nu$ for some noise value ν. This paper focuses on the question how a mechanism for input privacy can be combined with DP.

A straightforward solution would be the use of multi party computation (MPC). However, as we elaborate in Section 3, some properties of this approach such as the synchronous interaction between several computing parties, can be problematic in some cases. In such cases, non-interactive solutions based on processing encrypted data can be beneficial.

While literature discusses several approaches based on homomorphic encryption (HE), the use of functional encryption (FE) has not been investigated so far. As opposed to the case of HE, FE has the advantage that the computation of f^* does *not* need to be outsourced to another party that is independent of the analyst. For this and further reasons that we explain in Section 3, we introduce the concept of differential private computation on encrypted data using FE. More precisely, our contributions are as follows:

- We introduce and formalize the notion of *noisy* multi-input functional encryption (NMIFE). This includes a tailored security definition that adopts the concepts of message- and function-hiding and restricts these accordingly for the DP scenario as follows:
  - The analyst has access to the encryption of a single set of user data.

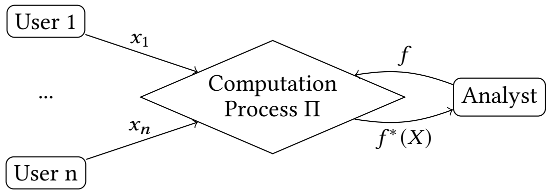


Figure 1: The generic scenario with $X = (x_1, \ldots, x_n)$ and >f^* being an appropriate noisy variant of f.

  - The decryption keys realize noisy functions where only the noise part is hidden.

  Consequently, we refer to it as single-message-and-noise-hiding (SMN-H).
- We describe a generic construction for a SMN-H NMIFE from a full-hiding multi input functional encryption (MIFE) scheme. Actually, the resulting NMIFE scheme fulfills an even stronger security notion where multiple message challenges are possible (referred to as message-and-noise-hiding (MN-H)).
- We build a SMN-H NMIFE scheme for linear functions, dubbed DiffPIPE, based on a full-hiding MIFE scheme for linear functions by Datta et al. [14] that allows for smaller elements. Experiments confirm the improved efficiency and the applicability of the scheme for privacy-preserving analysis with input- and output-privacy.

We emphasize that our paper represents basic research. More precisely, we present a new cryptographic primitive that has properties that do not yet exist. Although there are real use cases for this (see Section 4) and although such schemes can already be realized for linear functions, we consider research to be still in its early stages and the efficiency and functionality of such schemes still needs to be further increased (similar to research on functional encryption in general).

The paper is structured as follows. Section 2 provides technical preliminaries for the follow-up sections. In Section 3, we formulate the considered problem statement and requirements and discuss related work. Section 4 presents an overview, necessary definitions and open challenges how FE can be used to address this problem. In Section 5, we introduce the concept of noisy multi input functional encryption (NMIFE) and formalize correctness and security properties. A generic transformation from certain MIFE schemes into a NMIFE scheme is presented in Section 6. It includes the proof of correctness and security of the constructed NMIFE scheme. In Section 7 we build a concrete SMN-H NMIFE scheme named DiffPIPE and analyze its efficiency over real medical data sets. Section 8 concludes the paper and presents open questions for future work.

2 PRELIMINARIES

2.1 Notations

Let λ define the security parameter. $[n]$ denotes the set $\{1, \ldots, n\}$ for any $n \in \mathbb{N}$, where $\mathbb{N}$ defines the set of all positive integers and $\mathbb{Z}$ the set of all integers. For any prime $q \in \mathbb{Z}$, the term $\mathbb{F}_q$ refers to the finite field of of size q. Note that there is a natural assignment between $\mathbb{F}_q$ and the set of integers modulo q. We often use this relation implicitly. The all zero vector in $\mathbb{F}_q^m$ is denoted by $\vec{0}^m$. $\vec{x}$ denotes a vector, $\vec{x}^j$ the j^{th} element of the vector $\vec{x}$. $\vec{v} \cdot \vec{w}$ stands for the inner product of the vectors $\vec{v}, \vec{w}$. For a function $f : X_1 \times \cdots \times X_n \rightarrow Y$ and some $v \in Y$, we denote by $(f + v)(\vec{x}) := f(\vec{x}) + v$ for all $\vec{x} \in X_1 \times \cdots \times X_n$. For a subset $\Delta \subseteq Y$, we define $f + \Delta = \{f + v | v \in \Delta\}$. For a distribution $\mathbb{D}$ over some set Δ, $\Pr[x \overset{\mathbb{D}}{\leftarrow} \Delta]$ denotes the probability that x is sampled according to distribution $\mathbb{D}$ over Δ. For any set S, $s \overset{\$}{\leftarrow} S$ represents the process of uniformly sampling an element $s \in S$. We use the abbreviation PPT to mean probabilistic polynomial time. A function $\texttt{negl} : \mathbb{N} \rightarrow \mathbb{R}^+$ is said to be *negligible* if for every $c \in \mathbb{N}$, there exists a $\tau \in \mathbb{N}$ such that for all $x \in \mathbb{N}$ with $x > \tau$, it holds that $|\texttt{negl}(x)| < 1/x^c$.

2.2 Differential Privacy

DP is an established technique to achieve output privacy [17, 18, 29, 40]. In DP we have one party, the curator, that holds a set of sensitive data $X = (x_i)_{i \in [n]}$ and wants to make analysis on this data possible, while limiting the disclosure of private information of records that are in the data set. On request of a function or algorithm f from the analyst, the curator performs the evaluation of a corresponding differential private function or algorithm f^* and sends the result back to the analyst. Roughly an algorithm is differential private if an analyst seeing only the output of the algorithm cannot distinguish if a particular record was in the data set, that the algorithm was evaluated on, or was missing from the data set. A simple and common way to produce differential private numeric functions, especially in the setting of privacy-preserving analysis, is to perturb the function output with noise sampled through special distributions, $f^*(X) = f(X) + v$. In general, the noise is dependent on how much the function reveals about the input data and what privacy level we want to achieve. Since the analyst can make multiple requests on the same data set, it may learn additional information through combining the results. DP is still achieved if the noise is carefully handled. Therefore, when designing a DP system, normally the curator tracks a privacy budget for the analyst. The curator allows a series of questions whose total impact is not more than the whole privacy budget. For more technical details we refer to Appendix A.

3 PROBLEM STATEMENT AND RELATED WORK

The purpose of this section is to discuss benefits and possible shortcomings of existing cryptographic mechanisms[1] for realizing the use case of privacy preserving computation as displayed in Figure 1 where f^* is a noisy variant of f with an appropriately chosen noise. The requirement for input-privacy implies that parties executing Π (and no other party) do not receive the user inputs x_i directly. Moreover, to achieve output-privacy only the analyst should see the output $f^*(X)$ directly but should know nothing about the noise used in f^* despite its distribution.

[1]Note that we consider solutions based on trusted execution environments as out of scope as it is hardly possible to analyze their security and hence require a different trust model than plain cryptographic solutions.

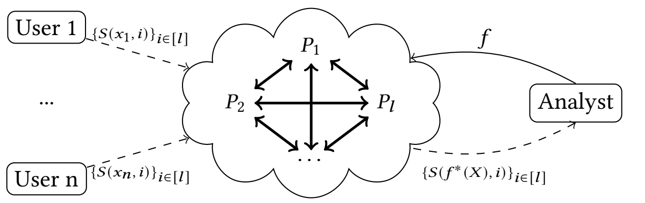

User 1
$\{S(x_1, i)\}_{i\in[l]}$
...
User n
$\{S(x_n, i)\}_{i\in[l]}$
P_1
P_2
P_l
...
f
Analyst
$\{S(f^*(X), i)\}_{i\in[l]}$

Figure 2: Differentially private multi party computation. $S(\cdot, i)$ denotes the ith share.

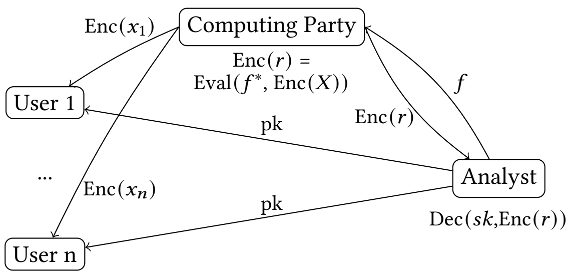


Figure 3: Differentially private homomorphic encryption, scenario 1.

The two most common cryptographic techniques to realize data analysis without revealing the data are multi party computation (MPC) and homomorphic encryption (HE).

Multi-Party Computation. In MPC [7, 21, 42, 48], a set of parties jointly compute a function over their inputs while keeping those inputs private from other parties. Consequently, it is nowadays a widely used technique to securely evaluate machine learning algorithms in the context of privacy-preserving analysis [2, 30, 45].

A typical application of MPC to realize Figure 1 is the following. We assume a set of computing parties $\{P_1, P_2, \ldots\}$ that interactively compute Π. At the beginning, each user i secretly shares its input data x_i to the computing parties of the MPC protocol. The parties then evaluate a predetermined function on the shared inputs over encrypted channels. The security of the MPC protocol ensures that the computing parties do not learn anything about the input data. Depending on the security model, the parties need to behave accordingly to the protocol or may be dishonest. The result is either disclosed to all computing parties, or in our use case more suitable, is forwarded again as shares to another party, e.g. the analyst. Only the analyst can combine the shares to achieve the result. An overview of this process is given in Figure 2, where $S(\cdot, i)$ stands for the ith share. To combine MPC with DP, for example the computing parties also need to add noise to the function result of f [8, 9, 19, 34, 35]. A main advantage of MPC protocols is, that almost any function that can be expressed as a sequence of addition and multiplications is very efficiently computable in contrast to HE or FE. But this comes with the cost of communication between the computing parties and requires a communication network with strong delivery guarantees. Moreover, the computation overhead is on the side of the computing parties and not the analyst. Each time the analyst wishes to evaluate a function, the computing parties that hold shares of the input data need to participate in the computation. In addition, it is necessary to trust that a certain fraction of computing parties is honest.

Homomorphic Encryption. HE schemes [3, 12, 20] allow to encrypt data in such a way that certain computations on the plaintext data can be executed on the encrypted data instead so that all intermediate results, including the final output, remain encrypted the whole time. Consequently, there exist several works using HE to realize privacy preserving analysis, where the concern lies on input privacy [22, 36, 37].

Similar to the case of MPC, the standard scenario assumes a computing party that executes the computation on the (encrypted) user inputs on behalf of the analyst (see Figure 3). That is, the analyst sends its public key to the users, who then encrypt their data x_i and send them to the computing party. Likewise, the analyst sends its function f to the computing party, who then can perform the analysis on the encrypted data, getting an encrypted result $Enc(r) = Enc(f(x_1, \ldots, x_n))$. The result can then be decrypted only by the analyst with its secret key, without being able to decrypt single data records x_i.

It is necessary that the computing party chooses and integrates the noise into the computation. One approach would be to sample the noise directly in the encrypted domain, hence preventing the computing party to learn the noise value ν. The problem is that sampling algorithms are computational expensive and, at the moment, HE is not fast enough to ignore the costs of the computational complexity. This leaves only the option that the computing party samples ν in plaintext and adapts the encrypted result accordingly. In general, existing work ([5, 38]) assume the computing party to act as curator, that is on the one hand collecting the encrypted data, on the other hand keeping track of the privacy budget and make sure, that only differential private functions are evaluated achieving input privacy and output privacy.

Zorarpacı and Özel [49] use a slightly different scenario. Next to the analyst that wants to compute a function f and the computing party that evaluates f on the encrypted data, they introduce an additional party, called key holder, that on the one hand holds the secret key and distributes the public key, and on the other hand ensures differential privacy by adding noise after decrypting r. Although this resolves the problem that the analyst should not be able to control the noise, the key holder now observes the output $f^*(X)$ directly.

Note that all these approaches suffer from the same problem as the MPC approach, namely that the computation overhead is on the side of the computing parties and not the analyst. Hence, the analyst relies on the availability of computational power of external parties. An alternative HE based solution could be the one displayed in Figure 4. As opposed to the previous approach, the computation is done directly by the analyst. Instead of requiring a separate computing party, now an authority is present that generates the public key and decrypts the result of the computation. However, the authority sees the output $f^*(X)$ directly and the analyst can possibly control the choice of f^*.

Conclusion. While both MPC and HE can be used to realize input- and output-privacy, they also have in common that if nobody except of the analyst should learn $f^*(X)$ while the analyst must not

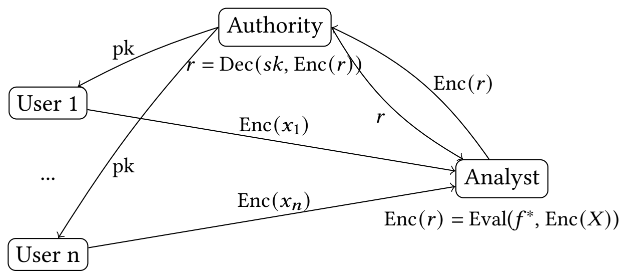

Authority
pk
$r = \mathrm{Dec}(sk, \mathrm{Enc}(r))$
Enc(r)
User 1
Enc(x_1)
r
...
pk
Analyst
Enc(x_n)
Enc(r) = Eval(f^*, Enc(X))
User n

Figure 4: Differentially private homomorphic encryption, scenario 2.

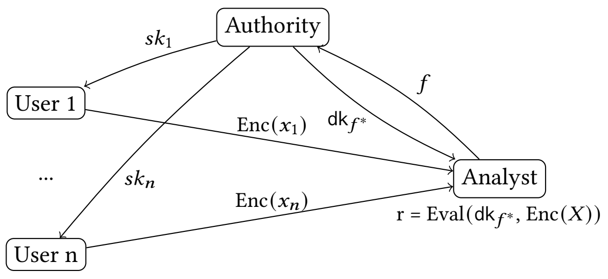


Figure 5: Differentially private multi-input functional encryption.

control the noise of f^*, the computation of $f^*(X)$ needs to be outsourced. That is the computation overhead is on the side of the computing parties and not the analyst and that the analyst relies on the availability and honesty of external parties. Depending on the concrete use case, this may be problematic. In the next section, we discuss an alternative approach based on functional encryption that likewise offers input- and output-privacy but where the computation does not need to be outsourced to third parties.

4 PRIVACY PRESERVING ANALYSIS WITH FUNCTIONAL ENCRYPTION

4.1 Overview

Similar to HE, FE [10] allows to compute on encrypted data. The main difference is that the computing party sees directly the result of the computation without the need to request another party for decryption first. Thus, as opposed to the approaches discussed in Section 3, the computation overhead is on the side of the analyst who does not rely on the availability of computational power of external parties.

This is accomplished by supporting restricted decryption keys which enable to learn specific functions of encrypted messages, without revealing any additional information. This involves a setup authority which holds a master secret key and publishes (on request) decryption keys dk_f that allow to compute $f(X)$ (and nothing else) for encrypted inputs X.

A formal definition of FE is the following:

Definition 4.1 (Multi-Input Functional Encryption [1]). Let $\{\mathcal{F}_n\}_{n\in\mathbb{N}}$ be an ensemble where each $\mathcal{F}_n$ is a family of n-ary functions. A function $f \in \mathcal{F}_n$ is defined as follows $f : \mathcal{X}_1 \times \cdots \times \mathcal{X}_n \rightarrow \mathcal{Y}$. A private key multi-input functional encryption scheme MIFE for $\mathcal{F}$ consists of the following four algorithms:

- $\mathtt{Setup}(\lambda, \mathcal{F}_n)$**:** On input the security parameter λ and a description of $\mathcal{F}_n \in \mathcal{F}$, outputs a public parameter $\mathtt{pp}$, a master secret key $\mathtt{msk}$ and a secret key $\mathtt{sk}_i$ for each slot $i \in [n]$. The other algorithms implicitly take $\mathtt{pp}$.
- $\mathtt{Enc}(\mathtt{sk}_i, i, x_i)$**:** On input the secret key $\mathtt{sk}_i$ for slot $i \in [n]$, and a message $x_i \in \mathcal{X}_i$, outputs a ciphertext $\mathtt{ct}$. We assume that each ciphertext has an associated index i, which denotes what slot this ciphertext can be used for.
- $\mathtt{KeyGen}(\mathtt{msk}, f)$**:** On input the master secret key $\mathtt{msk}$ and a function $f \in \mathcal{F}_n$, outputs a decryption key $\mathtt{dk}$.
- $\mathtt{Dec}(\mathtt{dk}, \mathtt{ct}_1, \ldots, \mathtt{ct}_n)$**:** on input a decryption key $\mathtt{dk}$ and n ciphertexts, outputs a $y \in \mathcal{Y}$ or a symbol $\perp$.

A solution based on a MIFE scheme could look as follows. Multiple users can encrypt their data x_i with an encryption key $\mathtt{sk}_i$, provided by an authority[2]. The users provide their encrypted data to the analyst. The analyst can request function keys dk_f in regard to a function f and gets as result a decryption key dk_{f^*} where $f^* = f + \nu$ with ν being some noise value sampled by the authority, see Figure 5. That is the analyst can compute the noisy result $r = \mathtt{Dec}(\mathtt{dk}_{f^*}, \mathtt{ct}_1, \ldots, \mathtt{ct}_n) = f(x_1, \ldots, x_n) + \nu$. The authority can ensure, that only differentially private functions are evaluated and keeps track of the privacy budget [13]. Note that this approach satisfies input- and output-privacy and keeps the evaluation effort to the side of the analyst. In particular, there is neither the need to rely on the availability of computing parties nor to trust these.

Another advantage is that FE can enable a scenario in which a data set is provided in encrypted form once. Theoretically even the noisy decryption keys for relevant or allowed functions can be uploaded beside the encrypted data set. The authority only has a one time effort of setting up the scheme and generating the decryption keys. Since the computation itself lies with the analyst, no third party needs to be involved after the setup. This most closely reflects the current situation, in which any researcher can directly use data from public databases for research.

Furthermore the combination of FE and DP can be helpful to achieve the training of machine learning algorithms using FE. Current MIFE schemes are limited in their functionality, basically either allowing to compute inner products [15, 44] or quadratic functions [4, 16, 39]. Most of the works combining privacy preserving analysis with FE [33] concentrate on evaluating machine learning algorithms [23, 28, 39]. To train a complex algorithm using only limited functions, e.g. linear or quadratic, on the input data means to compute only intermediate results on the encrypted data and then use them for further computation, yielding a lot of decryption key queries to train an algorithm. The bigger problem is that intermediate results leak information about the input data [11, 39] and therefore need be protected, for example with DP.

[2]In some schemes it is also possible to have multiple authorities. This resolves the issue of trust in the authority, but we will not go into more detail in this section.

4.2 Challenges

To the best of our knowledge, the work of Bakas and Michalas [6] is the only related work that also investigates the combination of FE with DP. However, they do not provide any formal definition, in particular lacking a concise security model and analysis. Actually, we consider coming up with these as the main challenges here.

In general a NMIFE should be correct and secure. In the following, we recall corresponding definitions for MIFE and explain why these cannot be applied directly to the case of NMIFE.

Correctness. Correctness of an MIFE essentially means that running the decryption operation Dec with a decryption key dk_f outputs the evaluation of f on the encrypted inputs.

Definition 4.2 (Correctness of MIFE). The scheme MIFE (Definition 4.1) is correct if for all $f \in \mathcal{F}_n$, all $x_i \in \mathcal{X}_i$ for $1 \le i \le n$, we have

$$\Pr\begin{bmatrix}(\mathsf{pp}, \mathsf{msk}, \{sk_i\}_{i\in[n]}) \leftarrow \mathtt{Setup}(\lambda, \mathcal{F}_n),\\ \mathsf{dk}_f \leftarrow \mathtt{KeyGen}(\mathsf{msk}, f) :\\ \mathtt{Dec}(\mathsf{dk}_f, \mathtt{ct}_1, \dots, \mathtt{ct}_n) = f(x_1, \dots x_n)\end{bmatrix} = 1 \qquad (1)$$

with $\mathtt{ct}_i = \mathtt{Enc}(\mathsf{sk}_i, i, x_i)$ where the probability is taken over the coins of Setup, KeyGen and Enc.

In the case of NMIFE, the analyst receives a decryption key for a noisy variant f^* of f where the noise is sampled according to some chosen, publicly known distribution. Thus, the condition that the equality displayed at the end of Equation (1) has to hold for sure is not applicable anymore.

Security. When using FE as explained in Section 4.1, two security requirements are obvious. First, the analyst should not learn anything about the user data x_i from its encryption $\mathtt{ct}_i = \mathtt{Enc}(\mathsf{sk}_i, i, x_i)$. This property is usually referred to as message hiding. Second, it is important that the analyst cannot learn anything about the noise v from the decryption key dk_{f^*}. This is covered by the notion of function-hiding which requires that a decryption key dk_f leaks nothing about the underlying function f.

The security notion of FE that covers both is full-hiding security [15]. For such a scheme, an attacker can neither distinguish between different ciphertexts $(x_{i,0}^{j_i}, x_{i,1}^{j_i})$ nor between different functions $(f_{l,0}, f_{l,1})$. That is, both the content of encrypted messages as well as the concrete functions encoded in a decryption key dk are hidden to an adversary. A formal definition is the following:

Definition 4.3 (Full-Hiding Security of MIFE). The full-hiding security notion for a private key MIFE scheme is formalized through the experiment $\mathsf{Expt}_{\mathcal{A}}^{MIFE}(\beta)$ for random $\beta \leftarrow \{0,1\}$, which involves a PPT adversary $\mathcal{A}$ and a PPT challenger C. The experiment involves three phases in the following order: a setup phase, a query phase, and a guess phase.

Setup phase: C generates $(\mathsf{pp}, \mathsf{msk}, \{sk_i\}_{i\in[n]})$ by invoking $\mathtt{Setup}(\lambda, \mathcal{F}_n)$, providing pp to $\mathcal{A}$, and sampling $\beta \overset{\$}{\leftarrow} \{0,1\}$.

Query Phase: $\mathcal{A}$ is allowed to adaptively make any polynomial number of queries of the following two types in arbitrary order:

- Decryption key query: In response to the l^{th} decryption key query of $\mathcal{A}$ corresponding to a pair of functions $(f_{l,0}, f_{l,1}) \in \mathcal{F}_n \times \mathcal{F}_n$, C forms a decryption key $\mathsf{dk}_l \leftarrow \mathtt{KeyGen}(\mathsf{msk}, f_{l,\beta})$ and hands dk_l to $\mathcal{A}$.
- Ciphertext query: To answer a ciphertext query of $\mathcal{A}$ for the i^{th} index corresponding to a pair $(x_{i,0}^{j_i}, x_{i,1}^{j_i})$, C prepares a ciphertext $\mathtt{ct}_i^j \leftarrow \mathtt{Enc}(\mathsf{sk}_i, i, x_{i,\beta}^{j_i})$ and gives $\mathtt{ct}_i^j$ to $\mathcal{A}$.

Let the total number of decryption key queries made by $\mathcal{A}$ be $q_{key}(\ge 0)$ and the total number of ciphertext queries made for the i^{th} index be $q_{\mathtt{ct},i}(\ge 0)$. The restriction on the queries of $\mathcal{A}$ are that if $q_{\mathtt{ct},i} \ge 1$ for all $i \in [n]$, then for all $l \in [q_{key}]$ and for all $(j_1, \dots, j_n) \in [q_{\mathtt{ct},1}] \times \dots \times [q_{\mathtt{ct},n}]$, we must have

$$f_{l,0}(x_{1,0}^{j_1}, \dots, x_{n,0}^{j_n}) = f_{l,1}(x_{1,1}^{j_1}, \dots, x_{n,1}^{j_n}) \qquad (2)$$

Guess: $\mathcal{A}$ eventually outputs a guess bit, which is the output of the experiment.

A private key MIFE scheme is said to be full-hiding if for any PPT adversary $\mathcal{A}$ and for any security parameter λ, the advantage of $\mathcal{A}$ in the above experiment is negligible.

$$\begin{aligned}\mathsf{Adv}_{\mathcal{A}}^{MIFE}(\lambda) = |Pr[\mathsf{Expt}_{\mathcal{A}}^{MIFE}(0) = 1]\\ -Pr[\mathsf{Expt}_{\mathcal{A}}^{MIFE}(1) = 1]| \le negl(\lambda)\end{aligned}$$

While full-hiding security is sufficient for realizing the solution explained in Section 4.1, we argue that it is actually too strong. For example, the function f is chosen by the analyst. Thus, it is not necessary that dk_{f^*} with $f^* = f + v$ does not leak any information but only needs to hide the noise value v. Therefore, we are going to present a new security notion (Definition 5.3) that is adapted to the solution explained in Section 4.1. As we are going to show in Section 7, this allows for new constructions with a better efficiency.

5 NOISY MULTI-INPUT FUNCTIONAL ENCRYPTION

In this section, we introduce a new variant of MIFE, being noisy multi input functional encryption (NMIFE). It allows to combine the benefits of functional encryption (Section 4) and differential privacy (Section 2.2). That is, one can realize a scenario where the analyst performs the computation of some function f on user inputs x_i non-interactively, such that these values are kept hidden and the authority can control how much the output of the computation leaks. In a nutshell, NMIFE extends the classic notion of MIFE by incorporating the possibility that a decryption key dk realizes a *noisy* variant of a given function f. That is, the decryption key does not encode f but $f + v$ where v is a noise value sampled according to some chosen distribution $\mathbb{D}$. For example, one may pick a distribution, so that $(\epsilon, 0)$-DP is achieved (see Appendix A).

In the following, we formally define NMIFE (Definition 5.1) and the corresponding notion of correctness (Definition 5.2). Afterwards, we introduce a security notion tailored to NMIFE schemes. These definitions build on the definitions recalled in Section 4. To help readability, we not only explain the differences but also mark these in the definitions by boxes.

The difference to MIFE (cf. Definition 4.1) affects the generation of decryption keys which takes as additional input also a distribution. The full formal definition is as follows:

Definition 5.1 (Noisy Multi-Input Functional Encryption). Let $\{\mathcal{F}_n\}_{n\in\mathbb{N}}$ be an ensemble where each $\mathcal{F}_n$ is a family of n-ary functions. A function $f \in \mathcal{F}_n$ is defined as follows $f : \mathcal{X}_1 \times \cdots \times \mathcal{X}_n \to \mathcal{Y}$. A noisy multi-input functional encryption scheme NMIFE for $\mathcal{F}$ consists of the following algorithms:

$\mathsf{Setup}(\lambda, \mathcal{F}_n)$**:** On input the security parameter λ and a description of $\mathcal{F}_n \in \mathcal{F}$, outputs a public parameter pp, a master secret key msk and a secret key sk_i for each slot $i \in [n]$. All of the remaining algorithms implicitly take pp.

$\mathsf{Enc}(\mathsf{sk}_i, i, x_i)$**:** On input the secret key sk_i for slot $i \in [n]$, and a message $x_i \in \mathcal{X}_i$, outputs a ciphertext ct_i. We assume that each ciphertext has an associated index i, which denotes what slot this ciphertext can be used for.

$\mathsf{KeyGen}(\mathsf{msk}, f, \boxed{\mathbb{D}})$**:** On input the master secret key msk, a function $f \in \mathcal{F}_n$ and a $\boxed{\text{distribution } \mathbb{D} \text{ over some } \Delta \subseteq \mathcal{Y}}$ $\boxed{\text{such that } f + \Delta \in \mathcal{F}_n, \text{ sample } v \overset{\mathbb{D}}{\leftarrow} \Delta}$ and outputs a decryption key dk.

$\mathsf{Dec}(\mathsf{dk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_n)$**:** On input a decryption key dk for function f and n ciphertexts, outputs a $y \in \mathcal{Y}$ or a symbol $\perp$.

Recall that correctness of a MIFE is defined as follows: given a decryption key $\mathsf{dk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f)$, one can run $\mathsf{Dec}(\mathsf{dk}_f, \ldots)$ to get the evaluation of f on encrypted inputs. Correctness of NMIFE is defined analogously with the difference that a noisy variant of f is evaluated. More precisely, given $\mathsf{dk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f, \mathbb{D})$, one can run $\mathsf{Dec}(\mathsf{dk}_f, \ldots)$ to get the evaluation of $f + v$ on encrypted inputs with v being sampled according to $\mathbb{D}$.

Definition 5.2 (Correctness of NMIFE). The scheme NMIFE (Definition 5.1) is correct if for all $f \in \mathcal{F}_n$, for all $x_i \in \mathcal{X}_i$, for $1 \leq i \leq n$, $\boxed{\text{for all distributions } \mathbb{D}_f \text{ over some set } \Delta \subseteq \mathcal{Y} \text{ with } f + \Delta \in \mathcal{F}_n,}$ $\boxed{\text{and for all } v \in \Delta}$ we have

$$\Pr\left[\begin{array}{c}(\mathsf{pp}, \mathsf{msk}, \{sk_i\}_{i\in[n]}) \leftarrow \mathsf{Setup}(\lambda, \mathcal{F}_n),\\ \mathsf{dk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f, \boxed{\mathbb{D}_f}) :\\ \mathsf{Dec}(\mathsf{dk}_f, \mathsf{ct}_1, \ldots, \mathsf{ct}_n) = f(x_1, \ldots x_n)\boxed{+v}\end{array}\right] = \boxed{\Pr[v \overset{\mathbb{D}_f}{\leftarrow} \Delta]}$$

with $\mathsf{ct}_i = \mathsf{Enc}(\mathsf{sk}_i, i, x_i)$ where the probability is taken over the coins of Setup, KeyGen and Enc.

Note that if we restrict all distributions $\mathbb{D}_f$ to be the all-zero distribution, i.e., that outputs only 0, then a correct NMIFE scheme becomes a MIFE scheme. This shows that the introduced definitions are extensions of the definitions in Section 4.

With respect to defining security of an NMIFE, we likewise aim to build on the existing definition of full-hiding security (Definition 4.3). One (straightforward) approach is to incorporate into the decryption key queries that an attacker can specify a function f *and* a distribution $\mathbb{D}$. Of course, the condition specified in Equation (2) would have to be adapted accordingly. As the distributions $\mathbb{D}$ can be chosen arbitrarily, including the all zero distribution, an attacker would not be restricted compared to the attacker considered for full-hiding security. In other words, this security definition would be (at least) as strong as full-hiding security.

In the following, we introduce two alternatives for more relaxed notion of security (Definition 5.3). To tailor the security definition more towards the DP scenario, we assume a fixed set of messages, meaning for each slot i there exists only one single message-pair $(x_{i,0}, x_{i,1})$. Additionally instead of requiring full function-hiding, the proposed definition relaxes it to noise-hiding. That is, instead of requiring that the decryption key does not leak the encoded function in its entirety, we assume that the "base" function f is known anyhow and only the noise value v needs to be protected. We dub this security definition as single-message-and-noise-hiding (SMN-H) security. The reasons that this more relaxed security definition is relevant are twofold: First, the security definition is more tailored towards the DP scenario. There, the analyst chooses the function f but eventually learns the evaluation of $f + v$ for an unknown noise value v on a prefixed data set X. There is no need to demand that multiple messages can be encrypted per slot or that the full function is hidden as part of the function is known anyway. Second, a relaxed (but still sufficient) security requirement allows for new or improved designs that were not possible for full-hiding schemes. In Section 7 we improve an existing design to build a SMN-H secure NMIFE scheme, named DiffPIPE, and show that it is more efficient.

Moreover, we extend our security definition to allow multiple messages, referring to it as message-and-noise-hiding (MN-H) security. We show how a full-hiding MIFE can be turned into a NMIFE that provides MN-H security. That is, going for the stronger security definition is still an option.

Next, we provide the formal security definitions. As before, boxes highlight the differences to Definition 4.3. While the meaningfulness of most changes are easy to see, Equation (3) may require some additional explanation. Recall that for a MIFE scheme, Equation (2) in Definition 4.3 prevents trivial attacks by assuring that the attacker only chooses *messages* and *functions* which do not reveal the chosen β simply by evaluating the functions on the plaintext and comparing the result to the encrypted evaluation. Similarly, this is assured for NMIFE schemes by Equation (3) in Definition 5.3 by restricting the adversary to *messages* and *noise pairs* that do not reveal anything beyond the expected result. Note that for the decryption key query, the adversary only queries one function but two noise values.

Definition 5.3 ((Single-)Message-and-Noise-Hiding Security). The (single-)message-and-noise-hiding security notion for a private key NMIFE is formalized through the experiment $\mathsf{Expt}^{NMIFE}_{\mathcal{A}}(\beta)$, for random $\beta \overset{\$}{\leftarrow} \{0, 1\}$, which involves a PPT adversary $\mathcal{A}$ and a PPT challenger C. The experiment involves three phases as follows:

Setup phase: C generates $(\mathsf{pp}, \mathsf{msk}, \{sk_i\}_{i\in[n]})$ by invoking $\mathsf{Setup}(\lambda, \mathcal{F}_n)$, providing pp to $\mathcal{A}$, and sampling $\beta \overset{\$}{\leftarrow} \{0, 1\}$.

Query Phase: $\mathcal{A}$ is allowed to adaptively make any polynomial number of queries of the following two types in arbitrary order:

- Decryption key query: In response to the l^{th} decryption key query of $\mathcal{A}$ corresponding to a function $f_l \in \mathcal{F}_n$ $\boxed{\text{and two noise values } v_{l,0}, v_{l,1} \in \Delta \text{ for some } \Delta \subseteq}$ $\boxed{\mathcal{Y} \text{ such that } f_l + \Delta \in \mathcal{F}_n}$, C forms a decryption key

$$\mathsf{dk}_l \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f_l, \mathbb{D}_{l,\beta})$$

$\boxed{\text{where } \mathbb{D}_{l,\beta} \text{ describes the distribution that outputs}}$

$\boxed{v_{l,\beta} \text{ with probability } 1}$ and hands dk_l to $\mathcal{A}$.

- Ciphertext query: To answer a ciphertext query of $\mathcal{A}$ for the i^{th} index corresponding to a pair $(x^{j_i}_{i,0}, x^{j_i}_{i,1})$, C prepares a ciphertext $\mathsf{ct}^j_i \leftarrow \mathsf{Enc}(\mathsf{sk}_i, i, x^{j_i}_{i,\beta})$ and gives ct^j_i to $\mathcal{A}$.

Let the total number of decryption key queries made by $\mathcal{A}$ be $q_{key}(\geq 0)$ and the total number of ciphertext queries made for the i^{th} index be $q_{\mathsf{ct},i}(\geq 0)$. The restriction on the queries of $\mathcal{A}$ are that if $q_{\mathsf{ct},i} \geq 1$ for all $i \in [n]$, then for all $l \in [q_{key}]$ and for all $(j_1, \dots j_n) \in [q_{\mathsf{ct},1}] \times \dots \times [q_{\mathsf{ct},n}]$, we must have

$$\boxed{f_l(x^{j_1}_{1,0}, \dots, x^{j_n}_{n,0}) + v_{l,0} = f_l(x^{j_1}_{1,1}, \dots, x^{j_n}_{n,1}) + v_{l,1}.} \tag{3}$$

Guess: $\mathcal{A}$ eventually outputs a guess bit $\beta' \in \{0,1\}$, which is the output of the experiment.

A private key NMIFE scheme is said to be message-and-noise-hiding if for any PPT adversary $\mathcal{A}$, for any security parameter λ, the advantage of $\mathcal{A}$ in the above experiment is negligible. A private key NMIFE scheme is said to be single-message-and-noise-hiding if for any PPT adversary $\mathcal{A}$ that makes at most one ciphertext query per index, for all $i \in [n]$ it holds that $q_{\mathsf{ct},i} = 1$, for any security parameter λ, the advantage of $\mathcal{A}$ in the above experiment is negligible.

$$\begin{aligned}\mathsf{Adv}^{NMIFE}_{\mathcal{A}}(\lambda) = |\Pr[\mathsf{Expt}^{NMIFE}_{\mathcal{A}}(0) = 1] \\ - \Pr[\mathsf{Expt}^{NMIFE}_{\mathcal{A}}(1) = 1]| \leq negl(\lambda)\end{aligned}$$

6 BUILDING A NMIFE SCHEME FROM A MIFE SCHEME

The purpose of this section is to show how one may transform a full-hiding MIFE into a secure NMIFE scheme. We use this transformation to construct a concrete secure and correct NMIFE scheme in Appendix C. The transformation and properties are discussed in the following theorem. Note that for NMIFE to support function f, MIFE at least needs to support function $f + v$, $v \in \Delta$, for some Δ.

Theorem 6.1 (Construction). *Consider a MIFE scheme $\mathcal{S}$ with*

$$\mathcal{S} = (\mathcal{S}.\mathsf{Setup}, \mathcal{S}.\mathsf{Enc}, \mathcal{S}.\mathsf{KeyGen}, \mathcal{S}.\mathsf{Dec})$$

with parameters as explained in Definition 4.1 and $f + \Delta \in \mathcal{F}_n$ for some Δ. We build an NMIFE scheme $\mathcal{S}^*$ from $\mathcal{S}$ with

$$\mathcal{S}^* = (\mathcal{S}^*.\mathsf{Setup}, \mathcal{S}^*.\mathsf{Enc}, \mathcal{S}^*.\mathsf{KeyGen}, \mathcal{S}^*.\mathsf{Dec})$$

as follows. The setup, encryption, and decryption algorithms are the same as for $\mathcal{S}$, that is

$$\mathcal{S}^*.\mathsf{Setup} := \mathcal{S}.\mathsf{Setup}; \mathcal{S}^*.\mathsf{Enc} := \mathcal{S}.\mathsf{Enc}; \mathcal{S}^*.\mathsf{Dec} := \mathcal{S}.\mathsf{Dec}. \tag{4}$$

Only the KeyGen *algorithm is different and works as follows:*

$\mathcal{S}^*.\mathsf{KeyGen}(\mathsf{msk}, f, \mathbb{D})$: *On input the master secret key* msk, *a function $f \in \mathcal{F}_n$ and a distribution $\mathbb{D}$ over some $\Delta \subseteq \mathcal{Y}$ such that $f + \Delta \in \mathcal{F}_n$, samples $v \overset{\mathbb{D}}{\leftarrow} \Delta$ and outputs a decryption key*

$$\mathsf{dk} \leftarrow \mathcal{S}.\mathsf{KeyGen}(\mathsf{msk}, f^*) \tag{5}$$

with $f^* = f + v$.

The following two properties hold:

Correctness: *If $\mathcal{S}$ is correct, then $\mathcal{S}^*$ is correct as well.*
Security: *If $\mathcal{S}$ is full-hiding, then $\mathcal{S}^*$ is message-and-noise-hiding*

We want to give a short intuition of the proof here, along with an overview in Figure 6 and present the full proof in Appendix B. Whereas the correctness claim essentially follows directly from the definition, the security claim is shown by a standard reduction argument. That is, we assume a PPT attacker $\mathcal{A}^*$ against $\mathcal{S}^*$ with non-negligible advantage and construct a PPT attacker $\mathcal{A}$ against $\mathcal{S}$ that has the same non negligible advantage in the full-hiding security game. An overview of the reduction can be found in Figure 6. The only critical part is to ensure that the queries of $\mathcal{A}$ are valid with respect to condition (2), that is $f_{l,0}(x^{j_1}_{1,0}, \dots, x^{j_n}_{n,0}) = f_{l,1}(x^{j_1}_{1,1}, \dots, x^{j_n}_{n,1})$ for all $l \in [q_{key}]$ and for all $(j_1, \dots, j_n) \in [q_{\mathsf{ct},1}] \times \dots \times [q_{\mathsf{ct},n}]$.

This follows from the fact that the queries of $\mathcal{A}^*$ need to fulfill equation (3), that is for all $l \in [q_{key}]$ and for all $(j_1, \dots j_n) \in [q_{\mathsf{ct},1}] \times \dots \times [q_{\mathsf{ct},n}]$ it holds

$$f_l(x^{j_1}_{1,0}, \dots, x^{j_n}_{n,0}) + v_{l,0} = f_l(x^{j_1}_{1,1}, \dots, x^{j_n}_{n,1}) + v_{l,1}.$$

To keep the description simple, we directly state the noise values $v_{l,\beta}$ instead of the corresponding distributions $\mathbb{D}_{l,\beta}$. Given this, it follows that

$$\begin{aligned}f_{l,0}(x^{j_1}_{1,0}, \dots, x^{j_n}_{n,0}) &= f_l(x^{j_1}_{1,0}, \dots, x^{j_n}_{n,0}) + v_{l,0} \\ &= f_l(x^{j_1}_{1,1}, \dots, x^{j_n}_{n,1}) + v_{l,1} \\ &= f_{l,1}(x^{j_1}_{1,1}, \dots, x^{j_n}_{n,1})\end{aligned}$$

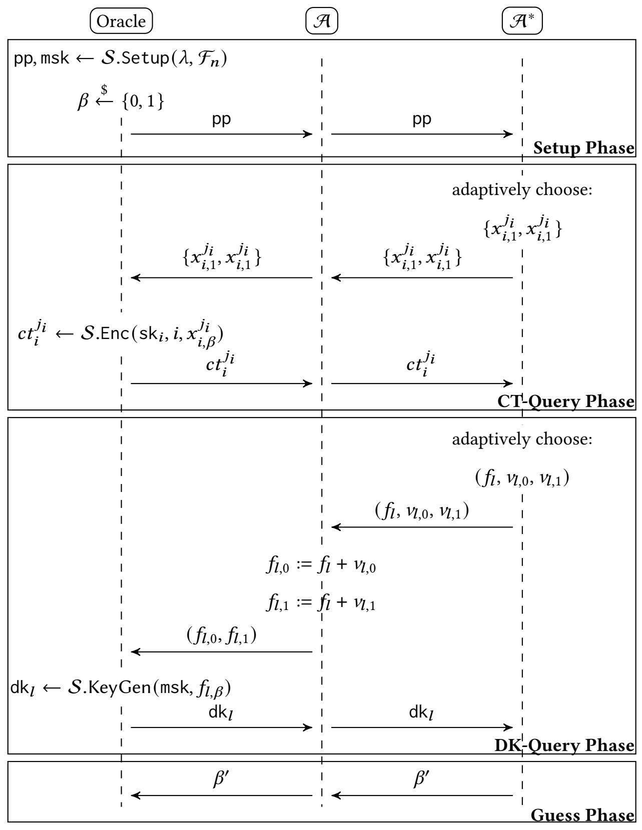
for all $l \in [q_{key}]$ and for all $(j_1, \dots, j_n) \in [q_{\mathsf{ct},1}] \times \dots \times [q_{\mathsf{ct},n}]$.

We can use this transformation on an existing MIFE scheme to construct a concrete NMIFE scheme that is MN-H secure. As starting point we take the multi-input inner product functional encryption (MIPE) scheme of Datta et al. [15], referring to it as *DOT* scheme, named after the authors Datta, Okamoto and Tomida. The choice is motivated by the fact that it is the only full-hiding MIFE so far. We first modify it to support *affine* functions i.e., $f_{y,c}(x_1, \dots, x_n) + c = c + \sum_{i \in [n]} y_i \cdot x_i$ with a constant value c such that correctness and full-hiding security are preserved This new scheme is dubbed *affine-DOT*. Then, we transform *affine-DOT* scheme into a NMIFE scheme according to the transformation presented above. Since the construction of *affine-DOT* is rather straight forward, the modifications to *DOT* and to security are described in Appendix C. In the following section, we depict a more advanced construction of a NMIFE scheme, DiffPIPE (short for differentially private inner product evaluation). It is SMN-H secure and more efficient than *noisy-DOT*.

7 A SINGLE-MESSAGE-AND-NOISE-HIDING NOISY MULTI-INPUT FUNCTIONAL ENCRYPTION SCHEME FOR INNER PRODUCTS

7.1 Overview

The goal of this section is to construct a concrete SMN-H NMIFE scheme dubbed DiffPIPE and to analyze its efficiency. To this end, we take the bounded full-hiding MIFE scheme for inner products of Datta et al. [15] as starting point and modify it into a scheme



Oracle
$\mathcal{A}$
$\mathcal{A}^*$
$\mathsf{pp}, \mathsf{msk} \leftarrow \mathcal{S}.\mathtt{Setup}(\lambda, \mathcal{F}_n)$
$\beta \overset{\$}{\leftarrow} \{0,1\}$
pp
pp
Setup Phase
adaptively choose:
$\{x^{j_i}_{i,1}, x^{j_i}_{i,1}\}$
$\{x^{j_i}_{i,1}, x^{j_i}_{i,1}\}$
$\{x^{j_i}_{i,1}, x^{j_i}_{i,1}\}$
$ct^{j_i}_i \leftarrow \mathcal{S}.\mathtt{Enc}(\mathsf{sk}_i, i, x^{j_i}_{i,\beta})$
$ct^{j_i}_i$
$ct^{j_i}_i$
CT-Query Phase
adaptively choose:
$(f_l, \nu_{l,0}, \nu_{l,1})$
$(f_l, \nu_{l,0}, \nu_{l,1})$
$f_{l,0} := f_l + \nu_{l,0}$
$f_{l,1} := f_l + \nu_{l,1}$
$(f_{l,0}, f_{l,1})$
$\mathsf{dk}_l \leftarrow \mathcal{S}.\mathtt{KeyGen}(\mathsf{msk}, f_{l,\beta})$
dk_l
dk_l
DK-Query Phase
β'
β'
Guess Phase

Figure 6: Security reduction used in the Proof of Theorem 6.1

that is a SMN-H NMIFE scheme for inner products. In the following, we refer to the scheme of Datta et al. [15] as *DOT* scheme. It is a function-hiding multi-input inner product functional encryption (FH-MIPE) scheme with security being based on the k-linear assumption (Definition 7.1). It supports polynomial number of encryption slots, incurring only polynomial loss in the security reduction and was the first and so far only practical and efficient MIFE scheme with function-hiding security. This FH-MIPE scheme operates over some finite field $\mathbb{F}_q$ with q being some prime number. The input data consists of vectors $\vec{x}_i \in \mathbb{F}_q^m$ for $i \in [n]$. The decryption keys dk are build based on a linear function f_y described through a concatenation of vectors $y = (\vec{y}_1 \parallel \cdots \parallel \vec{y}_n) \in \mathbb{F}_q^{n \cdot m}$ with $\vec{y}_i \in \mathbb{F}_q^m$. An evaluation of the decryption key $\mathsf{dk}_y \leftarrow \mathtt{KeyGen}(\mathsf{msk}, y, \mathbb{D})$ on ciphertexts $\mathtt{ct}_i \leftarrow \mathtt{Enc}(\mathsf{sk}_i, i, \vec{x}_i)$ yields the inner product: $\mathtt{Dec}(\mathsf{dk}, \mathtt{ct}_1, \ldots, \mathtt{ct}_n) = f_y(\vec{x}_1, \ldots, \vec{x}_n) = \sum_{i=1}^{n} \vec{x}_i \cdot \vec{y}_i$.

The design rationale behind DiffPIPE is as follows. As described in Appendix C *DOT* can be transformed into a scheme, that provides a full-hiding (FH) FE scheme for affine functions with no additional overhead or increase in size of elements. We refer to this scheme as *affine-DOT*. Then, Section 6 shows that we can use it to construct a MN-H scheme *noisy-DOT*, that has the same efficiency as *DOT*.

If we restrict our security definition to SMN-H security, we can construct a scheme with smaller elements. More concretely, we reduce the element sizes for ciphertexts from $2m + 2k + 1$ group elements to $m + 2k + 2$, where m is the number of attributes and n the number of records. The parameter k is associated with the k-LIN assumption and is independent of m and n. The size of the decryption key is reduced from $n(2m+2k+1)$ to $n(m+2k+2)$ group elements. Our experiments in Section 7.5 show that this allows for a better runtime compared to *noisy-DOT*.

The efficiency gain by reducing the size of ciphertexts by $m - 1$ and the keys by $n(m - 1)$ group elements depends strongly on the choice of the attribute number m. Additionally, DiffPIPE proves the conceptual differences between MIFE and NMIFE. In other words, reducing *DOT* or *affine-DOT* would not be possible without violating the standard security definitions. DiffPIPE as a NMIFE scheme comes with a new proof of security based on our new security definition (Definition 5.3).

In a nutshell, this reduction is achieved as follows. The ciphertexts of *DOT* (and hence *noisy-DOT*) contain a number of buffer slots that are used for the proof of security. Among these, $2m$ slots are reserved for the two different functions considered in the security definition of FH. As SMN-H considers one function only but two different noise values, we save m slots that are used to encode the second function, needing only one additional slot for the second noise value. While this allows to save elements, it also means that it is no longer possible to encode two different functions. That is DiffPIPE is not FH but, as we are going to prove, fulfills SMN-H

Note that the authors of *DOT* actually presented two schemes: a bounded scheme, where the number of encryption slots n is prior bounded and an unbounded scheme that can handle arbitrary number of encryption slots. Since we want to provide a scheme that is tailored for a privacy preserving analysis on a prefixed set of user data, we can limit us to the bounded variant, where the adversary makes one ciphertext query for each of the n encryption slots. Under this restriction, the scheme *DOT* used in its simplest form achieves full-hiding security. [3]

In the next sections, we present a more detailed description of DiffPIPE. As the construction is based on *DOT*, we omit some mathematical details and focus instead on the concepts that are necessary to understand our arguments. We refer to Datta et al. [15] for more details.

7.2 Mathematical Foundations

We make use of a bilinear pairing

$$e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T \tag{6}$$

with $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ being cyclic multiplicative groups, with generators $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$, each of size $q = |\mathbb{F}_q|$. The mapping e satisfies the following two properties:

- Bilinearity: $e(g_1^{\zeta}, g_2^{\eta}) = e(g_1, g_2)^{\zeta\eta}$ for all $\zeta, \eta \in \mathbb{F}_q$.
- Non-degeneracy: $e(g_1, g_2) \neq 1_{\mathbb{G}_T}$, where $1_{\mathbb{G}_T}$ denotes the identity element of the group $\mathbb{G}_T$.

Here, we implicitly identify elements in $\mathbb{F}_q$ with integers in $[0, q-1]$ as mentioned in Section 2.1.

For an integer $d \geq 1$, $\mathbb{V}_1 = \mathbb{G}_1^d$ and $\mathbb{V}_2 = \mathbb{G}_2^d$ are $\mathbb{F}_q$-vector spaces of dimension d. The bilinear pairing e in Equation (6) can be

[3] The modifications which are necessary for achieving full-hiding security, without the restriction that the adversary makes at least one ciphertext query, can be found in Datta et al. [15]

extended to a pairing on the dual vector space $\mathbb{V}_1 \times \mathbb{V}_2$ as follows:

$$e : \mathbb{V}_1 \times \mathbb{V}_2 \to \mathbb{G}_T, (\mathbf{v}, \mathbf{w}) \mapsto \prod_{j \in [d]} e(g_1^{v^{(j)}}, g_2^{w^{(j)}})$$

for $\mathbf{v} = (g_1^{v^{(1)}}, \dots, g_1^{v^{(d)}}) \in \mathbb{V}_1$ and $\mathbf{w} = (g_2^{w^{(1)}}, \dots, g_2^{w^{(d)}}) \in \mathbb{V}_2$. The newly defined map e is also non-degenerate bilinear, i.e., e satisfies the following two properties:

- Bilinearity: $e(\mu \circ \mathbf{v}, \nu \circ \mathbf{w}) = e(\mathbf{v}, \mathbf{w})^{\mu\nu}$, for $\mu, \nu \in \mathbb{F}_q, \mathbf{v} \in \mathbb{V}_1$ and $\mathbf{w} \in \mathbb{V}_2$.
- Non-degeneracy: If $e(\mathbf{v}, \mathbf{w}) = 1_{\mathbb{G}_t}$ for all $\mathbf{w} \in \mathbb{V}_2$, then $\mathbf{v} = \mathbf{1}^m_{\mathbb{G}_1}$.

Let $\mathbb{B} = \{\mathbf{b}_1, \dots, \mathbf{b}_d\}$ and $\mathbb{B}^* = \{\mathbf{b}^*_1, \dots, \mathbf{b}^*_d\}$ be a basis of $\mathbb{V}_1$ and $\mathbb{V}_2$, respectively. Then, $\mathbb{B}$ and $\mathbb{B}^*$ are dual orthogonal if for all $i, j \in [d]$ it holds that $e(\mathbf{b}_i, \mathbf{b}^*_j) = 1_{\mathbb{G}_t}$ if $i \neq j$.

7.3 Description

We are now ready to describe the four algorithms defining our scheme DiffPIPE.

DiffPIPE.$\mathsf{Setup}(\lambda, \mathcal{F}_n)$**:** This algorithm takes the security parameter λ, a description of $\mathcal{F}_n = \{m, n, \Delta, B\}$ with the length $m \in \mathbb{N}$ of the vectors, the arity $n \in \mathbb{N}$ of the multi-input functionality, $\Delta \subsetneq \mathbb{N}$ of polynomial size and a bound $B \in \mathbb{N}$ on each component inner product. It works as follows:

(1) Setup a bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ with $\mathsf{params}_{\mathbb{G}} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$, where the corresponding group sizes are chosen to be a prime number $q \gg nB + \max_{\delta \in \Delta}(\delta)$.

(2) Extend e to a pairing on a dual vector space $\mathbb{V}_1 \times \mathbb{V}_2$ with $\mathbb{V}_1 = \mathbb{G}_1^{m+2k+2}$ and $\mathbb{V}_2 = \mathbb{G}_2^{m+2k+2}$ for an appropriately chosen parameter k, associated with the k-LIN assumption.

(3) Sample random $\zeta \xleftarrow{\$} \mathbb{F}_q \backslash \{0\}$, and compute $g_T = e(g_1, g_2)^{\zeta}$ with g_1 and g_2 being the generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively.

(4) For $i \in [n]$, generate a dual orthogonal basis $\mathbb{B}_i = \{\mathbf{b}_{i,1}, \dots, \mathbf{b}_{i,m+2k+2}\}$ and $\mathbb{B}^*_i = \{\mathbf{b}^*_{i,1}, \dots, \mathbf{b}^*_{i,m+2k+2}\}$ such that $e(\mathbf{b}_{i,j}, \mathbf{b}^*_{i,j}) = g_T$ for all $j \in [m + 2k + 2]$ and set

$$\hat{\mathbb{B}}_i = \{\mathbf{b}_{i,1}, \dots, \mathbf{b}_{i,m}, \mathbf{b}_{i,m+1}, \mathbf{b}_{i,m+k+2}, \dots, \mathbf{b}_{i,m+2k+1}\},$$
$$\hat{\mathbb{B}}^*_i = \{\mathbf{b}^*_{i,1}, \dots, \mathbf{b}^*_{i,m}, \mathbf{b}^*_{i,m+1}, \mathbf{b}^*_{i,m+3}, \dots, \mathbf{b}^*_{i,m+k+1}\}.$$

(5) Publish public parameters $\mathsf{pp} = (\mathsf{params}_{\mathbb{G}}, g_T)$ and set the master secret key $\mathsf{msk} = \{\hat{\mathbb{B}}^*_i\}_{i \in [n]}$. Let $\mathsf{sk}_i = \hat{\mathbb{B}}_i$ be the secret encryption key for slot i. All of the remaining algorithms implicitly take pp.

DiffPIPE.$\mathsf{Enc}(\mathsf{sk}_i, i, \vec{x}_i)$**:** Takes as input the secret key sk_i, an index $i \in [n]$, a vector $\vec{x}_i = (x_i^{(1)}, \dots, x_i^{(m)}) \in \mathbb{F}_q^m$ and performs the following steps:

(1) Select random $\varphi_{i,1}, \dots, \varphi_{i,k} \xleftarrow{\$} \mathbb{F}_q$, and compute

$$\mathbf{c}_i = \sum_{j \in [m]} x_i^{(j)} \mathbf{b}_{i,j} + \mathbf{b}_{i,2+1} + \sum_{j \in [k]} \varphi_{i,j} \mathbf{b}_{i,m+1+j}$$
$$= (\vec{x}_i, 1, 0, \vec{0}^{k-1}, \varphi_{i,1}, \dots, \varphi_{i,k}, 0)_{\mathbb{B}_i}$$

(2) Output ciphertext $\mathsf{ct}_i = (i, \mathbf{c}_i)$.

DiffPIPE.$\mathsf{KeyGen}(\mathsf{msk}, f, \mathbb{D})$**:** On input the msk and a function described by a set of vectors $\{\vec{y}_i\}_{i \in [n]}$ and a distribution $\mathbb{D}$ over Δ, execute the following steps:

(1) Sample $\nu \xleftarrow{\mathbb{D}} \Delta$ and random $r_i, \gamma_{i,1}, \dots, \gamma_{i,k-1} \xleftarrow{\$} \mathbb{F}_q$, for $i \in [n]$, with the restriction that $\sum_{i \in [n]} r_i = \nu$.

(2) For each $i \in [n]$ compute

$$\mathbf{k}_i = \sum_{j \in [m]} y_i^{(j)} \mathbf{b}^*_{i,j} + r_i \mathbf{b}^*_{i,m+1} + \sum_{j \in [k-1]} \gamma_{i,j} \mathbf{b}^*_{i,m+1+j}$$
$$= (\vec{y}_i, r_i, 0, \gamma_{i,1}, \dots, \gamma_{i,k-1}, \vec{0}^k, 0)_{\mathbb{B}^*_i},$$

where $\vec{y}_i = (y_1^{(1)}, \dots, y_1^{(m)})$.

(3) Output decryption key $\mathsf{dk} = \{\mathbf{k}_i\}_{i \in [n]}$

DiffPIPE.$\mathsf{Dec}(\mathsf{dk}, \mathsf{ct}_1, \dots, \mathsf{ct}_n)$**:** This algorithm takes a decryption key $\mathsf{dk} = \{\mathbf{k}_i\}_{i \in [n]}$, n ciphertexts $\mathsf{ct}_1, \dots \mathsf{ct}_n$ and proceeds as follows:

(1) Compute $L_T = \prod_{i \in [n]} e(\mathbf{c}_i, \mathbf{k}_i)$

(2) Attempt to determine a value $\Lambda \in \mathbb{Z}$ such that $g_T^{\Lambda} = L_T$ by performing an exhaustive search over a specific polynomial-size range of possible values. If it succeeds, output Λ, else output $\perp$.[4]

7.4 Analysis

In the following, we show the correctness and security of the proposed scheme DiffPIPE.

7.4.1 Correctness. We need to show that for any set of ciphertexts X and any decryption key the probability that the evaluation of the decryption key, associated with a function f, on the set of ciphertexts outputs $f(X) + \nu$ with the same probability that ν was sampled over Δ. For any set of ciphertexts $\{\mathsf{ct}_i = (i, \mathbf{c}_i)\}_{i \in [n]}$ with

$$\mathbf{c}_i = (\vec{x}_i, 1, 0, \vec{0}^{k-1}, \varphi_{i,1}, \dots, \varphi_{i,k}, 0)_{\mathbb{B}_i}$$

where ct_i encrypts some vector $\vec{x}_i \in \mathbb{F}_q^m$ with respect to index $i \in [n]$ and any decryption key $\mathsf{dk}_f = \{\mathbf{k}_i\}_{i \in [n]}$ corresponding to a function f described by $\{\vec{y}_i\}_{i \in [n]}$ such that $\vec{y}_i \in \mathbb{F}_q^m$ for all $i \in [n]$ and $\nu \in \Delta$ with $\mathbf{k}_i = (\vec{y}_i, r_i, 0, \gamma_{i,1}, \dots, \gamma_{i,k-1}, \vec{0}^k, 0)_{\mathbb{B}^*_i}$ and therefore

$$L_T = \prod_{i \in [n]} e(\mathbf{c}_i, \mathbf{k}_i) = \prod_{i \in [n]} g_T^{\mathbf{c}_i \cdot \mathbf{k}_i}$$
$$= \prod_{i \in [n]} g_T^{\vec{x}_i \cdot \vec{y}_i + r_i} = g_T^{\sum_{i \in [n]} (\vec{x}_i \cdot \vec{y}_i + r_i)}$$
$$= g_T^{\nu + \sum_{i \in [n]} \vec{x}_i \cdot \vec{y}_i} = g_T^{\nu + f(\vec{x}_1, \dots \vec{x}_n)}$$

This follows from the fact that for each $i \in [n]$, $\mathbb{B}_i$ and $\mathbb{B}^*_i$ are dual orthogonal bases and $\sum_{i \in [n]} r_i = \nu$. Since Δ is of polynomial size and therefore, also the range $[nB + \max_{\delta \in \Delta}]$ is of polynomial size. Thus, if $\nu + \sum_{i \in [n]} \vec{x}_i \cdot \vec{y}_i$ is in the polynomial size range of possible values that the decryption algorithm searches, the algorithm would output $\Lambda = \nu + \sum_{i \in [n]} \vec{x}_i \cdot \vec{y}_i$ with the same probability that ν is sampled over Δ.

7.4.2 Security. In the following, we sketch a security proof of DiffPIPE, where missing details can be found in Appendix D. Our

[4]Similar exhaustive search steps are part of all bilinear map-based inner product constructions. The polynomial running time is guaranteed by restricting the output to lie within a fixed polynomial size range [15].

$$G_0 = \underbrace{G_{1,0,3} \xrightarrow{k-LIN} G_{1,1,1} \xrightarrow{=} G_{1,1,2} \xrightarrow{k-LIN} G_{1,1,3} \xrightarrow{k-LIN} G_{1,2,1} \xrightarrow{=} \cdots \xrightarrow{k-LIN} G_{1,q_{key},3}}_{\text{Sequence 1}}$$

$$= \underbrace{G_{2,0,3} \xrightarrow{k-LIN} G_{2,1,1} \xrightarrow{=} G_{2,1,2} \xrightarrow{k-LIN} G_{2,1,3} \xrightarrow{k-LIN} G_{2,2,1} \xrightarrow{=} \cdots \xrightarrow{k-LIN} G_{2,n,3}}_{\text{Sequence 2}} \xrightarrow{=} G_3 \xrightarrow[\vec{x}_{i,0},\vec{x}_{i,1}\text{ interchanged}]{Seq.1\text{ with}} G_4$$

Figure 7: Sequence of games in the proof of Theorem 7.2. G_l **denotes Game**$_l$**.**

schemes relies on the hardness of the general k-Linear assumption [41], being still the basis of various state-of-the-art papers, e.g., [46]. It works for any choice of k, including the Symmetric External Diffie-Hellman Assumption (SXDH) for $k = 1$ and the Decisional Linear Assumption (DLIN) for $k = 2$. By increasing k, the hardness of the problem can be increased as well.

Definition 7.1 (k-Linear Assumption: k-LIN). Let $k \geq 1$. Fix a number $\chi \in [2]$. The k-LIN problem is to guess a bit $\beta \overset{\$}{\leftarrow} \{0,1\}$ given $\varepsilon_\beta = (\mathtt{params}_{\mathbb{G}}, g_\chi^{\xi_1}, \ldots, g_\chi^{\xi_k}, g_\chi^{\delta_1\xi_1}, \ldots, g_\chi^{\delta_k\xi_k}, \rho_\beta)$; where $\mathtt{params}_{\mathbb{G}} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e) \overset{\$}{\leftarrow} \mathcal{G}_{BPG}(\lambda); \xi_1, \ldots, \xi_k, \sigma \overset{\$}{\leftarrow} \mathbb{F}_q \setminus \{0\}$; $\delta_1, \ldots, \delta_k \overset{\$}{\leftarrow} \mathbb{F}_q$; and $\rho_0 = g_\chi^{\sum j\in[k]\delta_j}$ and $\rho_1 = g_\chi^{\sigma+\sum_{j\in[k]}\delta_j}$. The k-LIN assumption states that for any PPT algorithm $\mathcal{A}$, for any security parameter λ the advantage of $\mathcal{A}$ in deciding the k-LIN problem is

$$\mathsf{Adv}_{\mathcal{A}}^{k-LIN}(\lambda) = |\Pr[1 \leftarrow \mathcal{A}(\varepsilon_0)] - \Pr[1 \leftarrow \mathcal{A}(\varepsilon_1)]| \leq negl(\lambda)$$

for some negligible function *negl*.

The security of our scheme is shown in the following theorem:

Theorem 7.2. *Assume that the the k-LIN problem is hard. Then the above described NMIFE scheme achieves SMN-H security under the restriction, that the adversary makes at least one ciphertext query for each encryption slot.*[5] *For any PPT adversary $\mathcal{A}$ against the SMN-H security of the NMIFE scheme described above, there exists a PPT algorithm $\mathcal{B}$ against the k-LIN problem such that for any security parameter λ, we have*

$$Adv_{\mathcal{A}}(\lambda) \leq (4q_{key} + 2n)Adv_{\mathcal{B}}^{k-LIN} \tag{7}$$

The proof is structured as a series of games which differ in the construction of the decryption keys and/or the ciphertexts queried by the adversary $\mathcal{A}$. The detailed description of the games is postponed to the end of this section. In the first game, Game$_0$, the queried ciphertexts and the queried decryption keys are constructed as those in the security experiment $\mathsf{Expt}_{\mathcal{A}}^{NMIFE}(0)$. We change the answers to the queries in multiple steps to those in the security experiment $\mathsf{Expt}_{\mathcal{A}}^{NMIFE}(1)$, being Game$_4$. The considered sequence of games is displayed in Figure 7.

For some pairs of subsequent games, the probability to distinguish between these is upper bounded by the advantage of an adversary $\mathcal{B}$ that aims to solve the k-LIN problem:

[5]This restriction can be rescinded if a generic transformation analogue to the one described in Datta et al. [14] is applied. Since we aim for a DP scenario, we assume a prefixed data set.

Lemma 7.3. *For any PPT adversary $\mathcal{A}$ between Game$_{j,l-1,3}$ and Game$_{j,l,1}$ and any $\mathcal{A}$ between Game$_{j,i,2}$ and Game$_{j,i,3}$, $j \in [2]$ there exists a PPT algorithm $\mathcal{B}$ for the k-LIN assumption (Definition 7.1) such that for any security parameter λ, we have*

$$\begin{array}{l} \left|Adv_{\mathcal{A}}^{Game_{j,i-1,3}}(\lambda) - Adv_{\mathcal{A}}^{Game_{j,i,1}}(\lambda)\right| \\ \left|Adv_{\mathcal{A}}^{Game_{1,i,2}}(\lambda) - Adv_{\mathcal{A}}^{Game_{1,i,3}}(\lambda)\right| \end{array} \leq Adv_{\mathcal{B}}^{k-LIN}(\lambda)$$

for all $i \in [q_{key}]$, if $j = 1$, $i \in [n]$, if $j = 2$.

In Figure 7, this is displayed by $\xrightarrow{k-LIN}$. For the remaining pairs of subsequent games, one can show that these are indistinguishable (being represented by $\xrightarrow{=}$):

Lemma 7.4. *For any PPT adversary $\mathcal{A}$, for any security parameter λ, we have*

$$Adv_{\mathcal{A}}^{Game_{1,l,1}}(\lambda) = Adv_{\mathcal{A}}^{Game_{1,l,2}}(\lambda), \text{ for all } l \in [q_{key}].$$
$$Adv_{\mathcal{A}}^{Game_{2,i,1}}(\lambda) = Adv_{\mathcal{A}}^{Game_{2,i,2}}(\lambda), \text{ for all } i \in [n].$$
$$Adv_{\mathcal{A}}^{Game_{2,n,3}}(\lambda) = Adv_{\mathcal{A}}^{Game_3}(\lambda).$$

The proofs including a more detailed analysis are presented in Appendix D. By counting the number of transitions in Figure 7 which have the same form as those in Lemma 7.3, the bound given in Equation (7) follows.

▷ **Sequence of Games:**
It remains to define the games used in Figure 7, what happens next. The framed parts indicate the terms that were modified in the transformation from the previous game.

Game$_0$: This experiment corresponds to the experiment $\mathsf{Expt}_{\mathcal{A}}^{NMIFE}(0)$ described in Definition 5.3, therefore the security experiment where the random bit is $\beta = 0$. More precisely, for all $i \in [n]$, in response to the ciphertext query of $\mathcal{A}$ with respect to index i corresponding to a pair of vectors $(\vec{x}_{i,0}, \vec{x}_{i,1} \in \mathbb{F}_q^m \times \mathbb{F}_q^m)\mathcal{B}$ returns $\mathtt{ct}_i = (i, \mathbf{c}_i)$ where

$$\mathbf{c}_i = (\vec{x}_{i,0}, 1, 0, \vec{0}^{k-1}, \varphi_{i,1}, \ldots, \varphi_{i,k}, 0)_{\mathbb{B}_i} \tag{8}$$

and for all $l \in [q_{key}]$, to answer the l^{th} decryption key query of $\mathcal{A}$ corresponding to a set of vectors and two noise values $(\{\vec{y}_{l,i}\}_{i\in[n]}, \nu_{l,0}, \nu_{l,1})$ such that $\vec{y}_{l,i} \in \mathbb{F}_q^m$ and $\nu_{l,0}, \nu_{l,1} \in \Delta$, $\mathcal{B}$ generates $\mathsf{dk}_l = \{\mathbf{k}_{l,i}\}_{i\in[n]}$, where

$$\mathbf{k}_{l,i} = (\vec{y_{l,i}}, r_{l,i,0}, 0, \gamma_{l,i,1}, \ldots, \gamma_{l,i,k-1}, \vec{0}^k, 0)_{\mathbb{B}_i^*} \tag{9}$$

and $\sum_{i\in[n]} r_{l,i} = \nu_{l,0}$ for $i \in [n]$. All other variables and parameters are generated as in Section 7.3.

Sequence 1
For each $l \in [q_{key}]$, execute the following three steps iteratively.
Game$_{1,l,1}$ ($l \in [q_{key}]$): Game$_{1,0,3}$ coincides with Game$_0$. This experiment is analogous to Game$_{1,l-1,3}$ with the only exception that in response to the l^{th} decryption key query of $\mathcal{A}$ corresponding to $(\{\vec{y}_{l,i}\}_{i\in[n]}, v_{l,0}, v_{l,1}) \in \mathbb{F}_q^m \times \Delta \times \Delta$ for all $i \in [n]$ $\mathcal{B}$ gives back $\mathsf{dk}_l = \{\mathbf{k}_{i,l}\}_{i\in[n]}$, where

$$\mathbf{k}_{l,i} = (\vec{y_{l,i}}, r_{l,i,0}, 0, \gamma_{l,i,1}, \ldots, \gamma_{l,i,k-1}, \vec{0}^k, \boxed{\rho_{l,i}})_{\mathbb{B}_i^*} \tag{10}$$

with $\rho_{l,i} \xleftarrow{\$} \mathbb{F}_q \setminus \{0\}$ for all $i \in [n]$, such that $\sum_{i\in[n]} \rho_{l,i} = 0$, and all other variables are generated as in Game$_{1,l-1,3}$.

Game$_{1,l,2}$ ($l \in [q_{key}]$): This experiment is identical to Game$_{1,l,1}$ except that in response to the l^{th} decryption key query of $\mathcal{A}$ corresponding to $(\{\vec{y}_{l,i}\}_{i\in[n]}, v_{l,0}, v_{l,1}) \in \mathbb{F}_q^m \times \Delta \times \Delta$ for all $i \in [n]$ $\mathcal{B}$ returns $\mathsf{dk}_l = \{\mathbf{k}_{i,l}\}_{i\in[n]}$, where

$$\mathbf{k}_{l,i} = (\vec{y_{l,i}}, r_{l,i,0}, \boxed{r_{l,i,1}}, \gamma_{l,i,1}, \ldots, \gamma_{l,i,k-1}, \vec{0}^k, \rho_{l,i})_{\mathbb{B}_i^*} \tag{11}$$

with $r_{l,i,1} \xleftarrow{\$} \mathbb{F}_q$ for all $i \in [n]$, such that $\sum_{i\in[n]} r_{l,i,1} = v_{l,1}$, and all other variables are generated as in Game$_{1,l,1}$.

Game$_{1,l,3}$ ($l \in [q_{key}]$): This experiment is analogous to Game$_{1,l,2}$ except that in response to the l^{th} decryption key query of $\mathcal{A}$ corresponding to $(\{\vec{y}_{l,i}\}_{i\in[n]}, v_{l,0}, v_{l,1}) \in \mathbb{F}_q^m \times \Delta \times \Delta$ for all $i \in [n]$ $\mathcal{B}$ returns $\mathsf{dk}_l = \{\mathbf{k}_{i,l}\}_{i\in[n]}$, where

$$\mathbf{k}_{l,i} = (\vec{y_{l,i}}, r_{l,i,0}, r_{l,i,1}, \gamma_{l,i,1}, \ldots, \gamma_{l,i,k-1}, \vec{0}^k, \boxed{0})_{\mathbb{B}_i^*}. \tag{12}$$

All variables are generated as in Game$_{1,l,2}$.

Sequence 2:
For each $i \in [n]$, execute the following three steps iteratively.
Game$_{2,i,1}$ ($i \in [n]$): Game$_{2,0,3}$ coincides with Game$_{1,q_{key},3}$. This experiment is analogous to Game$_{2,i-1,3}$ with the only exception that in response to the ciphertext query for slot i of $\mathcal{A}$ corresponding to $(\vec{x}_{i,0}, \vec{x}_{i,1}) \in \mathbb{F}_q^m \times \mathbb{F}_q^m$, $\mathcal{B}$ gives back $\mathtt{ct}_i = (i, \mathbf{c}_i)$, where

$$\mathbf{c}_i = (\vec{x}_{i,0}, 1, 0, \vec{0}^{k-1}, \varphi_{i,1}, \ldots, \varphi_{i,k}, \boxed{\omega_i})_{\mathbb{B}_i} \tag{13}$$

with $\omega_i \xleftarrow{\$} \mathbb{F}_q \setminus \{0\}$ and all other variables formed as in Game$_{2,i-1,3}$.

Game$_{2,i,2}$ ($i \in [n]$): This experiment is analogous to Game$_{2,i,1}$ except that in response to the ciphertext query for slot i of $\mathcal{A}$ corresponding to $(\vec{x}_{i,0}, \vec{x}_{i,1}) \in \mathbb{F}_q^m \times \mathbb{F}_q^m$, $\mathcal{B}$ returns $\mathtt{ct}_i = (i, \mathbf{c}_i)$, where

$$\mathbf{c}_i = (\boxed{\vec{x}_{i,1}, 0, 1}, \vec{0}^{k-1}, \varphi_{i,1}, \ldots, \varphi_{i,k}, \omega_i)_{\mathbb{B}_i}. \tag{14}$$

Here, all the variables are formed as in Game$_{2,i,1}$.

Game$_{2,i,3}$ ($i \in [n]$): This experiment is analogous to Game$_{2,i,2}$ except that in response to the ciphertext query for slot i of $\mathcal{A}$ corresponding to $(\vec{x}_{i,0}, \vec{x}_{i,1}) \in \mathbb{F}_q^m \times \mathbb{F}_q^m$, $\mathcal{B}$ returns $\mathtt{ct}_i = (i, \mathbf{c}_i)$, where

$$\mathbf{c}_i = (\vec{x}_{i,1}, 0, 1, \vec{0}^{k-1}, \varphi_{i,1}, \ldots, \varphi_{i,k}, \boxed{0})_{\mathbb{B}_i}. \tag{15}$$

All the variables are formed as in Game$_{2,i,2}$.

Game$_3$: In this experiment for all ciphertext queries of $\mathcal{A}$ corresponding to $(\vec{x}_{i,0}, \vec{x}_{i,1}) \in \mathbb{F}_q^m \times \mathbb{F}_q^m$, for $i \in [n]$ and all decryption key queries of $\mathcal{A}$ corresponding to $(\{\vec{y}_{l,i}\}_{i\in[n]}, v_{l,0}, v_{l,1}) \in \mathbb{F}_q^m \times \Delta \times \Delta$ for $l \in [q_{key}]$, $\mathcal{B}$ returns $\mathtt{ct}_i = (i, \mathbf{c}_i)$, where

$$\mathbf{c}_i = (\vec{x}_{i,1}, \boxed{1, 0}, \vec{0}^{k-1}, \varphi_{i,1}, \ldots, \varphi_{i,k}, 0)_{\mathbb{B}_i} \tag{16}$$

and $\mathsf{dk}_l = \{\mathbf{k}_{i,l}\}_{i\in[n]}$, where

$$\mathbf{k}_{l,i} = (\vec{y_{l,i}}, \boxed{r_{l,i,1}, r_{l,i,0}}, \gamma_{l,i,1}, \ldots, \gamma_{l,i,k-1}, \vec{0}^k, 0)_{\mathbb{B}_i^*}. \tag{17}$$

Here all the variables are formed as in Game$_{2,n,3}$.

Game$_4$: This experiment corresponds to the experiment $\mathsf{Expt}_{\mathcal{A}}^{NMIFE}(1)$ described in Definition 5.3, therefore the security experiment where the random bit is $\beta = 1$.

7.5 Implementation

We implemented our above described SMN-H NMIFE scheme DiffPIPE and used it in a series of experiments to privately evaluate counting queries on different data sets in the medical context. That is, we assume a set of n users. Each user has a record of m elements $\vec{x}_i = (x_i^1, \ldots, x_i^m)$, representing m different attributes. Moreover, there is an authority that runs DiffPIPE.Setup for generating a master secret key msk and the single secret keys $\{\mathsf{sk}_i\}_{i\in[n]}$ for the n users. After the authority distributes the secret keys to the clients over a secure channel, the clients encrypt their record using DiffPIPE.Enc. The analyst itself is interested in an evaluation over the data set $X = (\vec{x}_1, \ldots, \vec{x}_n)$, sending a function f to the authority. The authority itself first checks, if the analyst has proper entitlement, regarding differential privacy guidelines, e.g. enough privacy budget (cf. Section 2.2), and if $f \in \mathcal{F}_n$. In the positive case, it generates a decryption key dk with DiffPIPE. KeyGen(msk, f, $\mathbb{D}$) where $\mathbb{D}$ is an appropriate choice of a distribution achieving DP [18]. With this dk the analyst can evaluate the function on the encrypted data set, getting a noisy result that protects the data itself. As the described use case assumes only one data set X that is encrypted, the relaxed security notion of single-message-and-noise-hiding is sufficient.

In our experiments, we measured on the one hand the time required for basic operations but also for running a complete analysis on existing databases. We compared DiffPIPE with *noisy-DOT* (Appendix C) build according to Section 6 from a modified form of the *DOT* scheme of [15] implemented in the library [28][6]. As expected DiffPIPE is more efficient than *noisy-DOT*. For our experiments we used $k = 2$ and therefore based the security on the Decisional Linear assumption.

As the process of checking if the DP requirements are fulfilled and choosing the distribution to compute the noise depending on the requested analysis and the privacy-budget are a consequence of applying DP and not of FE, we skipped this part. More precisely, we focused on the implementation of the general setup, assuming only one counting query is made by the analyst where we know we can achieve DP with sampling noise from a Laplace distribution $Lap(1/\epsilon)$ choosing $\epsilon = 1$. To sample the noise for the key generation algorithm we used the differential privacy library of google [43].

[6]The scheme in [28] is slightly modified from original proposed scheme in [15] to achieve better performance, for more information see [28], and therefore also our implementation of the *DOT* scheme is slightly modified for better performance.

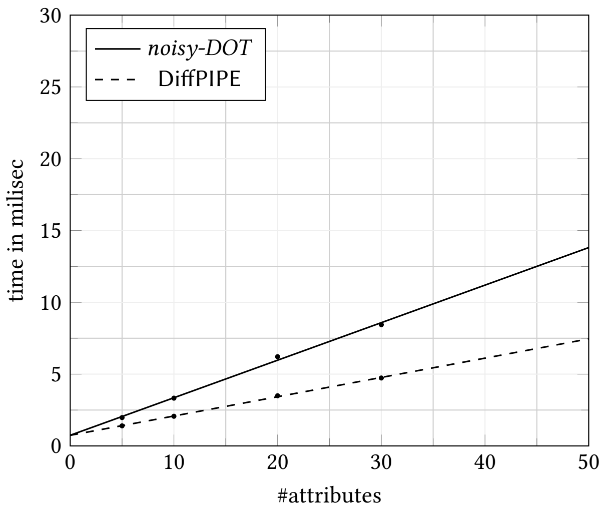


Figure 8: Time for encrypting one record in *noisy-DOT* and DiffPIPE (dashed lines) regarding the number of attributes.

Therefore our overhead for adding noise is the same as sampling the noise from the library with above parameters.

Since our scheme is basically a n-fold extension of a single input FE scheme with small modifications to achieve security, the size of the decryption keys and also the evaluation time grow linear with the data set size (# records × # attributes). To support this conjecture and also compare the basic operations to the *noisy-DOT* scheme, we did some benchmarking. As the record values have no impact, they have been randomly generated. The results of our experiments are visualized in Figure 8 to Figure 10 and confirm the expected behavior, that DiffPIPE has also linear growth but with a smaller growing rate then *noisy-DOT*. Figure 8 shows the time needed to encrypt one single record in dependency of the number of attributes. Figure 9 and Figure 10 show the time needed to generate the decryption key and the time needed to evaluate the function in dependency of the number of records for different attribute size. Note that since the setup algorithm of DiffPIPE is essentially the same as in *noisy-DOT*, the time needed is practically identical and therefore we omit a figure here.

To demonstrate the usefulness of the relaxed security notion of SMN-H and therefore our scheme DiffPIPE, we compared the evaluation process of our scheme compared to *noisy-DOT* on the following data sets:

- Low Birth Weight study data set [24]: “How many children with low birth weight are born in this data set?”
- Prostate Cancer Study data set [26]: “What is the percentage of tumor penetration of prostatic capsule?”
- Umaru Impact Study data set [27]: “How many percent are drugfree for at least 12 months?”
- Nhanes III data set [25]:“How many percent of the participants have high systolic blood pressure?”

The data sets vary in number of records and attributes (Table 1). We modified the data sets so that there were only complete records regarding the important attributes. Since most of the cryptographic

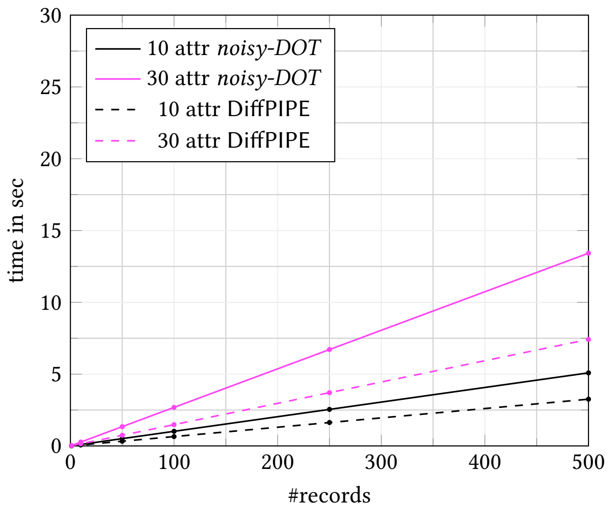


Figure 9: Time for generating the decryption key in *noisy-DOT* and DiffPIPE (dashed lines).

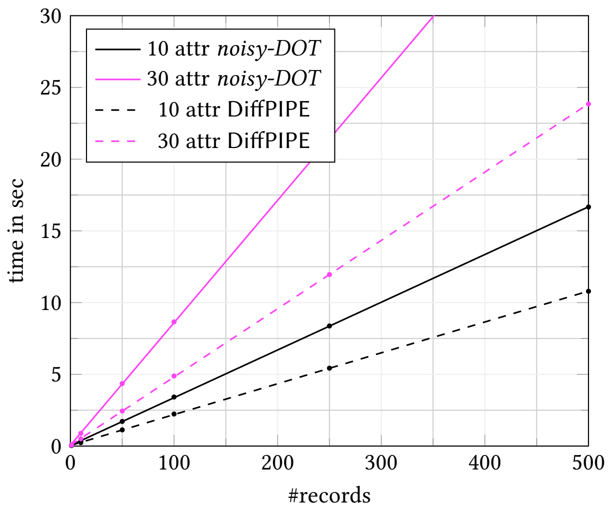


Figure 10: Time for evaluating the function in *noisy-DOT* and DiffPIPE (dashed lines).

Table 1: Number of records and attributes in cleaned data sets.

Data set	# records	# attributes
Low Birth Weight Study	189	10
Prostate Cancer Study	380	8
Umaru Impact Study	575	8
Nhanes III	16,427	12

techniques operate over finite fields, such as $\mathbb{F}_q$ but most numeric real world data is in $\mathbb{R}$ a well known problem is how to convert the real world data into input data for encryption schemes. A standard solution is to use a fix-point arithmetic, meaning to interpret a value $z \in \mathbb{R}$ as $x = \lfloor z \cdot s \rceil$ with a scaling factor s. For example the value $z = 8791.5467891$ with a scaling factor 100 would result in $x = 879155$. After the computation on the encrypted data is finished, the result needs to be scaled back, e.g. dividing by the scaling factor s. If s is chosen appropriate to the input data and to $\mathbb{F}_q$ the usefulness of the data is not compromised while preserving the ability to operate on the encrypted data. Most of the data sets from Table 1 were already represented as values in $\mathbb{N}$ or had only two decimal places. But this was not the case for the noise sampled from the Laplace distribution. Therefore for our experiments we set our scaling factor to 100,000 without compromising the result of our computation.

Another challenge was to handle the noise sampling process. Normally noise sampled from the Laplace distribution is distributed over $\mathbb{R}$. The DP libraries handles the problem of sampling over a discrete set, but we additionally needed to restrict the range, so we could assure that the sampled noise v is still in the range that can be decrypted, e.g. restricting Δ. We exploited that the Laplace distribution samples values close to zero. To make sure that the sampled noise values lie in Δ with probability p we need to choose $\Delta = [\lfloor log_e(1-p)\rfloor, \lceil -log_e(1-p)\rceil]$. For a probability of $p = 0.9$, sensitivity of the function f equals 1 and $\epsilon = 1$, this would mean $\Delta \approx [-2.303, 2.303]$, for probability $p = 0.95, \Delta \approx [-2.996, 2.996]$. Additionally we rounded the noise v not mathematically but always to $v' = sign(v) \cdot \lceil |v| \rceil$ with $sign(x) = 1, x \geq 0$ and $sign(x) = -1, x < 0$ to assure enough noise is added in the step of representing the noise as element of $\mathbb{F}_q$.

We performed all benchmarking experiments on a system running Ubuntu 22.04.2 LTS, 256GB RAM and 18 vCPUs (AMD Epyc 7272). For each data set, the following attributes measured for both DiffPIPE and *noisy-DOT*:

- the time to set up the scheme, e.g. generating pp, msk and the secret keys $\{\mathsf{sk}_i\}_{i\in[n]}$ which is in the nano seconds area and represents the one time effort of the authority,
- the time to encrypt one record, e.g. the effort of one participating user,
- the time to generate a noisy decryption key dk, e.g. the effort of the authority,
- and the time to evaluate the dk on the encrypted data set, e.g. the effort of the analyst.

Results can be seen in Table 2. As expected and consistent to our results of the basic operations, the experiments on the real data sets show an advantage of our scheme DiffPIPE: less time is needed to encrypt, generating decryption keys and evaluating a function while the setup algorithm only differs in a few nanoseconds.

8 CONCLUSION

Motivated by the need for cryptographic schemes that simultaneously realize input and output privacy, we introduced the concept of noisy multi input functional encryption (NMIFE) that allows to integrate differential privacy into the decryption procedure. Moreover, we explained its potential benefits compared to approaches that aim for combining DP with multi party computation or homomorphic encryption, respectively. We provided an adapted security definition, namely single-message-and-noise-hiding (SMN-H), and explained how certain function-hiding MIFE schemes can be transformed into secure SMN-H NMIFE schemes (actually even meeting a stronger security definition). This was demonstrated on a concrete NMIFE scheme, dubbed *noisy-DOT*. We additionally constructed a new secure SMN-H NMIFE scheme DiffPIPE and demonstrated its applicability for realizing privacy preserving counting queries in a number of experiments, comparing its efficiency to *noisy-DOT*.

While NMIFE may have already their use in practical use cases, we see various opportunities for further research. For instance, the restriction of *noisy-DOT* to linear functions is a consequence of the lack of function-hiding FE schemes for more complex functions and not of the definition of NMIFE or the transformation. While linear functions practically limit the use cases to simple statistical analysis, e.g. counting queries, quadratic functions allow for more complex analysis. In other words, progress made in the area of FE can most likely be used for building more powerful NMIFE schemes as well. If a full-hiding MIFE scheme for polynomials of degree 2 is constructed, with the help of our transformation it would directly yield an according NMIFE scheme. In fact, as the security notion of SMN-H is probably weaker than full-hiding, designing NMIFE schemes for more complex functions fulfilling SMN-H might be easier than for full-hiding FE.

Another potential line of research is the combination of FE and machine learning. If it comes to the training of a model, one main problem is still the restricted functionality of existing FE schemes. The schemes today are not able to train a whole model, but with iterative updates, the encrypted data can still be used. As we have seen in the context of federated learning ([31, 32, 47]), the intermediate results of the iterative algorithms often leak too much information about the underlying data. Here, one could use DP, and therefore a NMIFE scheme, to secure these intermediate results. To train complex models, linear NMIFE schemes are not sufficient. It has been shown that we can train complex machine learning algorithms with a quadratic FE scheme [11, 39], but these leak information about the input data through the intermediate results. With a quadratic NMIFE scheme we would be able to protect these data using DP.

Table 2: Run time of DiffPIPE vs. *noisy-DOT* for single operations for different data bases. Encryption time refers to encrypting x_i of a single user. First value represents the time for DiffPIPE, second value for *noisy-DOT*.

Data set	Setup [ns]	Enc [ms]	KeyGen [s]	Eval [s]
LBW	768.10	2.12	1.23	4.10
	759.60	3.23	1.93	6.33
PC	751.40	1.80	2.17	7.15
	662.95	2.70	3.25	10.63
UI	711.55	1.85	3.28	10.89
	780.55	2.69	4.92	16.14
Nhanes III	949.05	2.37	120.61	385.80
	965.50	3.82	194.88	620.52

ACKNOWLEDGMENTS

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

REFERENCES

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

[2] Mark Abspoel, Daniel Escudero, and Nikolaj Volgushev. 2020. Secure training of decision trees with continuous attributes. *Cryptology ePrint Archive* (2020).

[3] Abbas Acar, Hidayet Aksu, A Selcuk Uluagac, and Mauro Conti. 2018. A survey on homomorphic encryption schemes: Theory and implementation. *ACM Computing Surveys (Csur)* 51, 4 (2018), 1–35.

[4] Shweta Agrawal, Rishab Goyal, and Junichi Tomida. 2021. Multi-input quadratic functional encryption from pairings. In *Advances in Cryptology–CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16–20, 2021, Proceedings, Part IV 41*. Springer, 208–238.

[5] Yoshinori Aono, Takuya Hayashi, Le Trieu Phong, and Lihua Wang. 2016. Scalable and secure logistic regression via homomorphic encryption. In *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy*. 142–144.

[6] Alexandros Bakas and Antonis Michalas. 2022. Heal the Privacy: Functional Encryption and Privacy-Preserving Analytics. *arXiv preprint arXiv:2205.03083* (2022).

[7] Amos Beimel. 2011. Secret-sharing schemes: A survey. In *Coding and Cryptology: Third International Workshop, IWCC 2011, Qingdao, China, May 30-June 3, 2011. Proceedings 3*. Springer, 11–46.

[8] Jonas Böhler and Florian Kerschbaum. 2020. Secure multi-party computation of differentially private median. In *Proceedings of the 29th USENIX Conference on Security Symposium*. 2147–2164.

[9] Jonas Böhler and Florian Kerschbaum. 2021. Secure multi-party computation of differentially private heavy hitters. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 2361–2377.

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

[11] Sergiu Carpov, Caroline Fontaine, Damien Ligier, and Renaud Sirdey. 2020. Illuminating the Dark or how to recover what should not be seen in FE-based classifiers. *Proceedings on Privacy Enhancing Technologies* 2020, 2 (2020), 5–23.

[12] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. 2017. Homomorphic encryption for arithmetic of approximate numbers. In *Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I 23*. Springer, 409–437.

[13] Fida Kamal Dankar and Khaled El Emam. 2013. Practicing differential privacy in health care: A review. *Trans. Data Priv.* 6, 1 (2013), 35–67.

[14] Pratish Datta, Tatsuaki Okamoto, and Junichi Tomida. 2018. Full-Hiding (Unbounded) Multi-Input Inner Product Functional Encryption from the k-Linear Assumption. Cryptology ePrint Archive, Paper 2018/061. https://eprint.iacr.org/2018/061 https://eprint.iacr.org/2018/061.

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

[16] Edouard Dufour-Sans, Romain Gay, and David Pointcheval. 2018. Reading in the Dark: Classifying Encrypted Digits with Functional Encryption. Cryptology ePrint Archive, Paper 2018/206. https://eprint.iacr.org/2018/206 https://eprint.iacr.org/2018/206.

[17] Cynthia Dwork. 2006. Differential privacy. In *Automata, Languages and Programming: 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II 33*. Springer, 1–12.

[18] Cynthia Dwork, Aaron Roth, et al. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science* 9, 3–4 (2014), 211–407.

[19] Reo Eriguchi, Atsunori Ichikawa, Noboru Kunihiro, and Koji Nuida. 2022. Efficient Noise Generation Protocols for Differentially Private Multiparty Computation. *IEEE Transactions on Dependable and Secure Computing* (2022).

[20] Craig Gentry. 2009. *A fully homomorphic encryption scheme*. Stanford university.

[21] Oded Goldreich. 1998. Secure multi-party computation. *Manuscript. Preliminary version* 78, 110 (1998).

[22] Michela Iezzi. 2020. Practical Privacy-Preserving Data Science With Homomorphic Encryption: An Overview. In *2020 IEEE International Conference on Big Data (Big Data)*. 3979–3988. https://doi.org/10.1109/BigData50022.2020.9377989

[23] Damien Ligier, Sergiu Carpov, Caroline Fontaine, and Renaud Sirdey. 2017. Privacy preserving data classification using inner product encryption. In *Security and Privacy in Communication Networks: 12th International Conference, SecureComm 2016, Guangzhou, China, October 10-12, 2016, Proceedings 12*. Springer, 755–757.

[24] LogisticDx. 2023. Diagnostic Tests for Models with a Binomial Response. lbw: Low Birth Weight study data. https://rdrr.io/rforge/LogisticDx/man/lbw.html

[25] LogisticDx. 2023. Diagnostic Tests for Models with a Binomial Response. nhanes3: NHANES III data. https://rdrr.io/rforge/LogisticDx/man/nhanes3.html

[26] LogisticDx. 2023. Diagnostic Tests for Models with a Binomial Response. pcs: Prostate Cancer Study data. https://rdrr.io/rforge/LogisticDx/man/pcs.html

[27] LogisticDx. 2023. Diagnostic Tests for Models with a Binomial Response. uis: UMARU IMPATCT Study data. https://rdrr.io/rforge/LogisticDx/man/uis.html

[28] Tilen Marc, Miha Stopar, Jan Hartman, Manca Bizjak, and Jolanda Modic. 2019. Privacy-enhanced machine learning with functional encryption. In *Computer Security–ESORICS 2019: 24th European Symposium on Research in Computer Security, Luxembourg, September 23–27, 2019, Proceedings, Part I 24*. Springer, 3–21.

[29] Ilya Mironov, Omkant Pandey, Omer Reingold, and Salil Vadhan. 2009. Computational differential privacy. In *Advances in Cryptology-CRYPTO 2009: 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings*. Springer, 126–142.

[30] Payman Mohassel and Yupeng Zhang. 2017. Secureml: A system for scalable privacy-preserving machine learning. In *2017 IEEE symposium on security and privacy (SP)*. IEEE, 19–38.

[31] Viraaji Mothukuri, Reza M. Parizi, Seyedamin Pouriyeh, Yan Huang, Ali Dehghantanha, and Gautam Srivastava. 2021. A survey on security and privacy of federated learning. *Future Generation Computer Systems* 115 (2021), 619–640. https://doi.org/10.1016/j.future.2020.10.007

[32] Ahmed El Ouadrhiri and Ahmed Abdelhadi. 2022. Differential Privacy for Deep and Federated Learning: A Survey. *IEEE Access* 10 (2022), 22359–22380. https://doi.org/10.1109/ACCESS.2022.3151670

[33] Prajwal Panzade and Daniel Takabi. 2022. SoK: Privacy Preserving Machine Learning using Functional Encryption: Opportunities and Challenges. *arXiv preprint arXiv:2204.05136* (2022).

[34] Sikha Pentyala, Davis Railsback, Ricardo Maia, Rafael Dowsley, David Melanson, Anderson Nascimento, and Martine De Cock. 2022. Training differentially private models with secure multiparty computation. *arXiv preprint arXiv:2202.02625* (2022).

[35] Martin Pettai and Peeter Laud. 2015. Combining differential privacy and secure multiparty computation. In *Proceedings of the 31st Annual Computer Security Applications Conference*. 421–430.

[36] Bernardo Pulido-Gaytan, Andrei Tchernykh, Jorge M Cortés-Mendoza, Mikhail Babenko, Gleb Radchenko, Arutyun Avetisyan, and Alexander Yu Drozdov. 2021. Privacy-preserving neural networks with homomorphic encryption: C hallenges and opportunities. *Peer-to-Peer Networking and Applications* 14, 3 (2021), 1666–1691.

[37] Luis Bernardo Pulido-Gaytan, Andrei Tchernykh, Jorge M Cortés-Mendoza, Mikhail Babenko, and Gleb Radchenko. 2021. A survey on privacy-preserving machine learning with fully homomorphic encryption. In *High Performance Computing: 7th Latin American Conference, CARLA 2020, Cuenca, Ecuador, September 2–4, 2020, Revised Selected Papers 7*. Springer, 115–129.

[38] Jean Louis Raisaro, Gwangbae Choi, Sylvain Pradervand, Raphael Colsenet, Nathalie Jacquemont, Nicolas Rosat, Vincent Mooser, and Jean-Pierre Hubaux. 2018. Protecting Privacy and Security of Genomic Data in i2b2 with Homomorphic Encryption and Differential Privacy. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 15, 5 (2018), 1413–1426. https://doi.org/10.1109/TCBB.2018.2854782

[39] Théo Ryffel, Edouard Dufour-Sans, Romain Gay, Francis Bach, and David Pointcheval. 2019. Partially encrypted machine learning using functional encryption. *arXiv preprint arXiv:1905.10214* (2019).

[40] Rathindra Sarathy and Krishnamurty Muralidhar. 2011. Evaluating Laplace noise addition to satisfy differential privacy for numeric data. *Trans. Data Priv.* 4, 1 (2011), 1–17.

[41] Hovav Shacham. 2007. A cramer-shoup encryption scheme from the linear assumption and from progressively weaker linear variants. *Cryptology ePrint Archive* (2007).

[42] Adi Shamir. 1979. How to share a secret. *Commun. ACM* 22, 11 (1979), 612–613.

[43] Differential Privacy Team. 2022. Differential Privacy Library (DP Lib v2.0.0). https://github.com/google/differential-privacy

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

[45] Sameer Wagh, Divya Gupta, and Nishanth Chandran. 2018. SecureNN: Efficient and private neural network training. *Cryptology ePrint Archive* (2018).

[46] Brent Waters and David J. Wu. 2022. Batch Arguments for NP and More from Standard Bilinear Group Assumptions. Cryptology ePrint Archive, Paper 2022/336. https://eprint.iacr.org/2022/336 https://eprint.iacr.org/2022/336.

[47] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H. Yang, Farhad Farokhi, Shi Jin, Tony Q. S. Quek, and H. Vincent Poor. 2020. Federated Learning With Differential

Privacy: Algorithms and Performance Analysis. *IEEE Transactions on Information Forensics and Security* 15 (2020), 3454–3469. https://doi.org/10.1109/TIFS.2020.2988575

[48] Chuan Zhao, Shengnan Zhao, Minghao Zhao, Zhenxiang Chen, Chong-Zhi Gao, Hongwei Li, and Yu-an Tan. 2019. Secure multi-party computation: theory, practice and applications. *Information Sciences* 476 (2019), 357–372.

[49] Ezgi Zorarpacı and Selma Ayşe Özel. 2020. A hybrid approach of homomorphic encryption and differential privacy for privacy preserving classification. *International Journal of Applied Mathematics Electronics and Computers* (2020), 138 – 147. https://doi.org/10.18100/ijamec.801157

A DIFFERENTIAL PRIVACY

As shortly explained in Section 2.2 DP is a technique to achieve output privacy of a function. In the following we want to present the formal definitions.

Definition A.1 (Distance between Databases). The l_1 norm of a database $X = (\vec{x}_1, \ldots, \vec{x}_n)$ is denoted by $||X||_1$ and is defined to be

$$||X||_1 = \sum_{i=1}^{n} |\vec{x}_i|.$$

The l_1 distance between two databases X and Y is $||X - Y||_1$.

The l_1 distance measures how many records differ between database X and database Y. Typically in the standard definition of (ϵ, δ)-differential privacy ((ϵ, δ)-DP) we look at neighboring databases, meaning databases that differ in at most 1 record.

Definition A.2 (Differential Privacy). A randomized algorithm M with domain $\mathbb{N}^{|X|}$ is (ϵ, δ)-differential private if for all $S \subseteq \text{Range}(M)$ and for all $X, Y \in \mathbb{N}^{|X|}$ such that $||X - Y||_1 \le 1$:

$$\Pr[M(X) \in S] \le exp(\epsilon) \Pr[M(Y) \in S] + \delta,$$

where the probability space is over the coin flips of the mechanism M. If $\delta = 0$, we say that M is ϵ-differential private.

A simple and common way to produce differential private numeric functions, especially in the setting of privacy-preserving analysis, is to perturb the function output with noise sampled through special distributions.

Definition A.3 (Laplace Mechanism [18]). Given any function $f : \mathbb{N}^{|X|} \to \mathbb{R}^k$, the Laplace mechanism is defined as:

$$M_L(X, f(\cdot), \epsilon) = f(X) + (Y_1, \ldots, Y_k)$$

where Y_i are independent identical distributed random variables drawn from the Laplace Distribution $Lap(s_1(f)/\epsilon)$, centered at zero and $s_1(f)$ denotes the l_1-sensitivity of the function f[7]:

$$s_1(f) = max_{X,Y \in \mathbb{N}^{|X|}, ||X-Y||_1=1} ||f(X) - f(Y)||_1$$

B PROOF OF THEOREM 6.1

Proof. We start with the correctness claim which essentially follows directly from the definitions. More precisely, let

$$(\mathsf{pp}, \mathsf{msk}, \{sk_i\}_{i \in [n]}) \leftarrow \mathcal{S}^*.\mathtt{Setup}(\lambda, \mathcal{F}_n),$$
$$\mathtt{ct}_i = \mathcal{S}^*.\mathtt{Enc}(\mathtt{sk}_i, i, x_i), i \in [n],$$
$$\mathtt{dk}_{f^*} \leftarrow \mathcal{S}^*.\mathtt{KeyGen}(\mathtt{msk}, f, \mathbb{D}_f)$$

[7]Normally the l_1-sensitivity is denoted by Δf. However, since we're already using Δ elsewhere, we wanted to introduce a distinctive notation for the l_1-sensitivity of a function.

Our goal is to show that

$$\Pr\left[\mathcal{S}^*.\mathtt{Dec}(\mathtt{dk}_{f^*}, \mathtt{ct}_1, \ldots, \mathtt{ct}_n) = f(x_1, \ldots x_n) + v\right] = \Pr[v \xleftarrow{\mathbb{D}_f} \Delta]$$

for all $v \in \Delta$ and for all inputs x_i. By definition of $\mathcal{S}^*$, the decryption key $\mathtt{dk}_{f^*}$ is the result of invoking $\mathcal{S}.\mathtt{KeyGen}(\mathtt{msk}, f^*)$ for some chosen function f^*. Because of equation (4) and the correctness of $\mathcal{S}$, it follows that

$$\begin{aligned}&\Pr\left[\mathcal{S}^*.\mathtt{Dec}(\mathtt{dk}_{f^*}, \mathtt{ct}_1, \ldots, \mathtt{ct}_n) = f(x_1, \ldots x_n) + v\right]\\ &= \Pr\left[f^* = f + v\right]\end{aligned}$$

for all v. By construction of $\mathcal{S}^*.\mathtt{KeyGen}$, it holds that

$$\Pr[f^* = f + v] = \Pr[v \xleftarrow{\mathbb{D}_f} \Delta]$$

which shows the claim.

We show the security claim by a standard reduction argument. That is, we assume a PPT attacker $\mathcal{A}^*$ against $\mathcal{S}^*$ with non negligible advantage and construct a PPT attacker $\mathcal{A}$ against $\mathcal{S}$ that has the same non negligible advantage in the full-hiding security game. We have an oracle O for $\mathcal{A}$ that sets up the scheme and chooses a $\beta \leftarrow \{0, 1\}$, forwarding pp to $\mathcal{A}$. $\mathcal{A}$ acts as an oracle to $\mathcal{A}^*$ and forwards pp. $\mathcal{A}^*$ can adaptively make queries of the type decryption key query and ciphertext query. Ciphertext queries from $\mathcal{A}^*$ are simply forwarded to O and the responses are given back to $\mathcal{A}^*$.

For the decryption key queries, $\mathcal{A}^*$ chooses a triple $(f_l, v_{l,0}, v_{l,1})$ and sends it to $\mathcal{A}$.[8] $\mathcal{A}$ sets $f_{l,0} = f_l + v_{l,0}$ and $f_{l,1} = f_l + v_{l,1}$, and sends the request $(f_{l,0}, f_{l,1})$ to O, who generates the decryption key $\mathtt{dk}_l \leftarrow \mathcal{S}.\mathtt{KeyGen}(\mathtt{msk}, f_{l,\beta})$. $\mathcal{A}$ forwards $\mathtt{dk}_l$ to $\mathcal{A}^*$.

The queries of $\mathcal{A}^*$ need to fulfill the equation (3) that for all $l \in [q_{key}]$ and for all $(j_1, \ldots j_n) \in [q_{\mathtt{ct},1}] \times \cdots \times [q_{\mathtt{ct},n}]$, we have

$$f_l(x_{1,0}^{j_1}, \ldots, x_{n,0}^{j_n}) + v_{l,0} = f_l(x_{1,1}^{j_1}, \ldots, x_{n,1}^{j_n}) + v_{l,1}.$$

Because of this restriction the queries of $\mathcal{A}$ are valid with respect to condition (2), since

$$\begin{aligned}f_{l,0}(x_{1,0}^{j_1}, \ldots, x_{n,0}^{j_n}) &= f_l(x_{1,0}^{j_1}, \ldots, x_{n,0}^{j_n}) + v_{l,0}\\ &= f_l(x_{1,1}^{j_1}, \ldots, x_{n,1}^{j_n}) + v_{l,1}\\ &= f_{l,1}(x_{1,1}^{j_1}, \ldots, x_{n,1}^{j_n})\end{aligned}$$

for all $l \in [q_{key}]$ and for all $(j_1, \ldots, j_n) \in [q_{\mathtt{ct},1}] \times \cdots \times [q_{\mathtt{ct},n}]$.

$\mathcal{A}^*$ eventually outputs a bit $\beta' \in \{0, 1\}$. This β' is also the guess of $\mathcal{A}$. An overview of the reduction can be found in Figure 6.

Note that $\mathcal{A}$ perfectly simulates the oracle for $\mathcal{A}^*$. Hence, we have $\mathsf{Expt}_{\mathcal{A}}^{MIFE}(\beta) = \mathsf{Expt}_{\mathcal{A}}^{NMIFE}(\beta)$ for each $\beta \in \{0, 1\}$. In particular, it follows that $\mathcal{A}$ and $\mathcal{A}^*$ have the same advantage. □

C A MESSAGE-AND-NOISE-HIDING NOISY MULTI-INPUT FUNCTIONAL ENCRYPTION SCHEME FOR INNER PRODUCTS

C.1 Overview

As stated in Section 6 we want to show that we can use the transformation from Theorem 6.1 on an existing MIFE scheme to construct a concrete NMIFE scheme and to analyze its efficiency. The current state of the art does not include any efficient schemes that realize FE

[8]To keep the description simple, we directly state the noise values $v_{l,\beta}$ instead of the corresponding distributions $\mathbb{D}_{l,\beta}$.

for all functions. Existing FE usually support only inner product or quadratic functions. However, so far no FE for quadratic functions is known that has full-hiding security and allows for multi-inputs. Therefore we apply the transformation to an existing MIPE scheme.

More precisely, we take the scheme of Datta et al. [15] as starting point. In the following, we refer to it as *DOT* scheme. Note that only *linear* functions f_y are directly supported by *DOT*. However, an NMIFE deploys *affine* functions to add a noise value v to the function evaluation. That is, we need to extend *DOT* to support also affine functions, i.e., $f_{y,c}(x_1, \ldots, x_n)+c = y_1 \cdot x_1 + \cdots + y_n \cdot x_n + c$ with a constant value c such that correctness and full-hiding security are preserved. That is, our strategy is composed of the following two steps:

Step 1: Extend *DOT* to a new (correct and full-hiding) scheme *affine-DOT* that also supports affine functions

Step 2: Apply the transformation from Theorem 6.1 to construct a correct and message-and-noise-hiding NMIFE *noisy-DOT*

A simple way to achieve the first step with any FH-MIPE scheme would be that in the setup phase the data for slot 1 is set to a fix vector, e.g. $\vec{x} = (0, \ldots, 0, 1)$ resulting in a data set $X = ((0, \ldots, 0, 1) \parallel \vec{x}_1 \parallel \cdots \parallel \vec{x}_n) \in \mathbb{F}_q^{(n+1)\cdot m}$ with $\vec{x}_i \in \mathbb{F}_q^m$. For a requested decryption key dk for function f described by $(y, c) = ((\vec{y} \parallel \cdots \parallel \vec{y}), c)$ encode the constant at the beginning of dk, so that the key encodes $\hat{y} = ((0, \ldots, 0, c) \parallel \vec{y}_1 \parallel \cdots \parallel \vec{y}_n) \in \mathbb{F}_q^{(n+1)\cdot m}$. Evaluating the inner product results in $X \cdot \hat{y} = (0, \ldots, 0, 1) \cdot (0, \ldots, 0, c) + \sum_{i=1}^n \vec{x}_i \cdot \vec{y}_i = c + \sum_{i=1}^n \vec{x}_i \cdot \vec{y}_i$. This would mean to increase the size of the collected data set by one record $\vec{x} = (0, \ldots, 0, 1)$ and also increase the size of the decryption key dk, yielding more ciphertexts and larger decryption keys. This does not only impact the space requirement but also the effort for setting up the FH-MIPE scheme and evaluating the function.

In the following, we present another approach to turn the linear *DOT* scheme into an affine scheme and dub the new scheme *affine-DOT*. The approach exploits properties of the key generation algorithm for directly embedding the noise value. This has the benefit that neither the ciphertexts nor the decryption keys need to be enlarged. Thus, we consider this variant to be of interest on its own. Again we concentrate on the bounded scheme of Datta et al. [15]. We omit the mathematical foundations, since we described them before in Section 7.

C.2 Full-Hiding Bounded Multi-Input Functional Encryption Scheme for Affine Functions

In this subsection we describe the *affine-DOT* scheme for affine functions, build upon the scheme *DOT* for linear functions, and prove its correctness and security. Before we proceed, we give an overview of the underlying idea of the transformation.

DOT is build upon the single input FE scheme proposed by Tomida et al. [44]. The naive approach to build a MIFE scheme out of a single-input FE scheme is to use an n-fold extension. Consider a master secret key msk that consists of n independently generated master secret keys msk_i for the single-input scheme. A ciphertext for some vector $\vec{x}_i \in \mathbb{F}_q^m$ with respect to index i is simply a single-input FE ciphertext for $\vec{x}_i \in \mathbb{F}_q^m$ with respect to the master secret key msk_i for slot i. A decryption key for a set of n vectors $\{\vec{y}_i\}_{i\in[n]}$ for $\vec{y}_i \in \mathbb{F}_q^m$ is given by the set of decryption keys $\{\mathsf{dk}_{\vec{y}_i}\}_{i\in[n]}$ with respect to the master secret keys of each slot. Decrypting multiple inputs simply means to decrypt each input individually and to combine the results afterwards.

However this construction is not secure, since an attacker can easily recover $\vec{x}_i \cdot \vec{y}_i$ for a single i, instead of only receiving the final result $\sum_{i=1}^n \vec{x}_i \cdot \vec{y}_i$. For example, this would render the property of message-hiding impossible. Datta et al. [15] solves this problem analogue to Abdalla et al. [1] by introducing additional randomness in the ciphertexts and the decryption keys such that the random values eventually add up to zero. Our main idea is to choose the random values such that they do not add up to zero but to the constant value c in the affine function f described by $((\vec{y}_1 \parallel \cdots \parallel \vec{y}_n), c)$ with $f(\vec{x}_1 \parallel \cdots \parallel \vec{x}_n) = \sum_{i=1}^n \vec{x}_i \cdot \vec{y}_i + c$.

Description.

affine-DOT.$\mathsf{Setup}(\lambda, \mathcal{F}_n)$**:** This algorithm takes the security parameter λ, a description of $\mathcal{F}_n = \{m, n, \boxed{\Delta}, B\}$ with the length $m \in \mathbb{N}$ of the vectors, the arity $n \in \mathbb{N}$ of the multi-input functionality, $\boxed{\Delta \subsetneq \mathbb{N} \text{ of polynomial size}}$ and a bound $B \in \mathbb{N}$ on each component inner product. It works as follows:

(1) Setup a bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ where the group sizes are chosen to be a prime number $q \gg nB + \boxed{\max_{\delta\in\Delta}(\delta)}$,

(2) Extend e to a pairing on a dual vector space $\mathbb{V}_1 \times \mathbb{V}_2$ with $\mathbb{V}_1 = \mathbb{G}_1^{2m+2k+1}$ and $\mathbb{V}_2 = \mathbb{G}_2^{2m+2k+1}$ for an appropriately chosen parameter k, associated with the k-LIN assumption.

(3) Sample random $\zeta \xleftarrow{\$} \mathbb{F}_q \backslash \{0\}$, and compute $g_T = e(g_1, g_2)^\zeta$ with g_1 and g_2 being the generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively.

(4) For $i \in [n]$, generate a dual orthogonal basis $\mathbb{B}_i = \{\mathbf{b}_{i,1}, \ldots, \mathbf{b}_{i,2m+2k+1}\}$ and $\mathbb{B}_i^* = \{\mathbf{b}_{i,1}^*, \ldots, \mathbf{b}_{i,2m+2k+1}^*\}$ such that $e(\mathbf{b}_{i,j}, \mathbf{b}_{i,j}^*) = g_T$ for all $j \in [2m+2k+1]$ and set

$$\hat{\mathbb{B}}_i = \{\mathbf{b}_{i,1}, \ldots, \mathbf{b}_{i,m}, \mathbf{b}_{i,2m+1}, \mathbf{b}_{i,2m+k+1}, \ldots, \mathbf{b}_{i,2m+2k}\},$$
$$\hat{\mathbb{B}}_i^* = \{\mathbf{b}_{i,1}^*, \ldots, \mathbf{b}_{i,m}^*, \mathbf{b}_{i,2m+1}^*, \ldots, \mathbf{b}_{i,2m+k}^*\}.$$

(5) Publish public parameters $\mathsf{pp} = (\mathsf{params}_{\mathbb{G}}, g_T)$ and set the master secret key $\mathsf{msk} = \{\hat{\mathbb{B}}_i^*\}_{i\in[n]}$. Let $\mathsf{sk}_i = \hat{\mathbb{B}}_i$ be the secret encryption key for slot i. All of the remaining algorithms implicitly take pp.

affine-DOT.$\mathsf{Enc}(\mathsf{sk}_i, i, \vec{x}_i)$**:** Takes as input the secret key sk_i, an index $i \in [n]$, a vector $\vec{x}_i \in \mathbb{F}_q^m$ and performs the following steps:

(1) Select random $\varphi_{i,1}, \ldots, \varphi_{i,k} \xleftarrow{\$} \mathbb{F}_q$, and compute

$$\mathbf{c}_i = \sum_{j\in[m]} x_i^{(j)} \mathbf{b}_{i,j} + \mathbf{b}_{i,2m+1} + \sum_{j\in[k]} \varphi_{i,j} \mathbf{b}_{i,2m+k+j}$$
$$= (\vec{x}_i, \vec{0}^m, 1, \vec{0}^{k-1}, \varphi_{i,1}, \ldots, \varphi_{i,k}, 0)_{\mathbb{B}_i}$$

(2) Output ciphertext $\mathsf{ct}_i = (i, \mathbf{c}_i)$.

***affine-DOT*.**$\mathsf{KeyGen}(\mathsf{msk}, f)$**:** On input the msk and a function described by a set of vectors $\{\vec{y}_i\}_{i\in[n]}$ and a $\boxed{\text{constant } c \in \Delta}$, execute the following steps:

(1) Sample random $r_i, \gamma_{i,1}, \dots, \gamma_{i,k-1} \xleftarrow{\$} \mathbb{F}_q$, for $i \in [n]$, with the restriction that $\sum_{i\in[n]} r_i = \boxed{c.}$

(2) For each $i \in [n]$ compute

$$\mathbf{k}_i = \sum_{j\in[m]} y_i^{(j)} \mathbf{b}^*_{i,j} + r_i \mathbf{b}^*_{i,2m+1} + \sum_{j\in[k-1]} \gamma_{i,j} \mathbf{b}^*_{i,2m+1+j}$$
$$= (\vec{y}_i, \vec{0}^m, r_i, \gamma_{i,1}, \dots, \gamma_{i,k-1}, \vec{0}^k, 0)_{\mathbb{B}^*_i},$$

where $\vec{y}_i = (y_1^{(1)}, \dots, y_1^{(m)})$.

(3) Output decryption key $\mathsf{dk} = \{\mathbf{k}_i\}_{i\in[n]}$

***affine-DOT*.**$\mathsf{Dec}(\mathsf{dk}, \mathsf{ct}_1, \dots, \mathsf{ct}_n)$**:** This algorithm takes a decryption key $\mathsf{dk} = \{\mathbf{k}_i\}_{i\in[n]}$, n ciphertexts $\mathsf{ct}_1, \dots \mathsf{ct}_n$ and computes the following:

(1) first compute $L_T = \prod_{i\in[n]} e(\mathbf{c}_i, \mathbf{k}_i)$

(2) attemp to determine a value $\Lambda \in \mathbb{Z}$ such that $g_T^\Lambda = L_T$ by performing an exhaustive search over a specific polynomial-size range of possible values. If it succeeds, output Λ, else output $\perp$.[9]

Correctness. The correctness of the above MIFE scheme *affine-DOT* can be verified easily and is similar to the correctness of *DOT*. For any set of ciphertexts $\{\mathsf{ct}_i = (i, \mathbf{c}_i)\}_{i\in[n]}$ with

$$\mathbf{c}_i = (\vec{x}_i, \vec{0}^m, 1, \vec{0}^{k-1}, \varphi_{i,1}, \dots, \varphi_{i,k}, 0)_{\mathbb{B}_i}$$

where ct_i encrypts some vector $\vec{x}_i \in \mathbb{F}_q^m$ with respect to index $i \in [n]$ and any decryption key $\mathsf{dk}_f = \{\mathbf{k}_i\}_{i\in[n]}$ corresponding to a function f described by $(\{\vec{y}_i\}_{i\in[n]}, c)$such that $\vec{y}_i \in \mathbb{F}_q^m$ for all $i \in [n]$ and $c \in \Delta$ with $\mathbf{k}_i = (\vec{y}_i, \vec{0}^m, r_i, \gamma_{i,1}, \dots, \gamma_{i,k-1}, \vec{0}^k, 0)_{\mathbb{B}^*_i}$ and therefore

$$L_T = \prod_{i\in[n]} e(\mathbf{c}_i, \mathbf{k}_i) = \prod_{i\in[n]} g_T^{\mathbf{c}_i\cdot\mathbf{k}_i}$$
$$= \prod_{i\in[n]} g_T^{\vec{x}_i\cdot\vec{y}_i + r_i} = g_T^{\sum_{i\in[n]}(\vec{x}_i\cdot\vec{y}_i + r_i)}$$
$$= g_T^{c+\sum_{i\in[n]}\vec{x}_i\cdot\vec{y}_i}$$

This follows from the fact that for each $i \in [n]$, $\mathbb{B}_i$ and $\mathbb{B}^*_i$ are dual orthogonal bases and $\sum_{i\in[n]} r_i = c$. Since Δ is of polynomial size and therefore, also the range $[nB + \max_{\delta\in\Delta}]$ is of polynomial size. Thus, if $c + \sum_{i\in[n]} \vec{x}_i \cdot \vec{y}_i$ is in the polynomial size range of possible values that the decryption algorithm searches, the algorithm would output the correct $\Lambda = c + \sum_{i\in[n]} \vec{x}_i \cdot \vec{y}_i$.

Security. We prove that our *affine-DOT* scheme is again a full-hiding scheme under the restriction that the adversary makes at least one ciphertext query per slot. For more details how to achieve full-hiding security without that restriction, we refer to [15]. Our proof follows the same logic as the proof of scheme *DOT*. Therefore we refer to the full version of the paper of Datta et al. [14] for most parts of the proof and only want to discuss the parts that change.

The proof of *DOT* is structured as a hybrid argument over a series of experiments which differ in the construction of the decryption keys and/or ciphertexts. The first experiment reflects the security experiment $\mathrm{Expt}^{MIFE}_{\mathcal{A}}(0)$. They progressively change the decryption keys and ciphertexts in multiple steps to those in security experiment $\mathrm{Expt}^{MIFE}_{\mathcal{A}}(1)$. Proving that each hybrid is indistinguishable from the previous one, yielding the full-hiding security.

We discuss now how this proof can be adapted to cover *affine-DOT*. First of all, an adversary needs to be allowed to make decryption queries for affine functions. That is, whenever a i^{th} decryption key query $(\{\vec{y}_{\iota,i,0}\}_{\iota\in[n]}, \{\vec{y}_{\iota,i,1}\}_{\iota\in[n]})$ for scheme *DOT* is made, we require a decryption key query

$$(f_{i,0}, f_{i,1}) = ((\{\vec{y}_{\iota,i,0}\}_{\iota\in[n]}, c_{i,0}), (\{\vec{y}_{\iota,i,1}\}_{\in[n]}, c_{i,1}))$$

with $c_{i,0}, c_{i,1} \in \Delta$.

The proof description contains two different sequences of random values: $\{r_{\iota,i}\}_{\iota\in[n]}$ and $\{\tilde{r}_{\iota,i}\}_{\iota\in[n]}$. The former are used in situations where $f_{i,0}$ is encoded and consequently the latter for the case of $f_{i,1}$. To preserve correctness, whenever values $r_{\iota,i}$ are sampled uniformly and random from $\mathbb{F}_q$ we require that $\sum_{\iota\in[n]} r_{\iota,i} = c_{i,0}$ instead of $\sum_{\iota\in[n]} r_{\iota,i} = 0$ and analogously $\sum_{\iota\in[n]} \tilde{r}_{\iota,i} = c_{i,1}$.

As long as these two cases are separated, that is the values $r_{\iota,i}$ are only used in conjunction with $f_{i,0}$ and likewise $\tilde{r}_{\iota,i}$ with $f_{i,1}$, this has no impact on the security arguments. Here, we also make use of the fact that in the original proof, the conditions $\sum_{\iota\in[n]} r_{\iota,i} = 0$ and $\sum_{\iota\in[n]} \tilde{r}_{\iota,i} = 0$ have only been used to ensure correctness.

The only potentially critical step is transformation from $f_{i,0}$ to $f_{i,1}$ which is covered in lemma C.5. The indistinguishability is argued by a change of the dual orthogonal bases $\mathbb{B}_i$ and $\mathbb{B}^*_i$. This change also induces a change of the $r_{\iota,i}$ values, namely as follows:[10]

$$\tilde{r}_{\iota,i} = r_{\iota,i} + \vec{x}_{\iota,1,0} \cdot \vec{y}_{\iota,\nu,0} - \vec{x}_{\iota,1,1} \cdot \vec{y}_{\iota,\nu,1}. \tag{18}$$

Because of the restriction on the queries of $\mathcal{A}$ that

$$f_{i,0}(\vec{x}_{1,1,0}, \dots, \vec{x}_{n,1,0}) = f_{i,1}(\vec{x}_{1,1,1}, \dots, \vec{x}_{n,1,1})$$

(see Equation (2)), it follows:

$$\sum_{\iota\in[n]} \tilde{r}_{\iota,i} = \sum_{\iota\in[n]} (r_{\iota,i} + \vec{x}_{\iota,1,0} \cdot \vec{y}_{\iota,\nu,0} - \vec{x}_{\iota,1,1} \cdot \vec{y}_{\iota,\nu,1})$$
$$= \sum_{\iota\in[n]} r_{\iota,i} + \sum_{\iota\in[n]} \vec{x}_{\iota,1,0} \cdot \vec{y}_{\iota,\nu,0} - \sum_{\iota\in[n]} \vec{x}_{\iota,1,1} \cdot \vec{y}_{\iota,\nu,1}$$
$$= c_{\nu,0} + \sum_{\iota\in[n]} \vec{x}_{\iota,1,0} \cdot \vec{y}_{\iota,\nu,0} + (c_{\nu,1} - c_{\nu,1}) \sum_{\iota\in[n]} \vec{x}_{\iota,1,1} \cdot \vec{y}_{\iota,\nu,1}$$
$$= c_{\nu,1}$$

This shows that the random values after the transformation automatically fulfill the condition $\sum_{\iota\in[n]} \tilde{r}_{\iota,i} = c_{i,1}$. Moreover, as the values $r_{\iota,i}$ have been uniformly sampled (under the condition to sum up to $c_{i,0}$), the same property holds as well for the values $\tilde{r}_{\iota,i}$.

A further adaptation that is required affects values $\theta_{\iota,\nu}$ used in the proof of Lemma C.4. We require that the random values $\theta_{\iota,\nu}$ sum up to $\sum_{\iota\in[n]} \theta_{\iota,\nu} = c_{i,0}$ instead of 0 to ensure that $\sum_{\iota\in[n]} r_{\iota,i} = c_{i,0}$. To understand why this has no impact on the security claim, we have to quickly explain the role of these values. In lemma C.4 the advantage of deciding between to hybrid sequences is reduced to

[9]Similar exhaustive search steps are part of all bilinear map-based inner product constructions. The polynomial running time is guaranteed by restricting the output to lie within a fixed polynomial size range [15].

[10]Please be aware that in the original Paper of Datta et al. [14] there is a construction error. The $\tilde{r}_{\iota,i}$ need to be constructed as specified in Equation (18).

a decisional problem 1^* [15, Def. 2.6], that can be reduced itself to the k-LIN problem. In the proof, the values $r_{\iota,i}$ are replaced by values taken from the considered problem instance. In addition, the values $\theta_{\iota,\nu}$ are added for masking the problem-instance-values and to ensure that the values $r_{\iota,i}$ are still uniformly sampled. Because of $\sum_{\iota\in[n]} r_{\iota,i} = \sum_{\iota\in[n]} \theta_{\iota,i}$, the change mentioned above is necessary to maintain correctness, while they still mask the values of the problem instance.

Through applying the transformation presented in Section 6 we construct an equally efficient MN-H NMIFE scheme, which is dubbed *affine-DOT*.

D PROOF OF THEOREM 7.2

To prove Lemma 7.3, we need to reduce the advantage of an adversary to the k-LIN assumption. To this end, we follow the approach of Datta et al. [14]. In order to shorten the respective proofs and focus on the main idea, this reduction will not be done directly. Instead, they presented two problems (Problem 1 and 2), which can be reduced to a third problem (Problem 0) which in turn can be reduced to the k-LIN assumption. Since we modified the construction of the ciphertexts and keys in our scheme DiffPIPE, missing $m-1$ and $n(m-1)$ slots, we cannot directly adapt their problems. Therefore, we describe two problems, (Problem 1 and Problem 2), which are mostly similar but adapted the structure of our ciphertexts and decryption keys, respectively. For these, we show that they can be reduced to another problem, Problem 0, which is equal to the problem used in [14].

In favour of readability, Lemma 7.3 and Lemma 7.4 are broken down to a series of lemmas (Lemma D.3- Lemma D.9) and proven individually. Combined with Lemma D.2, this concludes the proof of Theorem 7.2.

Let us start by defining Problem 0 as well as the follow up problems, Problem 1 and Problem 2.

Problem 0 ([14]). *Fix an arbitrary number $\chi \in [2]$ Problem 0 is to guess a bit $\hat{\beta} \xleftarrow{\$} \{0,1\}$ given $\vartheta_{\hat{\beta}} = (\mathtt{params}_{\mathbb{G}}, \mathbb{D}, \hat{\mathbb{D}}^*, g_\chi^\zeta, \Omega_{\hat{\beta}})$; where $\mathtt{params}_{\mathbb{G}} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e) \xleftarrow{\$} \mathcal{G}_{BPG}()$; $\zeta, \pi \xleftarrow{\$} \mathbb{F}_q \setminus \{0\}$; $\alpha_1, \dots, \alpha_k \xleftarrow{\$} \mathbb{F}_q$; $D = (d_{j,t})_{(k+1)\times(k+1)} \xleftarrow{\$} \mathtt{GL}(k+1, \mathbb{F}_q)$; $D^\star = (d^*_{j,t})_{(k+1)\times(k+1)} = \zeta D^*$; $\boldsymbol{d}_j = (g_\chi^{d_{j,1}}, \dots, g_\chi^{d_{j,k+1}})$, $\boldsymbol{d}^*_j = (g_{3-\chi}^{d^*_{j,1}}, \dots, g_{3-\chi}^{d^*_{j,k+1}})$ for $j \in [k+1]$; $\mathbb{D} = \{\boldsymbol{d}_1, \dots, \boldsymbol{d}_{k+1}\}$; $\hat{\mathbb{D}}^* = \{\boldsymbol{d}^*_1\}$ and $\Omega_{\hat{\beta}} = (\alpha_1, \dots, \alpha_k, 0)_{\mathbb{D}}$ or $\Omega_{\hat{\beta}} = (\alpha_1, \dots, \alpha_k, \pi)_{\mathbb{D}}$ according as $\hat{\beta}$ is 0 or 1. For any PPT algorithm $\mathcal{A}$, the advantage of $\mathcal{A}$ in deciding Problem 0 is defined as*

$$Adv_{\mathcal{A}}^{P_0}(\lambda) = |\Pr[1 \leftarrow \mathcal{A}(\vartheta_0)] - \Pr[1 \leftarrow \mathcal{A}(\vartheta_1)]|.$$

Lemma D.1 ([14]). *For any PPT algorithm $\mathcal{A}$ for Problem 0, there exists a PPT algorithm $\mathcal{B}$ for the k-LIN assumption such that for any security parameter λ, we have $Adv_{\mathcal{A}}^{P_0}(\lambda) \le Adv_{\mathcal{B}}^{k-LIN}(\lambda)$.*

Problem 1 is used in the games that apply changes in the construction of the keys.

Problem 1. *Problem 1 is to guess a bit $\beta \xleftarrow{\$} \{0,1\}$ given $\theta_\beta = (\mathtt{params}_{\mathbb{G}}, g_T, \{\hat{\mathbb{B}}_i, \hat{\mathbb{B}}^*_i\}_{i\in[n]}, \{\phi_{i,\beta}\}_{i\in[n]})$; where $\mathtt{params}_{\mathbb{G}} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e) \xleftarrow{\$} \mathcal{G}_{BPG}()$; $\mathtt{params}_{\mathbb{V}} = (q, \mathbb{V}_1, \mathbb{V}_2, \mathbb{G}_T, \mathbb{A}_1, \mathbb{A}_2, e) \xleftarrow{\$} \mathcal{G}_{DPVS}(m+2k+2, \mathtt{params}_{\mathbb{G}})$; $o \xleftarrow{\$} \mathbb{F}_q \setminus \{0\}$; $g_t = e(g_1, g_2)^o$; $(\mathbb{B}_i, \mathbb{B}^*_i) \xleftarrow{\$} \mathcal{G}_{OB}(m+2k+2, \mathtt{params}_{\mathbb{G}}, o)$, $\hat{\mathbb{B}}_i = \{\boldsymbol{b}_{i,1}, \dots, \boldsymbol{b}_{i,m}, \boldsymbol{b}_{i,m+1}, \boldsymbol{b}_{i,m+k+2}, \dots, \boldsymbol{b}_{i,m+2k+1}\}$, $\hat{\mathbb{B}}^*_i = \{\boldsymbol{b}^*_{i,1}, \dots, \boldsymbol{b}^*_{i,m}, \boldsymbol{b}^*_{i,m+1}, \boldsymbol{b}^*_{i,m+3}, \dots, \boldsymbol{b}^*_{i,m+k+1}, \boldsymbol{b}^*_{i,m+2k+2}\}$ for $i \in [n]$; $\alpha_1, \dots, \alpha_k \xleftarrow{\$} \mathbb{F}_q$, $\pi \xleftarrow{\$} \mathbb{F}_q \setminus \{0\}$; and $\phi_{i,0} = (\vec{0}^m, \alpha_1, 0, \alpha_2, \dots, \alpha_k, \vec{0}^k, 0)_{\mathbb{B}^*_i}$, $\phi_{i,1} = (\vec{0}^m, \alpha_1, 0, \alpha_2, \dots, \alpha_k, \vec{0}^k, \pi)_{\mathbb{B}^*_i}$. For any PPT algorithm $\mathcal{A}$, the advantage of $\mathcal{A}$ in deciding Problem 1 is defined as*

$$Adv_{\mathcal{A}}^{P_1}(\lambda) = |\Pr[1 \leftarrow \mathcal{A}(\theta_0)] - \Pr[1 \leftarrow \mathcal{A}(\theta_1)]| \le \mathtt{negl}(\lambda)$$

for some negligible function negl.

Problem 2 is essentially the same as Problem 1, with the exception, that it concerns the basis vectors used for the construction of the ciphertexts.

Problem 2. *Problem 2 is to guess a bit $\beta \xleftarrow{\$} \{0,1\}$ given $\theta_\beta = (\mathtt{params}_{\mathbb{G}}, g_T, \{\hat{\mathbb{B}}_i, \hat{\mathbb{B}}^*_i\}_{i\in[n]}, \{\phi_{i,\beta}\}_{i\in[n]})$; where $\mathtt{params}_{\mathbb{G}} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e) \xleftarrow{\$} \mathcal{G}_{BPG}()$; $\mathtt{params}_{\mathbb{V}} = (q, \mathbb{V}_1, \mathbb{V}_2, \mathbb{G}_T, \mathbb{A}_1, \mathbb{A}_2, e) \xleftarrow{\$} \mathcal{G}_{DPVS}(m+2k+2, \mathtt{params}_{\mathbb{G}})$; $o \xleftarrow{\$} \mathbb{F}_q \setminus \{0\}$; $g_t = e(g_1, g_2)^o$; $(\mathbb{B}_i, \mathbb{B}^*_i) \xleftarrow{\$} \mathcal{G}_{OB}(m+2k+2, \mathtt{params}_{\mathbb{G}}, o)$, $\hat{\mathbb{B}}_i = \{\boldsymbol{b}_{i,1}, \dots, \boldsymbol{b}_{i,m}, \boldsymbol{b}_{i,m+1}, \boldsymbol{b}_{i,m+k+2}, \dots, \boldsymbol{b}_{i,m+2k+1}\}$, $\hat{\mathbb{B}}^*_i = \{\boldsymbol{b}^*_{i,1}, \dots, \boldsymbol{b}^*_{i,m}, \boldsymbol{b}^*_{i,m+1}, \boldsymbol{b}^*_{i,m+3}, \dots, \boldsymbol{b}^*_{i,m+k+1}, \boldsymbol{b}^*_{i,m+2k+2}\}$ for $i \in [n]$; $\alpha_1, \dots, \alpha_k \xleftarrow{\$} \mathbb{F}_q$, $\pi \xleftarrow{\$} \mathbb{F}_q \setminus \{0\}$; and $\phi_{i,0} = (\vec{0}^{m+k+1}, \alpha_1, \dots, \alpha_k, 0)_{\mathbb{B}_i}$, $\phi_{i,1} = (\vec{0}^{m+k+1}, \alpha_1, \dots, \alpha_k, \pi)_{\mathbb{B}_i}$. For any PPT algorithm $\mathcal{A}$, the advantage of $\mathcal{A}$ in deciding Problem 2 is defined as*

$$Adv_{\mathcal{A}}^{P_2}(\lambda) = |\Pr[1 \leftarrow \mathcal{A}(\theta_0)] - \Pr[1 \leftarrow \mathcal{A}(\theta_1)]| \le \mathtt{negl}(\lambda)$$

for some negligible function negl.

Next, we show that both Problem 1 and Problem 2 can be reduced to the k-LIN assumption by using Lemma D.1.

Lemma D.2. *For any PPT algorithm $\mathcal{A}$ for Problem $i \in [2]$, there exists a PPT algorithm $\mathcal{B}$ for the k-LIN assumption such that for any security parameter λ, we have $Adv_{\mathcal{A}}^{P_i}(\lambda) \le Adv_{\mathcal{B}}^{k-LIN}(\lambda)$*

Proof. First, we are going to show the reduction for Problem 1 and then state the differences for Problem 2. Suppose there exists PPT adversary $\mathcal{A}$ for Problem 1, which can be used by a PPT algorithm $\mathcal{B}$ as a subroutine trying to solve an instance of Problem 0. $\mathcal{B}$ is given this particular instance corresponding to $\chi = 2$,

$$\vartheta_{\hat{\beta}} = (\mathtt{params}_{\mathbb{G}}, \mathbb{D}, \hat{\mathbb{D}}^*, g_\chi^\zeta, \Omega_{\hat{\beta}}),$$

where everything is chosen in accordance with the notation of Problem 0. Then, $\mathcal{B}$ carries out the following steps:

- $\mathcal{B}$ generates $\mathtt{params}_{\mathbb{V}} = (q, \mathbb{V}_1, \mathbb{V}_2, \mathbb{G}_T, \mathbb{A}_1, \mathbb{A}_2, e) \xleftarrow{\$} \mathcal{G}_{DPVS}(m+2k+2, \mathtt{params}_{\mathbb{G}})$ and calculates $g_T = e(g_1, g_2^\zeta)$.
- For each $i \in [n]$, $\mathcal{B}$ acts as followed:
  - Sample a random invertible matrix $W_i = (w_{i,j,t})_{(m+2k+2)\times(m+2k+2)} \xleftarrow{\$} \mathtt{GL}(m+2k+2, \mathbb{F}_q)$

– Compute

$$\begin{aligned}
\boldsymbol{b}^*_{i,j} &= (\mathbf{1}^{k+j}_{\mathbb{G}_2}, g_2^{\zeta}, \mathbf{1}^{k+m+1-j}_{\mathbb{G}_2})W_i && j \in [m]\\
\boldsymbol{b}^*_{m+1} &= (\boldsymbol{d}_1, \mathbf{1}^{m+k+1}_{\mathbb{G}_2})W_i\\
\boldsymbol{b}^*_{m+2} &= (\mathbf{1}^{k+m+1}_{\mathbb{G}_2}, g_2^{\zeta}, \mathbf{1}^{k}_{\mathbb{G}_2})W_i\\
\boldsymbol{b}^*_{m+1+j} &= (\boldsymbol{d}_j, \mathbf{1}^{m+k+1}_{\mathbb{G}_2})W_i && j \in [2,k]\\
\boldsymbol{b}^*_{m+1+k+j} &= (\mathbf{1}^{k+m+1+j}_{\mathbb{G}_2}, g_2^{\zeta}, \mathbf{1}^{k-j}_{\mathbb{G}_2})W_i && j \in [k]\\
\boldsymbol{b}^*_{m+2k+2} &= (\boldsymbol{d}_{k+1}, \mathbf{1}^{m+k+1}_{\mathbb{G}_2})W_i\\
\boldsymbol{b}_{i,j} &= (\mathbf{1}^{k+j}_{\mathbb{G}_1}, g_1, \mathbf{1}^{k+m+1-j}_{\mathbb{G}_1})W^*_i && j \in [m]\\
\boldsymbol{b}_{i,m+1} &= (\boldsymbol{d}^*_1, \mathbf{1}^{k+m+1}_{\mathbb{G}_1})W^*_i\\
\boldsymbol{b}_{i,m+2} &= (\mathbf{1}^{k+m+1}_{\mathbb{G}_1}, g_1, \mathbf{1}^{k}_{\mathbb{G}_1})W^*_i\\
\boldsymbol{b}_{i,m+1+k+j} &= (\mathbf{1}^{k+m+1+j}_{\mathbb{G}_1}, g_1, \mathbf{1}^{k-j}_{\mathbb{G}_1})W^*_i && j \in [k]
\end{aligned}$$

– As $\mathcal{B}$ does not know the precise values of $\boldsymbol{d}^*_j$ for $j > 1$, they implicitly set

$$\boldsymbol{b}_{i,m+1+j} = (\boldsymbol{d}^*_j, \mathbf{1}^{k+m+1}_{\mathbb{G}_1})W^*_i j \in [2,k]$$
$$\boldsymbol{b}_{i,m+2k+2} = (\boldsymbol{d}^*_{k+1}, \mathbf{1}^{k+m+1}_{\mathbb{G}_1})W^*_i$$

to implicitly obtain

$$\mathbb{B}_i = \{\boldsymbol{b}_{i,1}, \dots, \boldsymbol{b}_{i,m+2k+2}\}$$
$$\mathbb{B}^*_i = \{\boldsymbol{b}^*_{i,1}, \dots, \boldsymbol{b}^*_{i,m+2k+2}\}.$$

These are indeed dual orthogonal bases of $\mathbb{V}_1$ and $\mathbb{V}_2$ which are uniformly and independently distributed due to the sampling of W_i. Moreover, $e(\boldsymbol{b}_{i,j}, \boldsymbol{b}^*_{i,j}) = g_T \ \forall j \in [m+2k+2]$.

– $\mathcal{B}$ defines

$$\hat{\mathbb{B}}_i = \{\boldsymbol{b}_{i,1}, \dots, \boldsymbol{b}_{i,m+1}, \boldsymbol{b}_{i,m+k+2}, \dots, \boldsymbol{b}_{m+2k+1}\}$$
$$\hat{\mathbb{B}}^*_i = \{\boldsymbol{b}^*_{i,1}, \dots, \boldsymbol{b}^*_{i,m+1}, \boldsymbol{b}^*_{i,m+3}, \dots \boldsymbol{b}^*_{i,m+k+1}, \boldsymbol{b}^*_{i,m+2k+2}\}.$$

Note that all of these values are known to cB.

– Define $\phi_{i,\hat{\beta}} = (\Omega_{\hat{\beta}}, \mathbf{1}^{m+k+1}_{\mathbb{G}_2})W_i$.

• $\mathcal{B}$ hands $\theta_{\hat{\beta}} = (\texttt{params}_{\mathbb{G}}, g_T, \{\hat{\mathbb{B}}_i, \hat{\mathbb{B}}^*_i\}_{i\in[n]}, \{\phi_{i,\beta}\}_{i\in[n]})$ to $\mathcal{A}$ and outputs the same guessing bit as $\mathcal{A}$. Note that

$$\phi_{i,\hat{\beta}} = (\Omega_{\hat{\beta}}, \mathbf{1}^{m+k+1}_{\mathbb{G}_2})W_i \tag{19}$$
$$= \alpha_1(\boldsymbol{d}_j, \mathbf{1}^{m+k+1}_{\mathbb{G}_2})W_i + \sum_{j=2}^{k}\alpha_j(\boldsymbol{d}_j, \mathbf{1}^{m+k+1}_{\mathbb{G}_2})W_i \tag{20}$$
$$+ \psi_{\hat{\beta}}(\boldsymbol{d}_{k+1}, \mathbf{1}^{m+k+1}_{\mathbb{G}_2})W_i \tag{21}$$
$$= \alpha_1\boldsymbol{b}^*_{m+1} + \sum_{j=2}^{k}\alpha_j\boldsymbol{b}^*_{m+1+j} + \psi_{\hat{\beta}}\boldsymbol{b}^*_{m+2k+2} \tag{22}$$
$$= (\vec{0}^m, \alpha_1, 0, \alpha_2, \dots, \alpha_k, \vec{0}^k, \psi_{\hat{\beta}})_{\mathbb{B}^*_i}, \tag{23}$$

where $\psi_{\hat{\beta}} = \pi$ if $\hat{\beta} = 1$ and 0 else. Therefore, $\theta_{\hat{\beta}}$ is indeed an instance of Problem 1 simulated by $\mathcal{B}$ with challenge bit $\hat{\beta}$. Thus, the claim follows for $i = 1$.

As mentioned, the proof for $i = 2$ is very similar. Instead of $\chi = 2$, we choose $\chi = 1$ and let $\mathcal{B}$ form the base vectors in the following manner:

$$\begin{aligned}
\boldsymbol{b}_{i,j} &= (\mathbf{1}^{j+k}_{\mathbb{G}_1}, g_1^{\zeta}, \mathbf{1}^{m+k+1-j}_{\mathbb{G}_1})W_i && j \in [m+k+1]\\
\boldsymbol{b}_{i,m+k+1+j} &= (\boldsymbol{d}_j, \mathbf{1}^{m+k+1}_{\mathbb{G}_1})W_i && j \in [k+1]\\
\boldsymbol{b}^*_{i,j} &= (\mathbf{1}^{j+k}_{\mathbb{G}_2}, g_2, \mathbf{1}^{m+k+1-j}_{\mathbb{G}_2})W^*_i && j \in [m+k+1]\\
\boldsymbol{b}_{i,m+k+2} &= (\boldsymbol{d}^*_1, \mathbf{1}^{m+k+1}_{\mathbb{G}_2})W^*_i
\end{aligned}$$

Afterwards, $\mathcal{B}$ continues as before by building the reduced bases as in Problem 2. This concludes the lemma. □

The following computation analyzes the security of Section 7 in greater detail. By definitions of the games, we have

$$\begin{aligned}
\mathsf{Adv}^{Game_0}_{\mathcal{A}}(\lambda) &= \Pr[\mathsf{Expt}^{NMIFE}_{\mathcal{A}}(0) = 1],\\
\mathsf{Adv}^{Game_{1,0,3}}_{\mathcal{A}}(\lambda) &= \mathsf{Adv}^{Game_0}_{\mathcal{A}}(\lambda),\\
\mathsf{Adv}^{Game_{2,0,3}}_{\mathcal{A}}(\lambda) &= \mathsf{Adv}^{Game_{1,q_{key},3}}_{\mathcal{A}}(\lambda),\\
\mathsf{Adv}^{Game_4}_{\mathcal{A}}(\lambda) &= \Pr[\mathsf{Expt}^{NMIFE}_{\mathcal{A}}(1) = 1].
\end{aligned}$$

Also the transition from Game$_3$ to Game$_4$ is essentially the reverse transition of the Game$_1$ sequence with $\vec{x}_{i,0}$ and $\vec{x}_{i,1}$ interchanged. Therefore, we have:

$$\begin{aligned}
\mathsf{Adv}_{\mathcal{A}}(\lambda) \leq 2 \cdot \sum_{l\in[q_{key}]} &\Big(|\mathsf{Adv}^{Game_{1,l-1,3}}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{Game_{1,l,1}}_{\mathcal{A}}(\lambda)|\\
&+ \sum_{j\in[2,3]} |\mathsf{Adv}^{Game_{1,l,j-1}}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{Game_{1,l,j}}_{\mathcal{A}}(\lambda)|\Big)\\
&+ \sum_{i\in[n]} \Big(|\mathsf{Adv}^{Game_{2,i-1,3}}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{Game_{2,i,1}}_{\mathcal{A}}(\lambda)|\\
&+ \sum_{j\in[2,3]} |\mathsf{Adv}^{Game_{2,i,j-1}}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{Game_{2,i,j}}_{\mathcal{A}}(\lambda)|\Big)\\
&+ |\mathsf{Adv}^{Game_{2,n,3}}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{Game_3}_{\mathcal{A}}(\lambda)|\\
&\leq (4q_{key} + 2n)\mathsf{Adv}^{k-LIN}_{\mathcal{B}}(\lambda).
\end{aligned}$$

The following lemmas conclude the security analysis. Each lemma represents one step in the sequence of games defined in 7.4.2.

Lemma D.3. *For any PPT adversary $\mathcal{A}$ between Game$_{1,l-1,3}$ and Game$_{1,l,1}$, there exists a PPT algorithm $\mathcal{B}$ for 1 such that for any security parameter λ, we have*

$$\left|Adv^{Game_{1,l-1,3}}_{\mathcal{A}}(\lambda) - Adv^{Game_{1,l,1}}_{\mathcal{A}}(\lambda)\right| \leq Adv^{P_1}_{\mathcal{B}}(\lambda), \text{ for all } l \in [q_{key}].$$

Proof. Suppose there exists a PPT adversary $\mathcal{A}$ between Game$_{1,l-1,3}$ and Game$_{1,l,1}$. Then we can construct a PPT algorithm $\mathcal{B}$ for Problem 1 using $\mathcal{A}$ as a subroutine in the following manner, where $\mathcal{B}$ takes the role of the challenger in the SMN-H security game as described in Definition 5.3.

• $\mathcal{B}$ is given an instance of Problem 1

$$\theta_{\hat{\beta}} = (\texttt{params}_{\mathbb{G}}, g_T, \{\hat{\mathbb{B}}_i, \hat{\mathbb{B}}^*_i\}_{i\in[n]}, \{\phi_{i,\hat{\beta}}\}_{i\in[n]})$$

where all variables are generated as in Problem 1. $\mathcal{B}$ hands $\mathsf{pp} = (\mathtt{params}_{\mathbb{G}}, g_T)$ to $\mathcal{A}$.

- For $t \in [q_{key}]$, in order to answer the t^{th} decryption key query of $\mathcal{A}$ corresponding to a set of vectors and two noise values $(\{\vec{y_{t,i}}\}_{i\in[n]}, \nu_{t,0}, \nu_{t,1})$, $\mathcal{B}$ generates the components of dk_t as follows.
  i) $(t < l)$. $\mathcal{B}$ selects random $r_{t,i,0}, r_{t,i,1} \xleftarrow{\$} \mathbb{F}_q$ for $i \in [n]$ such that $\sum_{i\in[n]} r_{t,i,0} = \nu_{t,0}$ and $\sum_{i\in[n]} r_{t,i,1} = \nu_{t,1}$. Further, sample $\gamma_{t,i,1}, \ldots, \gamma_{t,i,k-1} \xleftarrow{\$} \mathbb{F}_q$ to compute
  $$\mathbf{k}_{t,i} = (\vec{y}_{t,i}, r_{t,i,0}, r_{t,i,1}, \gamma_{t,i,1}, \ldots, \gamma_{t,i,k-1}, \vec{0}^k, 0)_{\mathbb{B}_i^*}.$$
  ii) $(t = l)$. $\mathcal{B}$ samples random $\omega_{l,i} \xleftarrow{\$} \mathbb{F}_q \setminus \{0\}$; $\delta_{l,i}, \kappa_{l,i,1}, \ldots \kappa_{l,i,k-1} \xleftarrow{\$} \mathbb{F}_q$ such that $\sum_{i\in[n]} \omega_{l,i} = 0$ and $\sum_{i\in[n]} \delta_{l,i} = \nu_{l,0}$. Compute
  $$\mathbf{k}_{l,i} = \sum_{j\in[n]} y_{l,i}^j \mathbf{b}_{i,j}^* + \delta_{l,i}\mathbf{b}_{i,m+1}^* + \sum_{j\in[k-1]} \kappa_{l,i,j}\mathbf{b}_{i,m+2+j}^* + \omega_{l,i}\phi_{i,\hat{\beta}}.$$
  iii) $(t > l)$. $\mathcal{B}$ selects random $r_{t,i,0}, \gamma_{t,i,1}, \ldots, \gamma_{t,i,k-1} \xleftarrow{\$} \mathbb{F}_q$ for $i \in [n]$ such that $\sum_i r_{t,i,0} = \nu_{t,0}$ and compute
  $$\mathbf{k}_{t,i} = (\vec{y}_{t,i}, r_{t,i,0}, 0, \gamma_{t,i,1}, \ldots, \gamma_{t,i,k-1}\vec{0}^k, 0)_{\mathbb{B}_i^*}.$$
  $\mathcal{B}$ gives $\mathsf{dk}_t = \{\mathbf{k}_{t,i}\}_{i\in[n]}$ to $\mathcal{A}$.
- For $i \in [n]$, in order to answer the i^{th} ciphertext query of $\mathcal{A}$ corresponding to a pair of vectors $(\vec{x}_{i,0}, \vec{x}_{i,1})$, $\mathcal{B}$ samples random $\varphi_{i,1}, \ldots \varphi_{i,k} \xleftarrow{\$} \mathbb{F}_q$ and computes
  $$\mathbf{c}_i = (\vec{x}_{i,0}, 1, 0, \vec{0}^{k-1}, \varphi_{i,1}, \ldots, \varphi_{i,k}, 0)_{\mathbb{B}_i}.$$
  $\mathcal{B}$ hands $\mathtt{ct}_i = (i, \mathbf{c}_i)$ to $\mathcal{A}$.
- $\mathcal{A}$ outputs a guess bit $\beta' \in \{0, 1\}$. $\mathcal{B}$ outputs $\hat{\beta}' = \beta'$ as its guess bit in its own Problem 1 challenge.

Note that if $\hat{\beta} = 0$, i.e., $\phi_{i,0} = (\vec{0}^m, \alpha_1, 0, \alpha_2, \ldots, \alpha_k, \vec{0}^k, 0)_{\mathbb{B}_i^*}$, then for all $i \in [n]$,
$$\mathbf{k}_{l,i} =(\vec{y}_{l,i}, \alpha_1\omega_{l,i} + \delta_{l,i}, 0, \alpha_2\omega_{l,i} + \kappa_{l,i,1}, \ldots, \alpha_k\omega_{l,i} + \kappa_{l,i,k-1}, \vec{0}^k, 0)_{\mathbb{B}_i^*},$$
which is the same form as in Equation (9) for $r_{l,i,0} = \alpha_1\omega_{l,i} + \delta_{l,i}$ and $\gamma_{l,i,j} = \alpha_{j+1}\omega_{l,i} + \kappa_{l,i,j}$ for $j \in [k-1]$ and this is the proper form of $\mathbf{k}_{l,i}$ in $\text{Game}_{1,l-1,3}$ for all $i \in [n]$. If $\hat{\beta} = 1$, i.e., $\phi_{i,1} = (\vec{0}^m, \alpha_1, 0, \alpha_2, \ldots, \alpha_k, \vec{0}^k, \pi)_{\mathbb{B}_i^*}$, then for all $i \in [n]$,
$$\mathbf{k}_{l,i} =(\vec{y}_{l,i}, \alpha_1\omega_{l,i} + \delta_{l,i}, 0, \alpha_2\omega_{l,i} + \kappa_{l,i,1}, \ldots, \alpha_k\omega_{l,i} + \kappa_{l,i,k-1}, \vec{0}^k, \omega_{i,l}\pi)_{\mathbb{B}_i^*},$$
which is the same form as in Equation (10) for $r_{l,i,0} = \alpha_1\omega_{l,i} + \delta_{l,i}$, $\gamma_{l,i,j} = \alpha_{j+1}\omega_{l,i} + \kappa_{l,i,j}$ for $j \in [k-1]$ and $\rho_{l,i} = \omega_{l,i}\pi$ and this is the proper form of $\mathbf{k}_{l,i}$ in $\text{Game}_{1,l,1}$ for all $i \in [n]$. In particular, $\sum_{i\in[n]} r_{l,i,0} = \nu_{l,0}$ and $\sum_{i\in[n]} \rho_{l,i} = 0$. For $t < l$, $\mathbf{k}_{t,i}$ has the form as in Equation (12) for all $i \in [n]$, whereas for $t > l$, $\mathbf{k}_{t,i}$ has the form as in Equation (10) for all $i \in [n]$. But these are the proper forms of $\mathbf{k}_{t,i}$ in the respective cases in both $\text{Game}_{1,l-1,3}$ and $\text{Game}_{1,l,1}$. Moreover, the answer to the ciphertext query in index i is given as in Equation (8), which is their proper form in both $\text{Game}_{1,l-1,3}$ and $\text{Game}_{1,l,1}$. Therefore, the view of the adversary $\mathcal{A}$ simulated by $\mathcal{B}$ is distributed as in $\text{Game}_{1,l-1,3}$ and $\text{Game}_{1,l,1}$, respectively, according as $\hat{\beta} = 0$ or 1. □

Lemma D.4. *For any probabilistic adversary $\mathcal{A}$, for any security parameter λ, we have*
$$Adv_{\mathcal{A}}^{Game_{1,l,1}}(\lambda) = Adv_{\mathcal{A}}^{Game_{1,l,2}}(\lambda), \text{ for all } l \in [q_{key}].$$

Proof. In order to prove this lemma, we show that the view of the adversary $\mathcal{A}$ in $\text{Game}_{1,l,1}$ and that in $\text{Game}_{1,l,2}$ are identically distributed. This is done by defining new sets of dual orthogonal bases $\{\mathbb{U}_i = \{\mathbf{u}_{i,1}, \ldots, \mathbf{u}_{i,m+2k+2}\}, \mathbb{U}_i^* = \{\mathbf{u}_{i,1}^*, \ldots, \mathbf{u}_{i,m+2k+2}^*\}\}_{i\in[n]}$ of the pair of vector spaces $(\mathbb{V}_1, \mathbb{V}_2)$ using the sets of dual orthogonal bases $\{\mathbb{B}_i, \mathbb{B}_i^*\}_{i\in[n]}$ generated from $\mathcal{G}_{OB}(m+2k+2, \mathtt{params}_{\mathbb{G}}, o)$ in $\text{Game}_{1,l,1}$ in the following manner:
$$\mathbf{u}_{i,m+2k+2}^* = \mathbf{b}_{i,m+2k+2}^* - \frac{r_{l,i,1}}{\rho_{l,i}}\mathbf{b}_{i,m+2}^*,\ i \in [n]$$
$$\mathbf{u}_{i,j}^* = \mathbf{b}_{i,j}^*,\ i \in [n], j \in [m + 2k + 1]$$
$$\mathbf{u}_{i,m+2} = \mathbf{b}_{i,m+2} + \frac{r_{l,i,1}}{\rho_{l,i}}\mathbf{b}_{i,m+2k+2},\ i \in [n]$$
$$\mathbf{u}_{i,j} = \mathbf{b}_{i,j},\ i \in [n], j \in [m + 2k + 2]\backslash\{m + 2\}.$$

Indeed, $\{\mathbb{U}_i, \mathbb{U}_i^*\}_{i\in[n]}$ is a set of dual orthogonal bases as they are obtained from the set of orthogonal bases $\{\mathbb{B}_i, \mathbb{B}_i^*\}$ by applying invertible linear transformations. Moreover, $\{\mathbb{U}_i, \mathbb{U}_i^*\}_{i\in[n]}$ are distributed uniformly at random due to $\{\mathbb{B}_i, \mathbb{B}_i^*\}_{i\in[n]}$ being so.

Now, observe that the components of the l^{th} answered decryption key $\mathsf{dk}_l = \{\mathbf{k}_{l,i}\}_{i\in[n]}$ corresponding to a set of vectors and two noise values $(\{\vec{y_{l,i}}\}_{i\in[n]}, \nu_{t,0}, \nu_{t,1})$ with $\vec{y_{l,i}} \in \mathbb{F}_q^m, \nu_{t,0}, \nu_{t,1} \in \mathbb{F}_q$ for all $i \in [n]$, in $\text{Game}_{1,l,1}$ can be displayed as
$$\begin{aligned}\mathbf{k}_{l,i} &= (\vec{y}_{l,i}, r_{l,i,0}, 0, \gamma_{l,i,1}, \ldots, \gamma_{l,i,k-1}, \vec{0}^k, \rho_{l,i})_{\mathbb{B}_i^*} \\ &= (\vec{y}_{l,i}, r_{l,i,0}, r_{l,i,1}, \gamma_{l,i,1}, \ldots, \gamma_{l,i,k-1}, \vec{0}^k, \rho_{l,i})_{\mathbb{U}_i^*}.\end{aligned} \tag{24}$$

Obviously, the form of $\mathbf{k}_{l,i}$ in the equation above is identical to that in Equation (11) corresponding to $\text{Game}_{1,l,2}$ for all $i \in [n]$. Therefore, the form of the l^{th} answered decryption key is changed from that in $\text{Game}_{1,l,1}$ to that in $\text{Game}_{1,l,2}$ through the basis transformations. Further, for all $t \neq l$, the components of the t^{th} answered decryption key corresponding to $(\{\vec{y_{t,i}}\}_{i\in[n]}, \nu_{t,0}, \nu_{t,1})$ in $\text{Game}_{1,l,1}$ can be expressed as follows for all $i \in [n]$:

i) $(t < l)$:
$$\begin{aligned}\mathbf{k}_{t,i} &= (\vec{y}_{t,i}, r_{t,i,0}, r_{t,i,1}, \gamma_{t,i,1}, \ldots, \gamma_{t,i,k-1}, \vec{0}^k, 0)_{\mathbb{B}_i^*} \\ &= (\vec{y}_{t,i}, r_{t,i,0}, r_{t,i,1}, \gamma_{t,i,1}, \ldots, \gamma_{t,i,k-1}, \vec{0}^k, 0)_{\mathbb{U}_i^*}.\end{aligned}$$

ii) $(t > l)$:
$$\begin{aligned}\mathbf{k}_{t,i} &= (\vec{y}_{t,i}, r_{t,i,0}, 0, \gamma_{t,i,1}, \ldots, \gamma_{t,i,k-1}, \vec{0}^k, 0)_{\mathbb{B}_i^*} \\ &= (\vec{y}_{t,i}, r_{t,i,0}, 0, \gamma_{t,i,1}, \ldots, \gamma_{t,i,k-1}, \vec{0}^k, 0)_{\mathbb{U}_i^*}.\end{aligned}$$

Thus, for all $t \neq l$, the forms of the components of dk_t are preserved under the basis transformations.

Further, for all $i \in [n]$, the answer to the ciphertext query in index i, $\mathtt{ct}_i = (i, \mathbf{c}_i)$ corresponding to a pair of vectors $(\vec{x}_{i,0}, \vec{x}_{i,1})$

in $\text{Game}_{1,l,1}$ can be expressed in the following manner:

$$\begin{aligned}\mathbf{c}_i &= (\vec{x}_{i,0}, 1, 0, \vec{0}^{k-1}, \varphi_{i,1}, \dots, \varphi_{i,k}, 0)_{\mathbb{B}_i}.\\ &= (\vec{x}_{i,0}, 1, 0, \vec{0}^{k-1}, \varphi_{i,1}, \dots, \varphi_{i,k}, 0)_{\mathbb{U}_i},\end{aligned}$$

i.e. the form of $\mathtt{ct}_i = (i, \mathbf{c}_i)$ is preserved under basis transformations for all $i \in [n]$.

In addition, note that $e(\mathbf{u}_{i,j}, \mathbf{u}^*_{i,j}) = e(\mathbf{b}_{i,j}, \mathbf{b}^*_{i,j}) = g_T$ for all $i \in [n], j \in [m+2k+2]$ and thus the basis transformations are compatible with the public parameters $\mathsf{pp} = (\mathtt{params}_{\mathbb{G}}, g_T)$ in $\text{Game}_{1,l,1}$ as well. Therefore, the view of $\mathcal{A}$ in $\text{Game}_{1,l,1}$ can be conceptually changed to that in $\text{Game}_{1,l,2}$. □

Lemma D.5. *For any PPT adversary $\mathcal{A}$ between $Game_{1,l,2}$ and $Game_{1,l,3}$, there exists a PPT algorithm $\mathcal{B}$ for Problem 1 such that for any security parameter λ, we have*

$$\left|Adv_{\mathcal{A}}^{Game_{1,l,2}}(\lambda) - Adv_{\mathcal{A}}^{Game_{1,l,3}}(\lambda)\right| \leq Adv_{\mathcal{B}}^{P_1}(\lambda),\ \textit{for all } l \in [q_{key}].$$

Proof. The proof of this lemma is essentially the same as the proof of Lemma D.3. Note that the only difference on the view of an adversary trying to distinguish $\text{Game}_{1,l,1}$ and $\text{Game}_{1,l-1,3}$ and that of one trying to differentiate between $\text{Game}_{1,l,2}$ and $\text{Game}_{1,l,3}$ for some $l \in [q_{key}]$ is a possible non-zero value in the $m + 2^{th}$ slot of the decryption key query. More precise, having the exact same setting as in the proof of Lemma D.3, $\mathcal{B}$ answers the t^{th} decryption key query for $t = l$ in the following manner.

$\mathcal{B}$ samples $\omega_{l,i} \xleftarrow{\$} \mathbb{F}_q \setminus \{0\}$; $\delta_{l,i}, r_{l,i,1}, \kappa_{l,i,1}, \dots, \kappa_{l,i,k-1} \xleftarrow{\$} \mathbb{F}_q$ such that $\sum_{i\in[n]} \omega_{l,i} = 0, \sum_{i\in[n]} r_{l,i,1} = \nu_{l,1}$ and $\sum_{i\in[n]} \delta_{l,i} = \nu_{l,0}$. Compute

$$\begin{aligned}\mathbf{k}_{l,i} = &\sum_{j\in[n]} y^j_{l,i}\mathbf{b}^*_{i,j} + \delta_{l,i}\mathbf{b}^*_{i,m+1} + r_{l,i,1}\mathbf{b}^*_{i,m+2}\\ &+ \sum_{j\in[k-1]} \kappa_{l,i,j}\mathbf{b}^*_{i,m+2+j} + \omega_{l,i}\phi_{i,\hat{\beta}}.\end{aligned}$$

and give $\mathsf{dk}_t = \{\mathbf{k}_{t,i}\}_{i\in[n]}$ to $\mathcal{A}$.

By the same arguments, all of the requirements hold and we have the same form as in Equation (11) or Equation (12), depending on whether $\hat{\beta}$ is 1 or 0. □

Lemma D.6. *For any PPT adversary $\mathcal{A}$ between $Game_{2,i-1,3}$ and $Game_{2,i,1}$, there exists a PPT algorithm $\mathcal{B}$ for Problem 2 such that for any security parameter λ, we have*

$$\left|Adv_{\mathcal{A}}^{Game_{2,i-1,3}}(\lambda) - Adv_{\mathcal{A}}^{Game_{2,i,1}}(\lambda)\right| \leq Adv_{\mathcal{B}}^{P_2}(\lambda),\ \textit{for all } i \in [n].$$

Proof. Suppose there exists a PPT adversary $\mathcal{A}$ between $\text{Game}_{2,i-1,3}$ and $\text{Game}_{2,i,1}$. Then we can construct a PPT algorithm $\mathcal{B}$ for Problem 2 using $\mathcal{A}$ as a subroutine in the following manner, where $\mathcal{B}$ takes the role of the challenger in the SMN-H security game as described in Definition 5.3.

- $\mathcal{B}$ is given an instance of Problem 2

  $$\theta_{\hat{\beta}} = (\mathtt{params}_{\mathbb{G}}, g_T, \{\hat{\mathbb{B}}_i, \hat{\mathbb{B}}^*_i\}_{i\in[n]}, \{\phi_{i,\hat{\beta}}\}_{i\in[n]})$$

  where all variables are generated as in Problem 2. $\mathcal{B}$ hands $\mathsf{pp} = (\mathtt{params}_{\mathbb{G}}, g_T)$ to $\mathcal{A}$.
- For $l \in [q_{key}]$, in order to answer the l^{th} decryption key query of $\mathcal{A}$ corresponding to a set of vectors and two noise values $(\{\vec{y}_{l,i}\}_{i\in[n]}, \nu_{l,0}, \nu_{l,1})$, $\mathcal{B}$ selects random $r_{l,i,0}, r_{l,i,1} \xleftarrow{\$} \mathbb{F}_q$ for $i \in [n]$ such that $\sum_{i\in[n]} r_{l,i,0} = \nu_{l,0}$ and $\sum_{i\in[n]} r_{l,i,1} = \nu_{l,1}$. Further, sample $\gamma_{t,i,1}, \dots, \gamma_{t,i,k-1} \xleftarrow{\$} \mathbb{F}_q$ to compute

  $$\mathbf{k}_{l,i} = (\vec{y}_{l,i}, r_{l,i,0}, r_{l,i,1}, \gamma_{l,i,1}, \dots, \gamma_{l,i,k-1}, \vec{0}^k, 0)_{\mathbb{B}^*_i}.$$

  $\mathcal{B}$ gives $\mathsf{dk}_l = \{\mathbf{k}_{l,i}\}$ to $\mathcal{A}$.
- For $t \in [n]$, in order to answer the ciphertext query of $\mathcal{A}$ in index t corresponding to a pair of vectors $(\vec{x}_{t,0}, \vec{x}_{t,1})$, $\mathcal{B}$ computes $\mathbf{c}_t$ as follows:
  - i) $(t < i)$ $\mathcal{B}$ samples random $\varphi_{t,1}, \dots \varphi_{t,k} \xleftarrow{\$} \mathbb{F}_q$ and calculates $\mathbf{c}_t$ as
    $$\mathbf{c}_t = (\vec{x}_{t,1}, 0, 1, \vec{0}^{k-1}, \varphi_{t,1}, \dots, \varphi_{t,k}, 0)_{\mathbb{B}_t}.$$
  - ii) $(t = i)$ $\mathcal{B}$ calculates $\mathbf{c}_i$ as
    $$\mathbf{c}_i = \sum_{j\in[m]} x^j_{i,0}\mathbf{b}_{i,j} + \mathbf{b}_{i,m+1} + \phi_{i,\hat{\beta}}$$
  - iii) $(t > i)$ $\mathcal{B}$ samples random $\varphi_{t,1}, \dots \varphi_{t,k} \xleftarrow{\$} \mathbb{F}_q$ and calculates $\mathbf{c}_t$ as
    $$\mathbf{c}_t = (\vec{x}_{t,0}, 1, 0, \vec{0}^{k-1}, \varphi_{t,1}, \dots, \varphi_{t,k}, 0)_{\mathbb{B}_t}.$$

  $\mathcal{B}$ hands $\mathtt{ct}_t = (t, \mathbf{c}_t)$ to $\mathcal{A}$.
- $\mathcal{A}$ outputs a guess bit $\beta' \in \{0, 1\}$. $\mathcal{B}$ outputs $\hat{\beta}' = \beta'$ as its guess bit in its own Problem 1 challenge.

Note that if $\hat{\beta} = 0$, i.e., $\phi_{i,0} = (\vec{0}^{m+k+1}, \alpha_1, \alpha_2, \dots, \alpha_k, 0)_{\mathbb{B}_i}$, then we have

$$\mathbf{c}_i = (\vec{x}_{i,0}, 1, 0, \vec{0}^{k-1}, \alpha_1, \dots, \alpha_k, 0)_{\mathbb{B}_i}$$

which is the same form as in Equation (8) for $\varphi_{i,j} = \alpha_j$, $j \in [k]$. If $\hat{\beta} = 1$, i.e., $\phi_{i,0} = (\vec{0}^{m+k+1}, \alpha_1, \alpha_2, \dots, \alpha_k, \pi)_{\mathbb{B}_i}$, then,

$$\mathbf{c}_i = (\vec{x}_{i,0}, 1, 0, \vec{0}^{k-1}, \alpha_1, \dots, \alpha_k, \pi)_{\mathbb{B}_i}$$

which is the same form as in Equation (13) for $\varphi_{i,j} = \alpha_j$, $j \in [k]$ and $\omega_i = \pi$ and this is the proper form of $\mathbf{c}_i$ in $\text{Game}_{2,i,1}$. For $t < i$, $\mathbf{c}_t$ has the form as in Equation (15) for all $t \in [n]$, whereas for $t > i$, $\mathbf{c}_t$ has the form as in Equation (13) for all $t \in [n]$. But these are the proper forms of $\mathbf{c}_t$ in the respective cases in both $\text{Game}_{2,i-1,3}$ and $\text{Game}_{2,i,1}$. Moreover, the answer to the l^{th} decryption key query is given as in Equation (12), which is their proper form in both $\text{Game}_{2,i-1,3}$ and $\text{Game}_{2,i,1}$. Therefore, the view of the adversary $\mathcal{A}$ simulated by $\mathcal{B}$ is distributed as in $\text{Game}_{2,i-1,3}$ or $\text{Game}_{2,i,1}$, according as $\hat{\beta} = 0$ or 1. □

Lemma D.7. *For any probabilistic adversary $\mathcal{A}$, for any security parameter λ, we have*

$$Adv_{\mathcal{A}}^{Game_{2,i,1}}(\lambda) = Adv_{\mathcal{A}}^{Game_{2,i,2}}(\lambda),\ \textit{for all } i \in [n].$$

Proof. In order to prove this lemma, we show that the view of the adversary $\mathcal{A}$ in $\text{Game}_{2,i,1}$ and that in $\text{Game}_{2,i,2}$ are identically distributed. This is done by defining new sets of dual orthogonal bases $\{\mathbb{U}_t = \{\mathbf{u}_{t,1}, \dots, \mathbf{u}_{t,m+2k+2}\}, \mathbb{U}^*_t = \{\mathbf{u}^*_{t,1}, \dots, \mathbf{u}^*_{t,m+2k+2}\}\}_{t\in[n]}$ of the pair of vector spaces $(\mathbb{V}_1, \mathbb{V}_2)$ using the sets of dual orthogonal bases $\{\mathbb{B}_t, \mathbb{B}^*_t\}_{t\in[n]}$ generated from $\mathcal{G}_{OB}(m+2k+2, \mathtt{params}_{\mathbb{G}}, o)$ in $\text{Game}_{2,i,1}$ in the following manner:

$$
\begin{aligned}
\mathbf{u}_{t,m+2k+2} &= \mathbf{b}_{t,m+2k+2} + \frac{1}{\omega_t}(\mathbf{b}_{t,m+1} - \mathbf{b}_{t,m+2} \\
&\quad + \sum_{j\in[m]} (x^j_{t,0} - x^j_{t,1})\mathbf{b}_{t,j}), t \in [n] \\
\mathbf{u}_{t,j} &= \mathbf{b}_{t,j}, t \in [n], j \in [m+2k+1] \\
\mathbf{u}^*_{t,j} &= \mathbf{b}^*_{t,j} + \frac{1}{\omega_t}(x^j_{t,1} - x^j_{t,0})\mathbf{b}^*_{t,m+2k+2}, t \in [n], j \in [m] \\
\mathbf{u}^*_{t,m+1} &= \mathbf{b}^*_{t,m+1} - \frac{1}{\omega_t}\mathbf{b}^*_{t,m+2k+2}, t \in [n] \\
\mathbf{u}^*_{t,m+2} &= \mathbf{b}^*_{t,m+2} + \frac{1}{\omega_t}\mathbf{b}^*_{t,m+2k+2}, t \in [n] \\
\mathbf{u}^*_{t,j} &= \mathbf{b}^*_{t,j}, t \in [n], j \in [m+3, m+2k+2]
\end{aligned}
$$

Indeed, $\{\mathbb{U}_t, \mathbb{U}^*_t\}_{t\in[n]}$ is a set of dual orthogonal bases as they are obtained from the set of orthogonal bases $\{\mathbb{B}_t, \mathbb{B}^*_t\}$ by applying invertible linear transformations. Moreover, $\{\mathbb{U}_t, \mathbb{U}^*_t\}_{t\in[n]}$ are distributed uniformly at random due to $\{\mathbb{B}_t, \mathbb{B}^*_t\}_{t\in[n]}$ being so.

Observe that the components of the ciphertext query in the i^{th} index corresponding to a set of two vectors $(\vec{x}_{i,0}, \vec{x}_{i,1})$ in Game$_{2,i,1}$ can be displayed as

$$
\begin{aligned}
\mathbf{c}_i &= (\vec{x}_{i,0}, 1, 0, \vec{0}^{k-1}, \varphi_{i,1}, \dots, \varphi_{i,k}, \omega_i)_{\mathbb{B}_i} \\
&= (\vec{x}_{i,1}, 0, 1, \vec{0}^{k-1}, \varphi_{i,1}, \dots, \varphi_{i,k}, \omega_i)_{\mathbb{U}_i}.
\end{aligned}
$$

Clearly, the form of $\mathbf{c}_i$ in the equation above is identical to that in Equation (14) corresponding to Game$_{2,i,2}$. Thus, the form of this ciphertext request is changed from that in Game$_{2,i,1}$ to Game$_{2,i,2}$ through the basis transformations. Further, for all $t \neq i$ corresponding to the ciphertext query in the t^{th} index corresponding to $(\vec{x}_{t,0}, \vec{x}_{t,1})$ in Game$_{2,i,1}$ can be expressed in the following manner:

i) $(t < i)$:

$$
\begin{aligned}
\mathbf{c}_t &= (\vec{x}_{t,1}, 0, 1, \vec{0}^{k-1}, \varphi_{t,1}, \dots, \varphi_{t,k}, 0)_{\mathbb{B}_t} \\
&= (\vec{x}_{t,1}, 0, 1, \vec{0}^{k-1}, \varphi_{t,1}, \dots, \varphi_{t,k}, 0)_{\mathbb{U}_t}
\end{aligned}
$$

ii) $(t > i)$:

$$
\begin{aligned}
\mathbf{c}_t &= (\vec{x}_{t,0}, 1, 0, \vec{0}^{k-1}, \varphi_{t,1}, \dots, \varphi_{t,k}, 0)_{\mathbb{B}_t} \\
&= (\vec{x}_{t,0}, 1, 0, \vec{0}^{k-1}, \varphi_{t,1}, \dots, \varphi_{t,k}, 0)_{\mathbb{U}_t}
\end{aligned}
$$

Hence, for all $t \neq i$, the forms of the components of $\mathtt{ct}_t$ are preserved under the basis transformations.

Further, for all decryption key queries $l \in [q_{key}]$ corresponding to a set of vectors and two noise values $(\{\vec{y_{l,t}}\}_{t\in[n]}, \nu_{l,0}, \nu_{l,1})$, it holds that for all $t \in [n]$ that

$$
\begin{aligned}
\mathbf{k}_{l,t} &= (\vec{y}_{l,t}, r_{l,t,0}, r_{l,t,1}, \gamma_{l,t,1}, \dots, \gamma_{lt,k-1}, \vec{0}^k, 0)_{\mathbb{B}^*_t} \\
&= (\vec{y}_{l,t}, r_{l,t,0}, r_{l,t,1}, \gamma_{l,t,1}, \dots, \gamma_{lt,k-1}, \vec{0}^k, 0)_{\mathbb{U}^*_t},
\end{aligned}
$$

i.e., the form of dk_l is preserved under basis transformations for all $t \in [n]$.

In addition, note that $e(\mathbf{u}_{t,j}, \mathbf{u}^*_{t,j}) = e(\mathbf{b}_{t,j}, \mathbf{b}^*_{t,j}) = g_T$ for all $t \in [n], j \in [m+2k+2]$ and thus the basis transformations are compatible with the public parameters $\mathsf{pp} = (\mathtt{params}_{\mathbb{G}}, g_T)$ in Game$_{2,i,1}$ as well. Therefore, the view of $\mathcal{A}$ in Game$_{2,i,1}$ can be conceptually changed to that in Game$_{2,i,2}$. □

Lemma D.8. *For any PPT adversary $\mathcal{A}$ between Game$_{2,i,2}$ and Game$_{2,i,3}$, there exists a PPT algorithm $\mathcal{B}$ for 2 such that for any security parameter λ, we have*

$$\left| Adv^{Game_{2,i,2}}_{\mathcal{A}}(\lambda) - Adv^{Game_{2,i,3}}_{\mathcal{A}}(\lambda) \right| \leq Adv^{P_2}_{\mathcal{B}}(\lambda), \ \textit{for all } i \in [n].$$

Proof. The proof of this lemma is essentially the same as the proof of Lemma D.6. Note that the only difference on the view of an adversary trying to distinguish Game$_{2,i,1}$ and Game$_{2,i-1,3}$ and one trying to differentiate between Game$_{2,i,2}$ and Game$_{2,i,3}$ is that the $m+1^{th}$ and $m+2^{th}$ slot in the i^{th} ciphertext query are interchanged and $\vec{x}_{i,0}$ is replaced by $\vec{x}_{i,1}$. In detail, having the exact same setting as in the proof of Lemma D.6, $\mathcal{B}$ answers the i^{th} ciphertext query in the following way.

$\mathcal{B}$ calculates

$$\mathbf{c}_i = \sum_{j\in[m]} x^j_{i,1}\mathbf{b}_{i,j} + \mathbf{b}_{i,m+2} + \phi_{i,\hat{\beta}}$$

and hands $\mathtt{ct}_i = (i, \mathbf{c}_i)$ to $\mathcal{A}$.

by the same arguments as before, all of the requirements hold and we have the same form as in Equation (14) or (15), depending on whether $\hat{\beta}$ is 1 or 0. □

Lemma D.9. *For any probabilistic adversary $\mathcal{A}$, for any security parameter λ, we have*

$$Adv^{Game_{2,n,3}}_{\mathcal{A}}(\lambda) = Adv^{Game_3}_{\mathcal{A}}(\lambda).$$

Proof. Proving this lemma is straightforward as it simply requires a base transformation where the base vectors of $\mathbb{B}_i$ and B^*_i in position $m+1$ and $m+2$ are interchanged. Clearly, this transformation maintains all the crucial properties of the bases and changes the view of the adversary accordingly. □