An architecture for practical confidentiality-strengthened face authentication embedding homomorphic cryptography

Nabil Bouzerna, Renaud Sirdey, Oana Stan, Thanh Hai Nguyen, Philippe Wolf

An architecture for practical confidentiality-strengthened face authentication embedding homomorphic cryptography

Nabil Bouzerna*, Renaud Sirdey*†, Oana Stan*†, Thanh Hai Nguyen*†, Philippe Wolf*
*IRT SystemX
8, av. de la Vauve, 91120 Palaiseau, France
Email: name.surname@irt-systemx.fr
†CEA, LIST,
91191 Gif-sur-Yvette Cedex, France
Email: name.surname@cea.fr

***Abstract*—In this paper, we propose and experiment a system architecture which intends to significantly strengthen the security of biometric authentication with respect to the confidentiality(-by-design) of the users' references needed to perform such a function. Our architecture has been designed to ensure that these biometric references are permanently encrypted and that the (single) server processing them has no decryption capability (in particular, does not have access to any decryption key). In order to do so, we use homomorphic encryption techniques which allow to perform calculations directly over encrypted data. We report on the careful architectural choices and agressive optimizations we had to make in order to be able to deploy an off-the-shelf face recognition module into this architecture. As the performance results presented in the paper demonstrate, we claim to have achieved practically relevant levels of performance and security in a realistic setting.**

1. Introduction

This paper reports on an experiment which aim was to demonstrate that homomorphic encryption techniques are amenable to practical performances in a clearly realistic setting.

As a demonstration playground, we have chosen to work on a cloud-based face authentication function and attempted to use homomorphic cryptographic techniques in order to practically strengthen the confidentiality(-by-design) of the users authentication references with respect to threats coming from the server running the function. The exercice was done genuinely in the sense that the payload function as well as the software implementing it was chosen a priori. In other words, the exercise we have done was more of the kind "a priori take an arbitrary interesting function and make it practically run over encrypted data in a meaningful architecture" (adressing at least part of a meaningful threat model) rather than "take a function which is well suited to practically run over encrypted data and a posteriori convince everyone it is useful". Of course, reality is never as clearcut as one wishes, but the former spirit more genuinely reflects our state of mind than the latter. So let us consider a setting in which an employer wishes to deploy face-based authentication to complement badge-based authentification on its premises and, in order to do so, needs to store (and process) authentication references for many of its employees on a server. In order to improve the confidentiality of these references, we propose a three parties architecture and process involving the employer, the employee (enrollment is performed at home) and a third party who is responsible for key management. The proposed architecture, which is described in due details later on, has the desirable property of splitting the responsiblity for employees' references confidentiality between the employer (which is entrusted with the encrypted references but *has no access to any decryption key*) and the aforementioned third party (which is entrusted with the encryption and decryption keys but *has no access to any encrypted data*). Despite of the fact that this model is, of course, not resistant to employer/third party collusion, it significantly strengthens the confidentiality guarantees provided by biometric authentication and does not appear too unrealistic if the third party is able to serve many companies with a turnkey solution kit (including the access devices as we shall later see).

It should also be emphasized that, despite of the use case choice, the present work should not be considered a paper on biometric authentication as the authors do not claim to have an in-depth expertise in that field. On the contrary, the contribution of the paper is more focused towards homomorphic cryptography and, more precisely, the costs (software engineering as well as performances) at which this technology can be brought into practice. Biometric authentication then being a challenging- and demonstrative-enough setting in which our points can be effectively illustrated.

This paper is organized as follows. Section 2 covers background and related work with respect to homomorphic cryptography as well as its (practical) use, with particular emphasis on biometric authentication. Section 3 presents our system architecture proposal and the rationale behind it. Section 4 reports on the pitfalls and issues we had to adress in order to operationally integrate homomorphic encryption in the architecture. Then, section 5 subsequently provides

Preprint of a paper presented at IEEE CloudCom'16

the experimental results that back up our claim of having successfully done so. Lastly, Section 6 concludes the paper with both lessons learnt and perspectives.

2. Background and related work

Since Gentry's 2009 breakthrough [12] demonstrated the theoretical feasability of Fully Homomorphic Encryption (FHE) for computing arbitrary functions directly over encrypted data, fast-paced progresses have been made in terms of both raw performances [13] as well as tooling support [7] for running "real" (lightweight-enough) computer programs over FHE. As a result, we are now very close to be able to deploy these new exciting techniques in real-world settings (see e.g. [8]). Still, the existence of FHE (or rather practically meaningful Somewhat FHE [4], [11][1]) does not necessarily render more restrictive homomorphic systems (such as the Paillier system [19] which is detailed Sect. 4.3) obsolete and deploying these cryptosystems in practical application settings is not without challenges. These challenges however, from the authors' experience, are very close to those of deploying FHE in meaningful settings. So, as we now have a rich toolbox of cryptographic techniques for computing over encrypted data, each with different pros and cons, this is really a matter of chosing the most appropriate cryptographic tool for the setting at hand. In particular, and although we do not claim to be the first to do so, we argue here that there are settings, such as the one presented in this paper (see Sect. 4.2 for details), in which it is enough to use only partially homomorphic encryption (allowing only one operation type, i.e. addition or multiplication but not both over encrypted data) along with well-designed optimization techniques.

In this section, we briefly survey previous works on the use of homomorphic encryption in the field of face recognition.

In [10] a two-party biometric face recognition system is presented, based on two additively homomorphic schemes: Paillier and DGK cryptosystems, under the assumption of honesty for both the authentication server and the user. The protocol uses an Eigen-Face recognition system with the matching consisting of several steps: on the user part, the computation of the encrypted face image and on the server side, the projection of the encrypted face image into the face space, computing Euclidean encrypted distances, selecting the minimum and comparing with a threshold. However, the average runtime for the matching of an image of size 92×112 against a database of 320 facial templates is quite long, taking approximatively 40 seconds on a conventional workstation.

An additively homomorphic scheme, along with oblivious transfer, is also used by Osadchy et al. in [18] for SCiFi, a system for secure computation of face identification. The protocol, relying on a two-party setting, makes the same assumption that both participants are honest and spends a significant time for precalculation and offline steps.

A recent work proposes Ghostshell, an approach for secure biometric authentication on iris detection using a Somewhat Homomomorphic Encryption (SHE) scheme and, for integrity purposes, a Message Authentication Code (MAC) [9]. This system design contains only two participants: the authentication server and the user while ensuring the security against two kind of threats: one coming from the malicious users and a second, on the server side, which would like to learn information about the private users biometrics. Two protocols between the user and the server are also described: a Woodenman protocol, running a MAC generation algorithm over homomorphic encryptions and protecting against the first kind of threats, and the Ironnman Protocol based on discrete logarithm and mitigating threats of the second kind. During the matching phase, the server computes the Hamming distances between the encrypted biometric challenge and the encrypted template stored on the server. The tests were performed using the HELIB library [13] with a BGV-type cryptosystem as well as SIMD optimizations (batching [4]) for packing several biometric features in a single ciphertext. The heaviest computation consists in obtaining the Hamming distances, with at least 500 msec. for the execution time, when varying N, the number of slots and m, the degree of the cyclotomic polynomial for the SHE scheme for an iris code of 2400 bits.

Additional works include [21] (using Gentry's far-from-practical initial FHE scheme [12]), [22] (outsourced identification with an interactive protocol), [20] (also outsourced identification hybridizing homomorphic encryption and garbled circuits) as well as [3] and [15]. Contrary to the present work, these focus mainly on the problem of face identification (determining whether a face is present in a database) rather than authentication (checking whether a face matches a particular entry in a database). Additionally, most, if not all, other works focus on the so-called eigenfaces algorithm [14] for face matching.

The contribution of this work is on face authentication without any hiding requirement with respect to the database record against which matching is performed. Additionally, we focus on a more recent (yet mainstream) face matching algorithm which turns out to be more "homomorphic-friendly". Another important point is that we use (and assess the practicality of) homomorphic encryption as a *focused* countermeasure to confidentiality threats from the server holding the face recognition reference database. With that respect we have obtained what we believe to be a quite pragmatic and practical system architecture (as described in Sect. 3) which illustrates the necessary trade-offs to achieve overall practical homomorphic performances (in terms of bandwidth and storage requirements as well as computing power and latency as argued in Sect. 5). Again, to be completly explicit, we do not claim that our architecture covers a full-blown threat model as, in a real-world setting, homomorphic encryption would only be one (yet important) "security ingredient" targeted at one (yet important) threat, amongst many others (countermeasures and threats).

1. SFHE must be a priori dimensionned in function of the multiplicative depth of the algorithms they are expected to evaluate.

3. Architecture description

This section is devoted to a more detailed presentation of the application set up. We consider the following so-called three parties: Company, Dilbert (an employee) and Dogbert (a third party). The first of these is an employer willing to deploy multi-factor authentication for access to sensitive areas on his premises by requiring that an employee needing to access such an area presents his badge to a card reader as well as checking that the face of the person in front of the door does indeed match the id of the badge. In order to do this, the Company is willing to operate a database with the facial references of most (if not all) of its employees but is also willing to avoid taking responsibility in case this database becomes compromized and its contents exfiltrated. The idea we pursue in this paper is that this (restrictive) goal, as so far stated in loose form, would be achievable if the Company was able to operate such a database in an oblivious fashion, without being granted access to its content. In short, we propose to experiment with an approach using homomorphic encryption techniques which allow—at least to some extent—to operate such a database while keeping its contents encrypted at all time and without providing the server hosting it with any decryption capability. In particular, the server in question would not need to be granted access to any decryption key in order to provide the required authentication service. As such, an attacker seeking to exfiltrate the employees facial references by compromising that server would, by construction, at most be able to exfiltrate encrypted data and would then have no other choices than either attacking the underlying cryptosystem (a presumably difficult task) or attacking yet another entity holding a decryption key.

Now that the high-level requirements have been stated in fairly loose terms, let us provide a more concrete view of the requirements, roles and responsibilites of the parties involved in the use-case architecture.

Figure 1 provides a high-level view of the architecture.

Let us start with Dogbert. For a given Company, Dogbert is responsible for key generation (i.e. generating a public encryption key, pk, and a secret decryption key, sk) and storage (of these keys). It is supposed not to share sk neither with the Company nor with anyone else. He is however supposed to provide pk to Dilbert or any other employee of the Company. The Company's employees i.e., Dilberts, are responsible for acquiring and encrypting their own facial references outside of the Company domain. They do so by interacting first with Dogbert to get the Company's pk (presumably embedded in some enrollment software also provided by Dogbert), and, once acquisition and encryption has been (locally to Dilbert) successfully completed, by transfering their encrypted references towards the Company (that way, as already emphasized, the Company has the data but not the decryption key and Dogbert has the decryption key but access to neither the clear nor the encrypted reference). The Company is then responsible for storing the encrypted references on its internal authentication server which processes them (in the encrypted domain) in response to authentication challenges sent from the access Devices (i.e. some kind of "intelligent" connected door locks). An important point is that the access Devices have to be able to decrypt the results of the calculation done by the authentication server and, hence, have to be provided with sk. As a consequence, despite of the fact that the Devices are (physically) deployed on the Company premises, they are (logically) assumed to remain in Dogbert's domain. In particular, this requires that Dogbert is providing the Devices to the Company and takes the necessary measures to avoid that physical access to a Device leaks access to sk (this may be enforced by different means from low-cost software protections up to deploying specific hardware security support in the Device at greater cost, depending on the overall operational threat model[2]). By construction, when in service, the Devices themselves have no connection to Dogbert (only during commisioning when sk is "burnt" into the Device) as well as no access to the (encrypted) reference data stored on the Company's server (they only have access to calculation results on these data).

In summary, the present architecture aims at enforcing the following properties:

- Key manager (Dogbert) has knowledge of pk and sk but no access to the reference data.
- Access control Devices have knowledge of sk, access to calculation results (distances to compare to a threshold) but no access to the reference data.
- Employer (Company) has no knowledge of sk, hence, neither access to the reference data nor access to (encrypted) calculation results.
- Employee (Dilbert) has to trust that the key manager (Dogbert) will not collude with its employer (Company).

There are several ways in which the above properties and role separations can be consistently mapped on practical architecture guidelines for multifactor physical access systems (see e.g. [1]). However, as already emphasized, covering a full-blown operational threat model is beyond the scope of this paper which much more modestly claims to investigate the practicality of a focused countermeasure against a narrow (yet relevant in many scenarios) confidentiality threat.

As has been hinted above (and this will be confirmed in the sequel), when executing a face-matching algorithm in the encrypted domain, it is tempting to do the final (lightweight) threshold comparison post-decryption in the clear domain. Still, in the context of the algorithm described in Sect. 4.1, it will be clear that a compromized Device could extract the reference data one by one by means of hundreds of thousands of (malicious) requests. We do not consider this an issue with respect to the present architecture as (fairly classical) system-level countermeasures can be deployed to counter these kind of threats coming from the Devices (e.g.

2. As recommended in [1], in highly sensitive settings, it may be required to move the decryption and action capability to another secure server (within the restricted perimeter) in order to ensure full transparency from the Devices (exposed outside of that perimeter). The present architecture can easily be adapted to such more stringent settings.

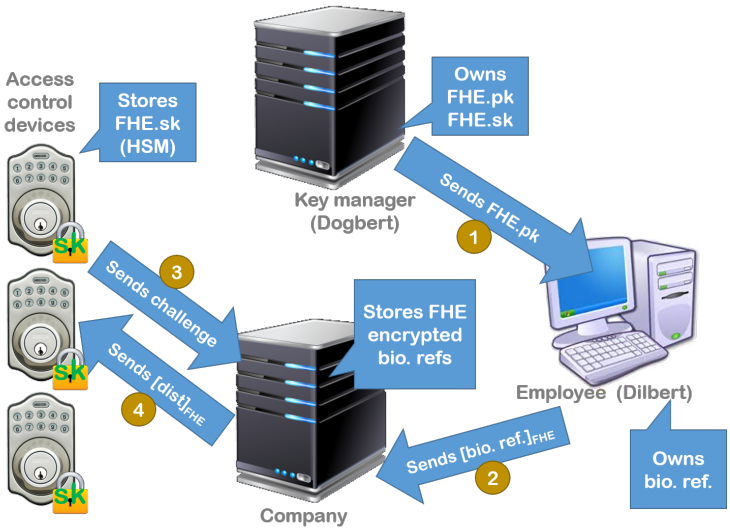

Figure 1. Overall authentication architecture summary.

throttling and/or imposing lower bounds on the challenge size). Furthermore, doing the threshold comparison on the Device may allow more flexibility when multiple factor (including multiple biometry) authentication is used.

4. HE integration

4.1. Face matching algorithm description

In terms of face matching algorithms, the starting point of this work was the OpenIMAJ[3] implementation of the algorithm specified in [17] and based on the so-called Local Ternary Patterns (LTP). OpenIMAJ is a well-known and respected open source set of libraries and tools for multimedia content analysis and exhibits both clean understandable code as well as decent matching performances. This is particularly true of the `LtpDtFeatureComparator` module which we have tried to test in harsh conditions before using it as a reference for our "homomorphization" experiment. Also note that this LTP-based algorithm is also implemented in OpenCV[4] which is the "gold standard" for image processing librairies. Also recall that this is not a paper on biometric authentication, as already emphasized in the introduction, so we have chosen what appeared to be a meaningful algorithm, yet from an outsider's point of view of that field.

It turns out that, from the point of view of the present exercise, the part of the code which has to be executed in the encrypted domain is quite simple.

Enrollment is done using one or more pictures of the individual face and uses a non-straightforward image processing pipeline to generate a reference consisting of 118 so-called distance maps, $M^{(0)}, ..., M^{(117)}$, each of which being an 80×80 matrix with values in $[0, 1]$. Authentication is also done using one or more pictures of the individual and is also based on a similarly non-straightforward processing pipeline which generates a challenge made up of 118 sets of coefficient coordinates $S^{(0)}, ..., S^{(117)}$. We are staying deliberatly vague on these two pipelines for reasons which are explained in the next section (due details can be found in [17]).

The matching part of the algorithm, which is the one which interests us for the sake of the present paper, as will be argued in the next section, then simply requires to compute

$$\ell = \sum_{i=k}^{117} \sum_{(i,j) \in S^{(k)}} M_{ij}^{(k)}. \quad (1)$$

Given a threshold T, authentication then succeeds if $\ell \geq T$ and fails otherwise. We used $T = 92$ in our experiment (that value having been chosen on empirical ground).

4.2. "Going homomorphic"

With respect to the architecture described in Section 3, we are not concerned that much about the image processing pipelines for reference and challenge generation as they are both executed in the clear domain near the camera or at least on the machine to which the camera is attached i.e., on Dilbert's home computer or on Dogbert's access devices. So, as far as the present study is concerned, we can essentially see them as black boxes (emphasizing again that this is a paper on the "homomorphization" of an off-the-shelf biometric authentication function) and do not provide any further details on the generation of both the M's and the S's.

Once generated (on Dilbert's home computer) the distance maps must be encrypted (on that same machine) before they are sent to Company. According to our reference

3. `www.openimaj.org`.
4. `opencv.org`.

architecture, the (public) encryption key is retrieved from Dogbert and Company has no access to the associated decryption key (Dogbert has access to it, but recall that he is never sent the data in the protocol). So once this transfer is done on the Company server, the system is ready to authenticate Dilbert.

The authentication process goes as follows. After acquisition of the image and generation of the challenge on the Device, the challenge is sent (in clear form w. r. t. the homomorphic encryption system) to the Company server. The server then evaluates Eq. (1) confronting the cleartext challenge (i.e., clear S's) to the encrypted reference (i.e. encrypted M's). As a result, the server obtains an encrypted value for ℓ which is returned to the Device.

Lastly, the Device, as provided with the secret decryption key, decrypts ℓ, compares the decrypted value to the threshold T, and decides whether or not to grant access. Still, it could be considered desirable to perform the threshold comparison in the encrypted domain on the server itself—although at the cost of more involved homomorphic processing. However, recall that, in the present work, homomorphic encryption is deployed as a countermeasure adressing confidentiality threats from the server and not from the access device. From that (narrow) threat model point of view, there is per se no added value to perform the threshold comparison on the server (recall also the discussion at the end of Sect. 3).

So, in essence, we now need a cryptosystem homomorphic-enough so as to allow the evaluation of Eq. (1) in the encrypted domain. In the above setting (i.e. where the challenge is in clear form), we only need to perform additions in the encrypted domain so only a simpler additive homomorphic cryptosystem is per se needed. The choice for not encrypting the challenge (with respect to the homomorphic encryption layer) is motivated by the fact that it appears not to provide any useful information as illustrated on Figure 2 or, at least, no information that would allow to reconstruct the reference[5]. In practice, the challenge would still of course be transmitted via a secure channel—using classical non-homomorphic encryption—such that its confidentiality (and integrity) would be ensured before decryption by the server.

Still, if required, challenge encryption would add some complexity, but not necessarily prohibitively so. The naive approach would be to send 118 80×80 matrices containing encryptions of 1 (for coordinates in the challenge) or encryptions of 0 (otherwise). Then equation (1) would become a degree-2 polynomial with respect to evaluation in the encrypted domain. However the main issue would be in producing the challenge (which would require as many encryptions as for the reference and, as will be explained in Sect. 5.1, this is challenging for the kind of lightweight devices presumably used for access) as well as transmitting it towards the server. Overall, this would most likely rule out any real-time or close-to-real-time authentication process. A slightly more subtle approach could be to add dummy coordinates to the challenge along with an encrypted value (0 or 1) depending on whether or not the coordinate indeed belongs to the challenge. This latter approach appears more realistic than the former, although it still raises a number of questions, most notably w. r. t. the dummy/non dummy coordinate ratio. This topics deserves more investigations which are out of the scope of the present work.

5. Of course, this assumption which is rather informally stated here, would deserve a more thorough (far from straightforward) investigation but this is beyond the scope of the present paper.

4.3. The Paillier cryptosystem

Following the algorithm description given in the previous section, it appears that an additively homomorphic system is sufficient in order to execute it. Thus, a first candidate would be the well-known (and well-respected) Paillier cryptosystem [19] for which we recall a few caracteristics[6]. Let p and q denote two large primes and $n = pq$. Then, the cleartext domain of the Paillier system is $\mathbb{Z}_n$ and the ciphertext domain is $\mathbb{Z}_{n^2}$. Additionally, let $\lambda = \text{lcm}(p-1, q-1)$ and $g < n^2$ be randomly chosen such that

$$\gcd(L(g^\lambda \mod n^2), n) = 1,$$

with $L(u) = \frac{u-1}{n}$.

The public (encryption) key is provided by n and g whereas the private (decryption) key is given by p and q or, equivalently, λ.

Then, encryption is done by computing

$$c = \text{enc}(m) = g^m r^n \mod n^2, \tag{2}$$

where $m < n$ is the message and r is uniformly chosen in $\mathbb{Z}_n$. Letting $D = L(g^\lambda \mod n^2)$ and D^{-1} denoting its multiplicative inverse in $\mathbb{Z}_n$, decryption is then performed by evaluating

$$m = \text{dec}(c) = L(c^\lambda \mod n^2) \times D^{-1} \mod n.$$

More importantly for the present purpose, this cryptosystem has the following homomorphic properties:

1) $\text{dec}(\text{enc}(m_1)\text{enc}(m_2) \mod n^2) = m_1 + m_2 \mod n$ (addition of two encrypted messages).
2) $\text{dec}(\text{enc}(m)g^k \mod n^2) = m + k \mod n$, for all $k \in \mathbb{Z}_n$ (addition of an encrypted message to a clear integer).
3) $\text{dec}(\text{enc}(m)^k \mod n^2) = km \mod n$, for all $k \in \mathbb{Z}_n$ (multiplication of an encrypted message by a clear integer).

With respect to the latter homomorphic operator, multiplication by a "negative" integer k works by taking the additive inverse modulo n of $-k$ before applying the operator. However, this generally induces a significant slowdown of the operator as the running time of a modular exponentiation generally increases linearly in the number of bits of the exponent (e.g., when using algorithms such as the repeated

6. Of course, other additive cryptosystems, from additive variants of El Gamal to depth-0-tuned sFHEs could have been used as well. With that respect, we refer the reader to the discussion in the conclusion of the paper.

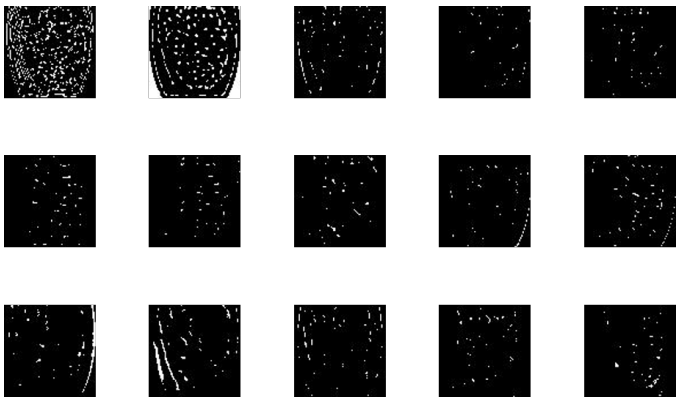

Figure 2. The first 15 S's of a typical challenge.

square-and-multiply). As a consequence, a sometimes critical optimization when using the Paillier system e.g. for computing a dot-product

$$\sum_i \alpha_i x_i,$$

where the α's are (presumably small) clear coefficients and the x's are encrypted data, consists in evaluating it as,

$$\sum_{i:\alpha_i \geq 0} \alpha_i x_i + (-1) \sum_{i:\alpha_i < 0} (-\alpha_i) x_i.$$

In terms of concrete dimensioning, using a 2048 bit modulus provides fairly decent medium-term security e.g., the French National Information Systems Security Agency (ANSSI) [2] recommends this as the minimum modulus length for usage not beyond 2030. Since, as discussed in the sequel, performances are the main issue that needs to be tackled, achieving decent performances with this concrete dimensioning will be our target.

Note that the Paillier cryptosystem can also be "boosted" in order to evaluate quadratic functions using a clever (yet simple) trick from [5]. Although this technique is far from free from a performance viewpoint, it allows to evaluates popular quadratic distances such as the Hamming, Euclidean or Mahalanobis distances [14] in the private versus public vectors setting. Also note that these kind of distances can also be evaluated using *only* an additive system at the cost of a quadratic increase in communication overhead as both distance evaluations turn out to be linear in xx^T (the tensor product of the private vector with itself). These latter remarks are important as many others algorithms used in biometric authentication (not limited to face-based) involve evaluating the aforementioned mathematical distances.

5. Experimental results

5.1. Encryption

According to the architecture defined in Sect. 3, encryption is performed by Dilbert on his home PC after downloading Company's public key from Dogbert, presumably as part of an enrollment application he (Dilbert) has to install on his machine. This application then locally performs face image acquisition, reference generation and encryption of the reference data (recall Sect. 4.1). This unfortunately is easier said than done due to the relatively large volume of data to be encrypted for one reference which amounts to around 750 000 values (i.e., $118 \times 80 \times 80$). Indeed, encrypting that many values using even a native C/C++ GMP-based[7] implementation of equation (2) on an average laptop takes more than 30 mins (with a 2048 bits modulus), a duration which appeared to us prohibitive for the present use case. Of course, as each of the reference values are encrypted separately, that task is embarassingly parallel and its processing time can almost linearly scale down with the number of processors thrown at it. However, it also appeared unreasonable to expect Dilbert to have an high-end multicore server at home so as to decrease encryption duration. So we have to be a bit more clever...

Looking back at equation (2), it appears that the most costly term to evaluate is

$$r^n \mod n^2. \tag{3}$$

Indeed, modular exponentiation is usually performed using algorithms such as the repeated-square-and-multiply algorithm which running time depends on the base-2 logarithm of the exponent. As n is a fairly large number, it can be expected that evaluating (3) takes much more time than evaluating $g^m \mod n^2$ since m, the message, is very small compared to n (each value to encrypt is a scalar in $[0, 1]$ with is premultiplied by 10000 to switch to fixed-point precision).

In order to lighten Dilbert's computing burden, the idea would then be to split the encryption function between him and Dogbert but without putting the security properties of the architecture in jeopardy (Dogbert is trusted with the keys but *not* with the reference data and conversely for Company). The good news with that respect is that (3) does not depend on the data to be encrypted. Hence, the following approach which is consistent with our security objectives:

7. `gmplib.org`.

1) Dogbert uniformly picks around 750 kilos-values uniformly in $\mathbb{Z}_n$ and computes the encryption helper values $h_i = r_i^n \mod n^2$ (offline).
2) When encrypting, Dilbert gets the (many) helper values along with the public key and encrypts its data by evaluating $c_i = h_i g^{m_i} \mod n^2$ (online).
3) Dilbert then transfers the c_i's to Company (online).

In terms of practicality, we now are in a seemingly better shape. The offline step above still takes around 30 mins on an average laptop (but it is now done offline by Dogbert[8]). Furthermore, for Dogbert, which can be assumed to have higher-end servers, it is embarassingly parallel. Thus, using a native GMP-based OpenMP implementation, we have been able to down this offline step to just 99 secs on a 128 cores SMP machine (Bullx S6130) with fairly minimal code refactoring (mostly in order to avoid serialization on the program I/Os). The overall h_i's represent a volume of about 400 Mb (it is the same size as the encrypted reference so we "just" double the transfer requirement for Dilbert), a volume which is by no means prohibitive to transfer by today's home bandwidth standards (such transfers took around 2 mins during the experiments we performed in a completly unremarkable setting). Once this is done, we have been able to perfom the encryption itself, using the h_i's, in only 21 secs on the same average laptop. So such a computational burden becomes acceptable for the kind of low- or average-end devices which Dilbert can be expected to possess at home. Lastly, Dilbert completes the enrollment by transfering the encrypted reference (i.e. around 400 Mb, again) towards Company.

Overall, for the online phase, enrollment thus takes less than 5 mins *without* specific computing power and bandwidth requirements on Dilbert's side and for an acceptably strong security level.

5.2. Homomorphic evaluation

Once Dilbert's (homomorphically) encrypted reference is stored on the Company server, the overall architecture can provide its authentication service by means of homomorphic execution of the algorithm of Sect. 4.1. It turns out that this evaluation step was less of a performance bottleneck (compared to the encryption step discussed in the previous section). Furthermore, it can be assumed that the Company server on which the evaluation is performed is a relatively high-end machine.

As a consequence, we implemented the evaluation algorithm natively, using both GMP as well as OpenMP directives for parallelization. The evaluation program has been architectured so as to avoid inter-task synchronization as well as serialization on the I/Os (the program has to "crunch" around 400 Mb of data in order to produce one encrypted value). The program assumes that each distance map $M^{(k)}$ is stored in a separate file (which is just what the encryption step outputs in order to also avoid serialization on its I/O's in case Dilbert eventually has a multi-core machine). Then each distance map, say $M^{(k)}$ is processed independanlty in parallel (thanks to an OpenMP `parallel for` pragma) using the following algorithm:

1) Organize the list of coordinates in $S^{(k)}$ in a logarithmic-time searchable data structure (e.g., an STL `set<pair<int,int> >`).
2) Initialize a(n encrypted) running sum, $\ell^{(k)}$, to (an encryption of) zero.
3) For $(i,j) \in \{0,79\} \times \{0,79\}$ do, load $M_{ij}^{(k)}$ and if $(i,j) \in S_{ij}^{(k)}$ then $\ell^{(k)} := \ell^{(k)} \Diamond M_{ij}^{(k)}$ (the latter operation being an homomorphic addition).

Then the (encrypted) $\ell^{(k)}$ are serially (homomorphically) summed in order to get the final (encrypted) result ℓ (recall Sect. 4.1).

As such, the program has no inter-task synchronization (except at the end) and scales up to 118 threads which is in line with the highest-end SMP machines presently on the market (we were able to test our programs on a Bullx S6130 128 cores server as well as on a HP Integrity Superdome X 120 cores server which are both representative of these kind of machines).

On an average 8 cores workstation, the homomorphic calculation of ℓ takes arounds 2 secs, and this duration drops significantly below 1 sec on higher-end machines (we consider that 1 sec is the threshold below which the solution becomes practical in terms of latency). This performance scaling is obtained using the same binary, by letting the OpenMP runtime automatically and dynamically adapts the execution to the platform at hand (i.e. the program was compiled using the `-fopenmp` directive offered by recent `gcc/g++` versions).

5.3. Decryption

Once homomorphically computed, the encrypted result ℓ is sent towards the Device at the origin of the challenge for decryption (recall that despite of the fact that the Device is on the Company premises, it is in Dogbert's domain and it is assumed to have a hardware-protected knowledge of the secret decryption key), comparison to a threshold and, if successfull, action. As there is only one value to decrypt, decryption is by no means a performance bottleneck and we could even afford to keep its implementation in the Java realm for our prototype.

Table 1 summarizes the overall performances obtained. As a further prototype work, we also made the effort to fully integrate the binaries for the authentication step into a Web Service queried from an Android tablet and observed RTDs of around 3 secs.

6. Conclusion

In terms of lessons learnt and results, we claim that this work contributes to demonstrate that practical homomor-

8. Which (recall) is assumed not to collude with the Company.

Step	When done	Where done	Duration
Precalculation	Offline	Dogbert	$\approx$30 mins
Encryption	Enrollment	Dilbert	21 secs
Evaluation	Authentication	Company	$<$2 secs
Decryption	Authentication	Device	ε

TABLE 1. PERFORMANCE SUMMARY

phic encryption performances are achievable in meaningful settings. However, we also consider that it can, at present, only be done at the (non negligible) cost of appropriate system architecturing (which involves finding ways to exploit computing power where there is some and doing as much as possible offline calculations) as well as at the cost of aggressive code optimization and parallelization.

Lastly, the reader may found it disappointing that we did not use the more advanced fully homomorphic encryption techniques in the present work. As pointed in the introduction, this is primarily due to the spirit of the exercise we wished to conduct which was (as already stated) "a priori take an arbitrary interesting function and make it practically run over encrypted data in a meaningful architecture". This meant that we did not know in advance which cryptosystem would be powerful-enough to be able to evaluate the selected algorithm and the fact that an additive system would do the trick did not seem to us to invalidate the exercise. Two conclusions may be drawn from this. The first one (which the present work is certainly not the first one to draw), is that, despite of the recent developments and investments in fully homomorphic encryption (and the encrypted-domain Turing-completeness that comes with it), many interesting applications can still be handled using simpler mono-operation homomorphic encryption schemes which practical relevance should then not be underestimated. The second conclusion, which is a matter of perspective to us, is that the more recent RLWE-based Somewhat Fully Homomorphic Encryption (SFHE) schemes (e.g. BGV [4] and Fan-Vercauteren [11]), when optimized for evaluating linear or very low-degree algorithms, may well turn out to be competitive with e.g. the Paillier cryptosystem (although by imposing different constraints on the architecture). At least, preliminary experiments we have performed in other contexts hint in that direction. Also, using SFHE would also unleash the ability to use recently proposed "transciphering" techniques [6], [16] in order to significantly reduce the bandwidth and storage requirement of the solution (yet at the cost of additional offline homomorphic calculations). So we are planning to investigate these questions as a sequel to the present work.

Acknowledgments

This work was done as part of IRT SystemX projects EIC (Environment for Cybersecurity Interoperability and Integration) and, for the third author, SCE (Smart City Energy Analytics).

References

[1] ANSSI, *Sécurité des technologies sans-contact pour le contrôle des accès physiques*, 2012.

[2] ANSSI, *Référenciel général de sécurité 2.0*, annexe B1, 2014.

[3] A. Bouti and J. Keller, *Towards practical homomorphic encryption in cloud computing*, IEEE NCCA'15, pp. 67-74, 2015.

[4] Z. Brakerski, G. Gentry and V. Vaikuntanathan, *(Leveled) fully homomorphic encryption without bootstrapping*, ITCS'12, pp 309-325, 2012.

[5] D. Catalano and D. Fiore, *Using linearly-homomorphic encryption to evaluate degree-2 functions on encrypted data*, ACM CCS'15, pp. 1518-1529, 2015.

[6] A. Canteaut, S. Carpov, C. Fontaine, T. Lepoint, M. Naya-Plasencia, P. Paillier and R. Sirdey, *Stream ciphers: a practical solution for efficient homomorphic-ciphertext compression*, FSE'16, LNCS 9783, pp. 1-21, 2016.

[7] S. Carpov, P. Dubrulle and R. Sirdey, *Armadillo: A Compilation Chain for Privacy Preserving Applications*, ACM SCC@ASIACCS'15, pp. 13-19, 2015.

[8] S. Carpov, T.-H. Nguyen, R. Sirdey, G. Constantino and F. Martinelli, *Practical privacy-preserving medical diagnosis using homomorphic encryption*, IEEE CLOUD'16, 2016 (to appear).

[9] J. Cheon, H. Chung, M. Kim and K.-W. Lee, *Ghostshell: Secure Biometric Authentication using Integrity-based Homomorphic Evaluations*, Cryptology ePrint Archive Report 2016/484, 2016.

[10] Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, I. Lagendijk and T. Toft, *Privacy-preserving face recognition*, PETS'09, LNCS 5672, pp. 235-253, 2009

[11] J. Fan and F. Vercauteren, *Somewhat practical fully homomorphic encryption*, Cryptology ePrint Archive Report 2012/144, 2012.

[12] C. Gentry, *Fully homomorphic encryption using ideal lattices*, STOC'09, pp. 169-178, 2009.

[13] S. Halevi and V. Shoup, *Algorithms in HELIB*, CRYPTO'14, LNCS 8616, pp. 554-571, 2014.

[14] A. K. Jain, P. Flynn, A. A. Ross, *Handbook of biometrics*, Springer, 2007.

[15] Y. Luo, S.-C. Cheung and S. Ye, *Anonymous biometric access control based on homomorphic encryption*, IEEE ICME'09, pp. 1046-1049, 2009.

[16] P. Méaux, A. Journault, F.-X. Standaert and C. Carlet, *Towards stream ciphers for efficient FHE with low-noise ciphertexts*, EUROCRYPT'16, LNCS 9665, pp. 311-343, 2016.

[17] X. Tan and B. Triggs, *Enhanced local texture feature sets for face recognition under difficult lighting conditions*, AMFG 2007, LNCS 4778, pp. 168-182, 2007.

[18] M. Osadchy, B. Pinkas, A. Jarrous and B. Moskovich, *SCiFI - A System for Secure Face Identification*, IEEE Symposium on Security and Privacy, pp. 239-254, 2010.

[19] P. Paillier, *Public-key cryptosystems based on composite degree residuosity classes*, EUROCRYPT 1999, LNCS 1592, pp. 223-238, 1999.

[20] A.-R. Sadeghi, T. Schneider and I. Wehrenberg, *Efficient privacy-preserving face recognition*, ICISC'09, LNCS 5984, pp. 229-244, 2010.

[21] J. R. Troncoso-Pastoriza and F. Pérez-González, *Fully homomorphic faces*, IEEE ICIP'12, pp. 2657-2660, 2012.

[22] C. Xiang, C. Tang, Y. Cai and Q. Xu *Privacy-preserving face recognition with outsourced computation*, Soft Computing, 2015 (in press).