A Framework for Analyzing Template Security and Privacy in Biometric Authentication Systems

Koen Simoens, Julien Bringer, Hervé Chabanne, and Stefaan Seys

***Abstract*—In this paper we analyze the vulnerabilities of biometric authentication protocols with respect to user and data privacy. The goal of an adversary in such context is not to bypass the authentication but to learn information either on biometric data or on users that are in the system. We elaborate our analysis on a general system model involving four logical entities (sensor, server, database and matcher), and we focus on internal adversaries to encompass the situation where one or a combination of these entities would be malicious. Our goal is to emphasize that when going beyond the usual honest-but-curious assumption much more complex attacks can affect the privacy of data and users.**

On the one hand, we introduce a new comprehensive framework that encompasses the various schemes we want to look at. It presents a system model in which each internal entity or combination of entities is a potential attacker. Different attack goals are considered and resulting requirements on data flows are discussed. On the other hand, we develop different generic attacks. We follow a blackbox approach in which we consider components that perform operations on biometric data but where only the input/output behavior is analyzed. These attack strategies are exhibited on recent schemes such as the distributed protocol of Bringer et al. (ACISP 2007), which is based on the Goldwasser-Micali cryptosystem, the related protocol of Barbosa et al. (ACISP 2008), which uses the Paillier cryptosystem, and the scheme of Stoianov (SPIE 2010), that features the Blum-Goldwasser cryptosystem. All these schemes have been developed in the honest-but-curious adversary model and show potential weaknesses when considered in our malicious insider attack model.

***Index Terms*—Biometrics, template protection, authentication, protocols, blackbox security model, malicious adversaries**

I. Introduction

Although biometric template protection is a relatively young discipline, already over a decade of research has brought many proposals. The main objective of template protection methods – and the main difficulty – is to prevent an attacker to compromise privacy of users or biometric data and not necessarily to thwart bypassing of the biometric authentication itself. These methods can be separated in three levels. The first one is to have biometric data coming in a self-protected form. Many algorithms have been proposed: quantization schemes [1], [2] for continuous biometrics; fuzzy extractors [3] and other fuzzy schemes [4]–[6] for discrete biometrics; and cancellable biometrics [7]–[9]. The security of such template-level protection has been intensively analyzed, e.g., in [10]–[13]. On a second level one can use hardware to obtain secure systems, e.g., [14], [15]. Finally, at a third level biometric authentication can be achieved from protocols that rely on advanced cryptographic techniques such as Secure Multiparty Computation, homomorphic encryption or Private Information Retrieval protocols [16, Ch. 9] [17]–[24]. The focus of our work is on this third level.

Our contributions are the following. We extend the blackbox framework initiated in [25] with the distributed system model of [19] in a way that it can handle different existing proposals for biometric authentication. We show how this blackbox approach can lead to attacks against these proposals, that have generally been conceived to only resist non-colluding honest-but-curious entities. We describe in detail our analysis of three existing protocols [19], [20], [22]. In the framework we propose, we develop generic attack strategies in the malicious adversary model. We list all the possible existing attacks points and the different internal entities that can lead the attacks, and we reveal the potential consequences. Some of the attacks that are presented in this paper can easily be solved or prevented. It should be noted, however, that the objective of this work is to demonstrate that existing solutions suffer from certain weaknesses. Moreover, the generic attacks that are defined in this paper can be used by developers and reviewers as a first evaluation for new protocol proposals.

As our aim is privacy leakage analysis, it justifies our focus on internal adversaries: they are stronger than external attackers when challenging the privacy properties of a system. Moreover, our framework models the internal components of a biometric system into four logical entities, namely the sensor, the authentication server, the database and the matcher. This is an important aspect as a system without any separation between these entities would not be able to ensure the highest privacy properties against internal adversaries such as malicious administrators. However the division in four entities cannot be considered as the sole warranty: so we study scenarios where several entities are malicious or collude; and we underline that some entities are more powerful than other in such situations.

Note finally that this representation of a biometric system architecture is not superficial: it is easy to imagine an application in which the authentication server is a service provider that relies on a third party to provide a biometric reference database (for instance government owned) for authentication or identification. Distributed biometric systems and the sharing of biometric databases is becoming more and more common, see for instance the joint project [26] between the Department of Defence (DoD) and the Department of Homeland Security (DHS) in the USA, the FBI Next Generation Identification (NGI) project [27], or the European Visa Information System (VIS) [28].

The rest of the paper is organized as follows. The framework is developed in Section II, which introduces the system and attack model. Generic attack strategies are proposed and formalized in Section III. These are then applied to existing protocols in Section IV, where detailed attacks are described. Section V concludes the paper with a discussion on the problems that need to be solved to achieve the optimal privacy properties.

II. Framework

In this section we present a framework that forms a basis for the security analysis of biometric authentication protocols. The framework models a generic distributed biometric system and the (internal) adversaries against such system. We define the roles of the different entities that are involved and their potential attack goals. From these

Manuscript received Month xy, 200Z; accepted Month xy, 200Z. First published Month xy, 200z; current version published Month xy, 2002. This work was sponsored in part by the EU project TURBINE, which is funded by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement nb. ICT-2007-216339. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Ajay Kumar.

J. Bringer is with Morpho, Osny 95520, France (e-mail: julien.bringer@morpho.com).

H. Chabanne is senior member of IEEE. He is with Morpho, Osny 95520, France and with Télécom ParisTech, Paris 75013, France (e-mail: herve.chabanne@morpho.com).

S. Seys and K. Simoens are with KU Leuven, 3001 Leuven-Heverlee, Belgium and with IBBT, 9050 Ghent-Ledeberg, Belgium (e-mail: stefaan.seys@esat.kuleuven.be; koen.simoens@esat.kuleuven.be). K. Simoens is student member of IEEE.

Corresponding author: K. Simoens.

Digital Object Identifier 10.1109/TIFS.20XX.XYZXYZ

roles and attack goals we derive the requirements that are imposed on the data that are exchanged between the entities.

Biometric Notation: Two measurements of the same biometric characteristic are never exactly the same. Because of this behavior, a biometric characteristic is modeled as a random variable B, with distribution p_B over some range $\mathcal{B}$. A sample is denoted as b. Two samples or templates are related if they originate from the same characteristic. In practice, we will say that they are related if their mutual distance is less than some threshold. Therefore, a distance function d is defined over $\mathcal{B}$ and for each value in the range of d that is used as the threshold when comparing two samples a false match rate (FMR) and a false non-match rate (FNMR) can be derived.

A. System Model

Our system model follows to a large extent the model defined by Bringer *et al.* [19], which was also used to define new schemes in [20] and [29]. This model is motivated by a separation-of-duties principle: the different roles for data processing or data storage on a server are separated into three distinct entities. Using distributed entities is a baseline to avoid one to control all information and it is a realistic representation of how biometric systems work in practice (cf. [30]).

System Entities: The different entities involved in the system are a user $\mathcal{U}_i$, a sensor $\mathcal{S}$, an authentication server $\mathcal{AS}$, a database $\mathcal{DB}$ and a matcher $\mathcal{M}$. User $\mathcal{U}_i$ wishes to authenticate to a particular service and has, therefore, registered his biometric data b_i during the *enrollment* procedure. In the context of the service the user has been assigned an identifier ID_i, which only has meaning within this context. The biometric reference data b_i are stored by $\mathcal{DB}$, who links the data to identifier i. The mapping from ID_i to i is only known by $\mathcal{AS}$, if relevant. Note that in some applications it is possible that the same user is registered for the same service or in the same database with different samples, b_i and b_j, and different identities, i.e., $ID_i \neq ID_j$ in the service context or $i \neq j$ in the database context. The property of not being able to relate queries under these different identities is the *identity privacy* requirement as defined in [19].

During the *authentication* procedure the sensor $\mathcal{S}$ captures a fresh biometric sample b'_i from user $\mathcal{U}_i$ and forwards the sample to $\mathcal{AS}$. The authentication server $\mathcal{AS}$ manages authorizations and controls access to the service. To make the authorization decision, $\mathcal{AS}$ will rely on the result of the biometric *verification* or *identification* procedure that is carried out by the matcher $\mathcal{M}$. It is assumed that there is no direct link between $\mathcal{M}$ and $\mathcal{DB}$. As such, $\mathcal{AS}$ requests from $\mathcal{DB}$ the reference data that are needed by $\mathcal{M}$ and forwards them to $\mathcal{M}$. It is further assumed that the system accepts only biometric credentials. This means that the user provides his biometric data and possibly his identity, but no user-specific key, password or token. Fig. 1 shows the participating entities.

Functional Requirements: Enrollment often involves offline procedures, like identity checks, and is typically carried out under supervision of a security officer. During verification (or identification) the system typically runs in an automated and non-supervised mode. Therefore, we assume that the system entities are trusted during enrolment, hence users are enrolled properly and only authentication procedures are analyzed in our framework. A distinction has to be made between verification and identification. Verification introduces a *selection step*, which implies that $\mathcal{DB}$ returns only one of its references, namely the b_i that corresponds to the identifier i that is used in the context of the database. The entity that does the mapping between ID_i and i, when applicable, is generally $\mathcal{AS}$. In identification mode, $\mathcal{DB}$ returns the entire set of references, in some protected form, to $\mathcal{AS}$. The database can then be combined with b'_i and forwarded to $\mathcal{M}$. The matcher $\mathcal{M}$ has to verify that b'_i matches

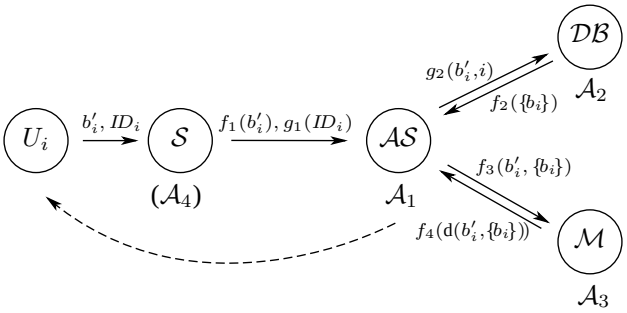

Fig. 1. System model with indication of generic information flows and attack points $\mathcal{A}_i$. User $\mathcal{U}_i$'s biometric is sampled by sensor $\mathcal{S}$. The sample b'_i and $\mathcal{U}_i$'s identity are forwarded to the authentication server $\mathcal{AS}$, who requests the corresponding reference b_i from database $\mathcal{DB}$. $\mathcal{AS}$ combines the sample and the reference and forwards the result to matcher $\mathcal{M}$, who performs the actual comparison and returns the result to $\mathcal{AS}$. The solid arrows represent the messages exchanged between the system entities. The dashed arrow represents the implicit feedback on the authentication result to the user $\mathcal{U}_i$, i.e., access to the requested service is granted if the sample matches the reference.

with one or a limited number of b_i in the received set of references or that one of the matching references has index i.

We define the minimal logical functionality to be provided by our system entities in terms of generic information flows, which are included in Fig. 1. In this functional model, we represent the result of the biometric comparison as a function of the distance $d(b'_i, b_i)$. This is a generic representation of the actual comparison method: $\mathcal{M}$ can evaluate simple distances but also run more complex comparisons and will output either similarity measures or decisions that are based on some threshold t. The information flows are as follows.

User $\mathcal{U}_i$ presents a biometric characteristic B_i that will be sampled by the sensor $\mathcal{S}$ to produce a sample b'_i. When operating in verification mode $\mathcal{U}_i$ will claim an identity ID_i :

$$\mathcal{U}_i \xrightarrow{b'_i \leftarrow B_i} \mathcal{S} \qquad \text{or} \qquad \mathcal{U}_i \xrightarrow{b'_i \leftarrow B_i\,,\, ID_i} \mathcal{S}\,. \tag{1}$$

The sensor $\mathcal{S}$ forwards b'_i and ID_i in some form to $\mathcal{AS}$:

$$\mathcal{S} \xrightarrow{f_1(b'_i)} \mathcal{AS} \qquad \text{or} \qquad \mathcal{S} \xrightarrow{f_1(b'_i)\,,\, g_1(ID_i)} \mathcal{AS}\,. \tag{2}$$

In general $g_1(ID_i) = ID_i$ but it can also be a mapping to an encrypted value to hide ID_i from $\mathcal{AS}$. If applicable, $\mathcal{AS}$ resolves the mapping $g_1(ID_i)$ to identifier i and requests reference data for one or more users from $\mathcal{DB}$ by sending at least one request $g_2(b'_i, i)$:

$$\mathcal{AS} \xrightarrow{g_2(b'_i, i)} \mathcal{DB}\,. \tag{3}$$

Note that the function g_2 does not necessarily use all the information in its arguments, e.g., the fresh sample b'_i may be ignored.

Database $\mathcal{DB}$ provides $\mathcal{AS}$ with reference data for one or more users in some form. It is possible that $\mathcal{DB}$ returns the entire database, e.g., in case of identification:

$$\mathcal{AS} \xleftarrow{f_2(\{b_i\})} \mathcal{DB}\,. \tag{4}$$

The authentication server $\mathcal{AS}$ forwards the fresh sample b'_i and the reference data b_i in some combined form to $\mathcal{M}$:

$$\mathcal{AS} \xrightarrow{f_3(b'_i, \{b_i\})} \mathcal{M}\,. \tag{5}$$

Note that $\mathcal{AS}$ has only $f_1(b'_i)$ and $f_2(b_i)$ at his disposal to compute $f_3(b'_i, \{b_i\})$.

The matcher $\mathcal{M}$ performs a biometric comparison procedure on the received b'_i and $\{b_i\}$ and returns the result to $\mathcal{AS}$. The result may contain decisions or scores or different identities but should at

least be based on one distance calculation between the fresh sample b_i' and a reference b_i :

$$\mathcal{AS} \xleftarrow{f_4(\mathrm{d}(b_i', \{b_i\}))} \mathcal{M}\,. \tag{6}$$

Different data are stored by the different entities. The database stores references $\{b_i\}$. The authentication service stores the information needed to map $g_1(ID_i)$ to i, if applicable. The matchers can store non-biometric verification data, e.g., hashes of keys extracted from biometrics, or decryption keys that are use to recover the result of combining sample and reference. Also, the sensor can store key material to encrypt the fresh sample.

B. Adversary Model

Attacker Classification: Based on the physical entry point of an attack a distinction is made between two types of attackers: *internal* attackers are corrupted components in the system and *external* attackers are entities that only have access to a communication channel. We will consider here only the issue of an insider attacker. As a baseline, we make the following assumption.

Assumption 1: The protocol ensures the security of the scheme against any external attacker.

As this can be reached by classical secure channel techniques, by an external security layer independent of the core protocol specification, we study further only the internal layer.

A second distinction is made based on an attacker's capabilities. Passive or honest-but-curious attackers only eavesdrop the communications in which they are involved and can only observe the data that passes through them. They always follow the protocol specifications, never change messages and never generate additional communication. Active or malicious attackers are internal components that can also modify existing or real transactions passing through them and that can generate additional messages. We mainly focus on malicious internal attackers and we formulate the following additional assumption.

Assumption 2: The protocol ensures the security of the scheme against honest-but-curious entities, i.e. internal system components that always follow the protocol specifications but eavesdrop internal communication.

We will explain in Section II-C how this has a direct impact on the properties of the different functionalities in our model.

Finally, we put aside the threats on the user or client side, by concentrating the analysis on the remote server's side. The information leakage for the user and the client is generally only the authentication or identification result. They can, however, try to gain knowledge on the reference data b_i by running queries with different b_i', e.g., in some kind of hill climbing attack. The difficulty can highly vary depending on the modalities, the threshold and the scenario. A basic line of defense is to limit the number of requests, to ensure the aliveness of the biometric inputs provided by the user and to hide the matching score, if possible. Although it is important to implement such defense mechanisms, the threats are inherent to any biometric authentication or identification system. So we do not take the user or the sensor into account as an attacker in this model and the primary attack points are $\mathcal{AS}$, $\mathcal{DB}$ and $\mathcal{M}$. Nonetheless, there may be inside attackers that also control the biometric inputs to some extent. We model this with a secondary attack point at the sensor.

Assumption 3: The user $\mathcal{U}_i$ or the sensor $\mathcal{S}$ cannot be attackers on their own but they can act as a secondary attack point in combination with a primary attack point at $\mathcal{AS}$, $\mathcal{DB}$ or $\mathcal{M}$. If this is the case an attacker can choose the input sample b_i' through $\mathcal{S}$ and observe whether the authentication request was successful through $\mathcal{U}_i$.

Of course, the baseline assumptions have to be checked before proceeding with a full analysis of the security of a scheme, but as such, they clarify what the big issues are that may remain in state-of-the-art schemes. They also underline what the hardest challenges are when designing a secure biometric authentication protocol. Fig. 1 sums up the different attack points from our attack model.

TABLE I
ATTACK GOAL RELEVANCE (V = relevant ; ? = relevant if designed to hide references from $\mathcal{DB}$; * = relevant if ID_i and i are hidden from $\mathcal{AS}$).

Attack goal	$\mathcal{AS}$	$\mathcal{DB}$	$\mathcal{M}$
Learn b_i	V	?	V
Learn b_i'	V	V	V
Trace $\mathcal{U}_i$ with different identities	V	?	V
Trace $\mathcal{U}_i$ over different queries	V*	V	V

Attack Goals: The security of a scheme is expressed in terms of specific attack goals or adversary objectives. The attack goals stem from the security and privacy issues that are commonly acknowledged in the literature, e.g., in [7] or [31]. In fact, our framework is quite similar to that proposed by Ratha *et al.* [7], but with two major differences. As opposed to [7] the objective of the attacker in our framework is not to get authenticated fraudulently, but to compromise the template security and user privacy. Moreover, as opposed to [7] the communication channels between the different modules are assumed to be secure.

Typical issues are that biometric data can be used to reconstruct artifacts (fake samples) to impersonate other users or that biometric data might reveal sensitive (medical) information. So biometric data, both references and samples, should be hidden. Then, even if the data are hidden, it should not be possible to abuse protected references as identifiers for cross-matching, i.e., linking data from different applications. Hence, users that are registered in multiple applications, thus having multiple identities, should not be traceable over these applications. Finally, privacy-enhanced applications that rely on properly authenticated users, e.g. anonymous ePetitions or online auctions where identities are hidden for all but the winning bid, should not lose their privacy-enhancing properties due to the introduction of biometrics.

These issues lead to the following global attack goals.

- **Learn reference** b_i **.** A scheme provides *biometric reference privacy* if it resistant to an attacker that wants to learn some information about a reference b_i .
- **Learn sample** b_i' **.** We call the security property associated with this attack goal *biometric sample privacy*.
- **Trace users with different identities.** This attack can be achieved when different references from the same user, possibly coming from different applications, can be linked. A system that is resistant to such attack is said to provide *identity privacy* [29].
- **Trace users over different queries.** This attack refers to linking queries, whether anonymized or not, based on i, b_i or b_i'. The property of a system that prevents such attack is called *transaction anonymity* [29]. An attacker that is able to learn b_i' can automatically trace users based on the learned sample.

With these attack goals we aim to analyze how the optimal privacy properties with respect to the privacy-by-design principles [32] can be achieved. Although this is not always currently required for all biometric applications, it is a challenging issue to be able to use biometrics fully anonymously and many researches are conducted to this aim. Of course, adding more information to be able to decrease the level of privacy on behalf of some more trusted entity, e.g., under a legal warrant, is a feasible option that allows addressing more applications in the same framework.

The formulated attack goals apply to the different internal attackers as shown in TABLE I. Attack goals can be generalized for combinations of inside attackers, e.g., $\mathcal{AS}$ and $\mathcal{M}$, and they are

relevant for the combination if they are relevant for each attacker individually. As a counterexample, learning b_i is not always relevant for the combination $\mathcal{AS}$-$\mathcal{DB}$. In some schemes it is assumed that $\mathcal{DB}$ stores references in the clear so the attack "learn b_i" becomes trivial. It is important, however, that such schemes explicitly mention the assumption that $\mathcal{DB}$ is fully trusted. It will become clear in the further sections that the main focus of our work is on $\mathcal{AS}$ who is a powerful attacker. This way of thinking is rather new and many protocols are not designed to be resistant to such attacker.

It should be noted that there are many more security and privacy issues relevant for biometric systems in general. Infrastructure, deployment context, application setting, human factors, etc. determine the different risks that are to be taken into account. We refer to the work of Jain et al. [33] for a comprehensive treatment and categorization of general biometric system vulnerabilities.

C. Requirements on Data Flows

Coming back to the functionalities in our system model (cf. Section II-A), we use the attack goals defined in TABLE I to impose requirements on the data that are being exchanged.

- $\mathcal{AS}$ should not be able to learn b'_i hence f_1 is at least one-way, meaning that b'_i should be unrecoverable from $f_1(b'_i)$ with overwhelming probability. To prevent tracing $\mathcal{U}_i$ over different queries it could also be required that f_1 is semantically secure.[1] Semantic security is a security notion that might be too strong but it ensures a minimum leakage of information.
- $\mathcal{AS}$ should not learn b_i hence f_2 is at least one-way. To prevent tracing users with different identities it may be required that f_2 is also semantically secure.
- If applicable, $\mathcal{AS}$ should not be able to trace $\mathcal{U}_i$ by linking queries on ID_i or i, and thus g_1 should be semantically secure.
- If applicable, $\mathcal{DB}$ may not learn b_i, hence b_i would be stored in a protected form using some semantically secure function.
- $\mathcal{DB}$ may not learn b'_i, hence g_2 is one-way on its first input. It should also be semantically secure to prevent tracing $\mathcal{U}_i$.
- $\mathcal{DB}$ may not be able to link the queries at all, hence g_2 should also be semantically secure on its second input.
- $\mathcal{M}$ may not learn the individual b_i or b'_i and must not be able to link references or queries from the same $\mathcal{U}_i$, hence f_3 should be semantically secure on tuples $\langle b'_i, b_j \rangle$

Because we demand that $\mathcal{M}$ returns a result to $\mathcal{AS}$ that is a function (f_4) of $d(b'_i, b_i)$ some operations must be malleable. Malleability refers to the property of some cryptosystems that an attacker can modify a ciphertext into another valid ciphertext that is the encryption of some function of the original message, without the attacker knowing this message. Depending on when the combination of b_i and b'_i is realized, either g_2, f_2 or f_3 would be malleable. In the following section, we will show the impact of this fundamental limitation and how it can be exploited to attack existing protocols.

III. Generic Attack Strategies

The goal of this section is to explore the different attack scenarios in our framework that can be used for analyzing actual protocol specifications. First, a blackbox attack model is explained. Then some generic attacks with $\mathcal{AS}$ as adversary are derived which will be demonstrated in the examples in Section IV. Finally,we briefly discuss the attack goals from Section II-B and potential attacks for every possible adversary combination, including both the case without $\mathcal{S}$ being involved and the case with $\mathcal{S}$ involved. Recall from Section II-B that $\mathcal{S}$ is not considered to be an autonomous attacker. However, an attacker that has control over $\mathcal{S}$ is very powerful and, therefore, this is a necessary additional dimension that should be analyzed.

A. Blackbox Attack Model

The different attacks that can be carried out by the attackers are modeled as *blackbox attacks*, following recent results from [25]. This allows us to clearly specify the focus of the attack. Our blackbox-attack model consists of two logical entities:

1) The attacker, i.e., one or more system entities that are fully under control of the attacker: internal data are known, messages can be modified and additional transactions can be generated.
2) The target or the blackbox, i.e., the combination of all other system entities. The attack is focused on the data that are protected by the system components within the blackbox.

The target is modeled as a blackbox because the attacker can only observe the input-output behavior of the box. This adequately reflects remote protocols where only the communication can be seen by the attacker. No details are known about the internal state of the remote components. During the attack, the attacker will "tweak" inputs to the blackbox. However, all communication must comply with the protocol specification. Any messages that are malformed or that are sent in the wrong order are rejected by the blackbox.

As explained in Section II only malicious internal attackers are considered, i.e., $\mathcal{AS}$, $\mathcal{DB}$, $\mathcal{M}$ and combinations of these entities. User $\mathcal{U}_i$ and the sensor $\mathcal{S}$ have been excluded as individual attackers. However, it should be noted that there are cases in which the attacker cannot generate additional transactions because he has to follow the protocol specifications. E.g., if $\mathcal{DB}$ is attacking he has to wait until a request is received from $\mathcal{AS}$. When analyzing protocols it should be assumed that this will occur with a reasonable frequency. If relevant, attack complexities can be expressed in function of this frequency. Similarly, if the attacker is $\mathcal{AS}$, he receives inputs from $\mathcal{S}$ and communicates with $\mathcal{DB}$ and $\mathcal{M}$. In this case we exclude $\mathcal{U}_i$ and $\mathcal{S}$ from the blackbox. It should be assumed, though, that a number of inputs from $\mathcal{S}$ is available to $\mathcal{AS}$. This does not necessarily imply that $\mathcal{S}$ is under control of $\mathcal{AS}$. The analysis of the attack can take into account the amount of data that is available.

B. Generic Attacks for $\mathcal{AS}$

The following three generic attacks are specific for the authentication server and will be demonstrated in Section IV.

Decomposed Reference Attack: Let's assume that only one reference b_i is returned by $\mathcal{DB}$. The goal of this attack is to learn b_i. Biometric samples or references are often represented as a "string", i.e., a concatenation (let $\|$ denote concatenation) of (binary) symbols. Let's assume that $f_2(b_i)$ is the concatenation of a subfunction $\hat{f}_2$ that is applied on each of the n components $b'_{i,j}$ of b'_i individually. If $\mathcal{AS}$ has to combine $f_2(b_i)$ and $f_1(b'_i)$ without knowing either the sample or the reference, it is likely that f_1 and f_3 will also be the concatenation of component-wise applied subfunctions, i.e., $f_3(b_i, b'_i) = \hat{f}_3(b_{i,1}, b'_{i,1}) \| \ldots \| \hat{f}_3(b_{i,n}, b'_{i,n})$. Note that in our model $\mathcal{AS}$ can generate the value $\hat{f}_3(b_{i,j}, b'_{i,j})$ but this value should not reveal to $\mathcal{AS}$ whether the inputs are the same or not. This decomposition of references leads to the following attack.

Suppose that $\mathcal{AS}$ is able to generate a value that is valid output of $\hat{f}_3$ when the two component inputs $b_{i,j}$ and $b'_{i,j}$ are the same and similarly when they are not the same, e.g., the output is the encryption of one or zero. If $\mathcal{AS}$ can also compute f_1, then $\mathcal{AS}$

[1] A cryptosystem is semantically secure if it is infeasible for a computationally bounded adversary to learn from a given ciphertext any useful information about the underlying plaintext. It implies that an adversary has no significant advantage over random guessing when trying to distinguish two encryptions of the same message from the encryptions of two different messages.

can fully reconstruct b_i. To do so $\mathcal{AS}$ choose the first component of b_i' at random, combines it with the first component of b_i and sends the result to $\mathcal{M}$. The other components that are sent to $\mathcal{M}$ are such that t of those are an output of $\hat{f}_3$ that reflects different inputs and the $n-t-1$ remaining components are outputs that reflect equal inputs. Note that t is the comparison threshold. If the guess of $\mathcal{AS}$ for the first component is correct then $\mathcal{M}$ will return a positive match. Otherwise the guess is wrong and $\mathcal{AS}$ can try again. This process can be repeated until all components of b_i are recovered. For binary samples, this requires n queries to $\mathcal{M}$ and 1 query to $\mathcal{DB}$.

A similar attack can be executed if the biometric data are represented as real-valued or integer-valued feature vectors. However, additional queries are required to fully recover the reference b_i.

Center Search Attack Using $\mathcal{S}$: In this attack, $\mathcal{S}$ is also compromised and under the control of the attacker. The attack goal is to learn the full reference b_i from a close sample. The input sample is obviously always known to $\mathcal{AS}$ and $\mathcal{S}$. Thus at some point in time $\mathcal{U}_i$ will present a sample b_i' that matches reference b_i. This sample will lie at some distance from the reference. In the case where biometrics are represented as binary strings and the system implements a hamming distance matcher the attacker can recover the exact b_i as follows.

The sensor flips the first bit of b_i' and sends the new sample to $\mathcal{AS}$ who performs the whole authentication procedure. If the authentication succeeds, $\mathcal{S}$ flips the second bit, leaving the first bit also flipped, and sends the sample to $\mathcal{AS}$ who follows the procedure again. This continues until the sample no longer matches b_i. Then the sensor starts again by restoring the first bit of the sample that is no longer accepted and forwards it to $\mathcal{AS}$. If it gets accepted this means that the first bit of the original sample b_i' was the same as the first bit of b_i. If not, then the first bits were different. One by one the bits in b_i' that are different from those in b_i can be corrected.

We call this the center search attack because we start from a sample that lies in a sphere with radius t, the matching threshold, and the reference as center point. The goal of this attack is to move the sample to the center of the sphere. The worst-case complexity of this attack for bitstrings of length n is the greatest of $2*t+n$ and $4t$. The complexity is $2*t+n$ if there are $t-1$ bit-errors in the beginning and one at the end of the string. The first $t-1$ errors get corrected by flipping them and t additional bits need to be flipped to invalidate the sample. Locating the bit-errors requires searching till the end of the string where the last error is. The complexity is $4t$ if there are $t-1$ correct bits followed by t wrong bits. So $2t$ flips are needed before the queries no longer match and then $2t$ positions need to be searched. In practice, $t \leq n/2$ and thus the worst-case complexity is $2t+n$.

An obvious limitation of this attack is that the matcher might easily detect suspicious transactions, e.g., consecutive inputs that have structures deviating from normal input. However, detecting such inputs may not always be feasible if a system is fully automated and an attacker can alternate malicious transactions with valid ones. Moreover, depending on the techniques that are used, the structure in the inputs may be easy to mask.

Unknown Matching Sample Attack: The goal of this attack is to learn b_i from a matching sample that is unknown to the attacker. This attack combines ideas from the previous attacks. The attacker is $\mathcal{AS}$, not including $\mathcal{S}$, and $\mathcal{AS}$ does not know how to compute an output of f_3 that reflects equal (or different) inputs. It is assumed, however, that the attacker can replace the components of b_i' in the value he received from $\mathcal{S}$, i.e., $f_1(b_i')$. This is definitely the case if f_1 is a concatenation of subfunctions and if $\mathcal{AS}$ can compute such subfunction $\hat{f}_1$.

The actual attack proceeds as follow. The attacker $\mathcal{AS}$ waits until a genuine user presents a valid sample. The attack is similar as in the center-search attack, only now $\mathcal{AS}$ will not flip bits but simply replace them with a known value, e.g., one. He will do this until the sample no longer matches. Then $\mathcal{AS}$ already knows that the last bit he replaced was not one and he will restore that bit. Then he continues to substitute the bits one by one, carefully observing whether the sample matches or not and learning all the bits. The first bits that were flipped to invalidate the sample can be learned simply by restoring them.

C. Attack Discussion (no $\mathcal{S}$ involved)

We iterate the attack goals from Section II-B for every possible adversary combination, not involving the sensor $\mathcal{S}$. When attackers are combined they inherit the potential attacks from the individual attackers. For example, if $\mathcal{AS}$ is able to learn b_i then the combination of $\mathcal{AS}$ and $\mathcal{DB}$ will also be able to learn b_i.

1) Attacker = $\mathcal{AS}$:

- Learn b_i: Potential attacks for achieving this goal have been described in Section III-B.
- Learn b_i': The same attacks as for learning b_i can be used.
- Trace U_i with different IDs: This goal could be achieved if $\mathcal{AS}$ can access different databases that have compatible system parameters. This would imply that samples from one system could be used for comparison in another system. However, such issue is unlikely if, e.g., encryption is used and the keys are generated per application.
- Trace U_i over different queries: This attack is trivially achieved if the system operates in verification mode and $\mathcal{AS}$ does the mapping from ID_i to i in the clear. If $\mathcal{AS}$ does not know the mapping but can influence the value i that is sent to $\mathcal{DB}$ in a deterministic way then $\mathcal{AS}$ can trigger an identification mode by repeatedly querying $\mathcal{DB}$ and iterating over all values of i.

2) Attacker = $\mathcal{DB}$: The attacker is the database $\mathcal{DB}$ who communicates with the authentication service $\mathcal{AS}$ only. The attacker cannot achieve any of the attack goals individually because his blackbox gives output, which he cannot influence, before receiving input. If this entity does not collude with other entities then it is simply a passive attacker and by Assumption 2 it cannot mount any attacks.

3) Attacker = $\mathcal{M}$: The matcher alone cannot achieve any of the goals for the same reason as $\mathcal{DB}$. By Assumption 2 it cannot attack because its blackbox provides output before input and $\mathcal{M}$ cannot trigger any additional transactions.

4) Attacker = $\mathcal{AS}$ and $\mathcal{DB}$: Achieving the attack goals depends on how the functionality of $\mathcal{DB}$ is implemented.

- Learn b_i: The attacker can learn the entire database because $\mathcal{DB}$ will return any b_i and $\mathcal{AS}$ can manipulate any transactions.
- Learn b_i': Inherits the attacks of $\mathcal{AS}$.
- Trace U_i with different IDs: The attacker can easily search different databases and operate in identification mode although the protocol could be designed to operate in verification mode.
- Trace U_i over different queries: Adding $\mathcal{DB}$ as an adversary makes it easier to run in identification mode or to learn i, if not already known by $\mathcal{AS}$.

5) Attacker = $\mathcal{AS}$ and $\mathcal{M}$: Depending on how $\mathcal{M}$ implements its functionality this can be a very powerful attacker, e.g., if $\mathcal{M}$ possesses decryption keys for encrypted samples/templates.

- Learn b_i: Because $\mathcal{AS}$ can forward to $\mathcal{M}$ any data received from $\mathcal{DB}$ without necessarily combining it it with a new sample, this is practically trivially achieved.
- Learn b_i': For the same reason as the previous attack goal, this is practically trivially achieved.
- Trace U_i with different IDs: Assisted by $\mathcal{M}$, $\mathcal{AS}$ may be able to more easily compare references from different databases.

- Trace U_i over different queries: This is practically trivially achieved from learning b_i and b'_i.

6) Attacker = $\mathcal{DB}$ and $\mathcal{M}$: In this combination of attackers, $\mathcal{DB}$ will manipulate its output so that it can be of use to the $\mathcal{M}$. The attacker is, however, limited in generating additional queries.

- Learn b_i: This combination of attacker most likely results in a situation that is equivalent to having all references in the clear.
- Learn b'_i: This is not necessarily achievable, because $\mathcal{AS}$ can transform the combination of b_i and b'_i before it is sent to $\mathcal{M}$.
- Trace U_i with different IDs: As a consequence of the ability to learn b_i this is practically trivially achieved.
- Trace U_i over different queries: Most likely this attacker is able to learn for which i the reference was requested and trace $\mathcal{U}_i$.

7) Attacker = $\mathcal{AS}$ and $\mathcal{DB}$ and $\mathcal{M}$: In this particular case, the attacker is a combination of $\mathcal{AS}$, $\mathcal{DB}$ and $\mathcal{M}$, and the attacker has, in principle, no limitations with regard to learning b_i, tracing $\mathcal{U}_i$ over different queries or linking different IDs of $\mathcal{U}_i$. For example, the attacker is in no way limited to perform a search (identification) on the database. The main attack goal that remains is to learn b'_i.

D. Attackers including $\mathcal{S}$

Two of the attack goals are trivially achieved if $\mathcal{S}$ is included. Obviously, the attacker learns the sample b'_i. At the same time this implies that the attacker can trace U_i over different queries by storing and using b'_i as a unique identifier.

1) Attacker = $\mathcal{AS}$ including $\mathcal{S}$:

- Learn b_i: Besides the attacks without $\mathcal{S}$ (cf. supra), $\mathcal{AS}$ can perform a center-search attack from a matching sample that is known through $\mathcal{S}$.
- Trace U_i with different IDs: The attacker can use a matching sample and start looking that up in different databases.

2) Attacker = $\mathcal{DB}$ including $\mathcal{S}$:

- Learn b_i: The same center-search attack can be performed as in the case where $\mathcal{AS}$ and $\mathcal{S}$ are the attacker. However, a matching sample is required.
- Trace U_i with different IDs: Unless references are stored in the clear this is most likely not achievable.

3) Attacker = $\mathcal{M}$ including $\mathcal{S}$:

- Learn b_i: The same center-search attack can be performed as in the case where $\mathcal{AS}$ and $\mathcal{S}$ are the attacker. Because the sensor can send any input and any identity, the attacker does not have to wait for a matching sample. Because $\mathcal{M}$ will most likely learn some information about the result of the comparison procedure, this goal is pracitally trivially achieved.
- Trace U_i with different IDs: Most likely not achievable because there is no direct access to or little control over the references from different $\mathcal{DB}$.

4) Attackers combining $\mathcal{AS}$, $\mathcal{DB}$ or $\mathcal{M}$ including $\mathcal{S}$: The individual attackers ($\mathcal{AS}$, $\mathcal{DB}$ or $\mathcal{M}$) including $\mathcal{S}$ are already very powerful and it is considered that the combinations of these are able to learn b_i. Tracing $\mathcal{U}_i$ with multiple IDs seems harder to achieve unless $\mathcal{AS}$ is involved.

IV. Application to Existing Constructions

The system model in Section II-A introduces a logical distinction in functionality and promotes a distributed implementation of these functionalities as a baseline to ensure the protection of biometric data. The schemes that will be analyzed in this section can be defined in this system model. As such they all have in common that they try to protect biometric references against attackers that can be mapped on one or more functional components, irrespective of a fully distributed implementation or a client-server setting. These attackers are at first sight consistent with the adversary model in Section II-B. The presented schemes also have in common that they are defined in the honest-but-curious adversary model. However, it is not known what the impact is when one or more of the logical entities become malicious. How our framework can be used to assess this impact will be demonstrated in this section.

The attacks in this section are based on the generic attacks described in Section III. We apply them against two complex cryptographic protocols that use homomorphic encryption, namely in Section IV-A for a scheme by Bringer *et al.* [19] and in Section IV-B for a scheme by Barbosa *et al.* [20]. We then describe another kind of attacks by looking at a scheme by Stoianov [22] in Section IV-C.

A. *Bringer* et al. *ACISP 2007*

1) Description: In [19], Bringer *et al.* presented a new security model for biometric authentication protocols that separates the tasks of comparing, storing and authorizing an authentication request amongst different entities: a fully trusted sensor $\mathcal{S}$, an authentication server $\mathcal{AS}$, a database $\mathcal{DB}$ and a matching service $\mathcal{M}$. The goal was to prevent any of the latter three to learn the relation between some identity and the biometric features that relate to it. Their model forms the basis of our current framework and in this model they presented a scheme that applies the Goldwasser-Micali cryptosystem [34]. Let $\mathcal{E}_{\text{GM}}(m, pk)$ denote encryption of a message bit m under public key pk and let $\mathcal{D}_{\text{GM}}(c, sk)$ denote decryption of a ciphertext c with private key sk. For any $m, m' \in \{0, 1\}$ we have the homomorphic property $\mathcal{D}_{\text{GM}}(\mathcal{E}_{\text{GM}}(m, pk) \times \mathcal{E}_{\text{GM}}(m', pk), sk) = m \oplus m'$. The scheme in [19] goes as follows.

During enrollment $\mathcal{U}_i$ registers with $\mathcal{AS}$. He then gets an index i and a pseudonym ID_i. Let N denote the total number of records in the system. Database $\mathcal{DB}$ receives and stores (b_i, i) where b_i stands for $\mathcal{U}_i$'s biometric template, a binary vector of dimension M, i.e., $b_i = (b_{i,1}, b_{i,2}, \ldots, b_{i,M})$. In the following, we suppose that i is also the index of the record b_i in the database $\mathcal{DB}$.

A key pair is generated for the system. Matcher $\mathcal{M}$ possesses the secret key sk. The public key pk is known by $\mathcal{S}$, $\mathcal{AS}$ and $\mathcal{DB}$. The authentication server $\mathcal{AS}$ stores a table of relations (ID_i, i) for $i \in \{1, \ldots, N\}$ and $\mathcal{DB}$ contains the enrolment data $b_1, \ldots, b_N$.

When user $\mathcal{U}_i$ wants to authenticate himself, $\mathcal{S}$ will send an encrypted sample $\mathcal{E}_{\text{GM}}(b'_i, pk)$ and ID_i to $\mathcal{AS}$. The authentication server $\mathcal{AS}$ will request the encrypted reference $\mathcal{E}_{\text{GM}}(b_i, pk)$ from $\mathcal{DB}$ and combine it with the encrypted sample. Because of the homomorphic property, $\mathcal{AS}$ is able to obtain $\mathcal{E}_{\text{GM}}(b'_i \oplus b_i, pk)$. Note that the encryption is bitwise so $\mathcal{AS}$ will permute the M encryptions and forward these to $\mathcal{M}$. Because $\mathcal{M}$ has the secret key sk, $\mathcal{M}$ can decrypt the permuted XOR-ed bits and compute the Hamming distance between the sample and the reference. Let the maximum allowed Hamming distance be t.

The security of this protocol is proved in [19] under the assumption that all the entities in the system will not collude and are honest-but-curious. It is this assumption that we challenge in our framework, which leads to the following attack.

2) Authentication Server Adversary ($\mathcal{A}$=$\mathcal{AS}$): The following attack shows how a malicious authentication server $\mathcal{AS}$ can learn the enrolled biometric template b_i corresponding to some identity ID_i. To do so the authentication server $\mathcal{AS}$ requests the template b_i without revealing ID_i and receives from $\mathcal{DB}$ the encrypted template that was stored during enrolment, i.e., $\mathcal{E}_{\text{GM}}(b_i, pk) = \langle \mathcal{E}_{\text{GM}}(b_{i,1}, pk), \ldots, \mathcal{E}_{\text{GM}}(b_{i,M}, pk) \rangle$.

The attack consists of a bitwise search performed by $\mathcal{AS}$ in the encrypted domain. First $\mathcal{AS}$ computes the encryption of a zero bit

$\mathcal{E}_{\text{GM}}(0, pk)$. Now $\mathcal{AS}$ will take the first encrypted bit $\mathcal{E}_{\text{GM}}(b_{i,1}, pk)$, repeat it $t+1$ times and add $M-t-1$ encryptions of a zero bit. Note that the ciphertext $\mathcal{E}_{\text{GM}}(b_{i,1}, pk)$ can be re-randomized so that it is impossible to detect that the duplicate ciphertexts are "copies". If $b_{i,1}$ is one, the total Hamming distance as computed by $\mathcal{M}$ will be $t+1$ and $\mathcal{M}$ will return "not ok". If $b_{i,1}$ is zero, $\mathcal{M}$ will return "ok". This process can be repeated for all bits of b_i, hence, $\mathcal{AS}$ can learn b_i bit by bit in M queries. To further disguise the attack $\mathcal{AS}$ can apply permutations and add up to t encryptions of one-bits to make the query look genuine.

3) Matcher and Sensor Adversary ($\mathcal{A}$=$\mathcal{M}$+$\mathcal{S}$): A bitwise search similar to the previous attack allows to learn b_i in M queries. Note that the sensor $\mathcal{S}$ can sends arbitrary inputs to the system. The attacker repeats the following steps for each bit in b_i:

- $\mathcal{S}$ sends the encryption of $\bar{0} = \langle 0, \ldots, 0 \rangle$;
- $\mathcal{M}$ receives $b_i \oplus \bar{0}$ bitwise but permuted and records the weight of $b_i \oplus \bar{0}$;
- $\mathcal{S}$ toggles a bit in the $\bar{0}$ vector in position x and sends it to $\mathcal{AS}$;
- $\mathcal{M}$ observes the changed weight (+1 or -1) and learns the bit at position x in b_i.

4) Discussion: What makes the first attack ($\mathcal{A}$=$\mathcal{AS}$) feasible is that all bits are encrypted separately. Moreover, it is not enforced that $\mathcal{AS}$ combines the input from the sensor and from the database. To counteract this threat, one could require $\mathcal{S}$ to sign the input and force $\mathcal{DB}$ to merge the input with the reference. In this way $\mathcal{AS}$ does not receive the reference $\mathcal{E}_{\text{GM}}(b_i, pk)$ but the combination $\mathcal{E}_{\text{GM}}(b'_i \oplus b_i, pk)$. However, $\mathcal{AS}$ can still learn b'_i and $b'_i \oplus b_i$. Additional measures have to be taken to prevent this, e.g., $\mathcal{DB}$ could be required to sign $\mathcal{E}_{\text{GM}}(b'_i \oplus b_i, pk)$, which will be verified by $\mathcal{M}$. Note that in the case where $\mathcal{AS}$ and $\mathcal{DB}$ collude, these countermeasures are not sufficient anymore.

The second attack ($\mathcal{A}$=$\mathcal{M}$+$\mathcal{S}$) is more difficult to prevent. Each authentication transaction can be expressed in our system model as follows.

- $\mathcal{S} \rightarrow \mathcal{AS} : f_1(b'_i)$
- $\ldots$
- $\mathcal{M} \leftarrow \mathcal{AS} : f_3(b'_i, b_i)$
- $\mathcal{M} \rightarrow \mathcal{AS} : f_4(d(b'_i, b_i))$

The attack can be executed as soon as $\mathcal{M}$ is able to learn from a pair $f_3(b'_i, b_i), f_3(b''_i, b_i)$ how the distances $d(b'_i, b_i)$ and $d(b''_i, b_i)$ relate to each other (i.e., greater than, smaller than or equal to) without necessarily knowing the actual distances.

B. *Barbosa* et al. *ACISP 2008*

1) Description: In [20] Barbosa *et al.* presented a new protocol for biometric authentication, following [19] (see previous Section IV-A). A notable difference between these two comes from the fact that [19] compares two biometric templates by their Hamming distancewhereas [20] classifies one biometric template into different classes using an SVM classifier (support vector machine, see [35] for details). Biometric templates are represented as features vector where each feature is an integer, i.e., $b_i = \langle b_{i,1}, \ldots, b_{i,k} \rangle \in \mathbb{N}^k$. Barbosa *et al.* encrypt this vector, feature by feature, with the Paillier cryptosystem [36]. In particular, they exploit its homomorphic property to compute its SVM classifier (think of a sum of scalar products) in the encrypted domain. However, as the features are encrypted one by one, an adversary can do something similar as the attack described in Section IV-A.

Let $\mathcal{E}_{\text{Paillier}}$ (resp. $\mathcal{D}_{\text{Paillier}}$) denote the encryption (resp. decryption) with Paillier's cryptosystem. This cryptosystem enjoys a homomorphic property which ensures that the product of two encrypted plaintexts corresponds to the encryption of their sum: for $m_1, m_2 \in \mathbb{Z}_n$ we have that $\mathcal{D}_{\text{Paillier}}(\mathcal{E}_{\text{Paillier}}(m_1) \times \mathcal{E}_{\text{Paillier}}(m_2)) = m_1 + m_2 \mod n$. Note that $\mathbb{Z}_n$ is the plaintext space of the Paillier cryptosystem.

The SVM classifier takes as input U classes (or users) and S samples per class, and determines support vectors $SV_{i,j}$ and weights $\alpha_{i,j}$ for $1 \leq i \leq S$ and $1 \leq j \leq U$. Following the notation in [20], let $v = (v_1, \ldots, v_k) = b_i$ denote a freshly captured biometric sample. For this sample the classifier computes

$$cl^{(j)}_{\text{SVM}}(v) = \sum_{i=1}^{S} \alpha_{i,j} \sum_{l=1}^{k} v_l (SV_{i,j})_l \text{ for } j = 1, \ldots, U\,. \qquad (7)$$

With this vector $cl_{\text{SVM}}(v)$, it is possible to determine which class is the most likely for v or to reject it. The support vectors $SV_{i,j}$ and the weight coefficients $\alpha_{i,j}$ are the references that are stored by $\mathcal{DB}$.

Briefly, the scheme of Barbosa *et al.* works as follows:

1) The sensor $\mathcal{S}$ captures a fresh biometric sample and encrypts each of the features of its template $v = (v_1, \ldots, v_k)$ with Paillier's cryptosystem and sends it to the authentication server $\mathcal{AS}$. Let auth $= (\mathcal{E}_{\text{Paillier}}(v_1), \ldots, \mathcal{E}_{\text{Paillier}}(v_k))$.
2) The database $\mathcal{DB}$ computes an encrypted version of the SVM classifier for this biometric data: $c_j = \prod_{i=1}^{S} (\prod_{l=1}^{k} [auth_j]_l^{[SV_{i,j}]_l})^{\alpha_{i,j}}$ where $[.]_l$ denotes the l^{th} component of a tuple. This c_j corresponds to the encryption of the $cl^{(j)}_{\text{SVM}}$ with Paillier's cryptosystem as defined above. The database returns the values c_j to $\mathcal{AS}$.
3) The authentication server $\mathcal{AS}$ scrambles the values c_j and forwards them to $\mathcal{M}$.[2].
4) The matcher $\mathcal{M}$, using the private key of the system, decrypts the components of the SVM classifier and performs the classification of v. The classification returns the class for which the value $cl^{(j)}_{\text{SVM}}$ is maximal.
5) Based on the output of $\mathcal{M}$, $\mathcal{AS}$ determines the real identity of $\mathcal{U}_i$ (in case of non-rejection).

2) Authentication Server Adversary ($\mathcal{A}$=$\mathcal{AS}$): The following attack shows how $\mathcal{AS}$ can learn b_i. In this scheme, the biometric reference data that are stored by $\mathcal{DB}$, i.e., the support vectors $SV_{i,j}$ and the weight coefficients $\alpha_{i,j}$, represent hyperplanes that are used for classification. These k-dimensional hyperplanes are expressed as linear combinations of enrolment samples (the support vectors). We will show how they can be recovered dimension by dimension.

Let us rewrite (7) as

$$\begin{aligned} cl^{(j)}_{\text{SVM}}(v) &= v_1 \sum_{i=1}^{S} \alpha_{i,j} (SV_{i,j})_1 + \cdots + v_k \sum_{i=1}^{S} \alpha_{i,j} (SV_{i,j})_k \\ &= v_1 \beta_{j,1} + \cdots + v_k \beta_{j,k}\,. \end{aligned}$$

By sending a vector $v = \langle 1, 0, \ldots, 0 \rangle$ to $\mathcal{DB}$, $\mathcal{AS}$ will retrieve the encryption of $\beta_{j,1} = \sum_{i=1}^{S} \alpha_{i,j} (SV_{i,j})_1$ for each user, indexed by j, in the database.

Instead of sending all $c_j = \mathcal{E}_{\text{Paillier}}(\beta_{j,1})$ to $\mathcal{M}$, only one value will be kept by $\mathcal{AS}$, e.g., $c_1 = \mathcal{E}_{\text{Paillier}}(\beta_{1,1})$. The authentication server will set $c_2 = \mathcal{E}_{\text{Paillier}}(x)$ for some value $x \in \mathbb{Z}_n$ and all other $c_j = \mathcal{E}_{\text{Paillier}}(0)$. The matcher $\mathcal{M}$ will return the index of the class with the greatest value, which is 1 if $\beta_{1,1} > x$ and 2 if $\beta_{1,1} \leq x$.

The initial value of $x = n/2$. If $\beta_{1,1} > x$ then $\mathcal{AS}$ will adjust x to $n/2 + n/4$, otherwise $x = n/2 - n/4$. By repeating this process and adjusting the value x, the exact value $\beta_{1,1}$ can be learned after $\log_2 n$ queries. Hence, the reference data of a single user can be learned in $k \log_2 n$ queries to the matcher.

3) Discussion: As in Section IV-A this attack succeeds because features are encrypted separately and there is no check to see if the sample and the reference were really merged.

[2]In [20], the entity that makes the decision is refered to as the verification server. To be consistent with our model we continue to use the term matcher.

C. Stoianov SPIE 2010

1) Description: In [22], Stoianov introduces several authentication schemes relying on the Blum-Blum-Shub pseudo-random generator. We focus on the database setting from the paper (cf. Section 7 of [22]. In this setting there is a service provider SP that performs the verification. Consistent with our model, we will call this entity the matcher $\mathcal{M}$. Sample and reference are combined before being sent to $\mathcal{M}$ and although this is not explicitly mentioned in [22] we designate this functionality to the authentication server $\mathcal{AS}$ in our model.

In the schemes of [22], the biometric data b are binarized and are combined with a random codeword c coming from an error-correcting code to form a secure sketch or code offset $b \oplus c$. When a new capture b' is made, whenever b' is close to b (using the Hamming distance) it is possible to recover c from $b \oplus b' \oplus c$ using error correction. This technique is known as the fuzzy commitment scheme of Juels and Wattenberg [5]. An additional layer of protection is added by encrypting the secure sketch using Blum-Goldwasser.

The Blum-Blum-Shub pseudo-random generator [37] is a tool used in the Blum-Goldwasser asymmetric cryptosystem [38]. From a seed x_0 and a public key, a pseudo-random sequence S is generated. In the following, S is XOR-ed to the biometric data to be encrypted. By doing so, the state of the pseudo-random generator is updated to x_{t+1}. From x_{t+1} and the private key, the sequence S can be recomputed.

In this system of Stoianov, $\mathcal{M}$ generates the keys and sends the public key to $\mathcal{S}$. On enrollment

1) Sensor $\mathcal{S}$ computes $(S \oplus b \oplus c, x_{t+1})$ where:
   - Sample b is the freshly captured biometric data,
   - String S is a pseudo-random sequence and x_{t+1} is the state of the Blum-Blum-Shub pseudo-random generator as described above, and
   - c is a random codeword which makes the secure sketch $c \oplus b$;
2) Sensor $\mathcal{S}$ sends $S \oplus b \oplus c$ to $\mathcal{DB}$;
3) Sensor $\mathcal{S}$ sends x_{t+1} and $H(c)$ to $\mathcal{M}$ where H is a cryptographic hash function.

Using the private key, $\mathcal{M}$ computes S from x_{t+1} and stores it along $H(c)$. Periodically, $\mathcal{M}$ (resp. $\mathcal{DB}$) updates S (resp. $S \oplus b \oplus c$) to $\check{S}$ (resp. $\check{S} \oplus c \oplus b$) with an independent stream cipher.

During authentication sensor $\mathcal{S}$ receives a new sample b' and forwards $(S' \oplus b', x'_{t+1})$ to $\mathcal{AS}$, where S' is a new pseudo-random sequence. It is assumed that there is some sort of authentication server $\mathcal{AS}$ that retrieves $\check{S} \oplus c \oplus b$ from $\mathcal{DB}$ and merges it with $S' \oplus b'$. Finally $S' \oplus b' \oplus \check{S} \oplus b \oplus c$ and x'_{t+1} are sent to $\mathcal{M}$. Using the private key $\mathcal{M}$ recovers S'. From S' and $\check{S}$, $\mathcal{M}$ computes $c \oplus b \oplus b'$, tries to decode it and verifies the consistency of the result with $H(c)$.

2) Matcher Adversary ($\mathcal{A}$=$\mathcal{M}$): Let $\mathcal{M}$ be the primary attacker. It is inherent to the scheme that $\mathcal{M}$ can always trace a valid user over different queries by looking at the codeword c, which is revealed after a successful authentication. Depending on the entity that colludes with $\mathcal{M}$ additional attacks can be deviced.

If $\mathcal{M}$ and $\mathcal{DB}$ collude ($\mathcal{A}$=$\mathcal{M}$+$\mathcal{DB}$) they learn the sketch $c \oplus b$. This implies that they can trace users with different identities following the linkability attack based on the decoding of the sum of two sketches as described in [11]. From a genuine match, $\mathcal{M}$ learns c and b.

If $\mathcal{M}$ and $\mathcal{S}$ collude ($\mathcal{A}$=$\mathcal{M}$+$\mathcal{S}$) they control and always learn the input sample b'. By setting $b' = 0$ they learn $c \oplus b$ from a single query. If a successful authentication occurred, the adversary learns everything.

If $\mathcal{M}$ and $\mathcal{AS}$ collude ($\mathcal{A}$=$\mathcal{M}$+$\mathcal{AS}$) they always learn the input sample b'. They can learn the sketch $c \oplus b$ for any reference and thus trace users with different identities as in the case ($\mathcal{A}$=$\mathcal{M}$+$\mathcal{DB}$). They learn the reference b after successful authentication.

3) Authentication Server Adversary ($\mathcal{A}$=$\mathcal{AS}$): In the current scheme, bits are not encrypted bit per bit independently. Moreover, they are masked with streams generated via Blum-Blum-Shub and a codeword so attacks as in Sections IV-A and IV-B are no longer possible. Nevertheless, there is still a binary structure that $\mathcal{AS}$ may exploit.

Assume that $\mathcal{AS}$ knows $S' \oplus b'$ that leads to a positive decision, i.e., $\mathcal{M}$ accepts b' because $\mathrm{d}(b, b') \leq t$. Then $\mathcal{AS}$ can start from $S' \oplus b'$ and add progressively some errors until he reaches a negative result. Then, he backtracks one step by decreasing the error weight by one to come back to the last positive result. This gives $\mathcal{AS}$ an encrypted template $S' \oplus b''$. Consider now the vector $S' \oplus b'' \oplus \check{S} \oplus c \oplus b$ and replace the first bits (say of small length l) by a l bits vector x.

- For all possible values of x, $\mathcal{AS}$ sends the resulting vector (the first block is changed by the value x) to $\mathcal{M}$ who acts as a decision oracle.
- If several values give a positive result, then $\mathcal{AS}$ increases the errors on all but the first block.
- This is repeated until only one value of x gives a positive result.
- When this step is reached, $\mathcal{AS}$ has found the value x with no errors, i.e., he learns the first block of $S' \oplus \check{S} \oplus c$.
- $\mathcal{AS}$ proceeds to the next block.

Following this strategy, it is feasible to recover all the bits of $b \oplus b'$. If $\mathcal{AS}$ colludes with $\mathcal{S}$, he can retrieve the full reference template b as soon as $\mathcal{S}$ knows one sample that is close to b. This attack corresponds to the center search attack (cf. Section III below).

4) Discussion: In a way similar to the inherent traceability of users by $\mathcal{M}$, there are no mechanisms described that protect against $\mathcal{DB}$ tracing $\mathcal{U}_i$ over different queries, i.e., by tracking $\check{S}+c+b$ lookups.

We note that the matcher $\mathcal{M}$ is very powerful because he knows the secret key, which allows computing S', and $\check{S}$. As soon as $\mathcal{M}$ colludes with one of the other entities he is able to learn everything from a genuine match or a false accept.

V. Conclusion

Biometric authentication protocols that are found in the literature are usually designed in the honest-but-curious model assuming that there are no malicious insider adversaries. In this paper, we have challenged that assumption and shown how some existing protocols are not secure against such adversaries. Such analysis is extremely relevant in the context of independent database providers. Much attention was given to an authentication server attacker, which is a central and powerful entity in our model.

Clearly, there are still a number of problems that must be solved before fully anonymous biometric authentication can be achieved. To prevent the attacks that were presented, stronger enforcement of the protocol design is needed: many attacks succeed because transactions can be duplicated or manipulated. For example, $\mathcal{M}$ cannot verify whether $\mathcal{AS}$ has correctly merged the input from $\mathcal{S}$ and $\mathcal{DB}$. Moreover, $\mathcal{DB}$ cannot verify whether $\mathcal{AS}$ is executing queries that originate from a query made by $\mathcal{S}$. The same holds for $\mathcal{M}$ who cannot verify that inputs originate from queries from $\mathcal{S}$ and to $\mathcal{DB}$.

Some attacks have a low complexity, e.g., linear in the size of the references. This is the case for the decomposed reference attack, which particularly targets bit-wise or feature-wise encryptions. The motivation for applying encryption on the feature level is to benefit from the homomorphic properties of the applied cryptosystem. As a consequence, better cryptographic primitives are needed that allow performing computations on encrypted data without allowing to attack individual features.

Since $\mathcal{S}$ is the entity that supposed to trigger an authentication transaction it is fundamental limitation that it can never be verified

whether $\mathcal{S}$ is making queries that really originate from an authentication request or is generating malicious queries. As a consequence, the combination of $\mathcal{S}$ and $\mathcal{M}$ as attacker requires that $\mathcal{M}$ is able to return the result of the comparison between a fresh sample and a reference to $\mathcal{AS}$ without learning anything about the result itself.

Due to the generic design of our model, several other schemes in the literature fit our model and might suffer from the presented attacks. Nevertheless, as they are not always designed with the same entities, an adaptation might be required. Some others are not compatible at all; for instance those for which the security relies on a user-secret key stored on the user side.

REFERENCES

[1] J.-P. M. G. Linnartz and P. Tuyls, "New shielding functions to enhance privacy and prevent misuse of biometric templates," in *AVBPA*, ser. LNCS, J. Kittler and M. S. Nixon, Eds., vol. 2688. Springer, 2003, pp. 393–402.

[2] I. Buhan, J. Doumen, P. H. Hartel, and R. N. J. Veldhuis, "Fuzzy extractors for continuous distributions," in *ASIACCS*, F. Bao and S. Miller, Eds. ACM, 2007, pp. 353–355.

[3] Y. Dodis, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," in *Advances in Cryptology - EUROCRYPT 2004*, ser. LNCS, C. Cachin and J. Camenisch, Eds., vol. 3027. Springer, 2004, pp. 523–540.

[4] G. Davida, Y. Frankel, and B. Matt, "On enabling secure applications through off-line biometric identification," *Proc. of the IEEE Symp. on Security and Privacy – S&P '98*, pp. 148–157, May 1998.

[5] A. Juels and M. Wattenberg, "A fuzzy commitment scheme," in *CCS '99: Proc. of the 6th ACM Conf. on Computer and Communications Security*. New York, NY, USA: ACM Press, 1999, pp. 28–36.

[6] A. Juels and M. Sudan, "A fuzzy vault scheme," in *Proc. of IEEE Int. Symp. on Information Theory, Lausanne, Switzerland*, A. Lapidoth and E. Teletar, Eds. IEEE Press, 2002, p. 408.

[7] N. K. Ratha, J. H. Connell, and R. M. Bolle, "Enhancing security and privacy in biometrics-based authentication systems," *IBM Systems J.*, vol. 40, no. 3, pp. 614–634, 2001.

[8] N. Ratha, J. Connell, R. Bolle, and S. Chikkerur, "Cancelable biometrics: A case study in fingerprints," in *Pattern Recognition, 2006. ICPR 2006. 18th Int. Conf. on*, vol. 4, 2006, pp. 370–373.

[9] N. K. Ratha, S. Chikkerur, J. H. Connell, and R. M. Bolle, "Generating cancelable fingerprint templates," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 4, pp. 561–572, 2007.

[10] X. Boyen, "Reusable cryptographic fuzzy extractors," in *CCS '04: Proc. of the 11th ACM Conf. on Computer and Communications Security*. New York, NY, USA: ACM, 2004, pp. 82–91.

[11] K. Simoens, P. Tuyls, and B. Preneel, "Privacy weaknesses in biometric sketches," in *2009 30th IEEE Symp. on Security and Privacy*, May 2009, pp. 188–203.

[12] I. Buhan, J. Breebaart, J. Guajardo, K. de Groot, E. Kelkboom, and T. Akkermans, "A quantitative analysis of crossmatching resilience for a continuous-domain biometric encryption technique," in *First Int. Workshop on Signal Processing in the EncryptEd Domain, SPEED 2009*, 2009.

[13] A. Nagar and A. Jain, "On the security of non-invertible fingerprint template transforms," in *Information Forensics and Security, 2009. WIFS 2009. First IEEE Int. Workshop on*, 2009, pp. 81–85.

[14] L. Rila and C. J. Mitchell, "Security protocols for biometrics-based cardholder authentication in smartcards," in *ACNS*, ser. LNCS, J. Zhou, M. Yung, and Y. Han, Eds., vol. 2846. Springer, 2003, pp. 254–264.

[15] J. Bringer, H. Chabanne, T. A. M. Kevenaar, and B. Kindarji, "Extending match-on-card to local biometric identification," in *Biometric ID Management and Multimodal Communication, BioID-Multicomm 2009*, ser. LNCS, J. Fierrez, J. Ortega-Garcia, A. Esposito, A. Drygajlo, and M. Faundez-Zanuy, Eds., vol. 5707. Springer, 2009, pp. 178–186.

[16] P. Tuyls, B. Škorić, and T. Kevenaar, Eds., *Security with Noisy Data: Private Biometrics, Secure Key Storage and Anti-Counterfeiting*. Springer-Verlag London, 2007.

[17] B. Schoenmakers and P. Tuyls, *Private Profile Matching*, ser. Philips Research Book Series. Springer-Verlag, New York, 2006, vol. 7, pp. 259–272.

[18] J. Bringer, H. Chabanne, D. Pointcheval, and Q. Tang, "Extended private information retrieval and its application in biometrics authentications," in *CANS*, ser. LNCS, F. Bao, S. Ling, T. Okamoto, H. Wang, and C. Xing, Eds., vol. 4856. Springer, 2007, pp. 175–193.

[19] J. Bringer, H. Chabanne, M. Izabachène, D. Pointcheval, Q. Tang, and S. Zimmer, "An application of the Goldwasser-Micali cryptosystem to biometric authentication," in *ACISP*, ser. LNCS, J. Pieprzyk, H. Ghodosi, and E. Dawson, Eds., vol. 4586. Springer, 2007, pp. 96–106.

[20] M. Barbosa, T. Brouard, S. Cauchie, and S. M. de Sousa, "Secure biometric authentication with improved accuracy," in *ACISP*, ser. LNCS, Y. Mu, W. Susilo, and J. Seberry, Eds., vol. 5107. Springer, 2008, pp. 21–36.

[21] J. Bringer and H. Chabanne, "An authentication protocol with encrypted biometric data," in *AFRICACRYPT*, ser. LNCS, S. Vaudenay, Ed., vol. 5023. Springer, 2008, pp. 109–124.

[22] A. Stoianov, "Cryptographically secure biometric," in *SPIE Biometric Technology for Human Identification VII, volume 7667*, 2010.

[23] P. Failla, Y. Sutcu, and M. Barni, "eSketch: a privacy-preserving fuzzy commitment scheme for authentication using encrypted biometrics," in *Proc. of the 12th ACM workshop on Multimedia and security (MM-Sec'10)*. ACM, 2010, pp. 241–246.

[24] M. D. Raimondo, M. Barni, D. Catalano, R. D. Labati, P. Failla, T. Bianchi, D. Fiore, R. Lazzeretti, V. Piuri, F. Scotti, and A. Piva, "Privacy-preserving fingercode authentication," in *Proc. of the 12th ACM workshop on Multimedia and security (MMSec'10)*. ACM, 2010, pp. 231–240.

[25] J. Bringer, H. Chabanne, and K. Simoens, "Blackbox security of biometrics (invited paper)," in *Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), 2010 6th Int. Conf. on*, 2010, pp. 337–340.

[26] HSNW, "DHS develops shared biometrics database with DoD," 8 March 2011,. [Online]. Available: http://www.homelandsecuritynewswire.com/dhs-develops-shared-biometrics-database-dod

[27] FBI, "Next generation identification." [Online]. Available: http://www.fbi.gov/about-us/cjis/fingerprints_biometrics/ngi

[28] European Commission DG Home Affairs, "Visa information system." [Online]. Available: http://ec.europa.eu/home-affairs/policies/borders/borders_it_vis_en.htm

[29] Q. Tang, J. Bringer, H. Chabanne, and D. Pointcheval, "A formal study of the privacy concerns in biometric-based remote authentication schemes," in *ISPEC*, ser. LNCS, L. Chen, Y. Mu, and W. Susilo, Eds., vol. 4991. Springer, 2008, pp. 56–70.

[30] A. K. Jain, P. Flynn, and A. A. Ross, Eds., *Handbook of Biometrics*. Springer, 2008.

[31] S. Prabhakar, S. Pankanti, and A. K. Jain, "Biometric recognition: Security and privacy concerns," *IEEE Security Privacy*, vol. 1, no. 2, pp. 33–42, March-April 2003.

[32] A. Cavoukian, "Privacy by design resolution," in *32nd International Conference of Data Protection and Privacy Commissioners, 27-29 October 2010, Jerusalem, Israel*, 2010. [Online]. Available: http://www.ipc.on.ca/site_documents/pbd-resolution.pdf

[33] A. K. Jain, K. Nandakumar, and A. Nagar, "Biometric template security," *EURASIP Journal on Advances in Signal Processing*, vol. 2008, no. Article ID 579416, p. 17, 2008.

[34] S. Goldwasser and S. Micali, "Probabilistic encryption and how to play mental poker keeping secret all partial information," in *STOC*. ACM, 1982, pp. 365–377.

[35] K. Crammer and Y. Singer, "On the algorithmic implementation of multiclass kernel-based vector machines," *J. of Machine Learning Research*, vol. 2, pp. 265–292, 2001.

[36] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *EUROCRYPT*, ser. LNCS, J. Stern, Ed., vol. 1592. Springer, 1999, pp. 223–238.

[37] L. Blum, M. Blum, and M. Shub, "A simple unpredictable pseudo-random number generator," *SIAM J. Comput.*, vol. 15, no. 2, pp. 364–383, 1986.

[38] M. Blum and S. Goldwasser, "An efficient probabilistic public-key encryption scheme which hides all partial information," in *CRYPTO*, ser. LNCS, G. Blakley and D. Chaum, Eds., vol. 196. Springer, 1984, pp. 289–302.