A Biometric Self Authentication Scheme

Hervé Chabanne[1,2][a]
[1]*IDEMIA, Paris, France*
[2]*Télécom Paris, France*

Keywords: Biometrics, Authentication Protocol, Verifiable Computation.

Abstract: Biometric authentication systems aim to authenticate users with their physiological characteristics. They have to deal with the inherent noise that occurs when biometric data from the same individual are captured several times. Leveraging verifiable computation techniques, we introduce a new biometric authentication scheme where users bring proofs of who they pretend to be, while keeping their biometrics. Experimental results show that our scheme is practical. We detail a real-life use-case of our anonymous self-scan protocol.

1 INTRODUCTION

Progresses in deep learning today enable to recognise people with tremendous performances. For biometrics, such as, for instance, faces (Wang and Deng, 2021), a vector called *template* is extracted from images of the same biometric trait, and associated to his owner. Unfortunately, for the same person, the template may slightly vary due to several reasons: surrounding perturbations, posture, sensor noise, aging. Biometric authentication has to deal, unlike conventional cryptographic authentication protocols, with different templates representing the same individual.

The difference between two templates is measured based on some metric (e.g. Hamming or Euclidean distance) in a vector space and all templates linked with the same individual lie, with a strong probability, within a maximum distance from a base template called *reference template* which was acquired, during an *enrollment* step. Therefore, the distance between a fresh and the reference templates is compared to a threshold which is fixed for a given system. Whenever the distance is inferior to the threshold, we consider to have a *match*.

In broad words, in our proposal, a user first, during enrollment phase, commits to his reference template. Later, when he wants to authenticate himself thanks to another template originating from a real-time reading of his features, the user has to convince a verifier that his new template is close to the committed reference template and a proof based on Verifiable Computing (VC) (Parno et al., 2013; Costello et al., 2015) for the distance evaluation operation is emitted. This way, leveraging on VC techniques, we can get a self-scan procedure.

Furthermore, to guarantee privacy of the underlying biometrics data, we are going to use *commitment schemes* (Blum, 1981) for the commitments of biometric templates and send zero-knowledge proofs of their opening by the user to the verifier. At the end, the user will authenticate himself without ever disclosing the reference or the fresh templates.

A practical illustration is airplane boarding where a company wants to verify that the person who is boarding the airplane actually is the one who is registered in the ticket. In an imaginary scenario based on our proposal, the passenger's reference template, which, for instance, can be derived from the biometric passport, is included in the ticket using the commit operation that *hides* the reference template. This use case relevantly capitalizes on the features of our biometric authentication scheme: the passenger's reference and fresh templates are kept secret thanks to the privacy-preserving aspect of the protocol, the main goal of boarding check is achieved based on the *binding* property with respect to the authentication of the passenger. Finally, we imagine that passengers runs our protocol walking between the entry and exit gates of boarding corridors.

Another example where our proposal is directly applicable is for retrieving your car in self-service car rentals. We think that numerous other applications of our anonymous self-scan protocol are emerging

[a] https://orcid.org/0000-0002-5916-3387

Chabanne, H.
A Biometric Self Authentication Scheme.
DOI: 10.5220/0011874400003405
In *Proceedings of the 9th International Conference on Information Systems Security and Privacy (ICISSP 2023)*, pages 749-756
ISBN: 978-989-758-624-8; ISSN: 2184-4356

2 RELATED WORKS

The analogy between biometric authentication and commitment schemes has already been made before. Fuzzy commitment schemes have been proposed by Juels and Wattenberg (Juels and Wattenberg, 1999) who defined the syntax and security goals of such schemes and proposed an instantiation based on error-correcting code. Their scheme apply to biometric templates for which the approximate relationship is the Hamming distance. They proved the security of their scheme if the input noisy data is uniformly distributed in the input space. However, biometric data are not uniformly distributed (see (Tuyls et al., 2007) for an overview of security applications dealing with noisy data). In contrast, our scheme leverages a standard commitment scheme to conceal the biometric data. We thus do not have to make any assumption on the distribution of noisy data for the security of our scheme. Moreover, compared to the previous proposals, our scheme has the advantage of being generic and can deal with various distances such as Hamming or Euclidean distance.

Another major difference between our proposal and Juels and Wattenberg's scheme is related to their security. Keller et al. (Keller et al., 2020) show that the fuzzy commitment scheme (Juels and Wattenberg, 1999) offers insufficient protection to biometric templates when produced by deep learning systems such as (Schroff et al., 2015; Wu et al., 2018; He et al., 2016). They present a reconstruction attack that takes a protected template, and recovers a facial image with a good chance of success. Note that a similar attack against a fingerprint access system has also been reported by (Hidano et al., 2012). Finally, (Rathgeb and Uhl, 2012) presents a statistical attack against fuzzy commitment schemes. Our proposal does not suffer from these shortcomings. Indeed, thanks to verifiable computation, the opening of the reference template is performed by the user. We thus obtain the privacy of the reference template beyond the opening of the commit.

Despite their convenience, biometric-based systems offer several attack points, which are for example listed in Jain et al. (Jain et al., 2006). Compared to several privacy-preserving biometric authentication systems, as our proposal does not leak information about the result of the distance computation, it is therefore immune to hill-climbing attacks (Simoens et al., 2012), whose strategy consists of recovering the reference template by successive trials, the next presented template being built by leveraging the last submitted template and score of the previous attempt

2.1 Liveness Detection

Replay attacks are particularly relevant: a fake biometric trait may be presented at the sensor. The same can happen at the template level: a stolen template may be sent again to the biometric matcher in a later authentication. Extensive research have been conduced to thwart these attacks, they belong to the field of *liveness detection* which is itself a whole research area (Marcel et al., 2019).

In order to leave the whole process to the user, we would also need to prove that his template was correctly extracted from his biometrics and that a liveness detection algorithm has accepted the biometric trait as a legitimate one. This goal is not reachable in the actual state of the art in verifiable computing. Therefore, we assume that a liveness detection algorithm is run on the user's device and make the following assumption, similar to (Abidin et al., 2016): we assume that the client's device that extracts his template is trusted in the sense that the biometric sensor and the biometric templates are only accessible to the authorized applications of the device; such trusted environment is today common for mobile devices (Zinkus et al., 2022). This precludes the replay during the authentication phase – and, even from his owner – of the reference biometric trait or the direct injection of any other images; i.e. we make the hypothesis that all the templates in our protocol are coming from a trusted sensor.

3 OVERVIEW OF OUR PROPOSAL

The protocol takes place between a prover called U and a verifier called S. From a high-level view, our goal is to allow a party to commit to a concealed reference value and to later prove that a freshly captured value is close to the committed one *without* revealing any of those. We stress that both the fresh and the reference value are known by the prover but are kept secret from a verifier, who only sees commitments on these values.

First, U commits to his reference template t_{ref} that will be kept secret, sends that commitment c_{ref} to S. U stores t_{ref} and the randomness involved in the commitment computing.

When U wants to authenticate himself, S sends him a nonce r_{nonce}, U captures his own fresh template, denoted by t_{live}, and compares t_{ref} and t_{live}.

The function $\text{match}(t_{\text{ref}}, t_{\text{live}})$ should thus achieve fuzzy matching in the sense that even if t_{ref} and t_{live} are not equal, they should still match if they are close enough in the sense of a distance computation.

Since the verifier S has no control on the freshly captured value, we ask the prover U to commit on that fresh value and to send S this commitment, c_{live}. Note that U has to use r_{nonce} – along with his a random value r_{live} of his own – to compute c_{live}.

At the end of the protocol the verifier is in possession of two commitments that he cannot open, the prover leverages the proof scheme to prove that the commitments open to two values t_{ref} and t_{live} that match. The proof π should therefore not reveal any information about the opened values except that i) the opening was computed following a protocol on which the verifier and the prover have agreed upon and that ii) these opened values match. U finally returns the proof mentioned above to authenticate himself and S checks the validity of the proof. If the proof passes, S can assume with overwhelming probability that the freshly captured value matches with the reference one that has been committed in the beginning of the protocol.

4 BUILDING BLOCKS

4.1 Verifiable Computation

A zero-knowledge Succinct Non-Interactive Argument of Knowledge (Bitansky et al., 2012) (zk-SNARK) is a protocol where a computationally bounded prover convinces a verifier that a statement is true without revealing the verifier anything on the statement witness. Additionally, a zk-SNARK proof has short length and requires no interaction between the verifier and the prover to produce the proof nor to verify its validity. The zk-SNARKs we will be interested in are also publicly verifiable, meaning that anyone can verify the resulting proof. All the existing implemented systems can produce proofs on the **NP**-complete language of circuit satisfiability: the computation to be proved has to be expressed as an arithmetic circuit, i.e. as an acyclic graph carrying finite field values where nodes are either additions or multiplications. Several practical instantiations of zk-SNARKs have been published (Parno et al., 2013; Ben-Sasson et al., 2013; Costello et al., 2015; Wahby et al., 2015; Groth, 2016), building on Gennaro et al.'s seminal paper (Gennaro et al., 2013), which introduces Quadratic Arithmetic Programs (QAPs) as a mean to efficiently encode circuit satisfiability.

4.1.1 Syntax

Let C be a **NP**-statement, expressed as an arithmetic circuit over a finite field $\mathbb{F}$ and λ be a security parameter. We denote by w a witness of a valid statement. A zk-SNARK scheme is defined by three polynomial-time algorithms $(\mathtt{G}, \mathtt{P}, \mathtt{V})$, such that:

- $(ek_C, vk_C) \leftarrow \mathtt{G}(1^\lambda, C)$: the randomized $\mathtt{G}$ algorithm takes as input a security parameter and an arithmetic circuit and produces two public keys, an evaluation key ek_C and a verification key vk_C.
- $(y, \pi) \leftarrow \mathtt{P}(ek_C, x, w)$: the deterministic $\mathtt{P}$ algorithm, takes as inputs a value x, a witness w and the evaluation key ek_C and outputs $y = C(x, w)$ along with π, a proof that y has been correctly computed.
- $\{0,1\} \leftarrow \mathtt{V}(vk_C, x, y, \pi)$: Given the verification key vk_C, the deterministic $\mathtt{V}$ algorithm outputs 1 if there exists a witness w such that $y = C(x, w)$ and 0 otherwise.

Note that the $\mathtt{G}$ algorithm occurs in a pre-processing phase. It has to be run once for a circuit and can be reused over multiple instances. A zk-SNARK reaches the following security properties that we describe informally.

- *Completeness:* if there exists a satisfying witness for the statement C, the verifier should always accept a proof produced by an honest prover.
- *Proof of Knowledge:* if the proof of a statement C is accepted by the verifier, it means that not only a witness exists for the statement but also that the prover indeed knows this witness. This is formalized with an efficient algorithm that is able to extract the witness from the proof.
- *Zero-Knowledge:* the proof reveals no more information about the witness that what could be inferred from the result of the computation. This is formalized with an algorithm that can simulate proofs, zero-knowledge is thus reached if a verifier cannot distinguish between a proof produced by the prover and a proof produced by the simulator.
- *Efficiency:* the proof has a polynomial size *in the security parameter* and the verifier algorithm runs in time that is polynomial in the security parameter and the input length.

The classical *Soundness* property which states that no cheating prover can convince an honest verifier that a false statement is true, except with some small probability can be hardened to *Knowledge Soundness* where a malicious prover cannot convince the verifier unless he knows the witness for the statement.

Finally, *Simulation Soundness* property tells whether an adversary can cheat knowing simulated proofs. Note that knowledge soundness combined with simulation soundness implies non-malleability of proofs.

4.1.2 Security

Gennaro et al. show in (Gennaro et al., 2013) that the above properties are reached for pairing-based zk-SNARKs built from QAPs as long as a Diffie-Hellman like assumption and a knowledge of exponent assumption in bilinear groups (Groth, 2010) hold. Groth (Groth, 2016) obtains an efficient pairing-based zk-SNARK at the cost of proving security in the generic group model. We denote this scheme Groth16 in the following.

4.2 Commitment Schemes

Commitment schemes (Blum, 1981) involve two parties and allow one party called U to commit to a message m to another party S. Later, the commitment can be opened, which means that U reveals the committed message. The commitment scheme is secure if it is *hiding* and *binding*. The hiding property guarantees that the commitment reveals no information about the message itself while the binding property states that, once a message is committed, the commitment cannot be opened to another value than the committed one. It is also required that the scheme is *correct*: if U opens the commitment to the good message, S must accept it. Depending on the scheme goal, the hiding and binding properties can be unconditional or computational but hiding and binding properties cannot be simultaneously unconditional (Fischlin, 2001). Syntax of non-interactive commitment schemes are given below.
Parameter Generation: $p_k \leftarrow \texttt{KeyGen}(1^\lambda)$: From a security parameter λ, the key generation algorithm KeyGen outputs a public key p_k.
Commit Stage: $c \leftarrow \texttt{Commit}(p_k, m; r)$: the commitment algorithm Commit takes a message m, a public key p_k and some randomness r and outputs a commitment c.
Open Stage: $b \in \{0,1\} \leftarrow \texttt{Verif}(p_k, c, r, m)$: the verification algorithm Verif takes as input a public key p_k, a message m, a commitment c and an opening value r and outputs 1 if c opens to m and 0 otherwise.

5 SECURITY PROPERTIES

Our proposal is an application of well-known cryptographic techniques and directly inherits its security properties from them.
Correctness. Following a definition by Abidin et al. (Abidin et al., 2016) that says that a biometric authentication protocol is correct if for all enrolled users with corresponding reference template t_{ref_i} and for all fresh biometric template t_{live_j}, successful authentication occurs when and only when $d(t_{\text{ref}_i}, t_{\text{live}_j}) < \tau$. It is easy to see that our scheme fulfils these requirements of correctness.

Besides correctness, on the one hand, we want users' privacy, i.e. the confidentiality of their biometrics and on the other hand, as biometric authentication is entirely completed by users, the elements they send must be faithful about how authentication goes; moreover, we have to enforce the fact that the biometric data involved in this procedure correspond to the actual person who is authenticating.

5.1 Confidentiality of Biometric Templates

By design, all biometric data are kept by their owner.

We rely on the hiding property of the commitment schemes to ensure the confidentiality of t_{ref} (resp. t_{live}) having access to c_{ref} (resp. c_{live}). The only other elements that are revealed by users are proofs. The zero-knowledge property of the VC system implies that S learns anything from π other than the value of the proven statement.

5.2 Delegation of Biometric Authentication Protocol to U

Our self-authentication scheme must be carried out as it was realized by S.

We introduce two procedural measures: trusted sensors (see Sec. 2.1) and boarding corridors (see Sec. 7.1).

Trusted sensors ensure that templates are extracted from the biometric traits of real persons.

The idea behind boarding corridors is to isolate users from the rest of the world during the authentication phase; preventing them from using their smartphone with their biometric data and then lend it to someone else. Note that at the entry gate of the corridor, a nonce r_{nonce} is sent by S, triggering the authentication phase; the use of r_{nonce} is made mandatory for the computation of c_{live} (see Sec. 6). In corridors, users perform a fresh biometric capture, getting

t_{live} and compute c_{live} together with the proof π. Our protocol is fast enough for being executed while the passenger is crossing the corridor. If the authentication succeeds, the exit boarding gate opens to let the passenger reach his plane.

Binding of U. The binding property of commitment schemes ensures that c_{ref} (resp. c_{live}) sent by U cannot be opened by other values than corresponding biometric template t_{ref} (resp. t_{live}).

π also indicates if t_{ref} is close to t_{live} and, whether or not r_{nonce} has been used during the computation of c_{live}.

Non Malleability of π. If the proof π is malleable (and in particular is not simulation sound), then a malicious party could impersonate the prover by taking a proof computed with respect to a random value – for instance, before the entrance of the passenger inside a boarding corridor – transforming it into a new proof for another value, e.g. r_{nonce} To thwart this threat, we have to use a zk-SNARK scheme which guarantees that an adversary cannot change π to prove a new statement. Note that the original Groth's zk-SNARK Groth16 is indeed malleable (Groth and Maller, 2017) but that there are alternatives (Baghery et al., 2020; Baghery et al., 2021; Atapoor and Baghery, 2019) providing non-malleability while achieving an efficiency close to Groth16.

6 OUR PROPOSAL

In this section, we describe in details our protocol overviewed in Section 3. We keep the notations defined in Section 4.

Let $\mathcal{C}$ be the circuit implementing the function f taking as input $(t_{\text{ref}}, t_{\text{live}}, r_{\text{ref}}, r_{\text{live}}, c_{\text{ref}}, c_{\text{live}})$ and returning:

$$\begin{cases} d_{\text{ref}} & \leftarrow & \texttt{Verif}(p_k, c_{\text{ref}}, r_{\text{ref}}, t_{\text{ref}}) \overset{?}{=} 1 \\ d_{\text{live}} & \leftarrow & \texttt{Verif}(p_k, c_{\text{live}}, r_{\text{auth}}, t_{\text{live}}) \overset{?}{=} 1 \\ d_{\text{dist}} & \leftarrow & d(t_{\text{ref}}, t_{\text{live}}) \overset{?}{<} \tau \\ d_{\text{nonce}} & \leftarrow & r_{\text{auth}} \overset{?}{=} r_{\text{nonce}} || r_{\text{live}} \end{cases} \tag{1}$$

where $r_{\text{auth}} = r_{\text{nonce}} || r_{\text{live}}$ with $||$ standing for the concatenation; r_{nonce} being sent by the verifier S and r_{live} being randomly chosen by the user.

The function f verifies in d_{dist} a matching between t_{live} and t_{ref} under the threshold τ, in d_{ref} (resp. d_{live}) commitments opening to t_{ref} (resp. t_{live}) and in d_{nonce} that r_{auth} is well-formed, beginning with r_{nonce}. The inputs t_{ref} and t_{live} and the commitments opening should be private in (1), i.e. the proof that f was correctly computed should reveal no information about t_{live} and t_{ref} nor about the computation of the commitment openings.

The protocol proceeds in the following way:

In **the setup phase**, the `KeyGen` algorithm of the commitment scheme is run to get the public key p_k later used in the commit function.

Then, the `G` algorithm of the zk-SNARK scheme is run and the evaluation key $ek_{\mathcal{C}}$ and the verification key $vk_{\mathcal{C}}$ are generated. The input circuit of the algorithm is the circuit $\mathcal{C}$ implementing the function f defined in formula (1). The distance and the threshold are algorithm dependent and are set once the circuit is defined. The commitment verification function `Verif` is also set in the circuit. The commit public key and the evaluation and verification keys are thus supplied to U and S.

We assume that there exists a function denoted by `GetTemplate` that outputs the biometric template of the user. We make the assumption that templates of the same person extracted thanks to `GetTemplate` are taken at random within a distance τ. I.e. for a given biometric trait, two calls to `GetTemplate`, will output t, t' such that $d(t, t') < \tau$. This is validated in practice where multiple acquisitions of the same biometric trait lead to different templates.

During **the enrollment phase**, U calls `GetTemplate` and obtains a reference template t_{ref} from his biometric trait. He gets $c_{\text{ref}} = \texttt{Commit}(p_k, t_{\text{ref}}; r_{\text{ref}})$ the commitment associated to t_{ref} and sends it to S.

Remark 1. *We assume that the function* `GetTemplate` *is implemented by a trusted sensor. The same is true for the commitments, i.e. the function* `CommitReferenceTemplate` *(resp.* `CommitFreshTemplateProve`*) is implemented by a trusted environment during the enrollment (resp. authentication) phase (see Sec. 2.1).*

For **the authentication phase**, U calls again the `GetTemplate` function and gets a fresh biometric template t_{live}.

U picks a random value r_{live}, gets another random value r_{nonce} from S and uses it to compute a commitment $c_{\text{live}} = \texttt{Commit}(p_k, t_{\text{live}}; r_{\text{auth}})$ where $r_{\text{auth}} = r_{\text{nonce}} || r_{\text{live}}$. Using the evaluation key $ek_{\mathcal{C}}$, U computes a proof π of correctness for the function f defined in (1). In this computation, t_{ref}, t_{live}, r_{ref} and r_{live} are private inputs for the prover. This means that, leveraging the zero-knowledge functionality of the zk-SNARK scheme, the proof π does not reveal information about them. Moreover, the argument of knowledge property of the zk-SNARK guarantees that the prover is in possession of t_{ref} and t_{live}. On the contrary, the values c_{ref} and c_{live} are public inputs in function f and the verifier can supply them in the zk-

Setup:

Given the circuit C associated to function f defined by formula (1)

$(ek_C, vk_C, p_k) \leftarrow \texttt{Setup}(1^\lambda)$,

where: $\begin{cases} (ek_C, vk_C) & \leftarrow & \texttt{G}(f, 1^\lambda) \\ p_k & \leftarrow & \texttt{KeyGen}(1^\lambda) \end{cases}$

Enrollment:

GetReferenceTemplate:

$t_{\text{ref}} \leftarrow \texttt{GetTemplate}()$

CommitReferenceTemplate:

$c_{\text{ref}} \leftarrow \texttt{Commit}(p_k, t_{\text{ref}}; r_{\text{ref}}), r_{\text{ref}} \overset{\$}{\leftarrow} \{0,1\}^n$

Authentication:

GetFreshTemplate:

$t_{\text{live}} \leftarrow \texttt{GetTemplate}()$

CommitFreshTemplateProve:

$(c_{\text{live}}, \pi) \leftarrow \texttt{CommitFTProve}(ek_C, p_k, t_{\text{ref}}, t_{\text{live}}, r_{\text{ref}}, r_{\text{auth}}, c_{\text{ref}})$

where:

- $r_{\text{live}} \overset{\$}{\leftarrow} \{0,1\}^n$
- $c_{\text{live}} \leftarrow \texttt{Commit}(p_k, t_{\text{live}}; r_{\text{auth}})$, where $r_{\text{auth}} = r_{\text{nonce}} || r_{\text{live}}$
- $\pi \leftarrow \texttt{P}(ek_C, t_{\text{ref}}, t_{\text{live}}, r_{\text{ref}}, r_{\text{auth}}, c_{\text{ref}}, c_{\text{live}})$

Matching:

$out \leftarrow \texttt{V}(vk_C, p_k, \pi, c_{\text{ref}}, c_{\text{live}})$.

Figure 1: Instantiation Syntax of Our Authentication Protocol (Functions in boxes are implemented inside U's smartphone trusted environment).

SNARK verification algorithm V. U then sends S the proof π and the second commitment c_{live}. Note that π contains the result of the matching since the function checks that the computed distance is inferior to the threshold.

In the following, we write CommitFTProve for the function which outputs both the commitment c_{live} of the fresh biometric template t_{live} together with the proof π.

For **the matching phase**, using the verification key vk_C and the inputs c_{ref} and c_{live}, S checks the validity of the proof. If the proof verification passes, the matching algorithm V outputs *accept* and S can conclude that U is in possession of a fresh template t_{live} that matches with the initially committed template t_{ref}.

The syntax of our protocol is given in Fig. 1, with the notations of Sec. 4.

7 APPLICATIONS

7.1 Use Case: Privacy-Preserving Boarding Check Systems

Let us consider a scenario where an airline proposes a privacy-preserving boarding check for its flights. We assume that users who want to benefit of this scheme are in possession of a smartphone, which is equipped with a face recognition algorithm with liveness detection.

User U will proceed in two steps to board. First, U extracts a template from his face, commits to it and builds an airplane ticket containing the commitment and additional information such as the flight number, the seat number and the validity duration of the ticket. This ticket is signed by the airline and sent back to U.

In a second phase, when U arrives at the airport, he has to cross a boarding corridor equipped with a device able to receive data and to store the signature verification keys and the verification keys vk_C of the system. In this corridor, he runs our protocol. Using the signature public key, the device runs the signature verification algorithm to check the authenticity of the commitment c_{ref}. If the signature verification passes, the device then checks that the ticket is valid by simply checking the date of validity field of the ticket. If so, the device runs the algorithm V with inputs c_{ref}, c_{live} and π to grant access to U .

7.1.1 Law Enforcement

The previous scenario enables to perform authentication without needing the user's identity. In real life, this might not seem realistic since no airline can accept to board a customer without having his name. Indeed, many countries require airlines to provide personal information about the passengers they carry. We thus propose a slight change to the previous scenario in order to take this requirement into account: during the registration phase, user U sends to a trusted party (which we can be for instance the state where the airplane ticket is bought or the ICAO) not only his commit, flight number and seat but also his identity information (e.g. a photo of his passport). The trusted party can check the latter to verify for instance that user U does not belong to a blacklist. Once the security checks have been performed, the trusted party can remove the identity of the user and sign the same ticket as in the previous section. The protocol then proceeds as before.

Table 1: Experimental results on the VC part.

Template component numbers	EK size	VK size	`Keygen`	`Prove`	`Verify`	Proof size
128	4.5 MB	61 kB	1.91 s	0.66 s	0.0015 s	120 B
256	8.2 MB	61 kB	3.37 s	1.32 s	0.0015 s	120 B
384	11.7 MB	61 kB	4.90 s	1.75 s	0.0015 s	120 B
512	15.9 MB	61 kB	6.46 s	2.57 s	0.0015 s	120 B

8 EXPERIMENTS

We here report some figures about the implementation of our proposal. The experiments were performed on a 8-core machine running at 2.9 GHz with 16 GB of RAM, using no parallelization. The verifiable computation system is Groth16 (Groth, 2016) – which seems to us representative in terms of performances – and its implementation within the `libsnark` library (available at https://github.com/scipr-lab/libsnark). We first assume that the biometric face recognition system is Schroff et al.'s Facenet (Schroff et al., 2015). This state of the art convolutional neural network algorithm takes as input a face image and outputs a vector with 128 floating-point numbers components. To decide if two vectors come from the same individual an Euclidean distance is computed. Afterwards, we also experiment the scalability of our scheme by letting the number of components of the templates increase.

Recall that to be able to verify a computation, a circuit representing this computation needs to be designed. Moreover, minimizing the size of the circuit improves the performance of the zk-SNARK scheme. We thus implement a squared euclidean distance to avoid an expensive square root computation. We also convert the floating-point numbers into a fixed-point representation on 32 bit integers. Doing that way, we keep the same biometric performances. The resulting biometric template is thus 4096 bit long. Regarding the commitment scheme, we choose Kawachi et al.'s commitment scheme (Kawachi et al., 2008) which is built from Ajtai hash function (Ajtai, 1996). Ajtai hash function is provably secure and is efficiently implementable in the zk-SNARK efficiency model (Kosba et al., 2015) (recall that a computation to be verified has to be expressed as a circuit over a finite field and that addition in this circuit are free). Kawachi et al. prove that the resulting commitment scheme is statistically hiding and computationally binding. Let q be the prime number equal to the cardinality of the prime field where the computations of the zk-SNARK scheme take place. Let (n,m) be two integers such that: $m > n \log q$. The Ajtai hash function family is defined as $\mathcal{H}(q,m) = \{f_{\mathbf{A}} \colon \{0,1\}^m \to \mathbb{F}_q^n | \mathbf{A} \in \mathcal{M}_{n,m}(\mathbb{F}_q)\}$ where $f_{\mathbf{A}}(x) = \mathbf{A} \cdot x \mod q$.
Let B and C be two randomly chosen (n,m)-matrices in $\mathcal{M}_{n,m}(\mathbb{F}_q)$. Denoting by A the matrix defined by $A = [B\ C]$ and by ρ a randomly chosen value in $\{0,1\}^m$, the commitment of an input string $s \in \{0,1\}^m$ is thus: $Com_{\mathbf{A}}(s;\rho) := f_{\mathbf{B}}(\rho) + f_{\mathbf{C}}(s)$. Using Merkle-Damgård construction, Kawachi et al. extend this commitment scheme to arbitrary length strings in (Kawachi et al., 2008). Based on Kosba et al. (Kosba et al., 2015), we choose q to be the 254-bit prime where arithmetic circuits are defined, $m = 1524$ and $n = 3$.
Table 1 reports our proving and verification times. Note that the key generation algorithm, besides being efficient, has only to be run once: the same evaluation and verification keys can be reused after that.

ACKNOWLEDGMENTS

A first version of this work has been written by Julien Keuffer while he was working at Idemia, his PhD supervisor Refik Molva and me. I thank Melek Önen and Vincent Despiegel for their help.

REFERENCES

Abidin, A., Aly, A., Argones-Rúa, E., and Mitrokotsa, A. (2016). Efficient verifiable computation of XOR for biometric authentication. In *CANS*, volume 10052 of *Lecture Notes in Computer Science*, pages 284–298.

Ajtai, M. (1996). Generating hard instances of lattice problems (extended abstract). In *STOC*, pages 99–108. ACM.

Atapoor, S. and Baghery, K. (2019). Simulation extractability in groth's zk-snark. In *DPM/CBT@ESORICS*, volume 11737 of *Lecture Notes in Computer Science*, pages 336–354. Springer.

Baghery, K., Kohlweiss, M., Siim, J., and Volkhov, M. (2021). Another look at extraction and randomization of groth's zk-snark. In *Financial Cryptography (1)*, volume 12674 of *Lecture Notes in Computer Science*, pages 457–475. Springer.

Baghery, K., Pindado, Z., and Ràfols, C. (2020). Simulation extractable versions of groth's zk-snark revisited. In *CANS*, volume 12579 of *Lecture Notes in Computer Science*, pages 453–461. Springer.

Ben-Sasson, E., Chiesa, A., Genkin, D., Tromer, E., and Virza, M. (2013). Snarks for C: verifying program executions succinctly and in zero knowledge. In *CRYPTO (2)*, volume 8043 of *Lecture Notes in Computer Science*, pages 90–108. Springer.

Bitansky, N., Canetti, R., Chiesa, A., and Tromer, E. (2012). From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *Innovations in Theoretical Computer Science 2012*, pages 326–349.

Blum, M. (1981). Coin flipping by telephone. In *CRYPTO*, pages 11–15. U. C. Santa Barbara, Dept. of Elec. and Computer Eng., ECE Report No 82-04.

Costello, C., Fournet, C., Howell, J., Kohlweiss, M., Kreuter, B., Naehrig, M., Parno, B., and Zahur, S. (2015). Geppetto: Versatile verifiable computation. In *IEEE Symposium on Security and Privacy*, pages 253–270. IEEE Computer Society.

Fischlin, M. (2001). *Trapdoor commitment schemes and their applications*. PhD thesis, Goethe University Frankfurt, Frankfurt am Main, Germany.

Gennaro, R., Gentry, C., Parno, B., and Raykova, M. (2013). Quadratic span programs and succinct nizks without pcps. In *EUROCRYPT*, volume 7881 of *Lecture Notes in Computer Science*, pages 626–645. Springer.

Groth, J. (2010). Short pairing-based non-interactive zero-knowledge arguments. In *ASIACRYPT*, volume 6477 of *Lecture Notes in Computer Science*, pages 321–340. Springer.

Groth, J. (2016). On the size of pairing-based non-interactive arguments. In *EUROCRYPT (2)*, volume 9666 of *Lecture Notes in Computer Science*, pages 305–326. Springer.

Groth, J. and Maller, M. (2017). Snarky signatures: Minimal signatures of knowledge from simulation-extractable snarks. In *CRYPTO (2)*, volume 10402 of *Lecture Notes in Computer Science*, pages 581–612. Springer.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *CVPR*, pages 770–778. IEEE Computer Society.

Hidano, S., Ohki, T., and Takahashi, K. (2012). Evaluation of security for biometric guessing attacks in biometric cryptosystem using fuzzy commitment scheme. In *BIOSIG*, volume P-196 of *LNI*, pages 1–6. GI.

Jain, A. K., Ross, A., and Pankanti, S. (2006). Biometrics: a tool for information security. *IEEE Trans. Inf. Forensics Secur.*, 1(2):125–143.

Juels, A. and Wattenberg, M. (1999). A fuzzy commitment scheme. In *CCS*, pages 28–36. ACM.

Kawachi, A., Tanaka, K., and Xagawa, K. (2008). Concurrently secure identification schemes based on the worst-case hardness of lattice problems. In *ASIACRYPT*, volume 5350 of *Lecture Notes in Computer Science*, pages 372–389. Springer.

Keller, D., Osadchy, M., and Dunkelman, O. (2020). Fuzzy commitments offer insufficient protection to biometric templates produced by deep learning. *CoRR*, abs/2012.13293.

Kosba, A. E., Zhao, Z., Miller, A., Qian, Y., Chan, T. H., Papamanthou, C., Pass, R., Shelat, A., and Shi, E. (2015). How to use snarks in universally composable protocols. *IACR Cryptol. ePrint Arch.*, 2015:1093.

Marcel, S., Nixon, M. S., Fiérrez, J., and Evans, N. W. D., editors (2019). *Handbook of Biometric Anti-Spoofing - Presentation Attack Detection, Second Edition*. Advances in Computer Vision and Pattern Recognition. Springer.

Parno, B., Howell, J., Gentry, C., and Raykova, M. (2013). Pinocchio: Nearly practical verifiable computation. In *IEEE Symposium on Security and Privacy*, pages 238–252. IEEE Computer Society.

Rathgeb, C. and Uhl, A. (2012). Statistical attack against fuzzy commitment scheme. *IET Biom.*, 1(2):94–104.

Schroff, F., Kalenichenko, D., and Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In *CVPR*, pages 815–823. IEEE Computer Society.

Simoens, K., Bringer, J., Chabanne, H., and Seys, S. (2012). A framework for analyzing template security and privacy in biometric authentication systems. *IEEE Trans. Inf. Forensics Secur.*, 7(2):833–841.

Tuyls, P., Skoric, B., and Kevenaar, T., editors (2007). *Security with noisy data : on private biometrics, secure key storage and anti-counterfeiting*. Springer, Germany.

Wahby, R. S., Setty, S. T. V., Ren, Z., Blumberg, A. J., and Walfish, M. (2015). Efficient RAM and control flow in verifiable outsourced computation. In *NDSS*. The Internet Society.

Wang, M. and Deng, W. (2021). Deep face recognition: A survey. *Neurocomputing*, 429:215–244.

Wu, X., He, R., Sun, Z., and Tan, T. (2018). A light CNN for deep face representation with noisy labels. *IEEE Trans. Inf. Forensics Secur.*, 13(11):2884–2896.

Zinkus, M., Jois, T. M., and Green, M. (2022). Sok: Cryptographic confidentiality of data on mobile devices. *Proc. Priv. Enhancing Technol.*, 2022(1):586–607.