Another Walk for Monchi

Riccardo Taiello[1], Emre Tosun[2], Alberto Ibarrondo[3], Hervé Chabanne[4], and Melek Önen[2]

[1] Inria, Sophia Antipolis
[2] Eurecom, Sophia Antipolis
[3] Arcium, Zurich
[4] Idemia, Paris

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. We here study the integration in that context of scores computation techniques recently introduced by Bassit et al. that eliminate homomorphic multiplications by replacing them by lookup tables. First, we extend this lookup tables biometric recognition solution by adding the use of function secret sharing for the final comparison of scores. Then, we introduce a two-party computation of the scores with lookup tables which fits nicely together with the function secret sharing scores comparison. Our solutions accommodate well with the flight boarding use case introduced by Monchi.

Keywords: Privacy-Preserving biometric systems, Homomorphic Encryption, Two-Party Computation, Function Secret Sharing, MFBR schemes

1 Introduction

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. Set on this scenario, Monchi [10,11] makes use of (i) a homomorphic encryption (HE) 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 [3,4], authors propose a method to compute these identification scores over encrypted data

through the use of lookup tables that entirely replace the multiplication operations. [4] 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 [1,6,7] and the FSS scheme [9].

To ensure the confidentiality of the values in the table and the computations of the scores, we also propose – in addition to HE – to study the use of two-party computation (2PC), and more specifically, additive secret sharing [15]. As no multiplications are needed, no interactions between the two parties are neither required. In the next section 2, we detail how multiplication-free recognition (MFBR) biometric schemes in the encrypted domain work. Sec. 3 describes our first proposal, named MonchiLuts, where MFBR LUTs replace multiplications with HE. Sec. 4 gives our second proposal, called Monchichi, where we implement scores computation in 2PC. Sec. 5 is devoted to the confidentiality of live templates. Finally, Sec. 7 gives conclusive remarks.

2 Multiplication-Free Biometric Recognition (MFBR)

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 [13] 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.

The first Multiplication-Free Biometric Recognition scheme was introduced in [2] by Bassit et al. This work has then been extended in [3] and further in [4]. 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) or arithmetic secret sharing [15]. 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 or secret-shared when

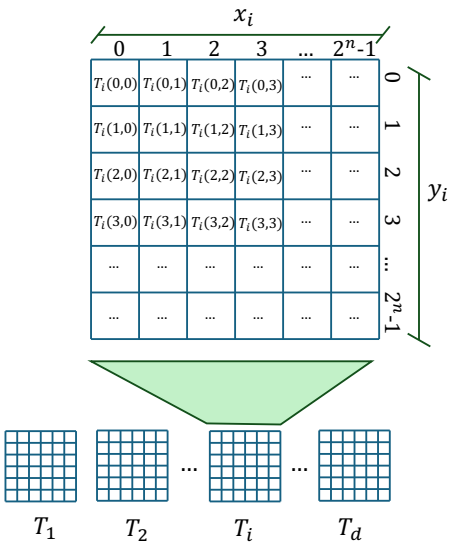

Fig. 1: Look-Up Table visualization.

queried. A cleartext 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 or secret-shared partial score. No multiplications are needed.

3 MONCHILUTS: MONCHI with MFBR LUTs

This section describes our first proposal – called MONCHILUTS – that combines the use of MFBR LUTs and function secret sharing.

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

- BIP (Biometric Identity Provider) 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, 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.

Notations We use the same notations as in [10]:

- The $n\log(d)$-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 [5] employed for comparison.
- The encryption of the masked score is: $c_{\hat{s}} = Enc(\hat{s}, \mathrm{pk})$ where pk stands for the public key of the system. The corresponding private key is sk. The decryption is denoted as $Dec(., \mathrm{sk})$. In practice, as in MONCHI, the private key is shared among the $\mathrm{P}_j, j = 0, 1$, each P_j holding the share $\langle \mathrm{sk} \rangle_j$.
- 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 [14]).

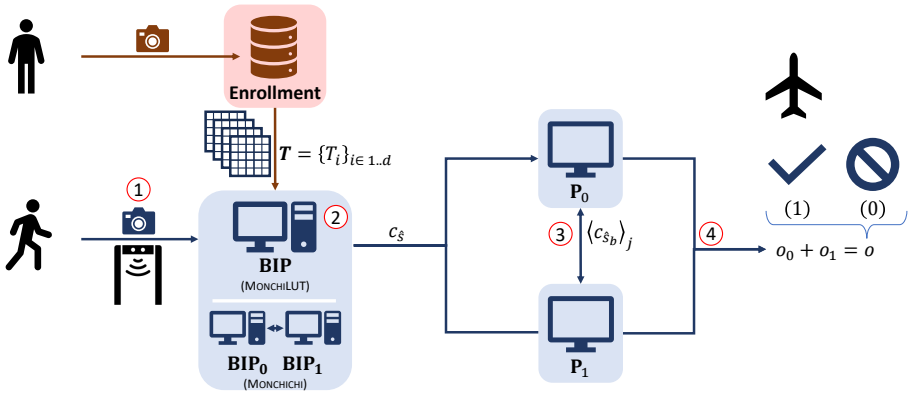

Fig. 2: Biometric access control system using MONCHICHI's protocol.

Score Evaluation by the BIP For one reference and one live template $Y = (y_1, \ldots, y_d)$ and $X = (x_1, \ldots, x_d)$, BIP has 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 $S_{Y,i}(.) = T_i(y_i, .) + r_i \mod 2^n, i = 1, \ldots, d$ 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:

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

We write $ES_{Y,i}, i = 1, \ldots, d$ to denote their encrypted counterparts, i.e. $ES_{Y,i}(x_i) = Enc(S_{Y,i}(x_i), \mathrm{pk})$, for all the x_i's (for the sake of simplicity, mentions to mask r have been omitted).

Hence, given one live template X, for each of its d features, the BIP will look-up and retrieve the relevant cells from the corresponding LUT.

FUNSHADE [9], i.e. the FSS protocol also used in [10] to compare scores with a pre-defined threshold, requires as input the masked score which is defined in $\mathbb{Z}_{2^n}$ where n is a small integer; typically, $n = 16$.

Correctness Our first proposal 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.

   For each reference template:

2. The BIP computes the encrypted masked score and sends this $c_{\hat{s}}$ to the P_j's.
3. The two P_j's decrypt the result. Let $c_{\hat{s}} = (c_{\hat{s}_a}, c_{\hat{s}_b})$. Each P_j computes: $\langle c_{\hat{s}_b} \rangle_j = \langle sk \rangle_j c_{\hat{s}_b} + e_i$ with $e_i \leftarrow \chi_{R_q}$ and then sends it to the other. Finally, each P_j gets the masked score as: $\hat{s} = \left[\left[\frac{t}{q} \left[(c_{\hat{s}_a} + \langle c_{\hat{s}_b} \rangle_0 + \langle c_{\hat{s}_b} \rangle_1) \right]_q \right]_t \right]_{2^n}$
4. 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.

Step 4 is the same as the one in MONCHI and FUNSHADE [10]. Step 3 distributes BFV decryption [12,14] of the masked score. The correctness of our scheme thus comes from the correctness of Step 2. We know that:

$$\hat{s} = Dec(\sum_{i=1}^{d} S_{Y,i}(x_i)) \mod 2^n$$

Hence, our first proposal is correct.

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

We now describe MONCHICHI, our second proposal where the Biometric Identity Provider is implemented by 2 entities, denoted as BIP^k, $k = 0, 1$. LUTs are now secret shared among the two BIP^ks, for $k = 0, 1$, we define $S^k_{Y,i}(.)$, s.t.

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

for all $i = 1, \ldots, d$ and all reference templates Y. Score computation (2) is implemented by each BIP^k which locally computes:

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

With the introduction of the two BIPs, P_j's now receive shares of the masked scores that first need to be reconstructed by simple addition: $\hat{s} = \hat{s}^0 + \hat{s}^1 \mod 2^n$.

P_j's can then launch the FSS operation and output the final result of the comparison.

5 Confidentiality of live templates

[3,4] 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 Biometric Identity Provider(s). We thus turn the constraint to have to cope with the need to only consume single-use pre-processed data for FSS to our advantage

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 the BIP or BIP^k and applies it to the live template before sending the permuted live template to the BIP or BIP^k's. The Gate and the BIP or BIP^k's have to be synchronized as the LUTs held by BIP^k's have to take into account these permutations.

6 Performance Evaluation

We implement MonchiLuts and Monchichi in Golang like the Monchi solution [10]. Similar to Monchi, these implementations use: (i) the Lattigo implementation [12] of the $(2, 2)$ threshold-variant of BFV, the LUT implementation from [4] the Funshade library [9] and a native Golang implementation for additive secret sharing. The code can be found in `https://anonymous.4open.science/r/another-walk-for-monchi/`. The performance of both solutions is evaluated through experiments executed on a single core machine with an Intel(R) Core(TM) i7-7800X CPU @ 3.50GHz and 126 GB of RAM.

For MonchiLuts, BFV parameters are set to: polynomial degree $N = 2^{11}$, ciphertext modulus size $q = 183$ bits, and plaintext modulus t is set to 32 bits.

For Monchichi, the two-party secret-sharing is executed in $Z_{2^{16}}$.

Regarding the implementation of LUTs, we use the same parameters as in [4], specifically the quantization factor is set to 3 bits for LUT creation, reference, and live template. Finally, for the FSS instantiation, we also use 16-bit modular arithmetic.

To study and evaluate the performance of our protocols in the context of the airport use case, we use the publicly available Labeled Faces in the Wild (LFW) dataset [8], which includes 13,233 facial images of 5,749 individuals. The same pre-processing steps proposed by [10] are applied, with template feature sizes set to $d = 128$.

Table 1 presents the execution time (in ms) and the communication overhead (in bytes) of the BIP for the different steps of an identification operation of one fresh template in MonchiLuts and Monchichi. The measurements are the result of an average across 20 executions.

Regarding the bandwidth cost, Monchichi only requires the transmission of a secret-shared masked score (a single value in $\mathbb{Z}_{2^{16}}$) and the final masked score,

	MonchiLuts				Monchichi			
	Time (ms)			Comm.	Time (ms)			Comm.
DB Size (T)	LUT	FSS	Total	(Bytes)	LUT	FSS	Total	(Bytes)
1	8.86	0.12	8.98	196730	< 0.00	0.12	0.12	< 10
200	1772	24	1796	196730	< 0.01	24	24.01	< 10
1024	9072.64	122.88	9195.52	196730	0.5	122.93	122.88	< 10

Table 1: Performance comparison between MonchiLuts and Monchichi protocols across different metrics: LUTs, FSS, Total Time (in milliseconds), and communication overhead (in bytes). The final three columns represent execution times (in seconds) for varying dataset sizes T.

Multiplication Time (μs)	Rotation Time (ms)	Addition Time (μs)
415.81 ± 15.38	6.89 ± 0.40	98.96 ± 5.89

Table 2: Performance metrics for multiplication, rotation, and addition.

which is again a scalar value. Although as opposed to Monchichi, MonchiLuts involves a single BIP, the latter sends two encrypted values to the P_i's which results in a non-negligible overhead compared to Monchichi.

In the scenario with a database with 1,000 faces, Monchichi (with 1-core machine) outperforms the original Monchi scheme (with 4-core machine) by 60%. On the other hand, MonchiLuts exhibits significant overhead compared to Monchi.

7 Conclusion

This work introduces two new protocols:

- MonchiLuts which is an extension of the work by Bassit et al. [3,4] to handle secure scores comparison to a threshold;
- Monchichi a full 2PC biometric identification solution combining MFBR LUTs and FSS.

These two schemes enable us to integrate at a system level the permutations needed to protect the confidentiality of live templates. These permutations were originally given to the users which forbids the hand-free boarding plane scenario introduced by Monchi. Our experiments confirm the practicability of both MonchiLuts and Monchichi for this use case.

References

1. Bajard, J., Eynard, J., Hasan, M. A., and Zucca, V. A full RNS variant of FV like somewhat homomorphic encryption schemes. In *SAC* (2016), vol. 10532 of *LNCS*.
2. 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).
3. Bassit, A., Hahn, F., Veldhuis, R. N. J., and Peter, A. Multiplication-free biometric recognition for faster processing under encryption. In *IJCB* (2022), IEEE.
4. 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.
5. 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. In *EUROCRYPT (2)* (2021), vol. 12697 of *LNCS*.
6. Fan, J., and Vercauteren, F. Somewhat practical fully homomorphic encryption. *IACR Cryptol. ePrint Arch.* (2012).
7. Halevi, S., Polyakov, Y., and Shoup, V. An improved RNS variant of the BFV homomorphic encryption scheme. In *CT-RSA* (2019), vol. 11405 of *LNCS*.
8. Huang, G. B., Ramesh, M., Berg, T., and Learned-Miller, E. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Tech. Rep. 07-49, University of Massachusetts, Amherst, October 2007.
9. Ibarrondo, A., Chabanne, H., and Önen, M. Funshade: Function secret sharing for two-party secure thresholded distance evaluation. *Proc. Priv. Enhancing Technol. 2023*, 4 (2023).
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.
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).
12. Insight, T. Lattigo v5. Online: https://github.com/tuneinsight/lattigo, Nov. 2023. EPFL-LDS, Tune Insight SA.
13. Jain, A. K., Flynn, P., and Ross, A. A. *Handbook of biometrics*. Springer Science & Business Media, USA, 2007.
14. 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).
15. Shamir, A. How to share a secret. *Commun. ACM 22*, 11 (1979).