Delegating Biometric Authentication with the Sumcheck Protocol

Hervé Chabanne, Julien Keuffer, Roch Lescuyer

Delegating Biometric Authentication with the Sumcheck Protocol

Hervé Chabanne[1], Julien Keuffer[2], and Roch Lescuyer[3]

[1] Safran Identity & Security, Télécom ParisTech
[2] Safran Identity & Security, Eurecom
[3] Safran Identity & Security
{herve.chabanne,julien.keuffer,roch.lescuyer}@morpho.com

Abstract. In this paper, we apply the Sumcheck protocol to verify the Euclidean (resp. Hamming) distance computation in the case of facial (resp. iris) recognition. In particular, we consider a border crossing use case where, thanks to an interactive protocol, we delegate the authentication to the traveller. Verifiable computation aims to give the result of a computation and a proof of its correctness. In our case, the traveller takes over the authentication process and makes a proof that he did it correctly leaving to the authorities to check its validity. We integrate privacy preserving techniques to avoid that an eavesdropper gets information about the biometric data of the traveller during his interactions with the authorities. We provide implementation figures for our proposal showing that it is practical.

Keywords: biometrics, verifiable computing, authentication

1 Introduction

1.1 Motivation

In order to increase the throughput in border crossing, controls operated by officers could be replaced with automated systems. Such systems often use biometrics to authenticate the travellers: a comparison is made between an official document such as a biometric passport and the traveller who needs to prove his identity. However biometric data need to be collected from the traveller to be compared with data stored on the official document and this step of the process can be time consuming. Delegating a part of the process to the traveller can save time but raises a confidence problem: how can the authority be sure that the traveller really ran the computation?

We use verifiable computing as a tool to address this problem. A verifiable computation system allows a *verifier* to delegate the computation of a function to a *prover*. Upon completion of the computation, the prover returns the result and a proof of the computation. In our use case, the traveller's smart device has the role of the prover and has thus restricted computational power and storage capacity. We stress that this reverses the classical roles played by the verifier and

the prover in most of verifiable computing scenarios, where a weak verifier usually delegates computations to a powerful but untrusted prover. The choice of the underlying verifying system has thus been driven according to this configuration. In particular, the requirements for the prover and the targeted computation led us to choose an interactive proof protocol, namely the SUMCHECK protocol [15].

1.2 Background on Biometrics

A biometric system is a pattern recognition system, which makes biometric data acquisition from an individual, then extracts a feature set from the acquired data which gives a *biometric template*. In an *authentication* scheme,the template is then compared against a referenced template and in an *identification* scheme it is compared against a database of templates. Due to external conditions such as light, moisture or the sensor used for the capture, two templates computed from the same individual can vary. However, the variation is expected to be small enough to be able to discriminate two templates coming from the same person from two templates coming from different individuals. This is why the comparison of two templates is usually a matching score, reflecting a similarity rate between the two data. A matching threshold has to be defined to discriminate the templates belonging to the same individual or not. Ideally, if the score of two templates is lower than the threshold, they belong to the same individual. However, in biometric systems, two different individuals can have a matching score lower than the threshold, which leads to the definition of the *false acceptance rate* (FAR) and the *false rejection rate* (FRR), see [13] for details.

In our scenario, we need an automated face recognition system. Today, many systems performing face recognition use machine learning techniques to transform a face picture into a biometric template. The model called convolution neural network (CNN)[14] has shown excellent results [18, 22]. CNNs have millions of parameters that are tuned in a learning phase, using a face database for the training. Once the training phase is over, the CNN can embed a picture in a Euclidean space where two vectors representing the same face are closer than two vectors that come from different faces, enabling face recognition.

1.3 Background on Verifiable Computation

Although the problem of verifying computations has been theoretically solved with tools from complexity theory and cryptography [1, 16], new challenges raised by verifiability in the setting of cloud computing recently attracted the interest of researchers. Huge progresses have been made and several research teams succeeded in implementing verifiable computing systems. All these systems start by turning the function to verify into a circuit composed of multiplication and addition gates and then perform verification on the circuit.

A first line of work has built on a refined version of probabilistically checkable proofs (PCP) [12] and resulted in a verifiable system called Pepper [20], which has been refined since [19, 24]. The second line was opened by Gennaro *et al.* [9], who achieved a breakthrough by building efficient objects to verify

computations called quadratic arithmetic programs (QAPs), resulting in an efficient system called Pinocchio [17]. Pinocchio and its refined version [7] allow public verifiability: anyone who has access to the public verification key can verify proofs. Moreover, the prover can make his proof zero-knowledge: he supplies a private input to the computation and builds a proof of the correctness of the result without revealing his input to the verifier. Finally, a system called TinyRAM and designed by Ben-Sasson *et al.* [3] uses QAPs and has the ability to verify a larger class of computations by modelling programs using RAM. The third line of work relied on the notion of interactive proofs, which was introduced by Goldwasser *et al.* [11]. In the verifiable computing setting, the verifier checks that the result of the computation is correct during a sequence of interactions with the prover. The more the verifier asks queries, the less the prover has chance to cheat. Goldwasser *et al.* [10] introduced an efficient protocol, later optimized and implemented by Cormode *et al.* [5]. The last version of this protocol, due to Thaler [23], is currently one of the fastest scheme for verifiable computing. Furthermore, Thaler proposed an implementation of matrix multiplication and also showed that the the main tool of interactive proofs protocols, namely the SUMCHECK protocol [15], can be used to design an efficient protocol for matrix multiplication verification.

However all the systems described above are only *nearly practical* for generic computations. The different systems all have advantages and drawbacks, depending on the type of computations to be verified. One important thing is that all systems building upon PCPs and QAPs need precomputations and amortize their costs by using the verified function several times. The fastest system needs no precomputation and uses the CMT protocol but it cannot handle general computations. Systems based on QAPs and on PCPs have better expressiveness and allow fast verification but the prover's overhead costs compared to native execution of the same computation is consequent. See [25] for comparisons between the different existing systems.

Cormode *et al.* [6] suggested that the SUMCHECK protocol could be used to verify an inner product in the setting of data streaming, where the verifier cannot store the inputs and has to update his computations while he is parsing the data. The recent work of [4] studies the use of verifiable computing for biometric verification in a non-interactive setting *i.e.* where the prover computes a proof without interacting with the verifier. In contrast, we focus on interactive proofs to design and implement a protocol which aims at verifying several distances used in biometric matchings and adapt the SUMCHECK protocol [15].

2 Use-Case : Fast Border Control

In many places, people living next to another country frequently cross the border with their cars to go to work. We want here to design an automated system to reduce the waiting time, taking profit of the waiting time in the cars queuing line. Our idea is to let the driver perform himself his face verification against his passport photo while crossing the border. Such operations could be performed

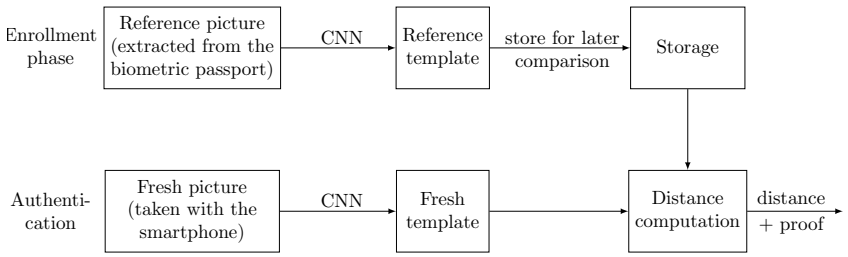

Fig. 1. The biometric matching process

by a dedicated application installed on his smartphone. At the end, the customs authority will get from this application: a fresh photograph, the official one and a proof that both belong to the same person (Figure 2). A high-level description of our solution (see also Figure 1):

- The traveller (who plays the role of the prover) uses a wireless communication device of the mobile to get the picture stored in his biometric passport.
- The picture is turned into a reference template using a CNN.
- The traveller takes a fresh picture of his face and uses the same CNN to create a biometric template.
- A biometric matching is performed on the traveller's mobile and interactions between the traveller and the automated border control device lead to a proof that the matching was correctly computed. The interaction begins with the prover sending two templates and the result of the distance computation to the verifier. The proof is stored on the mobile for a later examination.

We emphasize that our contribution is limited to the application of verifiable computation on distance computations involved in biometric matchings. This is only a part of what is needed to address the whole problem. For instance, liveness detection or verifying the CNN output seem necessary but those topics are outside the scope of this paper. Our purpose here is to deal with a realistic use case for a delegation of a verifiable face matching algorithm. Since a CNN embeds a pictures in a Euclidean space, the verifiable biometric matching involves a distance computation which is compared to a threshold. We first show how to verify an inner product and then extend the verification to euclidean distance computing.

3 The Sumcheck Protocol and Verifiable Distances

In the SUMCHECK protocol [15], a prover $\mathcal{P}$ wants to convince a verifier $\mathcal{V}$ that he knows the value of the expression: $H = \sum_{x_1 \in \{0,1\}} \sum_{x_2 \in \{0,1\}} \dots \sum_{x_n \in \{0,1\}} g(x_1, \dots, x_n)$, where g is a multivariate polynomial defined over a finite field. If g has n variables, the protocol has n rounds. In each round, the verifier picks a random value

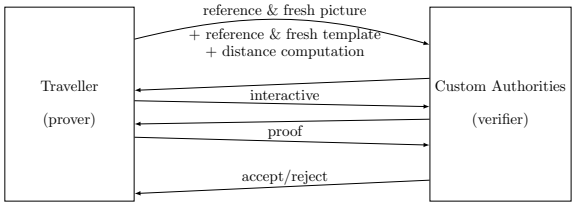

Fig. 2. Authentication process between the prover and the verifier

from a finite field and challenges the prover to compute a polynomial derived from the initial polynomial to verify. This polynomial is sent by the prover to the verifier and few computations are needed to check that it is linked to the previous one. If the check succeeds, the verifier can accept the new polynomial as a claim for the initial value. At the end, the verifier has to evaluate the original polynomial in a single value to decide if he accepts the initial claim with high probability. This is the only step where the verifier has a computing challenge.

Verifiable inner product We now give details on how we adapt the SUMCHECK protocol to obtain a verifiable inner product and verifiable distance computations. Consider two vectors $a, b \in \mathbb{Z}^n$. The inner product of a and b is defined by: $H = \sum_{i=0}^{n-1} a_i \cdot b_i$. Denoting $d = \log n$, the expression giving H can be rewritten, considering a and b as functions defined over $\{1, \ldots, n\}$ such that: $a : i \mapsto a_i$ and $b : i \mapsto b_i$. If the index i is written as a binary vector, $i = \sum_{k=1}^{d} i_k 2^{k-1} = (i_1, \ldots, i_d)$ then a and b define multivariate polynomials:
$a : (i_1, \ldots, i_d) \mapsto a_i$ and $H = \sum_{i_1 \in \{0,1\}} \ldots \sum_{i_d \in \{0,1\}} a(i_1, \ldots, i_d) \cdot b(i_1, \ldots, i_d)$
To increase the possibility of catching a lie from a cheating prover, the SUMCHECK protocol uses polynomials defined over a large finite field, which agree with a and b over $\{0,1\}^d$ and called low-degree extensions and denoted $\tilde{a}$ and $\tilde{b}$. The above relation still holds with low-degree extensions of a and b.

Squared Euclidean distance The protocol described in Section 3 can be adapted straightforwardly to verify Euclidean distance. Indeed, given two n-components biometric templates a and b, their squared Euclidean distance is:

$$d_E(a, b) = \sum_{i=1}^{n} (a_i - b_i)^2 = \sum_{i=1}^{n} a_i^2 + b_i^2 - 2a_i \cdot b_i \tag{1}$$

Denoting $d = \log_2 n$, we have to verify with the SUMCHECK protocol the evaluation of the polynomial g over $\{0,1\}^d$:

$$g(x_1, \ldots, x_d) = \tilde{a}(x_1, \ldots, x_d)^2 + \tilde{b}(x_1, \ldots, x_d)^2 - 2\tilde{a}(x_1, \ldots, x_d) \cdot \tilde{b}(x_1, \ldots, x_d) \tag{2}$$

The same ideas can be adapted to verify the distance involved in iris recognition, which is a weighted Hamming distance [8].

4 Adding Data Privacy to the Protocol

At the beginning of the protocol described in Section 2, the driver has to send his reference and his fresh templates to the authorities for the verification process. Since biometric template cannot be revoked, we propose to add masking techniques for the templates [2]. In our context, this means that the driver has to pick a random permutation of the template coordinates and a random vector of the same size than the template. More precisely, a template $t = (t_1, \ldots, t_n)$ masked becomes $t_{masked} = \pi(t) + (r_1, \ldots, r_n)$ where π is a random permutation of the n coordinates and $(r_1, \ldots, r_n)$ is a n components vector of $\mathbb{F}_p^n$.

So if t_{ref} and t are masked with the *same* permutation and random vector, computing their distance involves computing their difference:

$$\pi(t_{ref}) + (r_1, \ldots, r_n) - (\pi(t) + (r_1, \ldots, r_n)) = \pi(t_{ref}) - \pi(t) = \pi(t_{ref} - t)$$

And the scalar product of this difference has the same value than the scalar product computed on the vectors without masks: since π permutes the same coordinates on t and t_{ref}, the difference vector masked is the permutation of the original difference vector and computing the scalar product on this masked vector will give the same result.

The distance computation with masked templates gives information about the distance between the templates and the differences between the coordinates of the templates. But linking these differences coordinates to the unmasked template coordinates is hard because of the number of possible permutations and vectors.

We also stress that the driver has to store the permutation and the random vector. Therefore if the authorities have a doubt about the identity of the driver, the driver has everything on his phone to unmask the templates and compute the distance between them. Similar techniques can be used for iris recognition.

5 Experimental Results

We implement a verified inner product using the SUMCHECK protocol, the computations being over the prime finite field $\mathbb{F}_p$ where $p = 2^{61} - 1$. The size of a field element is thus inferior to the machine word size and the probability of being fooled by a dishonest prover is small, see Table 1. Note that optimizations are possible for the verifier but since in our use case the verifier has computational power, we did not implement them.

We run our benchmarks on random vectors of different sizes composed of natural numbers. Dealing with negative numbers or with floating-point rationals is possible with an additional step, *e.g.* the computations over negative numbers are mapped to computations over a finite field large enough so that the mapping

is a one-to-one function [21]. This step is done before the prove and verify steps. The protocol has therefore to be implemented in a larger field at the cost of a decrease of performances.

Communication costs and security For input vectors of size n, the SUMCHECK protocol has $\log_2 n$ rounds, the verifier sends one field element per round (the random challenge, see Section 3) and the prover three (the three values needed to interpolate round k polynomial). Not taking into account the sending of the input values, we obtain that the total communication during the protocol is $4\log_2(n) + 1$ field elements.

The security of the SUMCHECK protocol is the probability that a cheating prover builds a proof of a false result that will be accepted by the verifier, this value is given in Table 1 for different input sizes.

Benchmarks We run experiments on a laptop with a 2 GHz Intel Core i5 processor with 8 GB of RAM. The implementation is written in C++. Table 1 gives the average times of 1000 computations for each vector size. We note that this technique does not need the notion of arithmetic circuits. Using the optimized version of the CMT protocol (see Section 1) would lead to a slower protocol with two times more communication costs.

n	Inner prod. (ms)	Prover time (ms)	Verifier time (ms)	Security	Communication
128	$< 10^{-4}$	0.01	0.032	2^{-54}	232 B
256	0.0031	0.017	0.053	2^{-54}	264 B
512	0.0032	0.042	0.098	2^{-53}	296 B
1024	0.0032	0.065	0.25	2^{-53}	328 B
4096	0.0077	0.31	1.13	2^{-52}	392 B
2^{20}	3	122	600	2^{-51}	648 B

Table 1. Benchmark of the verified inner product of two n-components vectors

Acknowledgements. This work has been partially funded by the European H2020 TREDISEC (ICT-2014-644412) and the French ANR BIOPRIV (ANR-12-INSE-0013) projects.

References

1. Arora, S., Safra, S.: Probabilistic checking of proofs: A new characterization of NP. J. ACM pp. 70–122 (1998)
2. Atallah, M.J., Frikken, K.B., Goodrich, M.T., Tamassia, R.: Secure biometric authentication for weak computational devices. In: FC'05. pp. 357–371 (2005)
3. Ben-Sasson, E., Chiesa, A., Genkin, D., Tromer, E., Virza, M.: Snarks for C: verifying program executions succinctly and in zero knowledge. In: Advances in Cryptology - CRYPTO 2013. Proceedings, Part II. pp. 90–108 (2013)
4. Bringer, J., Chabanne, H., Kraiem, F., Lescuyer, R., Soria-Vazquez, E.: Some applications of verifiable computation to biometric verification. In: 2015 IEEE International Workshop on Information Forensics and Security, WIFS (2015)

5. Cormode, G., Mitzenmacher, M., Thaler, J.: Practical verified computation with streaming interactive proofs. In: ITCS '12. pp. 90–112 (2012)
6. Cormode, G., Thaler, J., Yi, K.: Verifying computations with streaming interactive proofs. In: Conference on Very Large Data Bases – VLDB'12 (2012)
7. Costello, C., Fournet, C., Howell, J., Kohlweiss, M., Kreuter, B., Naehrig, M., Parno, B., Zahur, S.: Geppetto: Versatile verifiable computation. In: Proceedings of the IEEE Symposium on Security and Privacy (2015)
8. Daugman, J.: How iris recognition works. IEEE Trans. Circuits Syst. Video Techn. 14(1), 21–30 (2004)
9. Gennaro, R., Gentry, C., Parno, B., Raykova, M.: Quadratic span programs and succinct NIZKs without PCPs. In: EUROCRYPT 2013. Springer (2013)
10. Goldwasser, S., Kalai, Y.T., Rothblum, G.N.: Delegating computation: Interactive proofs for muggles. In: STOC '08. pp. 113–122 (2008)
11. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof-systems. In: Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing. STOC '85 (1985)
12. Ishai, Y., Kushilevitz, E., Ostrovsky, R.: Efficient arguments without short pcps. In: IEEE Conference on Computational Complexity (CCC 2007) (2007)
13. Jain, A.K., Ross, A., Prabhakar, S.: An introduction to biometric recognition. IEEE Trans. Circuits Syst. Video Techn. 14(1), 4–20 (2004)
14. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE 86(11), 2278–2324 (1998)
15. Lund, C., Fortnow, L., Karloff, H., Nisan, N.: Algebraic methods for interactive proof systems. J. ACM pp. 859–868 (1992)
16. Micali, S.: Computationally sound proofs. SIAM J. Comput. pp. 1253–1298 (2000)
17. Parno, B., Howell, J., Gentry, C., Raykova, M.: Pinocchio: Nearly practical verifiable computation. In: IEEE Symposium on Security and Privacy, SP'13 (2013)
18. Schroff, F., Kalenichenko, D., Philbin, J.: Facenet: A unified embedding for face recognition and clustering. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2015)
19. Setty, S., Braun, B., Vu, V., Blumberg, A., Parno, B., Walfish, M.: Resolving the conflict between generality and plausibility in verified computation. In: EuroSys (2013)
20. Setty, S., McPherson, R., Blumberg, A.J., Walfish, M.: Making argument systems for outsourced computation practical (sometimes). In: NDSS (2012)
21. Setty, S.T.V., Vu, V., Panpalia, N., Braun, B., Blumberg, A.J., Walfish, M.: Taking proof-based verified computation a few steps closer to practicality. In: USENIX Security Symposium. pp. 253–268. USENIX Association (2012)
22. Taigman, Y., Yang, M., Ranzato, M.A., Wolf, L.: Deepface: Closing the gap to human-level performance in face verification. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2014)
23. Thaler, J.: Time-optimal interactive proofs for circuit evaluation. In: Advances in Cryptology – CRYPTO 2013, pp. 71–89. Springer (2013)
24. Wahby, R.S., Setty, S., Ren, Z., Blumberg, A.J., Walfish, M.: Efficient RAM and control flow in verifiable outsourced computation. In: NDSS (2015)
25. Walfish, M., Blumberg, A.J.: Verifying computations without reexecuting them: from theoretical possibility to near-practicality. Communication of the ACM (2015)