Journal of Computer Security 29 (2021) 613–650
DOI 10.3233/JCS-200007
IOS Press

Privacy-preserving policy evaluation in multi-party access control

Mina Sheikhalishahi [*], Ischa Stork and Nicola Zannone
Eindhoven University of Technology, Eindhoven, The Netherlands
E-mails: m.sheikhalishahi@tue.nl, i.stork@student.tue.nl, n.zannone@tue.nl

Abstract. Recent years have seen an increasing popularity of online collaborative systems like social networks and web-based collaboration platforms. Collaborative systems typically offer their users a digital environment in which they can work together and share resources and information. These resources and information might be sensitive and, thus, they should be protected from unauthorized accesses. Multi-party access control is emerging as a new paradigm for the protection of co-owned and co-managed resources, where the policies of all users involved in the management of a resource should be accounted for collaborative decision making. Existing approaches, however, only focus on the jointly protection of resources and do not address the protection of the individual user policies themselves, whose disclosure might leak sensitive information. In this work, we propose a privacy-preserving mechanism for the evaluation of multi-party access control policies, which preserves the confidentiality of user policies while remaining capable of making collaborative decisions. To this end, we design secure computation protocols for the evaluation of policies in protected form against an access query and realize such protocols using two privacy-preserving techniques, namely Homomorphic Encryption and Secure Functional Evaluation. We show the practical feasibility of our mechanism in terms of computation and communication costs through an experimental evaluation.

Keywords: Multi-party access control, collaborative system, privacy-preserving computation

1. Introduction

The widespread availability of the Internet has led to a significant growth in the use of online collaborative systems and platforms. Such systems generally offer their users the means for digital interactions and for the jointly creation and management of co-owned resources. These resources, however, can be sensitive and, thus, they should be protected from unauthorized usages by considering the access requirements of all co-owners. Multi-party access control is emerging with the aim of enabling collaborative governance of co-owned resources [45], thus overcoming the limitations of traditional access control models, which are based on the assumption that resources are governed by a single entity. Several approaches to multi-party access control have been proposed in the last years [14,18,35,48]. These approaches provide a means for collaborative decision making by reconciling the conflicts that can arise from the evaluation of the policies provided by the entities involved in the management of co-owned resources.

However, existing multi-party access control models do not account for the protection of the policies themselves, whose disclosure can leak sensitive information as well [34,54,60]. For instance:

- Collaborative commercial agreements often contain partners' policies specifying with who and under what conditions co-owned resources and assets can be shared. While each partner expects its

[*]Corresponding author. E-mail: m.sheikhalishahi@tue.nl.

policies to be enforced [14], the policies might contain confidential information about a company's business and commercial relations and, thus, their disclosure can provide insights on the company's business strategies, which can be used by competitors (possibly in the coalition) to "*evaluate sales coverage, modify compensation plans, renegotiate terms and conditions, adjust compliance policies, build advanced segmentation categories and uncover hidden supply chain risk*" [20].

- Contents uploaded by a user on her/his profile (or others' profile) in an online social network might refer to multiple users, *e.g.*, a photo shared in Facebook in which several users are tagged. The collaborative management of the contents requires the social network to consider the privacy preferences (specifying who is permitted to access the co-owned contents) of individual users. The disclosure of the users' privacy preferences might reveal the users' interpersonal relationship and reduce the users' willingness in sharing new contents [51].
- In critical-missions, *e.g.* in the military and counter-intelligence domain, international cooperation is becoming a key factor to ensure the success of the mission. In this context, intelligence is often collected from heterogeneous sources and fused to enable situation awareness and, thus, take the proper actions to handle potential threats. Information sources, however, can be under the control of different authorities. Due to the high sensitivity of data, each authority might want to enforce specific constraints on the access and usage of its data. For fused data, this implies that possibly conflicting access requirements from different parties should be accounted for. While there exist solutions that allow the collaborative specification of access control policies for fuse data [5,15], these solutions typically require the parties' individual policies to be disclosed in clear. However, the policies themselves might contain classified information and, hence, their disclosure can raise security concerns.

The aforementioned scenarios highlight the need of protecting not only the resources but also the security policies employed for their protection as the disclosure of those policies might leak sensitive information about the entities that contributed to their definition. Therefore, collaborative systems should be equipped with an access control mechanism that preserves the confidentiality of individual user policies while remaining capable of making collaborative access decisions. In this light, we assume a multi-owner-single-user setting where multiple entities are responsible for the security of co-owned resources; specifically, each co-owners defines policies stating their access constraints for the co-owned resources and these policies are combined into a single global policy (hereafter, called *multi-party policy*) for collaborative access decision making. The evaluation of the multi-party policy against an access request should not leak information about the policies defined by the single co-owners.

To address this issue, in previous work [51] we proposed a privacy-preserving multi-party access control framework, in which users provide their policies in private form and policy evaluation is performed over private inputs. In particular, we designed *secure computation* protocols for the evaluation of multi-party policies that preserve the confidentiality of the user policies forming the multi-party access control policies. However, the framework in [51] only allows the evaluation of policies expressed in a simple identity-based access control model and uses a three-valued decision set (*permit*, *deny*, and *not-applicable*) that is not able to capture the complexity of existing access control standards like XACML [44].

This paper extends our previous work to enable the secure evaluation of multi-party access control policies expressed in standardized Attribute-Based Access Control (ABAC) policy languages like XACML while protecting the confidentiality of user policies. Compared to the policies considered in our previous work, ABAC policies comprise a target that determines their applicability. The target of a policy essentially consists of a set of conditions defined over subject, resource, action and environment attributes that must be met for the policy to apply to a given query. In addition, standardized ABAC policy languages like XACML often rely on multi-valued decision sets that extend the three-valued decision set by accounting for situations for which the access control mechanism cannot make a definitive decision due, for instance, to missing attributes [38].[1] The privacy-preserving evaluation of such complex ABAC policies requires the definition of new protocols for target evaluation[2] and for the evaluation of composite policies. The design of such protocols, however, is not trivial and requires addressing a number of challenges to ensure their practical feasibility and prevent information leakages. The main challenge lies in identifying a suitable policy representation as secure computation protocols defined on a direct encoding of complex ABAC policies are inefficient due to the complex operations that this encoding requires performing over private input. Therefore, we investigated policy representations that enable the design of efficient secure computation protocols for target and policy evaluation. In particular, we adopted a Boolean encoding of ABAC policies, which by relying on AND, OR and negation operators provides an efficient structure for secure target and policy evaluation [51]. In addition, this Boolean encoding allows us to devise mechanisms to mask the size of multi-valued decisions, which if not protected, might leak information about the underlying user policies.

We investigate the realization of the proposed protocols using two alternative privacy preserving techniques, named Homomorphic Encryption (HE) [46] and Secure Functional Evaluation (SFE) [13]. Homomorphic Encryption and Secure Functional Evaluation are two well-known and established privacy-preserving techniques, which provide the cryptographic building blocks necessary for the realization of the proposed protocols. However, these techniques are usually computationally expensive [17,39] and, thus, they might be not practical in the real-world systems, hindering their use for the development of a multi-party access control framework. To this end, we have investigated optimizations for the realization of the proposed protocols in Homomorphic Encryption and Secure Functional Evaluation and evaluated their computation and communication costs through an experimental evaluation. The results show that the SFE-based protocols outperform the HE-based protocols both in terms of both computation and communication costs and provide a basis for the effective realization of privacy-preserving mechanisms for multi-party access control. We also discuss the security of the implementation of the proposed protocols in the presence of a semi-honest adversary, which guarantees that the policy evaluation does not leak any unintended information.

The contribution of this work can be summarized as follows:

- We design secure computation protocols that enable the secure evaluation of multi-party access control policies expressed in standardized ABAC policy languages like XACML while protecting the confidentiality of user policies. Based on a Boolean encoding of complex ABAC policies, we propose efficient secure computation protocols for target and policy evaluation. We also propose a new approach to hide the size of multi-valued decisions resulting from the evaluation of such policy, thus preventing the leakage of information about the individual user policies.
- We realize the proposed protocols using two well-known and largely used privacy-preserving techniques, namely Homomorphic Encryption and Secure Functional Evaluation and investigate further

[1]In this work, we use a seven-valued set $\mathcal{D}_7$ that extends the three-valued decision set $\mathcal{D}_3$ by considering any non-empty subset of $\mathcal{D}_3$, *i.e.* $\mathcal{D}_7 = \mathcal{P}(\mathcal{D}_3) \setminus \emptyset$. This decision set resembles the decision set supported by XACML where *Indeterminate* decisions in XACML are represented by non-singleton decisions in $\mathcal{D}_7$ [10,37].

[2]In our previous work [51], the applicability of policies to a given query was simply computed using the private set intersection protocol to check if the requester belongs to the set of authorized users while a target of an ABAC policy can comprise equality, inequality and inclusion conditions.

optimizations of the protocols based on the employed privacy-preserving techniques to reduce the computation and communication costs of the implementation.

- We demonstrate the security of our framework for the privacy-preserving evaluation of multi-party access control policies as well as of the underlying secure computation protocols and their composition.
- We demonstrate the practical feasibility of the proposed framework in terms of computational and communication costs through an experimental evaluation. Our experiments show that the evaluation of multi-party access control policies using SFE-based protocols considerably outperforms the use of HE-based protocols. In particular, evaluating large policies (of size 50) against queries consisting of 10 attributes using SFE-based protocols requires less than 2 seconds, thus showing their practical feasibility for the evaluation of multi-party access control policies.

The remainder of the paper is structured as follows. The next section introduces the background knowledge used in this work. Section 3 provides an overview of our framework for privacy-preserving multi-party access control. Section 4 presents our secure computation protocols for the evaluation of multi-party access control policies and Section 5 provides their implementation in Homomorphic Encryption and Secure Function Evaluation. Section 6 discusses the security of the proposed protocols. An experimental evaluation of their computation and communication costs is presented in Section 7. Finally, Section 8 discusses related work and Section 9 concludes the paper.

2. Preliminaries

This section introduces the policy language used for the specification of user and multi-party access control policies. We also present the building blocks used for the design and implementation of secure computation protocols for policy evaluation in two privacy-preserving techniques, namely Homomorphic Encryption and Secure Function Evaluation.

2.1. Policy specification and evaluation

For the specification of user and multi-party access control policies, we rely on an attribute-based access control (ABAC) policy language inspired by PTaCL [10,37], which provides an abstraction of the XACML policy language [44]. We first present an extension of the PTaCL syntax, which comprises two languages, one for targets, which is used to specify the applicability of a policy to a query, and another for policies, which is used to specify how policies are combined. Then, we present the semantics of target and policy evaluation.

ABAC Syntax: Let $\mathcal{A} = \{a_1, \dots, a_n\}$ be a finite set of attributes, where the domain of an attribute a is denoted by $\mathcal{V}_a$. The set of queries $\mathcal{Q}_\mathcal{A}$ is defined as $\mathcal{P}(\bigcup_{i=1}^{n} a_i \times \mathcal{V}_{a_i})$, and a query $q \in \mathcal{Q}_\mathcal{A}$ is the set of attribute name-value pairs:

$$q = \{(a_1, v_1), \dots, (a_k, v_k)\}$$

where $a_i \in \mathcal{A}$ and $v_i \in \mathcal{V}_{a_i}$. The set of attributes and values in q are denoted $\mathcal{A}_q = \{a_1, \dots, a_k\}$ and $\mathcal{V}_q = \{v_1, \dots, v_k\}$ respectively. Given an attribute $a_i \in \mathcal{A}$ and a query q, $\mathcal{V}_{a_i|q}$ denotes the set of a_i's values that appear in q.

Table 1

Operators on decision set $\mathcal{D}_3 = \{1, 0, \bot\}$

d_1	d_2	$\neg d_1$	$\sim d_1$	$d_1 \,\widetilde{\sqcap}\, d_2$	$d_1 \sqcap d_2$	$d_1 \vartriangle d_2$	$d_1 \,\widetilde{\sqcup}\, d_2$	$d_1 \sqcup d_2$	$d_1 \triangledown d_2$	$d_1 \rhd d_2$
1	1	0	1	1	1	1	1	1	1	1
1	0	0	1	0	0	0	1	1	1	1
1	$\bot$	0	1	$\bot$	$\bot$	1	1	$\bot$	1	1
0	1	1	0	0	0	0	1	1	1	0
0	0	1	0	0	0	0	0	0	0	0
0	$\bot$	1	0	0	$\bot$	0	$\bot$	$\bot$	0	0
$\bot$	1	$\bot$	0	$\bot$	$\bot$	1	1	$\bot$	1	1
$\bot$	0	$\bot$	0	0	$\bot$	0	$\bot$	$\bot$	0	0
$\bot$	$\bot$	$\bot$	0	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$

To determine the applicability of a policy to a query, we employ a target language, denoted by $\mathcal{T}_\mathcal{A}$, which extends the PTaCL target language to allow the specification of access constraints built using the set of binary predicates $\Phi = \{=, \neq, \leqslant, \geqslant\}$.[3] A target $t \in \mathcal{T}_\mathcal{A}$ is defined as:

$$t = a\,\phi\,v \mid op(t_1, \dots, t_n)$$

where $a\,\phi\,v$ is an *atomic target* with $a \in \mathcal{A}$, $v \in \mathcal{V}_a$ and $\phi \in \Phi$, and $op(t_1, \dots, t_n)$ is a *composite target* with op an n-ary three-valued logic operator. Here, we employ the combining operators proposed in PTaCL, which are presented in Table 1 (we only show how to combine two policies, but the semantics can be trivially extended to consider an arbitrary number of targets and policies). These operators represent combining algorithms largely used in ABAC languages. For instance, strong conjunction ($\widetilde{\sqcap}$) and strong disjunction ($\widetilde{\sqcup}$) resemble the operators used in XACML v.3 [44] for the evaluation of composite targets. The other operators encode policy conflict resolution strategies such as the XACML combining algorithms *permit-overrides* ($\triangledown$), *deny-overrides* ($\vartriangle$) and *first-applicable* ($\rhd$). We also consider the *negation* operator ($\neg$) and the *weakening* operator ($\sim$), which maps the *not-applicable* decision to *deny*. Note that the set of operators $\{\neg, \sim, \widetilde{\sqcup}\}$ is canonically complete [29], *i.e.* any three-valued logic operator can be constructed using these three operators.

PTaCL also provides a policy language $\mathcal{P}_\mathcal{A}$, where a policy $p \in \mathcal{P}_\mathcal{A}$ is defined as:

$$p \ = \ 1 \mid 0 \mid (t, p) \mid op(p_1, \dots, p_n)$$

where 1 and 0 represent the *permit* and *deny* decisions respectively, (t, p) is a *targeted policy*, and $op(p_1, \dots, p_n)$ is a *composite policy* with op an n-ary three-valued logic operator. Here again, we consider the operators defined in Table 1.

ABAC Semantics: Given the set of policies $\mathcal{P}_\mathcal{A}$, the set of queries $\mathcal{Q}_\mathcal{A}$, and the set of decisions $\mathcal{D}$, a policy evaluation function is a function $[\![\cdot]\!] : \mathcal{P}_\mathcal{A} \times \mathcal{Q}_\mathcal{A} \to \mathcal{D}$, such that for query q and policy p, $[\![p]\!](q)$ represents the decision of evaluating q against p.

To evaluate a query against a policy, we first need to determine whether the policy is applicable to the given query. Given a query, a target evaluates to a single value in $\mathcal{D}_3 = \{1, 0, \bot\}$, intuitively indicating if the target matches the query (1), if it does not match the query (0), or if the query does not contain the

[3]The PTaCL target language only supports the equality predicate.

attributes required to evaluate the applicability of the target ($\perp$), respectively. Formally, given a query q, the target evaluation function is defined as:

$$\llbracket \cdot \rrbracket_T : \mathcal{T}_\mathcal{A} \times \mathcal{Q}_\mathcal{A} \to \mathcal{D}_3$$

$$\llbracket a \, \phi \, v \rrbracket_T(q) = \begin{cases} 1 & \text{if } a \in \mathcal{A}_q, \exists v' \in \mathcal{V}_{a|q} \text{ s.t. } v' \, \phi \, v \\ 0 & \text{if } a \in \mathcal{A}_q, \nexists v' \in \mathcal{V}_{a|q} \text{ s.t. } v' \, \phi \, v \\ \perp & \text{if } a \notin \mathcal{A}_q \end{cases}$$

$$\llbracket op(t_1, \ldots, t_n) \rrbracket_T(q) = op\big(\llbracket t_1 \rrbracket_T(q), \ldots, \llbracket t_n \rrbracket_T(q)\big)$$

where op is an n-ary three-valued logic operator as defined in Table 1.

A policy is evaluated to decisions within $\mathcal{D}_7 = \mathbb{P}(\{1, 0, \perp\}) \setminus \emptyset$, where 1 and 0 indicate that access should be granted or denied respectively, and $\perp$ that the policy is not applicable to the given query. Non-singleton decisions are returned when the query does not provide the required information to evaluate a target. Intuitively, non-singleton decisions correspond to the *Indeterminate* decision in XACML [38]. It is worth mentioning that even though this set is syntactically equivalent to the one used for targets, the meaning of the values depends on whether it is used as a target or policy. Formally, the evaluation of policy p is given by the function:

$$\llbracket \cdot \rrbracket_P : \mathcal{P}_\mathcal{A} \times \mathcal{Q}_\mathcal{A} \to \mathcal{D}_7$$

$$\llbracket 1 \rrbracket_P(q) = \{1\}, \qquad \llbracket 0 \rrbracket_P(q) = \{0\}$$

$$\llbracket (t, p) \rrbracket_P = \begin{cases} \llbracket p \rrbracket_P(q) & \text{if } \llbracket t \rrbracket_T(q) = 1 \\ \{\perp\} & \text{if } \llbracket t \rrbracket_T(q) = 0 \\ \{\perp\} \cup \llbracket p \rrbracket_P(q) & \text{otherwise} \end{cases}$$

$$\llbracket op(p_1, \ldots, p_n) \rrbracket_P(q) = op^\uparrow\big(\llbracket p_1 \rrbracket_P(q), \ldots, \llbracket p_n \rrbracket_P(q)\big)$$

where given an operator $op : \mathcal{D}_3 \times \mathcal{D}_3 \to \mathcal{D}_3$ and non-empty sets $X, Y \subseteq \mathcal{D}_3$, $op^\uparrow : \mathcal{D}_7 \times \mathcal{D}_7 \to \mathcal{D}_7$ is defined as $op^\uparrow(X, Y) = \{op(x, y) \mid x \in X \wedge y \in Y\}$.

2.2. *Homomorphic encryption*

Homomorphic Encryption (*HE*) is a family of cryptographic schemes that enable computation over encrypted data. HE allows performing an operation on ciphertexts, such that the resulting ciphertext would decrypt to the same value that would have been obtained by performing the operation on the corresponding plaintexts. In this work, we employ an *additively* homomorphic cryptosystem, *e.g.* the *Paillier* cryptosystem [46], which preserves the result of the addition of two ciphertexts.

Let $\mathcal{E}_{p_k}(\cdot)$ and $\mathcal{D}_{s_k}(\cdot)$ represent the encryption function (with public-key p_k) and decryption function (with secret-key s_k), respectively. Let m_1 and m_2 be two messages and c a scalar value. The additive homomorphism has the following properties:

$$\mathcal{D}_{s_k}\big(\mathcal{E}_{p_k}(m_1) \cdot \mathcal{E}_{p_k}(m_2)\big) = m_1 + m_2,$$

$$\mathcal{D}_{s_k}\big(\mathcal{E}_{p_k}(m)^c\big) = c \cdot m.$$

Hereafter, we denote the encryption of a plaintext m, encrypted with public-key p_k, by $[m]_{p_k}$. Since additively homomorphic cryptosystems require ciphertexts encrypted with the same public-key, we omit the public-key (p_k) and simply write $[m]$ instead of $[m]_{p_k}$ when it is clear from the context.[4] Moreover, we use symbols $\oplus$ and $\ominus$ to denote homomorphic addition and subtraction, respectively. Specifically, $[m_1] \oplus [m_2] = [m_1 + m_2]$ and $[m_1] \ominus [m_2] = [m_1 - m_2]$.

In additive homomorphic cryptosystem, addition and scalar multiplication operations are performed over ciphertexts, without the need to decrypt them. However, performing more complex operations in additive homomorphic cryptosystem requires designing two-party interactive protocols. Next, we introduce secure two-party computation protocols, which serve as building blocks for the construction of our HE-based protocols encoding policy evaluation.

Secure Equality Protocol: Secure equality is used to determine the equality of two ciphertexts [41]. Given two ciphertexts $[a]$ and $[b]$, the secure equality test between $[a]$ and $[b]$ is defined as follows:

$$[a \overset{?}{=} b] = \begin{cases} [1] & \text{if } a = b, \\ [0] & \text{otherwise.} \end{cases}$$

Secure Comparison Protocol: Secure comparison is used to compare two ciphertexts [42]. Given two ciphertexts $[a]$ and $[b]$, the secure comparison between $[a]$ and $[b]$ is defined as follows:

$$[a \overset{?}{\leqslant} b] = \begin{cases} [1] & \text{if } a \leqslant b, \\ [0] & \text{otherwise.} \end{cases}$$

Secure Multiplication Protocol: Secure multiplication aims to compute the multiplication between two ciphertexts [16]. Given two ciphertexts $[a]$ and $[b]$, the secure multiplication of $[a]$ and $[b]$ is defined as follows:

$$[a] \otimes [b] = [a \cdot b].$$

2.3. *Secure function evaluation*

Despite allowing computations in the ciphertext domain, homomorphic encryption is usually expensive in terms of computation cost. *Secure function evaluation* (*SFE*) is an alternative to homomorphic encryption that enables several parties to compute a function on their private inputs without revealing any information apart from the result of the function. In this work, we implement secure function evaluation in two-party setting using the ABY framework [13]. ABY provides the constructions for <u>A</u>rithmetic circuits [4], <u>B</u>oolean circuits [22], and <u>Y</u>ao's garbled circuits [57]. For our work, we only use Boolean circuits since they provide efficient constructions for nonlinear functions.

Given two parties P_1, P_2 and their corresponding inputs x and y, ABY first creates secret shares for each party and a circuit that computes a specific function f, and then evaluates f on the secret shares using the circuit. Secret shares of each party are represented as $\langle x\rangle_1$, $\langle x\rangle_2$ and $\langle y\rangle_1$, $\langle y\rangle_2$. Secret shares are created for each bit of the input: given a bit x_i, $\langle x_i\rangle_1$, $\langle x_i\rangle_2$ are such that $\langle x_i\rangle_1 \boxplus \langle x_i\rangle_2 \equiv x_i \mod 2$,

[4]Note that the encryption of two equal messages with the same public-key typically results in two different ciphertexts. In many cryptosystems like the Paillier cryptosystem, this is guaranteed by the fact that the encryption function is probabilistic.

where $\boxplus$ represents bitwise XOR operation. The result of the function is reconstructed by combing the secret shares obtained by each party through a bitwise XOR operation.

For this work, we adopt seven Boolean gates from the ABY framework as building blocks for the design of our SFE-based policy evaluation protocols. For more details on the mechanism and implementation of Boolean circuits, we refer the reader to [13]. Next, we present these gates.

Inverse Gate: The inverse gate is used to compute the negation of a secret shared input in modulus 2^ℓ. The inverse here refers to the additive inverse in mod 2^ℓ, such that the additive inverse of a number a is equivalent to $2^\ell - a$. Given a secret shared input $\langle a\rangle$, the inverse gate is defined as:

$$\langle \neg a\rangle = -\langle a\rangle \mod 2^\ell.$$

AND Gate: The AND gate is used to perform a bitwise AND operation between two secret shared inputs using. Given secret shared inputs $\langle a\rangle$ and $\langle b\rangle$, the AND gate is defined as:

$$\langle a \wedge b\rangle = \langle a\rangle \wedge \langle b\rangle \mod 2^\ell.$$

OR Gate: The OR gate is used to perform a bitwise OR operation between two secret shared inputs. Given the secret shared inputs $\langle a\rangle$ and $\langle b\rangle$, the OR gate is defined as:

$$\langle a \vee b\rangle = \langle a\rangle \vee \langle b\rangle \mod 2^\ell.$$

Subtraction Gate: The subtraction gate overloads integer subtraction such that the result is equal to the difference of two secret shared inputs in modulus 2^ℓ. Given two secret shared inputs $\langle a\rangle$ and $\langle b\rangle$, the subtraction gate is represented as:

$$\langle a - b\rangle = \langle a\rangle - \langle b\rangle \mod 2^\ell.$$

Multiplication Gate: The multiplication gate overloads integer multiplication such that the result is equal to the multiplication of two secret shared inputs in modulus 2^ℓ. Given two secret shared inputs $\langle a\rangle$ and $\langle b\rangle$, the multiplication gate is represented as:

$$\langle a \times b\rangle = \langle a\rangle \times \langle b\rangle \mod 2^\ell.$$

Equality Gate: The equality gate is used to check the equality of two secret shared inputs in modulus 2^ℓ. Given secret shared inputs $\langle a\rangle$ and $\langle b\rangle$, the equality gate is defined as:

$$\langle a \overset{?}{=} b\rangle = \begin{cases} \langle 1\rangle & \text{if } a = b \\ \langle 0\rangle & \text{otherwise} \end{cases}$$

Comparison Gate: The comparison gate is used to check the equality of two secret shared inputs in modulus 2^ℓ. Given secret shared inputs $\langle a\rangle$ and $\langle b\rangle$, the comparison gate is defined as:

$$\langle a \overset{?}{\leqslant} b\rangle = \begin{cases} \langle 1\rangle & \text{if } a \leqslant b \\ \langle 0\rangle & \text{otherwise} \end{cases}$$

3. A framework for privacy-preserving multi-party access control

This section presents our framework for privacy-preserving multi-party access control. First, we present the model for multi-party access control adopted in this work and then we present the architecture of our framework and the underlying security assumptions.

3.1. Multi-party policy model

Collaborative systems usually provide their users with an environment in which they can interact and jointly contribute to the creation and management of shared resources. To protect those resources, each user can specify access requirements stating who is authorized to access them and under which circumstances. The access requirements provided by different users, however, can be in conflict. Therefore, a collaborative system should resolve these conflicting requirements in order to determine whether access to co-owned resources should be granted.

Traditional access control models are centered on a single-owner governance model (*i.e.*, they assume that resources are controlled by single entities) and, thus, they are not suitable for collaborative systems [11,24]. To this end, recent years have seen the emergence of several models for multi-party access control [45]. These models enable collaborative access decision making by providing a means to reconcile the conflicts arising from the evaluation of the access requirements provided by users involved in the protection of co-owned resources.

In this work, we adopt the data governance model proposed in [35] as the underlying multi-party access control model. This model provides a general framework to explicitly reason on the level of authority that users have over co-owned resources based on their relations with the resource [12] and to build a multi-party access control policy based on their authorization requirements. Specifically, it captures the relations that users have with a co-owned resource and, based on these relations, determines suitable strategies to resolve possible policy conflicts, thus accounting for the level of authority that users have on co-owned resources. These policy conflict resolution strategies can be realized using the operators presented in Table 1. Compared to other models for multi-party access control (see [45] for a survey), the model in [35] allows a more fine-grained governance of co-owned resources by allowing the adoption of arbitrary conflict resolution strategies.

As an illustration of the application of this model, consider the following example, which is based on one of the scenarios presented in the introduction. In particular, consider the case where two car companies (C_1, C_2), a navigation company (N_1) and a ride sharing company (R_1) form a joint venture to build a classification model in order to support autonomous driving cars in finding the fast navigation paths with respect to traffic conditions. To this end, these companies share their camera data, LiDAR sensor information, digital mapping data and passengers' route preferences to train the classification model. The partners also agree that the model can be shared with other collaborators and clients for research purposes and/or additional revenue.

The classification model, however, can be used to reconstruct potentially privacy-sensitive training data [33]. Therefore, each partner in the joint venture might specify access control policies determining who is authorized to access the model and under which conditions based on their own business constraints. Below, we present some hypothetical policies, denoted by p_{C_1}, p_{C_2}, p_{R_1} and p_{N_1}, for each partner:

$$p_{C_1} = (\mathit{country} \in \texttt{EU}, 1) \qquad p_{C_2} = (\mathit{role} = \texttt{partner}, 1) \nabla (\mathit{type} = \texttt{car}, 0)$$

$$p_{R_1} = (\mathit{type} = \texttt{raider}, 0) \rhd 1 \qquad p_{N_1} = (\mathit{role} = \texttt{collaborator} \,\tilde{\sqcap}\, \mathit{purpose} = \texttt{research}, 1)$$

Intuitively, car company C_1 allows sharing the model with any company within the European Union (denoted by *country* $\in$ EU[5]). On the other hand, car company C_2 allows access to partners in joint venture but denies access to car companies; these two access constraints are combined using the permit-overrides policy combining operator (∇). In practice, p_{C_2} denies access to car companies that are not partners in the joint venture. The raider sharing company R_1 does not want other raider sharing companies to access the classification model but does not impose any other constraints, for instance, on the country of the companies that can access the model. This is represented in p_{R_1} by combining R_1's access constraint with a permit default policy using the first-applicable policy combining operator ($\triangleright$). Finally, the policy of the navigator company N_1, p_{N_1}, allows collaborators to use the model only for research purposes.

These policies can specify conflicting access constraints. For instance, C_1's policy would allow a client riding sharing company located in the EU to access the model, whereas R_1's policy would deny that request; on the other hand, car company C_2 and navigator company N_1 do not impose any constraints for this case (*i.e.*, the evaluation of their policies would return the *not-applicable* decision). To determine whether access to the classification model should be granted or not, the access control policies of every partner in the joint venture should be combined together, forming the multi-party access control policy for the classification model. Following the framework in [35], such a multi-party policy can be defined, for instance, by taking into account the shares of each partner in the joint venture. For example, the multi-party policy $p = (p_{C_1} \triangle p_{C_2}) \triangleright (p_{N_1} \triangle p_{R_1}) \triangleright 0$ indicates that the access constraints of the car companies C_1 and C_2 have priorities over the constraints of other partners (*e.g.*, due to their larger shares in the joint venture) and that access is denied in case none of the partners' policies applies.

The multi-party policies considered in this work can be evaluated using existing access control mechanisms. However, such mechanisms require the policies to be provided in plaintext in order to be evaluated. Making these policies available to other partners (or third parties) might reveal confidential information about a company's business strategies and commercial relationships with other companies, which the company might want to keep confidential (*e.g.*, by knowing p_{C_1}, one may infer that C_1 does not have clients and/or collaborators outside the EU). Information about companies' business relationships can be exploited by the competitors to drive more appropriate business strategies, to make better decisions towards pricing, terms, risk, and thus win business competition [20]. In this work, we focus on the protection of user policy confidentiality and propose a framework that allows the involved members to disclose their access control policies in a private form while still remaining capable of making collaborative access decisions based on the access constraints provided by each individual user.

3.2. Architecture

To enable the evaluation of multi-party policies while protecting the confidentiality of user policies, we design an access control framework that supports policy evaluation over protected input. An overview of the proposed framework for privacy-preserving multi-party access control is presented in Fig. 1. The framework is general and can be realized using various privacy-preserving techniques. To avoid confusion, hereafter, we denote the protected version of a message m regardless of the specific privacy-preserving technique used for its realization as $(\!|m|\!)$ and use the notation introduced in Section 2 only when explicitly referring to Homomorphic Encryption and Secure Functional Evaluation.

The proposed framework comprises four main entities:

[5]Here the inclusion ($\in$) constraint is used as a shorthand to check if the country of the requesting company is one of the EU countries. It is trivial to observe that an inclusion constraint can be rewritten as equality constraints over the elements of the set.

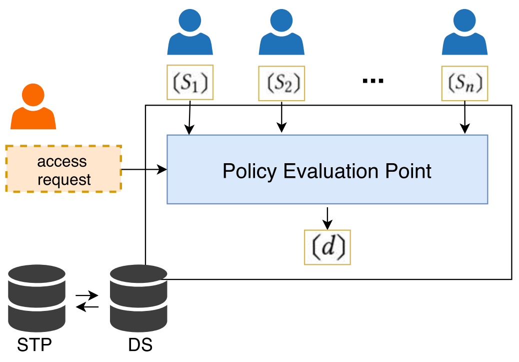

Fig. 1. Architecture.

- *Data holders* (👤) share their resources and data along with encrypted privacy policies determining to whom access is granted.
- *Access requester* (👤) requests access to the shared resource.
- *Data Server (DS)* stores the data holders' resources and is responsible for their protection. Specifically, the Data Server evaluates the encrypted data holders' policies to determine whether access to their resources should be granted.
- *Semi Trusted Party (STP)* is a semi-honest entity that assists Data Server in the secure evaluation of data holders' policies.

Data holders provide their policies in a private form (represented by $(s_1), \ldots, (s_n)$ in Fig. 1) to the Data Server. Data holders are not involved in the evaluation of either their policies or the multi-party policy and, thus, they are not required to be online for a decision to be made. This task is performed by the Data Server together with the STP.

Upon receiving an access request, the Data Server's *policy evaluation point* evaluates the request against the multi-party policy, which comprises the user policies, under privacy preservation with the assistance of the STP. To enable policy evaluation under privacy preservation, we propose secure computation protocols to determine the applicability of policies to the access request and to compute the combining operators in Table 1 over private inputs (*cf.* Section 4). These protocols can be used as building blocks for the secure evaluation of arbitrary multi-party policies expressed using those operators.

Once the multi-party policy has been evaluated, the policy evaluation point returns the access decision in private form, (d), corresponding to the evaluation of the access request against the multi-party policy. To be able to enforce the decision, the Data Server derives the decision in plaintext from the decision in private form together with the STP.

We have realized the framework using two alternative privacy preserving techniques, namely Homomorphic Encryption (HE) and Secure Functional Evaluation (SFE). The underlying privacy-preserving technique dictates the 'private form' in which policies are provided and the computations to be performed in order to obtain the access decision in plaintext. In HE, the STP generates public (p_k) and private (s_k) keys and sends the public key p_k to the data holders and to the Data Server. Data holders encrypt their policies using p_k and send the encrypted policies to the Data Server. To derive the access decision, the Data Server should not have access to the private key; otherwise, it will be able to learn users' policies. In order for the Data Server to decrypt the encrypted decision without the STP learning it, the Data Server adds random noise r to the encrypted decision $[d]$ and sends $[d + r]$ to the STP. The

STP decrypts the ciphertext and sends $d + r$ to the Data Server. The Data Server can obtain the decision to be enforced by removing noise r.

In SFE, data holders create two secret shares of their policies and provide one share to the Data Server and the other share to the STP. After policy evaluation, the Data Server derives the access decision in plaintext from the decision in private form by recombining the secret share obtained by evaluating the multi-party policy with the one obtained by the STP (*cf.* Section 2.3).

3.3. Security assumptions

We assume a semi-honest security model where all participants are assumed to be honest-but-curious [21]. This model implies that all entities follow the protocol specification properly, but they are interested in obtaining more information from their input, intermediary messages and output. Specifically, they keep track of the messages exchanged and try to learn as much information as possible from them. This assumption guarantees that computations do not leak any unintended information. It is worth noting that while the semi-honest adversary model is more restrictive compared to the malicious model, it is a well-accepted adversary model with many applications in real-world scenarios, *e.g.*, to protect sensitive information against passive insider attacks by administrators or government agencies, or when it is assured that the parties are trusted to not actively misbehave [13]. In this light, the semi-honest adversary model can be applied in the context of privacy preserving multi-party access control as the involved parties are typically engaged in a collaboration, implying that there exists a certain level of trust among them (see, *e.g.*, the scenarios in the introduction). We also remark that the semi-honest adversary model enables an efficient implementation of the secure computation protocols performing considerably faster than the malicious setting, while offering a sufficient level of security [47]. This is specifically a desirable property for applications like the evaluation of security policies in private forms where (collaborative) access decisions should be made in a short amount of time.

With respect to the semi-honest security assumption, our goal is to design protocols that provide security against honest-but-curious non-colluding Data Server and STP. The non-colluding two-server setting is typically employed to reduce the workload on the client side. Without the employment of a semi trusted party, all computations have to be performed between the client and Data Server. This, however, is not desirable because it requires the data holders to have enough computational resources and to be online during computations. Note that the definition of protocols aiming to prevent the Data Server and STP from colluding is orthogonal to the scope of this work and here we consider them as two servers with independent interest who do not wish to or cannot collude [30]. The assumption of non-colluding Data Server and STP can be achieved through physical means (*e.g.*, with the use of ballot boxes [32]) or by adding additional security verification on trusted communication channels (*e.g.*, with the use of mediator model [1]). Moreover, several approaches have been proposed to verify the faithfulness of servers in secure two-party computation. For instance, it has been shown that multi-party computation protocols secure against semi-honest adversaries can be transformed to zero-knowledge proofs [23,28].

Nonetheless, we assume that a semi-honest adversary can compromise any subset of data holders (where at least two data holders are honest), the access requester and at most one of the STP and Data Server (*i.e.*, if one is controlled by the adversary, the other behaves honestly[6]). This security definition assumes that such an adversary can only learn the policies of the data holders it has compromised and

[6]This captures the property that the Data Server and STP are not colluding parties.

the final output but nothing else about the policies of the honest data holders [36]. We require that at least two data holders are honest because colluding data holders can learn the policy evaluation of the other data holders by comparing the final decision with the evaluation of their policies due to the definition of the used policy combining operators (see also Section 6). Imposing that at least two data holders are honest allows us to focus on flaws in the design of the protocols rather than on leakages that are inevitable.

We also assume that all parties communicate over an authenticated channel. This assumption aims to prevent attacks coming from outside the framework.

4. Protocol design

For the realization of a practical mechanism capable of evaluating policies in protected form, we need efficient secure computation protocols. In this section, we investigate the design of such protocols. We first introduce a suitable representation of targets and policies, and then we present secure computation protocols for their evaluation. These protocols serve as building blocks and can be used to evaluate arbitrary multi-party policies. In the next section, we describe the implementation of these building block protocols based on two alternative privacy-preserving techniques, namely Homomorphic Encryption and Secure Functional Evaluation.

4.1. Data structures

This section presents the data structures used for the specification of policies in private form and the representation of the decision space to enable the design of efficient protocols for secure policy evaluation.

Policy Specification: Privacy-preserving technologies like Homomorphic Encryption and Secure Function Evaluation only operate on integer numbers. To this end, we encode every attribute ($a_i \in \mathcal{A}$) and their attribute values ($v \in \mathcal{V}_{a_i}$) into a unique integer number,[7] where for numeric attribute values the ordering is preserved.

This work aims at a privacy-preserving mechanism that allows the evaluation of multi-party access control policies while protecting the confidentiality of the user policies forming the multi-party policy. In this light, we aim to protect the constraints in the target, which determine the applicability of policies, along with atomic policies (*i.e.*, policies consisting of the *permit* and *deny* decisions) whereas combining operators are not protected. An atomic target $t = a\,\phi\,v$ is protected by protecting the attribute a, attribute value v, and predicate ϕ individually, *i.e.* $\llparenthesis t \rrparenthesis = (\llparenthesis a \rrparenthesis, \llparenthesis v \rrparenthesis, \llparenthesis \phi \rrparenthesis)$. Given that an atomic target always consists of three elements, protecting each element of the target individually does not disclose information about the target. Note that, in order to reason over predicates under privacy preservation, they are also encoded into integer numbers. Hereafter, we use 1, 2, 3 and 4 to denote $=$, $\neq$, $\leqslant$ and $\geqslant$, respectively. A composite target $t = op(t_1 \ldots t_k)$ is in protected form if its subtargets are in protected form, *i.e.* $\llparenthesis t \rrparenthesis = op(\llparenthesis t_1 \rrparenthesis, \ldots, \llparenthesis t_k \rrparenthesis)$. Atomic policies are protected by protecting the decision, *i.e.* $\llparenthesis 1 \rrparenthesis$ and $\llparenthesis 0 \rrparenthesis$. Targeted and composite policies p are in protected form if their subpolicies and target (for targeted policies) are in protected form.

[7]The uniqueness of attribute values can be easily achieved through a renaming of attribute values.

Note that, in this work, queries are not considered sensitive and, thus, they are received in plaintext. However, in order to be able to evaluate a policy against a query, the attributes and attribute values in the query should be mapped to integer numbers following the same encoding of attributes and attribute values used for the policy.

Decision Space: In this work, we adopt a policy language that is grounded on a three-valued logic and use $\mathcal{D}_7 = \mathbb{P}(\{1, 0, \perp\}) \setminus \emptyset$ as the decision set (*cf.* Section 2.1). Although this language is representative for several ABAC policy languages like XACML [38], the use of $\mathcal{D}_7$ and three-valued logic introduces a number of challenges when policy evaluation is performed under privacy preservation. Non-singleton decisions – even when each element of the decision is given in protected form – might reveal some information about user policies by observing the 'size' of the decision. For instance, by observing a decision consisting of three elements, an attacker can easily infer that the decision is $\{1, 0, \perp\}$ as this is the only decision consisting of three elements. Therefore, we need to mask the 'size' of decisions in order to preserve the confidentiality of user policies. One possible solution for hiding the number of elements in a decision is to use data packing [55]. The main idea behind data packing is to efficiently use the ciphertext message space, which is generally much larger than the size of encrypted values, to pack a set of values in one ciphertext. However, data packing techniques require packing and unpacking the set for each round of communication and, thus, it is not efficient to be employed in our context. Moreover, while data packing techniques have been proposed for Homomorphic Encryption [42], it is not supported by other secure computation frameworks like Secure Function Evaluation. In addition, previous work [51] has shown that a direct encoding of the three-valued logic operators in Table 1 requires the application of secure multiplication, equality and comparison protocols, which are usually heavy in terms of both computation and communication costs.

For the design of efficient secure computation protocols able to evaluate a policy in protected form against a query, we need suitable representation of the decision space encoding the results of policy evaluation. Inspired by previous work [37,56], we adopt a Boolean representation of the decision space and three-valued operators in Table 1. Given a policy p, we represent the decision space of p as a triple $(b_1, b_0, b_\perp)$ of propositional formulas representing sets of queries Q_1, Q_0 and $Q_\perp$ such that $d \in [\![p]\!]_P(q)$ exactly when $q \in Q_d$. Intuitively, each element b_i corresponds to a single decision $i \in \{1, 0, \perp\}$ with $b_i \in \{1, 0\}$ such that $b_i = 1$ means that $i \in d$ and $b_i = 0$ that $i \notin d$.

In order to encode the target and policy evaluation functions and three-valued logic operators in Table 1 into triples of propositional formulas, we adopt and extend the rules proposed in [37]. These rules can be employed to compute a triple of propositional formulas $(b_1, b_0, b_\perp)$ representing $[\![p]\!]_P$. More precisely, each propositional formula b_d denotes the set of queries $Q_d \subseteq \mathcal{Q}$ satisfying $d = [\![p]\!]_P(q)$, whenever $q \in Q_d$. The transformation rules τ (for targets) and π (for policies) presented in Tables 2 and 3 explain the construction of the propositional formula for all targets, (policy) constants and all (policy and target) operators in Table 1. We refer to [37] for details on the correctness of transformation rules τ and π.

This encoding of the target and policy evaluation functions allows the definition of protocols implementing the $\neg$, $\wedge$, and $\vee$ operators, which can efficiently be implemented using *inverse*, *AND* and *OR* gates in SFE and *negation*, *maximum* and *minimum* protocols in HE (see Section 5). Moreover, this encoding allows masking the number of elements forming a decision d by using a fix-size representation for all decisions.

Table 2
Transformation rules for targets

Target	τ_1	τ_0	$\tau_\bot$
$a\ \phi\ v$	$(a \in \mathcal{A}_q) \wedge (\bigvee_{v_i \in \mathcal{V}_{a\|q}} (v_i\ \phi\ v))$	$(a \in \mathcal{A}_q) \wedge \neg(\bigvee_{v_i \in \mathcal{V}_{a\|q}} (v_i\ \phi\ v))$	$a \notin \mathcal{A}_q$
$\neg t_1$	$\tau_0(t_1)$	$\tau_1(t_1)$	$\tau_\bot(t_1)$
$\sim t_1$	$\tau_1(t_1)$	$\tau_0(t_1) \vee \tau_\bot(t_1)$	0
$t_1 \,\widetilde{\sqcup}\, t_2$	$\tau_1(t_1) \vee \tau_1(t_2)$	$\tau_0(t_1) \wedge \tau_0(t_2)$	$(\tau_\bot(t_1) \wedge \neg\tau_1(t_2)) \vee (\tau_\bot(t_2) \wedge \neg\tau_1(t_1))$
$t_1 \,\widetilde{\sqcap}\, t_2$	$\tau_1(t_1) \wedge \tau_1(t_2)$	$\tau_0(t_1) \vee \tau_0(t_2)$	$(\tau_\bot(t_1) \wedge \neg\tau_0(t_2)) \vee (\tau_\bot(t_2) \wedge \neg\tau_0(t_1))$
$t_1 \sqcup t_2$	$(\tau_1(t_1) \wedge \neg\tau_\bot(t_2)) \vee (\tau_1(t_2) \wedge \tau_\bot(t_1))$	$\tau_0(t_1) \wedge \tau_0(t_2)$	$\tau_\bot(t_1) \vee \tau_\bot(t_2)$
$t_1 \sqcap t_2$	$\tau_1(t_1) \wedge_1 (t_2)$	$(\tau_0(t_1) \wedge \neg\tau_\bot(t_2)) \vee (\tau_0(t_2) \wedge \neg\tau_\bot(t_1))$	$\tau_\bot(t_1) \vee \tau_\bot(t_2)$
$t_1 \nabla t_2$	$\tau_1(t_1) \vee \tau_1(t_2)$	$(\tau_0(t_1) \wedge \neg\tau_1(t_2)) \vee (\tau_0(t_2) \wedge \neg\tau_1(t_1))$	$\tau_\bot(t_1) \wedge \tau_\bot(t_2)$
$t_1 \vartriangle t_2$	$(\tau_1(t_1) \wedge \neg\tau_0(t_2)) \vee (\tau_1(t_2) \wedge \neg\tau_0(t_1))$	$\tau_0(t_1) \vee \tau_0(t_2)$	$\tau_\bot(t_1) \wedge \tau_\bot(t_2)$
$t_1 \rhd t_2$	$\tau_1(t_1) \vee (\tau_\bot(t_1) \wedge \tau_1(t_2))$	$\tau_0(t_1) \vee (\tau_\bot(t_1) \wedge \tau_0(t_2))$	$\tau_\bot(t_1) \wedge \tau_\bot(t_2)$

Table 3

Transformation rules for policies

Policy	π_1	π_0	$\pi_\perp$
1	1	0	0
0	0	1	0
(t, p_1)	$\tau_1(t) \wedge \pi_1(p_1)$	$\tau_1(t) \wedge \pi_0(p_1)$	$\tau_0(t) \vee \tau_\perp(t) \vee (\tau_1(t) \wedge \pi_\perp(p_1))$
$\neg p_1$	$\pi_0(p_1)$	$\pi_1(p_1)$	$\pi_\perp(p_1)$
$\sim p_1$	$\pi_1(p_1)$	$\pi_0(p_1) \vee \pi_\perp(p_1)$	0
$p_1 \tilde{\sqcup} p_2$	$\pi_1(p_1) \vee \pi_1(p_2)$	$\pi_0(p_1) \wedge \pi_0(p_2)$	$(\pi_\perp(p_1) \wedge \neg\pi_1(p_2)) \vee (\pi_\perp(p_2) \wedge \neg\pi_1(p_1))$
$p_1 \tilde{\sqcap} p_2$	$\pi_1(p_1) \wedge \pi_1(p_2)$	$\pi_0(p_1) \vee \pi_0(p_2)$	$(\pi_\perp(p_1) \wedge \neg\pi_0(p_2)) \vee (\pi_\perp(p_2) \wedge \neg\pi_0(p_1))$
$p_1 \sqcup p_2$	$(\pi_1(p_1) \wedge \neg\pi_\perp(p_2)) \vee (\pi_1(p_2) \wedge \neg\pi_\perp(p_1))$	$\pi_0(p_1) \wedge \pi_0(p_2)$	$\pi_\perp(p_1) \vee \pi_\perp(p_2)$
$p_1 \sqcap p_2$	$\pi_1(p_1) \wedge \pi_1(p_2)$	$(\pi_0(p_1) \wedge \neg\pi_\perp(p_2)) \vee (\pi_0(p_2) \wedge \neg\pi_\perp(p_1))$	$\pi_\perp(p_1) \vee \pi_\perp(p_2)$
$p_1 \nabla p_2$	$\pi_1(p_1) \vee \pi_1(p_2)$	$(\pi_0(p_1) \wedge \neg\pi_1(p_2)) \vee (\pi_0(p_2) \wedge \neg\pi_1(p_1))$	$\pi_\perp(p_1) \wedge \pi_\perp(p_2)$
$p_1 \triangle p_2$	$(\pi_1(p_1) \wedge \neg\pi_0(p_2)) \vee (\pi_1(p_2) \wedge \neg\pi_0(p_1))$	$\pi_0(p_1) \vee \pi_0(p_2)$	$\pi_\perp(p_1) \wedge \pi_\perp(p_2)$
$p_1 \triangleright p_2$	$\pi_1(p_1) \vee (\pi_\perp(p_1) \wedge \pi_1(p_2))$	$\pi_0(p_1) \vee (\pi_\perp(p_1) \wedge \pi_0(p_2))$	$\pi_\perp(p_1) \wedge \pi_\perp(p_2)$

4.2. Target evaluation

To enable the secure evaluation of targets in protected form against a query, we design secure computation protocols implementing the rules presented in Table 2 over protected input. We first propose generic secure computation protocols for the evaluation of atomic targets (the first row of Table 2). As it can be observed in the table, two operations are needed for such an evaluation: *i)* an operation to determine whether the attribute in the target is present in the query ($a \in \mathcal{A}_q$) and *ii)* an operation to determine whether there exists an attribute name-value pair in the query that satisfies the constraint in the target ($\bigvee_{v_i \in \mathcal{V}_{a|q}}(v_i \, \phi \, v)$). To this end, we propose building-block protocols for the secure computation of these two operations – *secure membership* and *secure matching* – over protected inputs. Then, we present a generic protocol for target evaluation.

Secure membership protocol. Secure membership is used to determine whether an encrypted value belongs to a set of encrypted values [19]. Given a ciphertext $\llparenthesis a \rrparenthesis$ and a set of ciphertexts $\llparenthesis \mathcal{B} \rrparenthesis = \{\llparenthesis b_1 \rrparenthesis, \ldots, \llparenthesis b_n \rrparenthesis\}$, we can determine if $\llparenthesis a \rrparenthesis$ belongs to $\llparenthesis \mathcal{B} \rrparenthesis$ using the following protocol:

$$\llparenthesis P(a, \mathcal{B}) \rrparenthesis = \big(\llparenthesis b_1 \rrparenthesis - \llparenthesis a \rrparenthesis\big) \times \cdots \times \big(\llparenthesis b_n \rrparenthesis - \llparenthesis a \rrparenthesis\big)$$

Intuitively, this protocol returns $\llparenthesis 0 \rrparenthesis$ if $a \in \mathcal{B}$. Based on this protocol, the *secure membership* protocol is then defined as follows:

$$\llparenthesis a \overset{?}{\in} \mathcal{B} \rrparenthesis = \llparenthesis P(a, \mathcal{B}) \overset{?}{=} 0 \rrparenthesis$$

which returns $\llparenthesis 1 \rrparenthesis$ if $a \in \mathcal{B}$, and $\llparenthesis 0 \rrparenthesis$ otherwise.

Secure matching protocol. To determine whether a policy is applicable to a query we need to verify if the attribute name-value pairs forming the query satisfy the constraints in its target. To this end, we first define the *secure value-matching* protocol to determine whether a protected value satisfies the constraint defined in an atomic target under privacy preservation and, then, we show how this protocol can be used to determine whether there exists an attribute name-value pair in the query that satisfies the target.

Recall that our goal is to protect the confidentiality of user policies. Accordingly, the constraints establishing the applicability of these policies should be protected and, thus, we assume that not only the attribute name and value but also the predicate in an atomic target is in protected form. Given an atomic target in protected form $\llparenthesis t \rrparenthesis = (\llparenthesis a \rrparenthesis, \llparenthesis v \rrparenthesis, \llparenthesis \phi \rrparenthesis)$ and a protected value $\llparenthesis v' \rrparenthesis$, we define the *secure value-matching* protocol as follows:

$$\begin{aligned}
\textit{value-matching}\big(\llparenthesis v' \rrparenthesis, \llparenthesis v \rrparenthesis, \llparenthesis \phi \rrparenthesis\big) = {} & \big(\llparenthesis \phi \overset{?}{=} 1 \rrparenthesis \wedge \llparenthesis v' \overset{?}{=} v \rrparenthesis\big) \\
& \vee \big(\llparenthesis \phi \overset{?}{=} 2 \rrparenthesis \wedge \neg \llparenthesis v' \overset{?}{=} v \rrparenthesis\big) \\
& \vee \big(\llparenthesis \phi \overset{?}{=} 3 \rrparenthesis \wedge \llparenthesis v' \overset{?}{\leqslant} v \rrparenthesis\big) \\
& \vee \big(\llparenthesis \phi \overset{?}{=} 4 \rrparenthesis \wedge \big(\neg \llparenthesis v' \overset{?}{\leqslant} v \rrparenthesis \vee \llparenthesis v' \overset{?}{=} v \rrparenthesis\big)
\end{aligned}$$

where $\llparenthesis \cdot \overset{?}{=} \cdot \rrparenthesis$ and $\llparenthesis \cdot \overset{?}{\leqslant} \cdot \rrparenthesis$ represent the equality and comparison protocols, respectively. It is worth noting that the protocol tests the validity of constraint $v' \phi v$ for every predicate ϕ. This is because the predicate is in protected form and, thus, it is not known which predicate occurs in the atomic target.

Algorithm 1: *matching*: secure matching protocol

```
Data: Atomic target in protected form $(\llparenthesis a \rrparenthesis, \llparenthesis v \rrparenthesis, \llparenthesis \phi \rrparenthesis)$, query $q$
Result: Matching $M \in \{\llparenthesis 1 \rrparenthesis, \llparenthesis 0 \rrparenthesis\}$
1 $M = \llparenthesis 0 \rrparenthesis$
2 for $a_i \in \mathcal{A}_q$ do
3     for $v_j \in V_{a_i|q}$ do
4         $M = M \vee (\llparenthesis a_i \overset{?}{=} a \rrparenthesis \wedge \mathit{value\text{-}matching}(\llparenthesis v_j \rrparenthesis, \llparenthesis v \rrparenthesis, \llparenthesis \phi \rrparenthesis))$
5     end
6 end
7 return $M$
```

The *value-matching* protocol verifies whether a given attribute value satisfies the constraint defined in the target. Determining whether a policy is applicable to a given request requires extending this verification by checking if there exists an attribute name-value pair in the query that satisfies the target of the policy. To this end, we define the secure *matching* protocol (Algorithm 1). Given an atomic target in protected form $\llparenthesis t \rrparenthesis = (\llparenthesis a \rrparenthesis, \llparenthesis v \rrparenthesis, \llparenthesis \phi \rrparenthesis)$ and a query $q = \{(a_1, v_1), \dots, (a_n, v_n)\}$, the secure *matching* protocol determines whether the query contains an attribute name-value pair that satisfies the target under privacy preservation. The protocol encompasses two *for* loops to verify the target against every attribute name-value pair in the request.[8] If the attribute in the target is present in the request and at least one of its values in q satisfies the constraint in the target, the protocol returns $\llparenthesis 1 \rrparenthesis$; otherwise the protocol returns $\llparenthesis 0 \rrparenthesis$.

Target evaluation protocol. We have now the machinery to define the protocol for secure target evaluation. Given a target in protected form $\llparenthesis t \rrparenthesis$ and a query q, the secure target evaluation protocol $\mathit{eval}^t(\llparenthesis t \rrparenthesis, q)$ consists of three subprotocols $\mathit{eval}^t_i(\llparenthesis t \rrparenthesis, q)$ with $i \in \{1, 0, \perp\}$, and evaluates to a triple of protected values $(\llparenthesis d_1 \rrparenthesis, \llparenthesis d_0 \rrparenthesis, \llparenthesis d_\perp \rrparenthesis)$ with $d_i \in \{1, 0\}$ such as $\llparenthesis d_i \rrparenthesis = \mathit{eval}^t_i(\llparenthesis t \rrparenthesis, q)$. Next, we define protocol $\mathit{eval}^t(\llparenthesis t \rrparenthesis, q)$ per cases based on the grammar of $\mathcal{T}_\mathcal{A}$ presented in Section 2.1.

Atomic target: Given an atomic target in protected form $\llparenthesis t \rrparenthesis = (\llparenthesis a \rrparenthesis, \llparenthesis v \rrparenthesis, \llparenthesis \phi \rrparenthesis)$ and a query q, protocol $\mathit{eval}^t(\llparenthesis t \rrparenthesis, q)$ consists of subprotocols:

$$\mathit{eval}^t_1\big(\llparenthesis t \rrparenthesis, q\big) = \llparenthesis a \overset{?}{\in} \mathcal{A}_q \rrparenthesis \wedge \mathit{matching}\big(\llparenthesis t \rrparenthesis, q\big)$$

$$\mathit{eval}^t_0\big(\llparenthesis t \rrparenthesis, q\big) = \llparenthesis a \overset{?}{\in} \mathcal{A}_q \rrparenthesis \wedge \neg\big(\mathit{matching}\big(\llparenthesis t \rrparenthesis, q\big)\big)$$

$$\mathit{eval}^t_\perp\big(\llparenthesis t \rrparenthesis, q\big) = \neg\llparenthesis a \overset{?}{\in} \mathcal{A}_q \rrparenthesis$$

Composite target: Given a composite target in protected form $\llparenthesis t \rrparenthesis = op(\llparenthesis t_1 \rrparenthesis, \dots, \llparenthesis t_n \rrparenthesis)$ and a query q, protocol $\mathit{eval}^t(\llparenthesis t \rrparenthesis, q)$ consists of the following subprotocols:

$$\mathit{eval}^t_1\big(\llparenthesis t \rrparenthesis, q\big) = op\big(\mathit{eval}^t_1\big(\llparenthesis t_1 \rrparenthesis, q\big), \dots, \mathit{eval}^t_1\big(\llparenthesis t_n \rrparenthesis, q\big)\big)$$

[8]Recall that the attribute name-value pairs in a query are in plaintext and, thus, we determine which values of an attribute occur in the query.

$$eval^t_0\big(\llparenthesis t \rrparenthesis, q\big) = op\big(eval^t_0\big(\llparenthesis t_1 \rrparenthesis, q\big), \ldots, eval^t_0\big(\llparenthesis t_n \rrparenthesis, q\big)\big)$$
$$eval^t_\perp\big(\llparenthesis t \rrparenthesis, q\big) = op\big(eval^t_\perp\big(\llparenthesis t_1 \rrparenthesis, q\big), \ldots, eval^t_\perp\big(\llparenthesis t_n \rrparenthesis, q\big)\big)$$

Note that the secure target evaluation protocol for composite target depends on the operator *op*. We omit the definition of the protocol per each operator as it can be easily derived from the formulas presented in Table 2 using secure computation protocols that implement the $\neg$, $\wedge$, and $\vee$ operators. We provide details of the implementation of these protocols in Section 5.

4.3. *Policy evaluation*

The secure evaluation of a policy against a given query requires secure computation protocols implementing the formulas given in Table 3. Here, we provide the design of a generic protocol for policy evaluation under privacy preservation and then we discuss in Section 5 how it can be realized in Homomorphic Encryption and Secure Function Evaluation.

Given a policy in protected form $\llparenthesis p \rrparenthesis$ and a query q, the secure policy evaluation protocol $eval^p(\llparenthesis p \rrparenthesis, q)$ consists of three subprotocols $eval^p_i(\llparenthesis p \rrparenthesis, q)$, one per each single decision $i \in \{1, 0, \perp\}$, and evaluates to a triple of protected values $(\llparenthesis d_1 \rrparenthesis, \llparenthesis d_0 \rrparenthesis, \llparenthesis d_\perp \rrparenthesis)$ with $d_i \in \{1, 0\}$ such as $\llparenthesis d_i \rrparenthesis = eval^p_i(\llparenthesis p \rrparenthesis, q)$. Next, we define protocol $eval^p(\llparenthesis p \rrparenthesis, q)$ per cases based on the grammar of $\mathcal{P}_\mathcal{A}$ presented in Section 2.1.

Decision policy: Given a policy comprising the *permit* or *deny* decision in protected form ($\llparenthesis 1 \rrparenthesis$ and $\llparenthesis 0 \rrparenthesis$ respectively) and a query q, protocol $eval^p(\llparenthesis p \rrparenthesis, q)$ consists of the following subprotocols:

$$eval^p_1\big(\llparenthesis p \rrparenthesis, q\big) = \llparenthesis p \overset{?}{=} 1 \rrparenthesis$$
$$eval^p_0\big(\llparenthesis p \rrparenthesis, q\big) = \llparenthesis p \overset{?}{=} 0 \rrparenthesis$$
$$eval^p_\perp = \llparenthesis 0 \rrparenthesis$$

Targeted policy: Given a targeted policy in protected form $(\llparenthesis t \rrparenthesis, \llparenthesis p \rrparenthesis)$ and a query q, protocol $eval^p((\llparenthesis t \rrparenthesis, \llparenthesis p \rrparenthesis), q)$ consists of the subprotocols:

$$eval^p_1\big((\llparenthesis t \rrparenthesis, \llparenthesis p \rrparenthesis), q\big) = eval^t_1\big(\llparenthesis t \rrparenthesis, q\big) \wedge eval^p_1\big(\llparenthesis p \rrparenthesis, q\big)$$
$$eval^p_0\big((\llparenthesis t \rrparenthesis, \llparenthesis p \rrparenthesis), q\big) = eval^t_0\big(\llparenthesis t \rrparenthesis, q\big) \wedge eval^p_0\big(\llparenthesis p \rrparenthesis, q\big)$$
$$eval^p_\perp\big((\llparenthesis t \rrparenthesis, \llparenthesis p \rrparenthesis), q\big) = eval^t_0\big(\llparenthesis t \rrparenthesis, q\big) \vee eval^t_\perp\big(\llparenthesis t \rrparenthesis, q\big) \vee \big(eval^t_1\big(\llparenthesis t \rrparenthesis, q\big) \wedge eval^p_\perp\big(\llparenthesis p \rrparenthesis, q\big)\big)$$

Composite policy: Given a composite policy in protected form $op(\llparenthesis p_1 \rrparenthesis, \ldots, \llparenthesis p_n \rrparenthesis)$ and a query q, protocol $eval^p(op(\llparenthesis p_1 \rrparenthesis, \ldots, \llparenthesis p_n \rrparenthesis), q)$ consists of the subprotocols:

$$eval^p_1\big(op\big(\llparenthesis p_1 \rrparenthesis, \ldots, \llparenthesis p_n \rrparenthesis\big), q\big) = op\big(eval^p_1\big(\llparenthesis p_1 \rrparenthesis, q\big), \ldots, eval^p_1\big(\llparenthesis p_n \rrparenthesis, q\big)\big)$$
$$eval^p_0\big(op\big(\llparenthesis p_1 \rrparenthesis, \ldots, \llparenthesis p_n \rrparenthesis\big), q\big) = op\big(eval^p_0\big(\llparenthesis p_1 \rrparenthesis, q\big), \ldots, eval^p_0\big(\llparenthesis p_n \rrparenthesis, q\big)\big)$$
$$eval^p_\perp\big(op\big(\llparenthesis p_1 \rrparenthesis, \ldots, \llparenthesis p_n \rrparenthesis\big), q\big) = op\big(eval^p_\perp\big(\llparenthesis p_1 \rrparenthesis, q\big), \ldots, eval^p_\perp\big(\llparenthesis p_n \rrparenthesis, q\big)\big)$$

As for the secure target evaluation protocol, the secure policy evaluation protocol for composite policies depends on the operator *op*. We omit the definition of the protocol per each operator as it can be easily derived from the formulas presented in Table 3.

5. Protocol implementation

In this section, we describe the realization of the protocols for secure policy evaluation presented in Section 4 using two well-known privacy-preserving techniques, namely (additively) Homomorphic Encryption (HE) and Secure Functional Evaluation (SFE).

5.1. SFE-based protocols

To realize the protocols presented in Section 4 using Secure Functional Evaluation (SFE), we employ the building blocks presented in Section 2.3, *i.e. inverse*, *AND*, *OR*, *subtraction*, *multiplication*, *equality*, and *comparison* gates. We first observe that the protocols for the evaluation of composite targets (Table 2) and policies (Table 3) are built over operators $\neg$, $\wedge$ and $\vee$, which can be implemented in SFE using the *inverse*, *AND* and *OR* gates, respectively. Therefore, in this section, we only detail how the secure membership and matching protocols can be realized in terms of SFE gates.

Secure membership protocol. Given a secret shared input $\langle a\rangle$ and a set of secret shared values $\mathcal{B} = \{\langle b_1\rangle, \dots, \langle b_n\rangle\}$, the secure membership protocol checks whether a belongs to $\mathcal{B}$ under privacy preservation. This protocol can be realized in SFE as:

$$\langle a \overset{?}{\in} \mathcal{B}\rangle = \big\langle P(a, \mathcal{B}) \overset{?}{=} 0\big\rangle$$

where

$$\big\langle P(a, \mathcal{B})\big\rangle = \langle b_1 - a\rangle \times \cdots \times \langle b_n - a\rangle$$

It is worth noting that the realization of the secure membership protocols in SFE requires the use of the *subtraction*, *multiplication*, and *equality* gates.

Secure matching protocol. To verify the applicability of a secret shared policy to a query, we realize the value-matching protocol presented in Section 4 using Secure Function Evaluation. Given a secret shared target $\langle t\rangle = (\langle a\rangle, \langle v\rangle, \langle \phi\rangle)$, and a secret shared value $\langle v'\rangle$, the value-matching protocol is implemented in SFE as:

$$\begin{aligned}
\textit{value-matching}^{B}\big(\langle v'\rangle, \langle v\rangle, \langle \phi\rangle\big) = {} & \big(\langle \phi \overset{?}{=} 1\rangle \wedge \big\langle v' \overset{?}{=} v\big\rangle\big) \\
& \vee \big(\langle \phi \overset{?}{=} 2\rangle \wedge \neg\big\langle v' \overset{?}{=} v\big\rangle\big) \\
& \vee \big(\langle \phi \overset{?}{=} 3\rangle \wedge \big\langle v' \overset{?}{\leqslant} v\big\rangle\big) \\
& \vee \big(\langle \phi \overset{?}{=} 4\rangle \wedge \big(\neg\big\langle v' \overset{?}{\leqslant} v\big\rangle \vee \big\langle v' \overset{?}{=} v\big\rangle\big)
\end{aligned}$$

where $\neg$ $\wedge$, $\vee$, $\langle \cdot \overset{?}{=} \cdot \rangle$, and $\langle \cdot \overset{?}{\leqslant} \cdot \rangle$ are the *inverse*, *AND*, *OR*, *equality* and *comparison* gates presented in Section 2.3. Then, the secure matching protocol (Algorithm 1) can be easily realized in SFE by implementing operators $\wedge$ and $\vee$ using *AND* and *OR* gates, respectively.

5.2. HE-based protocols

As an alternative to Secure Function Evaluation, we realize the protocols for secure policy evaluation using (additively) Homomorphic Encryption (HE). Specifically, we implement the generic protocols presented in Section 4 using the five cryptographic building blocks presented in Section 2.2, *i.e.*, *addition*, *subtraction*, *multiplication*, *equality* and *comparison*. As shown in Tables 2 and 3, the protocols for secure policy evaluation rely on operators $\neg$, $\wedge$ and $\vee$. To this end, we first discuss how the secure computation of these operators can be realized in HE. Then, we present the implementation of the secure membership and matching protocols.

Secure HE-based computation of operators $\neg$, $\wedge$ and $\vee$. For the realization of the secure membership and matching protocols as well as for the realization of the protocols for secure evaluation of composite targets and policies, we need building block protocols implementing operators $\neg$, $\wedge$ and $\vee$.

It is worth noting that operators $\neg$, $\wedge$ and $\vee$ defined over Boolean logic are a reduction of the *negation* ($\neg$), *strong conjunction* ($\widetilde{\sqcap}$) and *strong disjunction* ($\widetilde{\sqcup}$) operators defined over three-valued logic [38], for which HE-based protocols have been proposed in [51]. Here, we propose an alternative implementation based on the Boolean encoding of the decision space (*cf.* Section 4.1), which is computationally cheaper compared to one proposed in [51] (*cf.* Section 5.3.2).

Operator $\neg$: Given an encrypted value $[a]$, protocol $\phi[a]$ computes the negation of a (*i.e.*, $\neg a$) under encryption as follows:

$$\phi[a] = [1] \ominus [a]$$

It is easy to verify that this protocol returns $[0]$ if $a = 1$ and $[1]$ if $a = 0$.

Operator $\wedge$: This operator can be realized using the secure multiplication protocol. Given two ciphertexts $[a]$ and $[b]$, we can observe that $[a] \otimes [b]$ returns $[1]$ when both a and b are equal to 1 and returns $[0]$ when at least one of a and b are equal to 0, thus having the same behavior of $\wedge$.

Operator $\vee$: Differently from operator $\wedge$, none of the building blocks presented in Section 5.2 implements operator $\vee$. This operator can be realized through a protocol that computes the minimum between two values under encryption. Given two ciphertexts $[a]$ and $[b]$, the *secure minimum* protocol $[a] \curlyvee [b]$ is defined as:

$$[a] \curlyvee [b] = [a] \oplus [b] \ominus \big([a] \otimes [b]\big)$$

Intuitively, this protocol returns $[1]$ when at least one of a or b is equal to 1; otherwise, $[0]$ is returned. It is easy to verify that this indeed captures the behavior of operator $\vee$.

Secure membership protocol. Given an encrypted value $[a]$ and a set of encrypted values $\mathcal{B} = \{[b_1], \dots, [b_n]\}$, the secure membership protocol can be implemented in HE as:

$$[a \overset{?}{\in} \mathcal{B}] = \big[P(a, \mathcal{B}) \overset{?}{=} 0\big]$$

where $[P(a, \mathcal{B})] = ([b_1] \ominus [a]) \otimes \cdots \otimes ([b_n] \ominus [a])$. We can observe that the HE-based membership protocol uses the subtraction, multiplication and equality protocols presented in Section 2.2.

Secure matching protocol. The protocols presented above can be used to implement the secure matching protocol in HE. Specifically, given an encrypted target $[t] = ([a], [v], [\phi])$ and an encrypted value $[v']$, the value-matching protocol can be implemented in HE as:

$$\begin{aligned} \textit{value-matching}^H\big([v'], [v], [\phi]\big) = {} & \big([\phi \overset{?}{=} 1] \otimes [v' \overset{?}{=} v]\big) \\ & \curlyvee \big([\phi \overset{?}{=} 2] \otimes \Phi[v' \overset{?}{=} v]\big) \\ & \curlyvee \big([\phi \overset{?}{=} 3] \otimes [v' \overset{?}{\leqslant} v]\big) \\ & \curlyvee \big([\phi \overset{?}{=} 4] \otimes \big(\Phi[v' \overset{?}{\leqslant} v] \curlyvee [v' \overset{?}{=} v]\big)\big) \end{aligned}$$

where Φ, $\otimes$, and $\curlyvee$ are the secure negation, multiplication and minimum protocols respectively, and $[\cdot \overset{?}{=} \cdot]$ and $[\cdot \overset{?}{\leqslant} \cdot]$ are respectively the equality and comparison protocols presented in Section 2.2.

It is worth noting that the secure minimum protocol uses homomorphic subtraction and the multiplication protocol to ensure that the outcome is in the range $\{[0], [1]\}$. While this is necessary in the general case to ensure the correct evaluation of policies, in the case of the secure value-matching protocol, these operations have to be executed for each predicate (four times in the value-matching protocol above). This results in an overhead in term of both computation and communication costs. Next, we present an optimized version of the secure matching protocol (Algorithm 2) in which the secure value-matching protocol uses homomorphic addition instead of secure minimum protocol and then the secure comparison protocol is applied to guarantee that the outcome of secure matching protocol is in the range $\{[0], [1]\}$.

Specifically, given an encrypted target $[t] = ([a], [v], [\phi])$ and an encrypted value $[v']$, the secure *value-matching* protocol can be realized as:

$$\begin{aligned} \textit{value-matching}^H\big([v'], [v], [\phi]\big) = {} & [\phi \overset{?}{=} 1] \otimes [v' \overset{?}{=} v] \\ & \oplus [\phi \overset{?}{=} 2] \otimes \big(\Phi[v' \overset{?}{=} v]\big) \end{aligned}$$

Algorithm 2: *matching*^{H}: secure HE-based matching protocol

```
Data:  An encrypted atomic target $[t] = ([a], [v], [\phi])$, query $q$
Result: Matching result $M \in \{[0], [1]\}$
1 $M = [0]$
2 for $a_i \in \mathcal{A}_q$ do
3     for $v_i \in V_{a_i}(q)$ do
4         $M = M \oplus ([a_i \overset{?}{=} a] \otimes \textit{value-matching}^H([v'], [v], [\phi]))$
5     end
6 end
7 return $[1 \overset{?}{\leqslant} M]$
```

$$\oplus\, [\phi \overset{?}{=} 3] \otimes \big[v' \overset{?}{\leqslant} v\big]$$
$$\oplus\, [\phi \overset{?}{=} 4] \otimes \big(\big(\phi\big[v' \overset{?}{\leqslant} v\big]\big) \oplus \big[v' \overset{?}{=} v\big]\big)$$

Then, instead of returning the result of the matching (*i.e.*, M) directly as in Algorithm 1, the secure matching protocol for HE (Algorithm 2) applies the secure comparison protocol (line 7). Intuitively, if at least one of the query's values satisfies the constrain specified in the target, M is greater than 0, and consequently the protocol ($[1 \overset{?}{\leqslant} M]$) returns [1]; otherwise it returns [0].

5.3. *Complexity analysis*

To gain insights on the computational complexity of the proposed framework, we first analyze the complexity of the protocols for privacy-preserving target and policy evaluation presented in Section 4 in terms of the building block protocols, *i.e.*, $\neg$, $\wedge$, $\vee$, $-$, $\times$, $(\cdot \overset{?}{=} \cdot)$, and $(\cdot \overset{?}{\leqslant} \cdot)$. Then, we analyze the realization of these building block protocols in terms of Secure Function Evaluation and Homomorphic Encryption. An experimental evaluation of our framework is presented in Section 7.

5.3.1. *Protocol complexity*

This section presents an analysis of the complexity of the proposed protocols for target and policy evaluation on private inputs in terms of the building block protocols used for their definition. Before presenting such an analysis, we assess the complexity of the secure *membership*, *value-matching* and *matching* protocols, which have been introduced to support target and policy evaluation on private inputs.

Membership protocol: The *membership* protocol employs the $(\cdot \overset{?}{=} \cdot)$, $-$, and $\times$ protocols 1, k, and $k-1$ times (where k denotes the set size), respectively (Section 4.2).

Value-matching protocol: The *value-matching* protocol uses two times the $\neg$ protocol, four times the $\wedge$ protocol, four times $\vee$ protocol, seven times the $(\cdot \overset{?}{=} \cdot)$ protocol, and two times the $(\cdot \overset{?}{\leqslant} \cdot)$ protocol.

Matching protocol: The *matching* protocol (Algorithm 1) employs the *value-matching*, $\wedge$, $\vee$ and $(\cdot \overset{?}{=} \cdot)$ protocols m, m, m, and k times respectively, where k and m denote the number of distinct attributes and attribute values in the given query. Therefore, in terms of building block protocols, the *matching* protocol uses $2m$ times the $\neg$ protocol, $5m$ times the $\wedge$ protocol, $5m$ times $\vee$ protocol, $7m+k$ times the $(\cdot \overset{?}{=} \cdot)$ protocol, and $2m$ times the $(\cdot \overset{?}{\leqslant} \cdot)$ protocol.

Now we have the machinery to study the complexity of our protocols for secure target/policy evaluation. The analysis is performed per case based on the grammar of $\mathcal{P}_{\mathcal{A}}$ presented in Section 2.1.

Atomic target: Based on the encoding presented in Section 4.2, the secure evaluation of atomic targets in protected form $(t^a) = ((a), (v), (\phi))$ against a query q requires the execution of the secure *membership*, *matching*, $\neg$, and $\wedge$ protocols 3, 2, 2, and 2 times, respectively. Considering the application of building block protocols in the *membership* and *matching* protocols, in total the secure evaluation of an atomic target in private form (t^a) against query q uses the $\neg$, $\wedge$, $\vee$, $-$, $\times$, $(\cdot \overset{?}{=} \cdot)$, and $(\cdot \overset{?}{\leqslant} \cdot)$ protocols $4m+2$, $10m+2$, $10m$, $3k$, $3(k-1)$, $14m+2k$, and $4m$ times, respectively. Thus, the complexity of evaluating (t^a) against query q, denoted by $\mathcal{C}_{t^a}$, in terms of the complexity of building block protocols is:

$$\mathcal{C}_{t^a} = (4m+2)\,\mathcal{C}_{\neg} + (10m+2)\,\mathcal{C}_{\wedge} + 10m\,\mathcal{C}_{\vee} + 3k\,\mathcal{C}_{-} + 3(k-1)\,\mathcal{C}_{\times} + (14m+2k)\,\mathcal{C}_{(\cdot \overset{?}{=} \cdot)} + 4m\,\mathcal{C}_{(\cdot \overset{?}{\leqslant} \cdot)}$$

where k and m are the number of distinct attributes and attribute values in q, respectively; $\mathcal{C}_{\neg}, \mathcal{C}_{\wedge}, \mathcal{C}_{\vee}, \mathcal{C}_{-}, \mathcal{C}_{\times}, \mathcal{C}_{(\cdot \overset{?}{=} \cdot)}$ and $\mathcal{C}_{(\cdot \overset{?}{\leqslant} \cdot)}$ denote the complexity of $\neg$, $\wedge$, $\vee$, $-$, $\times$, $(\cdot \overset{?}{=} \cdot)$ and $(\cdot \overset{?}{\leqslant} \cdot)$ protocols, respectively.

Composite target: The evaluation of a composite target in protected form $(t^c) = (op(t_1, \dots, t_n))$ requires the secure evaluation of targets $(t_1), \dots, (t_n)$ and the secure evaluation of the operator (op) used to combine those targets. As targets $(t_1), \dots, (t_n)$ can be in turn composite targets, assessing the actual complexity of the secure evaluation of (t^c) requires unfolding its definition in terms of atomic targets. Assuming that (t^c) is formed by atomic targets $(t_1^a), \dots, (t_n^a)$ and operators $(op_1), \dots, (op_h)$, the complexity of evaluating (t^c) against q is:

$$\mathcal{C}_{t^c} = \mathcal{C}_{t_1^a} + \dots + \mathcal{C}_{t_n^a} + \mathcal{C}_{op_1} + \dots + \mathcal{C}_{op_h}$$

where $\mathcal{C}_{t_i^a}$ denotes the complexity of evaluating atomic target (t_i^a) against q (with $1 \leqslant i \leqslant n$) and $\mathcal{C}_{op_j}$ denote the complexity of applying operator op_j (with $1 \leqslant j \leqslant h$). As the secure evaluation of combining operators requires different building block protocols to be implemented (Table 2), the complexity differs depending on the operators used. In the worst case, a combining operator uses two times the $\neg$ protocol, three times the $\wedge$ protocol, and two times the $\vee$ protocol (*i.e.*, for ∇ and $\triangle$). Accordingly, we can rewrite the complexity of evaluating (t^c) against query q as:

$$\begin{aligned}\mathcal{C}_{t^c} &= n \cdot \mathcal{C}_{t^a} + h \cdot (2\,\mathcal{C}_{\neg} + 3\,\mathcal{C}_{\wedge} + 2\,\mathcal{C}_{\vee}) \\ &= (4mn + 2n + 2h)\,\mathcal{C}_{\neg} + (10mn + 2n + 3h)\,\mathcal{C}_{\wedge} + (10mn + 2h)\,\mathcal{C}_{\vee} \\ &\quad + 3kn\,\mathcal{C}_{-} + 3(k-1)n\,\mathcal{C}_{\times} + (14m+2k)n\,\mathcal{C}_{(\cdot \overset{?}{=} \cdot)} + 4mn\,\mathcal{C}_{(\cdot \overset{?}{\leqslant} \cdot)}\end{aligned}$$

where k and m are respectively the number of distinct attributes and attribute values appeared in q, n is the number of atomic targets forming the composite target t^c and h is the number of operators used to combine those (atomic) targets.

Decision policy: The secure evaluation of a decision policy p_d against a query q requires two executions of the $(\cdot \overset{?}{=} \cdot)$ protocol. Accordingly, the complexity of the secure evaluation of a decision policy p_d is $\mathcal{C}_{p_d} = 2\,\mathcal{C}_{(\cdot \overset{?}{=} \cdot)}$.

Targeted policy: The secure evaluation of a targeted policy $((t), (p))$ against a query q requires evaluating the target (t) two times and the policy (p) as well as executing three times the $\wedge$ protocol and two times the $\vee$ protocol (*cf.* Section 4.3). Accordingly, the complexity of evaluating $((t), (p))$ against query q depends on the complexity of evaluating (t) and (p), which in turns can be composite elements. Formally:

$$\mathcal{C}_{(t,p)} = \mathcal{C}_t + \mathcal{C}_p + 3\,\mathcal{C}_{\wedge} + 2\,\mathcal{C}_{\vee}$$

where $\mathcal{C}_t$ and $\mathcal{C}_p$ are the complexity of evaluating (t) and (p) against q, respectively.

Composite policy: The evaluation of a composite policy in private form $\llparenthesis p^c \rrparenthesis = \llparenthesis op(p_1, \dots, p_l) \rrparenthesis$ requires the secure evaluation of policies $\llparenthesis p_1 \rrparenthesis, \dots, \llparenthesis p_l \rrparenthesis$. Given that the combining operators in composite policies are not considered confidential, they are not protected. The complexity of executing these operators is negligible compared to the complexity of the building blocks used for secure target/policy evaluation. Thus, we omit their complexity in our analysis. Accordingly, the complexity of evaluating $\llparenthesis p_1 \rrparenthesis, \dots, \llparenthesis p_l \rrparenthesis$ against a query q depends on the complexity of evaluating its subpolicies, which in turns can be composite elements. Formally:

$$\mathcal{C}_{p^c} = \mathcal{C}_{p_1} + \dots + \mathcal{C}_{p_l}$$

where $\mathcal{C}_{p_i}$ represents the complexity of evaluating policy $\llparenthesis p_i \rrparenthesis$ against q (with $1 \leqslant i \leqslant l$).

As discussed above, targeted and composite policies are complex policy statements, whose elements can be in turn targeted and composite policies. Therefore, assessing their actual complexity requires unfolding their definition in terms of atomic targets, decision policies and targeted policies. Given an arbitrary policy $\llparenthesis p \rrparenthesis$ consisting of n atomic targets $\llparenthesis t^a \rrparenthesis$ combined with h operators $\llparenthesis op \rrparenthesis$ (with worst case complexity as presented for composite targets), n' decision policies $\llparenthesis p_d \rrparenthesis$ and n'' targeted policy $\llparenthesis (t, p') \rrparenthesis$, the complexity of evaluating $\llparenthesis p \rrparenthesis$ against a query q is:

$$\mathcal{C}_p = n\,\mathcal{C}_{t^a} + n'\,\mathcal{C}_{p_d} + n''\,(3\,\mathcal{C}_\wedge + 2\,\mathcal{C}_\vee) + h\,(2\,\mathcal{C}_\neg + 3\,\mathcal{C}_\wedge + 2\,\mathcal{C}_\vee)$$

where $\mathcal{C}_{t^a}$, $\mathcal{C}_{p_d}$ are the complexity of evaluating $\llparenthesis t^a \rrparenthesis$ and $\llparenthesis p_d \rrparenthesis$, respectively. Note that, in the formula above, the atomic targets and decision policies appearing in targeted policies is accounted for in the number of atomic targets and decision policies.

Table 4 summarizes the complexity of the main protocols in terms of the complexity of building block protocols. It is worth noting that the actual complexity depends on the specific privacy-preserving technique used for the implementation of the protocols. In next section, we discuss the complexity of the realization of the protocols in Secure Functional Evaluation and Homomorphic Encryption.

Table 4

Protocol complexity in terms of the number of building block protocols; k and m are respectively the number of distinct attributes and attribute-values in the query to be evaluated; n, n' and n'' are the number of atomic targets, decision policies, and targeted policies; and h is the number of combining operators occurring in composite targets (the worst case scenario)

	Building block protocols						
	$\mathcal{C}_\neg$	$\mathcal{C}_\wedge$	$\mathcal{C}_\vee$	$\mathcal{C}_-$	$\mathcal{C}_\times$	$\mathcal{C}_{(\cdot \overset{?}{=} \cdot)}$	$\mathcal{C}_{(\cdot \overset{?}{\leqslant} \cdot)}$
Membership	–	–	–	k	$k-1$	1	–
Value-matching	2	4	4	–	–	7	2
Matching (Algorithm 1)	$2m$	$5m$	$5m$	–	–	$7m+k$	$2m$
Atomic target ($\mathcal{C}_{t^a}$)	$4m+2$	$10m+2$	$10m$	$3k$	$3(k-1)$	$14m+2k$	$4m$
Composite target ($\mathcal{C}_{t^c}$)	$(4m+2)n+2h$	$(10m+2)n+3h$	$10mn+2h$	$3kn$	$3(k-1)n$	$(14m+2k)n$	$4mn$
Decision policy ($\mathcal{C}_{p_d}$)	–	–	–	–	–	2	–
Arbitrary policy ($\mathcal{C}_p$)	$(4m+2)n+2h$	$(10m+2)n+3n''+3h$	$10mn+2n''+2h$	$3kn$	$3(k-1)n$	$(14m+2k)n+2n'$	$4mn$

5.3.2. Complexity of the protocol realization in SFE and HE

This section analyzes the complexity of the realization of the proposed protocols in Secure Functional Evaluation and Homomorphic Encryption.

SFE-based protocols: in SFE, the $\neg$, $\wedge$, $\vee$, $-$, $\times$, $(\cdot \overset{?}{=} \cdot)$, and $(\cdot \overset{?}{\leqslant} \cdot)$ protocols are securely implemented using the *inverse*, *AND*, *OR*, *subtraction*, *multiplication*, *equality*, and *comparison* gates, respectively. Previous work [13] shows that the *multiplication*, *equality*, and *comparison* gates are significantly heavier compared to the other gates such as *inverse*, *AND*, and *OR* in terms of computation and communication costs. In this respect, we expect that the complexity of policy evaluation in SFE is mainly affected by the number of attributes and attributes values in the query (*i.e.*, by the query size) as well as by the number of atomic targets occurring in the policy.

HE-based protocols: in HE, only some of the building blocks used for the definition of the protocols in Section 4 can be directly mapped to the HE building blocks. In particular, the $-$, $\times$, $(\cdot \overset{?}{=} \cdot)$, and $(\cdot \overset{?}{\leqslant} \cdot)$ protocols can be implemented using the *subtraction*, *multiplication*, *equality* and *comparison* protocols, respectively. The other building block protocols (*i.e.*, the $\neg$, $\wedge$, and $\vee$ protocols) do not have a direct HE counterpart and their definition might require the use of multiple HE building blocks. In [51], these protocols were defined as follows:

$$[\neg a] = [a \overset{?}{=} 0] \tag{1}$$

$$[a \wedge b] = \big([a \overset{?}{\leqslant} b] \otimes [a]\big) \oplus \big([b \overset{?}{\leqslant} a] \otimes [b]\big) \tag{2}$$

$$[a \vee b] = \big([a \overset{?}{\leqslant} b] \otimes [b]\big) \oplus \big([b \overset{?}{\leqslant} a] \otimes [a]\big) \tag{3}$$

where $\oplus$, $\otimes$, $[\cdot \overset{?}{=} \cdot]$ and $[\cdot \overset{?}{\leqslant} \cdot]$ are the *addition*, *multiplication*, *equality*, and *comparison* protocols, respectively.

Similarly to what observed for the SFE encoding, some HE building block protocols are heavier than others in terms of computation and communication costs [16,41,42]. In particular, the *addition* and *subtraction* protocols are light in additively homomorphic cryptosystems such as the *Paillier* cryptosystem [46], while the *multiplication*, *equality*, and *comparison* protocols are considerably heavy protocols in those cryptosystems. For instance, it has been shown that the computation and communication costs of *multiplication* protocol are 200 times more than those of the *addition* protocol [16].

Based on these observations, we redesigned three building block protocols (*i.e.*, $\neg$, $\wedge$, $\vee$) proposed in [51] by replacing the use of heavy protocols with lighter protocols (*cf.* Section 5.2).[9] Next we present a comparison between the complexity of the $\neg$, $\wedge$, and $\vee$ protocols proposed in [51] (represented by Eq. (1), (2), and (3)) – $\mathcal{C}^H_{\neg}$, $\mathcal{C}^H_{\wedge}$ and $\mathcal{C}^H_{\vee}$ – and the complexity of their optimized versions presented in Section 5.2 – $\mathcal{C}^H_{\neg_{op}}$, $\mathcal{C}^H_{\wedge_{op}}$ and $\mathcal{C}^H_{\vee_{op}}$:

$$\mathcal{C}^H_{\neg} = \mathcal{C}_{[\cdot \overset{?}{=} \cdot]} \quad \xrightarrow{optimization} \quad \mathcal{C}^H_{\neg_{op}} = \mathcal{C}_{\ominus}$$

$$\mathcal{C}^H_{\wedge} = \mathcal{C}_{\oplus} + 2\mathcal{C}_{\otimes} + 2\mathcal{C}_{[\cdot \overset{?}{\leqslant} \cdot]} \quad \xrightarrow{optimization} \quad \mathcal{C}^H_{\wedge_{op}} = \mathcal{C}_{\otimes}$$

[9]Note that the correctness of the new protocols can be proved under the condition that the input data are binary (0 or 1) as discussed in Section 5.2.

$$\mathcal{C}^H_{\vee} = \mathcal{C}_{\oplus} + 2\mathcal{C}_{\otimes} + 2\mathcal{C}_{[\cdot \overset{?}{\leqslant} \cdot]} \xrightarrow{optimization} \mathcal{C}^H_{\vee_{op}} = \mathcal{C}_{\oplus} + \mathcal{C}_{\ominus} + \mathcal{C}_{\otimes}$$

where $\mathcal{C}_{\oplus}$, $\mathcal{C}_{\ominus}$, $\mathcal{C}_{\otimes}$, $\mathcal{C}_{[\cdot \overset{?}{=} \cdot]}$ and $\mathcal{C}_{[\cdot \overset{?}{\leqslant} \cdot]}$ are the complexity of the HE *addition*, *subtraction*, *multiplication*, *equality*, and *comparison* protocols, respectively.

We have also optimized the HE implementation of the *matching* protocol (Algorithm 2) to reduce the instances of the *multiplication* protocol to be executed compared to Algorithm 1. By comparing Algorithm 1 and Algorithm 2, it can be observed that the application of the *multiplication* and *subtraction* protocols has been reduced m times each (recall that m is the number of attribute values in the query), which in addition to the complexity reduction resulting from using the optimized versions of the $\neg$, $\wedge$, and $\vee$ protocols, significantly reduces the complexity of the *matching* protocol.

Table 5 summarizes the complexity of HE-based protocols in both general and optimized forms in terms of the number of HE-based building block protocols.[10] It can be observed that, for instance, in atomic target evaluation the application of the *comparison* (*multiplication*) protocol has been reduced from $22m$ ($20m$) times in general form to $2m$ ($10m$) times in optimized version. The application of other building protocols has also been reduced except from the *subtraction* protocol ($7m$ times). Thus, the implementation of the HE-based atomic target evaluation protocol in the optimized version is significantly more efficient compared to its implementation in the general form.

6. Security analysis

This section provides a security analysis of our framework for privacy-preserving multi-party access control (Section 3.2) and of the underlying secure computation protocols (Section 5).

6.1. Protocol security analysis

The proposed privacy preserving protocols were designed based on Homomorphic Encryption and Secure Function Evaluation. We chose Paillier's cryptosystem for the implementation of HE-based protocols. The security of the Paillier cryptosystem is based on computational difficulty of solving decisional composite residuosity assumption [46]. We employed three HE-based building blocks to shape our complex protocols, namely secure equality [41], secure comparison [42] and secure multiplication protocols [16]. These building blocks have been proven to be secure in the presence of semi-honest adversaries. We refer readers to corresponding papers for the details of the security analysis for each building block.

For policy evaluation in Secure Function Evaluation, we employed the Boolean circuits provided by the ABY framework [22]. The use of these circuits provides information theoretic security against semi-honest adversaries and malicious adversaries in the existence of honest majority. We refer readers to [13,22] for the details of secure implementation and security proofs.

In the design of our protocols, we use a combination of the building blocks that are proven secure. According to the universally composable security theorem of Canetti [7], a protocol that is obtained by arbitrary combination of secure subprotocols guarantees security. Therefore, we can conclude that the protocols presented in Section 5 as well as their combination are secure since they are composed from subprotocols proven to be secure.

[10]The *membership* and *decision policy* which have the same complexity when designed in general and optimized versions have not been reported in this table.

Table 5

The complexity of HE-based protocols in terms of the number of HE building block protocols in generic and optimized form; k and m are respectively the number of distinct attributes and attribute-values appeared in query q; n, n', and n'' are the number of atomic targets, decision policy, and targeted policy, respectively; and h is the number of combining operators appearing in composite targets (the worst case scenario)

	HE-based building block protocols				
	$\mathcal{C}_{\oplus}$	$\mathcal{C}_{\ominus}$	$\mathcal{C}_{\otimes}$	$\mathcal{C}_{[\cdot \overset{?}{=} \cdot]}$	$\mathcal{C}_{[\cdot \overset{?}{\leqslant} \cdot]}$
Negation $\mathcal{C}^H_{\neg}$				1	
Optimized Negation $\mathcal{C}^H_{\neg_{op}}$		1			
Min $\mathcal{C}^H_{\wedge}$	1		2		2
Optimized min $\mathcal{C}^H_{\wedge_{op}}$			1		
Max $\mathcal{C}^H_{\vee}$	1		2		2
Optimized max $\mathcal{C}^H_{\vee_{op}}$	1	1	1		
Value-matching $\mathcal{C}^H_{vmatch}$	8		16	9	18
Optimized Value-matching $\mathcal{C}^H_{vmatch_{op}}$	4	6	8	7	2
Matching $\mathcal{C}^H_{match}$	$10m$		$20m$	$9m + k$	$22m$
Optimized matching $\mathcal{C}^H_{match_{op}}$	$5m$	$7m$	$10m$	$7m + k$	$2m$
Atomic target $\mathcal{C}^H_{t^a}$	$20m + 2$	$3k$	$40m + 3k + 1$	$18m + 2k + 5$	$44m + 4$
Optimized atomic target $\mathcal{C}^H_{{t^a}_{op}}$	$10m$	$14m + 3k + 2$	$20m + 3k - 1$	$14m + 2k + 3$	$4m$
Composite target $\mathcal{C}^H_{t^c}$	$(20m + 2)n + 5h$	$3kn$	$(40m + 3k + 1)n + 10h$	$(18m + 2k + 5)n + 2h$	$(44m + 4)n + 10h$
Optimized composite target $\mathcal{C}^H_{{t^c}_{op}}$	$10mn + 2h$	$(14m + 3k + 2)n + 4h$	$(20m + 3k - 1)n + 5h$	$(14m + 2k + 3)n$	$4mn$
Arbitrary policy $\mathcal{C}^H_{p}$	$(20m + 2)n + 5(n'' + h)$	$3kn$	$(40m + 3k + 1)n + 10(n'' + h)$	$(18m + 2k + 5)n + 2n' + 2h$	$(44m + 4)n + 10(n'' + h)$
Optimized arbitrary policy $\mathcal{C}^H_{p_{op}}$	$10mn + 2(n'' + h)$	$(14m + 3k + 2)n + 2n'' + 4h$	$(20m + 3k - 1)n + 5(n'' + h)$	$(14m + 2k + 3)n$	$4mn$

6.2. Framework security analysis

Our framework is designed to be secure against the semi-honest adversary model. The two parties involved in our setting, *i.e.* the STP and Data Server, are semi-honest non-colluding parties who try to obtain additional information from the execution of the protocols. The data holders provide the protected data properly (*i.e.*, encrypted data for HE-based protocols and secret shared data for SFE-based protocols) to the intended parties.

Given that our protocols based on HE and SFE are secure, in the following we discuss the security of our whole framework with respect to the interactions between participants.

- The access requester can observe whether access is granted or not. From this, she can learn the final decision but not the evaluation of the data holders' policies. It is worth noting that, by testing all possible queries, an attacker can only reconstruct the overall multi-party policy, but she would not be able to reconstruct the individual user policies.
- The STP learns neither the data holders' policies nor the final decision. In the HE setting, the STP generates the public and private keys and sends the public key to the data holders and Data Server. Moreover, this entity interacts with the Data Server to evaluate the policies. As discussed above, the building blocks we use in the design of operators are secure and, thus, the semi-honest STP is not able to learn the data holders' individual policies. Moreover, the STP receives the encrypted final decision, which it decrypts using the private key. However, because of the noise added by the Data Server to the encrypted message (see Section 3.2), the STP is not able to infer the final decision.[11] On the other hand, in the SFE setting, the STP only receives one secret share of the data holders' policies. Therefore, neither the data holders' policies nor the result of policy evaluation are revealed to the STP.
- The Data Server does not learn the data holders' policies. The Data Server receives the policies of each data holder in private form (encrypted in HE-based protocols and secure shared in SFE-based protocols). In the HE setting, since it does not have the private key, it is not able to learn the data holders' policies. On the other hand, in the SFE setting, the Data Server only receives secret shares of the data holders' policies. After policy evaluation, the Data Server learns the final decision, but not each individual policy. The Data Server evaluates the multi-party policy through communication with the STP, which has been proven secure. The final decision, which the Data Server derives with the help of the STP, is not considered as a secret information for the Data Server since this entity is responsible for its enforcement.

Our security model allows entities to collude to obtain information about data holders' policies. We discuss a number of threats scenarios to illustrate the robustness of our framework against colluding entities:

- **Data Server and requester collusion:** in the SFE setting, neither the Data Server nor the requester has access to the other share of the data holders' policies. Thus, even if they collude, they are not able to learn the intermediate decisions and data holders' policies. Similarly, in the HE setting, neither the Data Server nor the requester has the private key. Therefore, they cannot decrypt the messages to obtain the policies of the data holders. Both the Data Server and the requester have access to the final decision; thus, neither of these entities learns new information.

[11]Note that for all HE-based protocols in the two-party setting (including the HE-based building blocks considered in this work), the party that does not have the secret key adds random noise to the encrypted messages before they are sent to the party possessing the private key.

- **STP and requester collusion:** the security in the SFE setting can be demonstrated similarly to the case where the Data Server and requester collude as the STP and requester together only possess one share of the data holders' policies. In the HE setting, the Data Server adds random noise to the encrypted data before transferring them to the STP. Thus, even if the STP and the requester have the private key (which is generated by the STP), they cannot obtain any unintended information. It is worth noting that the collusion of the STP and requester might reveal the final decision to the STP. Yet, from the final decision the STP and requester cannot reconstruct the individual policies of the data holders.
- **Data Server and data holders collusion:** some data holders might collude with the Data Server to learn the policies of the other data holders, where at least two data holders are honest (*cf.* Section 3.3). However, the colluding data holders and the Data Server together only have one share of the data of the non-colluding data holders in the SFE setting and do not have the private key of the other data holders in the HE setting. Thus, they cannot derive any unintended information from intermediary messages. Nonetheless, by colluding, the Data Server and data holders might infer the overall evaluation of the honest data holders' policies (from the final decision known by the Data Server), but they are not able to reconstruct the individual policies of honest parties. This is because at least two data holders are assumed to be honest and, in most cases, knowing the overall evaluation of their policies is not enough to learn the individual policies (see later for a discussion).
- **STP and data holders collusion:** since the Data Server adds random noises to messages in the HE setting and the STP does not possess the share of the data of the honest data holders in the SFE setting, even if the STP and the data holders collude, they cannot understand the inputs of honest data holders.

These scenarios can be generalized, for example, to the case where the requester collude with a subset of data holders (with at least two data holders to be honest) and one of the Data Server and STP. Using the reasoning above, it is easy to see that the only information that can be leaked is the final decision that the STP can learn by colluding with the requester. By knowing the final decision and the multi-party policy, the entities involved (requester, data holders, Data Server, STP) may be able to learn some information about the users' individual policies. For example, if the final decision is the *not-applicable* decision ($\perp$), it is easily inferred that all user policies evaluate to *not-applicable*. However, we argue that these leakages are intrinsic in the definition of the combining operators in Table 1 rather than due to flaws in the design of the secure computation protocols proposed in Section 4 or their combination. Since revealing the final decision is inevitable, we do not consider these leakages as security leakages.

7. Experiments

We have implemented the protocols presented in Section 5 in C++. We used GMP multiprecision library for the implementation of big integer operations. For the HE-based protocols, we used a cryptographic key of length 2048 bits as recommended by the NIST [3]. For the protocols based on the ABY framework, we used 32-bit Boolean circuits.[12]

To assess the practical feasibility of our mechanism for privacy-preserving multi-party access control, we performed three sets of experiments: 1) the first set of experiments is used to evaluate the impact of

[12]The implementation of the proposed protocols in HE and SFE can be found at https://github.com/IschaStork/he-policy-eval and https://github.com/IschaStork/sfe-policy-eval, respectively.

protected atomic target evaluation against the queries of different sizes; 2) the next set of experiments studies the scalability of implementing the combining operators over private inputs; 3) the last set of experiments investigates the impact of protected complex policies' evaluation, where the number of atomic targets forming a complex policy varies. In all experiments, we used computation time (in ms) to measure computation cost and bandwidth usage (in bytes) to measure communication cost.

In a practical scenario, the two parties running the experiments (in both HE and SFE setting) would be remote parties communicating through an Internet connection. For the experiments such a connection was simulated with the use of local sockets. This does not matter for the communication costs as the data that is sent in a local network is equal to the data usage of a remote connection. The experiments were performed on a virtual single machine running Ubuntu 19.04 LTS with 64-bit microprocessor and 5 GB of RAM, with Intel Core i5-7200U CPU, 2.5 GHz.

7.1. Atomic target evaluation

The evaluation of an atomic target has a significant impact on policy evaluation as it serves as a building block for the evaluation of any target (*cf.* Table 2). As shown in Section 5.3, the secure evaluation of atomic targets depends on the query size (*i.e.*, the number of attributes and attribute name-value pairs constituting the query). In the first set of experiments we assess their impact in terms of computation and communication costs by evaluating atomic targets against queries of different size for both HE-based protocols and SFE-based protocols.

Setting. For the experiments, we generated a set of 10 attributes $\mathcal{A} = \{a_1, \ldots, a_{10}\}$, where the domain of each attribute $a_i \in \mathcal{A}$ contains 10 attribute values. Every attribute and attribute value is represented by a unique integer number. To assess the impact of the query size on the computation and communication costs of the evaluation of an atomic target in protected form $\langle t \rangle$, we randomly generated atomic targets and queries of different sizes. An atomic target $t = a\ \phi\ v$ is generated by randomly selecting an attribute $a_i \in \mathcal{A}$, attribute-value $v_j \in \mathcal{V}_{a_i}$, and predicate $\phi \in \{1, 2, 3, 4\}$. A query of size n is generated by creating n attribute name-value pairs, where the first element of each pair is an attribute a_i randomly selected from $\mathcal{A}$ and the second element is a value randomly selected from its domain $\mathcal{V}_{a_i}$. We varied the query size from 1 to 20 and repeated each experiment 50 times.

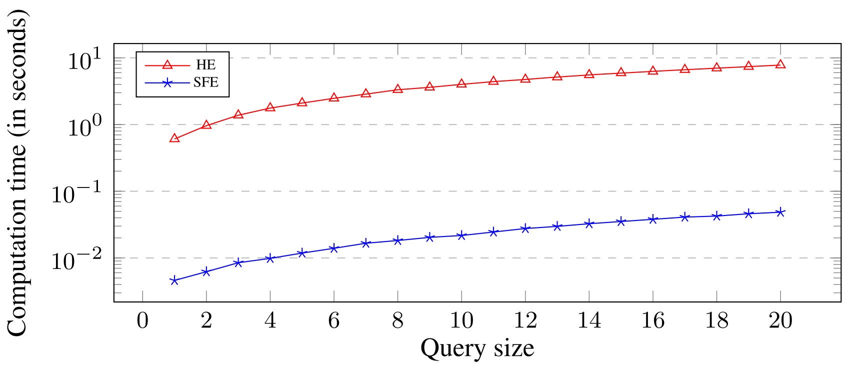

Fig. 2. Computation time (in seconds) for the secure evaluation of protected atomic target against queries of different sizes. Y-axis is in log scale.

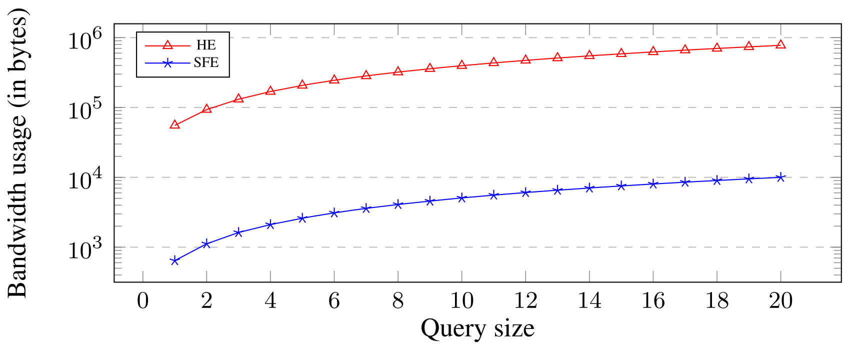

Fig. 3. Bandwidth usage (in bytes) for the secure evaluation of protected atomic target against queries of different sizes. Y-axis is in log scale.

Results. Figures 2 and 3 show respectively the average computation time and bandwidth usage (in log scale) for the secure evaluation of protected atomic target against queries of different sizes (from 1 to 20) over 50 rounds of the experiments. The results show that the SFE-based protocols outperform the HE-based ones in terms of both computation and communication costs. In particular, the evaluation of an atomic target against a query of size 20 required 4.84 ms (9998 bytes) using SFE-based protocols and 779 ms (776171 bytes) using HE-based protocols. Moreover, we can observe that both computation and communication costs of atomic target evaluation are linear in the query size.

7.2. Combining operators

This set of experiments aims to assess the computation and communication costs required by the protocols for the secure computation of the combining operators presented in Table 1. This provides an indication of the resources required by the combining operators in Boolean encoding (Table 3) when implemented using Homomorphic Encryption and Secure Function Evaluation.

Settings. For the experiments, we have created two policies for each operator in Table 1, one for each privacy-preserving approach (HE and SFE). Each policy consists of one or two policy depending on the number of arguments required by the operator. For each operator, we repeated the experiments 50 times.

Results. Table 6 reports the average computation time and bandwidth usage for the protocols implementing the combining operators in Table 1 over 50 runs.

As it can be observed, the SFE-based protocols outperform the HE-based protocols both in terms of computation and communication costs. On average, the computation and communication costs of HE-based protocols are approximately 8000 and 32 times worse than SFE-based ones, respectively. It is worth noting that binary operators exhibit a similar computation and communication costs in binary operators (for both HE and SFE protocols). This is because these protocols comprise an equal number of $\wedge$, $\vee$, and $\neg$ operators (*cf.* Table 3). On the other hand, the negation protocol is slightly cheaper in Homomorphic Encryption as it only requires swapping values rather than operating on logic formulas.

7.3. Composite policies

To gain more insights on the feasibility of our mechanism for privacy preserving multi-party access control, we performed experiments to assess the scalability of policy evaluation on private input in more complex scenarios.

Table 6

Computation time (in ms) and communication bandwidth (in bytes) for single operator

	Time (ms)		Bandwidth (bytes)	
	HE	*SFE*	*HE*	*SFE*
$\neg$	0	0.09	0	51
$\sim$	213	0.09	768	65
$\sqcup$	880	0.1008	3842	121
$\sqcap$	873	0.1032	3841	121
$\tilde{\sqcup}$	873	0.1159	3842	121
$\tilde{\sqcap}$	882	0.1068	3842	121
$\triangle$	882	0.0929	3841	121
$\triangledown$	873	0.1072	3842	121
$\triangleright$	827	0.1041	3842	121

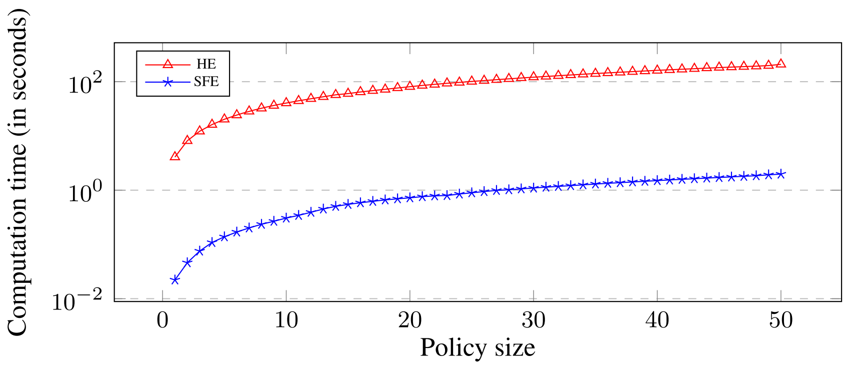

Fig. 4. Computation time (in seconds) for the secure evaluation of protected composite policies against queries of size 10. Y-axis is in log scale.

Settings. For this set of experiments, we generate random policies by varying the policy size. As target evaluation is more computationally and communicationally expensive compared to the evaluation of combining operators,[13] we regard the size of a policy as the number of atomic targets constituting the policy. We varied the policy size from 1 to 50, while we fixed the query size to 10.

Results. Figures 4 and 5 report the average computation time and bandwidth usage (in log scale) for each policy size. This result confirms that SFE-based protocols outperform HE-based protocols in terms of both time and bandwidth usage. On average, the computation time required for the evaluation of policies of size 1 to 10 in HE is 13 times worse than the computation time required by SFE-protocols. This ratio increases to 105 when we compare the time required for evaluating the policy of size 50 in HE and SFE. Moreover, the bandwidth usage for the evaluation of policy of size 50 is 86 times worse when implemented in HE compared to SFE.

This difference between HE-based and SFE-based protocols arises for two reasons. The first reason is that HE-based protocols, beside needing to perform encryption and decryption at the end of each communication, also require implementing heavier multiplication operation for the implementation of $\wedge$ and $\vee$ operators. These operations require modular exponentiations, which is computationally and

[13]The previous experiments show that, on average, the computation and communication costs of atomic target evaluation (against queries of size 10) are 22 and 72 times worse than the evaluation of heaviest combining operator implementation.

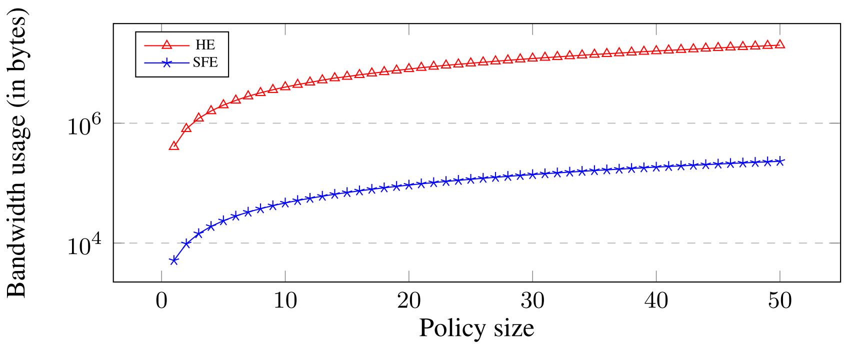

Fig. 5. Bandwidth usage (in bytes) for the secure evaluation of protected composite policies against queries of size 10. Y-axis is in log scale.

communicationally heavy. On the other hand, operations in SFE are simpler, encompassing low-level circuit operations such as such as *AND*, *OR* and *inverse* check. The second reason is related to the size of messages used in HE and SFE. In HE, all operations are performed on 2048 bits ciphertexts. On the other hand, in SFE, a much smaller bit message size is used.

From these experiments, we conclude that the implementation of policy evaluation protocols using Homomorphic Encryption has some limitations in terms of computation and communication costs. On the other hand, Secure Functional Evaluation provides a viable solution for the realization of the proposed protocols for privacy-preserving policy evaluation in multi-party access control.

8. Related work

In recent years, multi-party access control has received increasing attention [45]. This interest has resulted in several access control solutions for the protection of jointly-owned and jointly-managed resources, either through co-owners' negotiation [18] or automatic built-in interface [27]. Approaches for collaborative decision making and conflict resolution have been largely investigated through the application of game theory [49], computational mechanisms [52], social-friend circle model [59], and veto voting [53]. These approaches, however, do not consider users' policies as sensitive information by themselves.

The confidentiality of policies has typically been addressed in trust negotiation [34,60]. The aim of trust negotiation is to establish mutual trust between parties that do not have a pre-existing relationship through an exchange of (extensional) policies, represented by digital credentials. Disclosure of credentials, in turn, is regulated by policies specifying which credentials must be received before the requested credentials can be disclosed. Similarly, Trivellato et al. [54] propose a goal evaluation algorithm for trust management that detects termination in a completely distributed way without the need of disclosing the policies of parties, thereby preserving their confidentiality. In this work, we pursue a different direction and assume that parties disclose their policies in private form. These policies are then evaluated in privacy-preserving way using secure computation protocols.

A large body of research has investigated the use of cryptographic techniques for the enforcement of access control policy. Identity-based Encryption (IBE) reconsiders the concept of public-key cryptography, in which a message is encrypted for a specific receiver using its public-key, by allowing the public-key to be an arbitrary string [50], *e.g.*, the receiver's email address. Attribute-Based Encryption (ABE) go one step further by modeling an identity as a set of attributes [25], thus supporting the enforcement of ABAC policies. ABE schemes can be classified into two main classes: Key-Policy ABE (KP-ABE) [2], in which the access control policy is encoded into the user's private-key and the ciphertext is computed with respect to a set of attributes, and Ciphertext-Policy ABE (CP-ABE) [6], in which the user's private-key is associated with a set of attributes and the access control policy is specified in the ciphertext. Our solution is closer to CP-ABE where the access control policy is closer to the resource to be protected. While CP-ABE schemes originally focus on the protection of the plaintext (*i.e.*, ciphertexts reveal no information about the underlying plaintext) and rely on a single authority, a number of schemes have been proposed to extend CP-ABE to protect policy confidentiality and to deal with collaborative systems. Policy confidentiality can be protected using anonymous ABE schemes [31,43,62] where the access control policy is hidden and the decryptor cannot obtain more information about the policy associated with the encrypted data besides the fact that it can decrypt the data. On the other hand, multi-authority ABE schemes [8,9] have emerged to account for that more than one entity could be responsible for managing the attributes (*e.g.*, one managing driver licenses and one managing voter registration). However, these schemes target a different type of collaborative systems, in which resources are governed by a single entity and delegation is used to enable decentralized authorization across administrative domains; contrarily, our solution focuses on the evaluation of multi-party access control policies which integrate possibly conflicting access constraints from multiple parties. To the best of our knowledge, no CP-ABE schemes support the collaborative specification of access control policies for co-owned resources. Moreover, CP-ABE and, in general, ABE schemes have restrictions in the types of ABAC policies that can be enforced (*i.e.*, only equality constraints are supported), while our solution supports the specification of a larger range of access constraints (*cf.* Section 4) and, given its compositional nature, can be extended to support additional types of access constraints.

An application of cryptographic-based techniques to multi-party access control can be found in [26], which proposes a collaborative access control model based on threshold-based secret sharing. Data holders upload their co-owned resources encrypted and disclose secret shares of the decryption key to *trusted* friends, who are responsible to partially enforce the collective policy. A user can only decrypt a resource if she collects 'enough' shares of the key. This work, however, mainly focuses on the protection of resources, whereas the confidentiality of users' policies is not addressed. Moreover, compared to our work that supports the definition of arbitrary strategies to combine users' policies, this approach only allows a simple strategy based on the number of shares of the decryption key that the user should have.

For secure policy evaluation, we rely on previous work on secure multiparty computation. The basic concepts for secure multiparty computation were first introduced by Yao [57]. Since, several approaches for the secure evaluation of a function have been developed, *e.g.* based on combinatorial circuits [22] or one-dimensional look up tables [40]. These approaches tend to be impractical due to the high computation/communication costs. Homomorphic Encryption and Secure Functional Evaluation have shown a promising result in terms of communication and computation costs in the context of multi-party access control [51]. However, differently from [51], the secure computation protocols proposed in this work allow the evaluation of policies expressed in ABAC and have been implemented using a Boolean encoding of the decision space, which provides a more efficient implementation compared to the use of a three-valued encoding.

There are a few attempts in the literature implementing logical operations over protected inputs, *e.g.* privacy-preserving min computation in mobile sensing data [61] or privacy-preserving Max/Min query protocol for two-tiered sensor network [58]. However, this body of work fail to address the secure implementation of the complete set of operations required in this study (Tables 2 and 3).

9. Conclusions and future work

In this work, we proposed a framework for the secure evaluation of multi-party access control policies expressed in ABAC, which preserves the confidentiality of individual user policies forming the multi-party policy. In particular, we designed secure computation protocols for the evaluation of multi-party policies based on a Boolean encoding of the decision space and able to account for the evaluation of complex targets. We realized such protocols using two privacy-preserving approaches, namely Homomorphic Encryption and Secure Functional Evaluation. An experimental evaluation of the proposed protocols shows that the SFE-based protocols outperform the HE-based ones in terms of both computation time and bandwidth usage and provide an effective foundation for the realization of privacy-preserving mechanisms for multi-party access control.

As future work, we aim to extend our approach to support the evaluation of multi-party policies in which the operators used for combining atomic targets and policies are also protected. This should minimize the risks that an attacker can learn some information on the underlying user policies.

Acknowledgments

This work is supported by the H2020-ECSEL programme of the European Commission through the SECREDAS project (grant no. 783119).

References

[1] J. Alwen, A. Shelat and I. Visconti, Collusion-free protocols in the mediated model, in: *Advances in Cryptology*, Springer, 2008, pp. 497–514.
[2] N. Attrapadung, B. Libert and E. de Panafieu, Expressive key-policy attribute-based encryption with constant-size ciphertexts, in: *International Conference on Practice and Theory in Public Key Cryptography*, 2011, pp. 90–108.
[3] E.B. Barker, L. Chen, A.R. Regenscheid and M.E. Smid, SP 800-56B. Recommendation for pair-wise key establishment schemes using integer factorization cryptography, Technical report, Gaithersburg, MD, United States, 2009.
[4] D. Beaver, Efficient multiparty protocols using circuit randomization, in: *Advances in Cryptology*, 1991, pp. 420–432.
[5] C. Bertolissi, J. den Hartog and N. Zannone, Using provenance for secure data fusion in cooperative systems, in: *Proceedings of Symposium on Access Control Models and Technologies*, ACM, 2019, pp. 185–194.
[6] J. Bethencourt, A. Sahai and B. Waters, Ciphertext-policy attribute-based encryption, in: *Proceedings of IEEE Symposium on Security and Privacy*, IEEE, 2007, pp. 321–334.
[7] R. Canetti, Universally composable security: A new paradigm for cryptographic protocols, in: *Proceedings of Symposium on Foundations of Computer Science*, IEEE, 2001, pp. 136–145. doi:10.1109/SFCS.2001.959888.
[8] M. Chase, Multi-authority attribute based encryption, in: *Theory of Cryptography*, Springer, 2007, pp. 515–534. doi:10.1007/978-3-540-70936-7_28.
[9] M. Chase and S.S.M. Chow, Improving privacy and security in multi-authority attribute-based encryption, in: *Proceedings of ACM Conference on Computer and Communications Security*, ACM, 2009, pp. 121–130.
[10] J. Crampton and C. Morisset, PTaCL: A language for attribute-based access control in open systems, in: *Principles of Security and Trust*, Springer, 2012, pp. 390–409.
[11] J. Cui, H. Zhong, X. Tang and J. Zhang, A fined-grained privacy-preserving access control protocol in wireless sensor networks, in: *International Conference on Utility and Cloud Computing*, ACM, 2016, pp. 382–387. doi:10.1145/2996890.3007850.
[12] S. Damen, J. den Hartog and N. Zannone, CollAC: Collaborative access control, in: *Proceedings of International Conference on Collaboration Technologies and Systems*, IEEE, 2014, pp. 142–149.
[13] D. Demmler, T. Schneider and M. Zohner, ABY – A framework for efficient mixed-protocol secure two-party computation, in: *Proceedings of Annual Network and Distributed System Security Symposium*, 2015.
[14] J. den Hartog and N. Zannone, Collaborative access decisions: Why has my decision not been enforced? in: *Information Systems Security*, LNCS, Vol. 10063, Springer, 2016, pp. 109–130. doi:10.1007/978-3-319-49806-5_6.

[15] J. den Hartog and N. Zannone, A policy framework for data fusion and derived data control, in: *Proceedings of ACM International Workshop on Attribute Based Access Control*, ACM, 2016, pp. 47–57.

[16] Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, I. Lagendijk and T. Toft, Privacy-preserving face recognition, in: *Proceedings of International Symposium on Privacy Enhancing Technologies*, Springer, 2009, pp. 235–253. doi:10.1007/978-3-642-03168-7_14.

[17] D. Evans, V. Kolesnikov and M. Rosulek, A pragmatic introduction to secure multi-party computation, *Found. Trends Priv. Secur.* **2**(2–3) (2018), 70–246. doi:10.1561/3300000019.

[18] R.L. Fogues, P.K. Murukannaiah, J.M. Such and M.P. Singh, Sharing policies in multiuser privacy scenarios: Incorporating context, preferences, and arguments in decision making, *ACM Trans. Comput.-Hum. Interact.* **24**(1) (2017), 5–1529. doi:10.1145/3038920.

[19] M.J. Freedman, K. Nissim and B. Pinkas, Efficient private matching and set intersection, in: *Advances in Cryptology*, Springer, 2004, pp. 1–19.

[20] M. Goldberg, How understanding relationships drives better data and analytics, 2016.

[21] O. Goldreich, *The Foundations of Cryptography – Volume 2, Basic Applications*, Cambridge University Press, 2004.

[22] O. Goldreich, S. Micali and A. Wigderson, How to play any mental game or a completeness theorem for protocols with honest majority, in: *Proceedings of Annual Symposium on Theory of Computing*, ACM, 1987, pp. 218–229.

[23] C. Hazay and M. Venkitasubramaniam, On the power of secure two-party computation, *J. Cryptol.* **33**(1) (2020), 271–318. doi:10.1007/s00145-019-09314-2.

[24] D. He, J. Bu, S. Zhu, M. Yin, Y. Gao, H. Wang, S. Chan and C. Chen, Distributed privacy-preserving access control in a single-owner multi-user sensor network, in: *Proceedings of INFOCOM*, IEEE, 2011, pp. 331–335.

[25] D. Huang, Q. Dong and Y. Zhu, *Attribute-Based Encryption and Access Control*, CRC Press, 2020.

[26] P. Ilia, B. Carminati, E. Ferrari, P. Fragopoulou and S. Ioannidis, SAMPAC: Socially-aware collaborative multi-party access control, in: *Proceedings of Conference on Data and Application Security and Privacy*, ACM, 2017, pp. 71–82.

[27] P. Ilia, I. Polakis, E. Athanasopoulos, F. Maggi and S. Ioannidis, Face/off: Preventing privacy leakage from photos in social networks, in: *Proceedings of Conference on Computer and Communications Security*, ACM, 2015, pp. 781–792.

[28] Y. Ishai, E. Kushilevitz, M. Prabhakaran, A. Sahai and C.-H. Yu, Secure protocol transformations, in: *Advances in Cryptology*, Springer, 2016, pp. 430–458.

[29] W.H. Jobe, Functional completeness and canonical forms in many-valued logics, *The Journal of Symbolic Logic* **27**(4) (1962), 409–422. doi:10.2307/2964548.

[30] S. Kamara, P. Mohassel and M. Raykova, Outsourcing multi-party computation, *IACR Cryptol. ePrint Arch.* **2011** (2011), 272.

[31] A. Kapadia, P.P. Tsang and S.W. Smith, Attribute-based publishing with hidden credentials and hidden policies, in: *Proceedings of Annual Network and Distributed System Security Symposium*, 2007, pp. 179–192.

[32] M. Lepinksi, S. Micali and A. Shelat, Collusion-free protocols, in: *Proceedings of Annual Symposium on Theory of Computing*, ACM, 2005, pp. 543–552.

[33] J. Li, N. Li and B. Ribeiro, Membership inference attacks and defenses in classification models, 2020.

[34] N. Li and W. Winsborough, Towards practical automated trust negotiation, in: *Proceedings of International Workshop on Policies for Distributed Systems and Networks*, IEEE, 2002, pp. 92–103.

[35] R. Mahmudlu, J. den Hartog and N. Zannone, Data governance and transparency for collaborative systems, in: *Data and Applications Security and Privacy*, LNCS, Vol. 9766, Springer, 2016, pp. 199–216.

[36] P. Mohassel and Y. Zhang, SecureML: A system for scalable privacy-preserving machine learning, in: *Proceedings of IEEE Symposium on Security and Privacy*, IEEE, 2017, pp. 19–38.

[37] C. Morisset, T.A.C. Willemse and N. Zannone, Efficient extended ABAC evaluation, in: *Proceedings of Symposium on Access Control Models and Technologies*, ACM, 2018, pp. 149–160.

[38] C. Morisset and N. Zannone, Reduction of access control decisions, in: *Proceedings of Symposium on Access Control Models and Technologies*, ACM, 2014, pp. 53–62. doi:10.1145/2613087.2613106.

[39] M. Naehrig, K. Lauter and V. Vaikuntanathan, Can homomorphic encryption be practical? in: *Proceedings of Cloud Computing Security Workshop*, ACM, 2011, pp. 113–124.

[40] M. Naor and K. Nissim, Communication complexity and secure function evaluation, *CoRR* (2001). arXiv:cs/0109011.

[41] M. Nateghizad, Z. Erkin and R.L. Lagendijk, Efficient and secure equality tests, in: *Proceedings of International Workshop on Information Forensics and Security*, 2016, pp. 1–6.

[42] M. Nateghizad, Z. Erkin and R.L. Lagendijk, An efficient privacy-preserving comparison protocol in smart metering systems, *EURASIP Journal on Information Security* **2016**(1) (2016), 11. doi:10.1186/s13635-016-0033-4.

[43] T. Nishide, K. Yoneyama and K. Ohta, Attribute-based encryption with partially hidden encryptor-specified access structures, in: *Proceedings of International Conference on Applied Cryptography and Network Security*, Springer, 2008, pp. 111–129. doi:10.1007/978-3-540-68914-0_7.

[44] OASIS, eXtensible Access Control Markup Language (XACML) Version 3.0, OASIS Standard, 2013.

[45] F. Paci, A.C. Squicciarini and N. Zannone, Survey on access control for community-centered collaborative systems, *ACM Comput. Surv.* **51**(1) (2018), 6–1638.
[46] P. Paillier, Public-key cryptosystems based on composite degree residuosity classes, in: *Proceedings of International Conference on Theory and Application of Cryptographic Techniques*, Springer, 1999, pp. 223–238.
[47] B. Pinkas, T. Schneider, N.P. Smart and S.C. Williams, Secure two-party computation is practical, in: *Advances in Cryptology*, LNCS, Vol. 5912, Springer-Verlag, 2009, pp. 250–267.
[48] S. Rajtmajer, A. Squicciarini, C. Griffin, S. Karumanchi and A. Tyagi, Constrained social-energy minimization for multi-party sharing in online social networks, in: *Proceedings of International Conference on Autonomous Agents & Multiagent Systems*, International Foundation for Autonomous Agents and Multiagent Systems, 2016, pp. 680–688.
[49] S. Rajtmajer, A. Squicciarini, C. Griffin, S. Karumanchi and A. Tyagi, Constrained social-energy minimization for multi-party sharing in online social networks, in: *Proceedings of International Conference on Autonomous Agents & Multiagent Systems*, International Foundation for Autonomous Agents and Multiagent Systems, 2016, pp. 680–688.
[50] A. Shamir, Identity-based cryptosystems and signature schemes, in: *Advances in Cryptology*, LNCS, Vol. 196, Springer, 1985, pp. 47–53. doi:10.1007/3-540-39568-7_5.
[51] M. Sheikhalishahi, G. Tillem, Z. Erkin and N. Zannone, Privacy-preserving multi-party access control, in: *Proceedings of International Workshop on Privacy in the Electronic Society*, ACM, 2019.
[52] J.M. Such and N. Criado, Resolving multi-party privacy conflicts in social media, *IEEE Transactions on Knowledge and Data Engineering* **28**(7) (2016), 1851–1863. doi:10.1109/TKDE.2016.2539165.
[53] K. Thomas, C. Grier and D.M. Nicol, UnFriendly: Multi-party privacy risks in social networks, in: *Privacy Enhancing Technologies*, LNCS, Vol. 6205, Springer, 2010, pp. 236–252. doi:10.1007/978-3-642-14527-8_14.
[54] D. Trivellato, N. Zannone and S. Etalle, GEM: A distributed goal evaluation algorithm for trust management, *Theory and Practice of Logic Programming* **14**(3) (2014), 293–337. doi:10.1017/S1471068412000397.
[55] J.R. Troncoso-Pastoriza, S. Katzenbeisser, M.U. Celik and A.N. Lemma, A secure multidimensional point inclusion protocol, in: *Proceedings of Workshop on Multimedia & Security*, 2007.
[56] F. Turkmen, J. den Hartog, S. Ranise and N. Zannone, Formal analysis of XACML policies using SMT, *Computers & Security* **66** (2017), 185–203.
[57] A.C. Yao, Protocols for secure computations, in: *Proceedings of Annual Symposium on Foundations of Computer Science*, IEEE, 1982, pp. 160–164.
[58] Y. Yao, N. Xiong, J.H. Park, L. Ma and J. Liu, Privacy-preserving max/min query in two-tiered wireless sensor networks, *Computers & Mathematics with Applications* **65**(9) (2013), 1318–1325. doi:10.1016/j.camwa.2012.02.003.
[59] L. Yu, S.M. Motipalli, D. Lee, P. Liu, H. Xu, Q. Liu, J. Tan and B. Luo, My friend leaks my privacy: Modeling and analyzing privacy in social networks, in: *Proceedings of Symposium on Access Control Models and Technologies*, ACM, 2018, pp. 93–104.
[60] T. Yu, M. Winslett and K.E. Seamons, Supporting structured credentials and sensitive policies through interoperable strategies for automated trust negotiation, *ACM Trans. Inf. Syst. Secur.* **6**(1) (2003), 1–42. doi:10.1145/605434.605435.
[61] Y. Zhang, Q. Chen and S. Zhong, Efficient and privacy-preserving min and *k*th min computations in mobile sensing systems, *IEEE Transactions on Dependable and Secure Computing* **14**(1) (2017), 9–21.
[62] Y. Zhang, X. Chen, J. Li, D.S. Wong and H. Li, Anonymous attribute-based encryption supporting efficient decryption test, in: *Proceedings of ACM SIGSAC Symposium on Information, Computer and Communications Security*, ACM, 2013, pp. 511–516.