Specification and Deployment of Integrated Security Policies for Outsourced Data

Anis Bkakria, Frédéric Cuppens, Nora Cuppens-Boulahia, David Gross-Amblard

Specification and Deployment of Integrated Security Policies for Outsourced Data

Anis Bkakria [1], Frédéric Cuppens [1], Nora Cuppens-Boulahia [1], and David Gross-Amblard [2]

Télécom Bretagne [1]
{anis.bkakria,frederic.cuppens,nora.cuppens}@telecom-bretagne.eu
IRISA,Université de Rennes 1 [2]
david.gross_amblard@irisa.fr

Abstract. This paper presents a well-founded language allowing in one hand data owners to easily specify their security and utility requirements over the data to be outsourced and in an another hand to formalize the set of security mechanisms that can be used for the protection of outsourced data. Based on the formalization of security and utility requirements and security mechanisms properties, we formally identify the best mechanisms, and the best way to combine them to get the best trade-off between utility and security.

Keywords: Security policy, data confidentiality, privacy-preserving, data outsourcing, relational databases, temporal logics of knowledge.

1 Introduction

Because of the rapid evolution of communication technologies, data storage and data processing, outsourcing data to a third-party has grown up over the last few years. Information system architecture adopted by public and private companies is changing for mainly two causes. First, it offers several advantages to the client companies, especially for small ones with limited IT budget as it allows them to reduce the cost of maintaining computing infrastructure and data-rich applications. Second, data collected by companies generally contain sensitive information which must be protected.

Data outsourcing gives rise to many security issues, e.g., confidentiality, integrity, authentication, copyright protection, privacy and anonymity, because outsourced data contains often highly sensitive information which will be stored and managed by third parties. These security issues are traditionally addressed by using security and cryptography mechanisms such as encryption, anonymization, watermarking, fragmentation, etc. In our work, we consider that the set of security mechanisms that can be used for the protection of outsourced data are represented as a toolbox giving security administrators the ability to enforce their security requirements. We develop a logic-based language allowing the security administrators to specify their security and utility requirements

and automatically choose the best mechanisms, and the best way to combine them in order to enforce defined security requirements.

As a case of study, we address the problem of secure data integration in which two data owners storing two private tables having the same set of records on different sets of attributes want to create a joint table containing involved attributes of both private tables. The joint table must satisfy the set of security and utility requirements defined by both data owners. To meet these goals, we first develop a well-founded logic based language allowing to model the used system, the security and utility requirements defined by both data owners, and the security mechanisms that can be used to satisfy them. Second, we show how to use those specifications to choose the best combination of security mechanisms that can satisfy the selected security and utility requirements defined by both data owners.

The reminder of this paper is organized as follows, Section 2 discusses related work. Section 3 describes our approach. Section 4 presents our defined language and the specification of the used system. Section 5 shows the modeling of the policy to be applied over the outsourced information. Section 6 presents the specification of the security mechanisms that can be used to satisfy a defined policy. Section 7 shows how to choose the best combination of security mechanisms that can satisfy a defined policy. Section 8 gives a demonstration of our approach. Finally, Section 9 reports our conclusions.

2 Related work

Many approaches to protect confidentiality and privacy of outsourced data are based on encryption [5,9,10]. Hacigümüs et al. [9] have proposed the first approach aiming to query encrypted data. The proposed technique is based on the definition of a number of buckets on the attribute domain which allows the server-side evaluation of point queries. Hore et al. [10] improve the bucket-based index methods by presenting an efficient method for partitioning the domain of attributes. Many interesting techniques providing protection of outsourced data are based on order preserving encryption OPE schemes [5]. OPE schemes are symmetric-key deterministic encryption schemes which produce cipher-texts that preserve the order of the plain-texts. However, in all mentioned approaches, authors use only encryption to protect sensitive outsourced data which makes query execution on the outsourced encrypted data much more difficult.

Few research efforts have investigated how to combine security mechanisms to protect sensitive outsourced data. In [6], authors combine data fragmentation together with encryption to ensure confidentiality of outsourced mono-relational database. This approach was improved in [1] by combining the best features of encryption and fragmentation to deal efficiently with multi-relation normalized databases. Popa et al. [15] propose an interesting approach called CryptDB. The proposed system relies on a trusted party containing a proxy server allowing the interception of users queries which will be executed over the protected databases. The proxy stores a set of encryption keys allowing to encrypt and

decrypt data and queries. In order to allow the execution of different kind of SQL queries, CryptDB system combines different encryption schemes. For range query, it use an implementation of the Order Preserving Encryption (OPE) [5], computations on numeric data are supported using homomorphic encryption based on the Paillier cryptosystem [14] and matching keywords are supported using searchable encryption [17]. The CryptDB approach offers a solution to the encryption type selection problem by proposing an adaptive scheme that dynamically adjusts encryption strategies. The idea of this approach is to encrypt each data item in many *onions*: onion for equal and order comparison, onion for aggregate operations and onion for word search operations. In each onion, the values is dressed in layers of increasingly stronger encryption, each of these layers provides certain kind of functionality. This approach has two main drawbacks. First, it will significantly increase the size of the encrypted database. Second, in order to enable certain functionality, some encryption layers must be removed by updating the database which can be, for big databases, very expensive in terms of execution time.

Formal verification of security protocols [3,4] has been extensively used to verify security properties such as secrecy [3] and strong secrecy [4]. A security protocol involves two or more principal actors, these actors are classified into honest actors aiming to securely exchange information and dishonest actors (attackers) aiming to subvert the protocol. Therefore, dishonest actors are not constrained to follow the protocol rules. Despite that formal verification-based approaches are efficient in security properties verification, they cannot be used in our case as we consider that the actors are honest-but-curious. They are honest as they are constrained to follow the chosen combination of security mechanisms and they are curious in that they will try to infer protected sensitive information by analyzing the joined table.

3 Proposed approach

In our approach, we strive to design a support tool allowing, for a given security policy, selection of the best combination of mechanisms to enforce this security policy. To achieve this goal, we suggest the following methodology :

- Using an Epistemic Linear Temporal Logic, we defined an expressive language allowing to formally model a system composed of involved entities and the data on which the security policy should be enforced, and formally express the security policy defined by the security administrators.
- We conducted a formal study of the security mechanisms allowing the achievement of a chosen goal. This formal study enables us to extract the security and utility properties that characterize each security mechanism. These properties are formally expressed using our language.
- Based on the system formalization, the security policy formalization and the security mechanisms properties formalization, we formally identify the relevant combination of mechanisms to efficiently enforce the defined security policy.

4 System specification using Epistemic LTL

In this section, we will define and use the language $\mathcal{L}$ to formalize our system. In particular, we will define axioms which describe the basic knowledge of each agent and the formalization of the chosen goal.

4.1 Syntax and semantics

The first-order temporal epistemic language $\mathcal{L}$ is made up of a set of predicates $\mathcal{P}$, propositional connectives $\vee$, $\wedge$, $\neg$, $\rightarrow$ and $\leftrightarrow$, the quantifiers $\forall$, $\exists$. We take the usual set of future connectives $\bigcirc$ (next), $\Diamond$ (Sometime, or eventually), $\Box$ (always) [8]. For knowledge we assume a set of agents $A_g = \{1, \cdots, m\}$ and use a set of unary modal connectives K_j, for $j \in A_g$, in which a formula $K_j\psi$ is to be read as "agent j knows ψ".

Definition 1. *Let φ and ψ be propositions and P_i be a predicate of arity n in $\mathcal{P}$. The set of well-formed formulas of $\mathcal{L}$ is defined as follows:*

$$\phi ::= P_i(t_1, \cdots, t_n) |\ K_i\psi |\ \neg\varphi |\ \varphi \vee \psi |\ \varphi \wedge \psi |\ \bigcirc\varphi |\ \Diamond\varphi |\ \Box\varphi | \varphi \rightarrow \psi |\ \varphi \leftrightarrow \psi \mid \exists x\psi \mid \forall x\psi$$

Definition 2. *An interpretation of the language $\mathcal{L}$ is the triple $\mathcal{K} = (\mathcal{W}, \mathcal{I}, \Phi)$ consisting of a sequence of states $\mathcal{W} = \{w_0, w_1, \cdots\}$, a set of classical first-order structures $\mathcal{I}$ that assigns for each states $w_i \in \mathcal{W}$ a predicate $I_{w_i}(P) : |I_{w_i}|^n \rightarrow \{True, False\}$ for each n-places predicate $P \in \mathcal{P}$ and Φ a transition function which defines transitions between states due to the application of mechanisms (actions). $\Phi(w_i, m_k) = w_j$ if the mechanism m_k transits our system from states w_i to state w_j.*

Definition 3. *Let $\mathcal{W}$ be a sequence of states, w_i $(i \geq 0)$ denote a state of $\mathcal{W}$ and v an assignment. The satisfaction relation $\models$ for a formula ψ of $\mathcal{L}$ is defined as follows:*

- $(w_i, \mathcal{W}) \models P(t_1, \cdots, t_n) \iff I_{w_i}(P)(v(t_1), \cdots, v(t_n)) = True$
- $(w_i, \mathcal{W}) \models \neg\psi \iff (w_i, \mathcal{W}) \not\models \psi$
- $(w_i, \mathcal{W}) \models \psi \rightarrow \varphi \iff (w_i, \mathcal{W}) \not\models \psi$ *or* $(w_i, \mathcal{W}) \models \varphi$
- $(w_i, \mathcal{W}) \models \psi \leftrightarrow \varphi \iff (w_i, \mathcal{W}) \models (\psi \rightarrow \varphi) \wedge (\varphi \rightarrow \psi)$
- $(w_i, \mathcal{W}) \models \forall x\psi \iff (w_i, \mathcal{W}) \models \psi[x/c]$ *for all* $c \in |I_{w_i}|$
- $(w_i, \mathcal{W}) \models \psi \wedge \varphi \iff (w_i, \mathcal{W}) \models \psi$ *and* $(w_i, \mathcal{W}) \models \varphi$
- $(w_i, \mathcal{W}) \models \psi \vee \varphi \iff (w_i, \mathcal{W}) \models \psi$ *or* $(w_i, \mathcal{W}) \models \varphi$
- $(w_i, \mathcal{W}) \models \bigcirc\psi \iff (w_{i+1}, \mathcal{W}) \models \psi$
- $(w_i, \mathcal{W}) \models \Diamond\psi \iff (w_k, \mathcal{W}) \models \psi$ *for some* $k \geq i$
- $(w_i, \mathcal{W}) \models \Box\psi \iff (w_k, \mathcal{W}) \models \psi$ *for all* $k \geq i$

In our approach, we choose to work with relational databases which are composed of tables, attributes, records and values. We suppose that relational databases schemes are known to all agents in the system. Epistemic operator K is only used to represents the knowledge of relation between objects (attribute and records) and values which represent an instantiations of these objects. These relations are represented using the three-places predicates $valueOf$,

$valueOf(R, A, V)$ is to be read "the value of the attribute A in the record R is V". In order to simplify our language $\mathcal{L}$, we transform the epistemic operator K using the two-places predicate $knows$ as following:

$$K_i\ valueOf(R, A, V) \rightarrow valueOf(R, A, V) \wedge knows(i, V) \tag{1}$$

$knows(i, v)$ is to be read "the agent i knows V".

4.2 Data model

A system $\mathcal{S} = \langle \mathcal{O}, \mathcal{T}, \mathcal{A}, \mathcal{R}, \mathcal{V} \rangle$ consists of a finite set of owners $\mathcal{O}$, a finite set of relational tables $\mathcal{T}$, a finite set of attributes $\mathcal{A}$, a finite set of records $\mathcal{R}$ and a finite set of values $\mathcal{V}$. We use the following syntactic conventions. Let $O_1, O_2, \cdots$ be variables over owners $\mathcal{O}$, $T_1, T_2, \cdots$ be variables over relational tables $\mathcal{T}$, $A_1, A_2, \cdots$ be variables over attributes $\mathcal{A}$, $R_1, R_2, \cdots$ be variables over records $\mathcal{R}$ and $V_1, V_2, \cdots$ be variables over the set of values $\mathcal{V}$. We identify the following predicates :

- $belongs(O_1, T_1)$ is satisfied if the owner of the relational table T_1 is O_1.
- $attribute\_of(T_1, A_1)$ is satisfied if A_1 is an attribute of the relational table T_1.
- $recordOf(T_1, R_1)$ is satisfied if R_1 is a record of the relational table T_1.
- $valueOf(R_1, A_1, V_1)$ is satisfied if V_1 is the value of the attribute A_1 in the record R_1.

We denote by Σ the set of formulas representing the formalization of our system using previous predicates. We suppose that the set of formulas Σ are always true in the system (e.g., the table T belongs to the owner O and will belong always to the owner O). This can be formalized as follows:

$$\forall f \in \Sigma.\ \ (w_0, \mathcal{W}) \models \Box f \tag{2}$$

4.3 Specifying Basic Knowledge Axioms

In our system, we consider each data owner as an agent. An owner's knowledge is specified using the following axioms:

$$\begin{aligned} \forall T_1, O_1, A_1, R_1, V_1.\ \ [\ & belongs(O_1, T_1)\ \wedge\ attribute\_of(T_1, A_1)\ \wedge \\ & recordOf(T_1, R_1)\ \wedge\ valueOf(R_1, A_1, V_1) \rightarrow\ knows(O_1, V_1)\ ] \end{aligned} \tag{3}$$

$$\begin{aligned} \forall T_1, O_1, A_1, R_1, V_1.\ \ [\ & \neg belongs(O_1, T_1)\ \wedge\ attribute\_of(T_1, A_1)\ \wedge \\ & recordOf(T_1, R_1)\ \wedge\ valueOf(R_1, A_1, V_1) \rightarrow\ \neg knows(O_1, V_1)\ ] \end{aligned} \tag{4}$$

$$\forall O_1, V_1.\ \ knows(O_1, V_1) \rightarrow\ \bigcirc\ knows(O_1, V_1) \tag{5}$$

$$\forall O, T_1, R, A, V.\ attribute\_of(T_1, A)\ \wedge\ recordOf(T_1, R) \wedge\ valueOf(R, A, V) \wedge \\ knows(O, V) \leftrightarrow\ belongs(O, T_1) \vee \big(\exists T_2, T_3. belongs(O, T_2) \wedge joinOf(T_1, T_2, T_3) \\ \wedge join\_involved(T_1, A)\ \wedge\ \neg protected(T_1, A, O)\big) \quad (6)$$

Axiom 3 means that an owner knows all information stored in tables that belong to him while axiom 4 means that an owner has no knowledge about the information stored in tables that do not belong to him. Axiom 5 states that data owners never forget information they know. Axiom 6 means that an owner O knows the values assumed by an attribute A of the table T_1 if and only if the table T_1 belongs to the owner O or there exists a table T_3 representing the join of a table T_2 and the table T_1 in which the attribute A is not protected.

4.4 Goal representation

According to our scenario, the goal consists in joining two private relational tables. This goal is specified using the axioms 7 and 8 in which we use the following predicates:

- $join(T_1, T_2)$ is satisfied if both owners of T_1 and T_2 want to join their private tables T_1 and T_2.
- $joinAttribute(T_1, T_2, A_1)$ is satisfied if the tables T_1 and T_2 are joined over the attribute A_1.
- $join\_involved(T_1, A_1)$ is satisfied if the attribute A_1 of the table T_1 is involved in join operations. This predicate allows us to specify which are the attributes concerned by the joint.

Axiom 7 states that if the data owners of two tables T_1 and T_2 want to integrate their private data then eventually, there will exist a table T_j representing the join of T_1 and T_2. Axiom 8 states that the set of attributes of the joined table T_j is composed of the union of sets of join-involved attributes of private tables T_1 and T_2.

$$\forall T_1, T_2.\ \ join(T_1, T_2)\ \rightarrow\ \Diamond\ \big(\exists T_3\ \ JoinOf(T_1, T_2, T_3)\big) \quad (7)$$

$$\forall T_1, T_2, T_j, A.\ \ JoinOf(T_1, T_2, T_j)\ \wedge\ \big(join\_involved(T_1, A) \vee \\ join\_involved(T_2, A)\big) \rightarrow\ attribute\_of(T_j, A) \quad (8)$$

5 Security policy specification

The policy to be deployed is composed of a set of abstract-level constraints. Using these constraints, data owners will be able to model in a quite simple and powerful way, their security and utility requirements. In this section, we present different kinds of constraints: security constraint and utility constraint. We present for each kind of constraint the abstract-level representation and their corresponding transformation to the concrete level.

5.1 Security Constraint

Confidentiality Constraint: Using confidentiality constraint, a data owner will be able to require that the values assumed by some attributes are sensitive and therefore must be protected. For this purpose, we define the two-places predicate $SAttributeOf$. The formula $SAttributeOf(t, a)$ means that "The attribute a of the table t is a sensitive attribute". A confidentiality constraint is transformed to the concrete level using the following rule:

$$\begin{aligned} \forall A, T.\ SAttributeOf(T, A) \rightarrow \Box\ [\forall O, R, V. recordOf(T, R) \wedge \\ valueOf(R, A, V)\ \wedge\ \neg belongs(O, T)\ \rightarrow\ \neg knows(O, V)] \end{aligned} \tag{9}$$

Anonymization constraints: Using this kind of constraints, a data owner will be able to require the prevention of identity disclosure by protecting personal identifiers. We define the one-place predicate $withoutIDDisclosure$. Thus, the formula $withoutIDDisclosure(t)$ is to be read "Prevent identity disclosure in the table t". An Anonymization constraint is transformed to the concrete level using the following rule:

$$\begin{aligned} \forall T.\ withoutIDDisclosure(T) \rightarrow \Box\ \big(\forall A, O, R, V.\ IDAttributeOf(T, A) \\ \wedge\ recordOf(T, R)\ \wedge valueOf(R, A, V)\ \wedge\ \neg belongs(O, T)\ \rightarrow\ \neg knows(O, V)\big) \end{aligned} \tag{10}$$

5.2 Utility Constraint

Confidentiality and privacy protection is offered at the expense of data utility. Utility constraint gives the ability to a data owner to require that particular properties on data must be respected. The violation of these properties makes the data useless. As we work with relational databases, utility requirements are properties allowing the data owner to execute certain kind of queries over the protected data. These utility requirements can be classified into four classes.
Equality check requirements. With this kind of requirements, a data owner wants to be able to perform equality checks, which means that he or she wants to be able to perform selects with equality predicates, equality joins, etc.
Order check requirements. A data owner can use this kind of requirement in order to perform order check, which means that he or she wants to have the ability to execute range queries, order joins, ORDER BY, MIN, MAX, etc.
Computational requirements. With this kind of requirements, a data owner wants to have the ability to perform computation over encrypted data, which means the ability to execute queries with SUM, AVG, etc.
Keyword search requirements. Using keyword search requirements, a data owner wants to have the ability to perform keyword based search over the encrypted data (e.g, to check if a word exists in an encrypted text).

To be able to express these different kinds of utility requirements, we define the one place predicate $utility\_requirement()$. Then, an utility constraint defined

over the attribute A can be expressed by the axiom 11, which is to be read: "the ability to perform the utility requirement U over the attribute A".

$$utility\_requirement(req) \ \wedge \ provides(req, o) \tag{11}$$

6 Security mechanisms specification

Security policies are enforced through the application of security mechanisms which can be methods or approaches for supporting the requirements of the security policies. Each security policy is specified using three groups of formulas: preconditions formulas, effects formulas, and properties formulas.

Preconditions. For each security mechanism, preconditions are represented by a set of formulas which are necessary conditions under which the security mechanism can be applied. We define the two-places predicated $is\_applicable$. The formula $is\_applicable(M, O)$ is to be read "the mechanism M can be applied over the object O", O can be a table, an attribute, or a value. Preconditions of a security mechanism M are specified using a formula of the following form:

$$\Box \ (is\_applicable(M, O) \ \rightarrow \ \Delta_M) \tag{12}$$

Where Δ_M represents necessary conditions for the applicability of the mechanism M. A formula of the form 12 is to be read "At any state of the system, M can be applied if the preconditions Δ_M hold".

Effects. Effects of the application of a mechanism M that transits the system from a state w_i to a state w_j are modifications applied to the system during this transition. We use the two-places predicate $apply(M, O)$ to say that the mechanism M is applied over the object O. For a mechanism M, effects are represented by a set of formulas Σ_M such that:

$$\Phi(w_i, \ apply(M, O)) = w_j \ \rightarrow \ (w_j \ \models \ \Sigma_M) \tag{13}$$

Axiom 13 states that if the application of the mechanism M over the object O transits the system from a state w_i to a state w_j, therefore the set of effects Σ_M of the application of the mechanism M is satisfied on the state w_j.

Properties. The set of security and utility properties $P_1, \cdots, P_n$ that can be derived from the effects of the mechanism application.

$$\Sigma_M \rightarrow \bigwedge_{i=1}^{n} P_i \tag{14}$$

In our approach, security policies are composed mainly of confidentiality constraints and anonymization constraints. In the next section, we specify using the three previously presented groups of formulas (preconditions, effects, and properties) the set of security mechanisms that can be used to enforce the security policy. We classify these security mechanisms into encryption-based mechanisms and anonymization-based mechanisms.

6.1 Encryption-based mechanism specification

Encryption-based security mechanism can be classified using two main factors: the security properties they offer and the level of security they provide (e.g, the amount of information revealed about the encrypted data). Encryption-based security mechanisms are to be applied over an attribute A if the following preconditions hold: (1) the attribute A is considered sensitive, (2) the attribute A is involved in the joint table. This can be specified as follows:

$$\begin{aligned}\Box\ \big[\forall M, A.\ \ enc\_based\_mechanism(M) \wedge is\_applicable(M, A)\ \rightarrow \\ \exists T.\ \ SAttributeOf(T, A)\ \wedge join\_involved(T, A)\big]\end{aligned} \tag{15}$$

The effects of the application of encryption-based mechanisms are specified using the following axiom:

$$\begin{aligned}\forall M, A, T, K.\ \ enc\_based\_mechanism(M) \wedge attribute\_of(T, A)\ \wedge \\ enc\_key(K)\ \wedge\ apply(M, A) \rightarrow\ encrypted(T, A, K)\end{aligned} \tag{16}$$

Once we have defined the above axiom describing the effects of encryption-based mechanisms, we can specify the conditions under which an encryption-based mechanism can protect the values of an attribute. Obliviously, the values of an attribute over which an encryption-based mechanism is applied are protected from unauthorized data owners if those data owners have no knowledge about the used encrypted key (axiom 17). An attribute is protected from an unauthorized data owner means that this data owner has no knowledge about the values of the protected attribute (axiom 18).

$$\begin{aligned}\forall A, T, K, O.\ \ enc\_key(K) \wedge encrypted(T, A, K) \wedge \neg knows(O, K) \rightarrow \\ protected(T, A, O)\end{aligned} \tag{17}$$

$$\begin{aligned}\forall A, T, O, R, V.\ \ protected(T, A, O) \wedge recordOf(T, R) \wedge \\ valueOf(R, A, V)\ \rightarrow\ \neg knows(O, V)\end{aligned} \tag{18}$$

Encryption-based mechanisms can be classified using the security properties they offer into four categories: deterministic encryption based mechanisms, order-preserving encryption based mechanisms, homomorphic encryption based mechanisms, and searchable encryption based mechanisms. For each of these four categories, we formalize the security and utility properties that characterize them.
Deterministic encryption based mechanisms: Deterministic encryption based mechanisms allow logarithmic time equality check over encrypted data. This means that it can perform select queries with equality predicates, equality joins, etc. Deterministic encryption based mechanisms cannot achieve the classical notions of security of probabilistic encryption because it leaks which encrypted values correspond to the same plaintext value. Therefore, each attribute over which a deterministic encryption based mechanism is applied will

have the deterministic (det) security level (axiom 19).

$$\forall M, A.\ \ det\_enc\_mechanism(M) \wedge\ apply(M, A) \rightarrow \\ provides(equality\_check, A) \wedge sec\_level(A, det) \tag{19}$$

Order-preserving encryption based mechanisms: Order preserving symmetric encryption (OPE) mechanisms are based on deterministic symmetric encryption schemes which produce encrypted values that preserve numerical ordering of the plaintext values. OPE mechanisms are weaker than deterministic encryption based mechanisms as they leak the order between plaintext values. Based on this fact, each attribute over which an OPE mechanism is applied will have the order-preserving (ope) security level (axiom 20).

$$\forall M, A.\ \ ope\_mechanism(M) \wedge\ apply(M, A) \rightarrow\ sec\_level(A, ope) \wedge \\ provides(equality\_check, A)\ \wedge\ provides(order\_check, A) \tag{20}$$

Homomorphic encryption based mechanisms: Homomorphic encryption mechanisms are based on secure probabilistic encryption schemes which enable to perform computation over encrypted data. For efficiency, we suppose that we will use mechanisms based on partially homomorphic encryption as fully homomorphic encryption schemes have a long way to go before they can be used in practice [13]. In our approach, we will use mechanisms based on Paillier cryptosystem [14] to support summation. Paillier cryptosystem provides indistinguishability under an adaptive chosen-plaintext attack (IND-CPA). Therefore, each attribute over which an Homomorphic encryption based mechanisms is applied will have the probabilistic (prob) security level (axiom 21).

$$\forall M, A.\ \ hom\_mechanism(M) \wedge\ apply(M, A) \rightarrow \\ sec\_level(A, prob)\ \wedge provides(addition, A) \tag{21}$$

Searchable encryption based mechanisms: Searchable encryption mechanisms allow searching for keywords on an encrypted database without revealing the keyword. Therefore, this kind of mechanisms can be used to perform operations such as SQL's LIKE operator. We suppose that we will use the SEARCH mechanism defined in [18] which is proved to be nearly as secure as a probabilistic encryption. Based on this fact, each attribute over which the SEARCH mechanism is applied will have the probabilistic (prob) security level. Properties of the SEARCH-based mechanism is specified as follows:

$$\forall A.\ \ searchable\_enc\_mechanism(SEARCH) \wedge\ apply(SEARCH, A) \rightarrow \\ sec\_level(A, prob)\ \wedge\ provides(keywork\_search, A) \tag{22}$$

6.2 Anonymization-based mechanism specification

The anonymization technique aims to prevent identity disclosure by protecting personal identifier. To meet this requirement, we use the existing anonymization

approach *k-anonymity* [16] in which identifier attributes values are removed from the private table (axiom 25). However, a *Quasi-identifier* attribute value in the released table may lead to infer the value of removed identifier attributes (axiom 23). Therefore, *Quasi-identifier* attributes values are generalized (axiom 26) in such a way that removed identifier attributes values cannot be recovered.

$$\begin{aligned} \forall O, T_1, A_1, V_1, R.\ \ IDAttributeOf(T_1, A_1) \wedge\ recordOf(T_1, R) \wedge \\ valueOf(R, A_1, V_1)\ \wedge\ knows(O, V_1) \leftrightarrow belongs(T_1, O) \vee \\ \Big[\exists T_2, T_3.\ belongs(O, T_2)\ \wedge\ joinOf(T_1, T_2, T_3) \wedge \big(join\_involved(T_1, A_1) \\ \vee\ (\exists A_2.\ QIDAttributeOf(T_1, A_2)\ \wedge\ \neg\ anonymized(T_1, A_2))\big)\Big] \end{aligned} \tag{23}$$

Anonymization mechanism is applied over a table T if and only if the table T contains at least an identifier attribute or a quasi-identifier attribute. These preconditions is specified as follows:

$$\begin{aligned} \Box\big[\forall T.\ \ is\_applicable(kanonymity, T) \rightarrow \exists A.(IDAttributeOf(T, A) \vee \\ QIDAttributeOf(T, A)) \wedge \neg\ encrypted(T, A)\big] \end{aligned} \tag{24}$$

Effects of the application of Anonymization mechanism over a table T are specified using the following axioms:

$$\forall A.\ \ IDAttributeOf(T, A) \rightarrow \neg\ join\_involved(T, A) \tag{25}$$

$$\forall A. QIDAttributeOf(T, A) \wedge join\_involved(T, A) \rightarrow anonymized(T, A) \tag{26}$$

The use of anonymization mechanisms such as k-anonymity offers the data owner the ability to ensure the prevention of identities disclosure by protecting personal identifiers at the same time supporting data analysis (e.g., data mining). In terms of security, anonymization based mechanisms are weaker than encryption based mechanisms as anonymized data can sometimes be re-identified with particular individuals by using *homogeneity Attack* or *Background Knowledge Attack* [12]. Based on this fact, each protected identifier attribute will have the anonymization (anonym) security level. Properties of anonymization-based mechanism is specified as follows:

$$\begin{aligned} \forall M, T. anon\_mechanism(M) \wedge apply(M, T) \rightarrow \\ provides(data\_analysis, T) \wedge \big(\forall A, O. IDAttributeOf(T, A) \wedge \neg \\ belongs(T, O) \rightarrow protected(T, A, O) \wedge sec\_level(anonym, A)\big) \end{aligned} \tag{27}$$

In order to compare different security levels provided by previously presented security mechanisms, we define the transitive predicate *more_secure_than*. The formula $more\_secure\_than(l_1, l_2)$ is to be read: "the level l_1 is more secure than the level l_2". A mechanism M_1 is more secure than a mechanism M_2 if the application of M_1 leaks less information about sensitive data than the application

of M_2. Therefore, based on the amount of leaked information, we define a rule (axiom 28) stating that: probabilistic security level *prob* is more secure than deterministic security level *det*, the deterministic security level *det* is more secure than the order-preserving security level *ope*, and the order-preserving security level *ope* is more secure than the anonymization security level *anonym*.

$$\begin{aligned} more\_secure\_than(prob, det) \wedge more\_secure\_than(det, ope) \\ \wedge more\_secure\_than(ope, anonym) \end{aligned} \tag{28}$$

7 Choosing the right mechanisms

The right mechanism or combination of mechanisms is the one that fits in the best way the sets of security and utility constraints. As we have seen in the previous section, each security mechanism offers a different level of protection and a different kind of utility properties. The main challenge then is to choose the best mechanisms allowing the satisfaction of the chosen goal while enforcing the defined security policy. In our scenario, security issues come when data is joined. Before applying the joint operation, security constraints are satisfied. This hypothesis is also suitable in the general case of data outsourcing. As we can consider that, since the data is not outsourced, there is no security issue to worry about. Based on this, we defined several steps allowing the selection of the mechanisms to be applied.

7.1 First step: Satisfy the chosen goal.

We look for the mechanisms that satisfy the chosen goal. For instance, in our scenario, we look for the suitable join method that can join the data of the two private tables. Formally speaking, a mechanism M_g satisfies a goal G if from the specification of our system Σ and the effects of the mechanism Σ_{M_g} we are able to deduce the set of formula representing the goal G (29).

$$\Sigma \cup \Sigma_{M_g} \vdash \Sigma_G \tag{29}$$

7.2 Second step: Violated security constraints.

After getting the set of mechanisms $\mathcal{M}$ that can be applied to achieve the chosen goal, we start looking for the set of violated security and utility constraints for each mechanism $M_g \in \mathcal{M}$. A constraint C is violated while the chosen goal G is satisfied if from the specification of our system Σ, the effects Σ_{M_g} of the mechanism M_g and the set of formulas Σ_C representing the constraint C we can deduce a logic contradiction. This is can be formally represented as follows:

$$\Sigma \cup \Sigma_{M_G} \cup \Sigma_C \vdash \bot \tag{30}$$

Obviously, our toolbox may contain several mechanisms that can satisfy the chosen goal. In that case, we should be able to choose the best one.

Definition 4 (Best goal satisfier). *Given the set of mechanisms* $\mathcal{M} = \{M_1, \cdots, M_n\}$ *that can be used to satisfy the defined goal* G*. Let* $\mathcal{C}_i$ *be the set of violated constraints while applying the mechanism* M_i*. A mechanism* M_j *is a best goal satisfier if the following condition holds:*

$$\forall i \in \{1, \cdots, n\}. \ \ |\mathcal{C}_j| \leq |\mathcal{C}_i|, \textit{where } |\mathcal{C}_i| \textit{ is the cardinality of } \mathcal{C}_i.$$

7.3 Third step: Satisfying the violated constraints.

Once we get the *best goal satisfier* M_{bgs} for a defined goal G and the corresponding set of violated security and utility constraints $\mathcal{C}$, the challenge then is to, for each violated security constraint, looking for the properties that can satisfy that constraint. Formally speaking, a set of l properties $\mathcal{P} = \{P_1, \cdots, P_l\}$ satisfies a security constraint C in a state of the system if from: (1) the sets of formulas $\Sigma_{P_1}, \cdots, \Sigma_{P_l}$ representing respectively the specification of the properties $P_1, \cdots, P_l$, (2) the set of formulas Σ representing the system specification, and (3) the set of formulas $\Sigma_{M_{bgs}}$ representing the effects of M_{bgs}, we are able to deduce the set of formulas Σ_C representing the specification of the constraint C. This is can be formalized as follows:

$$\bigwedge_{i=1}^{l} \Sigma_{P_i} \ \cup \ \Sigma \ \cup \ \Sigma_{M_{bgs}} \ \vdash \ \Sigma_C \tag{31}$$

Informally, 31 means that if the set of security properties $\mathcal{P}$ is provided, the application of the M_{bgs} will not violate the security constraint C.

7.4 Fourth step: Choosing the best security mechanisms.

The previous steps allow us to select the *best goal satisfier* M_{bgs} that can satisfy the goal G, the corresponding set of violated security and utility constraints $\mathcal{C}$, and for each security constraint $C_i \in \mathcal{C}$, we select the set of properties $\mathcal{P}_i$ that can satisfy C_i when applying the M_{bgs}. Now, based on those properties, the main goal is to select from our toolbox, the *best combination of security mechanisms* that can *usefully satisfy* each violated constraint in $\mathcal{C}$.

Definition 5 (Useful satisfaction). *Given a violated security constraint* C *defined over an object (table or attribute)* Ob*, the set of security properties* $\mathcal{P}$ *that satisfy* C*, and the set of utility constraint* $\mathcal{U}_{Ob}$ *defined over the object* Ob*. A combination of mechanisms* MC *usefully satisfy the constraint* C *if:*

$$\Sigma \ \cup \ \{ \ \bigwedge_{M \in MC} apply(M, Ob)\} \ \models \Big(\ \bigwedge_{P \in \mathcal{P}} P \ \bigwedge_{U \in \mathcal{U}_{Ob}} provides(U, Ob) \ \Big) \tag{32}$$

Definition 6. *Given a violated security constraint C, the set of properties $\mathcal{P}$ that satisfy C, and a combination of mechanisms $CM = \{M_1, \cdots, M_n\}$ that usefully satisfy C. The security level l provided by the combination of mechanisms CM is the lowest level provided by the application of set of mechanisms $M_1, \cdots, M_n$.*

$$\Sigma \cup \{\forall Ob. \bigwedge_{i=1}^{n} apply(M, Ob)\} \models \\ \big(\forall l'. \ sec\_level(l, Ob) \wedge sec\_level(l', Ob) \rightarrow more\_secure\_than(l', l)\big) \tag{33}$$

Definition 7 (Best combination of mechanisms). *Given a violated security constraint C and the set of properties $\mathcal{P}$ that can satisfy C. Suppose that we find several combinations of security mechanisms $CM_1, \cdots, CM_n$ that provide the set of properties $\mathcal{P}$. Suppose that the set of combinations of security mechanisms $CM_1, \cdots, CM_n$ provides respectively the set of security levels $l_1, \cdots, l_n$. The combination of mechanisms CM_i is the best combination of mechanisms if it has the highest provided security level. For the combinations of mechanisms that provide the same security level, we choose the one that involves the minimal number of security mechanisms. This is can be specified as follows:*

$$\bigwedge_{i=1}^{n} \Big(more\_secure\_than(l_i, l_j) \ \vee (l_i = l_j \ \wedge \ |CM_i| < |CM_j|)\Big) \tag{34}$$

8 Best mechanisms selection

In this section, we demonstrate how to to select the best combination of mechanisms to satisfy defined security policies using different steps presented in the previous section. Due to the lack of space, proofs of this demonstration which can be found in [2] will be omitted here. Consider two data owners O_1 and O_2 which store respectively two private tables $T_1(SSN, Age, Adress, Balance)$ and $T_2(SSN, Job, ZIP, Salary)$. They want to integrate data stored in both tables. In one side, O_1 defined a policy P_1 composed of two security constraints $SC_{1,1} = \{withoutIDDisclosure(T_1)\}$ and $SC_{2,1} = \{SAttributeOf(T_1, Balance)\}$ and two utility constraints $UC_{1,1} = \{provides(equality, Balance)\}$ and $UC_{2,1} = \{provides(addition, Balance)\}$. O_1 specifies that the attribute SSN is an *identifier* attribute and that the attributes Age and $Address$ are *quasi-identifier* attributes. In another side, O_2 defines a policy P_2 composed of the security constraint $SC_{1,2} = \{withoutIDDisclosure(T_2)\}$. O_2 specifies that the attribute SSN is an *identifier* attribute and that the attributes Job and Zip are *quasi-identifier* attributes. Suppose that all attributes in T_1 and T_2 are involved in the join and that our toolbox is composed of the set of security mechanisms presented in 6.1 and 6.2, and two other mechanisms, *rel_join* and *tds* representing respectively the relational join operation and the *top-down specialization* mechanism [7]. Axiom 36 in [2] specifies the effects Σ_{rel_join} of the application of

rel_join mechanism. Axiom 37 in [2] describes the effects Σ_{tds} of the application of the mechanism *tds*. The first step to select the best combination of mechanisms allowing to satisfy P_1 and P_2 while achieving the chosen goal consists in selecting the set of mechanisms to achieve the chosen goal. According to 29, the mechanism *rel_join* and *tds* can be applied to satisfy the jointure of the private tables T_1 and T_2 (See Proof 1 in [2]).

After we select the set of mechanisms to satisfy the goal, we choose the *best goal satisfier* from this set of mechanisms. In this demonstration, according to Definition 4, the *tds* mechanism represents the *best goal satisfier* to join the private tables T_1 and T_2 as it violates only $SC_{2,1}$ (See Proof 2 in [2]. The next step consists in finding the set of security properties to satisfy the violated security constraints that rose from the application of the *best goal satisfier*. When provided for the attribute *Balance*, the protection property *protected* can satisfy the confidentiality constraint $SC_{2,1}$ even when the *tds* mechanism is applied (See Proof 3 in [2]. Next, we choose from our toolbox the combination of mechanisms that can *usefully satisfy* the security constraint $SC_{2,1}$. Two combinations of mechanisms can *usefully satisfy* the security constraint $SC_{2,1}$: (1) combines an order-preserving encryption based mechanism and an homomorphic encryption based mechanism, and (2) combines a deterministic encryption based mechanism and an homomorphic encryption based mechanism (See Proof 4 in [2]. The final step consists in choosing the *best combination of mechanisms* that can *usefully satisfy* the security constraint $SC_{2,1}$ which is (2) (See Proof 5 in [2]. In conclusion, we can say that the application of the combination of mechanisms (2) before the application of *tds* mechanism allows us to enforce defined security policies P_1 and P_2 while joining the two tables T_1 and T_2.

9 Conclusion

We defined a well-founded language to select, from a toolbox containing a set of security mechanisms, the best combination of security mechanisms allowing the enforcement security and utility requirements for outsourced data. Our approach can be improved by detecting the incompatibilities and conflicts between security mechanisms to be able to decide which mechanisms can be applied together without losing provided utility requirements.

Acknowledgments. This work has received a French government support granted to the CominLabs excellence laboratory and managed by the National Research Agency in the "Investing for the Future" program under reference ANR-10-LABX-07-01, and to the Frag&Tag project and managed by the Dual Innovation Support Scheme (RAPID) under convention Nº 132906023

References

1. Bkakria, A., Cuppens, F., Cuppens-Boulahia, N., Fernandez, J.M., Gross-Amblard, D.: Preserving multi-relational outsourced databases confidentiality using fragmen-

tation and encryption. Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA) 4(2), 39–62 (6 2013)
2. Bkakria, A., Cuppens, F., Cuppens-Boulahia, N., Gross-Amblard, D.: `https://portail.telecom-bretagne.eu/publi/public/fic_download.jsp?id=30178`
3. Blanchet, B.: An efficient cryptographic protocol verifier based on prolog rules. In: CSFW. pp. 82–96. IEEE Computer Society (2001)
4. Blanchet, B.: Automatic proof of strong secrecy for security protocols. In: IEEE Symposium on Security and Privacy. pp. 86–. IEEE Computer Society (2004)
5. Boldyreva, A., Chenette, N., Lee, Y., O'Neill, A.: Order-preserving symmetric encryption. In: Joux [11], pp. 224–241
6. Ciriani, V., di Vimercati, S.D.C., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Fragmentation and encryption to enforce privacy in data storage. In: Biskup, J., Lopez, J. (eds.) ESORICS. Lecture Notes in Computer Science, vol. 4734, pp. 171–186. Springer (2007)
7. Fung, B.C.M., Wang, K., Yu, P.S.: Top-down specialization for information and privacy preservation. In: Aberer, K., Franklin, M.J., Nishio, S. (eds.) ICDE. pp. 205–216. IEEE Computer Society (2005)
8. Gabbay, D., Pnueli, A., Shelah, S., Stavi, J.: On the temporal analysis of fairness. In: Proceedings of the 7th ACM SIGPLAN-SIGACT symposium on Principles of programming languages. pp. 163–173. POPL '80, ACM, New York, NY, USA (1980)
9. Hacigümüs, H., Iyer, B.R., Li, C., Mehrotra, S.: Executing sql over encrypted data in the database-service-provider model. In: SIGMOD Conference. pp. 216–227. ACM (2002)
10. Hore, B., Mehrotra, S., Tsudik, G.: A privacy-preserving index for range queries. In: Nascimento, M.A., Özsu, M.T., Kossmann, D., Miller, R.J., Blakeley, J.A., Schiefer, K.B. (eds.) VLDB. pp. 720–731. Morgan Kaufmann (2004)
11. Joux, A. (ed.): Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings, Lecture Notes in Computer Science, vol. 5479. Springer (2009)
12. Machanavajjhala, A., Gehrke, J., Kifer, D., Venkitasubramaniam, M.: l-diversity: Privacy beyond k-anonymity. In: Liu, L., Reuter, A., Whang, K.Y., Zhang, J. (eds.) ICDE. p. 24. IEEE Computer Society (2006)
13. Naehrig, M., Lauter, K., Vaikuntanathan, V.: Can homomorphic encryption be practical? In: Cachin, C., Ristenpart, T. (eds.) CCSW. pp. 113–124. ACM (2011)
14. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT. Lecture Notes in Computer Science, vol. 1592, pp. 223–238. Springer (1999)
15. Popa, R.A., Redfield, C.M.S., Zeldovich, N., Balakrishnan, H.: Cryptdb: Protecting confidentiality with encrypted query processing. In: In SOSP (2011)
16. Samarati, P., Sweeney, L.: Generalizing data to provide anonymity when disclosing information (abstract). In: Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems. pp. 188–. PODS '98, ACM, New York, NY, USA (1998)
17. Song, D.X., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In: IEEE Symposium on Security and Privacy. pp. 44–55. IEEE Computer Society (2000)
18. di Vimercati, S.D.C., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: A data outsourcing architecture combining cryptography and access control. In: Ning, P., Atluri, V. (eds.) CSAW. pp. 63–69. ACM (2007)