COLMADE: **Collaborative Masking in Auditable Decryption for BFV-based Homomorphic Encryption**

Alberto Ibarrondo
IDEMIA & EURECOM
Sophia Antipolis, France

Hervé Chabanne
IDEMIA & Telecom Paris
Paris, France

Vincent Despiegel
IDEMIA
Paris, France

Melek Önen
EURECOM
Sophia Antipolis, France

ABSTRACT

This paper proposes a novel collaborative decryption protocol for the Brakerski-Fan-Vercauteren (BFV) homomorphic encryption scheme in a multiparty distributed setting, and puts it to use in designing a leakage-resilient biometric identification solution. Allowing the computation of standard homomorphic operations over encrypted data, our protocol reveals only one least significant bit (LSB) of a scalar/vectorized result resorting to a pool of N parties. By employing additively shared masking, our solution preserves the privacy of all the remaining bits in the result as long as one party remains honest. We formalize the protocol, prove it secure in several adversarial models, implement it on top of the open-source library Lattigo and showcase its applicability as part of a biometric access control scenario.

CCS CONCEPTS

• **Security and privacy** → **Privacy-preserving protocols**; *Biometrics.*

KEYWORDS

Muptiparty Homomorphic Encryption, Decryption, Masking, Biometric Identification, Secure Computation Leakage, Privacy Preserving Technologies

ACM Reference Format:
Alberto Ibarrondo, Hervé Chabanne, Vincent Despiegel, and Melek Önen. 2022. COLMADE: Collaborative Masking in Auditable Decryption for BFV-based Homomorphic Encryption. In *Proceedings of the 2022 ACM Workshop on Information Hiding and Multimedia Security (IH&MMSec '22), June 27–28, 2022, Santa Barbara, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3531536.3532952

1 INTRODUCTION

Data has been frequently labelled as the 21st century oil. There are innumerable modern applications fueled by data, ranging from Data Analytics & Machine Learning to Biometrics to name a few, whose impact in society is undeniable. Yet, the tremendous potential of data manipulation is coupled with high risks. Data privacy essential to deal with risks of data misuse or theft, even more so when dealing with personal data (as covered in present-day data protection legislations such as GDPR [16] or HIPAA [11]).

Moreover, there are sectors where these risks are unacceptable or even illegal. Biometric Identification must rely on secure hardware or trusted parties to hold the personal data vital for their recognition models, and all the biometric data manipulation must follow strict security rules. Hospitals and health specialists are deprived of the advantages of training and using models with all the available data from patients. Banks and finance institutions are limited to the locally available data to prevent fraud and prosecute tax evasion. Child Exploitative Imagery detection models [54] need training data that is in itself illegal to possess.

An observant reader might notice that many of the use-cases just described share two important characteristics: there are multiple parties interested in performing a chosen computation over personal data, and the output of such computation can often be expressed with a single bit: Is a user in the database/list of registered users? Does the patient present indicators of cancer? Is a certain entity committing fraud? These aspects will play a key role in the present work.

Under the field of advanced cryptography, several privacy preserving technologies arise to the challenge. *Fully Homomorphic Encryption* (FHE)[23] is a family of encryption schemes that support certain operations between ciphertexts (typically addition and multiplication), yielding the results of these operations when decrypting. Secure Multiparty Computation (MPC) covers a series of techniques (garbled circuits[55], secret sharing[49], or the more recent replicated secret sharing[1] and functional secret sharing[9]) that split computation of a given function across multiple distinct parties, so that each individual party remains ignorant of the global computation, and collaborate to jointly compute the result. FHE and MPC can also coexist in Multiparty Homomorphic Encryption[36] (MHE), where distributed versions of the FHE protocols are carried out by several collaborating parties.

These technologies alone offer private computation capabilities suited for biometric operations, often based on threat models with *Honest-But-Curious* adversaries, where parties involved perform the selected protocol faithfully and without deviation while attempting to obtain as much information from the private data as possible. Indeed FHE & many MPC techniques fall under this category, whereas some MPC techniques can deal with *Malicious* adversaries capable

IH&MMSec '22, June 27–28, 2022, Santa Barbara, CA, USA

ACM ISBN 978-1-4503-9355-3/22/06…$15.00
https://doi.org/10.1145/3531536.3532952

of deviating arbitrarily from the protocol. Besides requiring these stronger threat models, industrial instantiations of these technologies are frequently sought to be auditable, so that an independent auditor may inspect some of the protocol execution based on the public protocol transcript [5].

Even then, the output of a private computation protocol always leaks some information about the input, as this leakage is inherent to the function being computed and independent from the chosen private computation technology. E.g., model extraction attacks [39, 53] and membership inference attacks[50] in privacy-preserving machine learning inference, or reference biometric template extraction and brute force impersonation in privacy-preserving biometric identification [27]. The most straightforward way to thwart these attacks is to limit the output to yield strictly minimal information (e.g., one bit for binary decisions).

In designing a secure biometric identification system, all these properties are clearly desired: minimal output leakage, auditability and guaranteed data privacy in stronger threat models. The main motivation of this work is to combine them all, improving on previously proposed biometric systems that only tackle a subset of these properties.

Our Contribution. We propose a novel decryption protocol with collaborative masking based on the multiparty variant [36] of the Brakerski-Fan-Vercauteren (BFV) [22] Homomorphic Encryption scheme. Our protocol makes use of predefined pools of users to perform a decryption in a distributed setting, where each user masks a fragment of the ciphertext during decryption while remaining agnostic of the full computation. We guarantee the privacy of all but one bit of the disclosed output in diverse threat models. COLMADE effectively reduces the input leakage to the minimum possible by masking all but the Least Significant Bit of the encrypted output produced by a HE-based private computation. We display its relevance by using it to construct an auditable privacy-preserving biometric identification system. Lastly, we open-source an implementation of this protocol on top of the Lattigo HE library[20].

A fitting application. We target biometric access control for groups where multiple users are expected to get together right before identifying themselves to request access. Together with classic biometric access control scenarios such as airplane boarding or multitudinous events (popular sport matches or concerts), we consider events such as entering a museum, a temporary exhibition, a fair or a semi-professional sports competition. In these selected applications, the computation to perform the biometric identification of the user Alice would happen in the FHE domain, and a set of users (who may or may not include Alice) would collaborate with the gatekeeper to decrypt only the one bit of vital information required to answer "Is Alice in the list of registered people?".

This work is arranged as follows. Section 2 introduces the single-party & multi-party BFV encryption schemes. Section 3 details our design of a biometric access control solution, and leans on it to formalize three protocols: a simplified single-party masked decryption, our flagship privacy-preserving multiparty decryption protocol already displayed in the biometric system, and an extended protocol to provide abort against a malicious adversary in an honest majority setting. Section 4 covers an in-depth security analysis of our protocols. The paper wraps up with a succinct mention to the implementation in Section 5, previous works in Section 6 and some takeaways in Section 7.

2 PRELIMINARIES

2.0 Notation

We use regular letters for integers and polynomials, and boldface letters for vectors of integers and of polynomials. R_z expresses a polynomial ring with integer coefficients modulo z. $a[j]$ denotes the $j-th$ coefficient/element of a polynomial/vector a with N coefficients/elements. Given a sampling of an individual coefficient $a[j]$ from a distribution $\mathcal{D}$, written as $a[j] \sim \mathcal{D}$, we denote the sampling of a polynomial a over a ring R_z as $a \leftarrow \mathcal{D}_{[R_z]}$. A computing party is denoted as P_i in a pool of K parties $\mathcal{P} = \{\mathsf{P}_1, \ldots, \mathsf{P}_i \ldots, \mathsf{P}_K\}$.

We use $\langle a \rangle_i$ to refer to share i in an arithmetic secret sharing of $a = \sum_i^K \langle a \rangle_i$. We denote $[\cdot]_q$ the reduction modulo q, and $\lfloor \cdot \rfloor$, $\lfloor \cdot \rceil$, $\lceil \cdot \rceil$ the rounding to the previous, nearest and next integer respectively. When applied to polynomials or vectors, these reductions are performed coefficient/element-wise. We use $\mathcal{U}(X)$ to denote a uniformly random distribution in the set X, and $\mathcal{N}(\mu, \sigma)$ to denote a univariate gaussian distribution with mean μ and standard deviation σ. For a polynomial a, we write its infinity norm as $\|a\|$. For an input integer $x \in \mathbb{Z}$ we use $sign(x) = x/|x|$, and define $sgb(x)$ as the sign bit such that:

$$sgb(x) = \begin{cases} 1 & \text{if } x < 0 \\ 0 & \text{if } x \geq 0 \end{cases}$$

2.1 Homomorphic Encryption

A *homomorphic* encryption scheme allows certain operations over ciphertexts, which is equivalent to encrypting the result of those same plaintext operations. Thanks to this, third parties can perform computations on encrypted data without learning the inputs or the computation results. In contrast to *partially* homomorphic encryption, which supports only one type of arithmetic operation (e.g. only additions[38] or only multiplications[46]), *fully* homomorphic encryption allows encrypted multiplications and additions, theoretically enabling private computation of arbitrary functions. This concept was conceived by Rivest et al. in the 1970s[45], but it remained unrealized until Craig Gentry presented a first feasible FHE scheme in 2009[23]. Since then, FHE has gone from theoretical breakthrough to practical deployment, dropping the initial 30 minutes required to compute a multiplication between two encrypted values down to less than 20 milliseconds. Even then, FHE multiplications are still around seven orders of magnitude slower than native CPU integer multiplication instructions. Therefore, practical FHE requires that applications be specifically adapted and optimized.

The majority of modern FHE schemes are based on the Learning with Errors (LWE) hardness assumption [42] and its variants (e.g., Ring LWE) and rely on a small amount of *noise* added during encryption to guarantee security. During homomorphic operations, this noise grows negligibly for additions, and significantly for multiplications. Should the noise grow too large, correct decryption would no longer be possible. Theoretically, a computationally expensive technique known as *bootstrapping* can be used to homomorphically reset the noise in a ciphertext. Instead, schemes are instantiated with parameters large enough to allow the computation to complete without requiring bootstrapping.

We now introduce the Brakerski/Fan-Vercauteren (BFV) [10, 22] scheme, foundational to our work, leaving out other schemes such as Cheon-Kim-Kim-Song (CKKS) [12] or TFHE [13].

2.2 BFV scheme

The Brakerski/Fan-Vercauteren scheme [10, 22] is a ring-learning-with-errors (RLWE)[35] homomorphic encryption scheme. Messages are encoded in the plaintext space $R_t = \mathbb{Z}_t[X]/(X^N + 1)$ of polynomials of degree up to $N-1$, and then encrypted into the ciphertext space $R_q = \mathbb{Z}_q[X]/(X^N+1)$ with $t < q$ (typically $t \ll q$), N a power of 2 and $\Delta = \lfloor q/t \rfloor$.

The BFV scheme utilizes secrets sampled from two small-normed distributions: the secret key distribution $\mathcal{S}_{[R_q]}$ with coefficients sampled from a uniform distribution $s[j] \sim \mathcal{S} \triangleq \mathcal{U}(\{-1,0,1\})$ so that Image$(\mathcal{S}_{R_q}) = \mathbb{Z}_{\{-1,0,1\}}[X]/(X^N+1)$. (although in some implementations they are instead sampled from $\mathcal{U}(\{0,1\})$), and the error distribution $\mathcal{X}_{[R_q]}$ with coefficients $e[j] \sim \mathcal{X} \triangleq \mathcal{N}_{[-B,B]}(0,\sigma)$ sampled following a discrete Gaussian with standard deviation σ truncated into $[-B,B]$ where σ and B are two cryptosystem parameters.

The security of BFV is rooted in the hardness of the *decisional-RLWE problem*, informally stated as: given a uniformly random $a \leftarrow \mathcal{U}(R_q)$, a secret $s \leftarrow \mathcal{S}_{[R_q]}$, and an error term $e \leftarrow \mathcal{X}_{[R_q]}$, it is computationally hard for an adversary that does not know s and e to distinguish between the distribution of $(sa+e, a)$ and that of (b, a) where $b \leftarrow \mathcal{U}(R_q)$. A more formal definition can be found in Section 3.1 of [22].

While BFV supports bootstrapping in theory, it is slow and thus the scheme is commonly instantiated with parameters large enough to handle the noise growth result of a limited number of multiplications (the *multiplicative depth*). Even then, a public *relinearization* should be used between multiplications to reshape the ciphertext without changing the underlying message, lowering noise growth and ciphertext size by employing a specific public key named relinearization key (rlk).

Scheme 1 outlines the subset of algorithms conforming the BFV scheme that are pertinent for this work. We will now focus on the decryption algorithm, as it constitutes the foundation of Colmade. A reasoning on how to select these parameters in practice can be followed in Section 3.4 of [36].

2.2.1 Decryption in BFV. The decryption of a ciphertext $\mathbf{c}_t = (c_{t_0}, c_{t_1})$ can be described as a two-step process. The first step takes the secret key to compute a noisy upscaled plaintext in R_q as:

$$[c_{t_0}, +sc_{t_1}]_q = \Delta m + e_{\mathbf{c}_t} \tag{1}$$

where $e_{\mathbf{c}}$ is the ciphertext error/noise. In the second step, the message is decoded from this upscaled noisy term in R_q to a plaintext in R_t, by downscaling and rounding:

$$\left[\left\lfloor \frac{t}{q}(\Delta m + e_{\mathbf{c}_t}) \right\rceil\right]_t = [\lfloor m + \alpha t + v \rceil]_t \tag{2}$$

where $m \in R_t$, $\alpha \in \mathbb{Z}^N$, $v \in \mathbb{Q}^N$. The correctness of the decryption is preserved as long as the noise residue v in the plaintext space be $\|v\| < 1/2$, which translates into an upper bound to the ciphertext noise term of $e_{ct} < \Delta/2$. This is often achieved by choosing a sufficiently large q.

Scheme 1 BFV(t, n, q, w, σ, B)

BFV.SecKeyGen() $\rightarrow sk$:
Sample $s \leftarrow \mathcal{S}_{[R_q]}$
Output $sk = s$

BFV.PubKeyGen(sk **)** $\rightarrow$ **pk**:
Let $sk = s$ a secret key
Sample $p_1 \leftarrow \mathcal{U}(R_q)$, and $e \leftarrow \mathcal{X}_{[R_q]}$
Output $\mathbf{pk} = (p_0, p_1) = (-sp_1 + e, \ \ p_1)$

BFV.Encrypt($\mathbf{pk}$, m **)** $\rightarrow \mathbf{c}_m$:
Let $\mathbf{pk} = (p_0, p_1)$ a public key
Sample $u \leftarrow \mathcal{S}_{[R_q]};\ e_0 \leftarrow \mathcal{X}_{[R_q]};\ e_1 \leftarrow \mathcal{X}_{[R_q]}$
Output $\mathbf{c}_m = (c_{m_0}, c_{m_1}) = (\Delta m + up_0 + e_0, \ \ up_1 + e_1)$

BFV.Add($\mathbf{c}_a, \mathbf{c}_b$ **)** $\rightarrow \mathbf{c}_{add}$:
Let $\mathbf{c}_a = (c_{a_0}, c_{a_1})$, $\mathbf{c}_b = (c_{b_0}, c_{b_1})$ two ciphertexts.
Output $\mathbf{c}_{add} = ([c_{a_0} + c_{b_0}]_q, \ \ [c_{a_1} + c_{b_1}]_q)$

BFV.Decrypt($sk, \mathbf{c}_t$ **)** $\rightarrow$ m$_{res}$:
Let $sk = s$ a secret key, $\mathbf{c}_t = (c_{t_0}, c_{t_1})$ a ciphertext.
Output $m_{res} = \left[\left\lfloor \frac{t}{q}[c_{t_0} + sc_{t_1}]_q \right\rceil\right]_t$

2.3 Multiparty BFV Scheme

Multiparty Homomorphic Encryption (MHE) provides a natural extension of FHE to an N-party setting. We will focus on the Distributed BFV scheme or DBFV of [36], summarizing some of the key protocols in Scheme 2. The secret key generation is now performed by local generation of shares $\langle sk \rangle_i$, then the collective public key protocol is based on the underlying global secret key sk. To preserve practical security, sk is never reconstructed, but parties can collaborate to generate a collective public key cpk corresponding to sk. Homomorphic encryption and evaluation are left untouched from the original BFV scheme (Scheme 1), whereas relinearization and key switching present specific multi-party protocols (protocols 3 and 4 of [36]).

2.3.1 Decryption and Noise in DBFV. The main difference between the single-party BFV.Decrypt protocol summarized in equations 1 and 2 and its multiparty counterpart DBFV.ColDecrypt is that an additional error is introduced to preserve the security of the $\langle c_{1s} \rangle_i$ shares based on the decisional RLWE problem.

The distributed secret-key generation protocol yields a global secret key sk whose coefficients, as a sum of K samples of $\mathcal{R}_s$, add up to a maximum of $\|sk\| \leq K$. As a result of DBFV.ColPubKeyGen, the collective public key cpk contains noise $e_{cpk} = \sum_i^K e_i$, implying that $\|e_{cpk}\| \leq KB$.

Thus, a freshly encrypted ciphertext $\mathbf{c}_m = (c_{m_0}, c_{m_1})$ of a message m under a collective public key cpk will decrypt, following equation 1 with the single-party BFV.Decrypt, to $[c_{m_0}, +sc_{m_1}]_q = \Delta m + e_{fresh}$, where $\|e_{fresh}\| \leq B(2NK+1)$. Thus, the worst-case fresh ciphertext noise is linear in the number K of parties.

Scheme 2 DBFV(t, N, q, σ, B, K)

DBFV.SecKeyGen() $\to \langle sk \rangle_1, \ldots \langle sk \rangle_i, \ldots \langle sk \rangle_K$:

P_i: Sample $s_i \leftarrow \mathcal{S}_{[R_q]}$

P_i: Output $\langle sk \rangle_i = s_i$, where $sk = \left[\sum_i^K sk_i\right]_q$

DBFV.ColPubKeyGen($\langle sk \rangle_1, \ldots \langle sk \rangle_i, \ldots \langle sk \rangle_K$ **)** $\to$ **cpk** :

Let $sk_i = s_i$ private key share of P_i.

Any: Sample $p_1 \leftarrow \mathcal{U}(R_q)$. Disclose to all P_i.

P_i: Sample $e_i \leftarrow \mathcal{X}_{[R_q]}$

P_i: Compute $\langle p_0 \rangle_i = -p_1 s_i + e_i$

Any: Output $\mathbf{cpk} = (p_0, p_1) = (\sum_i^K \langle p_0 \rangle_i, \ \ p_1)$.

DBFV.Encrypt(**cpk**, m **)** $\to \mathbf{c}_m$:

Any: Output $\mathbf{c}_m$ = BFV.Encrypt(**cpk**, m).

DBFV.ColDecrypt($\mathbf{c}, \langle sk \rangle_1, \ldots \langle sk \rangle_i, \ldots \langle sk \rangle_K$ **)** $\to m_{res}$:

Let $s_i = \langle sk \rangle_i$ private key share of P_i, $\mathbf{c}_t = (c_{t_0}, c_{t_1})$ a ciphertext.

P_i: Sample $e_i \leftarrow \mathcal{X}_{[R_q]}$

P_i: Compute $\langle c_{1s} \rangle_i = s_i c_1 + e_i$

Any: Output $m_{res} = \left[\left\lfloor \frac{t}{q} \left[c_0 + \sum_i^K \langle c_{1s} \rangle_i\right]_q \right\rceil\right]_t$

Conversely, the freshly encrypted ciphertext $\mathbf{c}_m = (c_{m_0}, c_{m_1})$ of a message m under a single-party public key pk will generate, following equation 1 and using the multi-party DBFV.ColDecrypt, a similar error term, which then doubles if both DBFV.ColDecrypt and a collective public key are used.

2.4 Encoding, Packing and modular operations

Inputs to BFV.Encrypt are first to be encoded into the plaintext space R_t. We consider two main encoding techniques (see Scheme 3): *base* encoding, where a single integer fills an entire plaintext, and *packed* encoding, where a vector of integers is mapped elementwise to the coefficients of the plaintext. Figure 1 illustrates an example of these encodings.

The *packing* technique enables Single Instruction Multiple Data (SIMD) parallelism, making it highly efficient for applications working over larger amounts of data while supporting both additive and multiplicative homomorphic operations. Due to its practicality, it is implemented in most of the current lattice-based cryptographic libraries [20, 26, 41, 48] and is part of the draft HE standard [2].

Homomorphic addition is naturally performed elementwise when adding two packed polynomials. To obtain homomorphic multiplication applied elementwise, one needs to follow the instructions for RLWE-based *packing* from section 3.2 of [51]. In short, input integer vectors need to be encoded using the inverse Number Theoretic Transform (InvNTT) over R_t to turn polynomial multiplications into coefficient-wise multiplications. Additionally, *rotation* operations cyclically rotate the elements inside the vectors, allowing elements originally stored at different indices (also known as "slots") to interact.

Furthermore, typical applications of homomorphic encryption deal with operations in the non-modular domains $\mathbb{Z}$ or $\mathbb{R}$. Their coercion to arithmetic modulo t forces the underlying plaintext operations to not overflow their coefficients modulo t. Hence, the

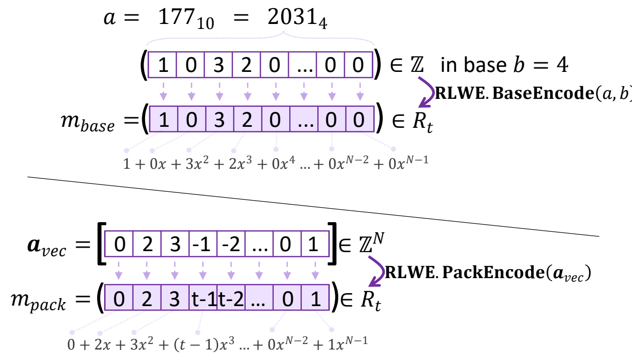

Figure 1: Visualization of the BaseEncode (top) and PackEncode (bottom) algorithms for arbitrary inputs

Scheme 3 RLWE Codec(t, N)

RLWE.BaseEncode(a, b **)** $\to m$:

Let $a \in \mathbb{Z}$ an input integer value with up to N digits in base-b representation.

Output: Polynomial $m \in R_t$ with

$m[j] = (\lfloor |a| \rfloor_{b^{(j+1)}} - \lfloor |a| \rfloor_{b^j}) \ \ \forall j$ for *unsigned* encoding,

$m[j] = (\lfloor |a| \rfloor_{b^{(j+1)}} - \lfloor |a| \rfloor_{b^j}) * sign(a) + t * sgb(a) \ \ \forall j$ for *signed* encoding.

RLWE.BaseDecode(m, b **)** $\to a_{res}$:

Let $m \in \mathbb{Z}_t^N$ the coefficients of an encoded polynomial in R_t.

Output Integer $a_{res} \in \mathbb{Z}$ with

$a_{res} = \sum_{i=1}^N m[i] * b^{(i-1)}$ for *unsigned* encoding,

$a_{res} = \sum_{i=1}^N m[i] * b^{(i-1)} * (-1) * sign(m[i] - t/2)$ for *signed* encoding.

RLWE.PackEncode(**a** **)** $\to m$:

Let $\mathbf{a} \in \mathbb{Z}^N$ an input vector with N elements in:

$[0, t)$ for *unsigned* encoding,

$[-t/2, t/2)$ for *signed* encoding,

where t must be a prime congruent to 1 mod $2N$.

Output Polynomial $m \in R_t$ where m = InvNTT($\mathbf{a}$ mod t).

RLWE.PackDecode(m **)** $\to \mathbf{a}_{res}$:

Let $m \in Z_t^N$ the coefficients of an encoded polynomial of degree $N - 1$ in R_t.

Compute $\mathbf{a}_{dec}$ = NTT(m).

Output

$\mathbf{a}_{res} = \mathbf{a}_{dec}$ for *unsigned* encoding,

$\mathbf{a}_{res}[i] = (\mathbf{a}_{dec}[i] - t)$ if $\mathbf{a}_{dec}[i] > t/2$ else$(\mathbf{a}_{dec}[i]) \ \forall i$ for *signed* encoding.

encrypted vector elements are limited to t when using *packing*, and the digits of the encrypted value in base-b representation must fall below t when using *base* encoding. If using signed encoding, the underlying values/digits are limited to the interval $[-t/2, t/2)$. For deep arithmetic circuits, this overflow limitation causes t to take higher values, at a non-negligible performance cost. [1]

3 OUR CONTRIBUTION

3.1 Towards biometric database protection

Firstly, let us sketch the main components of a biometric system for access control, depicted in Figure 2:

- A *Biometric Identity Provider (BIP)*, holding a database of reference biometric templates and executing the identification operations upon reception of a live biometric template.
- A service provider acting as *gatekeeper (Gate)*, in charge of capturing live biometric data of an individual requesting access, submitting it to the BIP and authorizing/denying him access based on the identification result.
- *Users/individuals*, seeking to access the premise or service.

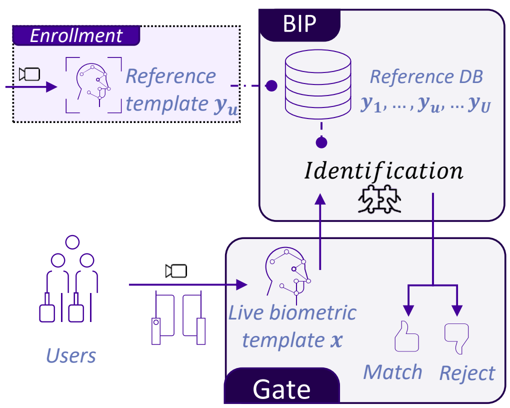

Figure 2: Sketch of a biometric system for access control

In biometric systems such as this, the most sensitive asset is the database of stored biometric samples that act as references for identification requests. These *reference templates* could lead to successful impersonation attempts were they to fall in the wrong hands, and their unintended disclosure can lead to severe privacy breaches: knowing who can enter a biometric access system may make these individuals subject of targeted attacks.

To protect the privacy of this database, a straightforward solution is to encrypt it. Enhancing this system with FHE allows the BIP to hold the encrypted database and execute the identification operations over the encrypted domain. Yet, revealing the identification result to the Gate poses several non-trivial concerns:

(1) The decryption algorithm makes use of the secret key, but this secret key could also be used to decrypt the database.
(2) It is impractical to have one secret key per identity in the database, as individuals might wish to gain access without a personal device holding his key (e.g., smartphone ran out of battery). Also, multi-key homomorphic encryption solutions [32] do not scale well with a high number of keys.

[1]CKKS[12] solves this elegantly at the expense of introducing additional noise in the computation result.

In a real-world instantiation of this biometric system, neither Gate nor the BIP should hold the secret key, as either could team up with the other to fully decrypt the database. Moreover, other issues emerge concerning identification requests formulated by Gate and the responses of BIP:

- The Gate could seek the creation of a False Acceptance (FA) to determine what are the identities present in the database, or search beforehand if a particular person is in there.
- The BIP could exploit the fact that his answers are encrypted to leak the identities present in the base.

One solution to deal with these issues would be to combine Homomorphic Encryption with Verifiable Computing. However, at the present day this combination is far from being practical[6].

We propose a different approach: splitting the decryption among the users seeking access. We thus hypothesize that a certain number of users will accept to cede a bit of CPU in their smartphones for this task. We could then envision individuals without a phone benefitting from this distributed decryption service thanks to other users. This way, the notion of distributing personal secret keys based on the principle of consent is replaced by a cooperative decryption carried out by those seeking to utilize the identification service. To entice users to do so, we limit their interactions to be only with the Gate, and not with other individuals. [2]

Under this setting the Gate is forced to communicate with the users to decrypt the results of the identification, producing a public communication transcript that will render the service auditable and consequently reduce the risks associated to his requests.

3.2 COLMADE for group biometric identification

Motivated by the use-case of biometric access control for group events, we display the COLMADE protocol in Figure 3, to answer if user Alice is allowed to access the museum. We distinguish three types of actors in our scenario:

- *BIP*, holding the encrypted database of reference templates belonging to users allowed to access.
- *Gate*, in charge of capturing the live biometric template of the users requesting access, submitting it to the BIP (possibly encrypted), receiving the encrypted result and aggregating the decryption shares.
- $\mathsf{P}_1 \dots \mathsf{P}_i \dots \mathsf{P}_K$. Pool of users holding shares of the global secret key and global masking polynomials. They collaborate to perform a masked decryption of the encrypted identification computation. Alice might or might not be among them. We contemplate multiple pools of users, each with their own sharing of the same global secret key. K needs not be fixed equally for all the pools of users.

This protocol requires a trusted setup to function. Nevertheless, most of these setup operations could be instantiated with alternative protocols that do not require full trust.

Armed with these distinctions, the system would carry out Alice's identification following this steps:

(1) **Key Generation**: The trusted setup generates a global secret and public BFV keys and secret-shares the secret key

[2]Foreseeing some unwillingness on the user side to participate in this protocol in the real world, we suggest to encourage participation using incentives (e.g., discounts, benefits) to overcome their reluctance.

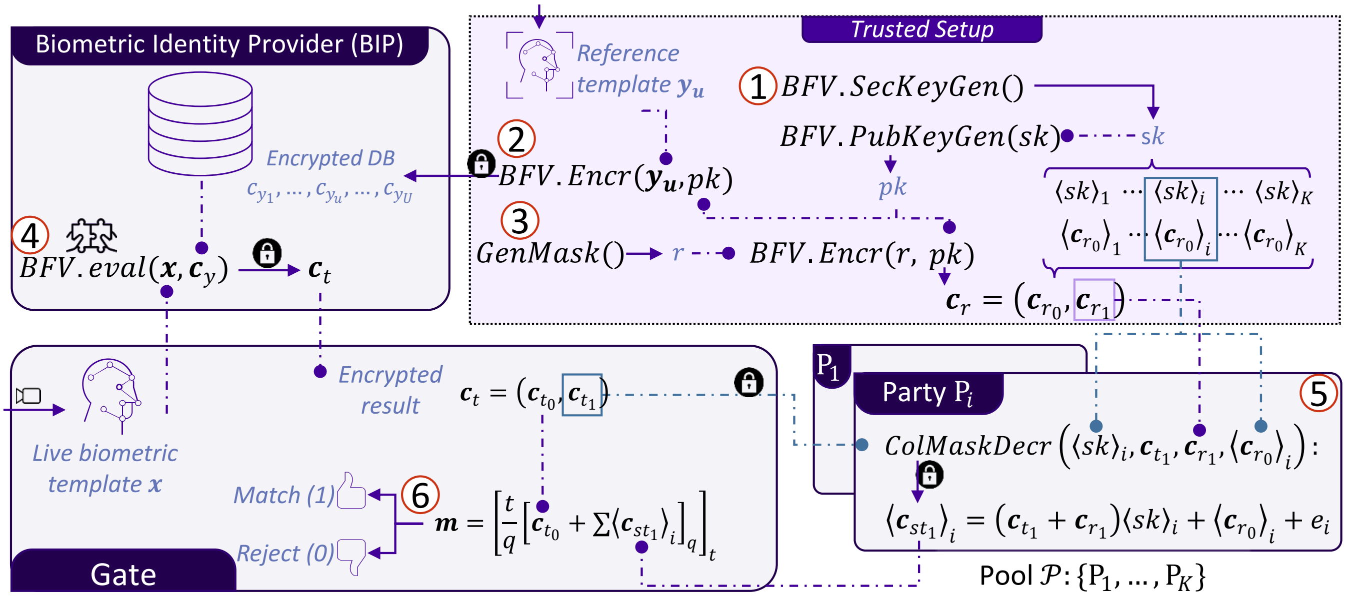

Figure 3: System diagram of a biometric access control using the ColMaDe protocol for single-bit masked results.

in K shares. A distributed alternative would be to instead employ DBFV.SecKeyGen and DBFV.ColPubKeyGen for key generation, and then use the Enc2Share protocol from [36].

(2) **Reference Database Encryption**: During the enrollment process, the users to be included in the access list would yield biometric templates to act as references once they arrive at the gate. These reference templates are encrypted with the public key generated in the previous step, and are then sent to the BIP for safe storing.

(3) **Randomness generation** Following the setup of Protocol 5, the masking polynomial is sampled, encrypted and secret shared.The trusted entity then sends all the required pieces to each of the parties enrolling. In practice this would typically happen in a previous "offline" phase, after the enrollment, that could alternatively be based on correlated randomness[29].

(4) **Encrypted identification** Once Alice approaches the gate, a biometric template is extracted from her by the Gate and sent to the BIP. This live template could be encrypted at the cost of slower operations in the BIP, but guaranteeing privacy of the live template in the BIP. Once received, the BIP would perform the encrypted identification (e.g., vector-matrix multiplication followed by comparison to threshold[28] and aggregation) with the encrypted DB, sending the encrypted result back to the Gate.

(5) **Collaborative Masked Decryption** Upon receiving the encrypted result, the Gate would request a decryption to a pool of users by sending them all the second polynomial of the encrypted result. Each of the parties answers back with a share of that decrypted and masked polynomial.

(6) **Result** The Gate aggregates all the decrypted polynomial shares with the first polynomial of the encrypted result, decrypts and decodes the underlying message and answers, based on the single LSB bit disclosed, if Alice is in the list and can access the premises/service.

3.3 Masked Decryption

The goal of the ColMaDe protocol is to conceal all bits save a single LSB of the underlying message during BFV decryption. To that end we add a masking term as part of the decryption protocol. Portrayed in Figure 4 as an additive term in the plaintext ring, this mask will depend on the type of encoding being used:

- *Base encoding* places one base-b digit per polynomial coefficient. We distinguish two cases:
  - For an even b, the LSB of the underlying integer value depends only of the polynomial coefficient $j = 0$, corresponding to the b^0 term. The desired masking term r would have to fully hide all coefficients except $j = 0$, and this coefficient ought to have all but the LSB bit masked.
  - For odd b, the LSB depends on all the coefficients of the polynomial, and thus no suitable additive mask can be applied. We disregard this case from this point onward.
- *Packed encoding* places one vector element per polynomial coefficient. The desired mask r would have to completely conceal all coefficients but one, and that coefficient should have all its bits obscured except the LSB.

Protocol 4 **MaskDecrypt**(sk, $\mathbf{c}_t$)$\longrightarrow$ m_{LSB}

Let $sk = s$ a secret key, $\mathbf{c} = (c_0, c_1)$ a ciphertext.
Sample $r \leftarrow \mathcal{R}_{[R_t]}$
Compute $m_{LSB_q} = [c_0 + sc_1 + \Delta r]_q$
Output $m_{LSB} = \left[\left\lfloor m_{LSB_q} * t/q \right\rceil \right]_t$

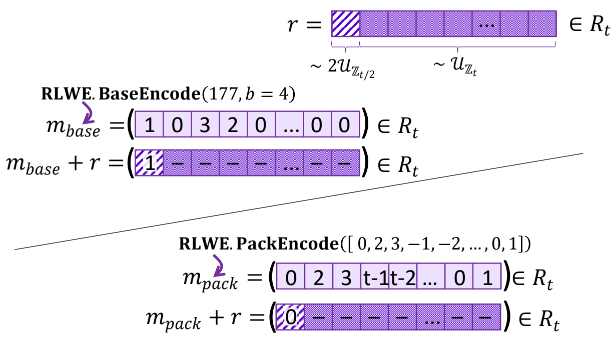

Figure 4: Visualization of masking in the plaintext domain for the arbitrary encodings of Fig. 1

Notice how the mask for packed encoding and the mask for even-based encoding would look very similar. For our desired masking polynomial r, we draw $N-1$ coefficients as $r[j] \sim \mathcal{U}\left(\mathbb{Z}_{[0,t)}\right) \quad \forall j \neq 0$, and one single coefficient from $r[0] \sim 2\mathcal{U}\left(\mathbb{Z}_{[0,t/2)}\right)$ to preserve the LSB.[3] We define this mask distribution as $\mathcal{R}_{[R_t]}$.

If we introduce this mask r in the right hand side of Equation 2, we would achieve our desired functionality. To balance the equation, we add Δr in the other side.

$$r \leftarrow \mathcal{R}_{[R_t]} \quad m_{LSB} = [m + (r)]_t$$
$$LSB = (Decode(m_{LSB})[0]) \bmod 2 \tag{3}$$

$$\left[\left\lfloor \frac{t}{q} \left(\Delta(m+r) + e_{c_t}\right)\right\rceil\right]_t = [\lfloor (m+r) + at + v \rceil]_t \tag{4}$$

Since $\|\Delta r\| < q$, we can introduce this mask in Equation 1:

$$[c_0 + sc_1 + \Delta r]_q = \Delta m + e_{c_t} + \Delta r \tag{5}$$

We remark that adding masking might cause $m[0] + r[0] \geq t$, in which case the modulo operation would kick off and flip the LSB (recall that t is prime and thus an odd number). The following limitation is imposed for the coefficient $j = 0$ containing the LSB[4]:

$$m[0] + r[0] < t \tag{6}$$

This limitation effectively imposes $m[0] \in \{0, 1\}$ to avoid the LSB flip and preserve correctness. To overcome it, the (mod t) operation could be approximated by (mod $(t-1)$) with $(t-1)$ being an even number, ensuring the LSB preservation after applying modular reduction. Translated into the R_q domain, mod q reductions during and after decryption would then be approximated by mod q' with $q' = \Delta * (t-1) = q - \Delta$.

3.4 Collaborative Masked Decryption

Extending the masked decryption to a multi-party collaborative setting requires merging Protocol 4 with DBFV.ColDecrypt. To do so we require the sampling of $r \leftarrow \mathcal{R}_{[R_t]}$ to be handed to the different parties in the form of shares $\langle r \rangle_i$. In addition, we encrypt the mask shares. For convenience and performance, we swap the order, first encrypting the global masking polynomial $\mathbf{c}_r$ = BFV.Encrypt(r, pk) and then splitting it into shares in the encrypted domain by adding encoded secret shares of zero.

Thanks to the standard properties of arithmetic secret sharing [49], you require the results of all the K parties to reconstruct the correct masking polynomial r, and any sum of shares from less than K parties $\sum_i^{<K} \langle r \rangle_i$ is indistinguishable from a uniformly random sampling $r' \leftarrow \mathcal{U}(R_q)$.

The full ColMaDe decryption is outlined in Protocol 5.

Protocol 5 ColMaskDecr(c, $\langle sk \rangle_1, \ldots \langle sk \rangle_K) \rightarrow m_{LSB}$

Let $\langle sk \rangle_i$ the private share of a global secret key $s = \sum_i^K \langle sk \rangle_i$; $\mathbf{c} = (c_0, c_1)$ a ciphertext; $r \leftarrow \mathcal{R}_{[R_t]}$ a masking polynomial encrypted with BFV.Encrypt$(r, pk) \rightarrow \mathbf{c}_r = (c_{r_0}, c_{r_1})$ and splitting c_{r_0} in K shares $\langle c_{r_0} \rangle_i$

Setup: P_i holds s_i, $\langle c_{r_0} \rangle_i$ and c_{r_1}.

P_i: Sample $e_i \leftarrow \mathcal{X}_{[R_q]}$

P_i: Compute $\langle c_{1s} \rangle_i = \left[\langle sk \rangle_i (c_1 + c_{r_1}) + e_i + \langle c_{r_0} \rangle_i\right]_q$

Any: Output $m_{LSB} = \left[\left\lfloor \frac{t}{q} \left[c_0 + \sum_i^K \langle c_{1s} \rangle_i\right]_q \right\rceil\right]_t$

3.5 Replicated Masked Decryption

We can customize the ColMaDe protocol under a malicious setting by sending/setting J shares of the secret key $\langle sk \rangle_i$ and J shares of the first polynomial of the encrypted mask $\langle c_{r_0} \rangle_i$ to each party P_i, in a replicated sharing scheme. Protocol 6 details this modification.

In this replicated setting, each party would compute J shares of $\langle c_{1s} \rangle_i$, whose individual decryptions should yield uniformly random yet equal results for the same input shares of sk and c_{r_0}. By comparing these auxiliary decryptions, the party in charge of aggregating all the results and outputting m can detect up to $J-1$ parties deviating from the protocol, thus allowing the aggregator to *detect* these malicious adversaries and *abort* the decryption. However, this replication technique lowers the number of parties required to reconstruct the entire mask from K to at least $K - J$, thus making this technique suitable only for when a majority of parties is honest. We study the relation between J and the number of malicious corruptions in the pool $|\mathcal{A}_{\mathcal{P}}|$ in section 4.4.

4 SECURITY ANALYSIS

The ColMaDe protocol seeks to guarantee the privacy of all bits but a single LSB in a ciphertext. The underlying real-world motivation in the biometrics domain was mentioned already: the most precious resource in our system is the database of reference templates, since they are often tightly linked to the person, and thus non revocable like passwords or tokens. Hence, in scenarios where the reference templates are used for multiple applications, its theft could lead not only to a potential impersonation when accessing the desired service/premise, but to a severe identity theft across applications.

To study the security of our protocol, we generalize that an adversary corrupting the Gate also corrupts the BIP, which means that this combined corruption would grant the adversary full access to the encrypted database and can perform chosen ciphertext attacks

[3] In the packed encoding we could chose to preserve the LSB of an arbitrarily chosen coefficient j. We set $j = 0$ for convenience.

Protocol 6 **ReplColMaskDecr(** $\langle sk\rangle_1 \ldots \langle sk\rangle_K, \mathbf{c}, J) \rightarrow \mathrm{m}_{LSB}$

Let $s_i = \langle sk\rangle_i$ private shares of the global secret key $s = \sum s_i$; $\mathbf{c} = (c_0, c_1)$ a ciphertext; $r \leftarrow \mathcal{R}_{[R_t]}$ a masking polynomial encrypted with $\mathsf{BFV.Encrypt}(r, pk) \rightarrow \mathbf{c}_r = (c_{r_0}, c_{r_1})$ and secret-sharing c_{r_0} in K shares $\langle c_{r_0}\rangle_i$;
with $\{i_J\} = \{(i+j)\%K\}$ for $\{0 \cdots j \cdots J-1\}$

Setup: P_i holds J shares $s_{\{i_J\}}$, c_{r_1} and J shares $\langle c_{r_0}\rangle_{\{i_J\}}$.

P_i: Sample J times $e_{\{I\}} \leftarrow \mathcal{X}_{[R_q]}$.

P_i: Compute $\langle c_{1s}\rangle_{\{i_J\}} = \left[s_{\{i_J\}}(c_1 + c_{r_1}) + e_{\{i_J\}} + \langle c_{r_0}\rangle_{\{i_J\}}\right]_q$

Any: Check equality among all $\lfloor t/q \ \langle c_{1s}\rangle_{\{i_J\}}\rceil \ \forall i, j$.
Abort if non equal.
Output $m_{LSB} = \left[\left\lfloor \frac{t}{q}\left(c_0 + \sum_i^k \langle c_{1s}\rangle_i\right)_q\right\rceil\right]_t$

(CCA) using the pool of users as decryption oracle. We analyze several threat models:

(1) Gate is semi-honest, following the protocol but tries to extract as much information as possible), and up to $K-1$ parties in the pool are semi-honest (at least one honest user per pool).
(2) Gate is malicious, capable of deviating from the protocol arbitrarily, and up to $K-1$ parties are semi-honest (at least one honest user per pool).
(3) Gate is semi-honest, a minority of parties in the pool are malicious (at least $\lceil K/2\rceil$ honest user per pool).

4.1 On Privacy of Colmade

4.1.1 Privacy in the semi-honest pool. We first provide a security proof for the proposed Colmade protocol in the standalone passive adversary model for the pool of users, that we base on the *decision RLWE assumption*[35]. We formulate our proof using the ideal/real simulation paradigm[30]: We show that, for every possible adversarial subset $\mathcal{A}$ of all the computing parties in the pool $\mathcal{P} = \{\mathsf{P}_1, \ldots, \mathsf{P}_K\}$, there exists a simulator program S that can simulate $\mathcal{A}$'s view in the protocol, when provided only with $\mathcal{A}$'s input and output. To achieve semantic security[24], we require $\mathcal{A}$ not be able to distinguish the simulated view from the real one. Note that the view of the adversary after the setup is the full transcript (public transcript property). For a given value x, we denote $\tilde{x}$ its simulated equivalent. We consider computational indistinguishability between distributions, and denote it as $x \overset{c}{\equiv} \tilde{x}$. We denote $\mathrm{view}^{\mathsf{ColMaskDecr}}$ to the transcript of Protocol 5, consisting of all the shares $\{\langle c_{1s}\rangle_1, \ldots, \langle c_{1s}\rangle_i, \ldots, \langle c_{1s}\rangle_K\}$ of c_{1s} in R_q.

Theorem 1. *(ColMaskDecr* privacy in the semi-honest pool model*) For each possible set of corrupted parties $\mathcal{A} \subset \mathcal{P}$ by a passive adversary with $|\mathcal{A}| \leq K-1$, there exists a simulator $S^{ColMaskDecr}$ such that:*

$$S^{ColMaskDecr} \overset{c}{\equiv} view^{ColMaskDecr}$$

Proof. First, Theorem 1 forces at least one arbitrarily chosen party P_h to be honest. We denote $\mathcal{H} \triangleq \mathcal{P}\backslash(\mathcal{A} \cup \{\mathsf{P}_h\})$ to the set of all other honest parties, so that the tuple $(\mathcal{A}, \mathcal{H})$ represent any partition of $\mathcal{P}\backslash\mathsf{P}_h$. We consider the error term e_i sampled as a part of the protocols as private input to the protocol.

Simulator 1 $S^{\mathsf{ColMaskDecr}}$

Input: The simulator is given $\{\langle sk\rangle_i, \langle c_{r_0}\rangle_i, e_i\}\forall i$ and c_{r_1} by the trusted setup.
The simulator receives $\mathbf{c}_{t_1}$ from the Gate.

Output: for each party P_i in the pool:

$$\langle\widetilde{c_{1s}}\rangle_i = \begin{cases} \left[(c_{t_1} + c_{r_1})\langle sk\rangle_i + e'_i + \langle c_{r_0}\rangle_i\right]_q & \text{if } \mathsf{P}_i \in \mathcal{A} \\ \leftarrow \mathcal{U}(R_q) & \text{if } \mathsf{P}_i \in \mathcal{H} \\ \left[c_{1s} - \sum_{\mathsf{P}_i \in \mathcal{P}\backslash\mathsf{P}_h} \langle\widetilde{c_{1s}}\rangle_i\right]_q & \text{if } \mathsf{P}_i = \mathsf{P}_h \end{cases}$$

We can now consider the distribution of the simulated and real views. The *decision-RLWE* assumption suffices to prove it in the absence of the masking term, as for an adversary that does not know $\langle sk_i\rangle$ nor e'_i, we get that:

$$(\langle sk_i\rangle c_{t_1} + e'_i, c_{t_1}) \overset{c}{\equiv} (a \leftarrow R_q, c_{t_1})$$

The addition of the masking terms only increases the randomness of the first element in the tuple, thus the equation holds true. □

4.1.2 Minimum leakage of the output. When considering the threat model #1 (all semi-honest), we resort to Theorem 1 to show that the protocol itself does not reveal more than what the output does. Now we seek to prove that the output reveals only one bit.

Theorem 2. *(ColMaskDecr* 1-bit leakage of output in the semi-honest pool model*) For each possible set of corrupted parties $\mathcal{A} \subset \mathcal{P}$ by a passive adversary with $|\mathcal{A}| \leq K-1$, the protocol reveals a maximum of one bit from the encrypted message, the LSB of the first coefficient in the underlying encoded polynomial.*

Proof. Since the output $m_{LSB} = [m + r]_t$ is the result of adding a mask r to the underlying message m, and this mask contains uniformly random values in $\mathbb{Z}_{[0,t)}$ for all $j \neq 0$, we get that:

$$\begin{aligned} &P(m[j] = a \mid m_{LSB}[j] = b) = \\ &\qquad \frac{P(m[j] = a) \cdot P(m_{LSB}[j] = b | m[j] = a)}{P(r[j] = b)} = \\ &\qquad P(m[j] = a) \end{aligned} \tag{7}$$

This is because $m_{LSB}[j] \overset{c}{\equiv} r[j] \sim \mathcal{U}(\mathbb{Z}_{[0,t)})$, and thus an adversary receiving $m_{LSB}[j]$ obtains no information beyond what he already knew about $m[j]$, showing how all slots $j \neq 0$ of the message are perfectly masked.

For $j = 0$ we can proceed in a similar fashion to prove that $P(m[0] \bmod 2 = a \mid m_{LSB}[0] \bmod 2 = a) = 1$ with $a \in \{0, 1\}$ if $m[0] \in \{0, 1\}$, which requires the Gate & BIP to follow the protocol as specified (honest or semi-honest) and comply with the limitation from equation 6. At the same time, $P(m[0]/2 = b \mid m_{LSB}[0]/2 = c) = P(m[0]/2 = b)$, showing that all but the LSB of $m_{LSB}[0]$ are perfectly masked.

In the event of $m[0] \notin 0, 1$, arising from a malicious Gate & BIP submitting a chosen $\mathbf{c}_t$ (threat model #2), we get that m_{LSB} yields the correct LSB if $(b + r[0]) < t$ with probability $(t - b)/t$, and flips with probability b/t. Interestingly, all the other bits remain perfectly masked, since $P(m[0]/2 = b \mid m_{LSB}[0]/2 = c) = P(m[0]/2 = b)$ in this case too. [5]

Based on this, we can conclude that, with up to $K-1$ semi-honest parties in the pool, our protocol discloses only one bit if the protocol is followed by the Gate & BIP and less than one bit otherwise. □

4.2 On Correctness of Colmade

To ensure that the message requested by the Gate is correctly decrypted in the presence of a minority of malicious users in the pool (threat model #3), we employ the ReplColMaskDecr enhanced protocol described in Section 3.5. We resort to the replication of shares inside each pool to effectively overcome a small number of malicious users by detecting misbehavior and aborting. Following Protocol 6, a semi-honest Gate in charge of aggregating all the masked shares can detect when two replicas are different in an honest-majority pool (with some additional limitations studied in Section 4.4), and abort the decryption, potentially requesting the decryption again to a different pool.

Beyond this, the correctness of the biometric system depends on Gate & BIP following the protocol by performing the valid operations for encrypted identification and submitting the encrypted result to a pool for decryption.

4.3 On well known FHE attacks

Fully Homomorphic Encryption schemes are known to be secure against Chosen Plaintext Attacks (CPA) as a direct consequence of the indistinguishability property that is the base of their security (e.g., the *decision RLWE assumption* for BFV and CKKS schemes). However, many of these schemes fail catastrophically against Chosen Ciphertext Attacks (CCA), and they are all insecure against adaptive CCA (or CCA-2)[31].

Beyond this, an adversary with access to a decryption oracle can, with one single well-chosen query, reveal the entire secret key in the standard BFV scheme [40],being just as applicable to DBFV to extract the global secret key. Since our solution provides a sort of decryption oracle to the Gate, we analyze the impact of this attack on our system.

The attack of [40] is rooted in a malicious adversary corrupting the Gate and performing one decryption request to a decryption oracle:

1. Craft a fake ciphertext $\mathbf{c}_t = (c_{t_0}, c_{t_1}) = (\mathbf{0}, \Delta)$[6]
2. Request decryption of $\mathbf{c}_t$ to the oracle. Following equation 1, he obtains in result $p_s = [0 + \Delta * s]_q$, where s is the secret key and $e_{c_t} = 0$.
3. Locally downscale this plaintext using equation 2 to obtain an estimation of key: $\hat{s} = p_s/\Delta$.

In the case of Colmade, this attack is much less problematic. Step 2 of the attack would consist of Gate requesting a pool $\mathcal{P}$ to decrypt the result. Using equations 3 and 4, the result would be $p_s = [0 + \Delta * s + \Delta r]_q$, which in step 3 translates into $\hat{s} = p_s/\Delta + r$. The crucial difference is the mask addition. For all $j \in [1, N-1]$ save the first coefficient $j = 0$, this translates into $s[j] + r[j]$, where $r[j] \sim \mathcal{U}(\mathbb{Z}_{[0,t)})$ acts as a perfect one-time pad over the underlying plaintext space $[0, t)$, completely hiding that secret key coefficient. For $j = 0$ this leaks one bit of $s[0] \sim \mathcal{U}(\{-1, 0, 1\})$, thus $s[0]$ would be recoverable with two queries. Most importantly, the coefficient of s being leaked is always $j = 0$, as the other coefficients are perfectly hidden and no rotations are applied as part of the decryption, which would hypothetically help to extract other coefficients.

While CCA attacks are still feasible, our single-bit output coupled with the auditability property makes these attacks[7] much less practical due to the number of malicious requests required.

4.4 On the choice of users, pools and replicas

In principle the choice of pool for each decryption request of the Gate must be a random choice among all the pools available. We propose the use of a Verifiable Delay Function [7] with a chosen random beacon [8, 15] to guide the choice of pool for every decryption, as a way to have an unbiased choice and facilitate auditability.

The number of users per pool K is left open, knowing that in practice bigger pools are harder to manage (e.g., bigger delay) and more error prone, but also more theoretically secure, since per Theorem 1 it in the real world it increases the chances of including one honest user needed to preserve the privacy of the collaborative decryption. The choice of users for each pool should ideally also be random, and it is included in the trusted setup.

The ReplColMaskDecr protocol can used to address threat model #3. If so, the number of replicas per user J should be set such that the adversary cannot reconstruct the secret key. This leads to:

$$|\mathcal{A}_{\mathcal{P}}| * J < K \tag{8}$$

To support a maximum number of malicious users in the pool $|\mathcal{A}_{\mathcal{P}}|$, J should thus be set small. However, to ensure that each replica is at least in the hands of one honest user we require

$$J \geq |\mathcal{A}_{\mathcal{P}}| + 1 \tag{9}$$

Hence, to ensure security in an honest majority, we limit the maximum number of malicious corruptions to:

$$|\mathcal{A}_{\mathcal{P}}| < \lfloor \sqrt{K} \rfloor \tag{10}$$

4.5 On Auditability of Colmade

An external auditor could enroll as a user in the pool to take part in the Colmade decryption protocol, and since the protocol requires communication with all parties, an eavesdropping auditor with access to the public transcript of these communications would know about all decryption requests.

The biometric solution based on Colmade can thus be audited by an external entity with the following items:

- The public transcript of all the communications between Gate and users testify of the number of decryption requests performed. Following 2, the auditor could infer an upper bound on the number of bits extracted from the database if all the decryption requests contained maliciously crafted ciphertexts.
- Since our solution does contain a trusted setup, an auditor suspecting malfeasance could request access to the secret key material to open some decryption requests.

[5] If using the modulo approximation hinted at the end of section 3.3, the LSB would be retrieved correctly $\forall\ m[0] \in \mathbb{Z}_{[0,t)}$, covering threat models #1 and #2.

[6] All the coefficients in polynomial c_{t_0} set to 0 and in polynomial c_{t_1} set to Δ.

[7] An overview of FHE key recovery attacks can be found in [19].

- For the choice of the pools, a Verifiable Delay Function with some well chosen random beacon could serve as an unbiased yet auditable way for the Gate to select what pool to use for each decryption request.

5 IMPLEMENTATION

We implement the COLMADE protocol on top of the Lattigo [20] homomorphic encryption library, including examples of usage and correctness checks. Our Golang implementation is open-sourced in https://github.com/ibarrond/colmade.

Simply encoding each vector element into an upscaled coefficient in R_q would lead to slower ring operations when $q < 2^{64}$, as it requires arbitrary-precision arithmetic that is much more computationally expensive than standard integer arithmetic in a 64-bit machine. In practice[20], vectors are encoded to polynomials using the Chinese Remainder Theorem (CRT) into a Residue Numeral System (RNS) form [3] by decomposing $q = q_1 \cdot q_2 \cdot \ \ldots \ \cdot q_l$ into coprime factors smaller yet close to machine word size of 2^{64}.

While all our protocols apply in the RNS variant of BFV, it is worth noting that the modulo approximation of Section 3.3 could be applied to a single factor, $q' = q_1' \cdot q_2 \cdot \ \ldots \ \cdot q_l$ with $q_1' = (q_1 - q_1/\Delta)$.

6 PREVIOUS WORK

Preceding work employing masking for RLWE instances has been focused on protecting the secret key during decryption operations, to upgrade the Chosen-Plaintext-Attack (CPA) security guarantees of RLWE cryptosystems into Chosen-Ciphertext-Attack (CCA1/CCA2). In this line, additively homomorphic masking was proposed to output a a secret-shared result that will later be reconstructed during decoding [43], and an follow-up work proposed a decryption outputting boolean shares suitable for derivation of a symmetric key to be used during decoding [44]. Further down the line, [37] proposes an adaptation of RLWE schemes to render them CCA2-Secure based on a post-quantum variant of the Fujisaki-Okamoto (FO) transform combined with masked binomial sampling to secure a re-encryption process.

The idea of masking in HE has also been studied previously in the form of slot masking, a method to collapse multiple unique-repeated-value ciphertexts into a single ciphertext for encrypted vector-matrix multiplication: multiply each ciphertext with a mask containing a 1 set in a chosen slot and 0 in all the other slots. We saw this technique applied for HE-based applications in the context of phishing web page classification[14], and in HE-Friendly privacy-preserving mobile neural network architectures[33].

In other line of works, FHE has been widely studied as a technique for privacy-preserving biometrics, from the HE-based biometric access control system of [34], to the packing technique of [56], or [52] showing a clever encoding using packing to perform a biometric matching with one single homomorphic multiplication. [4] used Homomorphic Encryption for fingerprint biometrics, whereas [18] employed both CKKS and BFV for face identification, and [25] proposed the protection of a multi-biometric system.

While there are many previous works studying secure biometrics with MPC[21] and FHE[47], to the best of our knowledge this is the first work to contemplate the intersection of multiparty homomorphic encryption [17, 36] with biometrics. Lastly, while the vanilla DBFV decryption of [36] would already provide auditability and data privacy against a semi-honest adversary, our work extends it to malicious corruptions and yields minimum input leakage thanks to the collaborative masking embedded in the decryption protocol.

7 CONCLUSIONS

COLMADE proposes a novel collaborative masking decryption protocol for the multiparty BFV scheme guaranteeing data privacy, minimal output leakage (1 bit), and auditability. Our protocol makes use of predefined pools of users to perform a decryption in a distributed setting while adding an additively shared encrypted masking term. We showcase its applicability as part of a biometric access control solution where groups of users get together for orderly individual identification. We prove this protocol secure against $K - 1$ corruptions of a semi-honest adversary, and show an enhanced version using replicas to be resilient against $\lfloor \sqrt{K} \rfloor$ active corruptions of a malicious adversary. We analyze practical security aspects of the biometric solution, and open-source implementations of these protocols on top of the Lattigo library.

REFERENCES

[1] Toshinori Araki, Jun Furukawa, Yehuda Lindell, Ariel Nof, and Kazuma Ohara. 2016. High-throughput semi-honest secure three-party computation with an honest majority. In *Proceedings of the 2016 ACM SIGSAC CCS Conference*. 805–817.

[2] Multiple authors. 2018. *Homomorphic Encryption Security Standard.* Technical Report. HomomorphicEncryption.org, Toronto, Canada.

[3] Jean-Claude Bajard, Julien Eynard, M Anwar Hasan, and Vincent Zucca. 2016. A full RNS variant of FV like somewhat homomorphic encryption schemes. In *International Conference on Selected Areas in Cryptography*. Springer, 423–442.

[4] Mauro Barni, Tiziano Bianchi, Dario Catalano, Mario Di Raimondo, Ruggero Donida Labati, Pierluigi Failla, Dario Fiore, Riccardo Lazzeretti, Vincenzo Piuri, Alessandro Piva, et al. 2010. A privacy-compliant fingerprint recognition system based on homomorphic encryption and fingercode templates. In *2010 Fourth IEEE International Conference on Biometrics: Theory, Applications and Systems (BTAS)*. IEEE, 1–7.

[5] Carsten Baum, Ivan Damgård, and Claudio Orlandi. 2014. Publicly auditable secure multi-party computation. In *International Conference on Security and Cryptography for Networks*. Springer, 175–196.

[6] Alexandre Bois, Ignacio Cascudo, Dario Fiore, and Dongwoo Kim. 2021. Flexible and efficient verifiable computation on encrypted data. In *IACR International Conference on Public-Key Cryptography*. Springer, 528–558.

[7] Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. 2018. Verifiable delay functions. In *Annual international cryptology conference*. Springer, 757–788.

[8] Joseph Bonneau, Jeremy Clark, and Steven Goldfeder. 2015. On bitcoin as a public randomness source. *Cryptology ePrint Archive* (2015).

[9] Elette Boyle, Niv Gilboa, and Yuval Ishai. 2015. Function secret sharing. In *EUROCRYPT*. Springer, 337–367.

[10] Zvika Brakerski. 2012. Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP. In *Advances in Cryptology – CRYPTO 2012*. Springer Berlin Heidelberg, 868–886. https://doi.org/10.1007/978-3-642-32009-5_50

[11] Centers for Medicare & Medicaid. 1996. The Health Insurance Portability and Accountability Act of 1996 (HIPAA). Online at http://www.cms.hhs.gov/hipaa/.

[12] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. 2017. Homomorphic Encryption for Arithmetic of Approximate Numbers. In *Advances in Cryptology – ASIACRYPT 2017*. Springer International Publishing, 409–437. https://doi.org/10.1007/978-3-319-70694-8_15

[13] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. 2020. TFHE: Fast Fully Homomorphic Encryption Over the Torus. *Journal of Cryptology. The Journal of the International Association for Cryptologic Research* 33, 1 (1 Jan. 2020), 34–91. https://doi.org/10.1007/s00145-019-09319-x

[14] Edward J Chou, Arun Gururajan, Kim Laine, Nitin Kumar Goel, Anna Bertiger, and Jack W Stokes. 2020. Privacy-preserving phishing web page classification via fully homomorphic encryption. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2792–2796.

[15] Jeremy Clark and Urs Hengartner. 2010. On the use of financial data as a random beacon. In *2010 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE 10)*.

[16] European Commission. [n.d.]. *2018 reform of EU data protection rules.* https://gdpr-info.eu/

[17] Ivan Damgård, Valerio Pastro, Nigel Smart, and Sarah Zakarias. 2012. Multiparty computation from somewhat homomorphic encryption. In *Annual Cryptology Conference*. Springer, 643–662.

[18] Pawel Drozdowski, Nicolas Buchmann, Christian Rathgeb, Marian Margraf, and Christoph Busch. 2019. On the application of homomorphic encryption to face identification. In *2019 International Conference of the Biometrics Special Interest Group (BIOSIG)*. IEEE, 1–5.

[19] Keita Emura. 2021. On the Security of Keyed-Homomorphic PKE: Preventing Key Recovery Attacks and Ciphertext Validity Attacks. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* 104, 1 (2021), 310–314.

[20] EPFL-LDS. 2022. Lattigo v2.4.0. Online: https://github.com/ldsec/lattigo.

[21] Diana-Elena Fălămaş, Kinga Marton, and Alin Suciu. 2021. Assessment of Two Privacy Preserving Authentication Methods Using Secure Multiparty Computation Based on Secret Sharing. *Symmetry* 13, 5 (2021), 894.

[22] J Fan and F Vercauteren. 2012. Somewhat Practical Fully Homomorphic Encryption. *IACR Cryptology ePrint Archive* (2012). https://eprint.iacr.org/2012/144

[23] Craig Gentry et al. 2009. *A fully homomorphic encryption scheme*. Vol. 20. Stanford.

[24] Oded Goldreich. 2009. *Foundations of cryptography: volume 2, basic applications*. Cambridge university press.

[25] Marta Gomez-Barrero, Emanuele Maiorana, Javier Galbally, Patrizio Campisi, and Julian Fierrez. 2017. Multi-biometric template protection based on homomorphic encryption. *Pattern Recognition* 67 (2017), 149–163.

[26] Shai Halevi and Victor Shoup. 2020. Design and implementation of HElib: a homomorphic encryption library. Cryptology ePrint Archive, Report 2020/1481. https://eprint.iacr.org/2020/1481

[27] Alberto Ibarrondo, Hervé Chabanne, and Melek Önen. 2021. Practical Privacy-Preserving Face Identification based on Function-Hiding Functional Encryption. In *International Conference on Cryptology and Network Security*. Springer, 63–71.

[28] Ilia Iliashenko and Vincent Zucca. 2021. Faster homomorphic comparison operations for BGV and BFV. *Proceedings on Privacy Enhancing Technologies* 2021, 3 (2021), 246–264.

[29] Yuval Ishai, Eyal Kushilevitz, Sigurd Meldgaard, Claudio Orlandi, and Anat Paskin-Cherniavsky. 2013. On the power of correlated randomness in secure computation. In *Theory of Cryptography Conference*. Springer, 600–620.

[30] Yehuda Lindell. 2017. How to simulate it–a tutorial on the simulation proof technique. *Tutorials on the Foundations of Cryptography* (2017), 277–346.

[31] Jake Loftus, Alexander May, Nigel P Smart, and Frederik Vercauteren. 2011. On CCA-secure somewhat homomorphic encryption. In *International Workshop on Selected Areas in Cryptography*. Springer, 55–72.

[32] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. 2012. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*. 1219–1234.

[33] Qian Lou and Lei Jiang. 2021. HEMET: A Homomorphic-Encryption-Friendly Privacy-Preserving Mobile Neural Network Architecture. In *International Conference on Machine Learning*. PMLR, 7102–7110.

[34] Ying Luo, S Cheung Sen-ching, and Shuiming Ye. 2009. Anonymous biometric access control based on homomorphic encryption. In *2009 IEEE International Conference on Multimedia and Expo*. IEEE, 1046–1049.

[35] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. 2010. On ideal lattices and learning with errors over rings. In *Annual international conference on the theory and applications of cryptographic techniques*. Springer, 1–23.

[36] Christian Mouchet, Juan Troncoso-Pastoriza, Jean-Philippe Bossuat, and Jean-Pierre Hubaux. 2020. Multiparty homomorphic encryption from ring-learning-with-errors. *Cryptology ePrint Archive* (2020).

[37] Tobias Oder, Tobias Schneider, Thomas Pöppelmann, and Tim Güneysu. 2016. Practical CCA2-secure and masked ring-LWE implementation. *Cryptology ePrint Archive* (2016).

[38] Pascal Paillier. 1999. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *Advances in Cryptology — EUROCRYPT '99*. Springer Berlin Heidelberg, 223–238. https://doi.org/10.1007/3-540-48910-X_16

[39] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. 2017. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*. 506–519.

[40] Zhiniang Peng. 2019. Danger of using fully homomorphic encryption: A look at microsoft SEAL. *arXiv preprint arXiv:1906.07127* (2019).

[41] Yuriy Polyakov, Kurt Rohloff, and Gerard W Ryan. 2017. *PALISADE lattice cryptography library user manual*. Technical Report. NJIT. https://git.njit.edu/palisade/PALISADE/wikis/resources/palisade_manual.pdf

[42] Oded Regev. 2005. On Lattices, Learning with Errors, Random Linear Codes, and Cryptography. In *In STOC*. ACM Press, 84–93.

[43] Oscar Reparaz, Ruan de Clercq, Sujoy Sinha Roy, Frederik Vercauteren, and Ingrid Verbauwhede. 2016. Additively homomorphic ring-LWE masking. In *Post-Quantum Cryptography*. Springer, 233–244.

[44] Oscar Reparaz, Sujoy Sinha Roy, Ruan De Clercq, Frederik Vercauteren, and Ingrid Verbauwhede. 2016. Masking ring-LWE. *Journal of Cryptographic Engineering* 6, 2 (2016), 139–153.

[45] Ronald L Rivest, Len Adleman, and Michael L Dertouzos. 1978. On Data Banks and Privacy Homomorphisms. *Foundations of secure computation* 4, 11 (1978), 169–180. https://people.csail.mit.edu/rivest/RivestAdlemanDertouzos-OnDataBanksAndPrivacyHomomorphisms.pdf

[46] R L Rivest, A Shamir, and L Adleman. 1978. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM* 21, 2 (1 Feb. 1978), 120–126. https://doi.org/10.1145/359340.359342

[47] Zhang Rui and Zheng Yan. 2018. A survey on biometric authentication: Toward secure and privacy-preserving identification. *IEEE access* 7 (2018), 5994–6009.

[48] SEAL 2020. Microsoft SEAL (release 3.6). https://github.com/Microsoft/SEAL. Microsoft Research, Redmond, WA.

[49] Adi Shamir. 1979. How to share a secret. *Comm. of the ACM* 22, 11 (1979), 612–613.

[50] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*. IEEE, 3–18.

[51] Nigel P Smart and Frederik Vercauteren. 2014. Fully homomorphic SIMD operations. *Designs, codes and cryptography* 71, 1 (2014), 57–81.

[52] Hiroto Tamiya, Toshiyuki Isshiki, Kengo Mori, Satoshi Obana, and Tetsushi Ohki. 2021. Improved Post-quantum-secure Face Template Protection System Based on Packed Homomorphic Encryption. In *2021 International Conference of the Biometrics Special Interest Group (BIOSIG)*. IEEE, 1–5.

[53] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. 2016. Stealing machine learning models via prediction apis. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*. 601–618.

[54] Paulo Vitorino, Sandra Avila, Mauricio Perez, and Anderson Rocha. 2018. Leveraging deep neural networks to fight child pornography in the age of social media. *Journal of Visual Communication and Image Representation* 50 (2018), 303–313.

[55] Andrew Chi-Chih Yao. 1986. How to generate and exchange secrets. In *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*. IEEE, 162–167.

[56] Masaya Yasuda, Takeshi Shimoyama, Jun Kogure, Kazuhiro Yokoyama, and Takeshi Koshiba. 2013. Packed homomorphic encryption based on ideal lattices and its application to biometrics. In *International Conference on Availability, Reliability, and Security*. Springer, 55–74.