A New Construction Method for More Efficient Quadratic One-Time Noisy Multi-Client Functional Encryption Schemes[1]

Jasmin Zalonis, Linda Scheu-Hachtel, Frederik Armknecht

University of Mannheim
Mannheim, Germany

Abstract We introduce a new construction method for one-time multi-client functional encryption schemes that support noisy quadratic functions, are resistant against corruption and allow for labels. Such schemes can be used as building blocks in many practical applications, e.g., privacy preserving machine learning on arbitrarily split data. In contrast to earlier constructions, ours uses a different structural design that allows to make use of less complex, hence more efficient building blocks. The security of our construction relies solely on its underlying building blocks and no additional hardness assumptions, making it more generic than related work. More specifically, the construction itself does not rely on structures given by bilinear groups.

We present a concrete instantiation, dubbed QUILT, and show in a series of experiments that it outperforms existing comparable schemes by far. For example, in the case of private logistic regression training, using QUILT yields a speed-up of 4.8x to 6.8x.

Moreover, in contrast to these schemes, our construction allows for the use of labels. This weakens the one-time restriction, since multiple encryptions are possible, if each ciphertext is tied to a different label.

Keywords: Quadratic Functional Encryption, Noisy Functional Encryption, Multi-Client Functional Encryption, Privacy Preserving Machine Learning

[1] This is an updated version of the article [ZSHA26] published in the proceedings of the 21th ACM Asia Conference on Computer and Communications Security.

1 Introduction

1.1 State of the Art

Functional encryption (FE) [BSW11] is a cryptographic primitive that overcomes the all-or-nothing principle of standard encryption. It allows to create decryption keys dk for supported functions f, such that applying the decryption algorithm with dk to some encryption of data $\mathbf{x}$ yields $f(\mathbf{x})$ instead of $\mathbf{x}$. In particular, FE encompasses attribute-based encryption [BSW07; GPSW06; GVW15], broadcast encryption [BH08; BSS05; FN94] and many more, while also having implications for powerful primitives such as indistinguishability obfuscation (iO) [AJ15; BV18]. As no information about $\mathbf{x}$ is revealed besides the result of the function evaluation, FE can be used to give untrusted parties controlled access to perform evaluations on encrypted data.

[GGGJ+14] were the first to generalize FE to the multi-input setting. Multi-input functional encryption (MIFE) schemes support functions f that perform aggregated computations over multiple, separately encrypted inputs. Although this initiated a lot of research focusing on achieving MIFE for arbitrary functions [BCP14; GGHR+16; JLS21; Wat15], these schemes are still far from being applicable in practice. As a result, another line of research has emerged, focusing on developing efficient schemes for restricted functionalities, such as inner-product (IP) or quadratic functions [ABG19; ABKW19; ACFG+18; AGRW17; AGT21; AGT22; CDGP+18; DOT18; LT19].

Even with restricted functionality these works enabled a discussion about the utility of FE in the context of privacy preserving machine learning (PPML). Recent works [CZGQ23; SHZ25; ZASH24; ZSHA25] highlight the unique beneficial properties of FE for the PPML use case, e. g., in the medical field. Consider a scenario where several data holders, e. g., patients hold individual data and are willing to share it as long as their privacy remains protected. However, they do not want to be involved in any heavy computation or ongoing analysis protocols. On the other side, there is an analyst with substantial computational resources. Additionally there is at least one trusted party with limited resources to enforce privacy regulations, e. g., an ethics-committee. Using FE in such a scenario allows the data holder to send their encrypted data directly to the analyst, who then only interacts with the authority. The computation is performed directly by the analyst, who is neither dependent on the immediate availability of multiple computing parties nor needs to perform a high-communication protocol,

in contrast to solutions based on multi-party computation. Moreover, the trusted party does not require heavy resources, contrary to solutions based on verifiable homomorphic encryption. While the main limitation of FE in PPML remains its limited functionality, several works demonstrate how FE can still be effectively applied in this setting [CMAK23; CZGQ23; PT23; PTC23; QLHY+22].

To realize more complex algorithms f, those are decomposed in two parts $f = f_2 \circ f_1$, where f_1 is a linear or quadratic function, which is evaluated using an MIFE scheme, and the output is post-processed in clear through f_2. Unfortunately, this approach allows the analyst to observe the intermediate results of f_1 in clear, which can lead to privacy breaches of the underlying data [CFLS20].

To avoid this, two works independently developed the concept of noisy MIFE (NMIFE) [ZASH24][2] respectively randomized FE [ECLP24]. In this setting, the decryption key realizes a noisy function evaluation $f(\mathbf{x}_1, \ldots, \mathbf{x}_n) + \nu$ such that ν is not leaked from the decryption key. This can be used to combine (MI)FE with differential privacy (DP) to mitigate privacy loss and perform statistical analysis on encrypted data. However, the schemes proposed by [ZASH24] and [ECLP24] are limited to IP functionality. In addition, [ECLP24] is further restricted to a single-input. As shown in [SHZ25], using NMIFE schemes for IP already enables the training of a private logistic regression. They interpret the training algorithm, namely the gradient-descent algorithm, as a sequence of linear functions applied to the input vectors, that are iteratively evaluated using multiple decryption keys. Since each evaluation is protected by some noise, the overall protocol remains DP. However, using an NMIFE scheme for IP only allows to train ML models for horizontal split data, i. e., if each whole data record is in possession of one client.

Agrawal et al. [AGT21] were the first to propose a quadratic non-noisy MIFE scheme. However, all of the clients share the same encryption key, which is impractical for applications such as PPML, where it is realistic to assume that an adversary may corrupt at least one data holder. In [AGT22], Agrawal et al. improve their work and propose a new MIFE scheme, dubbed AGT in the following, supporting individual encryption keys. Still, AGT does not support noise and even for a small number of clients and short plaintext size the setup-algorithm exceeds 10h. Later, Zalonis et al. [ZSHA25] extended AGT to the setting of NMIFE and propose

[2] This notion is not to be confused with noisy FE by Agrawal [Agr19], which does not allow for arbitrary distributions.

a more efficient variant, dubbed ZSHA, which they use to train a logistic regression over vertical split data, i. e., one record is distributed between several clients. The caveat of their scheme is that it is restricted to one-time (OT) security, meaning that only one ciphertext per client can be securely generated.

This should not be confused with 1- or q-FE schemes like [AV19; GVW12; SS10], which support an unlimited number of ciphertexts but only a predefined, limited number of decryption keys. Although they allow more complex functions, they do not support multiple clients. Moreover, it is unclear how to obtain noisy outputs, making them unsuitable for the considered use case.

In contrast, allowing one ciphertext per client may sound like a severe limitation, however related works in NMIFE [ECLP24; ZASH24; ZSHA25] observe that many applications, particularly in the context of PPML, inherently operate on fixed databases. Especially when combining PPML with DP, allowing only a single ciphertext per slot i is not only sufficient but even mandatory. Even though ZSHA is more efficient than AGT, the training time of a logistic regression on split datasets extends to several hours, even when applied to relatively small datasets, e.g. with 189 records and 8 attributes.

1.2 Contribution

In this work, we introduce a new construction method for quadratic one-time noisy multi-client functional encryption (NMCFE) schemes and apply it to train a private logistic regression model. NMCFE is a generalization of NMIFE in the sense that each ciphertext and decryption key is additionally equipped with a label and only those associated with the same label can be successfully combined. Moreover, our scheme allows for corruptions, i. e., the adversary obtains encryption keys for certain corrupted slots.

In contrast to prior schemes, the structural simplicity of our construction ensures that its security relies solely on its underlying building blocks and thus on their hardness assumption. Although some building blocks can currently only be realized from pairings, we never rely directly on their hardness, unlike related work. Consequently, new instantiations of our building blocks based on alternative hardness assumptions can be directly integrated into our construction without any modification of the security proof.

We describe and benchmark a concrete instantiation, dubbed QUILT (QUadratic multI-cLienT functional encryption). As can be seen in Table 1,

Table 1: Overview of existing quadratic NMIFE schemes.

Scheme	Noisy	Corruption	Many Encryptions	Labels
AGT [AGT22]	(√)	√	√	×
ZSHA [ZSHA25]	√	√	×	×
QUILT	√	√	×	√

QUILT is comparable with the previously existing schemes AGT and ZSHA in terms of functionality and security.

More precisely, all three schemes are resistant against static corruptions (Definition 3) and support noisy functions. Although, AGT in its original definition is not noisy, [ZSHA25] showed that it can be made noisy by a slight extension. Additionally, they improve AGT in the setup algorithm introducing only minimal additional effort in the encryption algorithm. We refer to this improved and noisy version in the following whenever we use the name AGT.

The only restriction is that QUILT, just like ZSHA, allows only one ciphertext per slot. However, as previously mentioned, this is not an issue in many practical applications, such as PPML, where data is often collected up to a fixed point in time and only then processed further, especially when DP is used. Moreover, in contrast to AGT and ZSHA, QUILT supports labels. That is, both the encryption process and the key generation process take labels as additional inputs. Decryption keys can only be used for decrypting ciphertexts if the labels for all components match, enabling a more fine-grained access. Hence, multiple encryptions are possible if each ciphertext is tied to a different label.

Besides the support of labels, a major benefit of the new construction is that it is by magnitudes faster. Table 2 shows the runtime of our schemes for exemplary datasets. As one can see, our construction is by far the fastest quadratic NMIFE scheme at the current point in time. This will be further confirmed in Section 6, where we provide detailed benchmarks on synthetic data. Additionally, we train a private logistic regression model on real-world medical data, achieving a 4.8x to 6.8x speed-up compared to ZSHA.

1.3 Outline

The paper is structured as follows. Section 2 provides an overview of our construction, followed by the preliminaries in Section 3. In Section 4, the construction is explained and its correctness shown. Afterwards, we prove its security in Section 5 and investigate its performance in Section 6.

2 Technical Overview

Before we explain our new quadratic NMCFE scheme in detail in Section 4, we provide a high-level overview below. In particular, we explain how our scheme differs from previous constructions and why these changes allow for an enormous speed-up.

For the sake of readability, we focus in this chapter on the realization of quadratic functions in which all monomials are of degree 2, i. e., functions of the form

$$Q(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \sum_{i,k \in [n], j,l \in [m]} \boldsymbol{c}[(i,j,k,l)]\mathbf{x}_i[j]\mathbf{x}_k[l]. \tag{1}$$

As explained in Remark 3, this does not pose any restriction.

To further ease the presentation, we introduce some additional notation for this chapter. First, we combine pairs of indexes, e. g., (i, j), into a single identifier, e. g., I. That is, we use expressions $I = (i, j), K = (k, l) \in [n] \times [m]$ to abbreviate, for example, $\mathbf{x}_I = \mathbf{x}_i[j]$, $\mathbf{x}_K = \mathbf{x}_k[l]$, and $\boldsymbol{c}_{I,K} = \boldsymbol{c}[(i, j, k, l)]$.

Second, we are going to use a more compact representation for ciphertexts and decryption keys. Given some single-input FE scheme S and input $\mathbf{x}$, we denote by $[\mathbf{x}]_{S,\mathsf{ct}}$ an encryption of $\mathbf{x}$ under S, and likewise by $[f]_{S,\mathsf{dk}}$ a decryption key for some function f under S. We typically associate functions f by their coefficient vector $\boldsymbol{c}$ and write $[\boldsymbol{c}]_{S,\mathsf{dk}}$ instead of $[f]_{S,\mathsf{dk}}$. This means that in our shortened notation, for a single-input IP scheme it holds that

$$\mathsf{Dec}([\boldsymbol{c}]_{S,\mathsf{dk}}, [\mathbf{x}]_{S,\mathsf{ct}}) = \langle \boldsymbol{c}, \mathbf{x} \rangle. \tag{2}$$

Recall that for an MIFE scheme S, ciphertexts are associated with distinct encryption slots. In such cases, we denote by $[\mathbf{x}]_{S,\mathsf{ct},i}$ an encryption of $\mathbf{x}$ for slot i.

Table 2: Runtime comparison of existing quadratic NMIFE schemes performed on a virtual machine with access to 425 GB of DDR-5 memory and 22 CPU threads. Encryption time is measured for encrypting one record with length m for n clients.

	n	2			10			100		200
	m	1	2	3	1	2	3	1	2	1
Setup	AGT [AGT22]	0.2s	5.2s	3.2m	5.8s	29.9m	>10h	6.8h	>10h	>10h
	ZSHA [ZSHA25]	39.0ms	44.7ms	52.2ms	45.0ms	51.2ms	58.8ms	0.1s	0.1s	0.2s
	QUILT	1.3ms	1.7ms	2.1ms	3.2ms	5.3ms	7.8ms	25.6ms	50.4ms	50.4ms
Enc	AGT [AGT22]	0.4s	6.1s	40.1s	2.8s	59.9s	-	8.7s	-	-
	ZSHA [ZSHA25]	33.0ms	78.5ms	0.1s	34.9ms	75.8ms	0.1s	33.4ms	76.5ms	34.4s
	QUILT	0.7ms	1.3ms	2.0ms	0.9ms	1.9ms	2.4ms	2.1ms	3.2ms	1.9ms
Key Gen	AGT [AGT22]	32.6ms	0.2s	0.7s	0.5s	4.3s	-	49.6s	-	-
	ZSHA [ZSHA25]	8.6ms	8.6ms	8.7ms	42.9ms	42.9ms	43.2ms	0.4s	0.5s	0.9s
	QUILT	4.0ms	7.9ms	11.9ms	19.8ms	39.1ms	59.2ms	0.2s	0.4s	0.4s
Dec	AGT [AGT22]	0.4s	4.5s	30.9s	24.6s	7.4m	-	3.6h	-	-
	ZSHA [ZSHA25]	89.9ms	0.5s	0.6s	1.3s	5.1s	12.4s	1.75m	7.9m	6.9m
	QUILT	38.2ms	52.1ms	71.5ms	0.1s	0.1s	0.3s	0.8s	1.4s	1.4s

2.1 Design Principles

One of the main challenges is how to combine several *linear* FE schemes to realize a *quadratic* FE scheme. The previous works by Agrawal et al. [AGT21; AGT22] and the adaptation by Zalonis et al. [ZSHA25] exploit Equation (2) to realize the quadratic terms $\mathbf{x}_I \cdot \mathbf{x}_K$.

More precisely, one of the used building blocks is a single-input FE scheme supporting IP functionality, which we will call iFE. Then, each monomial $\mathbf{x}_I$ is represented *both* as a ciphertext $[\mathbf{x}_I]_{\mathsf{iFE,ct}}$ and as decryption key $[\mathbf{x}_I]_{\mathsf{iFE,dk}}$ and all these values are published as part of the ciphertext. Due to Equation (2), all quadratic terms $\mathbf{x}_I \cdot \mathbf{x}_K$ can be realized by computing

$$\mathsf{Dec}([\mathbf{x}_I]_{\mathsf{iFE,dk}}, [\mathbf{x}_K]_{\mathsf{iFE,ct}}) = \langle \mathbf{x}_I, \mathbf{x}_k \rangle = \mathbf{x}_I \cdot \mathbf{x}_K. \tag{3}$$

To avoid leakage of intermediate values and prevent mix-and-match attacks, additional randomness is introduced to the ciphertexts and decryption keys. A second *clean-up* MIFE scheme is later used to remove this randomness. Here, the randomness only cancels out if the adversary combines the results of Equation (3) with the appropriate coefficients $\boldsymbol{c}_{I,K}$ and computes the sum over all I, K.

With respect to the computational effort, there are three main driving factors. First, their construction requires a pairing structure of the underlying building blocks to combine $\mathbf{x}_I \cdot \mathbf{x}_K$ with $\boldsymbol{c}_{I,K}$. Second, iFE needs to be function-hiding (FH)[3] to ensure that $[\mathbf{x}_I]_{\mathsf{iFE,dk}}$ does not reveal $\mathbf{x}_I$ or the embedded randomness. Third, the number of decryption operations (3) required to realize all pairs $\mathbf{x}_I \mathbf{x}_K$ is quadratic in $n \cdot m$.

One main goal of our construction was to avoid these costly factors. To this end, we opt for a different decomposition of the quadratic function, namely:

$$Q(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \sum_{I,K} \boldsymbol{c}_{I,K} \mathbf{x}_I \mathbf{x}_K = \sum_I \underbrace{\left(\sum_K \boldsymbol{c}_{I,K} \mathbf{x}_K \right)}_{=: Q_I(\mathbf{x}_1, \ldots, \mathbf{x}_n)} \mathbf{x}_I. \tag{4}$$

We observe that $Q(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \sum_I Q_I \cdot \mathbf{x}_I$ is now a linear function, if we interpret $Q_I(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ as fixed values.

However, to securely multiply evaluations of $Q_I(\cdot)$ to $\mathbf{x}_I$ is non-trivial. First, we cannot treat $Q_I(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ as function coefficients directly, since

[3] Function-hiding (FH) means that the decryption key $[f]_{S,\mathsf{dk}}$ does not leak any information about the function f.

they depend on multiple plaintexts. That is, an authority cannot generate appropriate decryption keys without knowing the plaintexts. Second, the intermediate results of $Q_I(\cdot)$ must not be leaked to any party directly, as this may compromise the confidentiality of the plaintexts.

We solve this by using an IP-NMCFE scheme, nh. That is, nh allows for noisy evaluations of given linear functions where the noise is secret. Concretely, given ciphertexts $\{[\mathbf{x}_i]_{\mathsf{nh},\mathsf{ct},i}\}_{i\in[n]}$ and a decryption key $[Q_I(\cdot)+\phi_I]_{\mathsf{nh},\mathsf{dk}}$, running the decryption operation yields $z_I := Q_I(\mathbf{x}_1,\ldots,\mathbf{x}_n)+\phi_I \mod q$. The security properties of nh guarantee that the decryption key does not leak any information about ϕ_I. Moreover, we sample the noise ϕ_I uniformly over $\mathbb{Z}_q$, which essentially means that the decryption operation yields an one-time-pad (OTP) ciphertext of Q_I, mitigating any information leakage coming from z_I. Note that similar to the previous approaches, we are going to use a second *clean-up* multi-client functional encryption (MCFE) scheme to remove the random values at the end. However, our construction does not directly rely on bilinear groups.

It remains to realize the multiplication of z_I with $\mathbf{x}_I$. Inspired by the previous works [AGT22; ZASH24], it seems promising to exploit the following homomorphic property of certain MIFE schemes S:

$$\sum_t a_t \cdot [\mathbf{x}_t]_{S,\mathsf{ct},i} = \left[\sum_t a_t \cdot \mathbf{x}_t\right]_{S,\mathsf{ct},i}, \quad \text{if } \sum_t a_t = 1, \tag{5}$$

where each $[\mathbf{x}_t]_{S,\mathsf{ct},i}$ is a ciphertext for the same slot i.

However, we face two problems. First, an attacker could run mix-and-match attacks: instead of faithfully combining $[\mathbf{x}_I]_{S,\mathsf{ct},I}$ with the corresponding value z_I, an attacker could try to decrypt combinations $[\mathbf{x}_I \cdot z_K]_{S,\mathsf{ct},I}$ or, more general, to replace the values z_I with arbitrary scalars σ. This would allow to derive further information about the plaintext values $\mathbf{x}_I$. To prevent this, we follow a similar approach as above: we replace the ciphertexts $[\mathbf{x}_I]_{S,\mathsf{ct},I}$ with $[\mathbf{x}_I+\zeta_I]_{S,\mathsf{ct},I}$ where ζ_I are uniformly sampled over $\mathbb{Z}_q$ using a secret key associated with I. As the random ζ_I need to be removed at the end, malicious combinations would still contain random values, rendering the decryption results useless for an attacker. Second, all current MIFE and MCFE schemes fulfilling the homomorphic property (Equation (5)) require the sum of the scalars for each slot to be equal to 1. However, we cannot ensure that $z_I = 1$ for any I as z_I is random over $\mathbb{Z}_q$. A common circumvention, which is also used in related work, is to provide for each slot I an additional encryption of zero,

$[0]_{S,\mathsf{ct},I}$, and use the homomorphic property as follows:

$$[\mathbf{x}_I]_{S,\mathsf{ct},I} \cdot z_I + [0]_{S,\mathsf{ct},I} \cdot (1 - z_I) = [\mathbf{x}_I \cdot z_I]_{S,\mathsf{ct},I}\,. \tag{6}$$

While this solves the problem with respect to the scalars, an attacker now has access to two encryptions per I: one of $\mathbf{x}_I$ and one of 0. This causes problems in the security proof of the scheme, as reducing the security of our construction to that of S would grant the adversary significantly more power than typically allowed by the underlying security game of S. Previous constructions relying on homomorphy do not have this problem as it is used in the encryption process only, i. e., Equation (5) is not performed by the attacker.

Our way out is based on a double use of the values ζ_I. On the one hand, they serve as randomness to avoid mix-and-match attacks. On the other hand, as the ζ_I are uniformly sampled, they also realize OTP encryptions of the $\mathbf{x}_I$ values. Thus, we can remove the additional FE encryption layer and get $[z_I\mathbf{x}_I]_{\mathsf{otp},\mathsf{ct},I}$ directly through simple scalar multiplication.

Combining these pieces over $\mathbb{Z}_q$ yields:

$$\begin{aligned}
\sum_I z_I\,(\mathbf{x}_I + \zeta_I) &= \sum_I (Q_I(\mathbf{x}_1, \ldots, \mathbf{x}_n) + \phi_I) \cdot (\mathbf{x}_I + \zeta_I) \\
&= \sum_I \left(\sum_K \boldsymbol{c}_{I,K}\mathbf{x}_K + \phi_I \right) \cdot (\mathbf{x}_I + \zeta_I) \\
&= \underbrace{\sum_{I,K} \boldsymbol{c}_{I,K}\mathbf{x}_I\mathbf{x}_K}_{Q(\mathbf{x}_1,\ldots,\mathbf{x}_n)} + \underbrace{\sum_{I,K} \boldsymbol{c}_{I,K}\mathbf{x}_K\zeta_I + \sum_I (\mathbf{x}_I\phi_I + \phi_I\zeta_I)}_{\tilde{f}(\mathbf{x}_1,\ldots,\mathbf{x}_n)}\,.
\end{aligned}$$

To remove the additional randomness $\tilde{f}$, it is sufficient to use an FH-MCFE scheme, similar to previous constructions. More precisely, $\tilde{f}$ can be seen as an affine function in the $\mathbf{x}_I$ values, with coefficients depending both on the original coefficients $\boldsymbol{c}_{I,K}$ of Q, and the noise terms ϕ_I and ζ_I. This allows the authority to generate decryption keys for $\tilde{f}$ without the need of knowing the $\mathbf{x}_I$ values. Moreover, the adversary can only remove the randomness if the decryption operations are executed as anticipated.

In contrast to previous schemes, the effort for this second *clean-up* FE scheme is slightly higher, since we require an FH-MCFE scheme that is linear in $n \cdot m$, whereas the scheme in [ZSHA25] only depends on $3n$. However, the costly operations in Equation (3) are avoided completely. We only require an NMCFE scheme, which can be constructed efficiently without the need for pairings [SHZ25], where the decryption operation of NMCFE is invoked nm times. As we show in Section 6, this results in an enormous speedup.

2.2 Future Directions

In the following, we discuss current limitations of our approach and outline possible directions for future improvements. First, the clean-up MCFE scheme needs to be FH. At the moment such schemes are only realized using pairings, which require solving a discrete logarithm at the end of the decryption process. Finding a pairing free realization of this building block could yield an efficiency gain and allow for quadratic functions whose output can be any value in $\mathbb{Z}_q$[4]. Additionally, this new building block could also be used to replace nh and yield a quadratic FH-OT-NMCFE scheme. Although it is not possible to construct an FH-IP-FE scheme from lattices, which allows for an unbounded number of ciphertexts and decryption keys simultaneously [Ünal20], as we are only considering one-time encryption, this impossibility result might not apply. For example, taking advantage of masking techniques, it might be possible to design constructions which hide the function embedded in the decryption key with randomness that cancels out during decryption.

Second, we present our construction only for quadratic functions. While our approach could also be used to construct schemes of higher degrees, constructing schemes for functions of degree d requires the deployed MCFE schemes to support functions of degree $d-1$. Additionally, for the clean-up MCFE scheme, fh, we need the FH property and for the NMCFE scheme, nh, it is necessary to choose noise that is uniformly sampled over the whole group $\mathbb{Z}_q$. As for now, no schemes for higher degrees, i. e., $d > 1$, are known that fulfill these requirements. All current higher degree schemes that are unbounded in the number of decryption keys use to some extent FH building blocks based on pairings, and thus do not support evaluations over $\mathbb{Z}_q$. On the other hand, at the moment there likewise do not exist any FH schemes for quadratic functionality.

Summing up, designing new nh and fh building blocks that are pairing free and/or support higher degrees are interesting open questions that would directly improve our proposed construction.

[4] Schemes that operate over lattices can support evaluations that are not bounded in the output but rather work over $\mathbb{Z}_q$ [ALS16].

3 Preliminaries

3.1 Notation

Let λ define the security parameter. $[n]$ denotes the set $\{1, \ldots, n\}$ and $[n_1, n_2]$ the set $\{n_1, \ldots, n_2\}$ for any $n, n_1, n_2 \in \mathbb{N}$, where $\mathbb{N}$ defines the set of all positive integers and $\mathbb{Z}$ the set of all integers. For any $q \in \mathbb{N}$, the term $\mathbb{Z}_q$ refers to the residue class ring of size q.

Vectors and matrices are denoted by bold letters, i.e. $\mathbf{x}$ denotes a vector, where $\mathbf{x}[j]$ is the jth element of $\mathbf{x}$. We denote by $\mathbf{x}[(i,j)] \in \mathbb{Z}_q^{nm+1}$ the $(i,j)^{th}$ element defined by: $\mathbf{x}[(i,j)] := \mathbf{x}[(i-1)m + j], i \in [n], j \in [m]$ and $\mathbf{x}[(0,0)] := \mathbf{x}[nm+1]$. Similarly for $\mathbf{x}[(i,j,k,l)] \in \mathbb{Z}_q^{(nm)^2+nm+1}$ the $(i,j,k,l)^{th}$ element is defined by: $\mathbf{x}[(i,j,k,l)] := \mathbf{x}[(i-1)nm^2 + (j-1)nm + (k-1)m + l], i, k \in [n], j, l \in [m]$ and $\mathbf{x}[(0,0,i,j)] := \mathbf{x}[(nm)^2 + (i-1)m + j]$ and $\mathbf{x}[(0,0,0,0)] := \mathbf{x}[(nm)^2 + nm + 1]$.

$\mathbf{0}^n$ denotes the zero vector of length n. For a function $f : \mathcal{X}_1 \times \cdots \times \mathcal{X}_n \to \mathcal{Y}$ and some $\nu \in \mathcal{Y}$, we denote by $(f + \nu)(\mathbf{x}) := f(\mathbf{x}) + \nu$ for all $\mathbf{x} \in \mathcal{X}_1 \times \cdots \times \mathcal{X}_n$. For a subset $\Delta \subseteq \mathcal{Y}$, we define the set $f + \Delta = \{f + \nu | \nu \in \Delta\}$. For a distribution $\mathbb{D}$ over some set Δ, $\Pr[x \xleftarrow{\mathbb{D}} \Delta]$ denotes the probability that x is sampled according to distribution $\mathbb{D}$ over Δ. For a value $\nu \in \Delta$ we denote the distribution $\mathbb{D}_\nu$ as the distribution that outputs ν with probability 1. $x \xleftarrow{\$} \Delta$ represents the process of uniformly sampling an element $x \in \Delta$. We denote by $\mathbb{D}_1 \equiv \mathbb{D}_2$ the equivalence of two distributions $\mathbb{D}_1$ and $\mathbb{D}_2$ over some Δ. We use the abbreviation PPT to mean probabilistic polynomial time. A function $\mathtt{negl} : \mathbb{N} \to \mathbb{R}^+$ 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 $\mathtt{negl}(x) < 1/x^c$.

3.2 (Noisy) Multi-Client Functional Encryption

In the following we recall the formal definition for an NMCFE scheme from [SHZ25]. An NMCFE scheme allows to evaluate perturbed functions $f_\ell(\mathbf{x}_{1,\ell}, \ldots, \mathbf{x}_{n,\ell}) + \nu_\ell$ on encrypted data, where f_ℓ is associated with some distribution $\mathbb{D}$ and a label ℓ, so that $\nu_\ell \xleftarrow{\mathbb{D}} \Delta$ and only ciphertexts and functions associated with the same label ℓ can be combined and evaluated.

Definition 1 (Noisy Multi-Client Functional Encryption [SHZ25])**.** *Let $\mathcal{F}_{n,\pi}$ be a family of n-ary functions $\{f : \mathcal{X}_1 \times \cdots \times \mathcal{X}_n \to \mathcal{Y}\}$, where π is an additional set of parameters and $\mathcal{L} = \{0,1\}^* \cup \{\perp\}$ be a set of labels. A*

noisy multi-client functional encryption scheme NMCFE *for* $\mathcal{F}_{n,\pi}$ *and label space* $\mathcal{L}$ *consists of the following algorithms:*

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

$\mathsf{Enc}(\mathsf{ek}_i, i, x, \ell)$ *On input the encryption key* ek_i *for slot* $i \in [n]$*, a message* $x \in \mathcal{X}_i$ *and a label* $\ell \in \mathcal{L}$*, 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.*

$\mathsf{KeyGen}(\mathsf{msk}, f, \ell, \mathbb{D})$**:** *On input the master secret key* msk*, a function* $f \in \mathcal{F}_{n,\pi}$*, a label* $\ell \in \mathcal{L}$ *and a distribution* $\mathbb{D}$ *over some* $\Delta \subseteq \mathcal{Y}$ *such that* $f + \Delta \in \mathcal{F}_{n,\pi}$*, samples* $\nu \xleftarrow{\mathbb{D}} \Delta$ *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* $\perp$*.*

Remark 1. *If one sets* $\mathbb{D}$ *to be the distribution that always outputs zero, i. e.,* $\mathbb{D}_0$*,* $\mathbb{D}$ *can be omitted in the* KeyGen *algorithm and Definition 1 reduces to the classical definition of an MCFE scheme, where the decryption keys are also labeled. If we set* $n = 1$ *we have a single-input FE scheme, i.e. a scheme with only one input. For the latter case we often omit* n *and* i*. If the label space* $\mathcal{L}$ *contains only a single label, i. e.,* $\mathcal{L} = \{\ell\}$*, this definition reduces to the classical MIFE definition and we omit the label.*

An NMCFE scheme should be correct and secure. Correctness essentially means that applying Dec to an encryption of some messages x_i associated with a label ℓ with a decryption key $\mathsf{dk}_{f,\ell}$ requested for some function f associated with the same label ℓ results in $f(x_1, \ldots, x_n) + \nu$, where the value of ν follows the distribution $\mathbb{D}$. More formally:

Definition 2 (Correctness of NMCFE)**.** *A noisy multi-client functional encryption scheme* NMCFE *(Definition 1) is correct if for all* $\ell \in \mathcal{L}$*, for all* $x_i \in \mathcal{X}_i$ *for* $1 \leq i \leq n$ *and for all distributions* $\mathbb{D}_f$ *over some set* $\Delta \subseteq \mathcal{Y}$ *with* $f + \Delta \in \mathcal{F}_{n,\pi}$*, we have*

$$\Pr\left[\begin{array}{c}(\mathsf{pp}, \mathsf{msk}, \{\mathsf{ek}_i\}_{i\in[n]}) \leftarrow \mathsf{Setup}(\lambda, \mathcal{F}_{n,\pi}),\\ \mathsf{dk}_{f,\ell} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f, \ell, \mathbb{D}_f) :\\ \mathsf{Dec}(\mathsf{dk}_{f,\ell}, \mathsf{ct}_{1,\ell}, \ldots, \mathsf{ct}_{n,\ell}) - f(x_1, \ldots x_n)\end{array}\right] \equiv \mathbb{D}_f$$

with $\mathsf{ct}_{i,\ell} \leftarrow \mathsf{Enc}(\mathsf{ek}_i, i, x_i, \ell)$*, where the probability is taken over the coins of* Setup*,* KeyGen *and* Enc*.*

We base our security notion of NMCFE on the definition from [SHZ25], where security ensures, that ciphertexts $\mathsf{ct}_{i,\ell}$ and decryption keys $\mathsf{dk}_{f,\ell}$ should not leak any information about the encrypted messages and the noise values other than the allowed results $f(\mathbf{x}_1, \ldots, \mathbf{x}_n) + \nu$ for functions and messages associated with the same label ℓ. If the adversary combines functions and ciphertexts generated under different labels, they should not learn any information at all. Besides labels, an NMCFE scheme allows the adversary to corrupt clients and obtain their encryption keys, which means the adversary is able to encrypt values $\mathbf{x}'_i \in \mathcal{X}_i$ on their own for corrupted clients i. This should not leak any information about the inputs of the honest clients.

Without loss of generality, we assume that the adversary queries distributions $\mathbb{D}_\nu$, which output with overwhelming probability only one value ν, thereby granting the adversary the greatest possible power.

Definition 3 (IND-CPA Security)**.** *Let* $\mathsf{xx} \in \{\mathsf{nh}, \mathsf{fh}\}, \mathsf{yy} \in \{\mathsf{ot}, \mathsf{many}\}$. *Given a PPT adversary* $\mathcal{A}$ *against an NMCFE scheme* NMCFE *for the function family* $\mathcal{F}_{n,\pi}$ *and label space* $\mathcal{L}$*, we define the experiment* $\mathsf{Expt}_\beta^{\mathsf{NMCFE}}(\mathcal{A}, 1^\lambda)$ *for* $\beta \in \{0,1\}$ *as in Figure 1, where* $\mathcal{O}.\mathsf{KeyGen}$ *is an oracle that on input* $(f^0, f^1, \nu^0, \nu^1, \ell)$ *outputs* $\mathsf{KeyGen}(\mathsf{msk}, f^\beta, \ell, \mathbb{D}_{\nu^\beta})$*,* L_{req} *the set of queried labels and* $\mathcal{CS} \subseteq [n]$ *denotes the set of corrupted clients. The adversary* $\mathcal{A}$ *is admissible, denoted by* $\mathsf{adm}(\mathcal{A}) = 0$*, if the following conditions hold. Otherwise, we say that* $\mathcal{A}$ *is not admissible and write* $\mathsf{adm}(\mathcal{A}) = 1$.

Let $q_{\mathsf{dk},\ell}$ *be the number of decryption key queries per label* $\ell \in \mathcal{L}$ *and* $\mathsf{Leak}(i, f)$ *the leakage of information from* dk *by knowing* ek_i*.*

- *If* $i \in \mathcal{CS}$*, then for all* $(x^0, x^1, i, \cdot) \in \mathcal{MS}$*,* $x^0 = x^1$ *and*

$$\mathsf{Leak}(i, f^0) = \mathsf{Leak}(i, f^1), \tag{7}$$

  i. e., the ciphertext queries for all corrupted clients and the information obtained by dk *through* ek_i *have to be independent of* β*.*

- *Let* $\mathcal{MS}_{\mathcal{CS}} = \{(x, x, i, \ell) |\ x \in \mathcal{X}, \ell \in L_{\mathsf{req}}, i \in \mathcal{CS}\}$ *and* $\widetilde{\mathcal{MS}} := \mathcal{MS} \cup \mathcal{MS}_{\mathcal{CS}}$*. For all* $(x^0, x^1, i, \ell) \in \widetilde{\mathcal{MS}}$*, denote by* $x^b_{i,\ell} := x^b$*,* $b \in \{0,1\}$*. Then*

$$f_\ell^{k,0}(x_{1,\ell}^0, \ldots, x_{n,\ell}^0) + \nu_\ell^{k,0} = f_\ell^{k,1}(x_{1,\ell}^1, \ldots, x_{n,\ell}^1) + \nu_\ell^{k,1} \tag{8}$$

  for all $(x^0, x^1, i, \ell) \in \widetilde{\mathcal{MS}}$*,* $k \in [q_\ell]$*, i. e.,* $\mathcal{A}$ *cannot distinguish both games through the evaluation of the function.*

$\mathsf{Expt}_{\beta}^{\mathsf{NMCFE}}(\mathcal{A}, 1^{\lambda})$

$(\mathsf{pp}, \mathsf{msk}, \{\mathsf{ek}_i\}_{i\in[n]}) \leftarrow \mathsf{Setup}(1^{\lambda}, \mathcal{F}^n)$

$(\mathcal{MS}, \mathcal{CS}) \leftarrow \mathcal{A}(1^{\lambda}, \mathcal{F}^n)$ s.t.

$\quad \mathcal{MS} = \{(x_{i,\ell}^{\mu_i,0}, x_{i,\ell}^{\mu_i,1}, i, \ell)\}_{i\in[n],\ell\in L_{\mathsf{req}}\subseteq\mathcal{L},\mu_i\in[q_{\mathsf{ct},i,\ell}]}$,

$\quad \mathcal{CS} \subseteq [n]$

$\mathsf{ct}_{i,\ell}^{\mu_i} \leftarrow \mathsf{Enc}(\mathsf{ek}_i, i, x_{i,\ell}^{\mu_i,\beta}, \ell)\ \forall i \in [n],\ \ell \in L_{\mathsf{req}}, \mu_i \in [q_{\mathsf{ct},i,\ell}]$

$\alpha \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{KeyGen}(\cdot,\cdot,\cdot,\cdot,\cdot)}(\mathsf{pp}, \{\mathsf{ek}_i\}_{i\in\mathcal{CS}}, \{\mathsf{ct}_{i,\ell}^{\mu_i}\}_{i\in[n],\ell\in L_{\mathsf{req}},\mu_i\in[q_{\mathsf{ct},i,\ell}]})$

if $\mathsf{adm}(\mathcal{A}) = 0$ **then**

$\quad$ **return** α

return $\alpha \xleftarrow{\$} \{0,1\}$

Figure 1: Security game $\mathsf{Expt}_{\beta}^{\mathsf{NMCFE}}(\mathcal{A}, 1^{\lambda})$ for Definition 3.

- *If* $\mathsf{xx} = \mathsf{nh}$*, then* $f_{\ell,k}^0 = f_{\ell,k}^1$ *for all* $\ell \in L_{\mathsf{req}}, k \in [q_\ell]$*, i. e., the functions have to be the same, only the noise values are hidden. In this case we shorten the input of* $\mathcal{O}.\mathsf{KeyGen}$ *to* (f, ν^0, ν^1, ℓ)
- *If* $\mathsf{yy} = \mathsf{ot}$*, then* $q_{\mathsf{ct},i,\ell} = 1$ *for all* $i \in [n], \ell \in L_{\mathsf{req}}$*.*
- *If* $q_{\mathsf{key},\ell} > 0$*, then* $\ell \in L_{\mathsf{req}}$*, i. e.,* $\mathcal{A}$ *cannot query functions for labels which do not have any ciphertexts.*

We say that NMCFE *is semi-adaptive* xx-yy*-secure for static corruptions if*

$$\begin{aligned} \mathit{Adv}_{\mathcal{A}}^{\mathsf{NMCFE}}(\lambda) &= |\Pr[\mathsf{Expt}_0^{\mathsf{NMCFE}}(\mathcal{A}, 1^{\lambda}) = 1] - \Pr[\mathsf{Expt}_1^{\mathsf{NMCFE}}(\mathcal{A}, 1^{\lambda}) = 1]| \\ &\leq \mathtt{negl}(\lambda). \end{aligned}$$

Remark 2. *If one sets* $\mathbb{D}$ *to be the distribution that always outputs zero, i. e.,* $\mathbb{D}_0$*, Definition 3 reduces to the definition of classical (FH-)MCFE, where the decryption keys are also labeled. If we omit the labels, we obtain the security definition for* xx-yy*-MIFE. Whenever we consider* ot *security,* μ_i *is omitted. For schemes, which do not support any corruptions, only* $\mathcal{CS} = \emptyset$ *is allowed.*

Functionalities of Functional Encryption

There exists only a small set of functionalities that can be currently achieved with efficient (N)MCFE schemes. A lot of them, especially all of the FH

schemes, are based on pairings. Consequently, the decryption algorithm includes an exhaustive search step to solve the discrete logarithm problem. To ensure decryption in groups where the problem is considered hard, we focus only on families of polynomials $\mathcal{F}_{n,\pi}$, where both the inputs and the coefficients are bounded by some values B_X and B_C, respectively. This makes it possible to place a bound on the function's result. In the following, we describe the functionalities in more detail:

Definition 4 (Bounded-Norm Functionality)**.** *For two positive values B_X and B_C, the expression $\mathcal{F}_{n,\{m,\mathsf{B}_X,\mathsf{B}_C\}}$ refers to a set of multivariate polynomials $f : (\mathcal{X}^m)^n \to \mathbb{Z}$ where the input space $\mathcal{X} = \{i \in \mathbb{Z}; |i| \leq \mathsf{B}_X\}$ is bounded by B_X and the coefficients of f, represented by a vector $\boldsymbol{c}$ over $\mathbb{Z}$, is bounded by B_C: $||\boldsymbol{c}||_\infty \leq \mathsf{B}_C$.*

$\mathcal{F}^{\mathtt{AF}}_{n,\{m,\mathsf{B}_X,\mathsf{B}_C\}}$ refers to affine functions specified by $\boldsymbol{c} \in \mathbb{Z}^{nm+1}$ as

$$f(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \boldsymbol{c}[(0,0)] + \sum_{i\in[n],j\in[m]} \boldsymbol{c}[(i,j)]\mathbf{x}_i[j]$$

with a constant value $\boldsymbol{c}[(0,0)]$. A special case is the inner-product functionality, dubbed $\mathcal{F}^{\mathtt{IP}}_{n,\{m,\mathsf{B}_X,\mathsf{B}_C\}}$, where $\boldsymbol{c}[(0,0)] = 0$ for all functions. In this case we often omit the constant value and interpret $\boldsymbol{c} \in \mathbb{Z}^{nm}$.

Lastly, $\mathcal{F}^{\mathtt{QF}}_{n,\{m,\mathsf{B}_X,\mathsf{B}_C\}}$ is a family of bounded-norm multi-input quadratic functions

$$f(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \sum_{\substack{i,k\in[n],\\ j,l\in[m]}} \boldsymbol{c}[(i,j,k,l)]\mathbf{x}_i[j]\mathbf{x}_k[l] + \sum_{\substack{i\in[n],\\ j\in[m]}} \boldsymbol{c}[(0,0,i,j)]\mathbf{x}_i[j] + \boldsymbol{c}[(0,0,0,0)]$$

with $\boldsymbol{c} \in \mathbb{Z}^{(nm)^2+nm+1}$. To ensure uniqueness, we require that if $(i,j) > (k,l)$, then $\boldsymbol{c}[(i,j,k,l)] = 0$.

We omit B_X and B_C whenever we consider unbounded functionalities, e. g., if no discrete logarithm has to be solved or only correctness modulo q is desired. In the latter case we interpret $\boldsymbol{c}$ as a vector over $\mathbb{Z}_q$.

Remark 3. *Observe that previous constructions often only achieve quadratic functions of the form $\sum_{i,k\in[n],j,l\in[m]} \boldsymbol{c}[(i,j,k,l)]\mathbf{x}_i[j]\mathbf{x}_k[l]$, i. e., functions without any linear or constant terms. This does not pose any restriction as the evaluation of arbitrary quadratic functions can be realized by either encrypting $(\mathbf{x}_i, 1)$ instead of $\mathbf{x}_i$ (that is one component is a constant term), or produce an additional ciphertext encrypting a one-hot or all-one vector. Note, that our scheme achieves full quadratic functionality by design and does not need to increase m or n by 1.*

Remark 4. *In the context of the security of NMCFE schemes for linear or quadratic functions, Equation (8) imposes certain restrictions on the types of functions an adversaries may query. More precisely, if we consider the existence of corrupted clients, i. e.,* $\mathcal{CS} \neq \emptyset$*, the adversary must not gain information over data of honest clients, denoted by* $\mathcal{HS} := [n] \setminus \mathcal{CS}$*. In case of affine functions associated with a vector* $\boldsymbol{c}$ *with* $\boldsymbol{c}_i := (\boldsymbol{c}[(i,1)], \ldots, \boldsymbol{c}[(i,m)])$*, this essentially breaks down to* $\sum_{i \in \mathcal{HS}} \langle \boldsymbol{c}_i^0, \mathbf{x}_i^0 \rangle = \sum_{i \in \mathcal{HS}} \langle \boldsymbol{c}_i^1, \mathbf{x}_i^1 \rangle$*, as it holds that* $\mathbf{x}_i^0 = \mathbf{x}_i^1$ *for* $i \in \mathcal{CS}$*. If these schemes are also FH,* $\boldsymbol{c}_i^0 = \boldsymbol{c}_i^1$ *is additionally required to ensure* $\mathsf{Leak}(i, f^0) = \mathsf{Leak}(i, f^1)$ *(Equation (7)).*

For quadratic functions, we have to deal with summands that mix corrupted and honest data, i. e., $\boldsymbol{c}[(i,j,k,l)]\mathbf{x}_i[j]\mathbf{x}_k[l]$*, where* $i \in \mathcal{CS}$ *and* $k \in \mathcal{HS}$ *or vice versa. As the adversary can encrypt any* $\mathbf{x} \in \mathcal{X}$ *for the slots of corrupted clients, this means, it must hold that*

$$\sum_{\substack{k \in [n], \\ l \in [m]}} \boldsymbol{c}_{i,j,k,l} \mathbf{x}_k^0[l] = \sum_{\substack{k \in [n], \\ l \in [m]}} \boldsymbol{c}_{i,j,k,l} \mathbf{x}_k^1[l].$$

For the linear part, i. e., $\boldsymbol{c}[(0,0,i,j)]$*, the same restriction as for affine functions has to hold, i. e.,* $\sum_{i \in \mathcal{HS}, j \in [m]} \boldsymbol{c}[(0,0,i,j)]\mathbf{x}_i^0[j] = \sum_{i \in \mathcal{HS}, j \in [m]} \boldsymbol{c}[(0,0,i,j)]\mathbf{x}_i^1[j]$*.*

4 Generic Construction

In the following we present a new construction method for quadratic OT-NMCFE schemes. As defined by the notion of NMCFE, this constitutes an OT-MCFE scheme when the noise distribution is set to the zero distribution (i.e., the distribution that always outputs zero). If the support of labels is not needed, the building blocks can be adapted accordingly for constructing an OT-(N)MIFE scheme.

We first describe the construction method in Section 4.1, show its correctness in Section 4.2, and prove its security through reduction on the underlying building blocks in Section 5.

4.1 Description

Our construction relies on three building blocks:

1. An OT-NMCFE scheme nh for $\mathcal{F}_{n,\{m\}}^{\mathtt{IP}}$ supporting noise distributions and inner-products over $\mathbb{Z}_q$.

2. A pseudo-random function $\mathsf{PRF} : \{0,1\}^\lambda \times \mathcal{L} \to \mathbb{Z}_q^m$ with key space $\{0,1\}^\lambda$ for realizing the OTP encryption of the plaintexts.
3. An FH-OT-MCFE scheme fh that supports affine functions $\mathcal{F}^{\mathtt{AF}}_{n,\{m,\mathsf{B}_X,\mathsf{B}_C\}}$.

Additionally, all schemes have to operate over $\mathbb{Z}_q$ and results from nh and fh can be combined. In the case that fh relies on bilinear pairing groups, the results of nh need to be lifted in the target group and combined with those from fh before the final decryption step. Specific instantiations of the building blocks are presented in Section 6.

Next, we explain the four algorithms Setup, Enc, KeyGen, and Dec in detail.

$\mathsf{Setup}(1^\lambda, \mathcal{F}^{\mathtt{Quad}}_{n,\{m,\mathsf{B}_X,\mathsf{B}_C\}})$ takes as input the security parameter λ and a description of $\mathcal{F}^{\mathtt{Quad}}_{n,\{m,\mathsf{B}_X,\mathsf{B}_C\}}$ where n denotes the number of clients, m the vector length, and B_X and B_C being the bounds for the input vectors $\mathbf{x}$ and the function coefficients $\boldsymbol{c}$, respectively. It performs the following steps:

$$(\mathsf{nh.msk}, \{\mathsf{nh.ek}_i\}_{i\in[n]}, \mathsf{nh.pp}) \leftarrow \mathsf{nh.Setup}(1^\lambda, \mathcal{F}^{\mathtt{IP}}_{n,\{m\}}),$$
$$\forall i \in [n] : K_i \xleftarrow{\$} \{0,1\}^\lambda,$$
$$(\mathsf{fh.msk}, \{\mathsf{fh.ek}_i\}_{i\in[n]}, \mathsf{fh.pp}) \leftarrow \mathsf{fh.Setup}(1^\lambda, \mathcal{F}^{\mathtt{AF}}_{n,\{m,\mathsf{B}_X,\mathsf{B}_C\}}).$$

It sets $\mathsf{msk} := (\mathsf{nh.msk}, \{K_i\}_{i\in[n]}, \mathsf{fh.msk})$ and $\mathsf{ek}_i := (\mathsf{nh.ek}_i, K_i, \mathsf{fh.ek}_i)$ and publishes $\mathsf{pp} := (\mathsf{nh.pp}, \mathsf{fh.pp})$. All of the following algorithms receive pp implicitly as input.

$\mathsf{Enc}(\mathsf{ek}_i, i, \mathbf{x}_i, \ell)$ takes as input the secret encryption key ek_i, a message $\mathbf{x}_i \in \mathcal{X}_i$ associated with slot i, and a label $\ell \in \mathcal{L}$. It proceeds as follows:

1. Generate
$$\mathsf{nh.ct}_i \leftarrow \mathsf{nh.Enc}(\mathsf{nh.ek}_i, i, \mathbf{x}, \ell).$$
2. Calculate $\boldsymbol{\zeta}_i \leftarrow \mathsf{PRF}(K_i, \ell)$ and set
$$\mathsf{otp.ct}_i := \mathbf{x}_i + \boldsymbol{\zeta}_i.$$
3. Compute
$$\mathsf{fh.ct}_i \leftarrow \mathsf{fh.Enc}(\mathsf{fh.ek}_i, i, \mathbf{x}_i, \ell).$$

It returns $\mathsf{ct}_i = (\mathsf{nh.ct}_i, \mathsf{otp.ct}_i, \mathsf{fh.ct}_i)$.

$\mathsf{KeyGen}(\mathsf{msk}, \boldsymbol{c}, \ell, \mathbb{D})$ takes as input the master secret key msk, a coefficient vector $\boldsymbol{c} \in \mathbb{Z}_q^{(nm)^2+nm+1}$, where $\boldsymbol{c}[(i,j,k,l)] = 0$ if $(i,j) > (k,l)$, a label ℓ, and a distribution $\mathbb{D}$. It generates the decryption key as follows:

1. Sample $\phi_{i,j} \xleftarrow{\$} \mathbb{Z}_q$ for all $i \in [n], j \in [m]$ and compute

$$\mathsf{nh.dk}_{i,j} \leftarrow \mathsf{nh.KeyGen}(\mathsf{nh.msk}, (\boldsymbol{c}[(i,j,1,1)], \ldots, \boldsymbol{c}[(i,j,n,m)]), \ell, \mathbb{D}_{\phi_{i,j}}),$$

2. Compute $\boldsymbol{\zeta}_i \leftarrow \mathsf{PRF}(K_i, \ell), \forall i \in [n]$, sample $\nu \xleftarrow{\mathbb{D}} \Delta$, and generate

$$\mathsf{fh.dk} \leftarrow \mathsf{fh.KeyGen}(\mathsf{fh.msk}, \mathbf{y}, \ell),$$

where

$$\mathbf{y}[(i,j)] = \sum_{k \in [n], l \in [m]} \boldsymbol{c}[(k,l,i,j)]\boldsymbol{\zeta}_k[l] + \phi_{i,j} - \boldsymbol{c}[(0,0,i,j)] \quad (9)$$
$$\mathbf{y}[(0,0)] = \sum_{i \in [n], j \in [m]} \phi_{i,j}\boldsymbol{\zeta}_i[j] - \boldsymbol{c}[(0,0,0,0)] - \nu.$$

It outputs $\mathsf{dk} = (\{\mathsf{nh.dk}_{i,j}\}_{i \in [n], j \in [m]}, \mathsf{fh.dk})$.

$\mathsf{Dec}(\mathsf{dk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_n)$ takes as input a decryption key dk and n ciphertexts. It performs the following steps:

1. For all $i \in [n], j \in [m]$ compute

$$z_{i,j} \leftarrow \mathsf{nh.Dec}(\mathsf{nh.dk}_{i,j}, \mathsf{nh.ct}_1, \ldots, \mathsf{nh.ct}_n) \quad (10)$$

and calculate

$$r_1 := \sum_{i \in [n], j \in [m]} z_{i,j} \cdot \mathsf{otp.ct}_i[j].$$

2. Compute $r_2 \leftarrow \mathsf{fh.Dec}(\mathsf{fh.dk}, \mathsf{fh.ct}_1, \ldots, \mathsf{fh.ct}_n)$. If fh operates over bilinear groups perform the decryption without the final search step.
3. Set $r := r_1 - r_2$. If fh operates over bilinear groups first lift r_1 to the target group and finally perform an exhaustive search for r over the range

$$\binom{nm+1}{2}\mathsf{B}_X^2\mathsf{B}_C + nm\mathsf{B}_X\mathsf{B}_C + \mathsf{B}_C + \delta,$$

where $\delta = \max\{\nu \in \Delta\}$.

4.2 Correctness

The correctness of our scheme relies on the correctness of the building blocks. For correct NMCFE schemes nh and fh it holds that

$$\begin{aligned} r_1 = & \sum_{i,k\in[n],j,l\in[m]} \boldsymbol{c}[(i,j,k,l)]\left(\mathbf{x}_i[j]\mathbf{x}_k[l] + \mathbf{x}_k[l]\boldsymbol{\zeta}_i[j]\right) \\ & + \sum_{i\in[n],j\in[m]} \phi_{i,j}\left(\mathbf{x}_i[j] + \boldsymbol{\zeta}_i[j]\right) \\ r_2 = & \sum_{i,k\in[n],j,l\in[m]} (\boldsymbol{c}[(k,l,i,j)]\mathbf{x}_i[j]\boldsymbol{\zeta}_k[l]) - \sum_{i\in[n],j\in[m]} (\boldsymbol{c}[(0,0,i,j)]\mathbf{x}_i[j] \\ & + \phi_{i,j}\left(\mathbf{x}_i[j] + \boldsymbol{\zeta}_i[j]\right)) - c[0000] - \nu \end{aligned}$$

and thus finally, with the same probability as ν is sampled from $\mathbb{D}$, that

$$\begin{aligned} r = & \sum_{i,k\in[n],j,l\in[m]} \boldsymbol{c}[(i,j,k,l)]\mathbf{x}_i[j]\mathbf{x}_k[l] + \sum_{i\in[n],j\in[m]} \boldsymbol{c}[(0,0,i,j)]\mathbf{x}_i[j] \\ & + \boldsymbol{c}[(0,0,0,0)] + \nu \end{aligned}$$

5 Security

In this section, we prove the security of our construction, based on the security of the building blocks. We start with a high-level overview in Section 5.1 before we prove the security in more detail in Section 5.2.

5.1 Overview

We prove the security of our construction according to Definition 3, which essentially means that an attacker should not be able to distinguish between games G_0 (attacker receives encryptions of the $\mathbf{x}_i^0$ values and decryption keys encoding noise ν^0) and G_1 (attacker receives encryptions of the $\mathbf{x}_i^1$ values and decryption keys encoding noise ν^1). This is accomplished via a sequence of games $\mathsf{G}_0, \mathsf{H}_1, \ldots, \mathsf{H}_4, \mathsf{G}_1$, where we sub-sequentially change the ciphertexts of the building blocks to encrypt $\mathbf{x}_i^1$ instead of $\mathbf{x}_i^0$ and the noise value from ν^0 to ν^1. Here, one needs to ensure that there are no inconsistencies in the evaluation which would allow an adversary to distinguish those games. To keep the overview simple, we will neither consider corruptions nor integrate the labels in the explanation. In the detailed proof, both will be addressed accordingly.

$\mathsf{G}_0 \to \mathsf{H}_1$:

The first change is to replace $\mathbf{x}_i^0$ with $\mathbf{x}_i^1$ in the specification of $\mathsf{nh.ct}$. To avoid inconsistencies in the overall evaluation, the decryption of nh still needs to yield $f_{i,j}(\mathbf{x}^0) := \sum_{k,l} \boldsymbol{c}[(i,j,k,l)]\mathbf{x}_k[l] + \phi_{i,j}$, which can be achieved by modifying the hidden noise value accordingly (Figure 3). The indistinguishably is ensured by the security properties of nh (cf. Lemma 2).

$\mathsf{H}_1 \to \mathsf{H}_2 \to \mathsf{H}_3$:

The next step is to switch the messages in $\mathsf{otp.ct}_i = \mathbf{x}_i^\beta + \boldsymbol{\zeta}_i$. At a first glance, it seems that this can be accomplished by adapting the $\boldsymbol{\zeta}_i$ values accordingly. However, these are not only used for the otp encryption but are also included in the coefficients $\mathbf{y}[(i,j)]$ of the linear function for which $\mathsf{fh.dk}$ is computed. The latter poses the greatest challenge, since we need to ensure, that the overall evaluation still holds even after that modification.

Our solution is to embed all parts that contain $\boldsymbol{\zeta}_i$ values into a separate scheme, so that the security of this scheme guarantees that the modifications cannot be observed. The scheme we use is the OT-NMCFE scheme noot (NOisy One-Time) [SHZ25], that we recall in Figure 2. To reduce the indistinguishability to the security of noot, it is important to represent all parts that contain $\boldsymbol{\zeta}_i$ values by terms that are available to a noot-attacker.

For the appearance of $\boldsymbol{\zeta}_i$ in $\mathsf{otp.ct}_i$, this is simple as these ciphertexts have exactly the same structure as noot ciphertexts. More problematic are the following appearances of $\boldsymbol{\zeta}_i$ in the function for the fh scheme, highlighted in color:

$$\mathbf{y}[(i,j)] = \sum_{k\in[n],l\in[m]} \boldsymbol{c}[(k,l,i,j)]\boldsymbol{\zeta}_k[l] + \phi_{i,j} - \boldsymbol{c}[(0,0,i,j)]$$
$$\mathbf{y}[(0,0)] = \sum_{i\in[n],j\in[m]} \phi_{i,j}\boldsymbol{\zeta}_i[j] - \boldsymbol{c}[(0,0,0,0)] - \nu^0$$

None of the marked terms has a direct representation in the noot system. However, we observe that we can shift the highlighted terms from $\mathbf{y}[(i,j)]$ multiplied with $\mathbf{x}_i^0[j]$ to the constant terms $\mathbf{y}[(0,0)]$ without changing the view of the adversary. Now, if we consider the *sum* of all these highlighted terms, we get the following equation:

$$\begin{aligned}\mathbf{y}[(0,0)] &= \sum_{\substack{i,k\in[n],\\ j,l\in[m]}} \boldsymbol{c}[(k,l,i,j)]\boldsymbol{\zeta}_k[l]\mathbf{x}_i^0[j] + \sum_{\substack{i\in[n],\\ j\in[m]}} \phi_{i,j}\boldsymbol{\zeta}_i[j] - \boldsymbol{c}[(0,0,0,0)] - \nu^0 \\ &= \mathsf{noot.dk}[1] - \boldsymbol{c}[(0,0,0,0)] - \nu^0.\end{aligned}$$

Here, $\mathsf{noot.dk}[1]$ refers the the first part of the noot decryption key if the parameters are chosen accordingly. Particularly we choose as function coefficients for noot the vector $\boldsymbol{z} := (z_{1,1}, \dots z_{n,m})$, with $z_{i,j}$ defined as in Equation (10). This means we reached a situation where all $\boldsymbol{\zeta}_i$ terms appear in expressions that either coincide with a noot ciphertext or part of a noot decryption key. In particular, both are expressions that are available to a noot attacker.

Summing up, to switch the messages in $\mathsf{otp.ct}_i = \mathbf{x}_i^\beta + \boldsymbol{\zeta}_i$, we proceed in two steps:

1. Shift all $\boldsymbol{\zeta}_i$ terms in $\mathbf{y}[(i,j)]$ for $(i,j) \neq (0,0)$ to the constant term $\mathbf{y}[(0,0)]$ (transition $\mathsf{H}_1 \to \mathsf{H}_2$). Indistinguishability follows from the FH property (cf. Lemma 3).
2. Change the messages in $\mathsf{otp.ct}_i = \mathbf{x}_i^\beta + \boldsymbol{\zeta}_i$ by considering these expressions as noot encryptions (transition $\mathsf{H}_2 \to \mathsf{H}_3$). Indistinguishability follows from the security of noot (cf. Lemma 4).

$\mathsf{noot.Setup}(1^\lambda, \mathcal{F}^{\mathtt{IP}}_{n,\{m\}})$

$\mathsf{ek}_i \leftarrow \{0,1\}^\lambda, \ \forall i \in [n]$

$\mathsf{msk} := (\{\mathsf{ek}_i\}_{i\in[n]})$

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

$\mathsf{noot.KeyGen}(\mathsf{msk}, \boldsymbol{c}, \ell, \mathbb{D})$

$\boldsymbol{c}_i := (\boldsymbol{c}[(i,1)], \dots, \boldsymbol{c}[(i,m)])$

$\boldsymbol{\zeta}_i = \mathsf{PRF}(\mathsf{ek}_i, \ell), \ \forall i \in [n],$

$\nu \xleftarrow{\mathbb{D}} \Delta$

return $\mathsf{dk} = \left(\sum_{i\in[n]} \langle \boldsymbol{\zeta}_i, \boldsymbol{c}_i \rangle - \nu, \boldsymbol{c}\right)$

$\mathsf{noot.Enc}(\mathsf{ek}_i, i, \mathbf{x}_i, \ell)$

$\boldsymbol{\zeta}_i = \mathsf{PRF}(\mathsf{ek}_i, \ell)$

return $\mathsf{ct} = \mathbf{x}_i + \boldsymbol{\zeta}_i$

$\mathsf{noot.Dec}(\mathsf{dk}, \mathsf{ct}_1, \dots, \mathsf{ct}_n)$

$\boldsymbol{c}_i := (\boldsymbol{c}[(i,1)], \dots, \boldsymbol{c}[(i,m)])$

return $\sum_{i\in[n]} \langle \mathsf{ct}_i, \boldsymbol{c}_i \rangle - \mathsf{dk}[1]$

Figure 2: One-time NMCFE scheme noot [SHZ25] operating over $\mathbb{Z}_q$.

$\mathsf{H}_3 \to \mathsf{H}_4 \to \mathsf{G}_1$:

The last step is to change the ciphertexts of fh to encryptions of $\mathbf{x}_i^1$ (instead of $\mathbf{x}_i^0$) and build the $\mathbf{y}[(i,j)]$ as before. This is a simple reduction to the

FH security of fh. The equivalence of H_4 and G_1 follows as $\phi + d$, with d the difference vector introduced in H_1, remains truly random for any $\phi \xleftarrow{\$} \mathbb{Z}_q$.

5.2 Detailed Proofs

In the remainder of the section, we give the detailed proofs. The security of our construction is shown by the following theorem:

Theorem 1. *For any PPT adversary $\mathcal{A}$ against the* nh-ot*-security of our construction as in Definition 3, there exist PPT adversaries $\mathcal{B}_1$, $\mathcal{B}_2$ and $\mathcal{B}_3$ against the* nh-ot*-security of* nh *and* noot *and* fh-ot*-security of* fh*, respectively, such that*

$$\mathit{Adv}_{\mathcal{A}}(\lambda) \leq \mathit{Adv}^{\mathsf{nh}}_{\mathcal{B}_1}(\lambda) + \mathit{Adv}^{\mathsf{noot}}_{\mathcal{B}_2}(\lambda) + 2\mathit{Adv}^{\mathsf{fh}}_{\mathcal{B}_3}(\lambda).$$

We conduct the claim through a series of hybrid games H_0-H_5. Throughout the rest of this section, we denote by $\mathsf{Adv}^{\mathsf{H}_{i-1} \leftrightarrow \mathsf{H}_i}_{\mathcal{A}}(\lambda)$ the advantage of $\mathcal{A}$ to distinguish games H_{i-1} and H_i for $i \in [5]$. H_0 refers to G_0, i. e., the experiment $\mathsf{Expt}^{\mathsf{NMCFE}}_0$ and H_5 to G_1, i. e., the experiment $\mathsf{Expt}^{\mathsf{NMCFE}}_1$. Figures 3 to 6 display the remaining hybrid games, where the changes to the previous game are highlighted. We only consider changes made in the key generation oracle $\mathcal{O}.\mathsf{KeyGen}$ and the encryption process, which we denote through an encryption oracle $\mathcal{O}.\mathsf{Enc}(i, \mathbf{x}^0_i, \mathbf{x}^1_i, \ell)$. In each game we perform all changes for all slots $i \in [n]$ and all labels $\ell \in L_{\mathsf{req}}$, therefore we omit ℓ whenever it is clear from the context. We identify the functions evaluated by nh, i. e., functions described by $\boldsymbol{c}[(i,j,1,1)], \ldots, \boldsymbol{c}[(i,j,n,m)]$ with $f_{i,j}$. We further abbreviate $\mathbf{x}^\beta := (\mathbf{x}^\beta_1 || \ldots || \mathbf{x}^\beta_n)$ for $\beta \in \{0,1\}$ and without loss of generality assume the corrupted clients occupy the first slots and denote by $\mathcal{HS} := [|CS| + 1, n]$ the set of honest clients.

Lemmas 2 - 5 bound the advantage in each game. Summing up over all other advantages yields Theorem 1.

Lemma 2. *For any PPT adversary $\mathcal{A}$, there exists a PPT adversary $\mathcal{B}$ such that $\mathit{Adv}^{\mathsf{H}_0 \leftrightarrow \mathsf{H}_1}_{\mathcal{A}}(\lambda) \leq \mathit{Adv}^{\mathsf{nh}}_{\mathcal{B}}(\lambda)$.*

Proof. The difference between both games is the switch from $\mathbf{x}^0_i$ to $\mathbf{x}^1_i$ in the ciphertexts of the nh scheme, i. e., the advantage in distinguishing both games is bounded by an adversary $\mathcal{B}$ against nh. By the admissibility of $\mathcal{A}$, $\mathcal{B}$ is admissible and by construction, it holds that $f_{i,j}(\mathbf{x}^1) + \phi_{i,j} + d_{i,j} = f_{i,j}(\mathbf{x}^0) + \phi_{i,j}$ as $d_{i,j} = f_{i,j}(\mathbf{x}^0) - f_{i,j}(\mathbf{x}^1)$. Since nh is noise-hiding, the values $d_{i,j}$ remain hidden. □

$\mathcal{O}.\mathsf{Enc}(i, \mathbf{x}_i^0, \mathbf{x}_i^1, \ell)$

$\mathsf{nh.ct}_i \leftarrow \mathsf{nh.Enc}(\mathsf{nh.ek}_i, i, \mathbf{x}_i^1, \ell)$

$\boldsymbol{\zeta}_i \leftarrow \mathsf{PRF}(K_i, \ell), \quad \mathsf{otp.ct}_i = \mathbf{x}_i^0 + \boldsymbol{\zeta}_i$

$\mathsf{fh.ct}_i \leftarrow \mathsf{fh.Enc}(\mathsf{fh.ek}_i, i, \mathbf{x}_i^0, \ell),$

return $\mathsf{ct}_i = (\mathsf{nh.ct}_i, \mathsf{otp.ct}_i, \mathsf{fh.ct}_i)$

$\mathcal{O}.\mathsf{KeyGen}(f, \nu^0, \nu^1 \ell)$

$\forall i \in [n], j \in [m] : \phi_{i,j} \xleftarrow{\$} \mathbb{Z}_q, \; d_{i,j} = f_{i,j}(\mathbf{x}^0) - f_{i,j}(\mathbf{x}^1)$

$\mathbf{c}_{i,j} = (\boldsymbol{c}[(i,j,1,1)], \ldots, \boldsymbol{c}[(i,j,n,m)])$

$\mathsf{nh.dk}_{i,j} \leftarrow \mathsf{nh.KeyGen}(\mathsf{nh.msk}, \mathbf{c}_{i,j}, \ell, \mathbb{D}_{\phi_{i,j}+d_{i,j}}),$

$\boldsymbol{\zeta}_i \leftarrow \mathsf{PRF}(K_i, \ell),$

$$\mathbf{y}[(i,j)] = \sum_{k\in[n], l\in[m]} \boldsymbol{c}[(k,l,i,j)]\boldsymbol{\zeta}_k[l] + \phi_{i,j} - \boldsymbol{c}[(0,0,i,j)]$$

$$\mathbf{y}[(0,0)] = \sum_{i\in[n], j\in[m]} \phi_{i,j}\boldsymbol{\zeta}_i[j] - \boldsymbol{c}[(0,0,0,0)] - \nu^0$$

$\mathsf{fh.dk} \leftarrow \mathsf{fh.KeyGen}(\mathsf{fh.msk}, \mathbf{y}, \ell)$

return $\mathsf{dk} = (\{\mathsf{nh.dk}_{i,j}\}, \mathsf{fh.dk})$

Figure 3: Hybrid game H_1. Changes compared to G_0 are displayed in color.

Lemma 3. *For any PPT adversary $\mathcal{A}$, there exists a PPT adversary $\mathcal{B}$ such that $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{H}_1 \leftrightarrow \mathsf{H}_2}(\lambda) \leq \mathsf{Adv}_{\mathcal{B}}^{\mathsf{fh}}(\lambda)$.*

Proof. The only difference between the two games lies within the construction of $\mathsf{fh.KeyGen}$. Thus, we only have to ensure that $\mathcal{B}$ is admissible with respect to fh. In both games, we have

$$\begin{aligned}\mathsf{fh.Dec}(\mathsf{fh.dk}, \mathsf{fh.ct}_1, \ldots, \mathsf{fh.ct}_n) = &\sum_{\substack{i\in[n],\\ j\in[m]}} (\mathbf{y}[(i,j)]\mathbf{x}_i^0[j] + \phi_{i,j}\boldsymbol{\zeta}_i[j]) \\ &- \boldsymbol{c}[(0,0,0,0)] - \nu^0\end{aligned}$$

Moreover, as fh supports corruptions, $\mathcal{B}$ can obtain the encryption keys of corrupted clients. For these corrupted slots, $\mathcal{B}$ remains admissible with respect to the ciphertext queries through $\mathcal{A}$'s queries and with respect to the leakage of the function queries, as for all $i \in \mathcal{CS}, j \in [m]$, $\mathbf{y}[(i,j)]$ is defined equally in both games, H_1 and H_2 . □

$\mathcal{O}.\mathsf{Enc}(i, \mathbf{x}_i^0, \mathbf{x}_i^1, \ell)$

$\mathsf{nh.ct}_i \leftarrow \mathsf{nh.Enc}(\mathsf{nh.ek}_i, i, \mathbf{x}_i^1, \ell)$

$\boldsymbol{\zeta}_i \leftarrow \mathsf{PRF}(K_i, \ell), \quad \mathsf{otp.ct}_i = \mathbf{x}_i^0 + \boldsymbol{\zeta}_i$

$\mathsf{fh.ct}_i \leftarrow \mathsf{fh.Enc}(\mathsf{fh.ek}_i, i, \mathbf{x}_i^0, \ell),$

return $\mathsf{ct}_i = (\mathsf{nh.ct}_i, \mathsf{otp.ct}_i, \mathsf{fh.ct}_i)$

$\mathcal{O}.\mathsf{KeyGen}(f, \nu^0, \nu^1 \ell)$

$\forall i \in [n], j \in [m] : \phi_{i,j} \xleftarrow{\$} \mathbb{Z}_q, \; d_{i,j} = f_{i,j}(\mathbf{x}^0) - f_{i,j}(\mathbf{x}^1)$

$\mathsf{nh.dk}_{i,j} \leftarrow \mathsf{nh.KeyGen}(\mathsf{nh.msk}, (\boldsymbol{c}[(i,j,1,1)], \ldots, \boldsymbol{c}[(i,j,n,m)]), \ell, \mathbb{D}_{\phi_{i,j}+d_{i,j}}),$

$\boldsymbol{\zeta}_i \leftarrow \mathsf{PRF}(K_i, \ell),$

$\forall i \in \mathcal{CS} :$

$$\mathbf{y}[(i,j)] = \sum_{k\in[n],l\in[m]} \boldsymbol{c}[(k,l,i,j)]\boldsymbol{\zeta}_k[l] + \phi_{i,j} - \boldsymbol{c}[(0,0,i,j)]$$

$$\forall i \in \mathcal{HS} : \; \mathbf{y}[(i,j)] = \phi_{i,j} - \boldsymbol{c}[(0,0,i,j)]$$

$$\alpha = \nu^0 - \sum_{i\in\mathcal{HS},k\in[n],j,l\in[m]} \boldsymbol{c}[(k,l,i,j)]\boldsymbol{\zeta}_k[l]\mathbf{x}_i^0[j]$$

$$\mathbf{y}[(0,0)] = \sum_{i\in[n],j\in[m]} \phi_{i,j}\boldsymbol{\zeta}_i[j] - \boldsymbol{c}[(0,0,0,0)] - \alpha$$

$\mathsf{fh.dk} \leftarrow \mathsf{fh.KeyGen}(\mathsf{fh.msk}, \mathbf{y}, \ell)$

return $\mathsf{dk} = (\{\mathsf{nh.dk}_{i,j}\}, \mathsf{fh.dk})$

Figure 4: Hybrid game H_2. Changes compared to H_1 are displayed in color.

Lemma 4. *For any PPT adversary $\mathcal{A}$, there exists a PPT adversary $\mathcal{B}$ such that $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{H}_2 \leftrightarrow \mathsf{H}_3}(\lambda) \leq \mathsf{Adv}_{\mathcal{B}}^{\mathsf{noot}}(\lambda)$.*

Proof. Since noot is robust under static corruptions and allows labeled encryptions it remains to show, that $\mathcal{B}$ is admissible and can simulate $\mathcal{A}$'s view.

Upon interaction, $\mathcal{B}$ receives the set of queried messages $\mathcal{MS} = \{(\mathbf{x}_{i,\ell}^0, \mathbf{x}_{i,\ell}^1, i, \ell)\}_{i\in[n],\ell\in L_{\mathsf{req}}\subseteq\mathcal{L}}$ and the set of corrupted clients $\mathcal{CS}$ from $\mathcal{A}$. Both are forwarded to $\mathcal{B}$'s own oracle to obtain the ciphertexts $\mathcal{C} := \{\mathsf{noot.ct}_{i,\ell} = \mathbf{x}_{i,\ell}^{\beta} + \mathsf{PRF}(K_i, \ell)\}_{i\in[n],\ell\in L_{\mathsf{req}}\subseteq\mathcal{L}}$, which can be seen as $\mathsf{otp.ct}_i$, and the corrupted secret keys $\{\mathsf{noot.ek}_i = K_i\}_{i\in[\mathcal{CS}]}$, which serve as

$\mathcal{O}.\mathsf{Enc}(i, \mathbf{x}_i^0, \mathbf{x}_i^1, \ell)$

$\mathsf{nh.ct}_i \leftarrow \mathsf{nh.Enc}(\mathsf{nh.ek}_i, i, \mathbf{x}_i^1, \ell)$

$\boldsymbol{\zeta}_i \leftarrow \mathsf{PRF}(K_i, \ell), \quad \mathsf{otp.ct}_i = \mathbf{x}_i^1 + \boldsymbol{\zeta}_i$

$\mathsf{fh.ct}_i \leftarrow \mathsf{fh.Enc}(\mathsf{fh.ek}_i, i, \mathbf{x}_i^0, \ell),$

return $\mathsf{ct}_i = (\mathsf{nh.ct}_i, \mathsf{otp.ct}_i, \mathsf{fh.ct}_i)$

$\mathcal{O}.\mathsf{KeyGen}(f, \nu^0, \nu^1 \ell)$

$\forall i \in [n], j \in [m] : \phi_{i,j} \xleftarrow{\$} \mathbb{Z}_q, \ d_{i,j} = f_{i,j}(\mathbf{x}^0) - f_{i,j}(\mathbf{x}^1)$

$\mathsf{nh.dk}_{i,j} \leftarrow \mathsf{nh.KeyGen}(\mathsf{nh.msk}, (\boldsymbol{c}[(i,j,1,1)], \ldots, \boldsymbol{c}[(i,j,n,m)]), \ell, \mathbb{D}_{\phi_{i,j}+d_{i,j}}),$

$\boldsymbol{\zeta}_i \leftarrow \mathsf{PRF}(K_i, \ell),$

$\forall i \in \mathcal{CS} :$

$$\mathbf{y}[(i,j)] = \sum_{k\in[n],l\in[m]} \boldsymbol{c}[(k,l,i,j)]\boldsymbol{\zeta}_k[l] + \phi_{i,j} - \boldsymbol{c}[(0,0,i,j)]$$

$$\forall i \in \mathcal{HS} : \ \mathbf{y}[(i,j)] = \phi_{i,j} - \boldsymbol{c}[(0,0,i,j)]$$

$$\alpha = \nu^0 - \sum_{i\in\mathcal{HS},k\in[n],j,l\in[m]} \boldsymbol{c}[(k,l,i,j)]\boldsymbol{\zeta}_k[l]\mathbf{x}_i^0[j] - \sum_{i\in[n],j\in[m]} (\mathbf{x}_i^1[j] - \mathbf{x}_i^0[j])z_{i,j}$$

$$\mathbf{y}[(0,0)] = \sum_{i\in[n],j\in[m]} \phi_{i,j}\boldsymbol{\zeta}_i[j] - \boldsymbol{c}[(0,0,0,0)] - \alpha$$

$\mathsf{fh.dk} \leftarrow \mathsf{fh.KeyGen}(\mathsf{fh.msk}, \mathbf{y}, \ell)$

return $\mathsf{dk} = (\{\mathsf{nh.dk}_{i,j}\}, \mathsf{fh.dk})$

Figure 5: Hybrid game H_3. Changes compared to H_2 are displayed in color.

the $\{K_i\}_{i\in\mathcal{CS}}$ sampled during setup.

$\mathcal{B}$ runs the setup algorithm for nh and fh and hands $(\mathsf{pp} := (\mathsf{nh.pp}, \mathsf{fh.pp}), \mathcal{C}, \{\mathsf{ek}_i := (\mathsf{nh.ek}_i, \mathsf{noot.ek}_i, \mathsf{fh.ek}_i)\}_{i\in\mathcal{CS}})$ to $\mathcal{A}$.

For each decryption key query $(\boldsymbol{c}, \nu^0, \nu^1, \ell)$, $\mathcal{B}$ samples $\phi_{i,j} \xleftarrow{\$} \mathbb{Z}_q$ and runs $\mathsf{nh.dk}_{i,j} \leftarrow \mathsf{nh.KeyGen}(\mathsf{nh.msk}, f_{i,j}, \mathbb{D}_{\phi_{i,j}+d_{i,j}}), \forall i \in [n], j \in [m]$. Then, for all $i \in [n]$, $\mathcal{B}$ computes

$$z_{i,j} = \sum_{k\in[n],l\in[m]} \boldsymbol{c}[(i,j,k,l)]\mathbf{x}_k^0[l] + \phi_{i,j}$$

and sets $\hat{\nu}^0 = 0$, $\hat{\nu}^1 = \sum_{i\in[n],j\in[m]}(\mathbf{x}_i^0[j] - \mathbf{x}_i^1[j])z_{i,j}$. $\mathcal{B}$ queries their own oracle for $(\boldsymbol{z} = (z_{1,1}, \ldots, z_{1,m}, \ldots, z_{n,m}), \hat{\nu}^0, \hat{\nu}^1)$ to obtain $\mathsf{noot.dk}$. Then, $\mathcal{B}$

computes

$$\mathbf{y}[(i,j)] = \begin{cases} \sum_{k\in[n],l\in[m]} \boldsymbol{c}[(k,l,i,j)]\boldsymbol{\zeta}_k[l] + \phi_{i,j} - \boldsymbol{c}[(0,0,i,j)], & i \in \mathcal{CS} \\ \phi_{i,j} - \boldsymbol{c}[(0,0,i,j)], & i \in \mathcal{HS} \end{cases}$$

using that if $k \in \mathcal{CS}$, $\mathsf{noot.ek}_k = \boldsymbol{\zeta}_k$ and if $k \in \mathcal{HS}, i \in \mathcal{CS}$, by Remark 4 and the uniqueness property it holds that $\boldsymbol{c}[(k,l,i,j)] = 0$. Finally, $\mathcal{B}$ calculates the constant value as

$$\mathbf{y}[(0,0)] = \mathsf{noot.dk}[1] - \sum_{i\in\mathcal{CS},j\in[m]} \boldsymbol{c}[(k,l,i,j)]\boldsymbol{\zeta}_k[l]\mathbf{x}_i^0[j] - \boldsymbol{c}[(0,0,0,0)] - \nu^0,$$

generates $\mathsf{fh.dk} \leftarrow \mathsf{fh.KeyGen}(\mathsf{fh.msk}, \mathbf{y})$ and returns $\mathsf{dk} = (\{\mathsf{nh.dk}_{i,j}\}, \mathsf{fh.dk})$.

In the end, they output the same bit as $\mathcal{A}$. Note that $\mathcal{B}$ perfectly simulates $\mathcal{A}$'s view as

$$\mathsf{noot.dk}[1] = \sum_{i\in[n],j\in[m]} \boldsymbol{\zeta}_i[j]\Big(\sum_{k\in[n],l\in[m]} \boldsymbol{c}[(i,j,k,l)]\mathbf{x}_k^0[l] + \phi_{i,j}\Big) + \hat{\nu}.$$

Due to $Q(\mathbf{x}^0) + \nu^0 = Q(\mathbf{x}^1) + \nu^1$, the value of $\mathbf{y}[(0,0)]$ is exactly as in H_2 and H_3, respectively.

$\mathcal{B}$'s own admissibility follows by the admissibility of $\mathcal{A}$, as $\mathcal{B}$'s chosen noise value is exactly the difference between evaluation $\boldsymbol{z}$ on $\mathbf{x}^0$ versus evaluation $\boldsymbol{z}$ on $\mathbf{x}^1$. □

Lemma 5. *For any PPT adversary $\mathcal{A}$, there exists a PPT adversary $\mathcal{B}$ such that $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{H}_3\leftrightarrow\mathsf{H}_4}(\lambda) \leq \mathsf{Adv}_{\mathcal{B}}^{\mathsf{fh}}(\lambda)$.*

Proof. The difference between both games is the change in the encryption algorithm of fh from $\mathbf{x}_i^0$ to $\mathbf{x}_i^1$ and how the values inside the $\mathsf{fh.dk}$ are distributed through the $\mathbf{y}$ coefficients. Hence, if the output of fh is the same within both games, $\mathcal{B}$ can perfectly simulate $\mathcal{A}$'s view. This is given through the admissibility of $\mathcal{A}$, as it holds that $Q(\mathbf{x}^0) + \nu^0 = Q(\mathbf{x}^1) + \nu^1$, where Q is the function described by $\boldsymbol{c}$ and the fact that $d_{i,j} = 0$ if $i \in \mathcal{CS}$ by the imposed restrictions on monomials of quadratic functions in Remark 4. □

Lemma 6. *For any PPT adversary $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{H}_4\leftrightarrow\mathsf{H}_5}(\lambda) = 0$.*

Proof. Note that the only difference between H_4 and H_5 is that in H_4, $\phi_{i,j}^k + d_{i,j}^k$ is used as a random pad for $f_{i,j}^k$ for all $i \in [n], j \in [m], k \in [q_\ell], \ell \in L_{\mathsf{req}}$, whereas in H_5, this random pad $\tilde{\phi}_{i,j}^k$ is uniformly sampled. However, $\phi_{i,j}^k + d_{i,j}^k$ and $\tilde{\phi}_{i,j}^k$ share the same uniform distribution due to $\phi_{i,j}^k$ being uniformly sampled. □

$\mathcal{O}.\mathsf{Enc}(i, \mathbf{x}_i^0, \mathbf{x}_i^1, \ell)$

$\mathsf{nh.ct}_i \leftarrow \mathsf{nh.Enc}(\mathsf{nh.ek}_i, i, \mathbf{x}_i^1, \ell)$

$\boldsymbol{\zeta}_i \leftarrow \mathsf{PRF}(K_i, \ell), \quad \mathsf{otp.ct}_i = \mathbf{x}_i^1 + \boldsymbol{\zeta}_i$

$\mathsf{fh.ct}_i \leftarrow \mathsf{fh.Enc}(\mathsf{fh.ek}_i, i, \mathbf{x}_i^1, \ell)$

return $\mathsf{ct}_i = (\mathsf{nh.ct}_i, \mathsf{otp.ct}_i, \mathsf{fh.ct}_i)$

$\mathcal{O}.\mathsf{KeyGen}(f, \nu^0, \nu^1 \ell)$

$\forall i \in [n], j \in [m] : \phi_{i,j} \xleftarrow{\$} \mathbb{Z}_q, \; d_{i,j} = f_{i,j}(\mathbf{x}^0) - f_{i,j}(\mathbf{x}^1)$

$\mathsf{nh.dk}_{i,j} \leftarrow \mathsf{nh.KeyGen}(\mathsf{nh.msk}, (\boldsymbol{c}[(i,j,1,1)], \ldots, \boldsymbol{c}[(i,j,n,m)]), \ell, \mathbb{D}_{\phi_{i,j}+d_{i,j}}),$

$\boldsymbol{\zeta}_i \leftarrow \mathsf{PRF}(K_i, \ell),$

$\forall i \in [n] :$

$$\mathbf{y}[(i,j)] = \sum_{k \in [n], l \in [m]} \boldsymbol{c}[(k,l,i,j)] \boldsymbol{\zeta}_k[l] + \phi_{i,j} + d_{i,j} - \boldsymbol{c}[(0,0,i,j)]$$

$\alpha = \nu^1$

$$\mathbf{y}[(0,0)] = \sum_{i \in [n], j \in [m]} \phi_{i,j} \boldsymbol{\zeta}_i[j] - \boldsymbol{c}[(0,0,0,0)] - \alpha$$

$\mathsf{fh.dk} \leftarrow \mathsf{fh.KeyGen}(\mathsf{fh.msk}, \mathbf{y}, \ell)$

return $\mathsf{dk} = (\{\mathsf{nh.dk}_{i,j}\}, \mathsf{fh.dk})$

Figure 6: Hybrid game H_4. Changes compared to H_3 are displayed in color.

6 Benchmarking

In the following section we compare the performance of our scheme QUILT to concurrent schemes. We discuss our primary results below, while further details can be found in Section B. All of the experiments are performed on a virtual machine with access to 425 GB of DDR-5 memory and 22 CPU threads of an AMD EPYC 9254 and are implemented on top of the *gofe* library [MSHB+19]. To the best of our knowledge, as stated in Section 2, there exist only two comparable schemes:

1. The (N)MIFE scheme of Agrawal et al. [AGT22], improved by [ZSHA25], dubbed AGT, which allows for multiple encrypted values per ciphertext slot.

2. The NMIFE scheme of Zalonis et al. [ZSHA25], dubbed ZSHA, that adapts [AGT22] and supports OT encryption similar to us.

In their work, the authors of [ZSHA25] show that their scheme ZSHA is significantly faster than AGT. Even for very small datasets, ZSHA only takes seconds while AGT does not even terminate. In Section 1.2 Table 2, we recall the results of some of the experiments reported there and compare the runtimes with QUILT. These already indicate that QUILT outperforms the previous schemes by far.

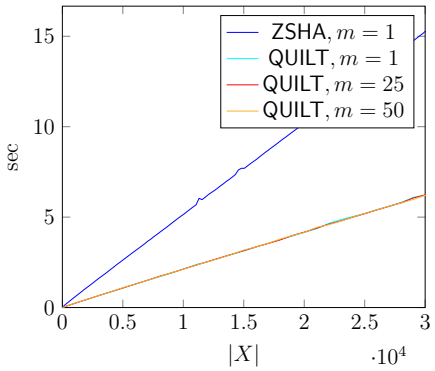
In the following, we present the results of extensive experiments, as well as a logistic regression training with QUILT, and compare these to ZSHA, which is the fastest scheme so far. The three building blocks listed at the beginning of Section 4 are implemented as follows:

1. For the OT-NMCFE scheme nh, we use an instance of noot [SHZ25], being recalled in Figure 2.
2. For the PRF, we make use of AES-256 as implemented in the Go crypto library [Aut24].
3. For the OT-FH-MCFE scheme fh, we use an extension of [DOT18]. This extension supports affine functionality and labeled keys and is presented in Section A.

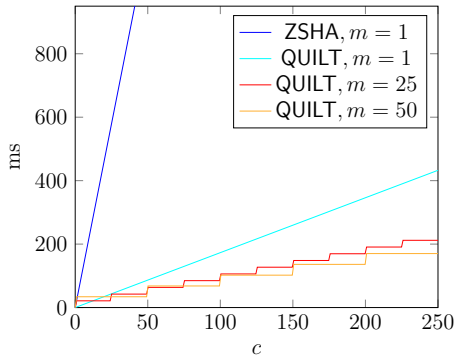
By comparison, ZSHA uses as building blocks an instance of [TAO20] and an instance of [DOT18]. We stress that both constructions, ZSHA and QUILT, are generic. Thus, their overall efficiency mainly depends on the efficiency of the underlying building blocks. As discussed in Section 2, we expect a significant efficiency gain, which is confirmed by our following experimental results.

6.1 Basic Operations

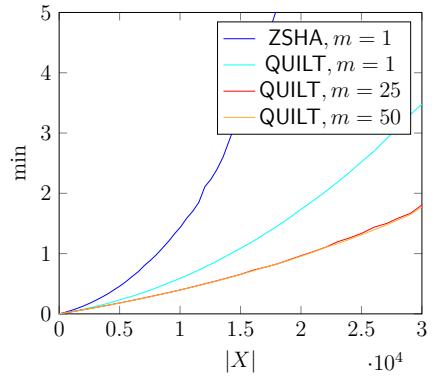
For the performance evaluation we consider datasets X consisting of c columns and r rows and, for comparability, we include an additional one-hot vector in the case of ZSHA. As explained in Remark 3, this is necessary for ZSHA to support any form of quadratic functions, i. e., functions that do not only contain quadratic monomials. As both schemes, ZSHA and QUILT, are multi-input schemes that allow for n inputs, each with vector length m, a natural setting would be $n = r$ and $m = c$. However, other choices of n and m are possible as long as $n \cdot m \geq r \cdot c$. For instance, ZSHA performs best if m is set to be as small as possible. Hence, another choice is to set $m = 1$ and consequently $n = r \cdot c = |X|$. In contrast, for our scheme m does not need to be small. Thus, in the following we compare our scheme for different choices for m with ZSHA with $m = 1$. Note that



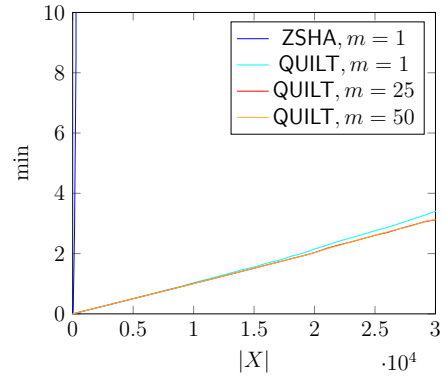
(a) Setup in seconds.



(b) Runtime for encrypting one row with c attributes in milliseconds for $|X| = 250$.



(c) Decryption key generation in minutes.



(d) Decryption in minutes.

Figure 7: Runtime of the different algorithms for ZSHA and QUILT.

setting m to 1 yields a maximally split dataset, which could be relevant in some use cases using highly distributed datasets.

For the benchmarking, we use random datasets with $\mathsf{B}_X, \mathsf{B}_C = 128$[5] and random quadratic functions Q. That is, both the entries $\mathbf{x}_i$ of X and the coefficients of Q are randomly sampled. Figure 7 shows the runtime of the different algorithms for ZSHA and QUILT. Figures 7a and 7c-7d show the runtimes for Setup, KeyGen and Dec of both schemes in dependency of the dataset size $|X| = rc$, fixing m to a predefined value and setting

[5] Note that the bounds are only relevant for the range in that the discrete logarithm is searched. Since this step is omitted in the basic benchmarking, the bounds do not have an effect on the runtime.

$n = \lceil \frac{c}{m} \rceil \cdot r$, if $c > m$, else $n = r$. Note that in the case of Figure 7a, the effort of QUILT is practically the same for the different choices of m. This is the reason why seemingly only one line for QUILT is shown.

Figure 7b compares the encryption process, which differs from the other tasks as it is distributed among the different clients. Hence, instead of measuring the runtime for encrypting the complete database, we compare the computation times for encrypting one row, containing c attributes, as this typically represents one record in various use cases such as PPML. More precisely, if $m \geq c$ this means running the encryption process once, for $m < c$, the encryption process needs to be performed $\lceil \frac{c}{m} \rceil$ times. We compare the encryption process in Figure 7b.

In Figure 7d we compare the decryption process of both schemes. ZSHA takes significantly longer than QUILT even for small input sizes; therefore we include a zoomed-in section of Figure 7d in Section B to allow for a clearer comparison of the curves. Additionally ZSHA also performs best for sparse polynomials, however QUILT is independent on the number of non-zero coefficients, and outperforms it even in the case of polynomials with only one non-zero coefficient, as presented in Figure 11. Similar to Figure 7a, there are no observable differences for QUILT for different choices of m.

Note that both schemes are based on pairings and require as final step the computation of the discrete logarithm, which represents a bottleneck. We stress that the time required for computing the discrete logarithm is *not* contained in the figures. The reason is that the time effort strongly varies as it depends on the result of the function evaluation and, more important, is exactly the same for both schemes. Thus, while the effort would certainly matter when evaluating the total efficiency of the schemes, it is not relevant for comparing their efforts. However in Table 2 and our final experiment, the logistic regression training, we present runtimes including this search step.

Our experiments show that in all cases, QUILT outperforms ZSHA by far. This is further supported by our last evaluation, the logistic regression training.

6.2 Private Logistic Regression

As final experiment we train a private logistic regression according to the protocol proposed in [ZSHA25] that we shortly recall in Section C. To be comparable to the evaluations of [ZSHA25] we consider a maximally split dataset, i. e., $m = 1$. We normalize the real-world data to the range

Table 3: Runtime in minutes for a logistic regression training with 25 iterations, with a scaling factor $\mathbf{s} = 10^4$ for converting real-world data to inputs in $\mathbb{Z}_q$.

Data Set	#rec	#attr	Runtime in min	
			ZSHA	QUILT
LBW[Log24a]	189	9	72	15
PCS [Log24b]	380	8	144	21
UIS [Log24c]	575	8	186	28

$[0, 1]$, conservatively assuming model weights in the range $[-10, 10]$ and apply fix-point arithmetic with a scaling factor of 10^4, resulting in bounds of $\mathsf{B}_X = 10^4$ and $\mathsf{B}_C = 10^5$. The training is done over various medical datasets with different sizes, where #rec describes the number of records and #attr the number of attributes, including the classification variable. Since the NMCFE scheme does not modify the output, the model achieves similar utility to [ZSHA25], measured as accuracy, i. e., the proportion of correctly classified records over the total number of test examples, as shown in Figure 8.

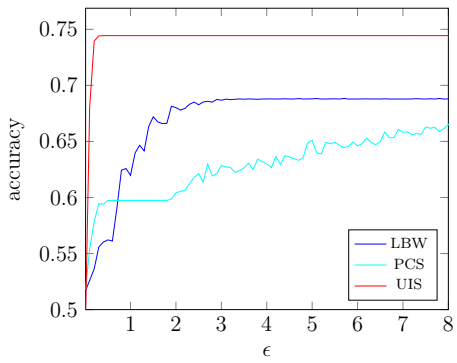


Figure 8: Utility for trained logistic regression models with $\texttt{it} = 25$ and $\delta = 1/n$.

The efficiency gains demonstrated in earlier experiments are also reflected in the model training, which is shown in Table 3, achieving a 4.8x to 6.8x speed-up compared to ZSHA.

Summing up, one can see that QUILT is currently the fastest scheme by far.

References

[ABG19] Michel Abdalla, Fabrice Benhamouda, and Romain Gay. „From single-input to multi-client inner-product functional encryption.“ In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2019, pp. 552–582 (cit. on p. 2).

[ABKW19] Michel Abdalla, Fabrice Benhamouda, Markulf Kohlweiss, and Hendrik Waldner. „Decentralizing inner-product functional encryption.“ In: *IACR International Workshop on Public Key Cryptography*. Springer. 2019, pp. 128–157 (cit. on p. 2).

[ACFG+18] Michel Abdalla, Dario Catalano, Dario Fiore, Romain Gay, and Bogdan Ursu. „Multi-input functional encryption for inner products: Function-hiding realizations and constructions without pairings.“ In: *Advances in Cryptology–CRYPTO 2018: 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19–23, 2018, Proceedings, Part I 38*. Springer. 2018, pp. 597–627 (cit. on p. 2).

[Agr19] Shweta Agrawal. „Indistinguishability Obfuscation Without Multilinear Maps: New Methods for Bootstrapping and Instantiation.“ In: *EUROCRYPT 2019, Part I*. Ed. by Yuval Ishai and Vincent Rijmen. Vol. 11476. LNCS. Springer, Heidelberg, May 2019, pp. 191–225. DOI: 10.1007/978-3-030-17653-2_7 (cit. on p. 3).

[AGRW17] Michel Abdalla, Romain Gay, Mariana Raykova, and Hoeteck Wee. „Multi-input inner-product functional encryption from pairings.“ In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2017, pp. 601–626 (cit. on p. 2).

[AGT21] Shweta Agrawal, Rishab Goyal, and Junichi Tomida. „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. 2021, pp. 208–238 (cit. on pp. 2, 3, 8).

[AGT22] Shweta Agrawal, Rishab Goyal, and Junichi Tomida. „Multi-input quadratic functional encryption: Stronger security, broader functionality.“ In: *Theory of Cryptography Conference*. Springer. 2022, pp. 711–740 (cit. on pp. 2, 3, 5, 7, 8, 9, 28, 29).

[AJ15] Prabhanjan Ananth and Abhishek Jain. „Indistinguishability obfuscation from compact functional encryption.“ In: *Annual Cryptology Conference*. Springer. 2015, pp. 308–326 (cit. on p. 2).

[ALS16] Shweta Agrawal, Benoît Libert, and Damien Stehle. „Fully secure functional encryption for linear functions from standard assumptions.“ In: *CRYPTO*. 2016, pp. 333–362 (cit. on p. 11).

[Aut24] The Go Authors. *crypto/aes package*. Version go1.23.2. 2024. URL: https://pkg.go.dev/crypto/aes@go1.23.2 (cit. on p. 29).

[AV19] Prabhanjan Ananth and Vinod Vaikuntanathan. „Optimal bounded-collusion secure functional encryption.“ In: *Theory of Cryptography Conference*. Springer. 2019, pp. 174–198 (cit. on p. 4).

[BCP14] Elette Boyle, Kai-Min Chung, and Rafael Pass. „On extractability obfuscation.“ In: *Theory of cryptography conference*. Springer. 2014, pp. 52–73 (cit. on p. 2).

[BH08] Dan Boneh and Michael Hamburg. „Generalized identity based and broadcast encryption schemes.“ In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2008, pp. 455–470 (cit. on p. 2).

[BSS05] Joonsang Baek, Reihaneh Safavi-Naini, and Willy Susilo. „Efficient multi-receiver identity-based encryption and its application to broadcast encryption.“ In: *Public Key Cryptography-PKC 2005: 8th International Workshop on Theory and Practice in Public Key Cryptography, Les Diablerets, Switzerland, January 23-26, 2005. Proceedings 8*. Springer. 2005, pp. 380–397 (cit. on p. 2).

[BSW07] John Bethencourt, Amit Sahai, and Brent Waters. „Ciphertext-policy attribute-based encryption.“ In: *2007 IEEE symposium on security and privacy (SP'07)*. IEEE. 2007, pp. 321–334 (cit. on p. 2).

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. „Functional encryption: Definitions and challenges.“ In: *Theory of Cryptography: 8th Theory of Cryptography Conference, TCC 2011, Providence, RI, USA, March 28-30, 2011. Proceedings 8*. Springer. 2011, pp. 253–273 (cit. on p. 2).

[BV18] Nir Bitansky and Vinod Vaikuntanathan. „Indistinguishability obfuscation from functional encryption.“ In: *Journal of the ACM (JACM)* 65.6 (2018), pp. 1–37 (cit. on p. 2).

[BW18] Borja Balle and Yu-Xiang Wang. „Improving the gaussian mechanism for differential privacy: Analytical calibration and optimal denoising.“ In: *International conference on machine learning*. PMLR. 2018, pp. 394–403 (cit. on p. 46).

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

[CFLS20] Sergiu Carpov, Caroline Fontaine, Damien Ligier, and Renaud Sirdey. „Illuminating the Dark or how to recover what should not be seen in FE-based classifiers.“ In: *Proceedings on Privacy Enhancing Technologies* 2020.2 (2020), pp. 5–23 (cit. on p. 3).

[CMAK23] Shuangyi Chen, Anuja Modi, Shweta Agrawal, and Ashish Khisti. „Quadratic functional encryption for secure training in vertical federated learning.“ In: *2023 IEEE International Symposium on Information Theory (ISIT)*. IEEE. 2023, pp. 60–65 (cit. on p. 3).

[CZGQ23] Yansong Chang, Kai Zhang, Junqing Gong, and Haifeng Qian. „Privacy-preserving federated learning via functional encryption, revisited.“ In: *IEEE Transactions on Information Forensics and Security* 18 (2023), pp. 1855–1869 (cit. on pp. 2, 3).

[DLCC+21] Jian Du, Song Li, Xiangyi Chen, Siheng Chen, and Mingyi Hong. „Dynamic differential-privacy preserving sgd.“ In: *arXiv preprint arXiv:2111.00173* (2021) (cit. on p. 45).

[DOT18] Pratish Datta, Tatsuaki Okamoto, and Junichi Tomida. „Full-hiding (unbounded) multi-input inner product functional encryption from the k-linear assumption.“ In: *IACR International Workshop on Public Key Cryptography*. Springer. 2018, pp. 245–277 (cit. on pp. 2, 29, 40, 41, 43).

[ECLP24] Ferran Alborch Escobar, Sébastien Canard, Fabien Laguillaumie, and Duong Hieu Phan. „Computational differential privacy for encrypted databases supporting linear queries.“ In: *Proceedings on Privacy Enhancing Technologies* (2024) (cit. on pp. 3, 4).

[FN94] Amos Fiat and Moni Naor. „Broadcast encryption.“ In: *Advances in Cryptology—CRYPTO'93: 13th Annual International Cryptology Conference Santa Barbara, California, USA August 22–26, 1993 Proceedings 13*. Springer. 1994, pp. 480–491 (cit. on p. 2).

[GGGJ+14] Shafi Goldwasser, S Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. „Multi-input functional encryption.“ In: *Advances in Cryptology–EUROCRYPT 2014: 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings 33*. Springer. 2014, pp. 578–602 (cit. on p. 2).

[GGHR+16] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. „Candidate indistinguishability obfuscation and functional encryption for all circuits.“ In: *SIAM Journal on Computing* 45.3 (2016), pp. 882–929 (cit. on p. 2).

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. „Attribute-based encryption for fine-grained access control of encrypted data.“ In: *Proceedings of the 13th ACM conference on Computer and communications security*. 2006, pp. 89–98 (cit. on p. 2).

[GVW12] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. „Functional encryption with bounded collusions via multi-party computation.“ In: *Annual Cryptology Conference*. Springer. 2012, pp. 162–179 (cit. on p. 4).

[GVW15] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. „Attribute-based encryption for circuits.“ In: *Journal of the ACM (JACM)* 62.6 (2015), pp. 1–33 (cit. on p. 2).

[JLS21] Aayush Jain, Huijia Lin, and Amit Sahai. „Indistinguishability obfuscation from well-founded assumptions.“ In: *Proceedings of the 53rd annual ACM SIGACT symposium on theory of computing*. 2021, pp. 60–73 (cit. on p. 2).

[Log24a] LogisticDx. *Diagnostic Tests for Models with a Binomial Response. lbw: Low Birth Weight study data*. June 2024. URL: https://rdrr.io/rforge/LogisticDx/man/lbw.html (cit. on p. 32).

[Log24b] LogisticDx. *Diagnostic Tests for Models with a Binomial Response. pcs: Prostate Cancer Study data*. June 2024. URL: https://rdrr.io/rforge/LogisticDx/man/pcs.html (cit. on p. 32).

[Log24c] LogisticDx. *Diagnostic Tests for Models with a Binomial Response. uis: UMARU IMPATCT Study data*. June 2024. URL: https://rdrr.io/rforge/LogisticDx/man/uis.html (cit. on p. 32).

[LŢ19] Benoît Libert and Radu Ţiţiu. „Multi-client functional encryption for linear functions in the standard model from LWE.“ In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2019, pp. 520–551 (cit. on p. 2).

[MSHB+19] Tilen Marc, Miha Stopar, Jan Hartman, Manca Bizjak, and Jolanda Modic. „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. 2019, pp. 3–21 (cit. on p. 28).

[PT23] Prajwal Panzade and Daniel Takabi. „FENet: Privacy-preserving neural network training with functional encryption.“ In: *Proceedings of the 9th ACM International Workshop on Security and Privacy Analytics*. 2023, pp. 33–43 (cit. on p. 3).

[PTC23] Prajwal Panzade, Daniel Takabi, and Zhipeng Cai. „Privacy-preserving machine learning using functional encryption: Opportunities and challenges.“ In: *IEEE Internet of Things Journal* 11.5 (2023), pp. 7436–7446 (cit. on p. 3).

[QLHY+22] Xinyuan Qian, Hongwei Li, Meng Hao, Shuai Yuan, Xilin Zhang, and Song Guo. „Cryptofe: Practical and privacy-preserving federated learning via functional encryption.“ In: *GLOBECOM 2022-2022 IEEE Global Communications Conference*. IEEE. 2022, pp. 2999–3004 (cit. on p. 3).

[Sha07] Hovav Shacham. „A cramer-shoup encryption scheme from the linear assumption and from progressively weaker linear variants.“ In: *Cryptology ePrint Archive* (2007) (cit. on p. 40).

[SHZ25] Linda Scheu-Hachtel and Jasmin Zalonis. *Enhancing Noisy Functional Encryption for Privacy-Preserving Machine Learning*. 2025. arXiv: 2505.05843 [cs.CR] (cit. on pp. 2, 3, 10, 12, 14, 21, 22, 29).

[SS10] Amit Sahai and Hakan Seyalioglu. „Worry-free encryption: functional encryption with public keys.“ In: *Proceedings of the 17th ACM conference on Computer and communications security*. 2010, pp. 463–472 (cit. on p. 4).

[TAO20] Junichi Tomida, Masayuki Abe, and Tatsuaki Okamoto. „Efficient inner product functional encryption with full-hiding security.“ In: *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* 103.1 (2020), pp. 33–40 (cit. on p. 29).

[Ünа20] Akın Ünal. „Impossibility results for lattice-based functional encryption schemes.“ In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2020, pp. 169–199 (cit. on p. 11).

[Wat15] Brent Waters. „A punctured programming approach to adaptively secure functional encryption.“ In: *Annual Cryptology Conference*. Springer. 2015, pp. 678–697 (cit. on p. 2).

[ZASH24] Jasmin Zalonis, Frederik Armknecht, and Linda Scheu-Hachtel. „Differentially Private Functional Encryption.“ In: *Proceedings on Privacy Enhancing Technologies* 2 (2024), pp. 509–530. DOI: 10.56553/popets-2024-0061 (cit. on pp. 2, 3, 4, 9, 40, 41).

[ZSHA25] Jasmin Zalonis, Linda Scheu-Hachtel, and Frederik Armknecht. „A New Quadratic Noisy Functional Encryption Scheme and Its Application for Privacy Preserving Machine Learning.“ In: *International Conference on Applied Cryptography and Network Security*. Springer. 2025. DOI: 10.1007/978-3-031-95767-3_9 (cit. on pp. 2, 3, 4, 5, 7, 8, 10, 28, 29, 31, 32, 40, 44, 45, 46).

[ZSHA26] Jasmin Zalonis, Linda Scheu-Hachtel, and Frederik Armknecht. „A New Construction Method for More Efficient Quadratic One-Time Noisy Multi-Client Functional Encryption Schemes.“ In: *Proceedings of the 21th ACM Asia Conference on Computer and Communications Security*. 2026. Forthcoming (cit. on p. 1).

A A Labeled Affine Function-Hiding Multi-Client Functional Encryption Scheme

As explained in Section 4.1, our construction approach makes use of three building blocks, the third being an FH-OT-MCFE scheme fh. In our construction of QUILT (cf. Section 6), we realize this based on an extension of the affine FH-MIFE scheme [ZASH24], that extends the FH-IP scheme from [DOT18]. We provide this scheme in Figure 9 and highlight in color all necessary changes to support labeled ciphertexts and keys, lifting [ZSHA25] to the MCFE setting. Before we prove the security of this scheme, we introduce some necessary notations.

Notations. We use in the following similar notation as in [DOT18]. We quickly recall it here and refer to this work for further details. Let q be a prime number and $\mathbb{F}_q$ the finite field of size q.

Bilinear groups are described by $\mathbb{G} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$, where $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ 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 order 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 $\zeta, \eta \in \mathbb{Z}_q$.
- Non-degeneracy: $e(g_1, g_2) \neq 1_{\mathbb{G}_T}$, where $1_{\mathbb{G}_T}$ denotes the identity element of the group $\mathbb{G}_T$.

We denote by $[\mathbf{x}]_i$ a vector over $\mathbb{G}_i$, whose j-th element is $g_i^{\mathbf{x}[j]}$, and use this notation for scalars similarly.

Let $\mathcal{G}_{\mathsf{BPG}}$ be an algorithm that outputs a description $\mathsf{params}_{\mathbb{G}} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ of a bilinear group. The DPVS generation algorithm $\mathcal{G}_{\mathsf{DPVS}}$ outputs $\mathsf{params}_{\mathbb{V}} = (q, \mathbb{V}_1, \mathbb{V}_2, \mathbb{G}_T, \mathbb{A}_1, \mathbb{A}_2, e)$ of a dual pairing vector space, and $\mathcal{G}_{\mathsf{OB}}$ describes an dual orthogonal basis generator that outputs an orthogonal basis $\mathbb{B}_i = \{\mathbf{b}_{i,1}, \ldots, \mathbf{b}_{i,2m+2k+2}\}, \mathbb{B}_i^* = \{\mathbf{b}_{i,1}^*, \ldots, \mathbf{b}_{i,2m+2k+2}^*\}$, as described in [DOT18].

The hardness of the scheme relies on the k-LIN hardness assumption:

Assumption 0.1 (k Linear Assumption: k-LIN [Sha07])**.** *Let $k \geq 1$. Fix a number $\chi \in \{1, 2\}$. The k-LIN problem is to guess a bit $\beta \xleftarrow{\$} \{0, 1\}$ given $\epsilon_\beta = (\mathsf{params}_{\mathbb{G}}, g_\chi^{\xi_1}, \ldots, g_\chi^{\xi_k}, g_\chi^{\delta_1\xi_1}, \ldots, g_\chi^{\delta_k\xi_k}, \rho_\beta)$ where $\mathsf{params}_{\mathbb{G}} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e) \xleftarrow{\$} \mathcal{G}_{\mathsf{BPG}}(\lambda)$, $\xi_1, \ldots, \xi_k, \sigma \xleftarrow{\$} \mathbb{F}_q \setminus \{0\}$, $\delta_1, \ldots, \delta_k \xleftarrow{\$} \mathbb{F}_q$,*

$\rho_0 = g_\chi^{\sum \delta_j}$, *and* $\rho_1 = g_\chi^{\sigma+\sum \delta_j}$. *The k-LIN assumption states that for any PPT algorithm* $\mathcal{A}$ *and for any security parameter* λ*, the advantage of* $\mathcal{A}$ *in deciding the k-LIN problem is*

$$\mathsf{Adv}_{\mathcal{A}}^{k\text{-}LIN}(\lambda) = |\Pr[1 \leftarrow \mathcal{A}(\epsilon_0)] - \Pr[1 \leftarrow \mathcal{A}(\epsilon_1)]| \leq \mathtt{negl}(\lambda)$$

for some negligible function $\mathtt{negl}$.

Correctness and Security. The correctness of the scheme presented in Figure 9 follows analogue to [ZASH24].

Theorem 7. *Assuming the hardness of the k-LIN assumption, the MCFE scheme for affine functions described in Figure 9 is semi-adaptive* fh-many-*secure for static corruptions, i. e.,*

$$\begin{aligned}\mathsf{Adv}_{\mathcal{A}}^{\mathsf{MCFE}}(\lambda) &= |\Pr[\mathsf{Expt}_0^{\mathsf{MCFE}}(\mathcal{A}, 1^\lambda) = 1] - \Pr[\mathsf{Expt}_1^{\mathsf{MCFE}}(\mathcal{A}, 1^\lambda) = 1]| \\ &\leq \mathtt{negl}(\lambda).\end{aligned}$$

Proof. The proof extends naturally from the IP scheme in [DOT18] and its affine extension in [ZASH24]. First, the scheme already supports corruptions, although not explicitly mentioned. Specifically, observe that the encryption keys in both [DOT18] and [ZASH24] are independent of each other, since the orthogonal basis $\{\mathbb{B}_i, \mathbb{B}_i^*\}_{i\in[n]}$ are sampled independently for each $i \in [n]$. The remainder of the proof can be conducted by following the sequence of games described in [DOT18] only for the honest clients, i. e., all $i \in \mathcal{HS}$ instead of $i \in [n]$, as in [DOT18]. The proof still holds, since for all corrupted parties $i \in \mathcal{CS}$, it needs to hold that

$$\boldsymbol{c}_i^{\nu,0} = \boldsymbol{c}_i^{\nu,1} \text{ and } \mathbf{x}_i^{\mu_i,0} = \mathbf{x}_i^{\mu_i,1} \tag{11}$$

for all queried functions $\boldsymbol{c}^{\nu,\beta}$, $\nu \in [q_{\mathsf{dk}}]$, and all ciphertexts with $\mu_i \in [q_{\mathsf{ct},i}]$ to yield an admissible adversary (see Remark 4).

Second, the labeled variant can be proven secure by performing the described sequence of games for each label $\ell \in \mathcal{L}$ consecutively. The only potential critical step is the transformation from $f_\ell^{\nu,0}$ to $f_\ell^{\nu,1}$ in Lemma C.5 ([ZASH24] Appendix C.2, respectively [DOT18] Appendix C) where the random values $r_i^\nu, \nu \in [q_{\mathsf{dk},\ell}]$ are changed as follows for all honest clients $i \in \mathcal{HS}$:

$$\tilde{r}_{i,\ell}^\nu = r_{i,\ell}^\nu + \mathbf{x}_{i,\ell}^{1,0}\boldsymbol{c}_{i,\ell}^{\nu,0} - \mathbf{x}_{i,\ell}^{1,1}\boldsymbol{c}_{i,\ell}^{\nu,1}$$

$\mathsf{Setup}(1^\lambda, \mathcal{F}^{\mathsf{IP}}_{n,\{m,\mathsf{B}_X,\mathsf{B}_C\}})$

$\mathsf{params}_{\mathbb{G}} \xleftarrow{\$} \mathcal{G}_{\mathsf{BPG}}(\lambda)$ with $q \gg (nm+1)\mathsf{B}_X\mathsf{B}_C$,

$\mathsf{params}_{\mathbb{V}} \xleftarrow{\$} \mathcal{G}_{\mathsf{DPVS}}(\lambda, 2m+2k+2, \mathsf{params}_{\mathbb{G}})$

$\zeta \xleftarrow{\$} \mathbb{Z}_q \setminus \{0\}, g_T = e(g_1, g_2)^\zeta$

$\forall i \in [n]: \quad K_i \xleftarrow{\$} \{0,1\}^\lambda$

$\mathbb{B}_i, \mathbb{B}_i^* \xleftarrow{\$} \mathcal{G}_{\mathsf{OB}}(\lambda, 2m+2k+2, \mathsf{params}_{\mathbb{V}}, \zeta)$

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

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

return $\mathsf{pp} = (\mathsf{params}_{\mathbb{V}}, g_T), \mathsf{msk} = \{(\hat{\mathbb{B}}_i^*, K_i)\}_{i\in[n]}, \{\mathsf{ek}_i = (\hat{\mathbb{B}}_i, K_i)\}_{i\in[n]}$

$\mathsf{Enc}(\mathsf{pp}, \mathsf{ek}_i, i, \mathbf{x}, \ell)$

$t_i = \mathsf{PRF}(K_i, \ell), \varphi_1, \ldots \varphi_k \xleftarrow{\$} \mathbb{Z}_q$

return $\mathsf{ct}_i = (\mathbf{x}_i, \mathbf{0}^m, t_i, 1, \mathbf{0}^{k-1}, \varphi_1, \ldots, \varphi_k, 0)_{\mathbb{B}_i}$

$\mathsf{KeyGen}(\mathsf{pp}, \mathsf{msk}, \boldsymbol{c}, \ell)$

$\boldsymbol{c}_i := (\boldsymbol{c}[(i,1)], \ldots, \boldsymbol{c}[(i,m)]) \in \mathbb{Z}_q^m$

$\forall i \in [n]: \quad t_i = \mathsf{PRF}(K_i, \ell)$

$r_i, \rho_i, \gamma_{i,1}, \ldots \gamma_{i,k-1} \xleftarrow{\$} \mathbb{Z}_q$ such that $\sum_{i\in[n]} r_i = \boldsymbol{c}[(0,0)] - \sum_{i\in[n]} t_i\rho_i$

return $\mathsf{dk} = \{\mathsf{dk}_i = (\boldsymbol{c}_i, \mathbf{0}^m, \rho_i, r_i, \gamma_{i,1}, \ldots, \gamma_{i,k-1}, \mathbf{0}^k, 0)_{\mathbb{B}_i^*}\}_{i\in[n]}$

$\mathsf{Dec}(\mathsf{pp}, \mathsf{dk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_n)$

$[r]_T = \prod_{i\in[n]} e(\mathsf{ct}_i, \mathsf{dk}_i)$, finally perform exhaustive search step to find r

return r

Figure 9: Affine function-hiding noisy multi-client functional encryption scheme supporting labeled keys.

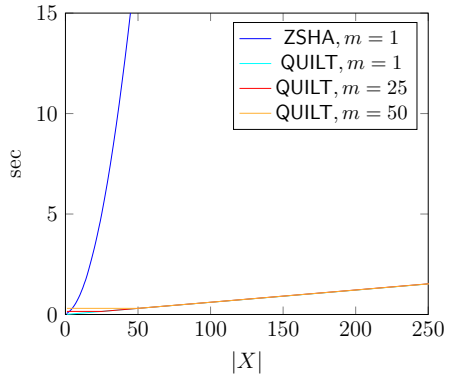


Figure 10: Runtime in seconds for small datasets.

and $\tilde{r}_{i,\ell}^{\nu} = r_{i,\ell}^{\nu}$ for $i \in \mathcal{CS}$. Because of the admissibility of the adversary, i. e., $f_{\ell}^{\nu,0}(\mathbf{x}_{1,\ell}^{\mu_i,0}, \ldots, \mathbf{x}_{n,\ell}^{\mu_i,0}) = f_{\ell}^{\nu,1}(\mathbf{x}_{1,\ell}^{\mu_i,1}, \ldots, \mathbf{x}_{n,\ell}^{\mu_i,1})$ and Equation (11), it holds that

$$\begin{aligned}
\sum_{i\in[n]} \tilde{r}_i^{\nu} &= \sum_{i\in[n]} \tilde{r}_i^{\nu} + \sum_{i\in\mathcal{HS}} (\mathbf{x}_{i,\ell}^{1,0} \boldsymbol{c}_{i,\ell}^{\nu,0} - \mathbf{x}_{i,\ell}^{1,1} \boldsymbol{c}_{i,\ell}^{\nu,1}) \\
&= \boldsymbol{c}[(0,0)]_{\ell}^{\nu,0} - \sum_{i\in[n]} t_{i,\ell}\rho_{i,\ell}^{\nu} + \sum_{i\in\mathcal{HS}} (\mathbf{x}_{i,\ell}^{1,0} \boldsymbol{c}_{i,\ell}^{\nu,0} - \mathbf{x}_{i,\ell}^{1,1} \boldsymbol{c}_{i,\ell}^{\nu,1}) \\
&\quad + (\boldsymbol{c}[(0,0)]_{\ell}^{\nu,1} - \boldsymbol{c}[(0,0)]_{\ell}^{\nu,1}) \\
&= \boldsymbol{c}[(0,0)]_{\ell}^{\nu,1} - \sum_{i\in[n]} t_{i,\ell}\rho_{i,\ell}^{\nu}
\end{aligned}$$

Therefore the random values $\tilde{r}_i^{\nu}$ still fulfill the required condition. The rest of the proof is analogue to [DOT18]. □

B Additional Benchmarking Results

This chapter provides supplementary experimental results in Figure 10 and Figure 11, that offer a more comprehensive view of our scheme's performance under various conditions.

In Section 6, we present results of the decryption process on datasets up to size $|X| = 30\,000$. However, even for for small input sizes, ZSHA takes significantly longer than QUILT. To allow for clearer comparison of the curves in Figure 10 we include a zoomed-in view for $|X| \leq 250$.

Our final comparison is motivated by the fact that ZSHA performs better for sparse polynomials, i. e., polynomials in which most of the

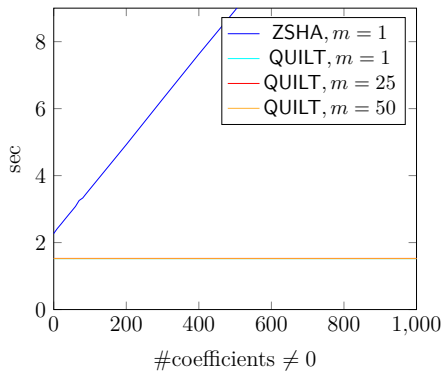


Figure 11: Runtime for decryption in dependency of the non-zero coefficients for $|X| = 250$.

coefficients are zero. This is because the costly paring operations only need to be computed for (i, j, k, l) where $\boldsymbol{c}[(i, j, k, l)] \neq 0$. This affects the decryption process only as all other algorithms are independent of the coefficients. Therefore, in Figure 11, we compare QUILT to ZSHA for sparse polynomials for a fixed dataset size $|X| = 250$. If we consider quadratic functions as described in Definition 4 and a dataset size of 250, we have at most 31 626 different monomials, while in Figure 11 we only present results up to 1 000 monomials. As can be seen, QUILT also significantly outperforms ZSHA in this case.

C Private Logistic Regression Training Using Noisy Multi-Client Functional Encryption

In the following we recall the protocol for PPML training from [ZSHA25].[6] The main idea is that the training algorithm in this case the gradient descent algorithm is decomposed in quadratic functions. Each function is then evaluated using an NMCFE decryption key, while the intermediate results are protected by adding noise according to DP.

[6] Since their schemes does not support labels, we will omit them in the following and assume for our adaptation, that each model training has its own label and is performed on one encrypted arbitrary distributed dataset using the same label.

C.1 Gradient Descent Algorithm

First, let us recall the standard gradient descent algorithm. Assuming a dataset X with n records, m attributes and 1 label per record, the goal is to optimize model weights $\boldsymbol{\theta}$ such that the label can be predicted with high accuracy. This is achieved over `it` iterations by iteratively updating the model weights $\boldsymbol{\theta}$ using an update function $F_{\boldsymbol{\theta}}(X) = \frac{\alpha}{n} \sum_{i \in [n]} \nabla L(\boldsymbol{\theta}, \mathbf{x}_i)$, where L is the loss function of the model. More precisely, in the case of logistic regression, for $j \in [0, m]$, the update is performed simultaneously as

$$\boldsymbol{\theta}^{(t+1)}[j] = \boldsymbol{\theta}^{(t)}[j] + \frac{\alpha}{n} \sum_{i \in [n]} \Big(\mathbf{x}_i[m+1] - \sigma\Big(\sum_{k \in [0,m]} \boldsymbol{\theta}^{(t)}[k] \mathbf{x}_i[k] \Big) \Big) \mathbf{x}_i[j],$$

with $\mathbf{x}_i[0] := 1$, where σ denotes the sigmoid function

$$\sigma(z) := \frac{1}{1 + \exp(-z)}.$$

To realize this using only a quadratic NMCFE scheme, [ZSHA25] propose to approximate the sigmoid function by $g(z) := 0.5 + 0.25z$, which yields a quadratic update procedure:

$$\boldsymbol{\theta}^{(t+1)}[j] = F_{\boldsymbol{\theta}^{(t)},j}(X) := \boldsymbol{\theta}^{(t)}[j] + \frac{\alpha}{n} \sum_{i \in [n]} \Big(\mathbf{x}_i[m+1] - \Big(0.5 + 0.25 \cdot \tag{12}$$

$$\sum_{k \in [0,m]} \boldsymbol{\theta}^{(t)}[k] \mathbf{x}_i[k] \Big) \Big) \mathbf{x}_i[j]. \tag{13}$$

C.2 Differential Privacy

The use of DP to achieve a private gradient descent training is well studied [DLCC+21]. The core idea of DP is that a function applied to a dataset is considered differentially private if an analyst, seeing only the function's output, cannot determine whether a specific data record is present in the dataset or absent from it. This is formally defined as follows:

Definition 5 (Differential Privacy)**.** *A randomized algorithm M with domain $\mathcal{X}$ is (ϵ, δ)-differentially private if for all $S \subseteq$ Range(M) and for all neighboring[7] datasets $X, X' \in \mathcal{X}, X \simeq X'$:* $\Pr[M(X) \in S] \leq \exp(\epsilon) \Pr[M(X') \in S] + \delta$*, where the probability space is over the coin flips of the mechanism M. If $\delta = 0$, we say that M is ϵ-differentially private.*

[7] "Neighboring" means that the databases differ by one element, written as $X \simeq X'$.

The privacy leakage of a function f can be measured by its sensitivity. Informally, the sensitivity captures how much one entry in a dataset can change the output of a function at most.

Definition 6 (l_2-sensitivity)**.** *The l_2-sensitivity of a function $f : \mathcal{X} \to \mathcal{Y}$ is $\Delta_2(f) = \max_{X,X' \in \mathcal{X}; X \simeq X'} ||f(X) - f(X')||_2$.*

A common way in PPML to achieve DP is to perturb the function output with noise sampled according to the analytic Gaussian Mechanism:

Definition 7 (Analytic Gaussian Mechanism [BW18])**.** *Let $f : \mathcal{X} \to \mathbb{R}^k$ be a function with sensitivity $\Delta_2(f)$ and Z a centered Gaussian random variable of dimension k with covariance matrix $\sigma^2 \mathbf{I}_k$. For any $\epsilon \geq 0$ and $\delta \in [0, 1]$, the Gaussian output perturbation mechanism $f^*(x) = f(x) + Z$ is (ϵ, δ)-DP if and only if*

$$\Phi\left(\frac{\Delta_2(f)}{2\sigma} - \frac{\epsilon\sigma}{\Delta_2(f)}\right) - \exp(\epsilon)\Phi\left(-\frac{\Delta_2(f)}{2\sigma} - \frac{\epsilon\sigma}{\Delta_2(f)}\right) \leq \delta,$$

where Φ denotes the Gaussian cumulative distribution function and $\Delta_2(f)$ describes the l_2-sensitivity.

The analyst can request multiple evaluations of selected functions f_i on the same target dataset. The combination of requests is still $(\sum_i \epsilon_i, \sum_i \delta_i)$-DP, if the analyst receives perturbed outputs $f_i(X) + \nu_i$ with ν_i drawn appropriately to achieve (ϵ_i, δ_i)-DP for this single request.

C.3 Protocol for Private Logistic Regression Training

The protocol proposed in [ZSHA25] leverages the idea of the private gradient descent, where the update function $F_{\boldsymbol{\theta}}(X)$ is perturbed by some additional noise ν, yielding

$$\boldsymbol{\theta}^{(t+1)}[j] = F_{\boldsymbol{\theta}^{(t)},j}(X) + \boldsymbol{\nu}^{(t)}[j]. \tag{14}$$

The sensitivity of the update function for normalized data, i. e., $\mathbf{x}_i \in [0, 1]$ is bounded by

$$\Delta_2(F_{\boldsymbol{\theta}^{(t)},j}) \leq \frac{\sqrt{(m+1)}\alpha}{n}\left(1 + 0.25||\boldsymbol{\theta}^{(t)}||_1\right).$$

Using the Gaussian Mechanism to sample the noise $\boldsymbol{\nu}$ yields a protocol for a differentially private logistic regression training.

More specific the protocol is divided into a setup and a training phase, which are displayed in Figure 12 and Figure 13, respectively.

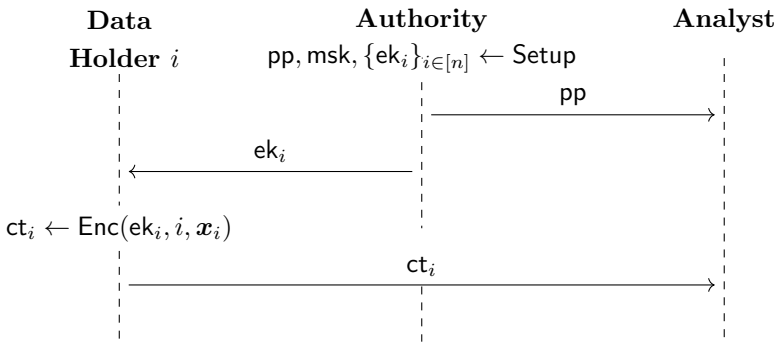


Figure 12: Setup phase of a logistic regression training using noisy functional encryption.

In the setup phase (Figure 12), the authority generates a master secret key msk and a set of encryption keys $\{\mathsf{ek}_i\}$ and distributes the latter over a secure channel to the data holders. Second, the data holders scale their input to convert their data into fix-point arithmetic using a predefined scaling factor s and encrypt it using their given encryption key. After this step, the data holders can either send the ciphertext directly to the analyst or upload the ciphertext to an untrusted server.

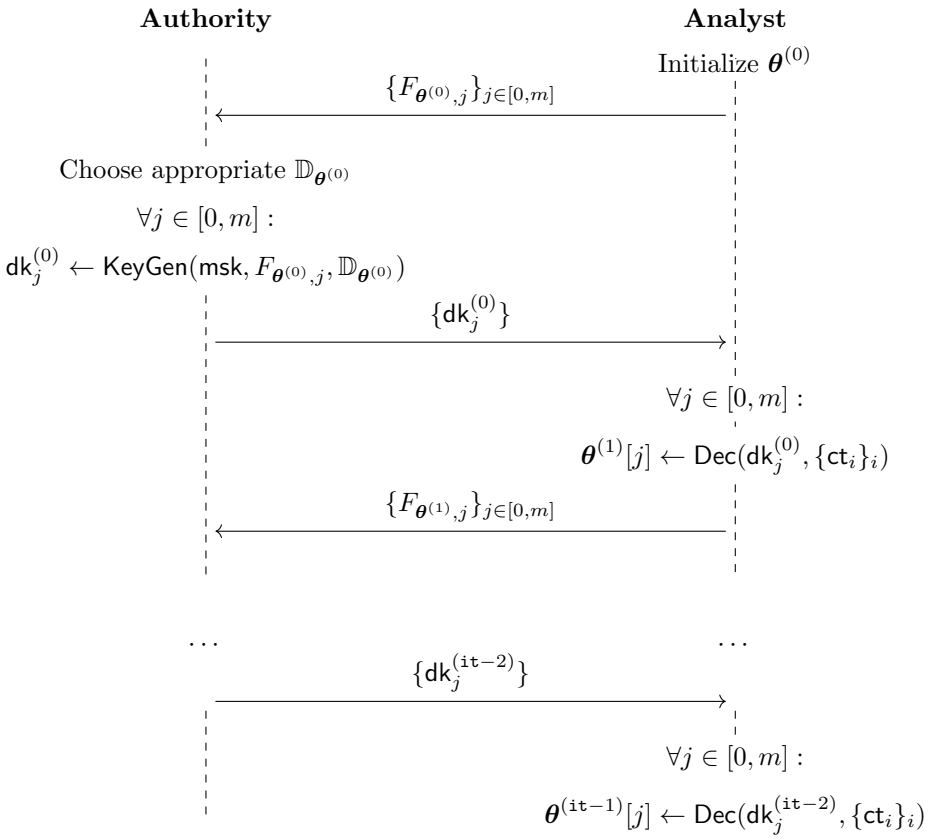


Figure 13: Training phase of a logistic regression training using noisy functional encryption.

In the actual training phase (Figure 13), the analyst first initializes a model weight vector $\boldsymbol{\theta}^{(0)}$. Then it requests $m+1$ decryption keys according to Equation (13) to update the model weights to compute $\boldsymbol{\theta}^{(1)}$. This is repeated for `it` iterations to compute the final model weights $\boldsymbol{\theta}^{(\mathtt{it}-1)}$. In each iteration, the analyst computes the updated weights by decrypting the ciphertexts containing the data of the data holders with the received noisy decryption keys. Throughout the training, the authority keeps track of the privacy budget and ensures an overall DP training by generating noisy decryption keys, according to a DP distribution using only the sensitivity of the update function.