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.

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

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

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

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...
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 multidisciplinary 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 explains 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 \( D \), written as \( a[j] \sim D \), we denote the sampling of a polynomial \( a \) over a ring \( R_z \) as \( a \leftarrow D_{R_z} \). A computing party is denoted as \( P_i \) in a pool of \( K \) parties \( P = \{ P_1, ..., P_i, ..., P_K \} \).

We use \( (a)_i \) to refer to share \( i \) in an arithmetic secret sharing of \( a = \sum^K \langle a \rangle \). We denote \([\cdot\]_q the reduction modulo \( q \), and \( [\cdot], [\cdot] \), \( [\cdot] \) 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 \( \mu \) and standard deviation \( \sigma \). For a polynomial \( a \), we write its infinity norm as \( ||a|| \). For an input integer \( x \in \mathbb{Z} \) we use \( \text{sign}(x) = x/|x| \), 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., RingLWE 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 (RLWEE)[35] homomorphic encryption scheme. Messages are encoded in the plaintext space $R_I = \mathbb{Z}/(X^N + 1)$ of polynomials of degree up to $N - 1$, and then encrypted into the ciphertext space $R_q = \mathbb{Z}/(X^N + 1)$ with $t < q$ (typically $t \ll q$), $N$ a power of 2 and $\Lambda = [q/t]$. The BFV scheme utilizes secrets sampled from two small-normed schemes that are pertinent for this work. We will now focus on the collective public key

#### 2.2.1 Decryption in BFV.

The distributed secret-key generation protocol yields a global secret key $sk$ whose coefficients, as a sum of $K$ samples of $R_q$, add up to a maximum of $\|sk\| \leq K$. As a result of BFV.Decrypt protocol, the collective public key $c_{pk}$ contains noise $e_{c_{pk}} = \sum_i e_i$, implying that $\|e_{c_{pk}}\| \leq K \cdot B_f$. Thus, a freshly encrypted ciphertext $c_m = (c_{m_e}, c_{m_c})$ of a message $m$ under a collective public key $c_{pk}$ will decrypt, following equation 1 with the single-party BFV.Decrypt, to $[c_{m_e} + sc_{m_c}]_q = \Delta m + e_{fresh}$, where $\|e_{fresh}\| \leq Bf(2NK + 1)$. Thus, the worst-case fresh ciphertext noise is linear in the number of parties $K$.
Scheme 2  DBFV(t, N, q, σ, B, K)

DBFV.SecKeyGen() → ⟨sk₁, ..., skₖ⟩ₖ .
P₁: SAMPLE s₁ ← Σ[Rₑ₁].
P₁: OUTPUT (sk₁) = s₁ where sk = [Σₖ sₖ]ᵣ

DBFV.ColKeyGen(⟨sk₁, ..., skₖ⟩ₖ) → cpk :
Let skᵢ = sᵢ private key share of Pᵢ.
Any: SAMPLE p₁ ← U(Rₑ₁). Disclose to all Pᵢ.
P₁: SAMPLE eᵢ ← Σ[Rₑ₁].
P₁: COMPUTE (p₀)ᵢ = −pₛᵢ + eᵢ.
Any: OUTPUT cpk = (p₀, p₁) = (Σₖ (p₀)ᵢ, p₁).

DBFV.Encrypt( cpk, m ) → cmₖ :
Any: OUTPUT cmᵢ = DBFV.Encrypt(cpk, m).

DBFV.CollDecrypt( cpk, m ) → mᵢₖ :
Let sᵢ = (skᵢ) private key share of Pᵢ, cᵢₖ = (cᵢ₀, cᵢ₁) a ciphertext.
Pᵢ: SAMPLE eᵢ ← Σ[Rₑ₁].
Pᵢ: COMPUTE (c₁₁)ᵢ = sᵢ eᵢ + eᵢ.
Any: OUTPUT mᵢᵢₖ = [q⁻¹ (cᵢ₀ + Σₖ (c₁₁)ᵢ)]ᵣ

Conversely, the freshly encrypted ciphertext cmᵢ = (cmᵢ₀, cmᵢ₁) of a message m under a single-party public key pk will generate, following equation 1 and using the multi-party DBFV.CollDecrypt, a similar error term, which then doubles if both DBFV.CollDecrypt 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 Rₑ. 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 one element, 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 Integer Number Theoretic Transform (InvNTT) over Rₑ 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 Z or R. Their coercion to arithmetic modulo t forces the underlying plaintext operations to not overflow their coefficients modulo t. Hence, the

\[ a = \frac{17710}{10} = 20314 \]

\[ a \in \mathbb{Z} \text{ in base } b = 4 \]

\[ m_{base} = \begin{pmatrix} 1 & 0 & 3 & 2 & 0 & \ldots & 0 \end{pmatrix} \]

\[ m_{rlwe} = \begin{pmatrix} 0 & 2 & 3 & 1 & 1 & \ldots & 0 & 1 \end{pmatrix} \]

\[ \mathbb{Z}^N \]

\[ \mathbb{R}_t \]

\[ a_{dec} = \begin{pmatrix} 0 & 2 & 3 & 0 & 1 & \ldots & 1 \end{pmatrix} \]

\[ m_{pack} = \begin{pmatrix} 0 & 2 & 3 & 1 & 1 & \ldots & 0 & 1 \end{pmatrix} \]

\[ \mathbb{R}_t \]

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

Scheme 3  RLWE Code(t, N)

RLWE.BaseEncode( a, b ) → m:
Let a ∈ Z an input integer value with up to N digits in base-b representation.
Output: Polynomial m ∈ Rₑ with
\[ m[j] = (|a|_{b, j+1} - |a|_{b, j}) \forall j \text{ for unsigned encoding,} \]
\[ m[j] = (|a|_{b, j+1} - |a|_{b, j}) \cdot sign(a) + t \cdot sgb(a) \forall j \text{ for signed encoding.} \]

RLWE.BaseDecode( m, b ) → aᵣₑ:
Let m ∈ Zᴺ the coefficients of an encoded polynomial in Rₑ.
Output Integer aᵣₑ ∈ Z with
\[ aᵣₑ = \sum_{i=1}^{N} m[i] \cdot b^{i(t-1)} \] for unsigned encoding,
\[ aᵣₑ = \sum_{i=1}^{N} m[i] \cdot b^{i(t-1)} \cdot (-1) \cdot sign(m[i] - t/2) \] for signed encoding.

RLWE.PackEncode( a ) → m:
Let a ∈ Zᴺ an input vector with N elements in:
\[ \begin{pmatrix} 0, t \end{pmatrix} \text{ for unsigned encoding,} \]
\[ \begin{pmatrix} -t/2, t/2 \end{pmatrix} \text{ for signed encoding.} \]
where t must be a prime congruent to 1 mod 2N.
Output Polynomial m ∈ Rₑ where m = InvNTT(a mod t).

RLWE.PackDecode( m ) → aᵣₑ:
Let m ∈ Zᴺ the coefficients of an encoded polynomial of degree N - 1 in Rₑ.
Output aᵣₑ = NTT(m).
Output
\[ aᵣₑ[i] = aᵣₑ \text{ for unsigned encoding,} \]
\[ aᵣₑ[i] = (aᵣₑ[i] - t) \text{ if } aᵣₑ[i] < t/2 else(aᵣₑ[i] - 1) \] 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.

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.

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.

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.
- $P_1 \ldots P_i \ldots P_K$. 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 these steps:

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

---

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

2Foreseeing 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.
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 only on the polynomial coefficient, 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**

\[
\text{MaskDecrypt}(sk, c_t) \rightarrow m_{\text{LSB}}
\]

Let $sk = s$ a secret key, $c = (c_0, c_1)$ a ciphertext.

**SAMPLE** $r \leftarrow \mathbb{R}_{[0, 1]}

**COMPUTE** $m_{\text{LSB}} = \left[ c_0 + c_1 + Ar \right]q$

**OUTPUT** $m_{\text{LSB}} = \left[ m_{\text{LSB}} + t/q \right]_t$
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 \( r[j] \sim \mathcal{U}(\mathbb{Z}[0, q]) \) \( \forall j \neq 0 \), and one single coefficient from \( r[0] = 2^t \mathcal{U}(\mathbb{Z}[0, t/2]) \) to preserve the LSB. We define this mask distribution as \( \mathcal{R}(r_i) \).

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 \( \Delta r \) in the other side.

\[
\begin{align*}
    r & \leftarrow \mathcal{R}(r_i) \\
    m_{\text{LSB}} & = \lfloor m + r \rfloor_t \\
    \text{LSB} & = (\text{Decode}(m_{\text{LSB}}[0])) \mod 2
\end{align*}
\]

(3)

\[
\left[ \frac{t}{q} \left( \Delta(m + r) + c_i \right) \right]_t \equiv \left[ \lfloor (m + r) + at + v \rfloor_t \right]_t
\]

(4)

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

\[
\lfloor c_0 + sc_1 + \Delta r \rfloor_q = \Delta m + c_i + \Delta r
\]

(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:\

\[
m[0] + r[0] < t
\]

(6)

This limitation effectively imposes \( m[0] \in \{0, 1\} \) to avoid the LSB flip and preserve correctness. To overcome it, the \( t \) operation could be approximated by \( \left( \mod (t-1) \right) \) with \( t-1 \) being an even number, ensuring the LSB preservation after applying modular reduction. Translated into the \( \mathcal{R} \) 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 \( \text{DBFV.ColDecrypt} \). To do so we require the sampling of \( r \leftarrow \mathcal{R}(r_i) \) to be handed to the different parties in the form of shares \( (r_i) \). In addition, we encrypt the mask shares. For convenience and performance, we swap the order, first encrypting the global masking polynomial \( c_r = \text{BFV.Encrypt}(r, pk) \) 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 \( i \leq K(r) \) is indistinguishable from a uniformly random sampling \( r' \leftarrow \mathcal{U}(\mathcal{R}_q) \).

The full \( \text{ColMADE} \) decryption is outlined in Protocol 5.

Protocol 5 ColMaskDecr(c, (sk)_1 \ldots (sk)_K) \rightarrow m_{\text{LSB}}

Let \( (sk)_i \) the private share of a global secret key \( s = \sum_{i}^K (sk)_i \); c = (c0, c1) a ciphertext; \( r \leftarrow \mathcal{R}(r_i) \) a masking polynomial encrypted with \( \text{BFV.Encrypt}(r, pk) \) to \( c_r = (c_{r_0}, c_{r_1}) \) and splitting \( c_{r_0} \) in \( K \) shares \( (c_{r_i})_i \). \( \text{Setup: P_i} \) holds \( s_i \), \( (c_{r_i})_i \) and \( c_{r_1} \).

\( \text{P_i: Sample } c_i \leftarrow X \left[ \mathcal{R}_q \right] \)

\( \text{P_i: Compute } \langle c_{1x_i} \rangle = \bigg[ \langle (sk)_i \rangle (c_1 + c_{r_1}) + e_i + \langle c_{r_1} \rangle q \bigg]_q \)

\( \text{Any: Output } m_{\text{LSB}} = \bigg[ \frac{q}{t} \lfloor c_0 + \sum_{i}^K \langle c_{1x_i} \rangle q \rfloor q \bigg]_q \)

3.5 Replicated Masked Decryption

We can customize the \( \text{ColMADE} \) protocol under a malicious setting by sending/setting \( J \) shares of the secret key \( (sk)_j \) and \( J \) shares of the first polynomial of the encrypted mask \( \langle c_{r_1} \rangle \) 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 \( \langle c_{1x_i} \rangle \), whose individual decryptions should yield uniformly random yet equal results for the same input shares of \( sk \) and \( c_{r_1} \). 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 \( |\mathcal{A}_P| \) in section 4.4.

4 SECURITY ANALYSIS

The \( \text{ColMADE} \) 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.
Let $s_k = (sk_j)$ private shares of the global secret key $s = \sum s_i$; $c = (c_0, c_1)$ a ciphertext; $r \leftarrow R_{\mathbb{R}_1}$ a masking polynomial encrypted with BFF. Encrypt $(r, pk) \rightarrow c_r = (c_{r_0}, c_{r_1})$ and secret-sharing $c_{r_0}$ in $K$ shares $(c_{r_{i_0}})$; with $\{j\} = \{(i + j)\%K\}$ for $\{0 \cdots J - 1\}$.

**Setup:** $P_1$ holds $j$ shares $s_{i_1}, c_0$, and $f$ shares $s_{i_0} \{i\}$. $P_1$: sample $f$ times $c_{i_0} \leftarrow X_{\mathbb{R}_1}$; $P_1$: compute $(c_{i_0})_i = s_{i_1} + c_{r_0} + c_{r_1} \{i\}$; Any: check equality among all $[t/q] (c_{i_0})_i \forall i, j$. Abort if non equal.

**Output** $m_{LSB} = \left[ \frac{s_{i_1}}{q} + \sum c_{i_0} \{i\} \right]_t$.

(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 $[K/2]$ 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 $\mathbb{P} = \{P_1, \ldots, P_k\}$, 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 $x \equiv \tilde{x}$. We denote $\text{view}_{\text{ColMaskDecr}}$ to the transcript of Protocol 5, consisting of all the shares $(\{c_{i_0}\}, \ldots, \{c_{i_0}\})$ of $c_{i_0}$ in $R_{q}$.

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

$$S_{\text{ColMaskDecr}} \equiv \text{view}_{\text{ColMaskDecr}}$$

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

**Simulator 1** $S_{\text{ColMaskDecr}}$

**Input:** The simulator is given $(sk_1, \{c_{r_0}\}, e_t) \forall i$ and $c_r$ by the trusted setup.

**Output:** for each party $P_i$ in the pool:

$$\langle c_{i_0}\rangle_i = \begin{cases} (c_{i_0} + c_{r_1}) \{sk\}_i + e_t \{c_{r_0}\}_i & \text{if } P_i \in \mathcal{A} \\
(\mathcal{U}(R_q)) & \text{if } P_i \in \mathcal{H} \\
\left[ \sum_{P_i \in \mathcal{P} \setminus \{P_k\}} (c_{i_0})_i \right]_q & \text{if } P_i = P_k \end{cases}$$

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 $(sk_i)$ nor $e_t$, we get that:

$$(sk_1) c_{i_1} + e_t c_{i_1} \xrightarrow{c} (a \leftarrow R_q, c_{i_1})$$

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.** ($\text{ColMaskDecr}$ 1-bit leakage of output in the semi-honest pool model) For each possible set of corrupted parties $\mathcal{A} \subset \mathbb{P}$ 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 $m_{LSB} = [m + r]_j$ is the result of adding a mask $r$ to the underlying message $m$, and this mask contains uniformly random values in $\mathbb{Z}_{[0, 2]}$ for all $j \neq 0$, we get that:

$$P(m[j] = a | m_{LSB}[j] = b) = \frac{P(m[j] = a) \cdot P(m_{LSB}[j] = b|m[j] = a)}{P(r[j] = b)} = \frac{P(m[j] = a)}{P(r[j] = b)}$$

This is because $m_{LSB}[j] \xleftleftrightsquigarrow r[j] \sim U([0, 2])$, 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 $P(m[0] \mod 2 = a | m_{LSB}[0] \mod 2 = a) = 1$ 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, $P(m[0]/2 = b | m_{LSB}[0]/2 = c) = P(m[0]/2 = b)$, showing that all but the LSB of $m_{LSB}[0]$ are perfectly masked.

In the event of $m[0] \not\equiv 0, 1$, arising from a malicious Gate & BIP submitting a chosen $c_{i_1}$ (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.
The crucial difference is the mask addition. For all \( j \) (e.g., the \( p \)), save the first coefficient and in polynomial space [0,1), completely hiding that secret key coefficient. For \( j = 0 \) this leaks one bit of secret \( s[0] \sim U(1,1,0,1) \), 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 \text{RepColMaskDecr} protocol can be 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:

\[
|\mathcal{A}_P| \cdot J < K \tag{8}
\]

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

\[
J \geq |\mathcal{A}_P| + 1 \tag{9}
\]

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

\[
|\mathcal{A}_P| < \left\lfloor \sqrt{K} \right\rfloor \tag{10}
\]

### 4.5 On Auditability of Colmade

An external auditor could enroll as a user in the pool to take part in the \text{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.

\(^7\)An overview of FHE key recovery attacks can be found in \([19]\).
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 \( K_q \) 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 \( q = q_1 \cdot q_2 \cdots q_t \) 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, \( q' = q_1' \cdot q_2' \cdots q_t' \) with \( q_1' = (q_1 - q_1/\Lambda) \).

## 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 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 DBVF 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 \( \sqrt{K} \) 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


