Another Walk for MONCHI

Anonymous

Anonymous

Abstract. MONCHI is a new protocol aimed at privacy-preserving biometric identification. It begins with scores computation in the encrypted domain thanks to homomorphic encryption and ends with comparisons of these scores to a given threshold with function secret sharing. This preliminary work shows how to follow the scores computation techniques recently introduced by Bassit et al. to eliminate homomorphic multiplications. This leads to system modifications for implementation optimization and security reasons which are discussed.

Keywords: Privacy in biometric systems, Homomorhic Encryption, Function Secret Sharing, MFBR schemes

1 Introduction

Similar to [10], we investigate the use of biometric identification in an airplane boarding use case, where passengers are authorized to enter the plane only if their faces are identified with respect to an early prepared, privacy-preserving database of registered passengers' faces. Our new solution extends the previously proposed MONCHI scheme ([10], see also [11] and Appendix A) which makes use of (i) a homomorphic encryption scheme and relevant packing solutions to protect both the live templates and the reference ones stored in the database and enable the computation of scalar products between them, and, inspired by FUNSHADE [9], (ii) a Function Secret Sharing (FSS) scheme to obliviously compare the score obtained from the scalar product to a threshold. Combining these two primitives enables MONCHI to only reveal the final authorization decision without disclosing the identification scores.

We revisit that same goal, aiming to optimize even further the computation cost incurred by the scalar products over encrypted templates. In [4,5], authors propose a method to compute these identification scores over encrypted data through the use of lookup tables that entirely replace the multiplication operations. [5] reports a faster runtime by a factor of 2 to 3 orders of magnitude on facial features while keeping the biometric accuracy of the system. We integrate this multiplication-free scheme while making it compatible with the underlying BFV encryption scheme [2,7,8] and the FSS scheme [9].

In the next section 2, we detail how multiplication-free biometric schemes in the encrypted domain work. We then identify the constraints that need to be be fulfilled to let the Function Secret Sharing scheme of MONCHI unchanged and introduce our new solution in Sec. 3. Sec. 4 describes its security analysis. Finally, Sec. 5 gives conclusive remarks.

2 Multiplication-Free Biometric Recognition (MFBR) schemes

In biometric systems, discriminative features are extracted from images of captured biometric traits (e.g., faces, fingerprints...) and transformed into vectors called *templates* via the execution of a neural network model. Two templates are assumed belonging to the same person if they are close with respect to some distance. To check whether a person is present in a database, scores of his live template against existing reference templates are computed – in MONCHI, scores for faces' templates are computed by a scalar product measuring their cosine similarity – and then compared with a given threshold which depends of the underlying biometric system (see [12] for more details).

To ease their use with cryptographic techniques, components of templates are often quantized. We denote $Y = (y_1, \ldots, y_d)$ (resp. $X = (x_1, \ldots, x_d)$) to a quantized reference (resp. live) template with d components of n bits each.

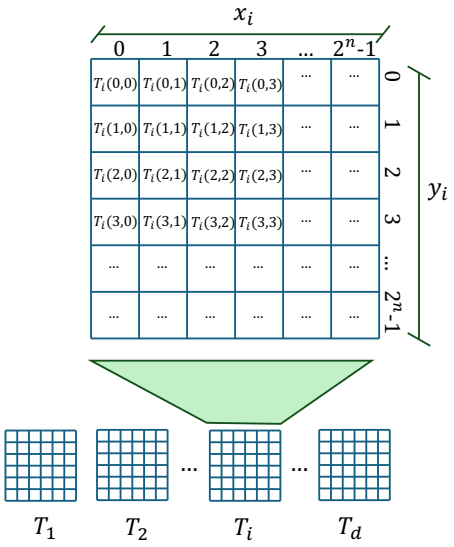

Fig. 1. Look-Up Table visualization.

The first Multiplication-Free Biometric Recognition scheme was introduced in [3] by Bassit et al. This work has then been extended in [4] and further in [5]. MFBR schemes substitute scalar products for the scores computation with well-crafted LookUp Tables (LUT) and additions. These system-dependent LUT provide, for each of the $i \in \{1, \ldots, d\}$ components, a "local" score between a live template component x_i and its corresponding reference template component y_i. The partial scores are stored in a 2D matrix T_i, accessible by indexing the desired

row with the value of y_i and the column with x_i. We compute the score s as:

$$s = T_1(y_1, x_1) + \ldots + T_d(y_d, x_d) \tag{1}$$

where $T_i(y_i, x_i)$ represents the lookup (indexing) operation.

Equation 1 can be easily adopted in the encrypted domain by virtue of homomorphic encryption (HE). Instead of encrypting the plain reference templates directly, rows $T_i(y_i, .)$ $\forall i$ are stored in database, now returning the corresponding looked-up values encrypted when queried. A non-encrypted live template $(x_1, \ldots, x_d)$ triggering a verification, leads to picking all $T_i(y_i, x_i)$ and then adding together the encrypted partial score. No multiplications are needed.

3 MONCHICHI: MONCHI with two-party look-up tables

This section describes our new proposal MONCHICHI that combines the use of look-up tables with homomorphic encryption and function secret sharing. The new solution makes use of multi-party computation (MPC) between two servers during the calculation of the score based on LUTs to reduce the size of the stored LUTs. BIP^0 and BIP^1 now replace the Biometric Identity Provider (BIP) of MONCHI.

3.1 Participants

Besides a trusted setup realized by a trusted key server, here are the main participants of our identification protocol:

- two BIP^k, holding the database containing LUTs corresponding to the reference templates and responsible for the computation of the encrypted masked scores between a freshly live template and each reference template.
- Gate, in charge of capturing the live biometric template of the users requesting access and forwarding them to the BIP^k, after protection, during the identification phase. Later on, Gate receives the final decision on this actual identification and allows or not the user to access the plane.
- P_j, (two) parties who decrypt the masked score and evaluate its comparison operation to the threshold following the FSS Funshade scheme. These two P_j were already present in MONCHI.

3.2 Notations

We use the same notations as in [10]:

- The n-bit identification score is denoted by s.
- Each evaluation of the FSS Funshade scheme requires the use of a fresh mask, which we denote by r. Let $\hat{s} = s + r$ be the masked score, i.e., the input of the FSS Interval Containment gate [6] employed for comparison.

- The encryption of the masked score is: $c_{\hat{s}} = Enc(\hat{s}, \text{pk})$ where pk stands for the public key of the system. The corresponding private key is sk. The decryption is denoted as $Dec(., \text{sk})$. In practice, as in MONCHI, the private key is shared among the $\text{P}_j, j = 0, 1$, each P_j holding the share $\langle \text{sk} \rangle_j$.
- The parameters associated with the BFV scheme are:
  - The plaintext space consists of polynomials of degree at most $N-1$ with coefficients in $\mathbb{Z}_t$.
  - The ciphertext modulus $R_q = \mathbb{Z}_q[X]/(X^N + 1)$ is set based on the security parameter q.
  - $e_i \leftarrow \chi_{R_q}$ stands for some error term (see Appendix A of [13]).

3.3 Score Evaluation by the BIP^k's

For a reference and live templates $Y = (y_1, \ldots, y_d)$ and $X = (x_1, \ldots, x_d)$, BIP^k's have to obliviously compute the encryption of a score s masked with r in $\mathbb{Z}_{2^n}$. This operation is no longer performed with multiplications as in the original MONCHI scheme but instead by using LUTs.

For each reference template, $Y = (y_1, \ldots, y_d)$, let $S_{Y,i}(.) = T_i(yi, .) + r_i \mod 2^n, i = 1, \ldots, d$ with $r_1 + \ldots + r_d = r$. The masked score between a live template $X = (x_1, \ldots, x_d)$ and Y, $\hat{s} = s + r \mod 2^n$ can thus be computed as:

$$\hat{s} = S_{Y,1}(x_1) + \ldots + S_{Y,d}(x_d) \mod 2^n$$

Because LUTs are now secret shared among the two BIP^{k}s, for $k = 0, 1$, we define $S^k_{Y,i}(.)$, s.t.

$$S^0_{Y,i}(.) + S^1_{Y,i}(.) \mod 2^n = S_{Y,i}(.) \tag{2}$$

for all $i = 1, \ldots, d$ and all reference templates Y.

We write $ES^k_{Y,i}(.), i = 1, \ldots, d, k = 0, 1$ to denote their encrypted counterparts, i.e. $ES^k_{Y,i}(x_i) = Enc(S^k_{Y,i}(x_i), \text{pk})$, for all the x_i's. Hence, given one live template X, for each of its d features, each BIP^k will look-up and retrieve the relevant cells from the corresponding LUT. Note that, since a score needs to be protected with a freshly generated mask before being sent to the P_js, there would be a need to generate and store one LUT per feature, per reference template and per live template. This would incur a non-negligible cost. In order to cope with this problem, we propose to integrate a new LUT table that is described in the next section.

3.4 LUTs Storage

For the sake of simplicity, mentions to mask r have been omitted in the previous section regarding $ES^k_{Y,i}$.

In this section, we address the renewal of r to a new mask $\overline{r}$. We have to cope with the update of the LUTs corresponding to r: ${}^{r}ES^k_{Y,i}$ to the ones ${}^{\overline{r}}ES^k_{Y,i}$ standing for $\overline{r}$. This renewal is needed for each new score computation.

FUNSHADE [9], i.e., the FSS protocol used by MONCHI to compare scores with a pre-defined threshold, requires as input the masked score which defined in $\mathbb{Z}_{2^n}$ where n is a small integer; typically, $n = 16$. Because n is small, to handle this storage issue, we introduce a vector $\mathrm{F}^k, k = 0, 1$ – which can also be considered as a dictionary of the encryption of all potential values – with the encryption of all values in $0, \ldots, 2^n - 1$. Each BIP^k holds its F^k during the boarding process. Rather than returning a value, in the encrypted domain LUTs $ES^k_{Y,i}$ will now output the actual index of this encrypted value in F^k. I.e. instead of returning an encrypted value ϵ, LUT will now store and output e s.t. $F^k(e) = \epsilon$ and BIP^k has to go through its LUTs and F^k to eventually get the required value.

That way, renewal of masks comes in handy, only keeping indexes to F^k.

3.5 Between the BIP^k's and FSS

With the introduction of the two BIPs, P_j's now receive shares of the encrypted scores that first need to be reconstructed by simple addition before its decryption: see new Algorithm 1). The decryption is distributed among the two P_j's as in MONCHI. [1,13].

In more details, each $\mathrm{BIP}^k, k = 0, 1$ first retrieves the result of each encrypted and masked product of the live template and reference template feature, computes $c^k_{\hat{s}}$ by adding the intermediate product results and sends this result to the two P_js. The two P_js will then reconstruct the masked score and decrypt the result such that:

$$Dec(c^0_{\hat{s}} + c^1_{\hat{s}}, \mathrm{sk}) \mod 2^n = \hat{s} \tag{3}$$

Algorithm 1 MONCHICHI.**BIP^{k}toFSS**$(c^k_{\hat{s}}, k = 0, 1, \langle sk \rangle_j)$

Players: P_j, $j \in \{0, 1\}$
Input:

$\langle sk \rangle_j$: Secret key share for BFV
$c^k_{\hat{s}}$ Received from the BIP^k's

Output: Decrypted masked scores $\hat{s}$.

Let $c_{\hat{s}} = c^0_{\hat{s}} + c^1_{\hat{s}} = (c_{\hat{s}_a}, c_{\hat{s}_b})$
$e_i \leftarrow \chi_{R_q}$

$$\langle c_{\hat{s}_b} \rangle_i = \langle sk \rangle_j \, c_{\hat{s}_b} + e_i$$

$$\hat{s} = \left[\left[\frac{t}{q} \left[(c_{\hat{s}_a} + \langle c_{\hat{s}_b} \rangle_0 + \langle c_{\hat{s}_b} \rangle_1) \right]_q \right]_t \right]_{2^n}$$

---

4 Security Considerations

Our MONCHICHI scheme can be described as follows:

1. The Gate gets a new live quantized template $X = (x_1, \ldots, x_d)$ and sends it to the BIP^k's.

   For each reference template:

2. Each BIP$^k, k = 0, 1$ computes a share of the encrypted and masked score $c^k_{\hat{s}}$ (see Sec. 3.3) and send it to the P$_j$'s .
3. The P$_j$'s run Algorithm 1.
4. Each P$_j$ evaluates whether the score is under a threshold or not thanks to the FSS Funshade scheme, and sends shares of this result to the Gate.

Correctness Step 4 is the same as the one in MONCHI and FUNSHADE [10]. The new Algorithm 1 distributes BFV decryption [13] of the masked score. The correctness of our scheme thus comes from the correctness of Step 2. We know that:

$$\hat{s} = Dec(\sum_{i=1}^{d} S^0_{Y,i}(x_i) + \sum_{i=1}^{d} S^1_{Y,i}(x_i), \mathrm{sk}) \mod 2^n$$

Hence MONCHICHI is correct.

Use of F^k (Sec. 3.4) We have to use a fresh mask and thus new LUTs at each execution. The splitting of BIP into two BIP^k's enables the sharing of values stored by LUTs as by (2). Hence, an Honest-but-Curious adversary that corrupts up to one of the participants as in [10] can only get one of these shares which looks random to him.

Confidentiality of live templates Note that [4,5] make use of client secret permutations to hide live template X as they are otherwise leaked by which indexes in the rows $T_i(y_i, .)$ are used. The boarding scenario of [10] cannot accommodate these permutations kept by passengers as we want them to come to cross the gates hand-free. In our scheme, LUTs are renewed at each score computation and we can implement secret permutations both at the Gate and in LUTs stored by BIP^k's. Permutations are not picked by clients/passengers anymore but are rather generated during the trusted setup by the system. At Step 1, Gate takes a new secret permutation for each BIP^k and applies it to the live template before sending the permuted live template to BIP^k's. The Gate and BIP^k's have to be synchronized as the LUTs held by BIP^k's have to take into account these permutations.

5 Conclusion

We explain how to integrate MFBR techniques into the MONCHI protocol [10].

We have to cope with the need to only consume single-use pre-processed data for FSS. Taking this constraint as an advantage, we explain how secret permutations used to protect live templates – handled by clients in [4,5] – can

be implemented rather by the Gate and the LUTs of BIP^k's in our proposal. We also modify the underlying system by distributing the Biometric Identity Provider into several entities BIP^k's to reduce LUTs size storage.

The next step following this preliminary work is to implement these idea to gauge performances benefits. One can also envisage to move from BFV encryption to an additive-only homomorphic scheme (e.g. [14]) as no multiplications in the encrypted domain are needed anymore. Finally, the fusion of BIP^0 with P_0 and BIP^1 with P_1 has to be considered.

References

1. Lattigo v5. Online: https://github.com/tuneinsight/lattigo, Nov. 2023. EPFL-LDS, Tune Insight SA.
2. Bajard, J.-C., Eynard, J., Hasan, M. A., and Zucca, V. A full RNS variant of FV like somewhat homomorphic encryption schemes, 2016.
3. Bassit, A., Hahn, F., Peeters, J., Kevenaar, T., Veldhuis, R. N. J., and Peter, A. Fast and accurate likelihood ratio-based biometric verification secure against malicious adversaries. *IEEE Trans. Inf. Forensics Secur. 16* (2021), 5045–5060.
4. Bassit, A., Hahn, F., Veldhuis, R. N. J., and Peter, A. Multiplication-free biometric recognition for faster processing under encryption. In *IJCB* (2022), IEEE, pp. 1–9.
5. Bassit, A., Hahn, F., Veldhuis, R. N. J., and Peter, A. Improved multiplication-free biometric recognition under encryption. In *IEEE Transactions on Biometrics, Behavior, and Identity Science* (2023), IEEE, p. Advance online publication. https://doi.org/10.1109/TBIOM.2023.3340306.
6. Boyle, E., Chandran, N., Gilboa, N., Gupta, D., Ishai, Y., Kumar, N., and Rathee, M. Function secret sharing for mixed-mode and fixed-point secure computation, 2021.
7. Fan, J., and Vercauteren, F. Somewhat practical fully homomorphic encryption, 2012.
8. Halevi, S., Polyakov, Y., and Shoup, V. An improved RNS variant of the BFV homomorphic encryption scheme, 2019.
9. Ibarrondo, A., Chabanne, H., and Önen, M. Funshade: Functional secret sharing for two-party secure thresholded distance evaluation, 2022.
10. Ibarrondo, A., Kerenciler, I., Chabanne, H., Despiegel, V., and Önen, M. Monchi: Multi-scheme optimization for collaborative homomorphic identification. In *IH&MMSec* (2024), ACM, p. To Appear.
11. Ibarrondo, A., Kerenciler, I., Chabanne, H., Despiegel, V., and Önen, M. Monchi: Multi-scheme optimization for collaborative homomorphic identification. In *IACR Cryptol. ePrint Arch.* (2024), p. 654.
12. Jain, A. K., Flynn, P., and Ross, A. A. *Handbook of biometrics.* Springer Science & Business Media, USA, 2007.
13. Mouchet, C., Troncoso-Pastoriza, J. R., Bossuat, J., and Hubaux, J. Multiparty homomorphic encryption from ring-learning-with-errors. *Proc. Priv. Enhancing Technol. 2021*, 4 (2021), 291–311.
14. Paillier, P. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT* (1999), vol. 1592 of *Lecture Notes in Computer Science*, Springer, pp. 223–238.

A Monchi VS Monchichi

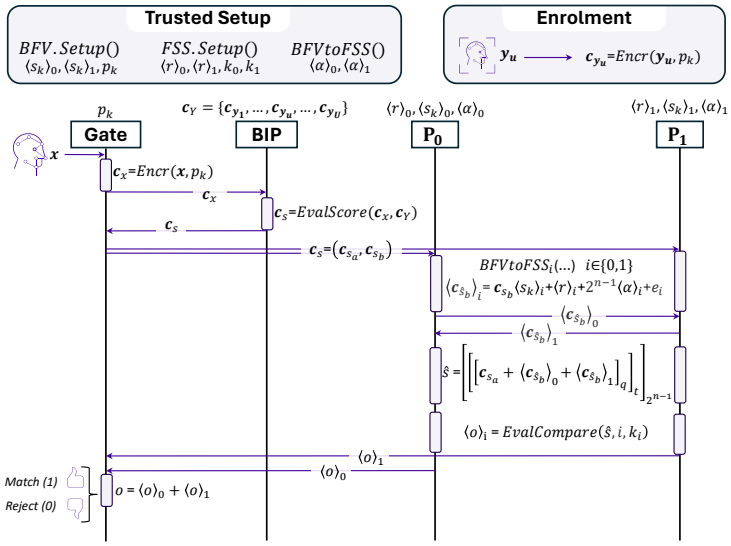

Fig. 2. System Diagram of a biometric access control using MONCHI's protocol.

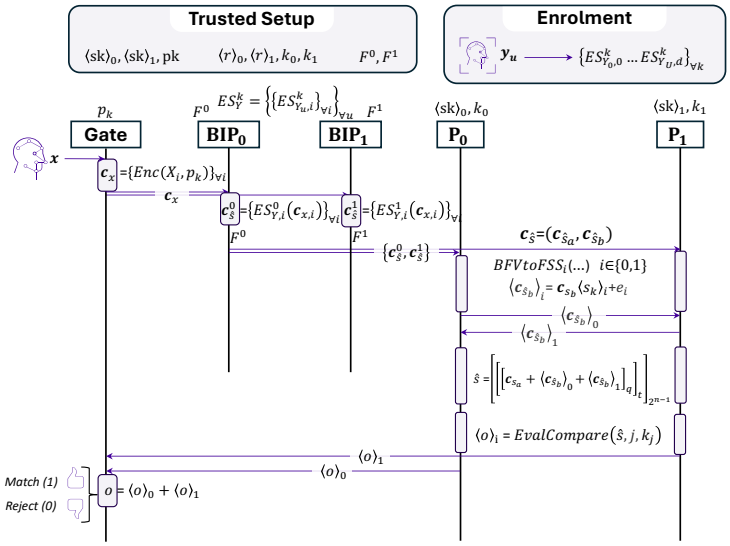

Fig. 3. System Diagram of a biometric access control using MONCHICHI's protocol.