Reasoning About Privacy Properties of Architectures Supporting Group Authentication and Application to Biometric Systems

Julien Bringer[1], Hervé Chabanne[1,2], Daniel Le Métayer[3], and Roch Lescuyer[1](✉)

[1] Morpho, Issy-Les-Moulineaux, France
roch.lescuyer@morpho.com
[2] Télécom ParisTech, Paris, France
[3] INRIA, Université de Lyon, Lyon, France

Abstract. This paper follows a recent line of work that advocates the use of formal methods to reason about privacy properties of system architectures. We propose an extension of an existing formal framework, motivated by the need to reason about properties of architectures including group authentication functionalities. By group authentication, we mean that a user can authenticate on behalf of a group of users, thereby keeping a form of anonymity within this set. Then we show that this extended framework can be used to reason about privacy properties of a biometric system in which users are authenticated through the use of group signatures.

Keywords: Privacy by design · Formal methods · Biometric systems

1 Introduction

The privacy-by-design approach promotes the consideration of privacy requirements from the early design stage of a system. As an illustration of the importance of this topic, the General Data Protection Regulation adopted by the European trilogue (the European Commission, the European Parliament and the Council) in December 2015 [7] introduces privacy-by-design and privacy-by-default as legal obligations. Architectural choices have a strong effect on the privacy properties provided by a system. For this reason, the authors of [1] argue that key decisions regarding the design of a system should be taken at the architecture level. They introduce a formal framework for reasoning about privacy properties of architectures. The description of an architecture within this framework specifies the capacities of each component, the communications between them, the location of the computations and the data, and the trust relationships between the stakeholders. A dedicated privacy logic is used to express the privacy properties of the architectures. The use of formal methods enables precise definitions of properties and comparisons between architectures. It also makes it possible to provide a rigorous justification for the design choices.

As a first contribution of this paper, we propose an extension of this formal framework and show that it can be used to reason about properties of architectures supporting group authentication. By group authentication, we mean that a user can authenticate on behalf of a group of users. Several cryptographic primitives have been designed to achieve this goal. Our work provides the formal tools needed to reason about the properties of architectures involving these primitives, especially the guarantees that are provided in terms of privacy.

As a second contribution of this paper, we apply our extended framework to biometric systems. In a biometric system, users are authenticated with their biometric traits. The work of [3] uses the formal framework of [1] to reason about privacy properties of biometric architectures but it cannot deal with group signatures. We show that the extended framework can be used to reason about privacy properties of a biometric system in which users are authenticated by group signatures.

The interest of group signature in the context of biometrics has been shown in different contexts. For example, the biometric system architecture analysed in this paper was proposed in TURBINE [16], a European project which aimed at solving privacy concerns regarding the use of fingerprint biometrics for ID management. The application of this architecture was a pharmacy product research system. Pharmacists, for instance working at their selling desks, authenticate themselves to a pharmacy administration system. Authentication is based on a card owned by the employee, as well as its fingerprint. Thanks to the use of group signatures, a remote server (which does not get the fingerprint) is convinced that a valid enrolled user authenticates without knowing precisely who he is among the set of valid users (aka the employees).

Organization of the paper. Section 2 supplies an overview of the formal framework of [1]. Section 3 introduces our extension of this model. Section 4 presents the biometric architecture we are interested in, describes it within the architecture language of the formal framework, and analyses its privacy properties. Finally, we discuss in Sect. 5 some variants of the biometric architecture, before concluding in Sect. 6.

2 Reasoning About Privacy Properties of Architectures

In this section, we provide an overview of the framework introduced in [1] which is the foundation for our work. The interested reader can refer to [1] for a more complete description of the framework.

This framework relies on a dedicated epistemic logic for expressing privacy properties. Epistemic logics are good candidates to express privacy properties since they deal with the notion of knowledge. However, the standard *possible worlds* semantics for these logics lead to a well-known issue called the *logical omniscience problem* [9]. In a nutshell, any agent knows all the logical consequences of his knowledge. To get around this issue, the authors of [1] adopt an approach based on *deductive algorithmic knowledge* [13]. In this context, each component of an architecture is endowed with its own deductive capabilities.

Architectures are described with a dedicated architecture language. Then the semantics of a privacy property is defined as the architectures in which the property holds.

2.1 A Privacy Architecture Language

First of all, the functionality of a system is described by a set $\Omega = \{X = T\}$ of equations over the following term language.

$$T ::= X \mid c \mid F(X_1, \ldots, X_m)$$

A term T might be a variable X ($X \in Var$), a constant c ($c \in Const$) or F a function applied to some variables ($F \in Fun$).

Then the architecture of a system is described by the following architecture language.

$$\begin{aligned}
A &::= \{R\} \\
R &::= Has_i(X) \mid Receive_{i,j}(\{St\}, \{X\}) \mid Trust_{i,j} \\
&\qquad\qquad\quad \mid Compute_G(X = T) \quad \mid Verify_i(\{St\})
\end{aligned}$$

$$\begin{aligned}
St &::= Pro \mid Att & Att &::= Attest_i(\{Eq\}) \\
Pro &::= Proof_i(\{P\}) & Eq &::= Pred(T_1, \ldots, T_m) \\
P &::= Att \mid Eq
\end{aligned}$$

An architecture A is associated to a set of components $\mathcal{C} = \{C_1, \ldots, C_{|\mathcal{C}|}\}$. In the architectural primitives, i and j stand respectively for C_i, C_j and $G \subseteq \mathcal{C}$ denotes a set of components.

In the above syntax, $\{Z\}$ denotes a set of elements of category Z. $Pred$ denotes a predicate, the set of predicates depending on the architectures to be considered. $Has_i(X)$ denotes the fact that component C_i possesses (or is the origin of) the value of X, which may correspond to situations in which X is stored on C_i or C_i is a sensor collecting the value of X. $Receive_{i,j}(\{St\}, \{X\})$ means that C_i can receive the values of variables in $\{X\}$ together with the statements in $\{St\}$ from C_j.

$Attest_i(\{Eq\})$ is the declaration by C_i that the properties in $\{Eq\}$ hold and $Proof_i(\{P\})$ is the delivery by C_i of a set of proofs of properties. $Verify_i$ is the verification by component C_i of the corresponding statements (proof or authenticity). $Compute_G(X = T)$ means that the set of components G can compute the term T and assign its value to X and $Trust_{i,j}$ represents the fact that component C_i trusts component C_j.

Graphical data flow representations can be derived from architectures expressed in this language. For the sake of readability, we use both notations in the next sections.

All architectures are assumed to satisfy minimal consistency assumptions, in order to restrict the analysis to those which make sense. For instance, if a component sends a variable, we assume that this variable can be sent, computed or received by the component.

Events are instantiations of the architectural primitives (trust relations excepted). Traces are sequences of events, defined according to the following trace language.

$$\begin{aligned}\theta &::= \mathsf{Seq}(\epsilon)\\ \epsilon &::= Has_i(X : V) \mid Receive_{i,j}(\{St\}, \{X : V\})\\ &\qquad \mid Compute_G(X = T^{\epsilon}) \qquad \mid Verify_i(\{St\})\end{aligned}$$

$\mathsf{Seq}(\epsilon)$ denotes an ordered sequence of events ϵ. When instantiating a primitive containing a variable X, the notation $X : V$ means that the variable X receives the value V. Let Val be the set of values that the variables can take. T^{ϵ} is a term where values have been assigned to variables. The set $Val_{\perp}$ is defined as $Val \cup \{\perp\}$ where $\perp \notin Val$ is a specific symbol used to denote that a variable has not been assigned.

As for architectures, only traces satisfying consistency assumptions are considered. $\langle\rangle$ denotes the empty trace (with no event).

A trace θ of events is said compatible with an architecture A if each event in θ (except the computations) can be obtained by instantiation of an element of A (*Receive*, *Verify*, etc.). Let $T(A)$ be the set of traces which are compatible with an architecture A.

Each component C_i is associated with a dependence relation Dep_i. For a variable Y and a set $\mathcal{X}$ of variables, $Dep_i(Y, \mathcal{X})$ – equivalently $(Y, \mathcal{X}) \in Dep_i$ – means that the value of Y can be obtained by the component C_i if it gets access to the value of X, for each $X \in \mathcal{X}$.

Each component C_i is also associated with a deductive system, noted $\rhd_i$, allowing it to derive new knowledge. $\rhd_i$ is defined as a relation between equations $\{Eq_1, \ldots, Eq_n\} \rhd_i Eq_0$, where equations over terms are defined according to the following syntax.

$$Eq ::= Pred(T_1, \ldots, T_m) \mid Eq \wedge Eq$$

By a slight abuse of notations, Eq is an overloaded notation of the Eq definition in the language architecture, where conjunctions of equations are also possible.

Finally, the semantics of an architecture is defined from the traces of events. Each component is associated with a state. Each event in a trace of events affects the state of each component involved in the event. The semantics $S(A)$ of an architecture A is defined as the set of states reachable by compatible traces.

2.2 A Privacy Logic

Privacy properties of architectures are expressed with the following language.

$$\phi ::= Has_i(X) \mid Has_i^{none}(X) \mid K_i(Eq) \mid \phi_1 \ \wedge \ \phi_2.$$

The knowledge operator K_i represents the knowledge of the component C_i. The formula Has_i represents the fact that C_i can get access to variable X.

The semantics $S(\phi)$ of a property ϕ is defined as the set of architectures where ϕ is satisfied. The fact that a property ϕ is satisfied by a (consistent) architecture A is defined for each property as follows.

- A satisfies $Has_i(X)$ if there is a reachable state of C_i in which X is not undefined.
- A satisfies $Has_i^{none}(X)$ if no compatible trace leads to a state in which C_i assigns a value to X.
- A satisfies $K_i(Eq)$ if from all reachable states C_i can deduce Eq.
- A satisfies $\phi_1 \wedge \phi_2$ if A satisfies ϕ_1 and A satisfies ϕ_2.

Based on the semantics of properties, [1] introduces a set of deductive rules which can be used to reason about privacy properties of architectures. This deductive system is shown correct and complete with respect to the semantics of the properties.

$A \vdash \phi$ denotes that ϕ can be derived from A – in other words, that there exists a derivation tree such that each step belongs to the axiomatics and the leaf is $A \vdash \phi$. A subset of this axiomatics, useful for this paper, is presented in Fig. (1a).

3 Adding a Group Attestation to the Formal Model

As a first step to extend the architecture language of [1], we introduce the primitive $Attest_G(E)$ where G is a group of components and E a set of equations. This primitive generalizes $Attest_i(E)$ which involves a single component C_i. Section 3.1 defines the semantics of the traces containing these events and Sect. 3.2 extends the set of deductive rules.

3.1 Semantics of Traces

The semantics of a trace is defined by specifying, for each event, its effect on the states of the components.

The state of a component is either the *Error* state or a pair consisting of: (i) a variable state assigning values to variables, and (ii) a property state defining the current knowledge of a component. In the initial state of an architecture A, denoted $Init^A = \langle Init_1^A, \ldots, Init_{|\mathcal{C}|}^A \rangle$, the variables are undefined and the knowledge state only contains the trust primitives.

Let σ denote the global state, and σ_i denote the state of component i. The semantics of traces, denoted S_T, is defined recursively over sequences of events.

$$\begin{aligned} S_T(\langle\rangle, \sigma) &= \sigma \\ S_T(\epsilon \cdot \theta, \sigma) &= S_T(\theta, S_E(\epsilon, \sigma)). \end{aligned}$$

$$\mathsf{H1}\ \frac{Has_i(X) \in A}{A \vdash Has_i(X)} \qquad \mathsf{H2}\ \frac{Receive_{i,j}(S,E) \in A \qquad X \in E}{A \vdash Has_i(X)}$$

$$\mathsf{H3}\ \frac{Compute_G(X = T) \in A \qquad C_i \in G}{A \vdash Has_i(X)} \qquad \mathsf{I}\wedge\ \frac{A \vdash \phi_1 \qquad A \vdash \phi_2}{A \vdash \phi_1 \wedge \phi_2}$$

$$\mathsf{H4}\ \frac{Dep_i(Y, \mathcal{X}) \qquad \forall X \in \mathcal{X}\colon A \vdash Has_i(X)}{A \vdash Has_i(Y)} \qquad \mathsf{HN}\ \frac{A \nvdash Has_i(X)}{A \vdash Has_i^{none}(X)}$$

$$\mathsf{K1}\ \frac{Compute_G(X = T) \in A \qquad C_i \in G}{A \vdash K_i(X = T)} \qquad \mathsf{K}\wedge\ \frac{A \vdash K_i(Eq_1) \qquad A \vdash K_i(Eq_2)}{A \vdash K_i(Eq_1 \wedge Eq_2)}$$

$$\mathsf{K3}\ \frac{Verify_i(Proof_j(E)) \in A \qquad Eq \in E}{A \vdash K_i(Eq)}$$

$$\mathsf{K}\rhd\ \frac{E \rhd_i Eq_0 \qquad \forall Eq \in E\colon A \vdash K_i(Eq)}{A \vdash K_i(Eq_0)}$$

(a) Subset of the axiomatics of [1]

$$\mathsf{K4}^{+}\ \frac{\begin{array}{cc} Verify_i(Proof_j(E)) \in A & \forall k \in G : Trust_{i,k} \in A \\ Attest_G(E') \in E & Eq \in E' \end{array}}{A \vdash K_i(Eq)}$$

$$\mathsf{K5}^{+}\ \frac{Verify_i(Attest_G(E)) \in A \qquad \forall k \in G : Trust_{i,k} \in A \qquad Eq \in E}{A \vdash K_i(Eq)}$$

(b) Our extended axioms

Fig. 1. Axiomatics

The function S_E, which defines the effect of the events, is defined for each type of event. The modification of a state is noted $\sigma[\sigma_i/(v, pk)]$ the variable and knowledge states of C_i being replaced by v and pk respectively. $\sigma[\sigma_i/Error]$ denotes that the *Error* state is reached for component C_i. A component reaching an *Error* state is no longer involved in any action.

Restricting our attention to the events which contains a group attestation leads us to consider the events $Verify_i(Attest_G(E))$ and $Verify_i(Proof_j(E))$. The semantics of the verification events are defined according to the (implicit) semantics of the underlying verification procedures. In both cases, the knowledge state of the component is updated if the verification passes, otherwise the component reaches an *Error* state. The variable state is not affected. Informally, a verification event containing a generalized attestation statement generates new knowledge only if all possible authors of the attestation are trusted by the verifying component C_i.

$$S_E(Verify_i(Proof_j(E)),\sigma) = \begin{cases} \sigma[\sigma_i/Error] & \text{if the proof is not valid,} \\ \sigma[\sigma_i/(\sigma_i^v, \sigma_i^{pk} \cup new^{pk}_{Proof})] & \text{otherwise,} \end{cases}$$

$$S_E(Verify_i(Attest_G(E)),\sigma) = \begin{cases} \sigma[\sigma_i/Error] & \text{if the attestation is not valid,} \\ \sigma[\sigma_i/(\sigma_i^v, \sigma_i^{pk} \cup new^{pk}_{Attest})] & \text{otherwise,} \end{cases}$$

where the new knowledge new^{pk}_{Proof} is defined as:

$$new^{pk}_{Proof} := \{Eq \mid Eq \in E \;\vee\; (\exists G \subseteq \mathcal{C} : (Attest_G(E') \in E \;\wedge\; Eq \in E' \\ \wedge\; \forall k \in G : Trust_{i,k} \in \sigma_i^{pk}))\}; \quad (1)$$

and the new knowledge new^{pk}_{Attest} is defined as:

$$new^{pk}_{Attest} := \{Eq \mid Eq \in E \;\wedge\; \forall k \in G : Trust_{i,k} \in \sigma_i^{pk}\}. \quad (2)$$

3.2 Axiomatics

The next challenge to deal with group attestation is the extension of the set of deductive rules and the proof of the correctness and completeness properties still hold. Our axioms for group attestation are presented in Fig. (1b). In the remaining of this section, we show that the correctness and the completeness of the axiomatics still hold with these new axioms.

Correctness. Let A be a consistent architecture and ϕ a property. The correctness theorem states that if there exists a derivation tree for this property ($A \vdash \phi$), then this property holds in the architecture ($A \in S(\phi)$).

The proof is made by induction on the depth of the tree $A \vdash \phi$. Let us restrict our attention to the cases where ($\mathsf{K4}^+$) and ($\mathsf{K5}^+$) are used. That is, let us assume that $A \vdash K_i(Eq)$, and that the derivation tree is of depth 1. By definition of the set of axioms, such a proof is obtained by application of ($\mathsf{K1}$), ($\mathsf{K3}$), ($\mathsf{K4}^+$) or ($\mathsf{K5}^+$). Let us focus on the $\mathsf{K4}^+$ and $\mathsf{K5}^+$ cases.

$\mathsf{K4}^+$. Let us assume that $Verify_i(Proof_j(E)) \in A$, $Attest_G\ (E') \in E$ and $\forall k \in G$: $Trust_{i,k} \in A$ for some i, j and G. Our goal is to prove that $\forall Eq \in E'$: $A \in S(K_i(Eq))$.

Let us consider a given state $\sigma' \in S_i(A)$. By the architecture semantics, there exists a consistent trace θ', compatible with A, such that $\sigma' = S_T(\theta', Init^A)$. Two cases may happen. Either θ' contains an event $Verify_i(Proof_j(E))$ such that $Attest_G(E') \in E$, and we let $\theta := \theta'$, or it is not. In the latter case, we extend θ' into a trace θ such that θ contains such an event without breaking the consistency of the trace.

In either cases, there exists a trace θ which extends θ' and contains an event $Verify_i(Proof_j(E))$ such that $Attest_G(E') \in E$. Let $\sigma = S_T(\theta, Init^A)$. Since an *Error* state has not been reached (we have $\sigma' \in S_i(A)$), and since $\forall k \in G$: $Trust_{i,k} \in \sigma_i^{pk}$ by definition of the initial state, then by the semantics of the group attestation (Eq. (1)) we have $\forall Eq \in E$: $Eq \in \sigma_i^{pk}$.

By the definition of the architectures semantics, we deduce that $\sigma \in S(A)$. The prefix order over the traces together with the definition of the semantics of the trace induce a prefix order over the states, hence $\sigma \geq_i \sigma'$. By the reflexivity of the deductive algorithmic knowledge, we have $\forall Eq \in E'$: $\sigma_i^{pk} \triangleright_i Eq$. By the semantics of the properties, we conclude that $\forall Eq \in E'$: $A \in S(K_i(Eq))$.

$\mathsf{K5}^+$. Let us assume that $Verify_i(Attest_G(E)) \in A$ and $\forall k \in G$: $Trust_{i,k} \in A$. We must show that $\forall Eq \in E$: $A \in S(K_i(Eq))$. Adaptation of the $\mathsf{K4}^+$ to the $\mathsf{K5}^+$ case is straightforward, invoking Eq. (2) of the trace semantics instead of Eq. (1).

Completeness. Let A be a consistent architecture and ϕ a property. The completeness theorem states that if the property holds in the architecture ($A \in S(\phi)$), then there exists a derivation tree for this property ($A \vdash \phi$).

The proof is made by induction over the definition of the property ϕ. We restrict our attention here to the knowledge operator K_i. Let us assume that $A \in S(K_i(Eq))$ for a given component C_i and equation Eq. We must show that $A \vdash K_i(Eq)$.

By the semantics of properties, $A \in S(K_i(Eq))$ means that $\forall \sigma' \in S_i(A)$: $\exists \sigma \in S_i(A)$: $\sigma_i^{pk} \triangleright_i Eq$. By the semantics of architectures, $\exists \theta \in T(A)$ such that ($\sigma = S_T(\theta, Init^A)$ and $\sigma_i^{pk} \triangleright_i Eq$). By the semantics of the traces, this implies one among the following statements: either there exists $Compute_G(X = T^\epsilon) \in \theta$ where $Eq := (X = T)$ and $C_i \in G$ and T^ϵ is obtained from T (by assigning values to variables); or there exists $Verify_i(Proof_j(E)) \in \theta$ where $Eq \in E$; or there exists $Verify_i(Proof_j(E)) \in \theta$ where $Attest_G(E') \in E$, $Eq \in E'$ and $\forall k \in G$: $Trust_{i,k} \in \sigma_i^{pk}$ and $Eq \in E'$; or there exists $Verify_i(Attest_G(E)) \in \theta$, $Eq \in E$ and $\forall k \in G$: $Trust_{i,k} \in \sigma_i^{pk}$.

By the compatibility of the traces, we deduce that: either $Compute_G(X) \in A$ where $Eq := (X = T)$ and $C_i \in G$; or $Verify_i(Proof_j(E)) \in A$ where $Eq \in E$; or $Verify_i(Proof_j(E)) \in A$ where $Attest_G(E') \in E$, $Eq \in E'$ and $\forall k \in G$: $Trust_{i,k} \in A$ and $Eq \in E'$; or $Verify_i(Attest_G(E)) \in A$, $Eq \in E'$ and $\forall k \in G$: $Trust_{i,k} \in A$. We conclude that $A \vdash K_i(Eq)$ by applying (respectively) ($\mathsf{K1}$), ($\mathsf{K3}$), ($\mathsf{K4}^+$) or ($\mathsf{K5}^+$).

4 Modelling a Biometric Architecture Supporting Group Authentication

4.1 A Biometric Architecture Using Group Signatures

Biometric systems involve two main phases: enrolment and verification (either authentication or identification) [10]. Enrolment is the registration phase, in which the biometric traits of a person are collected and recorded within the system. In the *authentication* mode, a fresh biometric trait is collected and compared with the registered one by the system to check that it corresponds to the claimed identity. In the *identification* mode, a fresh biometric data is collected and the corresponding identity is searched in a database of enrolled biometric references.

A group signature scheme [2] is an advanced cryptographic mechanism. It enables a user to sign messages on behalf of a group of users while staying anonymous inside this group. With a (public) verification algorithm, anyone can be convinced, given a group public key, a message, and a signature, that *a certain* member of the group authenticates the message.

The biometric system introduced in [4] aims at achieving some anonymity from the server's point of view. The server is convinced that the authentication was successful for a certain enrolled user, but has no information about which among them. During the enrolment, a biometric reference is registered by the issuer. The issuer derives a user secret key from the biometric template and computes a group secret key, that is, a certificate attesting the enrolment inside the group. The user gets a card containing its biometric reference and the group certificate.

During the verification phase, the terminal gets a fresh capture of the biometric trait and computes a fresh template. A match between the fresh template and the reference is performed by the terminal. In case of success, the terminal derives the user secret key from the reference, produces a group signature thanks to the user secret key and the certificate (both are needed to produce a valid signature), and sends the signature to the server. The server checks the signature attesting that a registered user authenticates. If the signature is valid, the server is convinced of the correctness of the matching. However, it has no access to the biometric templates, neither to the identity of the user who authenticates.

4.2 Description Within the Formal Framework

For the sake of clarity, let us distinguish the biometric system and its formalization. We denote by B_{gs} the biometric system introduced in [4] and A_{gs} its definition within the formal framework, which we present below.

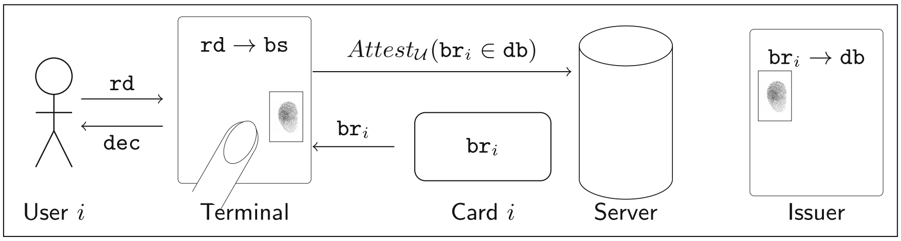
rd
dec
$\mathtt{rd} \rightarrow \mathtt{bs}$
$Attest_{\mathcal{U}}(\mathtt{br}_i \in \mathtt{db})$
$\mathtt{br}_i$
$\mathtt{br}_i$
$\mathtt{br}_i \rightarrow \mathtt{db}$
User i
Terminal
Card i
Server
Issuer

Fig. 2. High-level view of the biometric system architecture using group signatures

Upper case sans serif letters in A_{gs} denote components. Components of the A_{gs} architecture are a set of N enrolled users $\mathcal{U} := \{\mathsf{U}_1, \ldots, \mathsf{U}_N\}$ (each user U_i owning a card C_i), a server S, an issuer I and a terminal modelled by two components TM and TS. The issuer I enrols the users. The server S manages a database containing the enrolled templates. The terminal is equipped with a sensor used to acquire biometric traits. Formally, the terminal is split into two components TM and TS, corresponding respectively to its two functionalities. The matcher TM, acquires the fresh template and performs the comparison, and the signer TS authenticates on behalf of the group of users. As shown by the variants below, this distinction is motivated by the different trust assumptions a designer may consider.

Type letters denote variables. $\mathtt{br}_i$ denotes the biometric reference template of the user U_i built during the enrolment phase. $\mathtt{rd}$ denotes a raw biometric data provided by the user during the verification phase. $\mathtt{bs}$ denotes a fresh template derived from $\mathtt{rd}$ during the verification phase. A threshold $\mathtt{thr}$ is used during the verification phase as a closeness criterion for the biometric templates. The output $\mathtt{dec}$ of the verification is the result of the matching between the fresh template $\mathtt{bs}$ and the enrolled templates $\mathtt{br}$, considering the threshold $\mathtt{thr}$. $\mathtt{db}$ denotes the database of the registered biometric templates.

As in [3], we focus on the verification phase and assume that enrolment has already been done. The database $\mathtt{db}$ is computed by the issuer from all the references, using the function $DB \in Fun$. A verification process is initiated by the terminal receiving as input a raw biometric data $\mathtt{rd}$ from the user. The terminal, more precisely the TM component, extracts the fresh biometric template $\mathtt{bs}$ from $\mathtt{rd}$ using the function $Extract \in Fun$. The matching is expressed by the function $\mu \in Fun$ which takes as arguments two biometric templates and the threshold $\mathtt{thr}$. The terminal reads in the card the biometric template $\mathtt{br}$. The user receives the final decision $\mathtt{dec}$ of the matching from the terminal TM. Then the terminal, here the TS component, attests that the fresh template belongs to the set of enrolled templates.

The complete description of A_{gs} within the architecture language is as follows. Figure 2 sketches this description. When indices i are used, it is assumed that the corresponding primitive exists in A_{gs} for all users. For instance $Has_{\mathsf{I}}(\mathtt{br}_i) \in A_{\mathsf{gs}}$ implicitly means that $\forall \mathsf{U}_i \in \mathcal{U}$: $Has_{\mathsf{I}}(\mathtt{br}_i) \in A_{\mathsf{gs}}$.

$$\begin{aligned}
A_{\mathsf{gs}} := \big\{ & Has_{\mathsf{I}}(\mathtt{br}_i), Has_{\mathsf{U}_i}(\mathtt{rd}), Has_{\mathsf{TM}}(\mathtt{thr}),\\
& Compute_{\mathsf{I}}(\mathtt{db} = DB(\mathtt{br}_1, \ldots, \mathtt{br}_N)), Compute_{\mathsf{TM}}(\mathtt{bs} = Extract(\mathtt{rd})),\\
& Compute_{\mathsf{TM}}(\mathtt{dec} = \mu(\mathtt{br}_i, \mathtt{bs}, \mathtt{thr})), Trust_{\mathsf{S},\mathsf{U}_i}, Trust_{\mathsf{S},\mathsf{TM}}, Trust_{\mathsf{S},\mathsf{TS}},\\
& Receive_{\mathsf{I},\mathsf{U}_i}(\{Attest_{\mathsf{U}_i}(\mathtt{br}_i \in \mathtt{db})\}, \{\}), Receive_{\mathsf{TM},\mathsf{U}_i}(\{\}, \{\mathtt{rd}\}),\\
& Receive_{\mathsf{C}_i,\mathsf{I}}(\{Attest_{\mathsf{U}_i}(\mathtt{br}_i \in \mathtt{db})\}, \{\mathtt{br}_i\}), Receive_{\mathsf{U}_i,\mathsf{TM}}(\{\}, \{\mathtt{dec}\}),\\
& Receive_{\mathsf{TM},\mathsf{C}_i}(\{\}, \{\mathtt{br}_i\}), Receive_{\mathsf{TS},\mathsf{TM}}(\{\}, \{\mathtt{dec}\}),\\
& Receive_{\mathsf{TS},\mathsf{C}_i}(\{Attest_{\mathsf{U}_i}(\mathtt{br}_i \in \mathtt{db})\}, \{\mathtt{br}_i\}),\\
& Receive_{\mathsf{S},\mathsf{TS}}(\{Attest_{\mathcal{U}}(\mathtt{br}_i \in \mathtt{db})\}, \{\}), Verify_{\mathsf{S}}(\{Attest_{\mathcal{U}}(\mathtt{br}_i \in \mathtt{db})\})\big\}
\end{aligned}$$

To complete the description of A_{gs}, it remains to define the dependence relations between the variables. The database is computed from all the references: $\forall j \in \mathcal{C}$: $(\mathtt{db}, \{\mathtt{br}_1, \ldots, \mathtt{br}_N\})$. Conversely, access to $\mathtt{db}$ gives access to all $\mathtt{br}_i$: $\forall j \in \mathcal{C}, \mathsf{U}_i \in \mathcal{U}$: Dep_j $(\mathtt{br}_i, \{\mathtt{db}\})$. Moreover, $\forall j \in \mathcal{C}, \mathsf{U}_i \in \mathcal{U}$: we also have $(\mathtt{bs}, \{\mathtt{rd}\})$, $(\mathtt{dec}, \{\mathtt{br}_i, \mathtt{bs}\})$, $(\mathtt{dec}, \{\mathtt{br}_i, \mathtt{rd}\}) \in Dep_j$.

4.3 Trusting a Group of Users

In the biometric system architecture A_{gs}, the group of users is trusted by the server, which is denoted $\forall \mathsf{U}_i \in \mathcal{U}$: $Trust_{\mathsf{S},\mathsf{U}_i}$. However, the formalization does not define which cryptographic primitive is used in the concrete B_{gs} system. Let us discuss this point in more detail.

In a group signature scheme, users are typically not trusted, but a group manager, called the issuer, is trusted. When it enrols a user, the issuer provides a group secret key, aka a membership certificate – concretely, a signature of some secret user-specific data. In other words, it *attests* that the user is enrolled. Then the untrusted user *proves* that it is enrolled (by supplying a zero-knowledge proof of her user secret data and the corresponding membership certificate). In our case, the server does not trust the card, but trusts the issuer of the card. The card contains an attestation that the user was indeed enrolled by the issuer, here a certificate for a group signature, *i.e.*, a group secret key.

The point to be noticed is that we do not model its internal machinery in our formal architecture. We only express the fact that the group is trusted. Whether this trust assumption is justified or not in practice is not part of the reasoning about architecture: it rather regards the justification of the choice of certain primitives to achieve the functionality. With the same trust assumption (all users are trusted), other primitives can be used, as ring signatures [14], where a member authenticates on behalf of a group without group manager.

The use of group signatures is a choice made at the protocol level. Checking the conformity between the protocols and the architecture is out of scope of this paper. This line of work has been initiated in [15].

4.4 Application of the Axiomatics

We now reason about the privacy properties of the A_{gs} architecture from the server point's of view. A_{gs} should enable the server to be sure that a certain enrolled user authenticates, but the authenticated user is anonymous from the server's point of view: $A_{\mathsf{gs}} \vdash K_{\mathsf{S}}(\mathtt{br}_i \in \mathtt{db})$. But the server should have no access to the templates: $A_{\mathsf{gs}} \vdash Has_{\mathsf{S}}^{none}(\mathtt{br}_i)$.

Regarding the template protection, the statement $A_{\mathsf{gs}} \vdash Has_{\mathsf{S}}^{none}(\mathtt{br}_i)$ is shown using rule HN. A subtlety here is the presence of the dependence between the biometric template $\mathtt{br}_i$ and the database $\mathtt{db}$. Therefore we first need to show $A \nvdash Has_{\mathsf{S}}(\mathtt{db})$.

$$\frac{\begin{array}{cc} Has_{\mathsf{S}}(\mathtt{db}) \notin A_{\mathsf{gs}} & \nexists T\text{: } Compute_{\mathsf{S}}(\mathtt{db} = T) \in A_{\mathsf{gs}} \\ \nexists \overrightarrow{X} : (\mathtt{db}, \overrightarrow{X}) \in Dep_{\mathsf{S}} & \nexists j, \nexists S, \nexists E\text{: } Receive_{\mathsf{S},j}(S, E) \in A_{\mathsf{gs}} \wedge \mathtt{db} \in E \end{array}}{A_{\mathsf{gs}} \nvdash Has_{\mathsf{S}}(\mathtt{db})}$$

Now HN can be applied.

$$\mathsf{HN}\ \frac{\begin{array}{c}(\mathtt{br}_i, \{\mathtt{db}\}) \in Deps_{\mathsf{S}} \qquad A_{\mathsf{gs}} \not\vdash Has_{\mathsf{S}}(\mathtt{db}) \\ Has_{\mathsf{S}}(\mathtt{br}_i) \notin A_{\mathsf{gs}} \qquad \nexists T\colon Computes_{\mathsf{S}}(\mathtt{br}_i = T) \in A_{\mathsf{gs}} \\ \nexists j, \nexists S, \nexists E\colon Receives_{\mathsf{S},j}(S, E) \in A_{\mathsf{gs}} \wedge \mathtt{br}_i \in E \end{array}}{\dfrac{A_{\mathsf{gs}} \not\vdash Has_{\mathsf{S}}(\mathtt{br}_i)}{A_{\mathsf{gs}} \vdash Has_{\mathsf{S}}^{none}(\mathtt{br}_i)}}$$

$A_{\mathsf{gs}} \vdash Has_{\mathsf{S}}^{none}(\mathtt{bs})$ is also shown by an application of HN.

Since the server trusts the users, an application of $\mathsf{K5}^+$ shows that the server is ensured that some enrolled user authenticates.

$$\mathsf{K5}^+\ \frac{Verif_{\mathsf{S}}(Attest_{\mathcal{U}}(\mathtt{br}_i \in \mathtt{db})) \in A_{\mathsf{gs}} \qquad \forall \mathsf{U}_i \in \mathcal{U} : Trust_{\mathsf{S},\mathsf{U}_i} \in A_{\mathsf{gs}}}{A_{\mathsf{gs}} \vdash K_{\mathsf{S}}(\mathtt{br}_i \in \mathtt{db})}$$

5 Variants

Several variants [4] of the biometric system B_{gs} can be expressed and analyzed in our formal framework.

5.1 Lowering the Trust on the Group Signing Functionality

If the server trusts the matching functionality TM of the terminal but does not trust its signer functionality TS, then the component TS must supply a proof that some user is enrolled. The architecture, denoted $A_{\mathsf{gs}}^{\mathsf{p}}$, becomes:

$$\begin{aligned} A_{\mathsf{gs}}^{\mathsf{p}} := A_{\mathsf{gs}} \setminus\ & \big\{Receives_{\mathsf{S},\mathsf{TS}}(\{Attest_{\mathcal{U}}(\mathtt{br}_i \in \mathtt{db})\}, \{\}), Trust_{\mathsf{S},\mathsf{TS}}, \\ & \quad Verify_{\mathsf{S}}(\{Attest_{\mathcal{U}}(\mathtt{br}_i \in \mathtt{db})\big\} \\ \cup\ & \big\{Receives_{\mathsf{S},\mathsf{TS}}(\{Proof_{\mathsf{TS}}(Attest_{\mathcal{U}}(\mathtt{br}_i \in \mathtt{db}))\}, \{\}), \\ & \quad Verify_{\mathsf{S}}(\{Proof_{\mathsf{TS}}(Attest_{\mathcal{U}}(\mathtt{br}_i \in \mathtt{db}))\})\big\} \end{aligned}$$

An application of the new rule $\mathsf{K4}^+$ enable to prove that the server is ensured that some enrolled user authenticates.

$$\mathsf{K4}^+\ \frac{Verif_{\mathsf{S}}(Proof_{\mathsf{TS}}(Attest_{\mathcal{U}}(\mathtt{br}_i \in \mathtt{db}))) \in A_{\mathsf{gs}}^{\mathsf{p}} \qquad \forall \mathsf{U}_i \in \mathcal{U} : Trust_{\mathsf{S},\mathsf{U}_i} \in A_{\mathsf{gs}}^{\mathsf{p}}}{A_{\mathsf{gs}}^{\mathsf{p}} \vdash K_{\mathsf{S}}(\mathtt{br}_i \in \mathtt{db})}$$

5.2 Combination with Match-On-Card

In the A_{gs} architecture, the card is a plastic card. The biometric reference is just printed on it, together with a group secret key. To enhance the protection of the reference, a smart-card can be used instead of a plastic card, as in the Match-On-Card (MOC) technology [8,11,12]. The card stores the reference template, and the reference never leaves the card. During a verification, the card receives the fresh biometric template, carries out the comparison with its reference, and sends the decision back. The terminal trusts the smart card for the correctness of the matching. This trust is justified by the fact that the card is a tamper-resistant hardware element.

The A_{gs} architecture in which the plastic card is replaced by a smart-card performing a MOC is modelled as follows. In addition to the comparison, the card also computes the group authentication.

$$\begin{aligned}
&A_{\mathsf{gs}}^{\mathsf{moc}} := \big\{Has_{\mathsf{I}}(\mathtt{br}_i), Has_{\mathsf{U}_i}(\mathtt{rd}), Has_{\mathsf{TM}}(\mathtt{thr}), Trust_{\mathsf{TM},\mathsf{C}_i}, Trust_{\mathsf{S},\mathsf{U}_i},\\
&\quad Compute_{\mathsf{I}}(\mathtt{db} = DB(\mathtt{br}_1, \ldots, \mathtt{br}_N)), Compute_{\mathsf{TM}}(\mathtt{bs} = Extract(\mathtt{rd})),\\
&\quad Compute_{\mathsf{C}_i}(\mathtt{dec} = \mu(\mathtt{br}_i, \mathtt{bs}, \mathtt{thr})), Receive_{\mathsf{I},\mathsf{U}_i}(\{Attest_{\mathsf{U}_i}(\mathtt{br}_i \in \mathtt{db})\}, \{\}),\\
&\quad Receive_{\mathsf{C}_i,\mathsf{I}}(\{Attest_{\mathsf{U}_i}(\mathtt{br}_i \in \mathtt{db})\}, \{\mathtt{br}_i\}), Receive_{\mathsf{TM},\mathsf{U}_i}(\{\}, \{\mathtt{rd}\}),\\
&\quad Receive_{\mathsf{TM},\mathsf{C}_i}(\{Attest_{\mathsf{C}_i}(\mathtt{dec} = \mu(\mathtt{br}_i, \mathtt{bs}, \mathtt{thr})), Attest_{\mathcal{U}}(\mathtt{br}_i \in \mathtt{db})\}, \{\mathtt{dec}\}),\\
&\quad Receive_{\mathsf{C}_i,\mathsf{TM}}(\{\}, \{\mathtt{bs}\}), Receive_{\mathsf{U}_i,\mathsf{TM}}(\{\}, \{\mathtt{dec}\}),\\
&\quad Receive_{\mathsf{S},\mathsf{TM}}(\{Attest_{\mathcal{U}}(\mathtt{br}_i \in \mathtt{db})\}, \{\}), Verify_{\mathsf{S}}(\{Attest_{\mathcal{U}}(\mathtt{br}_i \in \mathtt{db})\}),\\
&\quad Verify_{\mathsf{TM}}(\{Attest_{\mathsf{C}_i}(\mathtt{dec} = \mu(\mathtt{br}_i, \mathtt{bs}, \mathtt{thr}))\})\big\}
\end{aligned}$$

Using rule HN, it is easy to show that no component apart from I and C_i gets access to $\mathtt{br}_i$.

The terminal should be convinced that the matching is correct: $A_{\mathsf{gs}}^{\mathsf{moc}} \vdash K_{\mathsf{TM}}(\mathtt{dec} = \mu(\mathtt{br}_i, \mathtt{bs}, \mathtt{thr}))$. The proof relies on the trust placed by the server in the matching component TM of the terminal.

$$\mathsf{K5}^{+}\ \frac{\begin{array}{c} Verify_{\mathsf{TM}}(Attest_{\mathsf{C}_i}(\mathtt{dec} = \mu(\mathtt{br}_i, \mathtt{bs}, \mathtt{thr}))) \in A_{\mathsf{gs}}^{\mathsf{moc}} \\ Trust_{\mathsf{TM},\mathsf{C}_i} \in A_{\mathsf{gs}}^{\mathsf{moc}} \end{array}}{A_{\mathsf{gs}}^{\mathsf{moc}} \vdash K_{\mathsf{TM}}(\mathtt{dec} = \mu(\mathtt{br}_i, \mathtt{bs}, \mathtt{thr}))}$$

Regarding the group authentication, an application of $\mathsf{K5}^{+}$ shows that the server is ensured that some enrolled user authenticates.

5.3 Anonymity Revocation

As shown in [4], an additional mechanism can be used to revoke the anonymity of a group authentication if there is any legal need to do so. After the matching phase, the terminal has to encrypt the fresh template under the public key of a specific tracing authority, to sign all messages together, and to send the authentication result to the server. Then, at a later stage, the tracing authority may decrypt the template and check, with an access to the database of the issuer, that the templates were indeed close. This *a posteriori* check ensures a form of accountability which can be requested in certain contexts.

The formal model introduced in [1] includes an additional architectural primitive, called SpotCheck, which can be used to carry out *a posteriori* checks and therefore to describe the above variant. However, the model including the

SpotCheck primitive is proven complete only when all the functions of the term language are at most unary. Since the comparison between templates, an essential operation of biometric systems, is inherently binary, we would then obtain a correct but incomplete system.

We leave for future work the definition of a formal model with *a posteriori* verifications which would be both correct and complete and would not suffer this arity restriction in the term language.

6 Conclusion

In this paper, we have analysed the privacy properties of a biometric system in which users can remain anonymous from the point of view of a remote server, while the server is still convinced that a valid user authenticates. Table 1 sums up the properties of the different architectures considered here. Architecture $A_{\mathsf{gs}}^{\mathsf{moc}}$ provides the best guarantees in terms of privacy. However, its deployment has a cost, since it requires that each user owns a card with powerful capabilities. Although quite demanding, these assumptions are not out of reach of the current technology [5]. The main variant A_{gs} is more realistic. The choice between A_{gs} and $A_{\mathsf{gs}}^{\mathsf{p}}$ depends on the trust placed on each component in a specific deployment. The possibility to express these trust assumptions in a formal way and to study their consequences is one of the main benefits of the framework presented here because it provides rigorous justifications to make well-informed design choices for the architecture of a system.

Table 1. Comparison between architectures

Arch.	Template protection		Trust relations
	Components accessing the reference $\mathtt{br}_i$	Components accessing the query $\mathtt{bs}$	
A_{gs}	I, C, TM, TS	TM	(S, U_i), (S, TS)
$A_{\mathsf{gs}}^{\mathsf{p}}$	I, C, TM, TS	TM	(S, U_i)
$A_{\mathsf{gs}}^{\mathsf{moc}}$	I, C	TM, C	(S, U_i), (TM, C_i)

Components are: users U_i, terminal components TM and TS, server S, card C, issuer I. A trust relation (i, j) means that component i trusts component j.

Acknowledgment. This work has been partially funded by the French ANR-12-INSE-0013 project BIOPRIV. Part of this work has been conducted within the Inria Project Lab on Privacy CAPPRIS [6].

References

1. Antignac, T., Le Métayer, D.: Privacy architectures: reasoning about data minimisation and integrity. In: Mauw, S., Jensen, C.D. (eds.) STM 2014. LNCS, vol. 8743, pp. 17–32. Springer, Heidelberg (2014)

2. Boneh, D., Shacham, H.: Group signatures with verifier-local revocation. In: ACM Conference on Computer and Communications Security, CCS 2004, pp. 168–177. ACM Press (2004)
3. Bringer, J., Chabanne, H., Le Métayer, D., Lescuyer, R.: Privacy by design in practice: reasoning about privacy properties of biometric system architectures. In: Bjørner, N., Boer, F. (eds.) FM 2015. LNCS, vol. 9109, pp. 90–107. Springer, Heidelberg (2015)
4. Bringer, J., Chabanne, H., Pointcheval, D., Zimmer, S.: An application of the Boneh and Shacham group signature scheme to biometric authentication. In: Matsuura, K., Fujisaki, E. (eds.) IWSEC 2008. LNCS, vol. 5312, pp. 219–230. Springer, Heidelberg (2008)
5. Canard, S., Girault, M.: Implementing group signature schemes with smart cards. In: Smart Card Research and Advanced Application, CARDIS 2002, pp. 1–10. USENIX (2002)
6. CAPPRIS. Collaborative Project on the Protection of Privacy Rights in the Information Society. Inria Project Lab on Privacy. https://cappris.inria.fr/
7. European Parliament. European Parliament legislative resolution of 12 March 2014 on the proposal for a regulation of the European Parliament, of the Council on the protection of individuals with regard to the processing of personal data, on the free movement of such data. General Data Protection Regulation, Ordinary legislative procedure: first reading (2014)
8. Govan, M., Buggy, T.: Acomputationally efficient fingerprint matching algorithm for implementation on smartcards. In: Biometrics: Theory, Applications, and Systems, BTAS 2007, pp. 1–6. IEEE (2007)
9. Halpern, J.Y., Pucella, R.: Dealing with logical omniscience. In: Conference on Theoretical Aspects of Rationality and Knowledge, TARK 2007, pp. 169–176 (2007)
10. Jain, A.K., Ross, A., Prabhakar, S.: An introduction to biometric recognition. IEEE Trans. Circuits Syst. Video Techn. **14**(1), 4–20 (2004)
11. National Institute of Standards and Technology (NIST). MINEXII - an assessment of Match-On-Card technology (2011). http://www.nist.gov/itl/iad/ig/minexii.cfm
12. International Standard Organization. International standard iso/iec 24787 information technology - identification cards - on-card biometric comparison (2010)
13. Pucella, R.: Deductive algorithmic knowledge. J. Log. Comput. **16**(2), 287–309 (2006)
14. Rivest, R.L., Shamir, A., Tauman, Y.: How to leak a secret. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, p. 552. Springer, Heidelberg (2001)
15. Ta, V.T., Antignac, T.: Privacy by design: on the conformance between protocols and architectures. In: Cuppens, F., Garcia-Alfaro, J., Zincir Heywood, N., Fong, P.W.L. (eds.) FPS 2014. LNCS, vol. 8930, pp. 65–81. Springer, Heidelberg (2015)
16. TURBINE. TrUsted Revocable Biometric IdeNtitiEs. Collaborative European project 216339 call FP7-ICT-2007-1 (2007). http://cordis.europa.eu/project/rcn/85447_en.html